GPU-ACCELERATED MODELING OF MICROSCALE ATMOSPHERIC FLOWS OVER COMPLEX TERRAIN

by

Anthony Rey DeLeon

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mechanical Engineering

Boise State University

August 2012

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Anthony Rey DeLeon

Thesis Title: GPU-accelerated Modeling of Microscale Atmospheric Flows over Complex Terrain

Date of Final Oral Examination: 13 June 2012

The following individuals read and discussed the thesis submitted by student Anthony Rey DeLeon, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Inanc Senocak, Ph.D.	Chair, Supervisory Committee
Ralph Budwig, Ph.D.	Member, Supervisory Committee
Paul Dawson, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Inanc Senocak, Ph.D., Chair, Supervisory Committee. The thesis was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

dedicated to my family

ACKNOWLEDGMENTS

I would like to express my appreciation to everyone who has helped me throughout my studies here at Boise State University. This thesis would not have been completed without their help. In particular, I thank Dr. Inanc Senocak for presenting me with the opportunity to work on the research discussed in this thesis and for guiding me throughout the entire process. The experience has been very beneficial to me, having been exposed to several new areas of study and learning numerous valuable skills that will most certainly help me throughout my career. I also want to thank Dr. Ralph Budwig and Dr. Paul Dawson for being members on my thesis committee and all the helpful feedback they provided.

I would also like to thank Kyle Felzien and Marianna Budnikova, at the time computer science undergraduate students, for their efforts in producing pre-processing software for this research, and Marty Lukes for his efforts in maintaining our GPU computing infrastructure here at Boise State. My thanks also go to Ken Blair, for maintaining Boise State's supercomputing cluster located at Idaho National Laboratory.

Being funded by a NASA EPSCoR fellowship, I would also like to thank the Idaho Space Grant Consortium (ISGC) for awarding me this fellowship. This work was also partially funded by the National Science Foundation (Award # 1043107).

ABSTRACT

With installed wind power capacities steadily on the rise, balancing the loads on electrical grids is challenging due to the intermittency of the wind. Short-term wind power forecasting can be a valuable tool for better informing grid operators on the available wind power. Current short-term wind forecasting techniques typically adopt mesoscale weather forecasting models with spatial resolutions on the order of a kilometer. On relatively flat terrain, use of mesoscale models may prove effective, but application to complex terrain induces large forecasting errors. To address this issue, a baseline incompressible flow solver for GPU (graphics processing unit) clusters is extended to simulate neutrally-stable atmospheric flows over complex terrain with the ultimate goal of developing a comprehensive short-term wind forecasting capability that can resolve winds at turbine hub height. In the extended wind model, the large-eddy simulation (LES) technique with a Lagrangian dynamic subgrid-scale (SGS) model is implemented to better capture the effects of atmospheric turbulence over complex terrain. Additionally, the immersed boundary method (IBM) is adopted to numerically represent the complex terrain on a Cartesian mesh. Validation is performed using common benchmark cases. Performance results obtained from simulating the Bolund Hill Experiment demonstrates that faster than real-time computations are realized with GPU clusters. While the results are encouraging and justifies the foundation for a short-term wind forecasting capability, the work does not account for all factors in wind forecasting and the results can be considered as a first attempt requiring further improvements.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

GPU – Graphics Processing Unit

CFD – Computational Fluid Dynamics

ABL – Atmospheric Boundary Layer

CUDA – Compute Unified Device Architecture

MPI – Message Passing Interface

LES – Large-eddy Simulation

SGS – Subgrid Scale

IBM – Immersed Boundary Method

NWP – Numerical Weather Prediction

ARMA – Autoregressive Moving Average

CHAPTER 1

INTRODUCTION

The integration of more renewable energy resources into our electrical power grids is driven by several factors including: energy security and stability; environmental and climate changes concerns; and economics. Wind energy has the potential to become a larger energy resource in the United States, however producing electricity from wind is far more complicated than just installing more wind turbines. Grid integration is a major challenge, the focus being how to balance the load on the grid given the highly variable nature of the wind. Economics is always a concern with the cost of additional transmission lines, wind turbine manufacturing, and the maintenance of wind farms. The uncertainty of forecasting wind power generation for short periods of time (anywhere from an hour to a few days) is a challenge as well, since the results from existing short-term wind forecasting capabilities vary greatly depending on the location and time period investigated [10, 60, 82]. These and other challenges are further described in a Department of Energy (DOE) report, 20% Wind by 2030 [93].

There is always uncertainty in wind predictions that is escalated when predicting wind power generation, because power is directly proportional to wind velocity cubed. This means that uncertainty in power is three times more than uncertainty in wind velocity. A small confidence interval causes utility companies to conservatively balance their reserves to compensate for the possibility of power underprediction.

Overprediction of wind power generation forces utility companies to curtail power generation causing monetary losses for the wind fleet [11, 81]. In general, uncertainty in wind speed and wind power generation causes economic losses for both utility companies and the wind fleet. Reducing the uncertainty would be beneficial to both parties.

Improvements in wind forecasting is the subject of several research projects in recent years [10, 60, 82]. Numerical weather prediction (NWP) models solve the complex mathematical models for wind velocity, temperature, pressure, and moisture using mesoscale initial conditions provided by weather services to estimate the wind conditions at wind farm locations [82]. The mesoscale is on the order of 1 km to 100 km horizontal spatial resolutions. The microscale, the scale applicable to wind turbines, is less than 2 km [77]. The challenge is then to reduce uncertainty in transforming mesoscale information to the microscale.

Several techniques have been proposed to improve wind forecasting. Ensemble forecasting is an approach that runs various NWP simulations to obtain a frequency distribution for weather events [83]. Ensemble forecasting is advantageous to use because the chaotic nature of the weather causes slight variations in initial conditions to be amplified. However, it requires extensive computational resources. Statistical methods based upon autoregressive moving averages (ARMA) are another popular approach to predict wind. ARMA relies on historical data to make predictions [10, 60, 82]. In general, the method performs very well over very short time horizons and only require historical wind data. However, accuracy degrades as the time horizon is extended and statistical methods cannot provide the wind flow details that NWP can. Artificial neural networks (ANN) have also been applied to develop relationships between the variables in statistical weather prediction approaches [10, 27]. While capable of producing better results than purely statistical approaches, ANN still

suffers from the same disadvantages. Even though improvements have been made over the years, no approach can be considered universally applicable to all wind farm locations and a significant uncertainty still exists that would be beneficial to reduce.

1.1 Thesis Statement

The focus of this thesis is to provide a foundation for a short-term wind forecasting capability that involves modeling the atmospheric boundary layer (ABL) over complex terrain at the microscale. A comprehensive microscale wind forecasting model has to consider atmospheric stability while taking into account the effect of surface roughness and fluxes of heat and moisture. Therefore, turbulence modeling and imposing complex terrain boundary conditions are of the utmost importance. For the core components, large-eddy simulation (LES) will be hybridized with Reynolds-averaged Navier-Stokes (RANS) for turbulence modeling. The immersed boundary method (IBM) is then adopted to impose the complex terrain boundary conditions.

The fundamental idea behind LES is to separate the flow field into large- and small-scales using a mathematical filter [48, 75]. Large-scales are resolved while the small-scales are treated as statistically universal and their effect on the resolved flow is modeled. Small-scales are often referred to as subgrid-scales (SGS) because their length scales are smaller than the numerical grid. RANS is very similar to LES from a mathematical perspective, however the statistical interpretation of the results are very different [94]. LES provides filtered quantities with random components while RANS time-averages the governing Navier-Stokes equations, which only provides mean quantities of the turbulent flow. Most flow structures cannot be resolved with RANS and their effect is accounted for using a turbulence model in conjunction with the time-averaged equations. While LES provides more detail than RANS, LES requires resolving near-wall boundary layers, which requires significant computational

resources and turn around times, particularly at high Reynolds numbers [70]. RANS does not need as much resolution as LES and is more widely adopted for industrial applications.

For wind forecasting and other atmospheric studies, LES is desirable for the detail it provides with highly separated flows but today's computational resources today do not allow for fully-resolved LES at high Reynolds numbers. Also, SGS models in LES do not take into account surface roughness or fluxes of heat and moisture at the surface. Using RANS greatly reduces the amount of computational grid points needed in the near-wall region and can act as a sort of wall model for the LES [75]. RANS can also provide a shear stress near the surface. Correct specification of stresses at the surface is critical because any misprediction can lead to erroneous results in the domain. Surface roughness and fluxes of heat and moisture can also be taken into account with RANS turbulence models [4]. One technique that is practical to implement is to hybridize the RANS and LES techniques where RANS acts as a sort of wall model for LES. However, the differences in the scales of LES and RANS causes a challenge when hybridizing the two approaches but several methods have been investigated [70]. Since micro-scale ABL flows are highly turbulent, a hybrid RANS/LES approach will be implemented similar to Senocak et al. [78].

A direct forcing IBM will be used to impose the boundary condition at the complex terrain without having to generate a mesh that conforms to the terrain. Surface-conforming meshes are tedious to generate and skewed cells can introduce errors into the simulation. The IBM is a numerical technique where boundary conditions created by complex objects immersed in a flow are imposed on a Cartesian mesh by adding a forcing term to the momentum equations [56]. The advantage of this technique is avoiding the cumbersome task of generating body-conforming grids and avoiding possible sources of error from skewed cells resulting from conforming a mesh

to complex terrain. Therefore, this thesis will extend an IBM described in Gilmanov et al. [25] that uses a stereolithography (STL) file of an arbitrarily complex object to micro-scale ABL flows over complex terrain using techniques proposed by Senocak et al. [79].

After implementing the hybrid RANS/LES and IBM, the wind simulation will be tested on Bolund Hill. Bolund is a small, isolated hill located off the coast of Denmark. It is 12 m high and is almost completely surrounded by water. Recently, Bolund Hill has been the subject of several numerical and experimental studies, and it constitutes a good test case to evaluate the accuracy and performance of wind models.

The forecasting time horizon for this particular simulation is short-term and requires use of high-performance supercomputing technologies. In recent years, the graphics processing unit (GPU) has become the new paradigm in high performance parallel computing for the tremendous speedups it provides to most numerical simulations, and the cost-efficiency of these performance gains. GPUs are used in a variety of fields including computational fluid dynamics (CFD), medical imaging, and molecular dynamics [65, 66], to name a few.

GPUs provide the potential of greatly reducing the turn around time of climate and meteorological models, which could greatly improve the speed of the weather forecasting capability that we have today. GPU computing can also provide the necessary performance gains to broaden the adoption of more intensive flow solver techniques and weather forecasting methods, such as ensemble forecasting [27, 45], which are considered infeasible due to long turn around times and significant computational resources. Therefore, the flow solver techniques are programmed for clusters with NVIDIA GPU accelerators.

The contributions of this thesis are built upon an existing multi-GPU, incompress-

ible, three-dimensional flow solver that is the prior work of Julien Thibault [85] and Thibault and Senocak [86], who developed a GPU-accelerated laminar flow solver that was demonstrated as a basis for an urban dispersion model, and Dana Jacobsen [34], who transformed the work of Thibault for use on GPU clusters [36, 37] and created a full-depth parallel geometric multigrid solver with an amalgamation strategy for the pressure Poisson equation [35]. Also, the pre-processor for the IBM (described later in Section 4.2) was jointly developed by the author of this thesis along with Kyle Felzien, a student in Computer Science at Boise State University [17].

This thesis is organized into chapters based on the numerical techniques used for this wind forecasting simulation developed in this study. A chapter is devoted to the governing equations and numerical methods of the present flow solver, which includes a discussion of the GPU and the programming implementation. The implementation and validation of LES and IBM each receive their own chapters. The coupling of the hybrid RANS/LES with IBM along with a simulation attempting to replicate an experimental field study is discussed in the fifth chapter. Conclusion and recommendations for the future direction of this study are given in the final chapter. Relevant literature reviews are provided in each chapter.

1.2 Works Published

Works published as part of this thesis:

- R. DeLeon, I. Senocak, "GPU-accelerated Large-Eddy Simulation of Turbulent Channel Flows," 50th AIAA Aerospace Sciences Meeting, Nashville, TN, January 2012.
- R. DeLeon, D. Jacobsen, I. Senocak, "Large Eddy Simulations of Turbulent Incompressible Flows on GPU Clusters," pre-print, Computing in Science and Engineering, March 2012.
- R. DeLeon, K. Felzien, I. Senocak, "Immersed Boundary Turbulent Flow Simulations on GPU Clusters," Poster presented at the NVIDIA GPU Technology Conference, San Jose, CA, May 2012.
- R. DeLeon, K. Felzien, I. Senocak, "Toward a GPU-accelerated Immersed Boundary Method for Wind Forecasting Over Complex Terrain," Conference paper to be presented at the ASME Fluids Engineering Division Summer Meeting, Rio Grande, PR, July 2012.

CHAPTER 2

TECHNICAL BACKGROUND

This chapter provides the governing equations and numerical methods used in this flow solver. The core components for the wind solver are accelerated by the massively-parallel, many-core graphics processing unit (GPU) to realize a forecasting capability. A discussion of the GPU and the programming implementation of the previous GPU-accelerated, incompressible flow solver [34, 85] that is the starting point for the wind forecasting capability developed in this study is also included.

2.1 Governing Equations

The governing equations for LES of incompressible flows are the filtered form of the Navier-Stokes equations given as,

$$\frac{\partial \overline{u}_j}{\partial x_j} = 0 \tag{2.1}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{u}_i \overline{u}_j\right) = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\nu \overline{S}_{ij} - \tau_{ij}\right), \tag{2.2}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) \tag{2.3}$$

is the deformation tensor, and

$$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j \tag{2.4}$$

is the tensor representing the interaction of the subgrid-scales on the resolved large-scales. The overbar in these equations represents a filtered quantity. The numerical mesh typically provides this filter.

2.2 Numerical Methods

The governing equations were solved on a directionally-uniform Cartesian grid using the projection algorithm [15] with second-order central difference schemes for spatial derivatives and a second-order Adams-Bashforth scheme for time advancement. The projection algorithm predicts the velocity by removing the pressure term from the governing Navier-Stokes equations to get

$$\mathbf{u}^* = \mathbf{u}^t + \Delta t \left(-\mathbf{u}^t \nabla \cdot \mathbf{u}^t + \nu \nabla^2 \mathbf{u}^t \right). \tag{2.5}$$

A Poisson equation for pressure can then be written by imposing a divergence free condition on the velocity field at time $t + 1$,

$$\nabla^2 P^{t+1} = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*. \tag{2.6}$$

The pressure field at time $t + 1$ is found by solving Equation 2.6 with a geometric, three-dimensional multigrid method with a weighted Jacobi solver. The pressure field is then used to correct the predicted velocity, $\mathbf{u}^*$ as follows

$$\mathbf{u}^{t+1} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla P^{t+1}. \tag{2.7}$$

The basic idea behind multigrid [12, 88] is to solve the problem on multiple meshes to reduce long- and short-wavelength errors and accelerate the convergence. The most basic multigrid routine, termed the *V-cycle*, is to coarsen the original mesh in levels (i.e., repeatedly halving the number of grid points) until the coarsest mesh possible is achieved. Coarsening the mesh is referred to as restriction. The solution is solved at each level to smooth the results until the coarsest mesh is achieved. The solution is directly solved on the coarsest grid and then interpolated back up the levels to the original mesh in the prolongation stage.

2.3 GPU Computing

GPU computing or general-purpose computing on GPUs (GPGPU) are terms referring to executing algorithms on the GPU, which are typically handled by the central processing unit (CPU), such as scientific numerical algorithms. The GPU is a massively-parallel, many-core architecture typically responsible for computer graphics and, in recent years, has been proven to accelerate scientific calculations in a variety of fields [65, 66]. GPUs have received a lot of attention by the scientific computing community over the last five years because of the introduction of NVIDIA's Compute Unified Device Architecture (CUDA) in 2007 [76]. Scientific computations had been attempted on the GPU prior to CUDA, but it was a painstaking task because the algorithms had to be disguised in a graphics programming language such as OpenGL or DirectX. The effort was usually not worth the performance gains to scientists. Brook [13] was released in 2004, which made programming for the GPU easier but it wasn't until CUDA was debuted that GPU computing became mainstream. CUDA-enabled devices boasted a unified shader pipeline that

previously had been two different pipelines, one for pixel shaders and the other for vertex shaders. With one pipeline, a programmer could easily harness all the resources on a GPU [42, 76]. Along with a GPU whose architecture was tailored towards scientific computations, NVIDIA also released the CUDA C programming language [63]. CUDA C, commonly referred to as just CUDA, is a very scalable, single instruction on multiple data (SIMD) language that is an extension of C. Because scientists no longer had to learn complicated graphics languages, the GPU was rapidly adopted by the scientific computing community for the massive data parallelism that could be achieved by the GPU hardware.

GPUs can provide significant speedups to traditional CPU codes. However, one must know the architecture of the GPU and the optimal programming techniques to realize these speedups [62]. While adding a bit more rigor to the programming task, disregarding the architecture may cause an application to run slower than its CPU counterpart. For a forecasting application where speed is essential, optimizing the code to best exploit the GPU architecture is also essential.

Figure 2.1 is a simple depiction of the CUDA architecture. NVIDIA GPUs consist of several streaming multiprocessors (SM), each of which consists of eight streaming processors (SPs) on the first generation CUDA architecture (e.g., the NVIDIA Quadro FX 5800), 32 SPs on the first release of the Fermi GPU architecture (e.g., the NVIDIA Tesla C2075), or 48 on the latest release of the Fermi GPU architecture (e.g., Quadro 2000D). A SP, also referred to as a CUDA core, is not a core in the traditional sense, but rather it is an arithmetic logic unit (ALU) capable of only arithmetic operations and relies on the SMs to give it instructions. The SM on newer architectures can deploy 32 threads per SP, referred to as a warp, which are of low latency and can be created easily [63]. Each SM also has its own set of memory caches, including thread registers and shared memory, which can only be used by the SPs on that

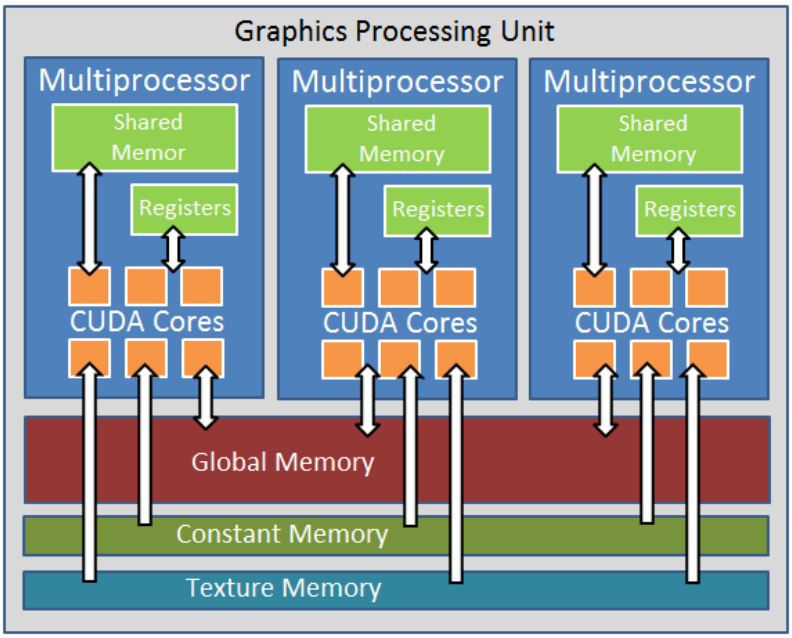

Figure 2.1: A simple illustration of a CUDA-enabled GPU hardware architecture. Actual configurations of streaming multiprocessors and CUDA cores vary depending on the particular model of NVIDIA GPU. The two-headed arrows show how information is transferred between different components.

specific SM [42]. NVIDIA GPUs have onboard dedicated memory referred to as global memory that can be used by any SM but is slower to access than the shared memory or thread registers [62]. The global memory is used to transfer data between the device (GPU) and the host (CPU) memory. Some GPUs, such as the NVIDIA Tesla C2075, have up to 6 GB of onboard memory and 448 SPs allowing researchers to tackle very large problems.

The CUDA programming model starts with a kernel. Each kernel is a set of instructions initiated on a CPU and performed by all the GPU threads. In CUDA, threads are grouped together in blocks where the optimal number of threads per block is a multiple of the half warp size for older architectures and full warp size for newer architectures [62]. Each block is given to a SM by a thread scheduler where the computations are performed [42]. The entire set of blocks is referred to as a grid [63].

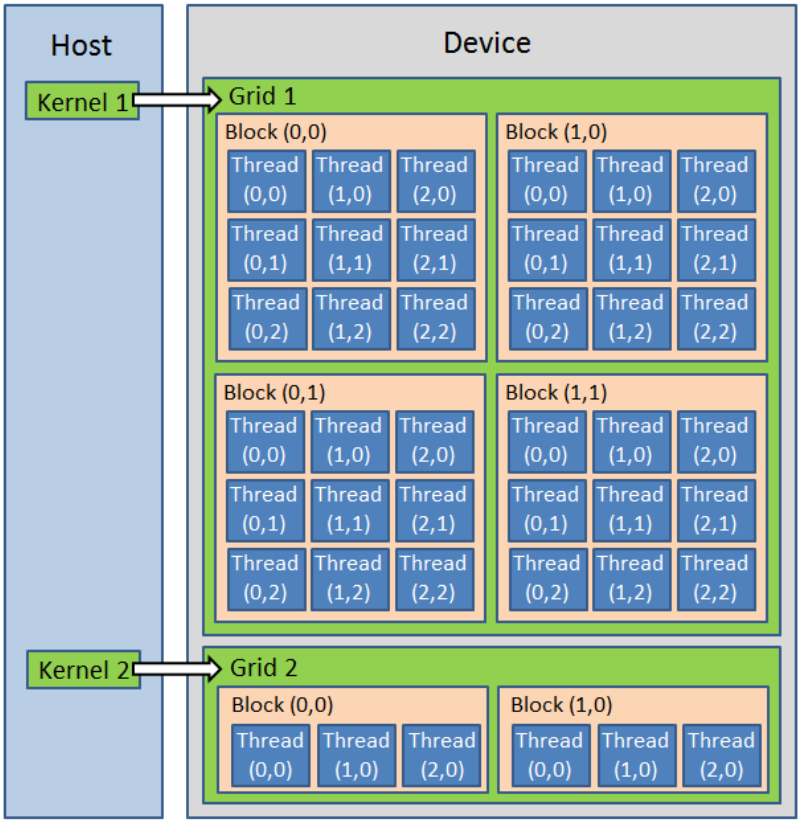

Figure 2.2: A simple depiction of the CUDA programming model. The kernel is initiated on the CPU and then divided up into a grid of blocks. Each block, which consists of multiple threads, is then given to a SM. Note that different grid sizes can be used for different kernels.

Figure 2.2 illustrates how blocks and threads are arranged.

Threads operate independently of each other and built-in variables in CUDA C are used to determine the address of the data a thread will access. The data resides in global memory, which is the onboard dedicated memory, and can be copied between the device and host at any time although global memory accesses and host/device communication are relatively slow processes. Shared memory, while limited, allows for communication among the threads and can improve performance by reducing global memory accesses. The latency of accessing global memory is high compared to accessing SM caches [62]. Also, the global memory is sequentially aligned in segments

and the thread accesses should reflect this same alignment otherwise performance decreases. Aligned thread accesses are not always possible and therefore global memory accesses must be coalesced [63]. Minimizing the amount of host-device data transfer is also imperative for achieving maximum performance due to the very slow bandwidth of CPU/GPU communication compared to the bandwidth of global memory accesses [62].

2.4 GPU Cluster Implementation

The work performed for this thesis was built upon a flow solver first developed by Julien Thibault [85] and later advanced by Dana Jacobsen [34]. Their work and others [35–37, 86] created a parallel three-dimensional, incompressible Navier-Stokes flow solver accelerated by multiple GPUs that uses dual-level parallelism by interleaving CUDA and Message Passing Interface (MPI). The same programming optimizations in the previous flow solver are adopted in the algorithms used in the current study.

MPI [30] is a parallel programming language that consists of portable message-passing systems that allows multiple processes to communicate with each other. Similar to CUDA, MPI is a C-based language. In the present flow solver, MPI handles the coarse-grain parallelism (partitioning the data into large sections) while CUDA handles the fine-grain parallelism (executing parallel instructions on individual data elements).

Communication is always a bottleneck in parallel computing and must be minimized to achieve the best performance. Interleaving MPI and CUDA produces two different communication bottlenecks: CPU/GPU data transfer and network communication among a cluster. In an effort to reduce the negative effects of communication, MPI communication and CPU/GPU data transfers are overlapped with computations using non-blocking MPI communication calls and asynchronous CUDA memory copy-

ing operations using CUDA streams. Non-blocking communication routines allow the program to continue without having to finish the data transfer. Asynchronous memory transfer are also non-blocking and allow the program to continue without the completion of the CPU/GPU data transfer. This communication strategy is described in detail in Jacobsen's thesis [34] and in Jacobsen et al. [37].

Multigrid methods are worthwhile to implement because of their superior convergence rates [88] but require special care when implementing in parallel. The problem that arises is that the entire domain needs to be coarsened to achieve the best convergence but the domain is distributed over multiple processes. Therefore, the entire computational mesh can only be coarsened so far on multiple processes before data starvation becomes an issue. To address this, Jacobsen [34] and Jacobsen and Senocak [35] developed an amalgamation strategy that brings the distributed grid partitions to one process and continues the multigrid cycle. During prolongation, the mesh is broadcasted back to the distributed process at the level where amalgamation took place.

CHAPTER 3

LARGE-EDDY SIMULATION TECHNIQUE

LES uses the idea that larger eddies, which interact with and extract energy from the mean flow, are highly dependent on the geometry of the problem domain, boundary conditions and body forces, while smaller eddies exhibit a more universal behavior being nearly isotropic [94]. Therefore, the effect of the smaller eddies could be captured by a model while the larger eddies could be resolved. Mathematically, this can be accomplished by specifying a cutoff length with a mathematical low-pass filter. The smaller eddies that pass below the filter are called subgrid-scale (SGS) eddies and must be replaced by a SGS model [48]. Several challenges face LES. One comes from the replacement of SGS eddies with a model that introduces errors to the simulation [75]. Another is the computational expense of wall-bounded flows at high Reynolds numbers [94]. A third challenge is boundary conditions introducing errors when the flow is not deterministically known, particularly with inflow conditions that do not introduce the proper turbulent kinetic energy and flow structure information [39].

James W. Deardorff is considered the pioneer of LES, being the first to simulate turbulent channel flow at a large Reynolds number [16] with a SGS model. He used the Smagorinsky eddy-viscosity model [80], which became one of the most popular SGS models. Even though the simulation had a very coarse domain, Deardorff's work

showed the feasibility of three-dimensional computation of turbulence. This led to future studies [58, 72, 74] of LES applied to turbulent channel flow to gain a better understanding of wall-bounded turbulent flows. Throughout the years, development of several SGS models and tremendous advances in computational hardware have enabled LES to make a major impact in applications such as combustion, aerodynamics of vehicles, aero-acoustics, propulsion, turbomachinery, and atmospheric modeling [48, 75].

LES is one of three numerical approaches to turbulence calculation. The other two are Reynolds-averaged Navier-Stokes (RANS) and direct numerical simulation (DNS). In the RANS approach, the governing equations are time-averaged such that the resulting quantities are mean values. This is the least computationally expensive form of turbulence calculation and is thus widely adopted in industry [94]. However, RANS cannot resolve most flow structures given the time-averaged governing equations and requires an additional turbulence model to account for their effect. To date, no single turbulence model can be considered universal for turbulent flow simulations, and therefore must be chosen depending on the flow scenario. The DNS approach numerically resolves all scales of turbulent flow as is by far the most computationally expensive. For example, a modest Reynolds number of 10^4 would require approximately 10^9 grid points in a DNS [94]. Thus, DNS is primarily used in fundamental turbulence research.

Determination of inflow conditions and resolution of wall-bounded flows are two of the major challenges of LES. Inflow conditions need to have the energy and flow structure information and are the focus of many studies. Keating et al. [39] survey several approaches to specify inflow conditions. Some examples being recycling the outflow planes using similarity or using a precursor simulation as an inflow condition. In the current study, inflow conditions are set as periodic and further investigation

will be the focus of future work.

The full resolution of boundary layers at high Reynolds numbers using LES is impractical. Thus, wall models have been proposed to alleviate the near-wall resolution requirements by modeling the inner-layer scales with a velocity profile relating the outer-layer velocity to wall stress or using a Reynolds-averaged parametrization. Piomelli [70] gives a survey of wall-modeling methods for LES with some examples being the use of a constant-stress shear layer near the wall or using a hybrid RANS/LES methods that uses RANS to calculate the near-wall flow while LES calculates the flow away from the wall. In the present study, a hybrid RANS/LES approach is implemented.

3.1 Subgrid Scale Models

For turbulence closure in LES, the subgrid-scale motions are replaced by a SGS model [47, 53]. Arguably the most common type of SGS model is the Smagorinsky eddy-viscosity model [80], which creates a proportionality between local SGS stresses and the second invariant of the strain rate tensor. There are several derivatives of the Smagorinsky model including dynamic models [21]. Another type of SGS model is the spectral eddy-viscosity model that closes the governing equations in Fourier space [43]. Giving information about possible stretching and dislocations of the vortex field from perturbations in the flow, spectral eddy-viscosity models provide very good results for mixing layers [47]. The structure-function models use local kinetic energy spectrums in physical space to model the SGS scales [55]. While being costly to implement, models based on structure functions provide good results for mixing layers particularly at high Mach numbers. Similarity models postulate that SGS scales are similar to those above the filter width [2]. Correlations between actual stresses and stresses produced by similarity models are high. This model also produces realistic

backscatter of energy but does not dissipate energy well and is typically blended with an eddy-viscosity model [53]. The present study uses the Lagrangian dynamic model [54] based on the original Smagorinsky eddy-viscosity model for its applicability to complex geometries and practicality of implementation.

The original Smagorinsky model [80] is perhaps the most popular SGS model but has well-known deficiencies. The model parameters are constant and are chosen *a priori* [16]. Constant parameters do not cause the eddy viscosity to vanish at near wall boundaries. An ad hoc fix to the problem is to use van Driest damping [18].

In 1991, Germano et al. [21] proposed an alternative method to dynamically calculate the empirical parameters in a SGS model using information from the resolved velocity field. The dynamic Smagorinsky model introduced by Germano et al. correctly predicts a decaying eddy viscosity near wall boundaries, but it has the disadvantage of requiring homogeneous directions in the flow problem at hand, which has later been addressed by other researchers [22, 54, 71].

The original Smagorinsky model relates local SGS stresses with the local rate of strain on the large-scale eddies. It is given by

$$\tau_{ij} = -2\nu_t \overline{S}_{ij} + \frac{1}{3}\tau_{ii}\delta_{ij} \tag{3.1}$$

where ν_t is the turbulent or SGS eddy viscosity and is calculated by

$$\nu_t = (C_S\Delta)^2\sqrt{2\overline{S}_{ij}\overline{S}_{ij}}. \tag{3.2}$$

Δ is the filter width and can be defined by either a mathematical filter (e.g., top-hat filter) or by using the numerical grid as $\Delta = \sqrt[3]{dx \cdot dy \cdot dz}$, where dx, dy and dz are the grid spacings [16]. Before dynamic SGS models were developed, the C_S model coefficient was a constant parameter in the original Smagorinsky model. Choosing a

proper C_S value, which depends upon the mesh and the flow problem being investigated, is critical. For wall boundaries in a channel flow, the model parameter C_S must be adjusted to reflect the vanishing eddies by multiplying C_S with the van-Driest damping function [18], which is given as

$$1 - exp\left(\frac{-y^+}{A}\right), \tag{3.3}$$

where y^+ is the non-dimensional distance given in wall units and A is a constant that is approximately 25.

Damping the Smagorinsky coefficient through an arbitrary function significantly improves the LES results, but the procedure is ad-hoc and does not readily extend to complex geometry. This particular shortcoming is overcome by the adopting the dynamic procedure [21, 49, 69]. The dynamic procedure uses information from the existing flow field to calculate the model coefficient dynamically while the simulation is advancing.

The first dynamic subgrid scale model was proposed by Germano et al. [21]. In their dynamic procedure, a second filter with a larger width, denoted by the hat, is applied to a resolved field. The basis of the dynamic procedure is the Germano identity

$$L_{ij} = T_{ij} - \widehat{\tau_{ij}}. \tag{3.4}$$

The individual terms in this algebraic relation are given by

$$T_{ij} = \widehat{\overline{u_i}\overline{u_j}} - \widehat{\overline{u}}_i\widehat{\overline{u}}_j, \tag{3.5}$$

$$\widehat{\tau_{ij}} = \widehat{\overline{u_i}\overline{u_j}} - \widehat{\overline{u_i}\overline{u_j}}. \tag{3.6}$$

The tensor, L_{ij}, is referred to as the Leonard stresses and can be calculated as follows

$$L_{ij} = \widehat{\overline{u}_i \overline{u}_j} - \widehat{\overline{u}}_i \widehat{\overline{u}}_i. \tag{3.7}$$

Using the Germano identity and the Smagorinsky model, Germano et al. [21] proposed to calculate C_S by

$$C_S^2 = \frac{1}{2} \frac{\langle L_{ij} \overline{S}_{ij} \rangle}{\langle M_{ij} \overline{S}_{ij} \rangle}, \tag{3.8}$$

which uses spatial averaging in homogeneous directions as denoted by the angle brackets. The advantages of this method are that an arbitrary damping function is no longer required to make eddy viscosity diminish near walls and determination of an *a priori* C_S is no longer necessary. The dynamic Smagorinsky model was later modified by Lilly [49] who used a least-squares method to obtain

$$C_S^2 = \frac{1}{2} \frac{\langle L_{ij} M_{ij} \rangle}{\langle M_{ij} M_{ij} \rangle}. \tag{3.9}$$

In both cases, the tensor, M_{ij}, is given by

$$M_{ij} = 2\Delta^2 \left[\widehat{|\overline{S}| \overline{S}_{ij}} - \alpha^2 |\widehat{\overline{S}}| \widehat{\overline{S}}_{ij} \right], \tag{3.10}$$

where α represents the ratio of filters and is typically 2. The dynamic Smagorinsky model has the disadvantage of requiring spatial averaging in homogeneous directions to smooth C_S and stabilize the computations. Ghosal et al. [22] put the dynamic procedure on a better mathematical foundation through a constrained variational formulation where the averaging of the dynamic coefficient in homogeneous direction is justified. But most practical flow problems lack a homogeneous direction. There-

fore, Meneveau et al. [54] proposed a dynamic model from a Lagrangian perspective by averaging along the flow pathlines rather than in homogeneous directions. The idea is to minimize the error caused by using the Smagorinsky model and the Germano identity by taking previous information along the pathline to obtain a current value. This formulation applies to fully inhomogeneous turbulent flows as seen in many engineering applications and requires less computational resources than other localized dynamic models [22], therefore making it a practical option in fluids engineering.

The Lagrangian dynamic model uses backward time integration and an exponential weighting function that decreases the weight of past events. The weighted backward time integration can then be written as two relaxation-transport equations

$$\frac{\partial \mathcal{J}_{LM}}{\partial t} + \overline{\mathbf{u}} \cdot \nabla \mathcal{J}_{LM} = \frac{1}{T} \left(L_{ij} M_{ij} - \mathcal{J}_{LM} \right), \tag{3.11}$$

$$\frac{\partial \mathcal{J}_{MM}}{\partial t} + \overline{\mathbf{u}} \cdot \nabla \mathcal{J}_{MM} = \frac{1}{T} \left(M_{ij} M_{ij} - \mathcal{J}_{MM} \right), \tag{3.12}$$

where T is the relaxation time scale. Meneveau et al. [54] chose to define T as

$$T = 1.5\Delta \left(\mathcal{J}_{LM} \mathcal{J}_{MM} \right)^{-1/8}. \tag{3.13}$$

After solving Equations 3.11 and 3.12, the value of C_S is then calculated using the relation,

$$C_S^2 = \frac{\mathcal{J}_{LM}}{\mathcal{J}_{MM}}, \tag{3.14}$$

which can be directly substituted into the Smagorinsky eddy-viscosity model.

The Lagrangian dynamic Smagorinsky model was chosen for the current application for being practical to implement and does not require statistically homogeneous directions, which do not exist in arbitrarily complex terrain. The Lagrangian dynamic

model requires two filters. The base filter is the computational mesh. A simple top-hat filter is used as the second filter in the Lagrangian dynamic model. Time advancement in the Lagrangian dynamic model (Equations 3.11 and 3.12) is performed using the first-order schemes recommended by Meneveau et al. [54], which are given as

$$\mathcal{J}_{LM}^{n+1}(\mathbf{x}) = H\left\{\epsilon\left[L_{ij}M_{ij}\right]^{n+1}(\mathbf{x}) + (1-\epsilon)\,\mathcal{J}_{LM}^{n}(\mathbf{x}-\overline{\mathbf{u}}^{n}\Delta t)\right\}, \tag{3.15}$$

$$\mathcal{J}_{MM}^{n+1}(\mathbf{x}) = \epsilon\left[M_{ij}M_{ij}\right]^{n+1}(\mathbf{x}) + (1-\epsilon)\,\mathcal{J}_{MM}^{n}(\mathbf{x}-\overline{\mathbf{u}}^{n}\Delta t), \tag{3.16}$$

where

$$\epsilon = \frac{\Delta t/T^{n}}{1+\Delta t/T^{n}} \tag{3.17}$$

and T is defined in Equation 3.13. The ramp function in Equation 3.15 is required to clip away negative C_S^2 values that result from numerical inaccuracies. A trilinear interpolation scheme evaluates the "upstream" values at $\mathbf{x}-\overline{\mathbf{u}}^{n}\Delta t$.

3.2 Validation of the LES Technique

Periodic turbulent channel flow at $\mathrm{Re}_\tau = 180$ and 395 were used to validate the LES capability. The dimensions of the computational domain are $(2\pi\delta, \pi\delta, 2\delta)$ in (x, y, z) where δ is the channel half-height, x, y and z are the streamwise, spanwise, and wall-normal directions, respectively. Two grids were used for the $\mathrm{Re}_\tau = 180$ case, a coarse resolution mesh with $64 \times 64 \times 96$ points, and a fine resolution mesh with $128 \times 96 \times 128$ points. Grid stretching was not applied in the wall-normal direction because a structured adaptive mesh refinement strategy is envisioned in future extensions of the present wind solver.

The turbulent channel flow was initialized in an approach similar to Gowardhan [28] that superimposes a sinusoidal fluctuating component on a logarithmic profile as in

$$u = u_\tau \left(\frac{1}{\kappa} \ln y^+ + 5.5 \right) + \sin(\pi z) \cos x \sin y, \tag{3.18}$$

$$v = -\left(1 + \cos(\pi z)\right) \sin x \sin y, \tag{3.19}$$

$$w = -\pi \sin(\pi z) \sin x \cos y. \tag{3.20}$$

The fluctuating components can be scaled by a constant. However, because periodic boundary conditions are imposed in the streamwise directions, the amplitude of the fluctuations did not matter for the current cases as the solution eventually converges on to a statistically stationary turbulent state. Thus, the sinusoidal fluctuation amplitudes were set to unity.

A C_S value of 0.1 was chosen for the original Smagorinsky model. The Lagrangian dynamic model was initialized with the initial conditions recommended in Meneveau et al. [54] that sets $\mathcal{J}_{LM} = C_S M_{ij} M_{ij}$ and $\mathcal{J}_{MM} = M_{ij} M_{ij}$, with C_S also being 0.1.

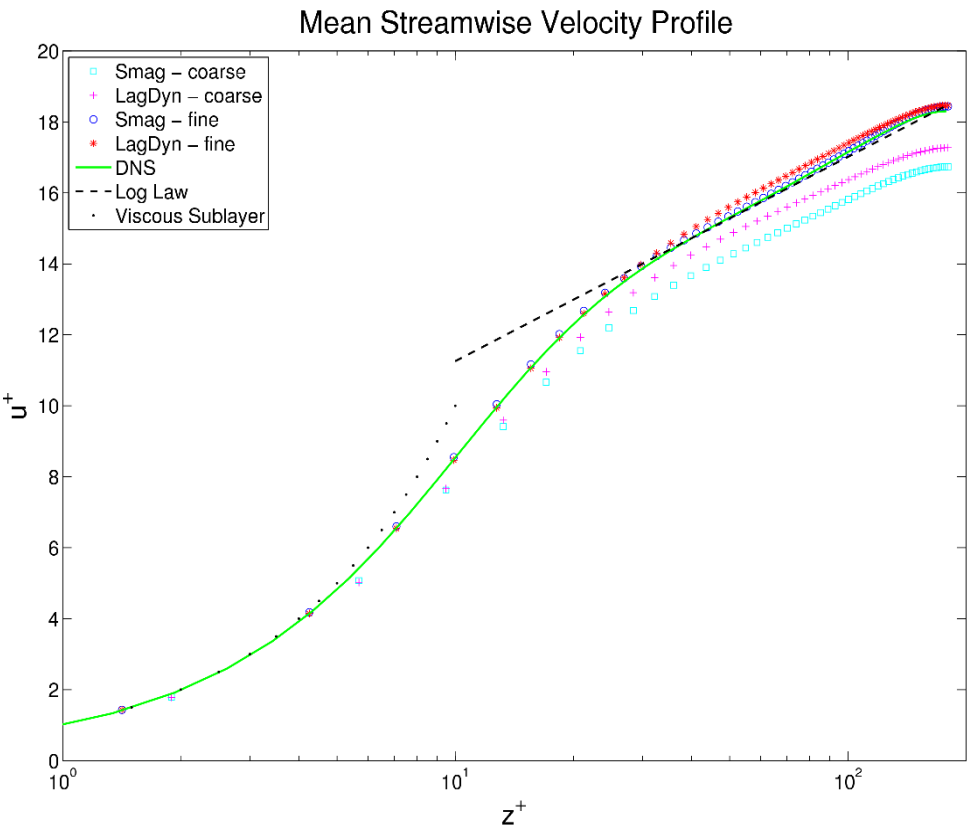

Figure 3.1: A comparison of the mean streamwise velocity profiles using different models and mesh sizes (coarse - $64 \times 64 \times 96$, fine - $128 \times 96 \times 128$): $\square$, Smagorinsky on coarse grid; $+$, Lagrangian dynamic on coarse grid; $\circ$, Smagorinsky on fine grid; $*$, Lagrangian dynamic on fine grid.

Periodic boundary conditions [29] were applied in the stream- and span-wise directions to both velocity and scalar quantities. On channel walls, the no-slip condition was imposed on the velocity field, and Neumann boundary conditions for pressure and the scalar quantities found in the Lagrangian dynamic model were set to zero. The flow was maintained by imposing a height independent constant pressure gradient in the streamwise direction that is u_τ^2/δ. The simulation was allowed to develop for 200 dimensionless time units ($u_\tau t/\delta$). Statistics were sampled for 20 dimensionless time units.

Figure 3.1 shows the mean velocity profiles for the fine and coarse grids with the

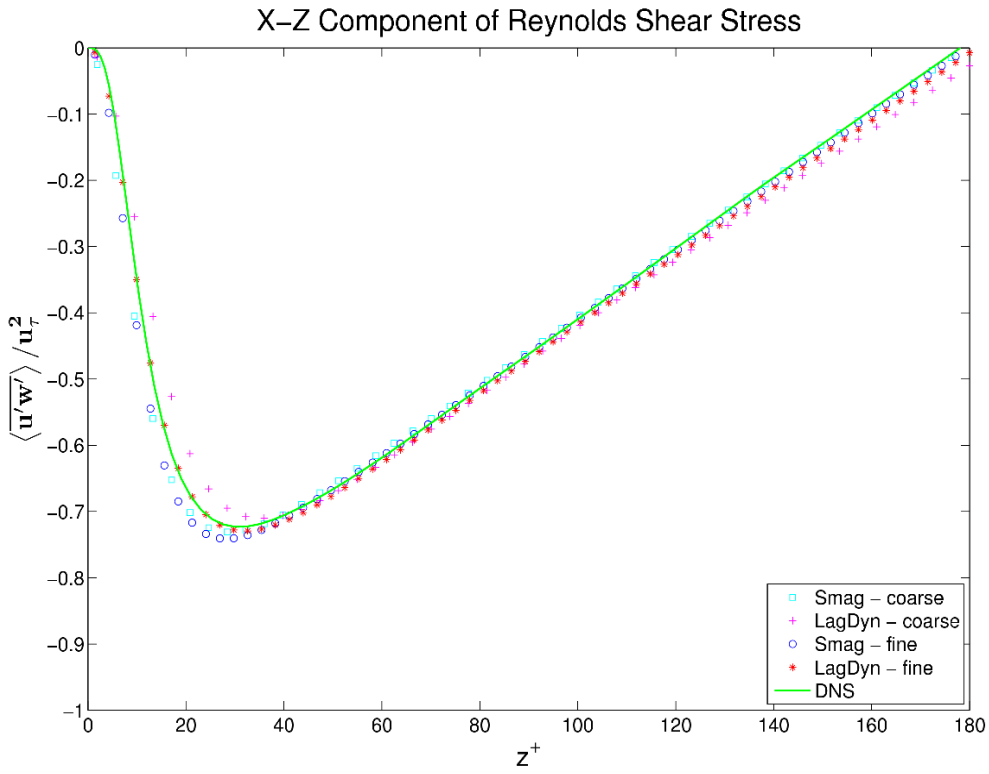

Figure 3.2: A comparison of the x-z component of the Reynolds shear stress tensor using different turbulence models at different grid resolutions. □, Smagorinsky on coarse grid (64 × 64 × 96); +, Lagrangian dynamic on coarse grid; ○, Smagorinsky on fine grid (128 × 96 × 128); ∗, Lagrangian dynamic on fine grid.

Smagorinsky model with van Driest damping and the Lagrangian dynamic model. The profiles were compared to both the theoretical law of the wall and the DNS performed by Moser et al. [59]. As expected, the finer mesh did considerably better than the coarse mesh, having two points in the viscous sublayer as opposed to one.

All simulations did well with the x-z component of the Reynolds shear stress as depicted in Figure 3.2. The Smagorinsky model gave a higher Reynolds shear stress near the wall than the DNS while the Lagrangian dynamic model gave lower values than the DNS. Both models gave larger values of Reynolds shear stress away from the wall, particularly the Lagrangian dynamic at the coarse grid resolution.

The root mean square (rms) values of the velocity fluctuations are shown in Figures

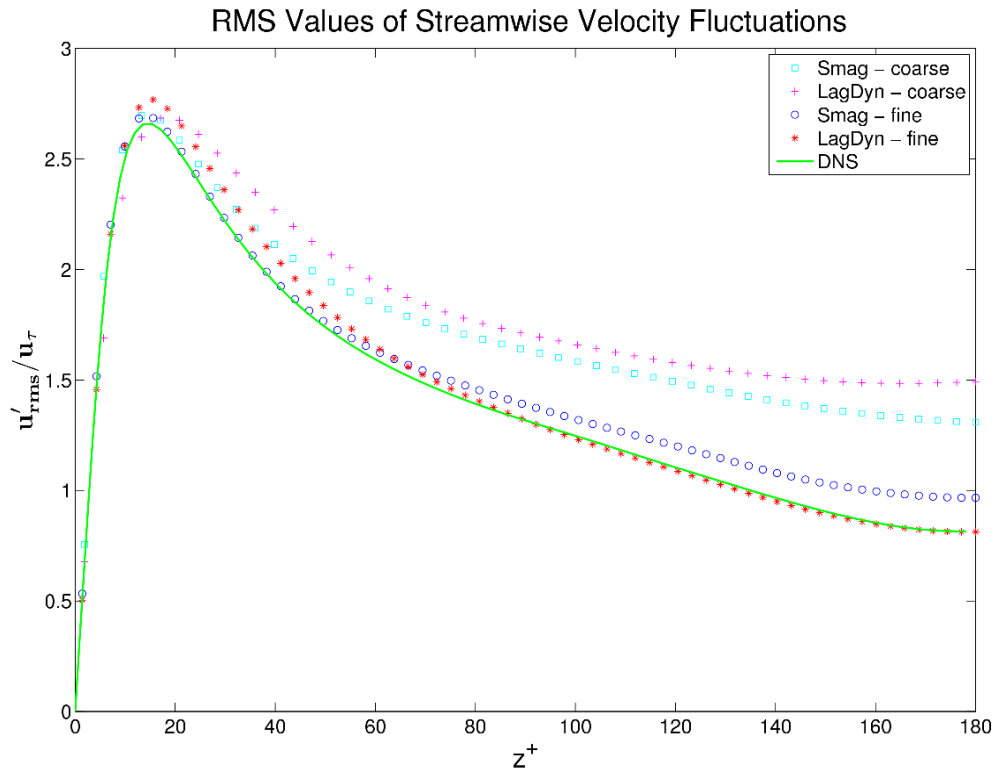

Figure 3.3: The rms values of streamwise velocity fluctuations: □, Smagorinsky on coarse grid ($64 \times 64 \times 96$); $+$, Lagrangian dynamic on coarse grid; ○, Smagorinsky on fine grid ($128 \times 96 \times 128$); $*$, Lagrangian dynamic on fine grid.

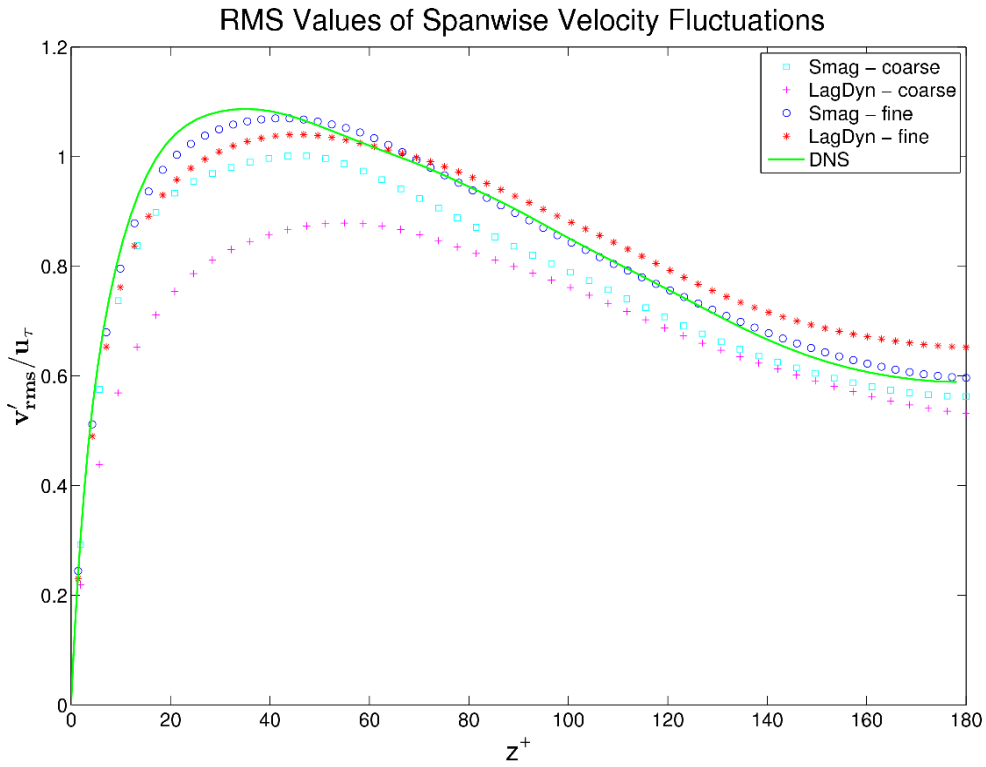

Figure 3.4: The rms values of spanwise velocity fluctuations: □, Smagorinsky on coarse grid (64 × 64 × 96); +, Lagrangian dynamic on coarse grid; ○, Smagorinsky on fine grid (128 × 96 × 128); ∗, Lagrangian dynamic on fine grid.

3.3-3.5. With the streamwise velocity fluctuations in Figure 3.3, the coarse grid Lagrangian dynamic model does worse than the coarse grid Smagorinsky toward the center of the channel but better toward the wall. However, the fine grid approaches yields the exact opposite, with the Smagorinsky model performing better toward the wall but worse away from the wall. With the velocity fluctuations in the spanwise direction (Figure 3.4) and the wall-normal direction (Figure 3.5), the Smagorinsky model gives better results than the Lagrangian dynamic model, on both grids.

Energy spectra are calculated by taking the Fourier transform of the turbulent fluctuation covariance [44]. The streamwise energy spectra from both models on the fine resolution mesh were compared to the theoretical -5/3 slope of the Kolmogorov spectrum [73] in Figure 3.6. Both models produced very similar results and gave a

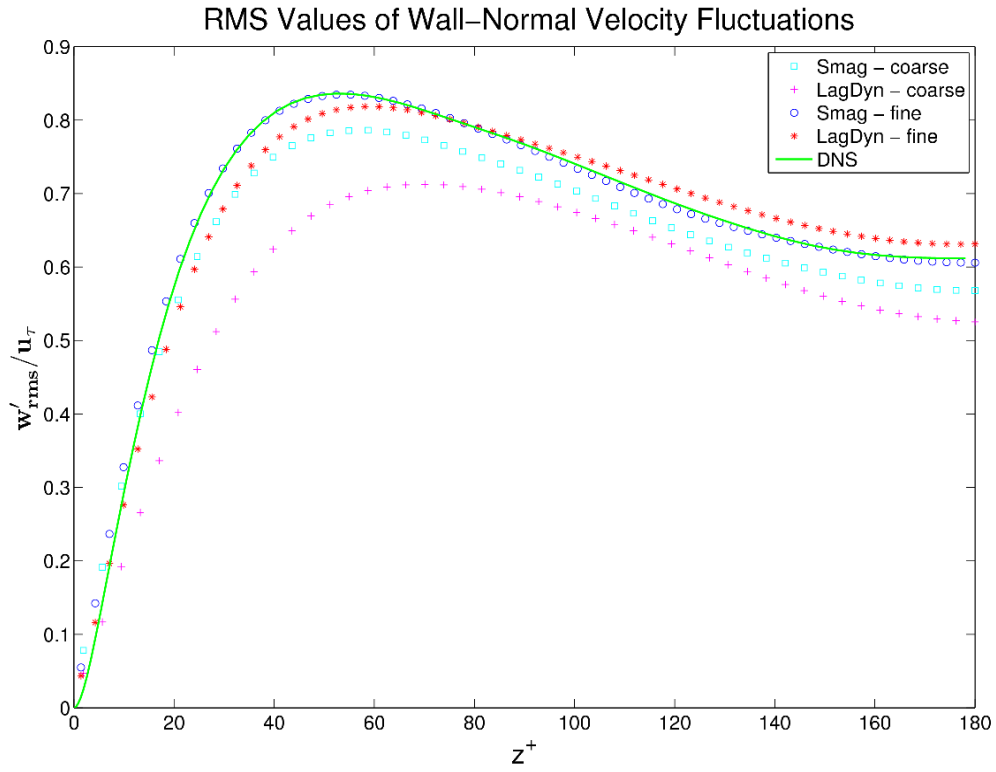

Figure 3.5: The rms values of wall-normal velocity fluctuations: □, Smagorinsky on coarse grid ($64 \times 64 \times 96$); +, Lagrangian dynamic on coarse grid; ∘, Smagorinsky on fine grid ($128 \times 96 \times 128$); ∗, Lagrangian dynamic on fine grid.

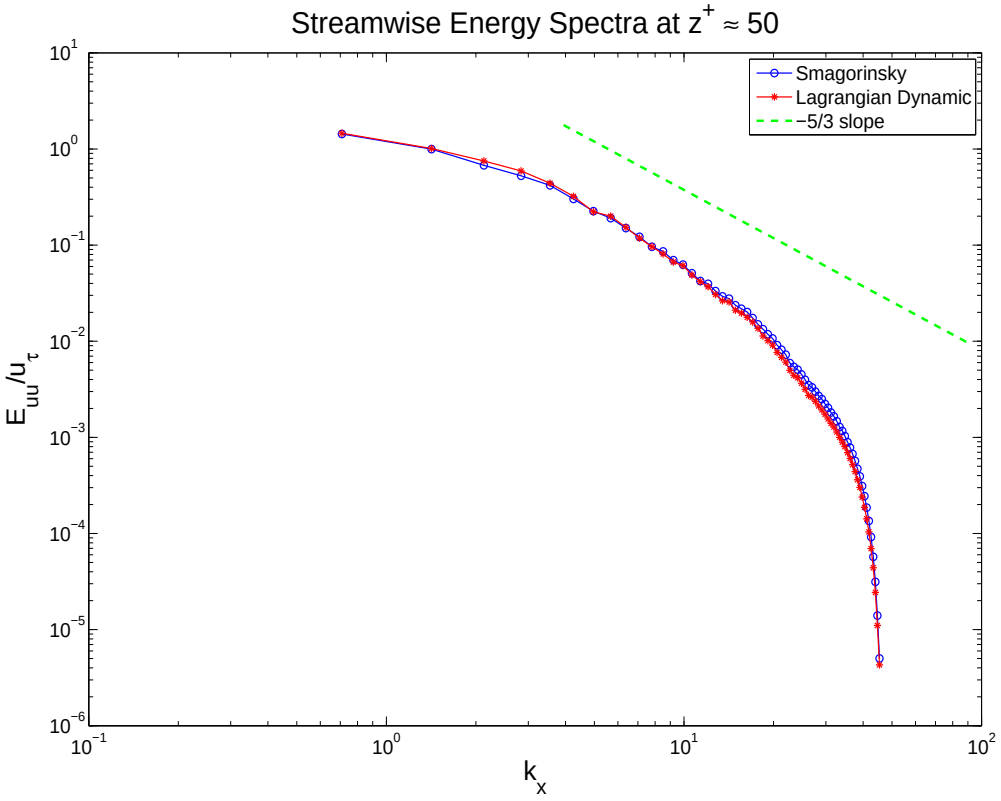

Figure 3.6: Streamwise spectra of turbulent kinetic energy normalized with friction velocity at approximately $z^+ \approx 50$ for $Re_\tau = 180$ on fine resolution mesh (128 × 96 × 128).

slope close to theoretical one roughly around wavenumbers five through ten.

The $Re_\tau = 395$ test case was performed in the same manner as the $Re_\tau = 180$ case, but with a computational mesh of 192 × 128 × 384 to get at least two grid points in the viscous sublayer. Figures 3.7 and 3.8 are the profiles of the mean streamwise velocity and Reynolds stress from the $Re_\tau = 395$ turbulent channel flow simulation using the Lagrangian dynamic SGS model. These exhibit very good agreement with the DNS results [59]. However, the resolution of the mesh is on the order of a DNS since directionally-uniform grids were used. A visualization of the channel flow is given in Figure 3.9, which depicts the isosurfaces of the Q-criterion at Q=400. The Q-criterion is a vortex-identification method useful for visualizing structures in a

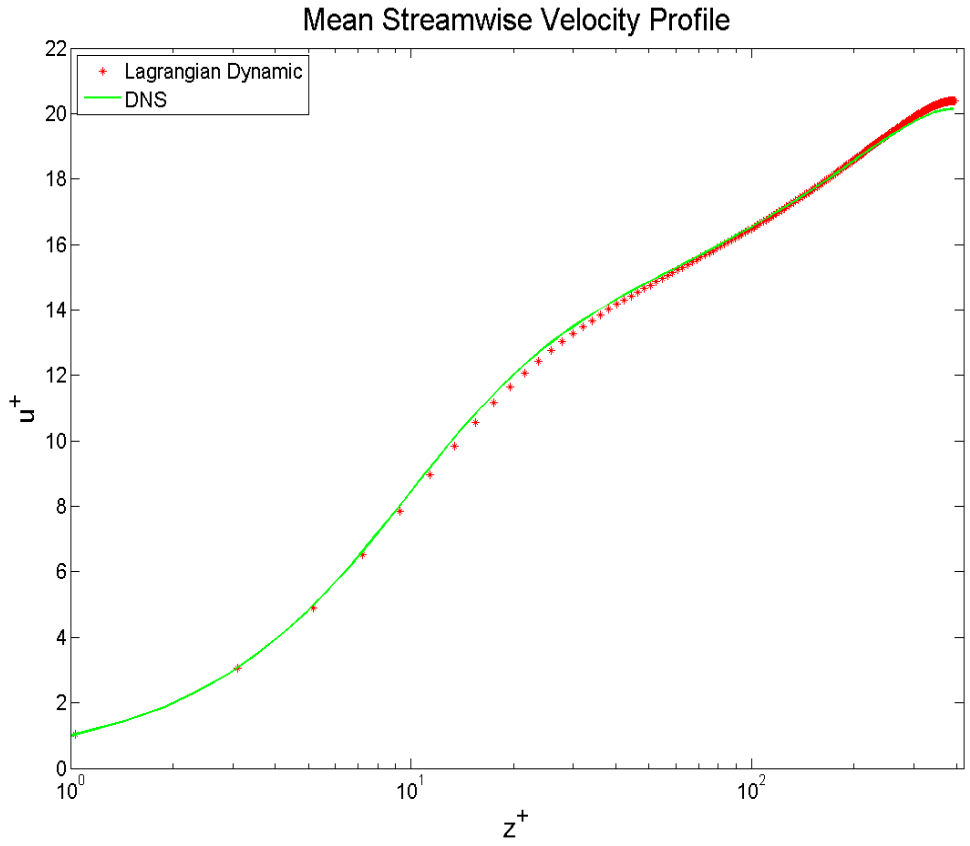

Figure 3.7: The mean streamwise velocity of turbulent channel flow at $\mathrm{Re}_\tau = 395$ compared to DNS results [59]. Only the Lagrangian dynamic model was used.

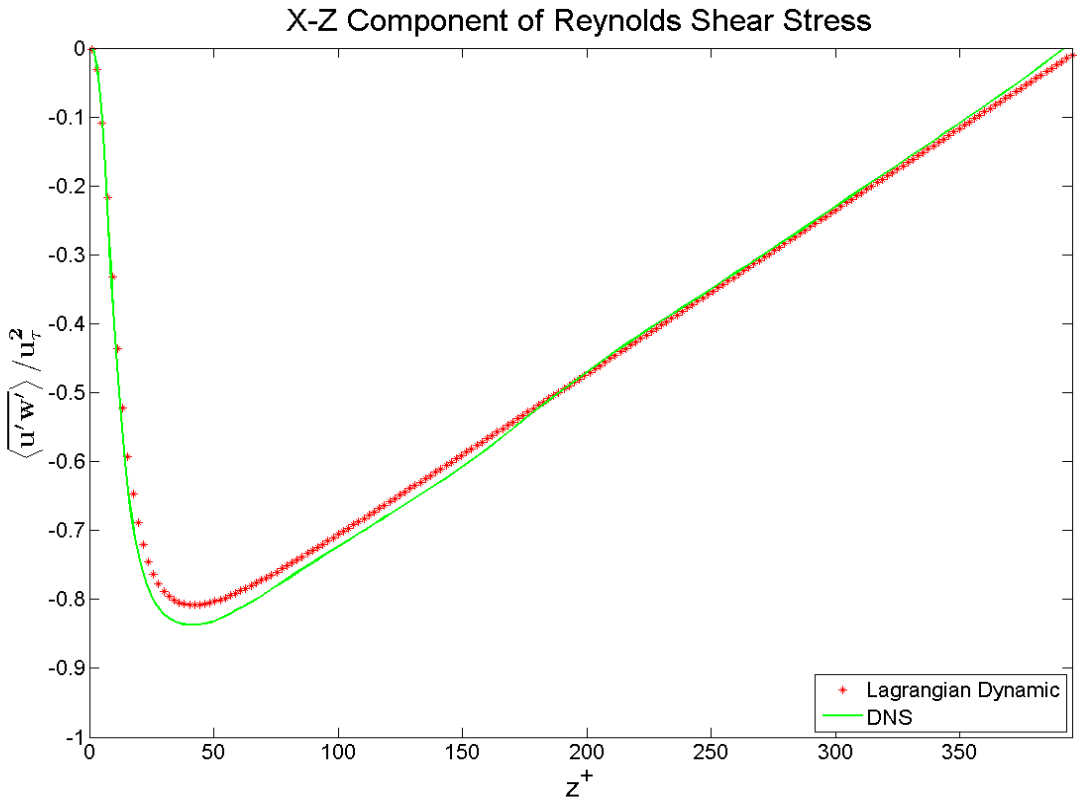

Figure 3.8: The x-z component of Reynolds stress from $\mathrm{Re}_\tau = 395$ turbulent channel flow compared to DNS results [59]. Only the Lagrangian dynamic model was used.

turbulent flow [48]. The abundance of flow structures depicted in Figure 3.9 is an outcome of LES where large-scale motions are resolved in the computations.

The GPU clusters used consisted of NVIDIA Tesla C2070 GPUs connected by PCI Express 2.0 x16 buses and a quad data rate (QDR) Infiniband interconnect. The $\mathrm{Re}_\tau = 180$ case was performed on a single GPU because the problem size was not large enough to benefit from multiple GPUs. The turn around time was 18 hours. The $\mathrm{Re}_\tau = 395$ case was executed on eight GPUs and the turn around time was 45 hours. To give an idea on a typical turn around time for CPU based implementations, Cheng and Liu [14] performed an 13.5 million grid LES with OpenFOAM [84] on eight CPU cores that finished in 1000 hours. The $\mathrm{Re}_\tau = 180$ case was also performed on

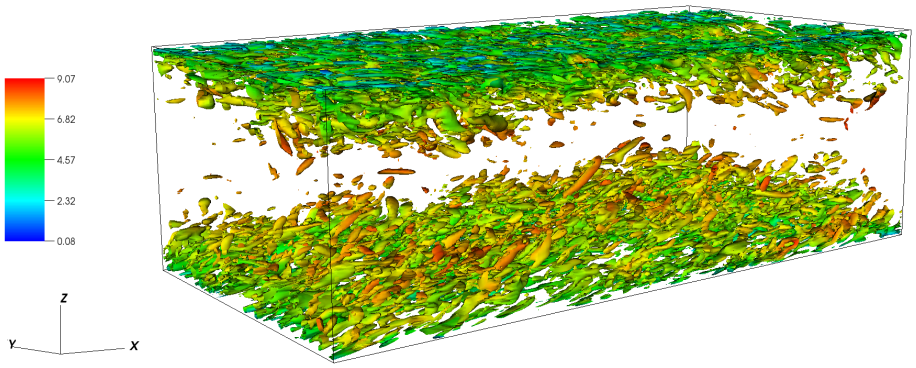

Figure 3.9: A visualization of vortical flow structures using the Q-criterion for the $Re_\tau = 395$ turbulent channel flow. The mesh size used was 192×128×384.

a single NVIDIA GeForce GTX 470 GPU that produced a turn around time of 19.5 hours while providing the same accuracy of results. This reinforces the point that the inexpensive GeForce gaming GPUs can provide performance similar to that of the more expensive Tesla GPUs and can allow researchers an inexpensive tool for problems involving relatively small data sets.

CHAPTER 4

IMMERSED BOUNDARY METHOD

Immersed boundary method (IBM) is a numerical technique that imposes boundary conditions created by embedded solids on Cartesian meshes. This technique eliminates the cumbersome task of generating body-fitted grids. Body-fitted grids may not be suitable for a short-term wind forecasting over highly complex terrain because of the possibility of skewed cells that would introduce significant error to the solution. Also, Cartesian meshes fit much more naturally to the GPU architecture as opposed to unstructured body-fitted grids, resulting in better acceleration of the computations. For these reasons, IBM was chosen for this study. This chapter provides general details of IBM and the specific details of the implementation.

4.1 Overview of Immersed Boundary Methods

The conventional approach to resolving flow around immersed solid bodies is to generate multi-block structured or unstructured grids that conform to the geometry, which can be a cumbersome endeavor. The essential idea behind IBM is to eliminate the time-consuming body-fitted grid generation task by imposing the proper boundary conditions on a Cartesian mesh [56]. This is accomplished by introducing body forcing terms into the governing equations. These forcing terms need to reflect that the boundary of the solid almost always does not coincide with the Cartesian mesh. The

accuracy of the body force terms and the stability of computations has been the focus of many studies.

There are two primary approaches in IBM: continuous forcing and discrete forcing. The discrete forcing approach is used in the present study. The continuous forcing approach came about when the IBM concept was first proposed by Peskin [67] to simulate blood flow in a heart using a Cartesian mesh. This approach adds a body forcing term to the governing equations *before* discretization, requiring the numerical method to resolve another continuous term. The continuous forcing approach works well for flow involving elastic boundaries, which usually falls into biological [9, 68] and multiphase flows [89, 92]. The disadvantage of the continuous forcing approach is formulating a proper body force term without creating additional numerical stability constraints in flow with rigid bodies [56].

The discrete forcing method, proposed by Mohd-Yusof [57] and later applied by Verzicco et al. [95], remedied the numerical stability issue by imposing the forcing term *after* the governing equations are discretized, eliminating the need for calculating an additional continuous term in the governing equations. The discrete forcing method can be categorized further into two different groups of boundary condition imposition: indirect and direct. Indirect imposition is used in this thesis.

In the direct imposition approach, the near-boundary computational grid is modified such that the boundary conditions are imposed directly at the boundary to create a "sharp" interface and is typically used at high Reynolds numbers where accurate boundary layer resolution is necessary [56]. An example of this approach is the ghost-cell method [90] used in finite difference approaches. Figure 4.1 is a simple sketch of the ghost-cell method. Image nodes are created by mirroring the solid nodes included in the computational stencil about the boundary. Solid nodes are assigned values by using interpolation reconstruction schemes involving the image nodes that

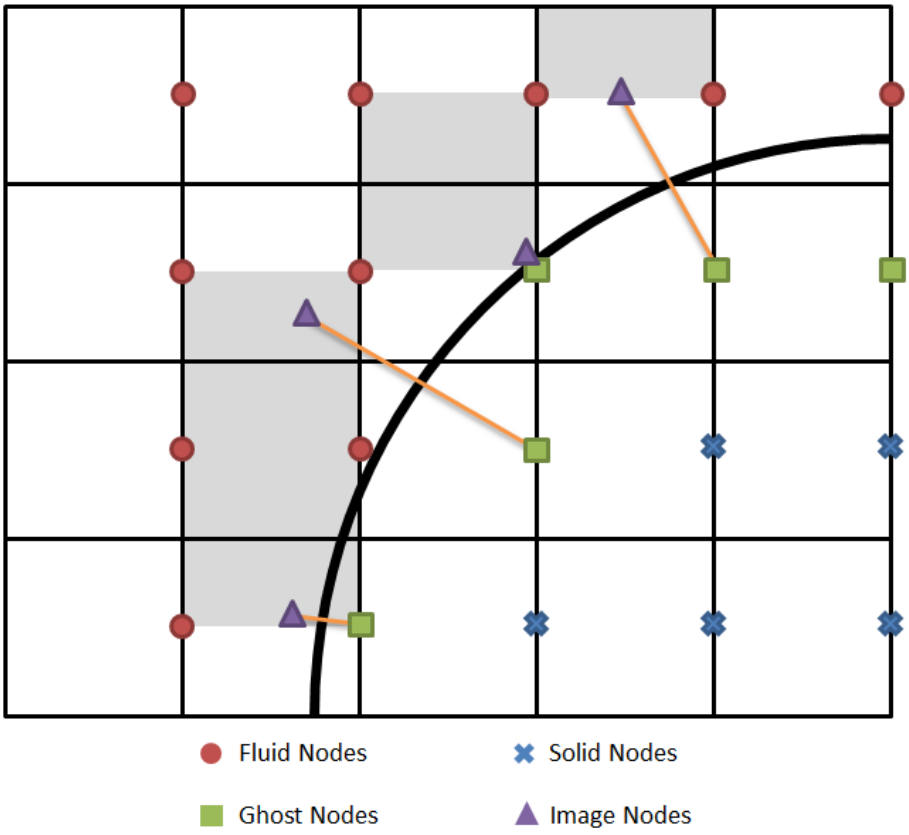

Figure 4.1: A simple sketch of the ghost cell method. Image nodes are created by mirroring the solid nodes included in the computational stencil about the boundary. Solid nodes are assigned values by using interpolation reconstruction schemes involving the image nodes that implicitly applies the boundary condition.

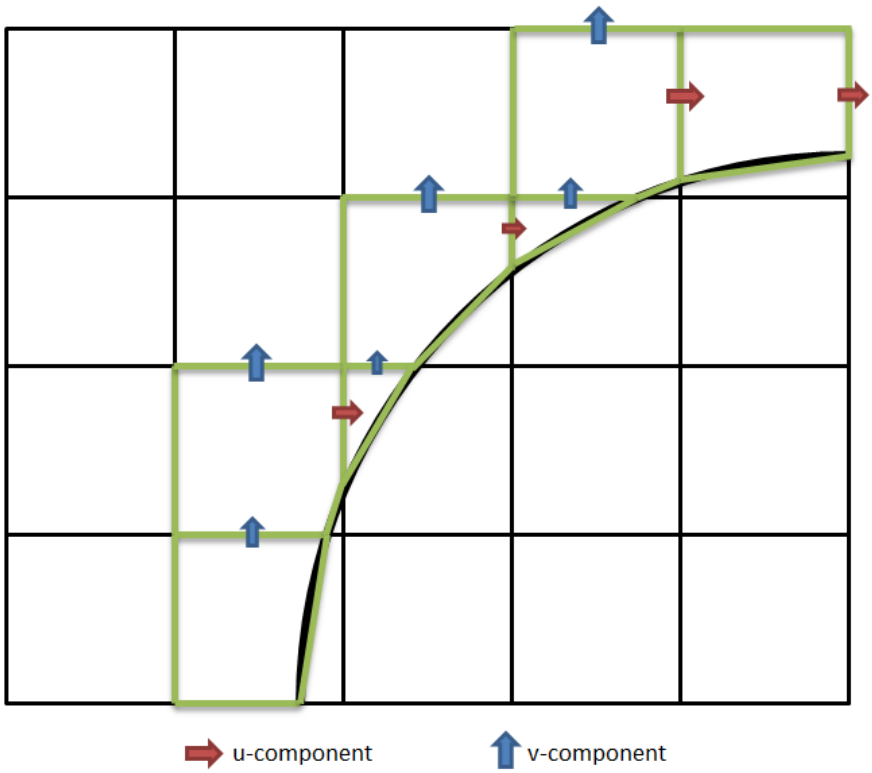

Figure 4.2: A sketch of the cut-cell method. Cells intersecting the solid are reshaped, creating an unstructured mesh at the solid-fluid interface. Cutting the cell essentially reshapes the control volume that the governing equations are solved over.

implicitly applies the boundary condition. As an example, linear interpolation of the velocity at the boundary between a solid node and its corresponding image node should be zero after reconstruction.

The cut-cell method [91] is another direct imposition approach used in finite volume methods. This creates an unstructured grid around the solid by reshaping cells intersecting the solid, as shown in Figure 4.2. Cutting the cell essentially reshapes the control volume that the governing equations are solved over. This approach guarantees the conservation of mass and momentum since control volumes are adjusted to conform with the solid when using a finite volume approach.

The indirect imposition approach reconstructs the velocity field at grid points near the solid boundary with a prescribed boundary condition. Figure 4.3 depicts

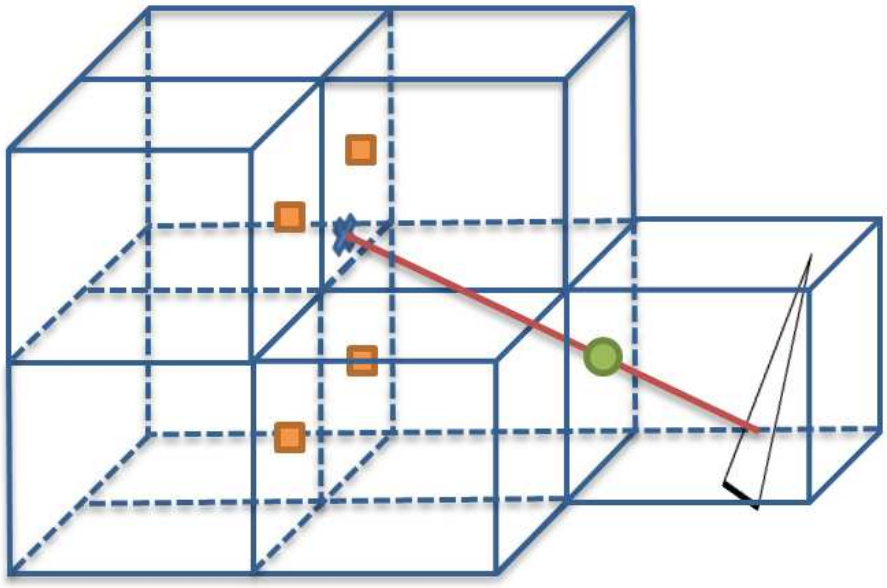

Figure 4.3: A sketch of the indirect imposition approach. A line normal to the surface (triangle) is projected through the immersed boundary node (green circle) until it intersects a plane of resolved values (orange squares). The resolved values are interpolated onto the line and then another interpolation is performed along the line to impose the boundary condition at the immersed boundary node.

the components required for the indirect imposition approach. A line normal to the surface is projected such that it passes through the immersed boundary node that will eventually intersect a plane of resolved fluid nodes. The resolved values are interpolated onto the line and then another interpolation is performed along the line to impose the boundary condition at the immersed boundary node. This indirect approach has been successfully applied by Fadlun et al. [19], Verzicco et al. [95], and Iaccarino and Verzicco [32] for engineering fluid flow applications at moderate Reynolds numbers.

The treatment of the pressure boundary condition at the immersed boundary is different among authors. Fadlun et al. [19] explain that because of the linearization of the governing equations at the immersed boundary, the pressure gradient in the normal direction is zero and an explicit application of a Neumann boundary condition is not necessary. They also described how not including the Neumann pressure boundary condition varies the solution on the order of 10^{-3}-10^{-4} for a linear reconstruction. This was also implemented in Balaras [1]. Kim et al. [41] describe a mass source/sink term in the pressure Poisson equation defined as (in two dimensions)

$$q = -\frac{u_1}{\Delta x} - \frac{v_1}{\Delta y} = \frac{1}{\Delta x \Delta y}\left(-\hat{u}_1 \Delta y - \hat{v}_1 \Delta x\right) \tag{4.1}$$

where u_1 and v_1 are the vector components inside the solid and the hat on the velocity components denotes a predicted velocity. This mass source/sink term can cancels out the error of not including the Neumann boundary condition. Other authors such as Tseng and Ferziger [90] and Ye et al. [96] include the Neumann boundary condition in their respective approaches that provide good results. Ikeno and Kajishima [33] mention that an inconsistency exists between velocity and pressure at the wall when using the aforementioned techniques and they provide an IBM reconstruction scheme

that provides consistency between pressure and velocity. The present study follows Fadlun et al. [19] with no explicit application of the Neumann pressure boundary condition.

4.2 Velocity Reconstruction Scheme

The discrete forcing with indirect boundary imposition was chosen for this flow solver because Cartesian meshes are well-suited for the GPU architecture and RANS will provide a mean velocity profile near the surface so a direct boundary imposition isn't necessary. In the discrete forcing IBM, a solid boundary is represented by adding a forcing term to the momentum equations given in Equation 2.2. The discretized form of the u-momentum equation is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = RHS_i + F_i, \tag{4.2}$$

where RHS_i includes the pressure gradient, convective, and diffusive terms, as well as SGS terms when addressing turbulent flows. Using the direct forcing method [19, 57], the velocity at the boundary can be prescribed as $u_i^{n+1} = V_i^{n+1}$, then the body force becomes

$$F_i = -RHS_i + \frac{V_i^{n+1} - u_i^n}{\Delta t}, \tag{4.3}$$

by solving Equation 4.2 for F_i. Using this approach, the body force can be taken into account implicitly by prescribing the velocity field or in other words, substituting Equation 4.3 into Equation 4.2. However, the complex geometry boundaries are usually not coincident with the Cartesian grid and reconstruction schemes (i.e., interpolation procedures) are required to impose the proper velocity boundary conditions

on grid points near the solid geometry. The steps in applying the IB method within the projection algorithm are summarized as follows

1. In the preprocessing stage, separate the Cartesian cells as solid, fluid, or immersed boundary (IB). Determine the necessary parameters for the velocity field reconstruction schemes.

2. Predict the velocity by solving the momentum equations as per the projection algorithm.

3. Set the solid Cartesian cells to zero and apply reconstruction scheme to IB nodes.

4. Solve the pressure Poisson equation by imposing the divergence free condition.

5. Correct the velocity field and set solid cells to zero.

For this application, the reconstruction scheme used will be similar to the IBM approach described in Gilmanov et al. [23–25] but will be extended to atmospheric flows with a rough surface following the approach described in [79]. No Neumann pressure boundary condition is applied in the IBM because studies by Fadlun et al. [19] have shown the error of omitting this step is very small (on the order of $10^3 - 10^4$).

The basic idea for the reconstruction scheme consists of linear interpolation along a line normal to the solid surface. This method is intended for stereolithography (STL) CAD geometry files where a surface is represented by an unstructured mesh of triangular elements. Each triangular element is defined by the coordinates of the vertices and a surface normal. Although the IB method can be implemented for analytical geometries (e.g., circle, sphere, etc.), its extension to arbitrarily complex geometries requires the development of a preprocessor to calculate the intersection

of the CAD geometry with the underlying Cartesian mesh. The preprocessor was jointly created by Kyle Felzien, a computer science undergraduate student at Boise State University, the author of this thesis, and Senocak [17].

The first stage of the preprocessor identifies all Cartesian points within the small search radius, ds_0, from the solid boundary. The value of the search radius is determined by finding the maximum distance from opposite corners of a cell (i.e. $ds_0 = \sqrt{dx^2 + dy^2 + dz^2}$). The position vectors of these points, r_{NB}, are compared to the position vector of the mth triangular element's centroid, $r_{m+1/2}$, until the following condition is satisfied

$$\min_{m=1,M} |r_{NB} - r_{m+1/2}| < ds_0. \tag{4.4}$$

Any point that satisfies the above condition is called a near-boundary node. Note that near-boundary nodes can be either internal or external to the solid boundary.

The next stage is to determine which of the near-boundary points are actually within the solid. For every near-boundary point, all triangular elements located within the search radius centered at the near-boundary node are identified. Examining the sign of the scalar product, $n_{m+1/2} \cdot (r_{nb} - r_{m+1/2})$, determines whether a near-boundary point is internal or external to the solid boundary. If $n_{m+1/2} \cdot (r_{nb} - r_{m+1/2}) > 0$ for *at least one* triangular element within the sphere from the point, then the Cartesian mesh point is external to the body and flagged as an IB node. If $n_{m+1/2} \cdot (r_{nb} - r_{m+1/2}) < 0$ for *all* triangular elements within the sphere from the point, the Cartesian mesh point is internal to the body and flagged as solid. All points residing in the solid can be identified by checking which points are between two nodes that are internal to the solid, or between a solid node and the computational domain's boundary in the case where the object extends beyond the boundaries.

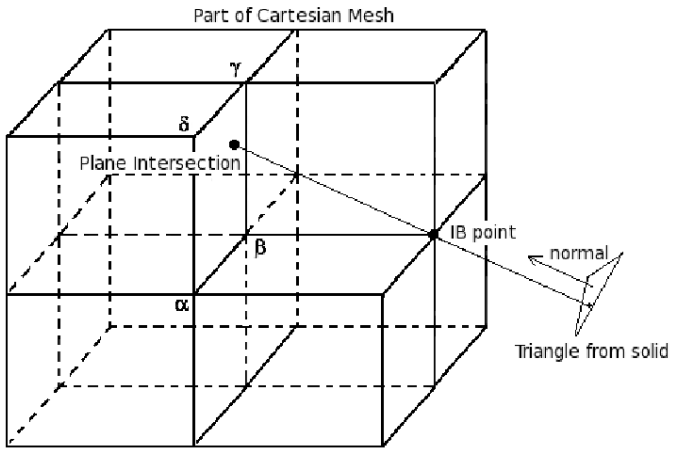

Figure 4.4: Sketch of the reconstruction scheme at an IB point, where a line is projected along the normal direction of the nearest triangular element into the fluid domain.

The final stage after the IB nodes are identified is to project a line starting at the IB node parallel to the surface normal of the nearest triangular element into the fluid domain as shown in Figure 4.4. This line is referred to as the IB line in this thesis. The IB line will eventually intersect a Cartesian cell face. The values of neighboring Cartesian grid points are then interpolated on to the IB line where the intersection with the cell face occurs. The velocity at the IB node is then found by an interpolation between the known boundary value at the surface and the values that are interpolated on the IB line. Multilinear interpolation of the points α, β, γ, and δ can be used to find the value of a quantity (e.g., velocity, pressure) on the IB line. For laminar flow conditions, the reconstruction of a quantity at the IB node is accomplished by linear interpolation between the point of intersection along the IB line and the value of the boundary condition at the surface.

4.3 Extending the Reconstruction Scheme to Atmospheric Boundary Layer Flows

The linear interpolation reconstruction scheme may also work well for turbulent flows if the grid resolution is fine enough to capture the viscous sublayer where $u^+ = y^+$ as suggested by the law of the wall. However, a clear viscous sublayer does not exist within the ABL due to the rough surface. A linear interpolation scheme could underestimate the surface stresses, because a logarithmic or power wind profile is typically observed in atmospheric flows. One also has to consider how the reconstruction scheme influences the SGS model. A consistency between the underlying assumptions in the turbulence model and the IBM reconstruction scheme is desirable to obtain satisfactory results. Therefore, a logarithmic reconstruction scheme [79] is proposed so when combined with Prandtl's mixing length model [94] (discussed later), the aforementioned consistency is maintained.

Atmospheric flows over complex terrain are influenced by roughness, atmospheric stability, and fluxes of sensible and latent heat and moisture, all of which play a major role in the observed wind profiles. Typically, the boundary conditions are imposed through stress and flux terms

$$\tau = \rho\overline{u'w'} \tag{4.5}$$

$$H = \rho C_p\overline{w'\Theta'} \tag{4.6}$$

$$E = \rho\overline{w'q'} \tag{4.7}$$

where τ is the turbulent stress at the surface, H and E are the fluxes for heat and moisture, respectively. u', w', Θ', q' represents the fluctuations of streamwise wind, vertical wind, potential temperature, and moisture, respectively. Direct implementa-

tion of these terms in Equations 4.5-4.7 within an immersed boundary method would be tedious and can complicate the IBM, which has historically become popular due to the simplicity of its implementation. Therefore, the reconstruction schemes should operate only on the primitive variables (e.g., u, v, w, Θ, q) to retain the simplicity of the IB method for atmospheric flows computations.

The reconstruction scheme proposed is the log-law reconstruction scheme [79] because of its consistency with the Monin-Obukhov similarity theory for neutral stability conditions that is also used in turbulence model assumptions. The logarithmic reconstruction is derived from the assumption that friction velocity remains constant in the direction normal to the wall. Therefore, similarity in the velocity profile is maintained at different distances from the wall. Dividing the rough surface log-law,

$$\frac{U}{u_\tau} = \frac{1}{\kappa} \ln\left(\frac{z}{z_0}\right), \tag{4.8}$$

where z_0 is the equivalent roughness height at the boundary results in

$$U_1 = U_2 \left[\frac{\ln\left(z_1/z_0\right)}{\ln\left(z_2/z_0\right)}\right], \tag{4.9}$$

where U_1 and U_2 are the magnitude of the velocity at locations shown in Figure 4.4, and z_1 and z_2 are the normal distances to the surface along the IB line as shown in Figure 4.4. The magnitude of the velocity must then be broken down into u, v and w components, which can be done using azimuth and elevation angles. For now, the assumption is made that the angles at location 1 and location 2 are the same and thus cancel out in Equation 4.9, meaning the relation can be directly applied when reconstructing each component. Also, only a neutrally stratified ABL is considered in this thesis but the scheme in Equation 4.9 can be extended to atmospheric conditions with stable and unstable stratification [79].

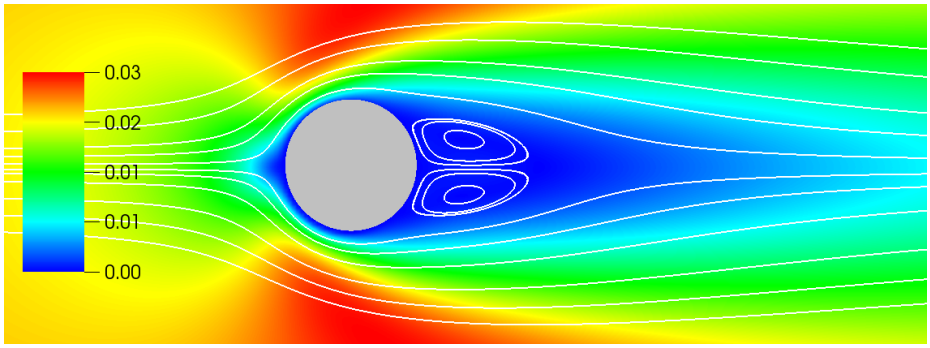

Figure 4.5: Streamlines of flow over a circular cylinder at Re = 20.

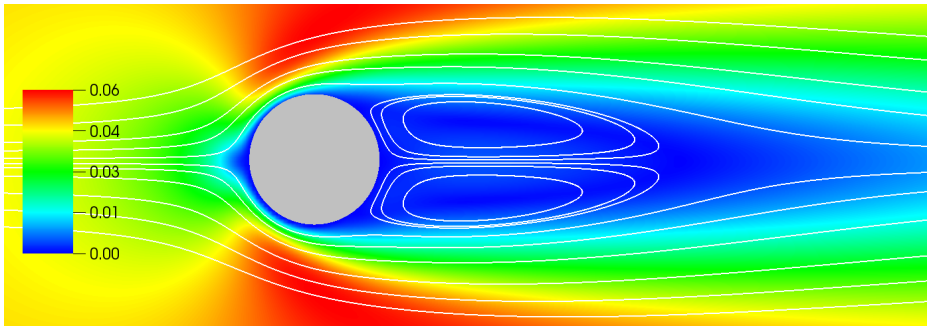

Figure 4.6: Streamlines of flow over a circular cylinder at Re = 40.

4.4 Immersed Boundary Method Validation

Laminar flow over a circular cylinder at Reynolds numbers of 20 and 40 were used to validate the immersed boundary implementation. The domain was $31D \times 24D$ in the x and z directions, respectively, where x is the direction of flow and D is the diameter of the cylinder. The center of the cylinder was placed at $10.5D$ in the x direction and at the halfway point in the z direction. The boundary conditions were an inlet and convective outlet in the streamwise direction with all other boundary condition set to freeslip. A linear reconstruction scheme was applied to this flow scenario. The resolution of the mesh was 1024×768 with 64 cells in the longitudinal direction. Approximately 30 cells were placed with the diameter of the cylinder.

The streamlines are presented in Figures 4.5 and 4.6. Figure 4.7 shows the u component of the centerline velocity in the wake compared to the computations of

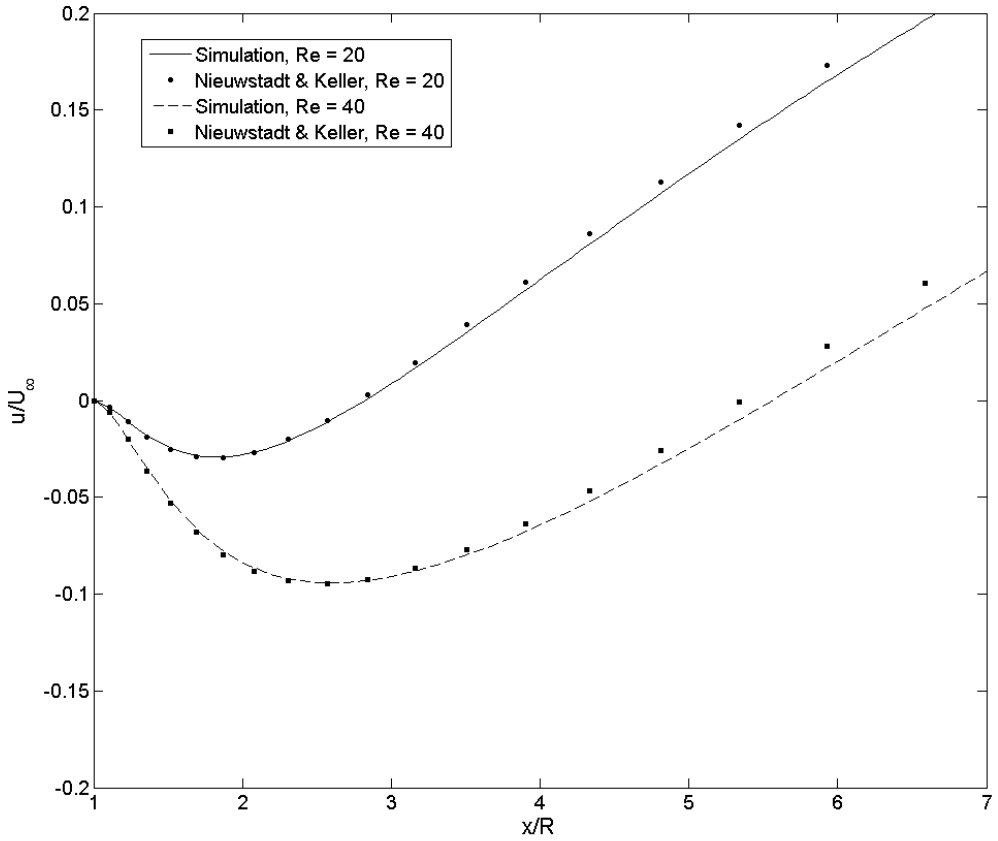

Figure 4.7: The u component of centerline velocity in the wake behind the circular cylinder for both Re = 20 and Re = 40 in a domain of $31D \times 24D$. Results are compared to Nieuwstadt and Keller [61].

Nieuwstadt and Keller [61] at both Reynolds numbers. The results in the near wake show very good agreement while the results deviate slightly from Nieuwstadt and Keller in the far wake in both cases. This deviation in the far wake mainly depends on the overall computational domain size and further extending the domain would alleviate these issues. The agreement in near wake of the cylinder is very good, and this test case serves as a good validation case for the IBM implementation.

CHAPTER 5

WIND FLOW OVER COMPLEX TERRAIN

In this chapter, the methods described in preceding chapters are extended to simulate wind flow over complex terrain. A literature review on recent wind forecasting techniques is also provided. The IBM extended to atmospheric flows as described in Section 4.3 along with a hybrid RAN/LES method will be implemented since fully resolving the ABL with LES is impractical. Results of wind flow over a complex terrain are presented as well.

5.1 Brief Survey of Wind Forecasting Over Complex Terrain

Balancing reserves is a challenge that becomes more difficult as more wind capacity is installed into an electrical grid. The difficulty arises from the intermittency of the wind. Wind forecasting is a tool used by the wind industry on a routine basis to alleviate this difficulty [81]. The accuracy of the forecasts is of the utmost importance as power generated from wind is directly proportional to wind velocity cubed. The forecasting takes place over different time horizons that require different simulation capabilities. Different time horizons include *very short-term*, *short-term*, *medium-term*, and *long-term* [82]. Very short-term covers a time less than a few minutes. Short-term describes a time horizon ranging from a half hour to 6 hours. Medium-

term forecasts generally range from 6 hours to a day while long-term forecasts are measured in days.

In the present survey, only short-term wind forecasting techniques are reviewed. The most basic short-term wind forecasting model that often serves as a benchmark for new models is a persistence model [27]. A persistence model uses the assumption that the future value will equal the current value [81]. Persistence models typically work well within an hour or two but the quality of the results rapidly degrades when the time horizon is increased.

There are two additional wind forecasting approaches that typically produce better results than persistence models: the physical approach and the statistical approach [82]. The physical approach is based on using numerical weather prediction (NWP) models. NWP solves the complex mathematical models describing wind flow over complex terrain, temperature, and pressure. A major challenge of this approach is the uncertainty when using the mesoscale wind speeds provided by a weather service on the microscale spatial domain of the wind farm site [45]. NWP requires significant computational resources and are often limited to medium- and long-term forecasts because of the difficulty of providing the mesoscale weather information. The best results are obtained in neutrally stratified weather conditions [82].

NWP is difficult because any slight variation in the initial conditions dramatically changes the outcome due to the dynamic nature of the weather. An alternative that is not often employed because of large computational requirements is the ensemble forecasting approach [83]. This approach runs several NWP simulations using slightly different initial conditions to get a frequency distribution of probable events. Each simulation is called an ensemble member. The number of ensemble members is often limited by available computational resources, although when used, the method demonstrates a strong potential in wind forecasting [26]. An approach similar to

ensemble forecasting is to take a combination of different NWP implementations with the same initial conditions to produce a frequency distribution, but it still requires the same resources as ensemble forecasting [45].

The statistical approach to wind forecasting is the use of time series using the auto-regressive moving average (ARMA). The basic idea of ARMA is to estimate a future value of an individual time series as a linear combination of previously observed values [27]. There are several derivatives to the ARMA approach [10, 60, 82] including the Box-Jenkins approach of autoregressive integrated moving average (ARIMA), the seasonally adjusted ARIMA (SARIMA), and the approach of fractional-ARIMA (f-ARIMA), to name a few. There are several other time series techniques that have been employed and surveys of these are given in Bhaskar et al. [10] and Soman et al. [82].

A popular area of wind forecasting research in recent years has been in artificial neural networks (ANN) [10]. The ANN technique is based on artificial intelligence where the program mimics the human learning process to discover relationships between the variables in a system [31]. ANN does not require explicit declarations of mathematical expressions and thus takes less development and computational time than ARMA models [10]. Depending on the implementation and application, ANN models can produce better or worse results than ARMA models as shown by comparing the conclusions of Gomes and Castro [27] and Soman et al. [82].

Hybridization of ANN with physical and statistical approaches is common as well [10]. Several techniques exist, too many to fit into this literature review. However, the basic idea is to use the physical and statistical approaches to train the neural network. In general, the results are improved but the implementation increases in complexity. Evolutionary optimization algorithms such as the genetic algorithms or particle swarm optimization can also be used to train ANN.

A physical approach is pursued in this study. Statistical approaches provide good results for short time horizons but the accuracy of the results degrade as the time horizon is extended. Typically, physical approaches are useful for medium- and long-term forecasts ($>$ 6 hours) because of the amount of computations and difficulty obtaining information in short time horizons [82]. Hence, the simulation has been developed for GPU clusters for their potential to accelerate the computations to predict in the short-term while still maintaining the applicability to longer time horizons. Also, time series only provide mean wind velocity predictions. Resolving the wind in the ABL can provide predictions on the mean and random components of wind velocity as well as other quantities such as pressure and temperature.

5.2 IBM in Atmospheric Flows

IBM in meteorological applications is not without precedent. Senocak et al. [79] extended the discrete forcing approach to atmospheric boundary layer simulations over a flat terrain by adopting the same length scale assumptions in the turbulence model and reconstruction schemes. Lundquist et al. [52] presented a 2D implementation of the ghost-cell approach within the Weather Research and Forecasting Model (WRF) for analytical geometries without any consideration for turbulence modeling and turbulent stresses at the immersed surfaces. Jafari et al. [38] performed a RANS simulation of wind flow over complex terrain also using a ghost-cell approach.

To the best of the author's knowledge, no study to date has extended an IBM approach to atmospheric flows and coupled it with either a RANS or LES approach for turbulence with provisions for addressing the issues of atmospheric stability, representing turbulent stresses on immersed surfaces, and accounting for land-surface fluxes of heat and moisture. The goal of this thesis is to implement the core components for a short-term wind forecasting simulation that address the coupling of IBM with

turbulence modeling and lay the foundation for future incorporation of models that maintain atmospheric stability and account for land-surface fluxes. One of the first steps is to couple the IBM with a logarithmic reconstruction schemes described in Chapter 4 with a hybrid RANS/LES turbulence modeling technique.

5.3 Hybrid RANS/LES

The LES technique is pursued because of the information it provides on turbulent fluctuations and has potential to capture the details of highly separated flows that are found in wind flow over complex terrain. The Lagrangian dynamic Smagorinsky model was chosen for the SGS model because homogeneous directions of turbulence are not a requirement. However, LES requires resolution in the viscous sublayer and with Reynolds numbers being on the order of 10^7 along with the surface being rough, fully resolving the boundary layer is impractical. This is the motivation for hybridizing LES with RANS at high Reynolds numbers to form a hybrid RANS/LES technique. RANS models the contribution of the near-wall eddies and acts as a sort of wall model for the LES. Hybrid RANS/LES has been applied to atmospheric scenarios with success in Senocak et al. [78], Bechmann et al. [5] and Bechmann and Sørensen [4], to name a few. There are several methods to hybridize LES and RANS [75], but the method chosen for this simulation is the smooth length scale transition suggested by Senocak et al. [78]

$$\nu_t = \left[\left(1 - \exp\left(\frac{-z}{h} \right) \right)^2 (C_S \Delta)^2 + \exp\left(\frac{-z}{h} \right)^2 (\kappa z)^2 \right] |S|, \tag{5.1}$$

where κ is the von Kármán constant and h is the RANS/LES transition height. Equation 5.1 blends the length scales produced from the dynamic SGS model with a mixing length RANS model. The logarithmic reconstruction in the IBM main-

tains consistency with the mixing length turbulence model. The transition height is determined by the following relationship based on the Nyquist theorem,

$$\gamma = \frac{h}{2\Delta}, \tag{5.2}$$

where Δ is the base filter width and γ is a parameter chosen depending on the flow. The value of γ dictates how many cells near the wall are modeled by RANS. Ensuring that h is large enough to encompass at least one full cell is of the utmost importance, particularly when the aspect ratio of a cell is larger in the wall-normal direction than in the lateral directions.

5.4 Evaluation of Hybrid RANS/LES

The consistent coupling of the hybrid RANS/LES and IBM implementations were evaluated using periodic turbulent channel flow (see Chapter 3). The friction Reynolds number was set to 1000 to ensure a sufficiently large log-law region. The dimensions of the channel were also increased to $(8\pi\delta, 3\pi\delta, 2\delta)$ in (x, y, z). To test the IBM, the walls were placed such that they were not coincident with any of the grid points, but were still parallel to the x-y plane. Thirty wall units separated a wall to the first u component. The computational grid was chosen to be $512 \times 192 \times 64$. No grid stretching was applied. Note the z dimension is actually larger than the height of the channel to maintain the thirty wall unit separation. The same forcing, initial conditions and time sampling intervals used in the LES validation were applied. A length scale transition height for the hybrid RANS/LES was chosen to be 4Δ or $\gamma = 2$ in Equation 5.2. The simulations were performed on four NVIDIA Tesla C2070 GPUs connected by an Infiniband QDR interconnect. The turn around time was twenty hours.

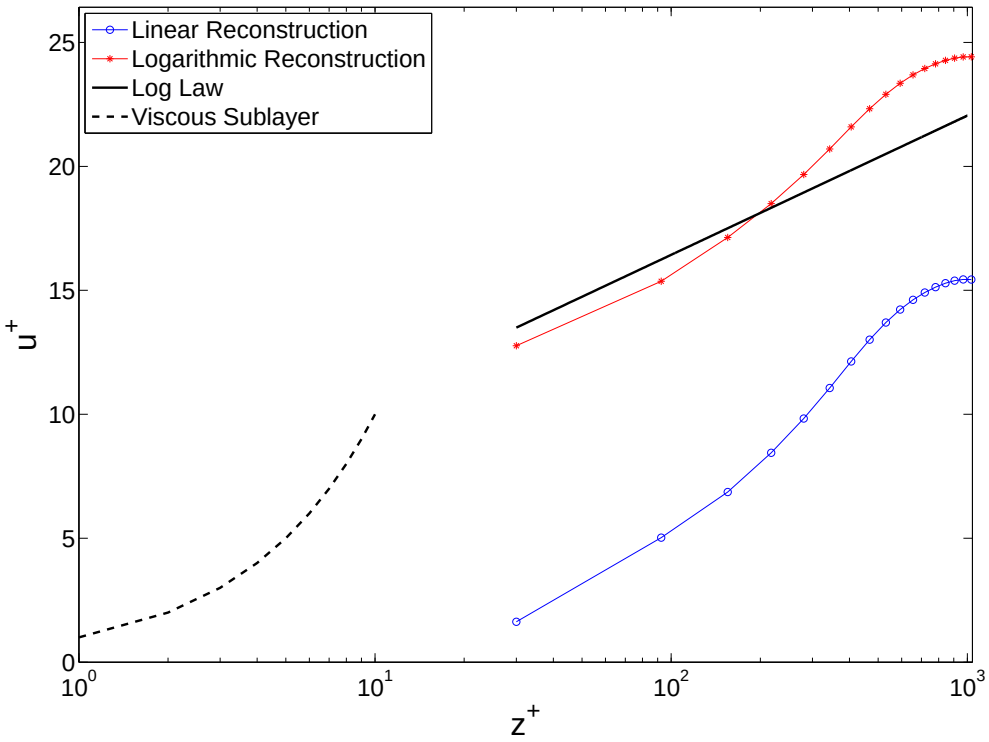

Figure 5.1: Comparison of mean streamwise velocity profile for a turbulent channel flow at $\mathrm{Re}_\tau = 1000$ using hybrid RANS/LES technique. Grid size was 512×192×64 and the separation between wall and first u-component was 30 wall units. IBM reconstruction schemes: $*$, logarithmic; $\circ$, linear

Figure 5.1 is a comparison of different IBM reconstruction schemes to the theoretical law of the wall. A logarithmic reconstruction scheme (Equation 4.9) was developed using smooth wall log-law given as,

$$\frac{U}{u_\tau} = \frac{1}{\kappa} \ln\left(\frac{y u_\tau}{\nu}\right) + B, \tag{5.3}$$

where κ is the von Kármán constant and B is a constant. The value of the von Kármán constant is 0.41 and B is 5.2 [73]. The logarithmic reconstruction performs much better than the linear reconstruction. The linear reconstruction severely underpredicts the velocity. On the other hand, the logarithmic reconstruction provides more reasonable result consistent with the law of the wall. The underprediction of velocity with linear reconstruction is not surprising, since the separation between the wall and the first calculated component is 30 wall units. At this distance, linear relations present in the viscous sublayer no longer hold and forcing a linear relationship does not provide the proper shear stress near the wall. The distance is in the region where the log-law holds, and therefore logarithmic reconstruction agrees well with theory.

These results demonstrate the importance of consistency between the IBM and turbulence modeling. The mixing length model used in the present study is derived from the log-law. When applying a linear reconstruction scheme in a region where the log-law holds, the velocity was underpredicted because of incorrect shear stress. On the other hand, applying a logarithmic reconstruction produced results agreeing well with the theory. Therefore, ensuring that underlying theories for the IBM and hybrid RANS/LES are consistent is essential for this study to reduce the error in future wind forecasts.

5.5 Bolund Hill Performance Tests

Using the implementation described in Sections 5.2 and 5.3, simulations were performed on Bolund Hill, a 12-meter-high isolated coastal hill located in Roskilde Fjord, Denmark. Because of its isolation and small shape, it has been the subject of several studies, both experimental [6] and computational [3, 38].

Figure 5.2 is a surface rendering of the Bolund Hill stereolithography (STL) file used in this paper. The feature that makes the Bolund Hill case challenging to simulate is the steep vertical escarpment. Figure 5.3 shows a slice of the Cartesian mesh used superimposed on the STL at the escarpment. In this solver, the x and y directions correspond to the lateral directions and z to the vertical direction, with the x direction being perpendicular to the escarpment in Figure 5.2. The computational Cartesian mesh used in this paper was $256 \times 192 \times 128$ in the x, y, and z directions, respectively, with a lateral resolution of 4 m and vertical resolution of 1 m. A no-slip condition was imposed at the terrain surface with a free-slip condition at the top of the domain and periodic lateral boundary conditions. The equivalent roughness height for the logarithmic reconstruction (Equation 4.9) was 0.0003 m for the water and 0.015 m for the hill as was suggested in Berg et al. [6]. A height-independent, constant forcing of 0.001 was applied.

In the Bolund Hill Experiment, masts with sonic anemometers were set up along in the vicinity of the line in Figure 5.2. This line is referred to as the 270° line in the Bolund Experiment when the escarpment is the windward side. The simulation was compared to data of wind 5 m above the ground along the 270° line. Figure 5.4 shows the velocity sampled over the hill normalized to a value sampled from the reference mast [3, 6]. The results are not very satisfactory and deviate from the experimental data but several factors may be responsible for the erroneous results.

One factor is that periodic boundary conditions imply periodically placed islands

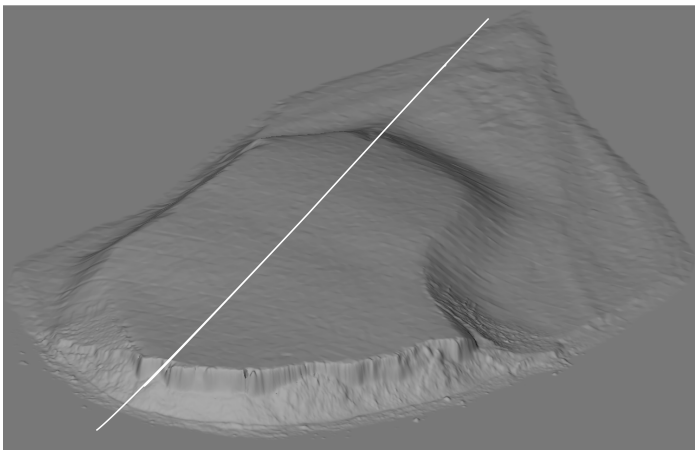

Figure 5.2: The surface created by the STL geometry of Bolund Hill used in this paper. The wind flow direction is parallel to the superimposed line with the windward side being the escarpment.

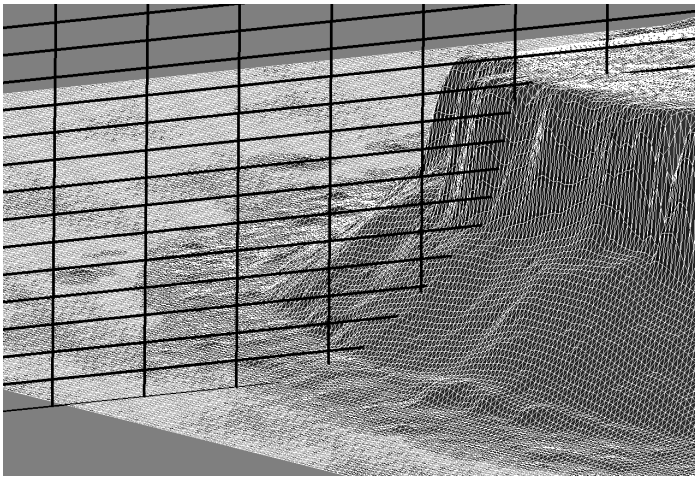

Figure 5.3: Closeup of a Cartesian mesh slice in the x-z plane superimposed on the Bolund Hill STL. One cell has dimensions of 4 m in the x and 1 m in the z.

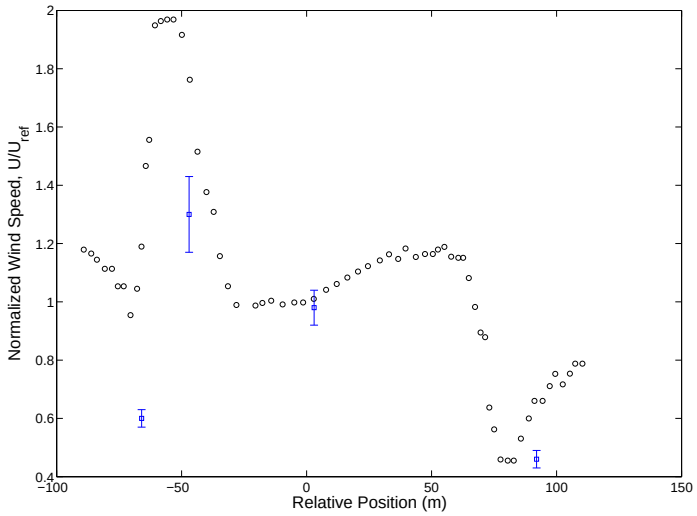

Figure 5.4: Wind speedup 5 m above ground along the 270° line. The experimental data is found in Berg et al. [6].

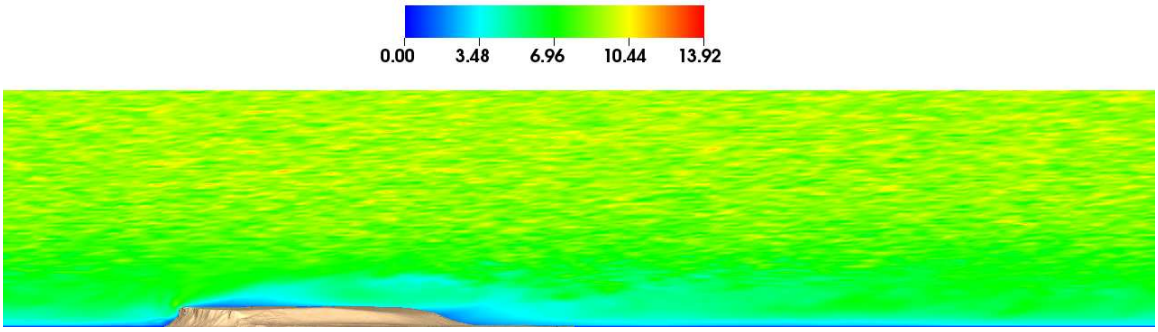

Figure 5.5: Instantaneous wind velocity along the 270° line. The existence of turbulent flow structures and vortex shedding in the wake demonstrate that the LES is able to generate eddies well but requires better boundary layer shear stresses to compute wind speed correctly.

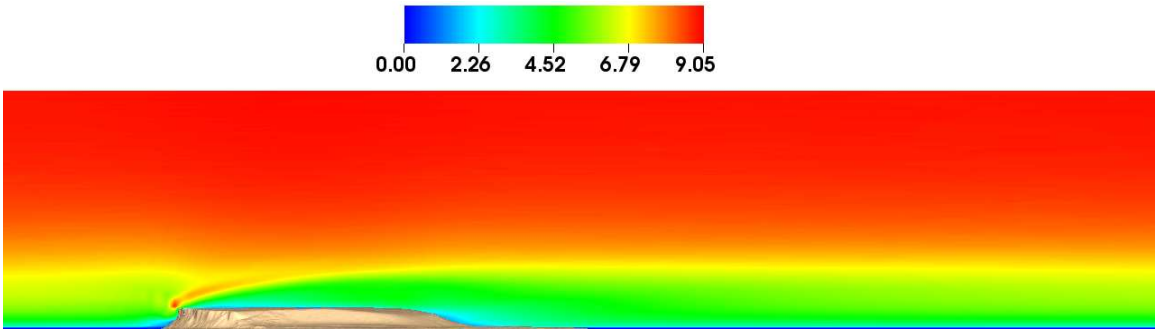

Figure 5.6: Ensemble-averaged wind velocity along the 270° line. The acceleration at the escarpment and the evidence of a wake are encouraging results.

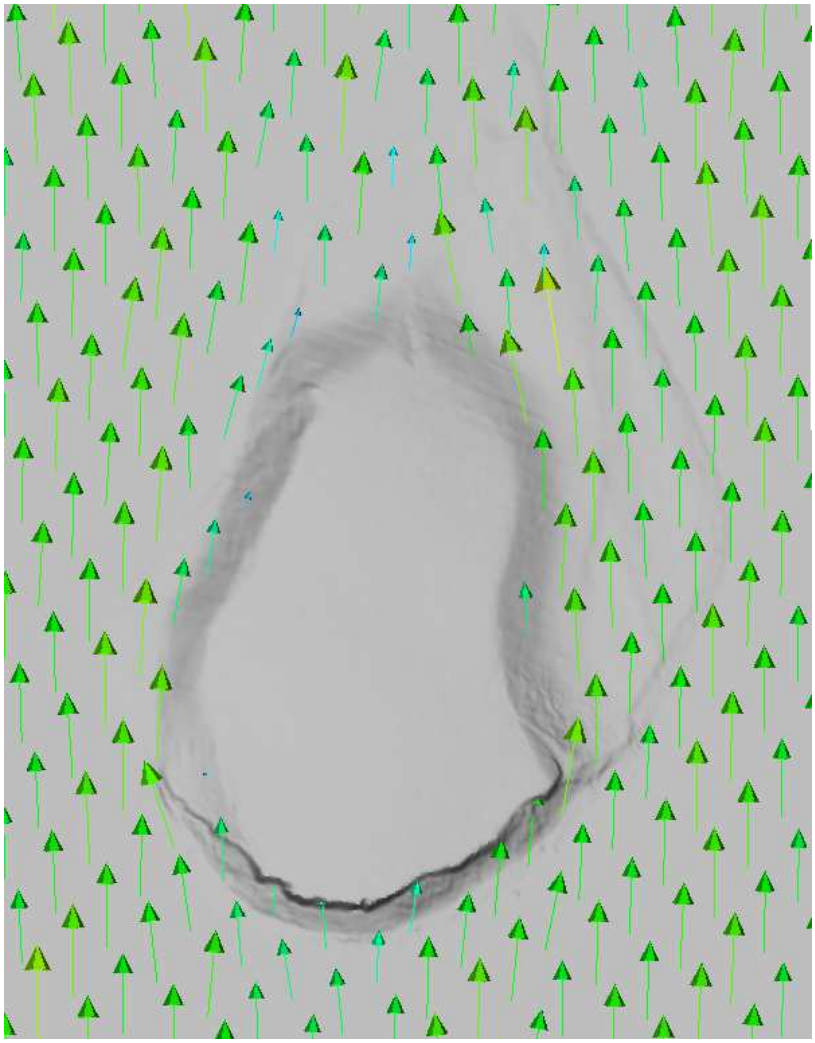

Figure 5.7: Instantaneous wind velocity vectors approximately 7 m from base of hill indicating the present flow solver does capture some of the effects of the complex terrain.

and may incorrectly influence the upstream velocity. While the influence might be minor because of the hill having a low profile, replacing the periodic lateral boundary conditions with open lateral boundary conditions would be worthwhile to pursue. Another factor is the spatial resolution, particularly in the vicinity of the surface. As shown in Figure 5.3, only 4 to 6 cells are currently resolving the vertical escarpment. When looking back to the laminar cylinder case, approximately 30 cells were needed to obtain good results. Therefore, grid refinement is necessary. Also, issues with the hybrid RANS/LES and IBM when applied to complex geometry may not have arisen when simple benchmark cases were performed. Further evaluation of these techniques is required. However, Figure 5.4 shows that the acceleration and deceleration of the wind is captured, but the errors in the magnitudes can be attributed errors in the surface stresses.

While unsatisfactory results were obtained in the wind speeds at several locations, the flow solver is able to capture the influence of the complex terrain qualitatively. Figure 5.5 is a visualization of the instantaneous velocity along the 270° line. The existence of turbulent flow structures and vortex shedding demonstrate that the turbulence modeling performs well but requires better boundary layer shear stresses to accurately model atmospheric turbulence.

Figure 5.6 depicts the time-averaged wind velocity values along the 270° line. The acceleration at the escarpment looks good from a qualitative perspective when compared to results from Jafari et al. [38]. The wake behind the hill compares reasonably well also compared to Jafari et al., again from a qualitative standpoint.

Figure 5.7 is a vector plot of wind velocity at approximately 7 m from the base of the hill. The flow simulation is capturing the influence of the complex terrain on the flow reasonably well as is evident with the vectors conforming around the hill. However, a few errors may exist because of the large vectors in the upper right of

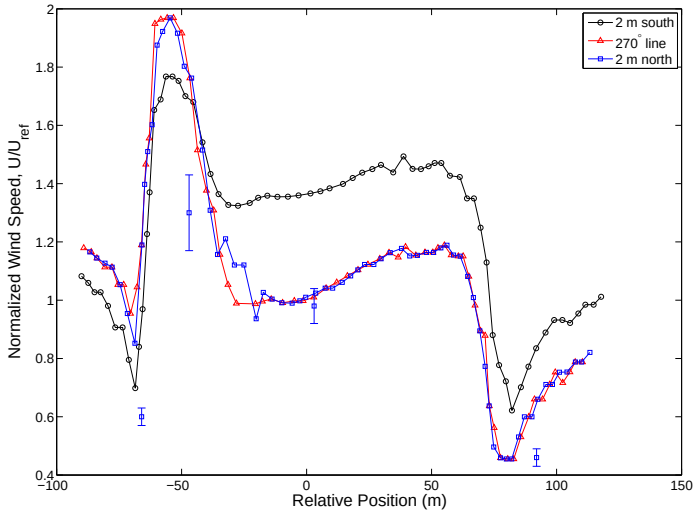

Figure 5.8: Comparison of normalized mean wind velocity along the 270° line (∗), 2 m south of the 270° line (∘) and 2 m north of the 270° line (□) reveals that the results are quite sensitive to the location meaning Bolund Hill is not an ideal case for simulation evaluation.

Figure 5.7, although this may be sudden acceleration due to turbulence. This may be an issue that did not arise when testing was performed on simple benchmark cases and prompts further evaluation of the IBM and hybrid RANS/LES.

Figure 5.8 provides another reason for further evaluation of the simulation. This figure compares numerical results along the 270° line against results from 2 m north and south of the same line. The results do not deviate greatly 2 m north of the 270° line. However, sampling 2 m south greatly deviates from the other two demonstrating that the results are highly sensitive to the location, which implies that simulation is also highly sensitive to the complex geometry of this particular case. The sensitivity suggests that Bolund Hill is too difficult a complex terrain to be an ideal case for evaluating a simulation. Other simpler test cases should be used alongside this one to gain a better understanding of the model.

Regarding the computational performance of the current wind solver, the 256

$\times$ 192 $\times$ 128 case performed at 20% of real-time. That is, for every 5 seconds of computational wall time, 1 second physical time was resolved. The computing platform used consisted of four NVIDIA Tesla C2070 GPUs connected by PCI Express 2.0 $\times$ 16 buses and a quad data rate (QDR) Infiniband interconnect. Mesh sizes of 192 $\times$ 128 $\times$ 64 and 128 $\times$ 128 $\times$ 33 with the same spatial resolutions were also attempted although the results are not shown in this thesis. The 192 $\times$ 128 $\times$ 64 mesh performed at approximately 70% of real-time on the same computing platform but with only two GPUs. Using four GPUs resulted in approximately the same performance but a large majority of the GPU remained idle for this problem size.

The 128 $\times$ 128 $\times$ 33 on average performed better than 50% faster than real-time. However, this one was tested on two different platforms. The first was one of the Tesla GPUs mentioned above, which executed at twice that of real-time. The other platform consisted of two NVIDIA GeForce GTX 470 GPUs with a single data rate (SDR) Infiniband interconnect, which gave a performance of about 1.5 that of real-time. This reinforces the point that the inexpensive GeForce gaming GPUs can provide performance similar to that of the more expensive Tesla GPUs and can allow researchers an inexpensive tool for problems involving relatively small data sets. Overall, the computational performance to realize forecasting is very encouraging.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

6.1 Conclusions

During the course of this study, the LES technique with a Lagrangian dynamic SGS model and the immersed boundary method were successfully implemented on GPU clusters. These two methods provide the foundation for a short-term wind forecasting application. LES of turbulent channel flows provides results that agree well with DNS data. These simulations, which typically take a week or more to complete, were performed in under two days. Simulation of laminar flow over a circular cylinder using the IBM produced results that agree well with benchmark data. No computational performance analysis or comparisons were made because no turn around times from CPU-based implementations were available. However, the turn around times were still reasonable even with the excessively large computational meshes. The coupling of LES and IBM was also successful in the test case of turbulent channel flow, which completed in a matter of a few hours. Thus, these implementations have reinforced the impact that the GPU can have on the field of computational fluid dynamics.

LES is no simple task when applied to complex terrain. This became apparent in this study when LES was applied to simulate the Bolund Hill Experiment. The wind speed results do not agree well with the experimental data. The results were also

very sensitive to location because of the complexity of Bolund Hill. However, other attempts by other researchers using LES on Bolund Hill have encountered similar problems [3], indicating that further in-depth study is needed. Difficulty modeling wind over Bolund Hill demonstrates that LES applications involving complex terrain suffer from poor representation of surface stress, inadequate spatial resolution, and turbulence modeling in the vicinity of the surface.

To pursue possible insights into atmospheric turbulence, the goal of this thesis was to extend a baseline incompressible flow solver to model neutrally stratified wind over complex terrain at the microscale to investigate if a coupling of hybrid RANS/LES and IBM would provide a foundation for a wind forecasting capability. While results did not agree well with experimental data, the potential to reach the goal has been demonstrated. The work in this thesis can be considered a first attempt as several modeling issues such as inflow conditions, mesh refinement near the vicinity of the surface, and modeling fluxes of heat and moisture were not addressed in this study. Resolving these modeling issues is expected to significantly improve the results. At this point in the development process, the approach undertaken in this study cannot be dismissed because not all of the necessary components are in place yet.

The computational performance of the present flow solver remains encouraging. The fact that real-time calculations can be sustained on GPU clusters opens doors for other wind and weather forecasting applications to be ported to GPU computing platforms. Concerning the present application, several numerical techniques exist that could potentially reduce the number of computations and further increase performance. Some of these are discussed in the next section. Also, programming optimizations to the current implementation of algorithms may be possible, particularly with new GPU technologies being released [64]. The real-time performance metrics mentioned here may increase with these possible improvements and a short-term wind

forecasting application may come to fruition.

6.2 Future Directions

This thesis laid the foundation for developing a short-term, microscale wind forecasting capability. Initial attempts at simulating wind over complex terrain have shown that further research and testing against different terrains is necessary. In particular, the calculation of surface stresses, the modeling of turbulence in the vicinity of the surface, and the critical issue of inflow conditions need to be addressed in the flow solver. First, further evaluation of both the hybrid RANS/LES technique and the IBM are needed with geometries more complex than the simple benchmark cases used in this study but simpler than the Bolund Hill. Other complex geometries may bring to light issues that were not evident with simple benchmark cases.

Next, periodic lateral boundary conditions should be replaced with open lateral boundary conditions. Periodic boundary conditions imply the existence of periodically placed islands and is not the case since the wind comes from an open body of water. These boundary conditions may be acceptable for the Bolund Hill case because of its relatively low height. However, other complex terrain cases are much higher and periodic boundary conditions will not be suitable for such cases. The proper replacement of the periodic boundary conditions is an on-going research topic and is one of the major challenges of LES [39, 75]. One possibility is to sample values from a precursor simulation of flow over flat terrain using periodic boundary conditions. The sampled values would provide the inflow conditions for the simulation with complex terrain [4]. Other possibilities to investigate are the stochastic and deterministic inflow conditions summarized and compared in Keating et al. [39] and Sagaut [75]. A popular deterministic approach described in both [39] and [75] is the recycling and rescaling method of Lund et al. [51] that rescales the outer and inner layers

of the downstream velocity profile separately and imposes the rescaled data as the inlet. This approach does require the similarity laws to remain intact throughout the domain, which may not be possible with complex terrain. A stochastic approach would be better suited, the basic idea being to prescribe a mean velocity profile at the inlet and superimpose a fluctuating component. The disadvantages is finding the correct mathematical description for the fluctuations to introduce the proper flow structures and turbulent kinetic energy. Which approach to use and how well it applies to complex terrain flow scenarios will be the focus of future work.

The interface region between the LES and RANS regions is also a possible area of future investigation, particularly at the inflow boundary conditions. This directly ties into the inflow condition challenge because the different length scales between RANS and LES regions would likely require different inflow conditions specified for each region [40].

Integrating an adaptive mesh refinement (AMR) [7, 8] is a future direction for this flow solver. The basic idea is to refine the mesh in areas of high interest (e.g., escarpment of Bolund Hill) but coarsen the mesh in areas of large-scale eddies (e.g., flat terrain). Currently, the directionally-uniform Cartesian mesh often provides more computational nodes than are necessary, which is particularly evident in the laminar flow over a circular cylinder validation study. AMR would improve the simulations by providing a more detailed description of flow around complex terrain and reducing the memory requirements.

To help improve the performance and achieve forecasting, the current Adams-Bashforth time advancement scheme could be replaced by a Runge-Kutta time advancement scheme [46]. The Runge-Kutta scheme would allow for a larger Courant-Friedrichs-Lewy (CFL) number. The central difference discretization of the viscous term could also be replaced by an implicit Crank-Nicolson scheme to eliminate the

viscous stability limit. The end result would be larger time steps, but with the extra cost of solving tri-diagonal matrices.

As of now, surface fluxes of heat and moisture are not accounted for in the model. These fluxes could have an effect on the wind in the ABL [20]. Therefore, future work will also require the incorporation of flux models [50, 87]. Accounting for the influence of surface fluxes is expected to reduce the uncertainty in wind forecasts.

Last but not least, the applicable time horizon of the wind forecasting capability after the aforementioned issues are addressed needs to be determined. The wind solver is meant to be used in the short-term forecasting time horizon (less than six hours). However, without the proper initial conditions, the flow may take longer to develop than the targeted time horizon and the forecasts would be incorrect. A possible idea would be to couple this microscale forecasting simulation with a mesoscale forecasting simulation such as the Weather Research and Forecasting (WRF) model. WRF could provide realistic initial and boundary conditions to this wind forecasting capability and reduce the development time of the wind flow. This wind forecasting capability could also improve the mesoscale predictions of WRF by providing better parameterizations for the boundary conditions. An endeavor such as this would be a worthwhile future investigation.

REFERENCES

[1] E. Balaras. Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations. *Computers & Fluids*, 33(3):375 – 404, 2004.

[2] J. Bardina, J. H. Ferziger, and W. C. Reynolds. Improved turbulence models based on large eddy simulation of homogeneous, incompressible, turbulent flows. *Technical Report No. TF-19, Department of Mechanical Engineering, Stanford University, Stanford, CA*, 1983.

[3] A. Bechmann, N. Sørensen, J. Berg, J. Mann, and P.-E. Réhoré. The Bolund experiment, part II: Blind comparison of microscale flow models. *Boundary-Layer Meteorology*, 141:245–271, 2011. 10.1007/s10546-011-9637-x.

[4] A. Bechmann and N. N. Sørensen. Hybrid RANS/LES method for wind flow over complex terrain. *Wind Energy*, 13(1):36–50, 2010.

[5] A. Bechmann, N. N. Sørensen, J. Johansen, S. Vinther, B. S. Nielsen, and P. Botha. Hybrid RANS/LES method for high reynolds numbers, applied to atmospheric flow over complex terrain. *Journal of Physics: Conference Series*, 75(1):012054, 2007.

[6] J. Berg, J. Mann, A. Bechmann, M. Courtney, and H. Jørgensen. The Bolund experiment, part I: Flow over a steep, three-dimensional hill. *Boundary-Layer Meteorology*, 141:219–243, 2011. 10.1007/s10546-011-9636-y.

[7] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64 – 84, 1989.

[8] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484 – 512, 1984.

[9] R. P. Beyer and R. J. Leveque. Analysis of a one-dimensional model for the immersed boundary method. *SIAM Journal on Numerical Analysis*, 29(2):pp. 332–364, 1992.

[10] M. Bhaskar, A. Jain, and N. Venkata Srinath. Wind speed forecasting: Present status. In *Power System Technology (POWERCON), 2010 International Conference on*, pages 1 –6, oct. 2010.

[11] A. Botterud, J. Wang, V. Miranda, and R. J. Bessa. Wind power forecasting in U.S. electricity markets. *The Electricity Journal*, 23(3):71 – 82, 2010.

[12] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[13] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: Stream computing on graphics hardware. In *ACM Transactions on Graphics*, volume 23, pages 777–786, New York, NY, USA, 2004. ACM Press.

[14] W. Cheng and C.-H. Liu. Large-eddy simulation of flow and pollutant transports in and above two-dimensional idealized street canyons. *Boundary-Layer Meteorology*, 139:411–437, 2011. 10.1007/s10546-010-9584-y.

[15] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):pp. 745–762, 1968.

[16] J. W. Deardorff. A numerical study of three-dimensional turbulent channel flow at large reynolds numbers. *Journal of Fluid Mechanics*, 41(02):453–480, 1970.

[17] R. DeLeon, K. Felzien, and I. Senocak. Toward a gpu-accelerated immersed boundary method for wind forecasting over complex terrain. To appear in ASME Fluids Engineering Division Summer Meeting 2012, Paper number 72145, 2012.

[18] E.R. Van Driest. On turbulent flow near a wall. *Journal of Aeronautical Sciences*, 23(11):1007–1011, 1956.

[19] E.A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, 161(1):35 – 60, 2000.

[20] J.R. Garratt. *The Atmospheric Boundary Layer*. Cambridge University Press, 1994.

[21] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 1991.

[22] S. Ghosal, T.S. Lund, P. Moin, and K. Akselvoll. A dynamic localization model for large-eddy simulation of turbulent flow. *Journal of Fluid Mechanics*, 286:229–255, 1995.

[23] A. Gilmanov and S. Acharya. A hybrid immersed boundary and material point method for simulating 3D fluidstructure interaction problems. *International Journal for Numerical Methods in Fluids*, 56(12):2151–2177, 2008.

[24] A. Gilmanov and F. Sotiropoulos. A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. *Journal of Computational Physics*, 207(2):457 – 492, 2005.

[25] A. Gilmanov, F. Sotiropoulos, and E. Balaras. A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids. *Journal of Computational Physics*, 191(2):660 – 669, 2003.

[26] T. Gneiting and A. E. Raftery. Weather forecasting with ensemble methods. *Science*, 310(248):248–249, Oct 2005.

[27] P. Gomes and R. Castro. Wind speed and wind power forecasting using statistical models: Autoregressive moving average (ARMA) and artificial neural networks (ANN). *International Journal of Sustainable Energy Development*, 1(1):36–45, 2012.

[28] A. A. Gowardhan. *Towards Understanding Flow and Dispersion in Urban Areas Using Numerical Tools*. PhD thesis, University of Utah, 2008.

[29] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[30] W. Gropp, R. Thakur, and E. Lusk. *Using MPI-2: Advanced Features of the Message Passing Interface*. MIT Press, 1999.

[31] Mohamad H. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge, MA, USA, 1st edition, 1995.

[32] G. Iaccarino and R. Verzicco. Immersed boundary technique for turbulent flow simulations. *Applied Mechanics Review*, 56:331–347, 2003.

[33] T. Ikeno and T. Kajishima. Finite-difference immersed boundary method consistent with wall conditions for incompressible turbulent flow simulations. *Journal of Computational Physics*, 226(2):1485 – 1508, 2007.

[34] D. Jacobsen. Methods for multilevel parallelism on gpu clusters: Application to a multigrid accelerated navier-stokes solver. Master's thesis, Boise State University, 2010.

[35] D. A. Jacobsen and I. Senocak. A full-depth amalgamated parallel 3D geometric multigrid solver for GPU clusters. In *49th AIAA Aerospace Science Meeting*, 2011.

[36] D. A. Jacobsen and I. Senocak. Scalability of incompressible flow computations on multi-GPU clusters using dual-level and tri-level parallelism. In *49th AIAA Aerospace Science Meeting*, jan 2011.

[37] D. A. Jacobsen, J. C. Thibault, and I. Senocak. An MPI-CUDA implementation for massively parallel incompressible flow computations on multi-GPU clusters. In *49th AIAA Aerospace Science Meeting*, 2010.

[38] S. Jafari, N. Chokani, and R. S. Abhari. An immersed boundary method for simulation of wind flow over complex terrain. *Journal of Solar Energy Engineering*, 134(1):011006, 2012.

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

[40] A. Keating, G. De Prisco, and U. Piomelli. Interface conditions for hybrid RANS/LES calculations. *International Journal of Heat and Fluid Flow*, 22(5):777–788, 2006.

[41] J. Kim, D. Kim, and H. Choi. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, 171(1):132 – 150, 2001.

[42] D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Elsevier Inc., 2010.

[43] R. H. Kraichnan. Eddy viscosity in two and three dimensions. *Journal of Atmospheric Science*, 33:1521–36, 1976.

[44] M. T. Landahl and E. Mollo-Christensen. *Turbulence and Random Processes in Fluid Mechanics*. Cambridge University Press, 2 edition, 1992.

[45] M. Lange and U. Focken. New developments in wind energy forecasting. In *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pages 1 –8, July 2008.

[46] H. Le and P. Moin. An improvement of fractional step methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 92(2):369 – 379, 1991.

[47] M. Lesieur and O. Métais. New trends in large-eddy simulations of turbulence. *Annual Review of Fluid Mechanics*, 28:45–82, 1996.

[48] M. Lesieur, O. Métais, and P. Comte. *Large-Eddy Simulations of Turbulence*. Cambridge University Press, 2005.

[49] D.K. Lilly. A proposed modification of the Germano subgrid-scale closure method. *Physics of Fluids A: Fluid Dynamics*, 4(3):633–35, 1992.

[50] J.-F. Louis. A parametric model of vertical eddy fluxes in the atmosphere. *Boundary-Layer Meteorology*, 17:187–202, 1979. 10.1007/BF00117978.

[51] T. S. Lund, X. Wu, and K. D. Squires. Generation of inflow data for spatially-developing boundary layer simulations. *Journal of Computational Physics*, 140(2):233–258, 1998.

[52] K.A. Lundquist, F.K. Chow, and J.K. Lundquist. An immersed boundary method for the weather research and forecasting model. *Monthly Weather Review*, 138:796–817, 2010.

[53] C. Meneveau and J. Katz. Scale-invariance and turbulence models for large-eddy simulation. *Annual Review of Fluid Mechanics*, 32:1–32, 2000.

[54] C. Meneveau, T.S. Lund, and W.H. Cabot. A Lagrangian dynamic subgrid-scale model of turbulence. *Journal of Fluid Mechanics*, 319:353 – 85, 1996.

[55] O. Métais and M. Lesieur. Spectral large-eddy simulation of isotropic and stably stratified turbulence. *Journal of Fluid Mechanics*, 239:157–194, 1992.

[56] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005.

[57] J. Mohd-Yusof. Combined immersed boundary/B-spline methods for simulations of flow in complex geometries. Annual Research Briefs, Center for Turbulence Research, NASA-Ames/Stanford University, 1997.

[58] P. Moin and J. Kim. Numerical investigation of turbulent channel flow. *Journal of Fluid Mechanics*, 118:341–377, 1982.

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

[60] M. Negnevitsky, P. Mandal, and A.K. Srivastava. An overview of forecasting problems and techniques in power systems. In *Power Energy Society General Meeting, 2009. PES '09. IEEE*, pages 1 –4, july 2009.

[61] F. Nieuwstadt and H.B. Keller. Viscous flow past circular cylinders. *Computers & Fluids*, 1(1):59 – 71, 1973.

[62] NVIDIA. *NVIDIA CUDA C Best Practices Guide*. 2011.

[63] NVIDIA. *NVIDIA CUDA C Programming Guide 4.0*. 2011.

[64] NVIDIA. Kepler architecture. `http://www.nvidia.com/object/nvidia-kepler.html`, 2012.

[65] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[66] J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone, and J.C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879 –899, May 2008.

[67] C. S. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25(3):220 – 252, 1977.

[68] C. S. Peskin. The fluid dynamics of heart valves: Experimental, theoretical, and computational methods. *Annual Review of Fluid Mechanics*, 14(1):235–259, 1982.

[69] U. Piomelli. High reynolds number calculations using the dynamic subgrid-scale stress model. *Physics of Fluids A: Fluid Dynamics*, 5(6):1484–1490, 1993.

[70] U. Piomelli. Wall-layer models for large-eddy simulations. *Progress in Aerospace Sciences*, 44(6):437 – 446, 2008. ¡ce:title¿Large Eddy Simulation - Current Capabilities and Areas of Needed Research¡/ce:title¿.

[71] U. Piomelli and J. Liu. Large-eddy simulation of rotating channel flow using a localized dynamic model. *Physics of Fluids A: Fluid Dynamics*, 7(4):839–848, 1995.

[72] U. Piomelli, P. Moin, and J. H. Ferziger. Model consistency in large eddy simulation of turbulent channel flows. *Physics of Fluids*, 31(7):1884–1891, 1988.

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

[74] R. S. Rogallo and P. Moin. Numerical simulation of turbulent flows. *Annual Review of Fluid Mechanics*, 16(1):99–137, 1984.

[75] P. Sagaut. *Large Eddy Simulation for Incompressible Flows: An Introduction*. Springer-Verlag Berlin Heidelberg New York, 2 edition, 2002.

[76] J. Sanders and E. Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley, 7 2010.

[77] S.H. Schneider, M.D. Mastrandrea, and T.L. Root. *Encyclopedia of Climate and Weather*. Number v. 1 in Encyclopedia of Climate and Weather. Oxford University Press, 2011.

[78] I. Senocak, A. Ackerman, M. Kirkpatrick, D. Stevens, and N. Mansour. Study of near-surface models for large-eddy simulations of a neutrally stratified atmospheric boundary layer. *Boundary-Layer Meteorology*, 124:405–424, 2007. 10.1007/s10546-007-9181-x.

[79] I. Senocak, A.S. Ackerman, D.E. Stevens, and N.N. Mansour. Topography modeling in atmospheric flows using the immersed boundary method. Annual Research Briefs, Center for Turbulence Research, NASA-Ames/Stanford University, 2004.

[80] J. Smagorinksy. General circulation experiments with the primitive equations. *Monthly Weather Review*, 91(3):99–164, 1963.

[81] J.C. Smith, M.L. Ahlstrom, R.M. Zavadil, A. Sadjadpour, and C.R. Philbrick. The role of wind forecasting in utility system operation. In *Power Energy Society General Meeting, 2009. PES '09. IEEE*, pages 1 –5, july 2009.

[82] S.S. Soman, H. Zareipour, O. Malik, and P. Mandal. A review of wind power and wind speed forecasting methods with different time horizons. In *North American Power Symposium (NAPS), 2010*, pages 1 –8, sept. 2010.

[83] J.W. Taylor, P.E. McSharry, and R. Buizza. Wind power density forecasting using ensemble predictions and time series models. *Energy Conversion, IEEE Transactions on*, 24(3):775 –782, sept. 2009.

[84] The OpenFOAM Foundation. Openfoam user guide. `http://http://www.openfoam.org/docs/user/`, 2011.

[85] J. Thibault. Implementation of a cartesian grid incompressible Navier–Stokes solver on multi-GPU desktop platforms using CUDA. Master's thesis, Boise State University, Boise, Idaho, May 2009.

[86] J. C. Thibault and I. Senocak. Accelerating incompressible flow computations with a Pthreads-CUDA implementation on small-footprint multi-GPU platforms. *The Journal of Supercomputing*, 59:693–719, 2012.

[87] I. B. Troen and L. Mahrt. A simple model of the atmospheric boundary layer; sensitivity to surface evaporation. *Boundary-Layer Meteorology*, 37:129–148, 1986. 10.1007/BF00122760.

[88] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Elsevier, 2001.

[89] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169(2):708 – 759, 2001.

[90] Y.-H. Tseng and J. H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, 192(2):593 – 623, 2003.

[91] H.S. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna. A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. *Journal of Computational Physics*, 174(1):345 – 380, 2001.

[92] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25 – 37, 1992.

[93] US DoE. 20% wind energy by 2030: Increasing wind energy's contribution to US electricity supply. Technical report, Washington, D.C., 2008.

[94] H.K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Prentice Hall, 2 edition, 2007.

[95] R. Verzicco, J. Mohd-Yusof, P. Orlandi, and D. Haworth. Large eddy simulation in complex geometric configurations using boundary body forces. *AIAA Journal*, 38:427–433, 2000.

[96] T. Ye, R. Mittal, H.S. Udaykumar, and W. Shyy. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, 156(2):209 – 240, 1999.