

Direct Numerical Simulation of Atmospheric Flow over Rough Terrain

Mohammad Abouali
March, 2007

Direct Numerical Simulation of Atmospheric Flow over Rough Terrain

by

Mohammad Abouali

This thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation, Advanced Use of Remote Sensing in Water Resource Management, Irrigation and Drainage.

Thesis Assessment Board

Prof. Dr. Ir. Bob Su (Chair)

Prof. Dr. Ir. Bernard J. Geurts (External Examiner, Supervisor)

Dr. Ambro S.M. Gieske (Supervisor)

Prof. Dr. Ing. Wouter Verhoef (Member)



**INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION
ENSCHDEDE, THE NETHERLANDS**

Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Abstract

Numerical approaches towards atmospheric flow simulation started at the beginning of previous century, but just recently, during last decades with the help of powerful super computers, scientists were able to practically perform such studies. The turbulent nature of the atmosphere has made this procedure even more cumbersome.

The best set of equations, describing the turbulent flow, is known to be Navier-Stokes equations and, with the help of proper boundary conditions, this set of equations can be applied to atmospheric flow modelling. Navier-Stokes equations, except in very special cases, have no mathematical closed form solution. The only possible method to solve them is using the numerical algorithms, of which Direct Numerical Simulation (DNS) is one. DNS is the most accurate method but at the same time the most expensive one and it requires super computers. It was understood that the smaller eddies near the earth surface (near wall region), which has a great impact on increasing the cost of DNS, can be modelled at much lower cost. This approach is called wall-modelling and the main focus of this thesis is on coupling DNS with a wall model.

During this thesis, a code based on direct numerical simulation using FORTRAN programming language is developed which, with the help of proper boundary condition, can be used to simulate atmospheric flow over rough terrains. By coupling the code with a wall model, the memory and processing power requirements is reduced drastically (almost 90%); thus, much faster and less expensive simulations become possible. Moreover, by removing very fine grids near the wall due to using a wall model, a greater time step in simulation can be selected; causing even a faster simulation. Transforming this code to a LES based code is straightforward and requires just adding a sub-grid scale model (SGS); a simple step planned for future.

As the developed code, during this thesis, performs the simulations much faster and more economically, it can be used in several applications such as studying the water vapour transport, air pollution studies, cloud formation and many more.

Acknowledgements

While trying to write this section, the memories of these 18 months with all its ups and downs are reviewed rapidly in my mind, reminding me that without the supports of many people it was impossible to go all this distance all alone. Yet, it is not possible to mention all the single names here and I hope that I would be excused by those whose names are not here.

First, I want to thank my supervisors, Dr. Ambro S.M. Gieske and Prof. Dr. Ir. Bernard J. Geurts, who introduced me to this fantastic topic, supervised me all throughout the thesis, and, beside the technical part of the thesis, taught me life lessons. They also gave me the chance to investigate and experience many new things, which I may never be able to. I also want to thank the staffs of ITC and Twente University who, during these 18 months, helped me a lot in education, in administration part, in coordination, and even in the residence.

I also want to thank Dr. Ali Abkar, Ms. Roshanak Darvishzadeh, Dr. Majid Daftari, and Ms. Atefeh Mohammadi, who, with their kindness, helped me not to feel far away from my home, my town, and my country. I also want to thank my friends, especially Joris Timmermans, who taught me many aspects of Dutch culture and language.

The last but the best, I want to thank my wonderful parents, who I owe any success that I have in my life, and I want to dedicate this thesis to them.

Table of contents

Abstract.....	i
Acknowledgements.....	ii
Table of contents.....	iii
List of figures.....	v
List of tables.....	vii
1. Introduction.....	1
1.1. Atmosphere.....	1
1.2. Turbulent Flow.....	2
1.2.1. Reynolds Averaged Navier-Stokes (RANS).....	3
1.2.2. Large Eddy Simulation (LES).....	4
1.2.3. Direct Numerical Simulation (DNS).....	5
1.3. Objective and Research Questions.....	6
1.4. Methodology.....	6
1.5. Thesis Structure.....	7
2. Developed Turbulent Channel Flow.....	9
2.1. Channel Flow.....	9
2.2. Numerical Method.....	11
2.3. Qualitative Inspection of Numerical result.....	13
2.4. Time – Averaging Analysis.....	14
2.5. Quantitative Inspection of the Numerical Result.....	18
2.6. Law of the Wall and Introducing Wall Modelling.....	22
2.7. Summary.....	27
3. Boundary Layer Flow over Smooth Wall.....	29
3.1. Boundary Flow.....	29
3.2. Simulation Domain Description.....	31
Simulation Height and Grid Generation.....	32
Time Step.....	32
3.3. Qualitative and Quantitative Inspection of Simulation Result.....	34
3.4. Summary.....	38
4. Wall Modelling for Turbulent Flow over Smooth Wall.....	39
4.1. Wall Modelling Methods - History.....	39
4.1.1. Equilibrium Laws.....	39
4.1.2. Zonal Models.....	40
4.2. Developed Wall Model.....	40
4.2.1. Averaged Value over Time and Space (M1).....	40
4.2.2. Averaged Value over Space (M2).....	44
4.2.3. Averaged Value over Time and Spanwise Direction (M3).....	46
4.2.4. Averaged Value over Time.....	47
4.3. Summary.....	49
5. Boundary Flow over Rough Surfaces.....	51
5.1. Introduction.....	51
5.2. Homogenous Rough Surface.....	51
5.3. Patchy Rough Surface.....	53

5.4.	Summary	54
6.	Conclusions and Recommendations	55
6.1.	Conclusions	55
6.2.	Implementation to Real Problems	56
6.3.	Interactions with RS-GIS	57
6.4.	Applications	57
A.	Mathematical Description of Fluid Flow (Navier-Stokes Equation).....	59
A.1.	Modelling methods for the fluid flow	59
A.2.	Conservative Versus Non-conservative Form of the Equations	60
A.3.	Continuity Equation	60
A.4.	Momentum Equation.....	60
A.5.	Energy Equation.....	61
A.6.	Poisson's Equation to Solve Pressure Term.....	61
B.	Validation with Taylor-Green Class of Vortices	63
B.1.	Taylor-Green Class of Vortices	63
B.2.	Special Case of Taylor-Green Class of Vortices:.....	65
B.3.	Skew-Symmetry Preserving Simulation Result	65
B.4.	Reynolds Number Effect on Energy Dissipation	73
C.	Connecting to Super Computers and Downloading/Uploading Data.....	75
C.1.	Connecting	75
C.2.	Downloading and Uploading Data.....	77
D.	Variable Used in the code.....	79
	Reference:	85

List of figures

Figure 1.1: Mean and fluctuation part of a variable (adapted from Stull, 1999).....	4
Figure 1.2: Pressure spectrum inside a cavity	5
Figure 1.3: Zebra pattern surface roughness	7
Figure 2.1: Channel geometry (adapted from www.ercoftac.org)	9
Figure 2.2: No-slip boundary condition implementation	10
Figure 2.3: Grid in x-y plane.....	11
Figure 2.4: Velocity vectors in middle x-y plane at time t=2500 dimensionless units, The background is the interpolated pressure field.....	14
Figure 2.5: Simulation result versus theoretical lines: long time averaging was adopted here to obtain proper statistical evaluation.....	15
Figure 2.6: Time variation of the time-averaged $\overline{u^+}$ against $\overline{y^+}$	16
Figure 2.7: Re_{u_τ} Variation over time.....	17
Figure 2.8: Re_{u_τ} Variation over time (zoomed).....	17
Figure 2.9: Streaks are usually visualized in a plane near the wall.....	18
Figure 2.10: Streaks.....	18
Figure 2.11: Comparison of two profiles	20
Figure 2.12: Comparison of normalized Reynolds shear stress.	20
Figure 2.13: Variance of u,v, and w components of velocity.....	21
Figure 2.14: Bulk energy variation	21
Figure 2.15: Definition of roughness length scale	23
Figure 2.16: Smooth and rough Wall.....	23
Figure 2.17: Number of required grid cells (adopted from Piomelli et al., 2002).....	26
Figure 2.18: Full grid	26
Figure 2.19: Reduced grid. Fine grid cells near the wall are removed.....	26
Figure 3.1: A) Staggered grid, B) Collocated grid	29
Figure 3.2: Location of the First 'u' Relative to the Wall	30
Figure 3.3: Numerical Implementation of Non-Slip Boundary Condition.....	31
Figure 3.4: L10, L20, L30 grid.....	33
Figure 3.5: Grid in different horizontal resolution	33
Figure 3.6: Variation of Reynolds number based on friction velocity	35
Figure 3.7: Derivative as convergence control.....	35
Figure 3.8: RMS profile for three components of the velocity	36
Figure 3.9: Spanwise velocity profile.....	36
Figure 3.10: Vertical velocity profile	37
Figure 3.11: Comparison of simulation result with theoretical lines	37
Figure 4.1: Wall modelled results	42
Figure 4.2: Wall modelled results (zoomed)	42
Figure 4.3: Exaggerated error.....	43
Figure 4.4: Simulation result with and without wall modelling	44
Figure 4.5: Wall model version 2.....	46
Figure 4.6: Wall modelling using averaged values over time and spanwise direction.....	47
Figure 4.7: Model performance using averaged values over time and spanwise direction	47

Figure 4.8: Wall model using averaged value over time	48
Figure 4.9: Wall model performance using averaged value over time	48
Figure 5.1: Wall model result over homogenous rough wall	52
Figure 5.2: Patchy rough surface.....	52
Figure 5.3: Velocity profile in different locations over patchy rough wall	53
Figure 5.4: Surface with Four Patches.....	54
Figure 5.5: Velocity Variation.....	54
Figure 6.1: Physical Domain Versus Computational Domain.....	56
Figure 6.2: Foot Print of a Measuring Device	58
Figure B.1: Special Case of Taylor-Green Class of Vortices at time 0	67
Figure B.2: Special Case of Taylor-Green Class of Vortices at time 1	67
Figure B.3: Special Case of Taylor-Green Class of Vortices at time 2	68
Figure B.4: Special Case of Taylor-Green Class of Vortices at time 3	68
Figure B.5: Special Case of Taylor-Green Class of Vortices at time 4	69
Figure B.6: Special Case of Taylor-Green Class of Vortices at time 5	69
Figure B.7: Bulk Energy Variation	70
Figure B.8: Bulk Energy Variation (zoomed)	70
Figure B.9: Energy Dissipation (Kuczaj, 2003)	71
Figure B.10: Vorticity Vectors at t=0, 1, 2, 3 (Kuczaj, 2003).....	71
Figure B.11: Bulk Energy in 64^3 and 32^3 cells.....	72
Figure B.12: Bulk Energy in 64^3 and 32^3 cells (zoomed).....	72
Figure B.13: Reynolds Number Effect on Bulk Energy.....	73
Figure C.1: Putty Screen.....	76
Figure C.2: Pocket Putty	77

List of tables

Table 4.1: Modelled and simulated velocities	41
Table 6.1: Common scales	57
Table A.1: Continuity Equation	60
Table A.2: Momentum Equation	60
Table A.3: Energy Equation	61
Table D.1: Variables used in Direct Numerical Simulation	79
Table D.2: Variables and parameters used in the text	83

1. Introduction

1.1. Atmosphere

All creatures, living on the earth's surface, depend on the atmosphere to survive. Most human activities require a thorough understanding of the behaviour of the atmosphere. Several examples come to the mind. The agricultural processes depend on sunshine; therefore, on the presence or absence of clouds, on the temperature, and on the amount of water vapour available in the atmosphere. Both navigation and aviation depend on how strong the wind is, in which direction it is blowing, and how well the visibility is. The performance and fuel consumption of mechanical devices depend on air pressure, temperature and humidity. In all electronic devices' manuals, there is a page related to the ranges of atmospheric characteristics that the device is designed to work in. All quantitative remote sensing algorithms have to deal with the atmosphere existing between the object on the earth and the sensor mounted on the satellite. Therefore, a thorough understanding of the atmosphere, its behaviour, and its interactions with the earth is indispensable.

Although understanding the atmosphere and its behaviour is very important, the amount of unknowns and uncertainties in this field are rather high. Although introduced models by scientists have become more sophisticated and more complex, it is still not possible to accurately define or predict the behaviour of the atmosphere. In spite of powerful super-computers with large amount of memory and huge processing power, atmospheric models cannot predict more than 5 days into the future. This arises from shortcomings of the model and from the underlying chaotic state of the turbulent atmosphere.

Despite its importance, the turbulent nature of the atmosphere has caused the field of atmospheric predictions to be still in the focus of the scientific developments. Turbulence and turbulent flow is one of the unsolved problems existing in classical physics (Stull, 1999). This makes it even a more challenging topic to approach.

Turbulent flow has a great capacity for mixing; therefore, modelling the turbulent behaviour of the atmosphere is also relevant for many practical environmental applications. For example, chimney plumes are spread over larger areas and diluted more in the presence of turbulence. Likewise, during severe storms the turbulent fluctuations can be fatal to tall buildings or bridges, (Holtslag and Ek, 1996).

Prior to modelling the atmosphere, one should know what the parameters affecting the behaviour of the atmosphere are and understand what the main reason for the turbulent nature of the atmosphere is, especially in the first 100 to 3000 meter above the earth. This region is called Atmospheric Boundary Layer or ABL and is being affected by surface properties of the earth and its undulations within an hour (Oke, 1988).

According to Duynkerke's studies, (Duynkerke and Driedonks, 1987), in a cloud-free ABL, the dynamics of the turbulence are primarily affected by:

- Wind shear at the earth's surface
- Wind shear near the ABL top
- Buoyancy flux at the surface

If the atmosphere is topped or capped with clouds, in addition to the above mentioned items the following aspects are also influencing the turbulence nature of the ABL:

- Long wave radiative cooling at the cloud top
- Short wave radiative heating inside the cloud layer
- Phase changes

Among the above mentioned parameters, the most important factors are "wind shear" and "buoyancy flux" at the surface. The focus in this thesis is especially on surface roughness, how it affects the flow, how the law of the wall has to be changed to take into account the effect of surface roughness, e.g., arising from vegetation or buildings, and how the spatial distribution of roughness affects the flow and scalar transport in the atmosphere. The Navier-Stokes equations are used to simulate the turbulence behaviour of the atmosphere and a finite volume numerical discretization method, based on work by Verstappen and Veldman (Verstappen and Van der Velde, 2006; Verstappen and Veldman, 2003) , is used to solve these equations.

Using the Navier-Stokes equations and a numerical algorithm is one of many methods for studying the turbulent flow and fluid dynamics. This approach is usually referred to as Computational Fluid Dynamics or CFD. Formerly, physical experiments were the only way to study the fluid flow. But nowadays, with the help of fast and powerful computers and suitable numerical algorithm, simulation, theoretical and physical experiments have joined together and made more complete investigations possible.

1.2. Turbulent Flow

The Navier-Stokes equations are used to simulate and study the turbulent flow. A French scientist, Claude Navier (1822), and an English scientist, George Stokes (1845), (Anderson, 1995; Polyanin et al., 2002; Wikipedia1) independently formulated mathematical equations to describe the turbulent behaviour of the fluid. These equations can be written for an incompressible fluid as:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u - \frac{1}{Re}(\nabla \cdot \nabla)u + \nabla p = 0 \quad (1.1)$$

$$\nabla \cdot u = 0 \quad (1.2)$$

The parameter Re is the Reynolds number and is defined as $Re = \frac{Uh}{\nu}$, where h is half the channel height, U is a reference velocity, and ν is the viscosity. In these equations, u is the velocity

and p is the pressure. For practical applications, these equations can only be solved by numerical methods. Navier-Stokes equations are summarized in appendix A.

It is perhaps surprising that one may assume air to be incompressible. According to Businger (Businger, 1982; Stull, 1999), if V and L are typical velocity and length scale for boundary layer and if they satisfy the following conditions, the fluid can be simulated with high accuracy using the incompressible form of Navier-Stokes equations:

- $V \ll 100 \text{ ms}^{-1}$
- $L \ll 12 \text{ km}$
- $L \ll \frac{c_s^2}{g}$
- $L \ll \frac{c_s}{f}$

The parameter c_s is the speed of sound, f is the frequency of any pressure waves, and g is the gravity acceleration. Except in some severe storms, these conditions are well satisfied in the atmospheric boundary layer (Stull, 1999). This is known as incompressibility approximation (Dutton and Fichtl, 1969). Within this approximation, it is not possible to use the equation of state. Instead, a Poisson equation has to be used for the pressure field. The detailed discussion on this subject is given in appendix A.

There exist three different major methods to solve the Navier-Stokes equations, either in full or after invoking specific modelling steps related to turbulence:

- Reynolds Averaged Navier-Stokes (RANS)
- Direct Numerical Simulation (DNS)
- Large Eddy Simulation (LES)

There is a crucial difference between turbulent flow simulation and modelling. According to Pope (Pope, 2000), turbulent flow simulation is solving the time-dependent velocity equations which represent to some extent the velocity field. However, turbulent modelling means solving the equations for some averaged value. In this research, the terms modelling and simulation have been used in the same meaning as turbulence simulation, except if noted.

1.2.1. Reynolds Averaged Navier-Stokes (RANS)

The Reynolds Averaged Navier-Stokes method is the first method used presently by engineers to tackle the Navier-Stokes equations. It requires the least amount of processing power and memory but is most demanding on the turbulence modelling. In this method, the velocity field is averaged over time; therefore, just a rough estimate of the actual velocity field would be achieved. In this method the variables are decomposed in two parts: (I) the average part or mean part and (II) the fluctuation part. A flow-variable V is usually decomposed as:

$$V = \bar{v} + v' \tag{1.3}$$

The first term \bar{v} is the mean part (averaged over time) and the second term v' is the fluctuation part. For better understanding of (1.3), refer to Figure 1.1, adapted from Stull (Stull, 1999). It can be easily shown that $\overline{v'} = 0$. This method is fast but not accurate enough in many situations. Still, it is being used widely in engineering studies.

After inserting the decomposition (1.3) in the Navier-Stokes equations, several extra terms appear in the equations for the mean flow prediction. These extra terms contain the fluctuations and are not explicitly known in terms of the mean flow. These contributions are known as the Reynolds stresses and are usually modelled with a turbulence model. Often an eddy-viscosity model is adopted such as a variant of the Prandtl mixing model or a more recent model such as the k-epsilon model. The RANS method often has to be calibrated for individual applications in order to be adapted to the regional properties. There is no single globally acceptable RANS model.

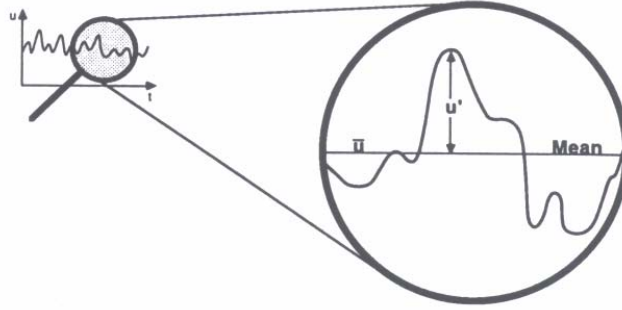


Figure 1.1: Mean and fluctuation part of a variable (adapted from Stull, 1999)

1.2.2. Large Eddy Simulation (LES)

The main hypothesis in the Large-Eddy Simulation (LES) method is that the structures of the large eddies, or large scales of motion, are being affected by the mean flow characteristics and the small fluctuations are represented by a sub-grid model. In other words, a scale-separation is imposed in which the variables are being decomposed into (I) a resolved and (II) an unresolved part. The resolved field of LES is somewhat analogous to the mean field and the unresolved field of LES resembles the fluctuation field of RANS method. However, the averaging is now formulated in terms of a spatial averaging, also known as filtering. This filtering is performed such that the instantaneous value of the variables can be written in the form of (1.3). The decomposition is being performed by using a general filter function like:

$$\bar{v}_i(x_1, x_2, x_3) = \iiint V_i(x'_1, x'_2, x'_3) \cdot G(x_1 - x'_1, x_2 - x'_2, x_3 - x'_3) dx'_1 dx'_2 dx'_3 \quad (1.4)$$

The filter is characterized by a filter-kernel G which has a characteristic width. The filter size used in (1.4) is crucial. If the filter size is very small, the accuracy of the LES results are very close to that of DNS, but it will be very costly, and if the filter size is large, the LES results will become very cheap to obtain but will be too much smoothed for most practical purposes (Geurts, 2004). Figure 1.2, adapted from (Sagaut and Meneveau, 2004), shows the differences between the RANS and LES

method in comparison to a measured pressure spectrum inside a cavity. It can be seen that the RANS method is just simulating the big variations and the overall behaviour of the variable and it deviates from the true value in time. But LES is able to even simulate small changes and gives more information and while time advances it deviates less from the true values.

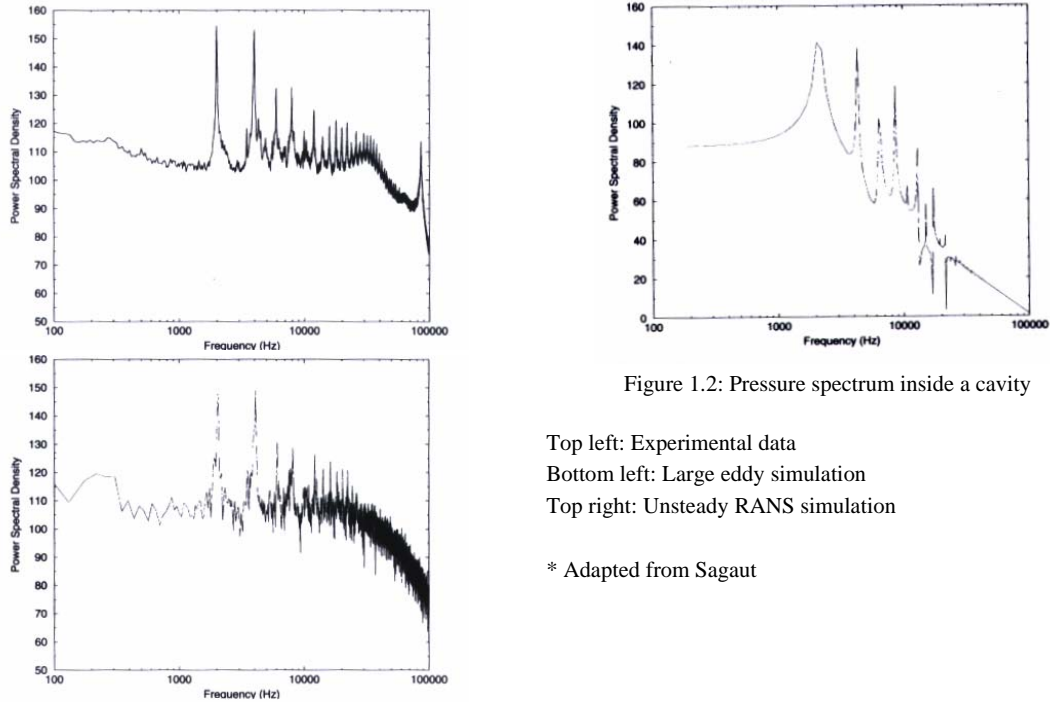


Figure 1.2: Pressure spectrum inside a cavity

Top left: Experimental data
 Bottom left: Large eddy simulation
 Top right: Unsteady RANS simulation

* Adapted from Sagaut

1.2.3. Direct Numerical Simulation (DNS)

In this method the Navier-Stokes equations are being solved while all scales of motion are being resolved completely. This is the most complete and accurate simulation that exists but it has one big disadvantage. The cost and time needed for this method is proportional to Re^3 ; an increase in the Reynolds number, Re , by a factor of 10 requires 1000 times more computational efforts. While simulating the atmospheric flow, Re can become very high; therefore, DNS is very costly and often not feasible.

Although being costly, it does not mean that DNS can not be used at all. In research it is one of the best available methods. Recently, by introducing new discretization methods, better time advancing algorithms, faster numerical methods, and increased memory and processing power of super computers, it is possible to use DNS in atmospheric research to complement other methods. In this research a DNS model is coupled with a wall model for a rough boundary to reduce the simulation costs which also requires much less memory. Coupling to a model for the small turbulent scales can readily be implemented in future research.

1.3. Objective and Research Questions

“The challenge of modelling the atmospheric boundary layer is the prediction of the temporal variation of the vertical and horizontal structures in response to the influence of the major processes acting in the atmosphere and at the earth’s surface” (Holtslag and Ek, 1996).

The main objective in this research is to study how CFD techniques can be used in atmospheric flow studies and to learn how physically and numerically the fluid flow has to be treated. Regardless of the environment that the Navier-Stokes equations are used, they do not change their form. The only possible method to adopt these equations to a specific environment is through applying boundary-conditions (Anderson, 1995). Therefore, as sub-objective, the boundary condition requirements and how they have to be numerically implemented will be studied.

Since simulating near wall fluid motion is both timely and costly, in this research the modelling of the near wall fluid motion will be focussed. This approach is called “wall modelling” and is fully described in section 2.6. Later, we study how wall roughness can be incorporated into the wall modelling. At the end, a DNS model, which is much faster and requires much less memory, would be developed during this thesis.

1.4. Methodology

As the first step, a CFD code, currently used at Twente University by Prof. B. J. Geurts, was used to familiarize with CFD concepts. This code was evaluated in two cases (I) fully developed channel flow, chapter 2, and (II) Taylor-Green class of vortices, appendix B.

After getting acquainted with the code and the concept of numerically simulating the fluid motion, the code was adapted to atmospheric flows by using proper far-field boundary conditions.

At this stage the code was still fully solving all the scales of the motion and very costly. Therefore, it was tried to combine the DNS code with a wall model in order to reduce the cost of simulation. The next step was to simulate a homogeneously roughed wall and evaluate the functionality of the new DNS code equipped with the developed wall model. After successfully evaluating the homogeneously roughed wall case, the patchy case or zebra pattern was studied, Figure 1.3.

In this thesis the FORTRAN language is used to develop the code and all the analysis is performed using this language and MATLAB is used for visualization and post processing.

It is worth to mention that there are currently many LES codes equipped with a wall model and almost the same procedure was used in this research. We adopt DNS, albeit at coarse resolutions, but extensions that include sub-grid models are readily made in the future. An example of these LES codes can be found on Parlange and Albertson article (Albertson, 1996; Albertson and Parlange, 1999) and (Parlange et al., 1995).

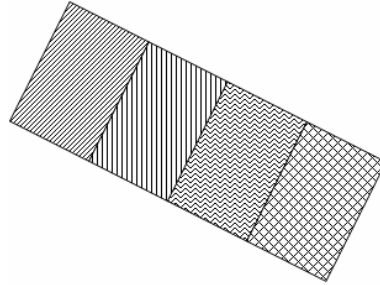


Figure 1.3: Zebra pattern surface roughness

Usually with regular PCs, one is not able to achieve the requirements of DNS simulations. Therefore, all the simulations were executed on two super computers called TERAS and ASTER, located in Amsterdam. Both TERAS and ASTER super computers belong to “SARA computing and networking services”. This institution is offering advanced high performance computing and visualizations, high performance networking and infrastructure services for more than 30 years.

TERAS super computer consists of 1024 Central Processing Units (CPUs), 1 terabyte of memory in total, 10 terabytes of on-line storage, and 100 terabytes near-line storage (1 terabyte equals 1024 Gigabytes). To make it more reliable, the super-computer is divided into 6 different partitions or units. The partitioning has been done according to the Cache-Coherent Non Uniform Memory Access (CC-NUMA) standard. It means that the system will run one operating system, UNIX in this case, and will show a single memory image to the user, even though the memory is physically distributed over several CPUs and machines. TERAS consists of two interactive partitions with 32 CPUs, three batch partitions with 64, 128 and 256 CPUs, and one partition with 512 CPUs, in total 1024 CPUs. The detailed instruction on how to connect to these super computers, download, and update data is discussed in appendix C.

1.5. Thesis Structure

This thesis is organized in 6 chapters, including this introduction chapter. In chapter 2, a brief description of the numerical scheme, used in this thesis, is given and the code is validated in a fully developed channel. Meanwhile, I have made myself acquainted with the concepts and basics of numerical modelling of the fluids or CFD.

Later in chapter 3, the required changes needed to adapt the code for atmospheric boundary layer flows are discussed and again the code was run and validated in several cases and the simulation results are given and discussed.

In chapter 4, the wall modelling concept is discussed and different wall models are presented. These wall models are validated over a smooth wall and the results are discussed. In chapter 5, the wall models are tested against homogenous rough wall, where the roughness is kept constant all over the simulation domain, and later against patchy surface, where a single rough patch is introduced in the middle of the simulation domain facing the wind direction.

Finally, in chapter 6, the conclusion and further recommendation for future is given. This thesis includes 4 appendices. The appendix A introduces the general dimensional form of the Navier-Stokes Equations, appendix B represents the code validation with Taylor-Green class of vortices, appendix C discusses some practical issues related to connecting and working with super computers, and finally appendix D summarizes the variables used in the thesis and in the computer code.

2. Developed Turbulent Channel Flow

2.1. Channel Flow

The DNS code, used in this thesis, is based on skew-symmetric discretization (Verstappen and Van der Velde, 2006; Verstappen and Veldman, 2003), which preserves the symmetry of the convective and dissipative terms. It will be applied and tested for a fully developed channel flow. A channel with dimensionless height of 1, spanwise length of π and streamwise length of 2π is chosen. A right-handed axis is used in a way that its x component is along the channel stream, y component is the height, and z component is perpendicular to x and y . The velocities along the x , y , and z axis are called u , v , and w respectively, see Figure 2.1.

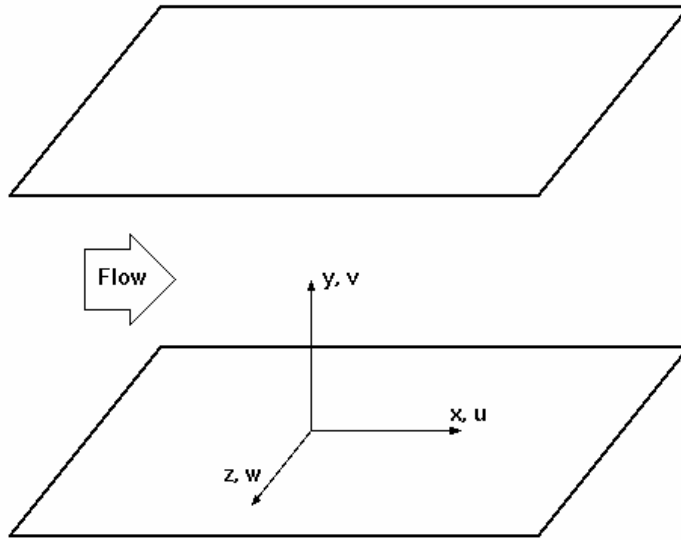


Figure 2.1: Channel geometry (adapted from www.ercoftac.org)

The flow in the channel is wall bounded in the y direction, which means that the no-slip condition is applied at the walls. If a computational grid (x_i, y_j, z_k) is introduced with $i=0, \dots, Nx$, $j=0, \dots, Ny$, and $k=0, \dots, Nz$ then the no-slip conditions are applied at both y_0 and y_{Ny} . Here N_y is the number of cells in y direction, similarly for N_x and N_z . The no-slip boundary condition states that all components of velocity (u, v, w) should be zero at the wall, (Wikipedia2), or:

$$y = y_0, y_{Ny} \rightarrow u = v = w = 0 \quad (2.1)$$

To apply the no-slip boundary condition, dummy nodes are used and the velocities are distributed over them using an odd function as in Figure 2.2.

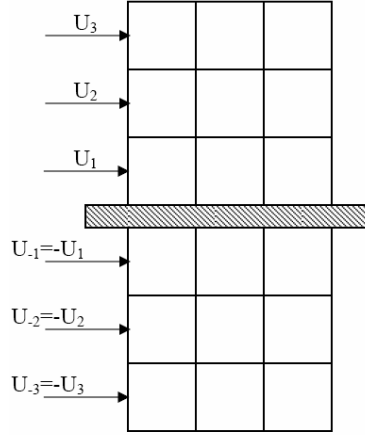


Figure 2.2: No-slip boundary condition implementation

In other words, the velocities are negative-equal along the wall or:

$$\left\{ \begin{array}{lll} u(0) = 0 & v(0) = 0 & w(0) = 0 \\ u(-1) = -u(1) & v(-1) = -v(1) & w(-1) = -w(1) \\ u(-2) = -u(2) & v(-2) = -v(2) & w(-2) = -w(2) \\ u(-3) = -u(3) & v(-3) = -v(3) & w(-3) = -w(3) \end{array} \right\} \quad (2.2)$$

The boundary conditions in the x and z directions are periodic. The Reynolds Number, Re , is set to 5600 which corresponds to $\nu = \frac{1}{Re} = 0.179 * 10^{-3}$, where ν is the viscosity. The time step is set to be $dt = 1.25 * 10^{-3}$ dimensionless time units.

The simulation space, which was selected to be $2\pi * 1 * \pi$, was divided into $32 * 32 * 32 = 32768$ grid cells. The mesh or grid size is chosen to be equidistant along the x and z directions, but along the y axis near the wall a smaller mesh size is chosen. The expression (2.3) is used to generate the y coordinates of the computational cells for half of the channel. For the other half, the coordinates were mirrored along a line parallel to the x axis passing through the middle of the channel. For better understanding of the grid in x - y plane refer to Figure 2.3.

$$y_j = a * \sinh\left(\frac{b * j}{N_y}\right)$$

where :

$$b = 6.5 \quad (2.3)$$

$$a = \frac{L_y}{2 * \sinh\left(\frac{b}{2}\right)}$$

$L_y = \text{channelheight}$

The chosen cell sizes are much larger than what is actually needed for DNS at $Re=5600$. Veldman and Verstappen (Verstappen and Van der Velde, 2006; Verstappen and Veldman, 1997; Verstappen and Veldman, 2003) claimed that using the symmetry-preserving discretization will help the numerical schemes to remain stable with larger cell sizes and larger time steps. Hence, it will also be tested how accurate the results are with a very coarse resolution.

The velocity and pressure fields are initialized at $time=0$, in the same way as proposed by Veldman and Verstappen, using the expression (2.4). It is worth to mention that this initial condition is very different than fully developed turbulent channel flow. Although, it causes the computational model to converge only gradually to the final state, but on the other hand it shows that the model is able to converge to the correct state in spite of a very distant initialization:

$$\begin{aligned}
 u &= 1.0 \\
 v &= 0.2 \sin\left(\frac{12x}{\pi}\right) \\
 w &= 0.2 \sin\left(\frac{24z}{\pi}\right) \\
 p &= 0
 \end{aligned}
 \tag{2.4}$$

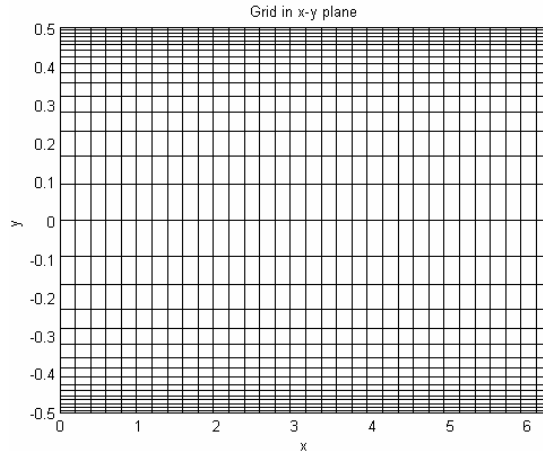


Figure 2.3: Grid in x-y plane

2.2. Numerical Method

The current code is based on the symmetry preserving discretization method. This method is based on the finite volume method and uses the transport theorem. The details of this discretization are given in (Verstappen and Van der Velde, 2006; Verstappen and Veldman, 2003). The method is reviewed here very briefly.

The computational domain is divided into N_x , N_y , and N_z grid cells in each direction. To reduce the error caused by the discretization, two control volumes are defined which are combined to yield a formally fourth order method from a combination of two second order discretizations. These

two volumes are denoted here by Ω_1 and Ω_3 in which the latter one is three times bigger than the first one. These volumes can be defined as:

$$\langle \Omega_1 \rangle_{i,j,k} = dx_i^1 dy_j^1 dz_k^1 \quad (2.5)$$

$$\langle \Omega_3 \rangle_{i,j,k} = dx_i^3 dy_j^3 dz_k^3 \quad (2.6)$$

Where:

$$\left\{ \begin{array}{ll} dx_i^1 = x_i - x_{i-1} & dx_i^3 = x_{i+1} - x_{i-2} \\ dy_j^1 = y_j - y_{j-1} & dy_j^3 = y_{j+1} - y_{j-2} \\ dz_k^1 = z_k - z_{k-1} & dz_k^3 = z_{k+1} - z_{k-2} \end{array} \right\} \quad (2.7)$$

Also the control volume $\langle \Omega_1 \rangle_{i+\frac{1}{2},j,k}$ and $\langle \Omega_3 \rangle_{i+\frac{1}{2},j,k}$ is defined as:

$$\langle \Omega_1 \rangle_{i+\frac{1}{2},j,k} = dx_i^{S1} dy_j^{S1} dz_k^{S1} \quad (2.8)$$

$$\langle \Omega_3 \rangle_{i+\frac{1}{2},j,k} = dx_i^{S3} dy_j^{S3} dz_k^{S3} \quad (2.9)$$

Where

$$\left\{ \begin{array}{ll} dx_i^{S1} = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}} & dx_i^{S3} = x_{i+\frac{3}{2}} - x_{i-\frac{3}{2}} \\ dy_j^{S1} = y_{j+\frac{1}{2}} - y_{j-\frac{1}{2}} & dy_j^{S3} = y_{j+\frac{3}{2}} - y_{j-\frac{3}{2}} \\ dz_k^{S1} = z_{k+\frac{1}{2}} - z_{k-\frac{1}{2}} & dz_k^{S3} = z_{k+\frac{3}{2}} - z_{k-\frac{3}{2}} \end{array} \right\} \quad (2.10)$$

And the half-way grid nodes are defined as the average of the upper and lower grid nodes, i.e.

$$x_{i+\frac{1}{2}} = \frac{x_i + x_{i+1}}{2}; x_{i-\frac{1}{2}} = \frac{x_i + x_{i-1}}{2}; x_{i+\frac{3}{2}} = \frac{x_{i+1} + x_{i+2}}{2}; x_{i-\frac{3}{2}} = \frac{x_{i-2} + x_{i-1}}{2}. \quad \text{The control volume is}$$

written in matrix form as:

$$\Omega = 3^{2+d} \Omega_1 - \Omega_3 \quad (2.11)$$

And d equals the dimension of the analyses.

The Navier-Stokes momentum equation is written in its discretized form as:

$$\Omega \frac{du}{dt} + C(u)u + D.u - M^* .P = 0 \quad (2.12)$$

Here, C is a matrix built from the convective terms, D is a matrix built from the diffusive term, and M is a matrix representing gradient operator. The star (*) denotes the matrix transpose operation. For the detailed definition of C , D , and M refer to Veldman and Verstappen (Verstappen and Veldman, 2003).

The discretized form of the continuity equation can be written as:

$$M .u = 0 \quad (2.13)$$

in terms of the same matrix M as used in (2.12).

The deriving force is defined as constant flow. This constant flow is provided by manipulating the pressure gradient term in (2.12). The pressure gradient is defined as:

$$\partial_i P = \beta(t) + \partial_i p \quad (2.14)$$

Equation (2.14) consists of an added pressure field ($\partial_i p$) on top of a linear pressure gradient, $\beta(t)$. In this research, the linear pressure drop was kept constant during the simulation; therefore, it does not depend on time.

2.3. Qualitative Inspection of Numerical result

The code was run for quite a long time. After each 40000 iterations, which equals 50 dimensionless time units, the velocity fields are backed up, the average values are calculated and written to a file, the turbulent energy is being calculated and stored in a separate file, all the files are backed up, and the program is restarted to simulate another 40000 time steps. The 40000 as the number of iterations was arbitrary chosen. The total time of simulation was 3200 dimensionless time units of which 64 frames at a time distance of 50 units were stored in 64 different files. The total amount of stored data is around 600 Megabytes.

After running the code for quite a long time the velocity fields in the x-y plane take the shape of the velocity in a fully developed channel flow, i.e., the velocity vector is zero at the walls and it grows logarithmically. The maximum value is at the centre of the channel and there is no flow going toward or away from the wall. Although in some cases we can see jets going away or toward the wall, these are part of the turbulent structure of any non-laminar flow and in fact they should exist. More discussion of such phenomena is given in (Pope, 2000). These types of structures are usually called “Ejections” if they are away from the wall and “Sweeps” if they are toward the wall, Figure 2.4.

The colour background of Figure 2.4 is the interpolated pressure field. It is obvious that the flow is from higher pressure to lower pressure in all cases. Note that the pressure field has been obtained by solving the Poisson equation. Further discussion on this subject is in appendix A.

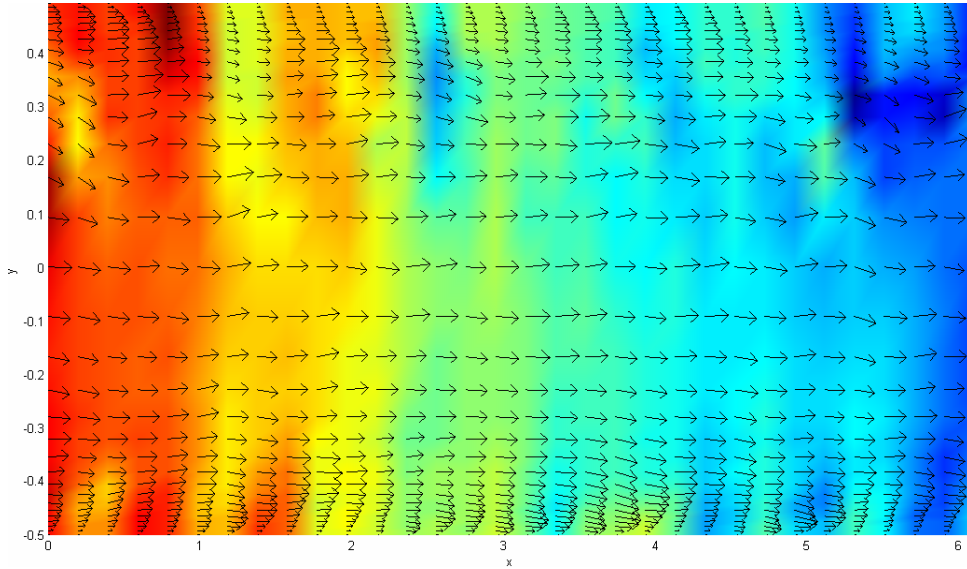


Figure 2.4: Velocity vectors in middle x-y plane at time $t=2500$ dimensionless units, The background is the interpolated pressure field.

The mean velocity profile along the x axis, i.e., the streamwise u velocity, is shown in figure 2.5. The y^+ axis is shown horizontally on a logarithmic scale. The line with circles on it is the simulation result. The circles show the actual nodes for which a value for u^+ exists. The theoretical lines are also drawn in the same figure for comparison. The wall coordinate is defined as in section 2.6.

Near the wall, the velocity in wall-coordinates is given by $u^+ = y^+$. We can see that although a very coarse resolution was chosen, the calculated profile still matches the theoretical line quite well near the wall, despite the existence of very small eddies relative to selected grid-cell size in that region. When the condition $y^+ > 35$ is satisfied, the velocity changes gradually toward a logarithmic profile. This approximation is well-known in the literature as the “Law of the Wall” or “Wall Function”, which is discussed extensively in section 2.6.

2.4. Time – Averaging Analysis

As mentioned earlier, the code was run for 3200 dimensionless time units to make sure that constant values have been achieved for the long-time averaging and the channel can be considered in a fully developed state. To clarify this better, the $u^+ - y^+$ profile was examined at 4 different times which are at 50, 100, 800, and 3200, Figure 2.6. It is evident that u^+ near the wall converges fast and quite quickly and it can not be distinguished from its final value at the end of simulation at time 3200. But when getting farther away from the wall a clear difference can be observed which does require rather lengthy time-intervals for proper time-averaging. It is worth to mention that the scale is logarithmic, so if a difference is seen, it is quite a big difference. Fortunately, the profile quickly converges to its final state at around 800 dimensionless time units.

Another method to decide whether the code was run long enough to reach a statistically stationary state, is through checking the variation of variables such as u^+ , $Urms$, $Vrms$, $Wrms$, and uv . The $Urms$, $Vrms$, and $Wrms$ are the variances of u , v , and w respectively. This can be shown as follows:

$$\begin{aligned} (u')^2 &= (u - \bar{u})^2 = u^2 - 2u\bar{u} + \bar{u}^2 \\ \text{Averaging :} & \\ \sigma^2 &= \overline{(u')^2} = \overline{u^2} - \bar{u}^2 \end{aligned} \quad (2.15)$$

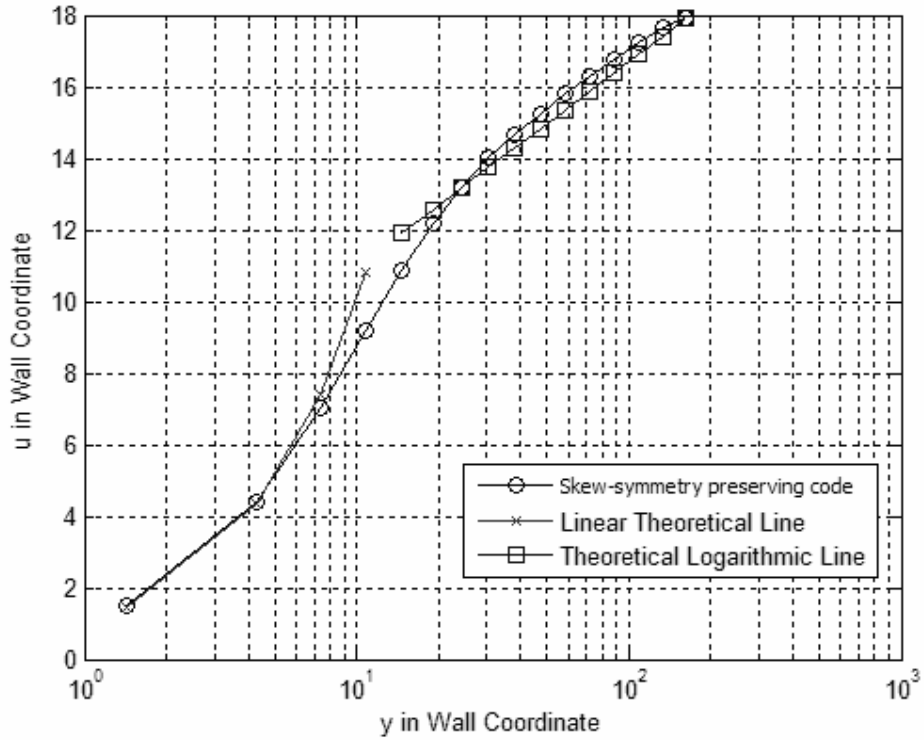


Figure 2.5: Simulation result versus theoretical lines: long time averaging was adopted here to obtain proper statistical evaluation.

The fluctuations in uv are actually the Reynolds shear stress. The variation of the L2-norm of u^+ , $Urms$, $Vrms$, $Wrms$, and uv as a function of the time used in the long-time averaging were also checked, which are not shown for simplicity. The L2 norm of a function is calculated using equation (2.16).

$$\|f\|_2^2 = \frac{1}{y_{N_y} - y_0} \sum_{i=0}^{N_y-1} \frac{(f_i + f_{i+1})^2 (y_{i+1} - y_i)}{2} \quad (2.16)$$

By monitoring the variation of $Re_{u_\tau} = \frac{u_\tau \delta}{\nu}$ over time, where δ is half the channel height, it can also be decided whether the simulation time was long enough or not. Usually a shorter simulation is being performed and the variation of Re_{u_τ} is inspected. If it has not yet reached its asymptotic state

then the simulation is advanced in time for another period using the current result as start-up. This procedure is repeated until the Re_{u_r} reaches a more or less constant value. Figure 2.7 and Figure 2.8 shows the Re_{u_r} variation over time. The supporting lines show one percent above and below the final value, i.e., the value at time 3200. It can be seen that after 1200 dimensionless time unit, Re_{u_r} is within the one percent margin of its value at $t = 3200$. So, the simulation time, 3200, was long enough for all time-averaged values to converge well to their final state..

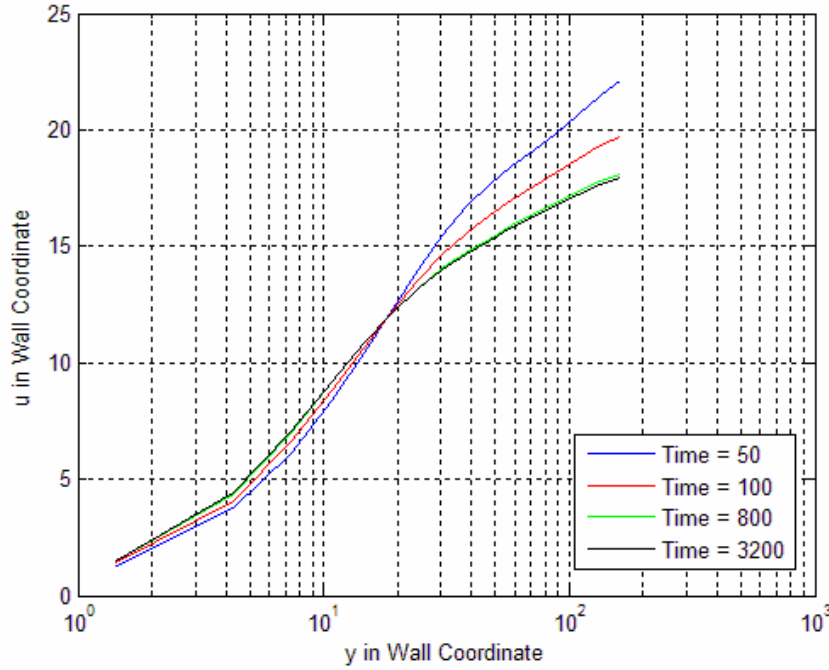


Figure 2.6: Time variation of the time-averaged $\overline{u^+}$ against $\overline{y^+}$

A slight jump at time equal to 1600 dimensionless time units can be seen in all figures. This jump is due to resetting the time and spatial averaging. The spin-up or warm-up time was decided to be 1600, in order to remove any dependency on the selected initial condition in (2.4). Subsequently, the real simulation in which data are collected was continued for another 1600 time-units. Around time=1600 there is a reset in averaging, causing this jump into the result.

One of the phenomena usually seen in a turbulent channel flow is the streaking effect of the velocity field in a plane parallel to and very near to the wall, Figure 2.9. The streamwise velocity in some regions is higher than in other regions. These regions are in a dynamic striped form along the flow; this behaviour is called streaking, Figure 2.10.

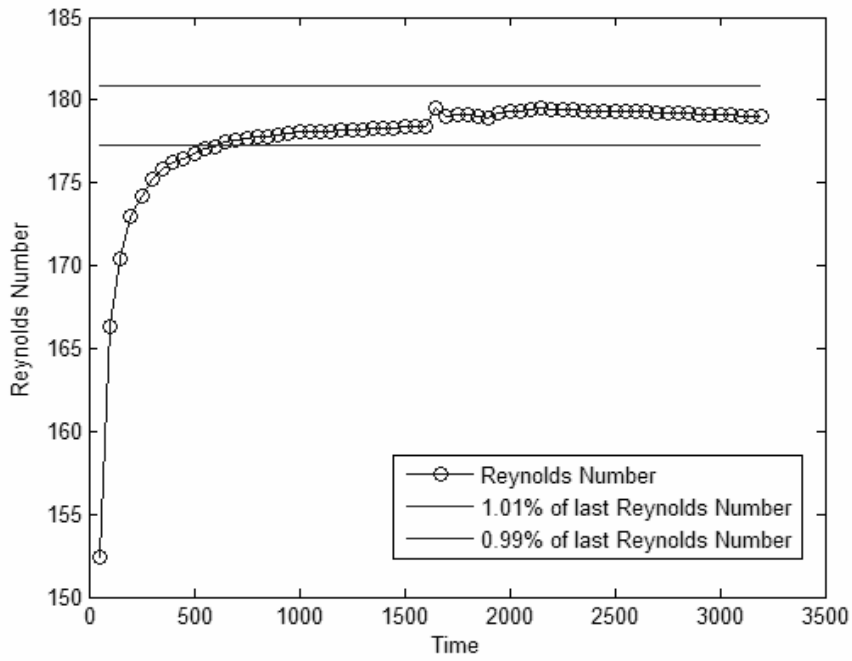


Figure 2.7: Re_{u_r} Variation over time

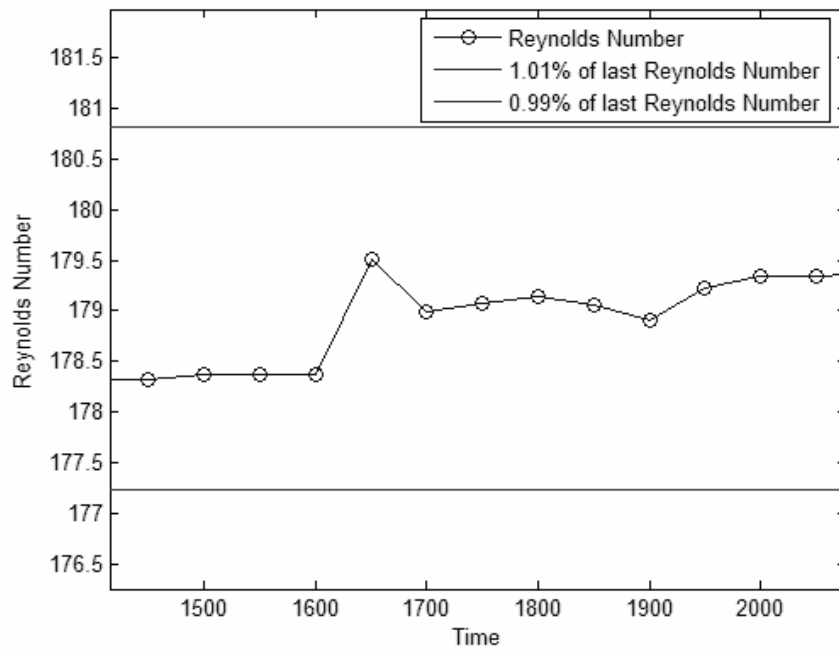


Figure 2.8: Re_{u_r} Variation over time (zoomed)

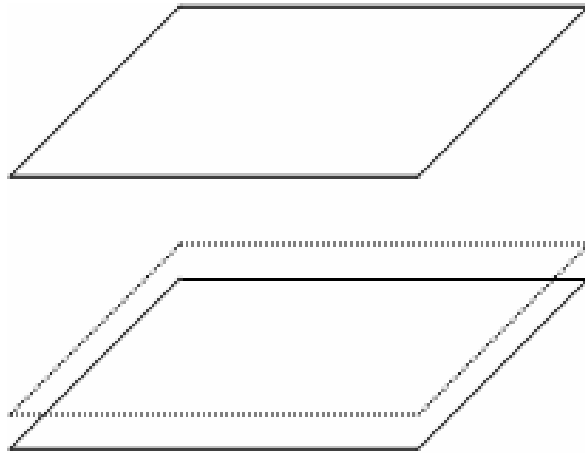


Figure 2.9: Streaks are usually visualized in a plane near the wall

2.5. Quantitative Inspection of the Numerical Result

In this section we compare the simulation results with (Kim et al., 1987). The spanwise channel dimension of Kim (Kim et al., 1987) is $2\pi\delta$, the streamwise channel dimension is $4\pi\delta$, and δ is the channel height. The computational domain is divided into $192*129*160=3,962,880$ cells in x , y , and z direction respectively. The database is available at www.ercoftac.org. Remember that the channel dimension used by Kim is different than those used in our simulation.

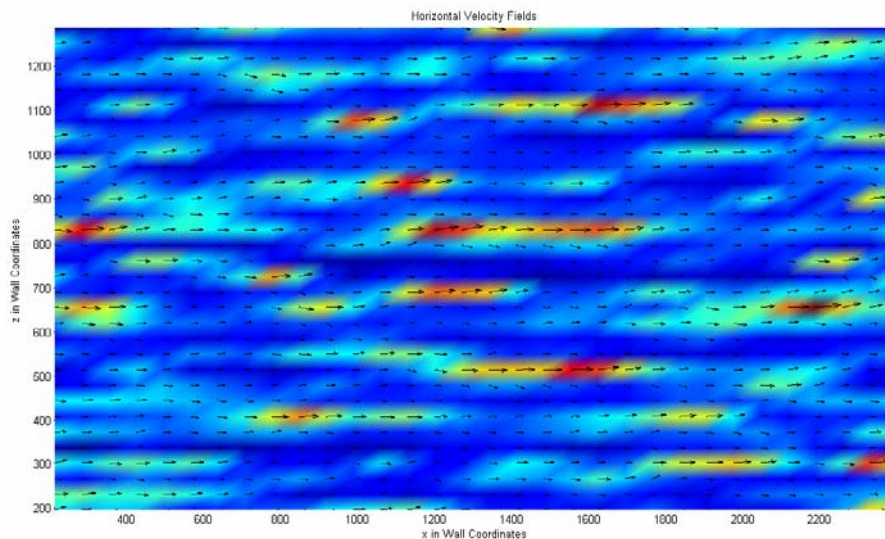


Figure 2.10: Streaks.

‘ERCOFTAC’ stands for “European Research Community on Flow, Turbulence, And Combustion”. Kim (Kim et al., 1987) data can be downloaded by browsing to the following link:

<http://cfm.me.umist.ac.uk/ercofold/database/test32/test32.html>.

Figure 2.11 shows the $u^+ - y^+$ profile comparison. The Kim profile, produced from 65 nodes ($129/2=65$), is shown in figure 2.11 and crosses on the line show where the actual data exist. The other line is the simulation output, consisting of only 16 nodes ($32/2=16$). It is clearly seen that both graphs, except in the near wall region, match each other quite perfectly. The near wall region contains the most variations; thus, more nodes would be needed there.

Figure 2.12 compares the normalized Reynolds shear stress or $\frac{\langle uv \rangle}{u_\tau^2}$. Again there is a very good match between these two graphs. Note that Kim (Kim et al., 1987) used 3962880 cells while in the current simulation just 32768 cells have been used. This means that Kim (Kim et al., 1987) used almost 121 times more grid cells. In other word, this is too costly for practical purposes and one of the motivations for our research here.

Finally, the variation of U_{rms} , V_{rms} , and W_{rms} with y^+ is given in Figure 2.13. These parameters are the variances of the u , v , and w components of the velocity respectively. Near the wall, there are more variations as can be clearly detected from the graph.

The turbulent energy variations may also be of interest. In a turbulent flow this energy fluctuates in time, but the mean value should be constant in our application. In Figure 2.14, the energy variation is given. The sampling of data is one per 50 dimensionless time units. Therefore, it does not seem to fluctuate a lot. If the data was sampled more frequently, more rapid variations may have been seen. However, the behaviour does point out the constant long-time averaged value and only small variations in its actual value.

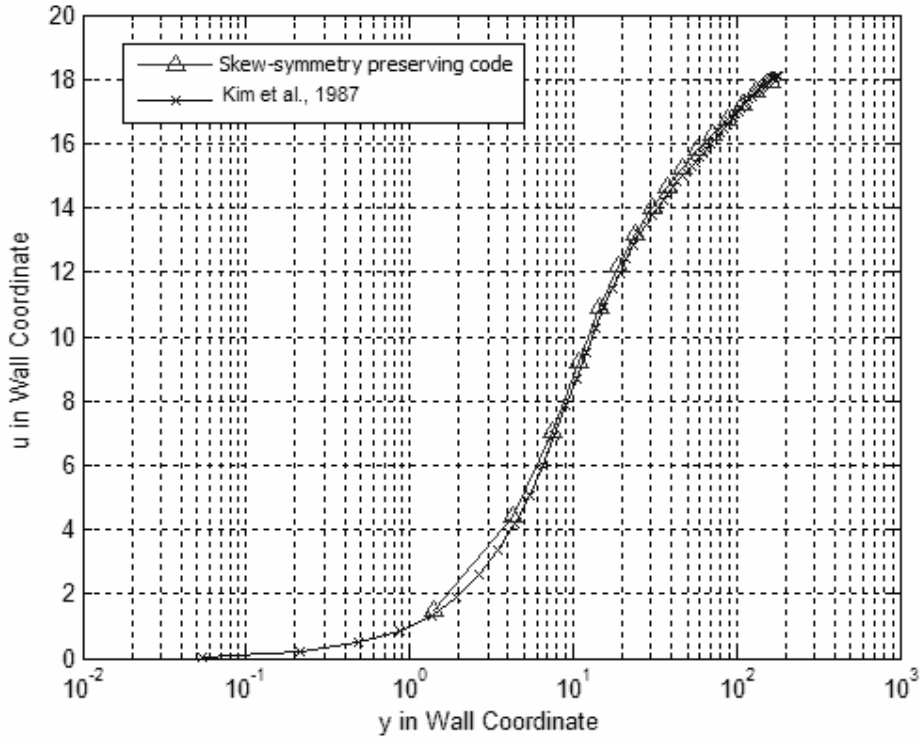


Figure 2.11: Comparison of two profiles

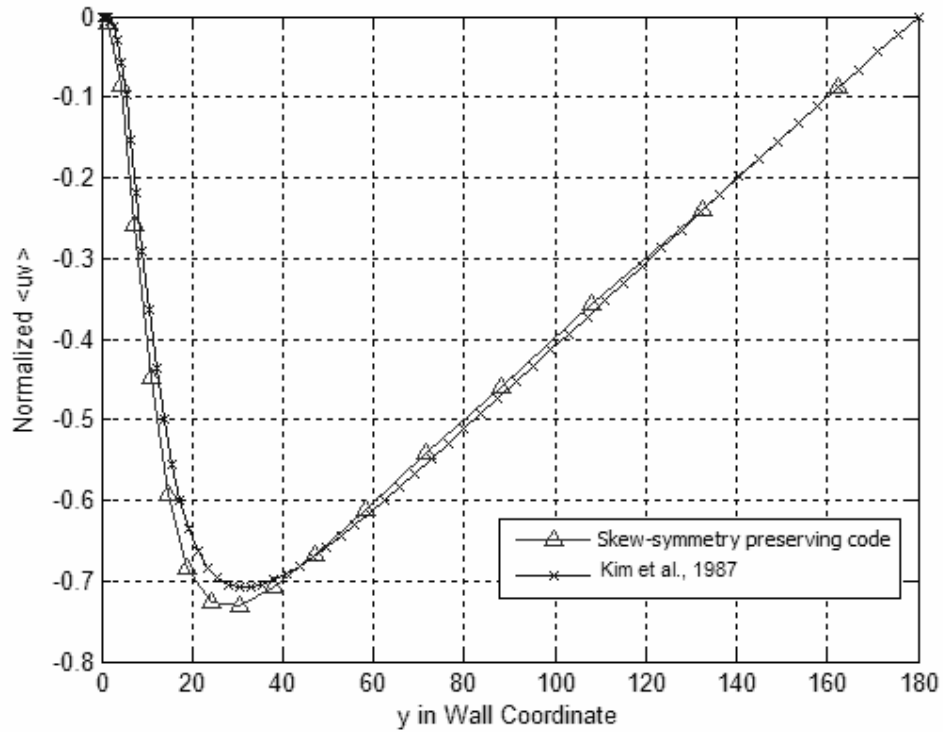


Figure 2.12: Comparison of normalized Reynolds shear stress.

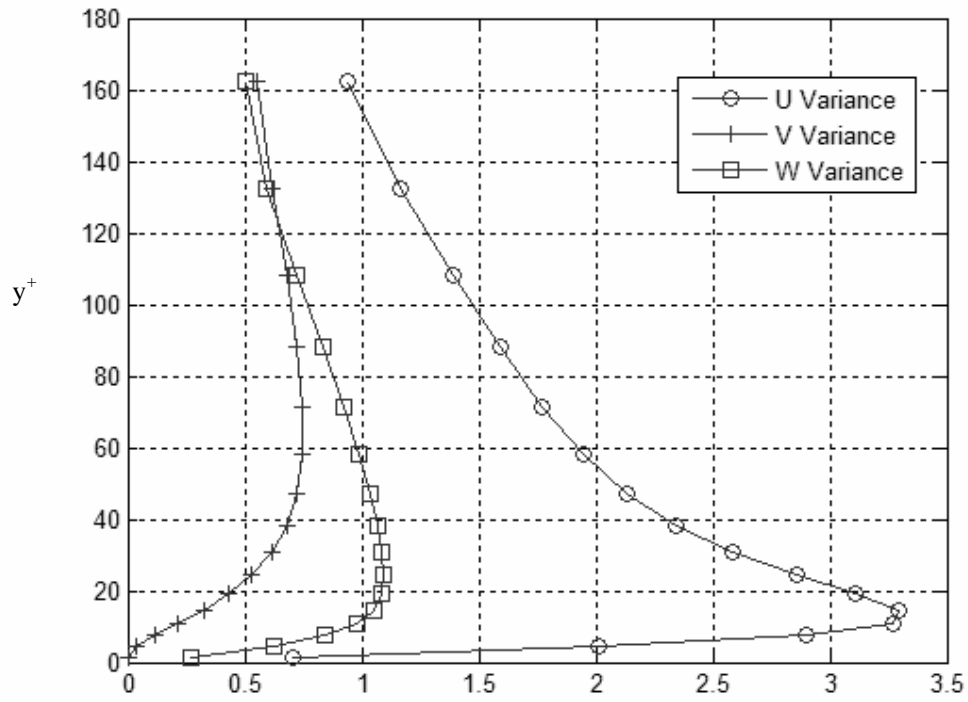


Figure 2.13: Variance of u,v, and w components of velocity

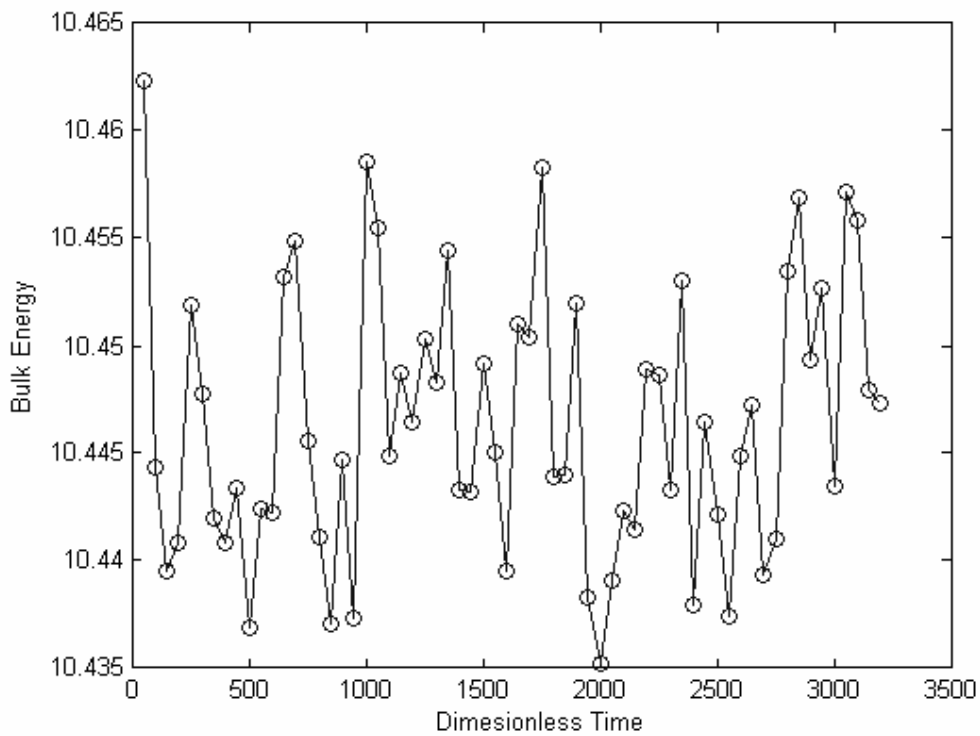


Figure 2.14: Bulk energy variation

2.6. Law of the Wall and Introducing Wall Modelling

In this section the law of the wall is discussed. The theory and the measurements (Pope, 2000) support that the velocity in the outer region of the fluid, when the condition $y^+ > 50$ is satisfied, obeys a logarithmic law given by:

$$u^+ = \frac{1}{\kappa} \ln(y^+) + C \quad (2.17)$$

Here κ is the ‘‘Von Karman’’ constant, which is reported to be between 0.36 and 0.42 in the literatures. Recent measurements by Zanoun (Zanoun et al., 2003) show that the best value for this constant is $\kappa = \frac{1}{e} = 0.3679$. But here, a value of 0.4 was used. The ‘‘+’’ implies that the velocity and distance from the wall are in wall units or wall coordinates, which are defined as:

$$u^+ = \frac{u}{u_\tau} \quad (2.18)$$

and

$$y^+ = \frac{yu_\tau}{\nu} \quad (2.19)$$

A closer inspection of the y^+ equation shows that it represents the local Reynolds numbers at each level, i.e. y . The velocity u_τ is the ‘‘friction velocity’’ and is defined as:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \quad (2.20)$$

In this case u_τ is equal to 180 and τ_w is the shear stress at the wall, which is defined as the sum of the viscous stress and Reynolds stress or:

$$\tau = \rho\nu \frac{d\langle u \rangle}{dy} - \rho\langle uv \rangle \quad (2.21)$$

Where $\langle \dots \rangle$ means an average value over a large enough patch of the flow domain. Using equation (2.21) and remembering that near the wall $\rho\langle uv \rangle$ can be neglected, the shear stress at the wall can be defined as:

$$\tau_w = \tau(y=0) = \rho\nu \left(\frac{d\langle u \rangle}{dy} \right)_{y=0} \quad (2.22)$$

Defining Roughness Parameter

The constant C in expression (2.17) is the constant produced by integration and in wall-modelling it is held responsible for the effect of wall roughness. It is reported to be between 4.8 and 5.4 for a smooth wall. Here, C was assumed to be 5.2. When the wall becomes rougher, this value will decrease. To understand this parameter, (2.17) can be arranged in a form which is more common to the meteorologists. Commonly, meteorologists report this value differently (Raupach et al., 1991; Wooding et al., 1973). They assume that $C = -\frac{1}{\kappa} \ln(y_0^+)$ and they report y_0^+ as roughness length. In that case, (2.17) can be written as:

$$\langle u \rangle = \frac{u_\tau}{\kappa} \ln\left(\frac{y^+}{y_0^+}\right) = \frac{u_\tau}{\kappa} \ln\left(\frac{y}{y_0}\right) \tag{2.23}$$

which defines y_0 as roughness length scale. The rougher the wall, the higher this parameter would be. Shortly speaking, y_0 defines the elevation or height where the wind velocity is assumed to be zero or in touch with the ground. Now by examining the above expression, it is obvious that the roughness length scale shows an average height where the wind velocity becomes zero. Instead of taking all the undulations of the earth surface, meteorologists define an average height and assume that the wind velocity is zero at that height instead of real height of the earth surface, Figure 2.15. In this case, that $C = 5.2$ and $\kappa = 0.4$, y_0^+ equals 0.1249.

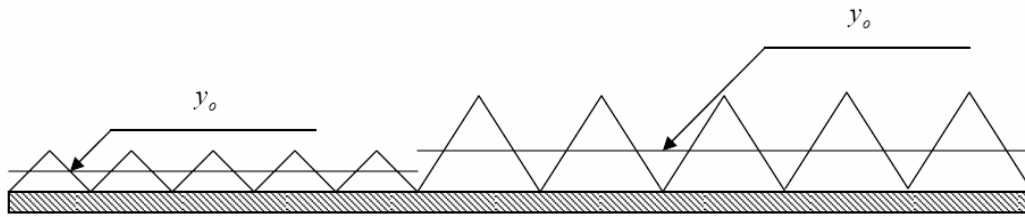


Figure 2.15: Definition of roughness length scale

Here, in the developed channel flow, the wall or the surface was assumed to be smooth. It means that there was no roughness introduced, i.e., the surface was assumed to be completely flat. But in real nature surfaces are not smooth and many natural and artificial undulations exist on the surface. Natural undulations are like different crops, different material, and natural topographical undulation. The artificial undulations are directly or sometimes indirectly related to human activities, Figure 2.16.

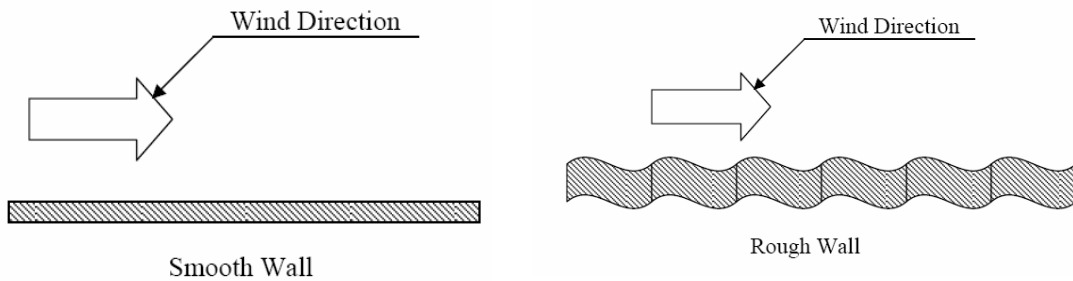


Figure 2.16: Smooth and rough Wall

The flow of a fluid over a smooth wall differs from the flow over a rough wall. Since the fluid in touch with the surface still has zero velocity, it is possible to use the same method to model rough walls. But it may be more appealing to modify boundary condition procedure. Although the boundary condition procedure in this case will be much more complicated but it will introduce no disadvantage. The real problem with this approach arises from some other reason.

In order to perform DNS simulation the grid cells should be fine enough to capture all eddy sizes. The number of grid-cells is proportional to the ratio of the largest eddy to the smallest one, i.e.

$N_x N_y N_z \propto \text{Re}^{\frac{9}{4}}$, (Piomelli and Balaras, 2002; Piomelli et al., 1989). To be more efficient, the grid cells are made finer near the wall and fewer grid cells, which are much coarser, are put farther away from the wall. This is a natural approach, since eddies close to the wall are much smaller than those farther away from the wall. Thus, a finer grid is required to resolve the near wall region. Using a finer grid requires more grid cells to be used to discretize the same volume. Moreover, smaller grid cells imply smaller time-steps. This means that the numerical scheme needs more memory, more processing power, and time on computers. In simple words, it becomes costly.

Chapman (Chapman, 1979) has proposed that the number of grid cells in the near wall region is proportional to $\text{Re}^{1.8}$. Therefore, by eliminating the grid-cells in this region, memory and processing power requirement will be reduced a lot. Piomelli (Piomelli and Balaras, 2002) has used the Chapman estimation for number of grid cells and concluded that at the Reynolds number of the order of 10^6 ; almost 99 percent of the grid cells have to be located in near wall region which is just 10 percent of the entire boundary layer. He has also compared the capabilities of a Pentium III processor with one gigabytes of memory to the grid requirements at different Reynolds numbers. This comparison is shown in Figure 2.17. The dashed line in this picture shows the amount of grid cell requirement in inner region. It can be seen that it grows very fast when the Reynolds number is increased. On the other hand the dashed-dotted line represents the grid cell requirement for the outer region. Although, it grows by increasing Reynolds number; but, the growth is much slower. The sum of these two lines is shown in bold-solid line. The gray-solid line shows the capacity of a Pentium III processor with 1GB memory. It can be seen that even with a moderate Reynolds number, the capacity of this computer is much lower than total requirements.

It is obvious that at the higher Reynolds numbers, eliminating the near wall grids will decrease the simulation cost drastically and even at very high Reynolds number no simulation is possible without eliminating the near wall region. The process of eliminating the near wall grid cells and using a model is called wall modelling, i.e., modelling the effect of the wall and using it at some distance farther away than the wall itself.

In this research, it is desired to eliminate the inner layer and have the first grid points in the outer layer, which obeys the log law, Figure 2.18. At first, one may think that it is very easy to do so, because the log law is known and it has been confirmed by many numerical and experimental methods. Thus, using the log law, the boundary condition can easily be implemented in the numerical scheme.

But there is a big drawback in these assumptions. The log law is a function of distance from the wall, the roughness length scale, and the friction velocity. The distance from the wall is definitely known. The roughness length scale can be measured with a high resolution digital elevation model of the earth or DEM, or it can be estimated and approximated via land cover maps resulting from classification of satellite images. But the friction velocity requires information from the inner layer, (2.22) which, however, is exactly what we wish to be eliminated. So, on one hand the near wall region has to be eliminated to make the code faster, while on the other hand it is required in calculating the friction velocity.

Therefore, in this thesis we use the information available in the log layer or the outer layer to have an approximation of the friction velocity without using any information from the inner layer, Figure 2.19. Later, this estimated friction velocity is used to calculate the velocity at the elevated boundary and is applied to the simulation domain as boundary condition. Several formulations are possible. By writing expression (2.17) in terms of the velocity at two y-locations, the well-known result among meteorologist, can be derived (Lee et al., 2005):

$$\overline{u}_\tau = \frac{\kappa(\overline{u_{r2}} - \overline{u_{r1}})}{\ln\left(\frac{y_2}{y_1}\right)} \quad (2.24)$$

where:

$$u_{rj} = \sqrt{u_j^2 + w_j^2} \quad (2.25)$$

The symbol $\overline{\dots}$ indicates an appropriate averaging operation which may involve both time and space. Several averaging method was used which are discussed in chapter 4.

In Chapter 4, we introduce each model in more detail and they are tested over a smooth wall. In chapter 5, these wall models are tested over homogenous rough wall. Later, in chapter 5, a rough surface with one and 4 patches is tested.

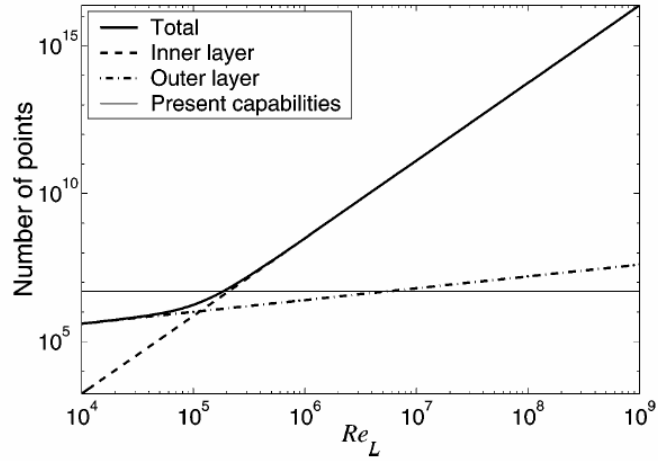


Figure 2.17: Number of required grid cells (adopted from Piomelli et al., 2002)

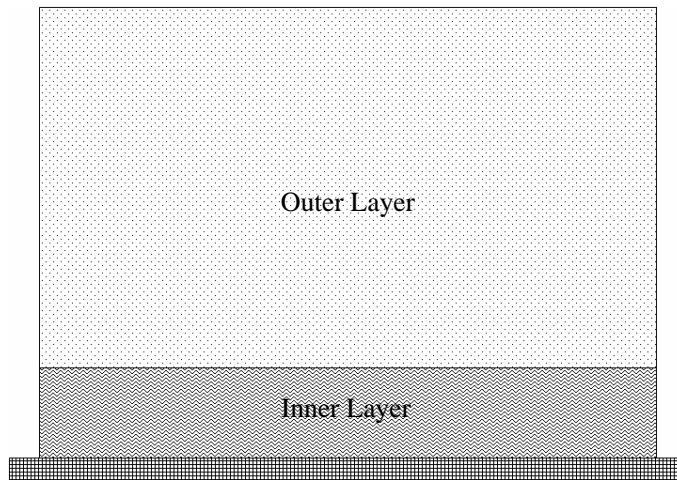


Figure 2.18: Full grid

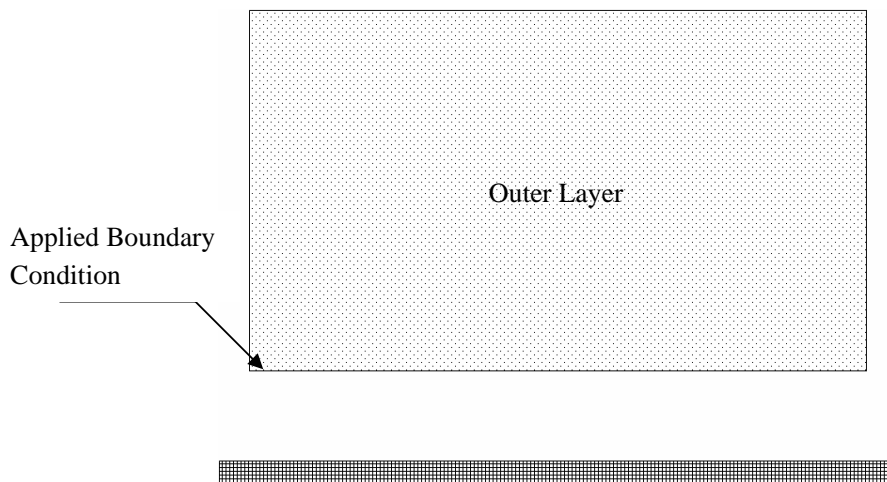


Figure 2.19: Reduced grid. Fine grid cells near the wall are removed.

2.7. Summary

In this chapter the code was applied to a channel flow and the results were inspected both qualitatively and quantitatively via comparing with available data and theoretical data. The code was studied and also different aspects of numerical analysis of fluids were presented. It was shown that the code based on the symmetry-preserving discretization is able to accurately simulate a channel flow even with a very coarse grid. The current code, as promised in chapter one, was also validated using Taylor-Green class of vortices, which is presented in appendix B. Later the law of the wall was introduced. This law is later used in developing a faster model.

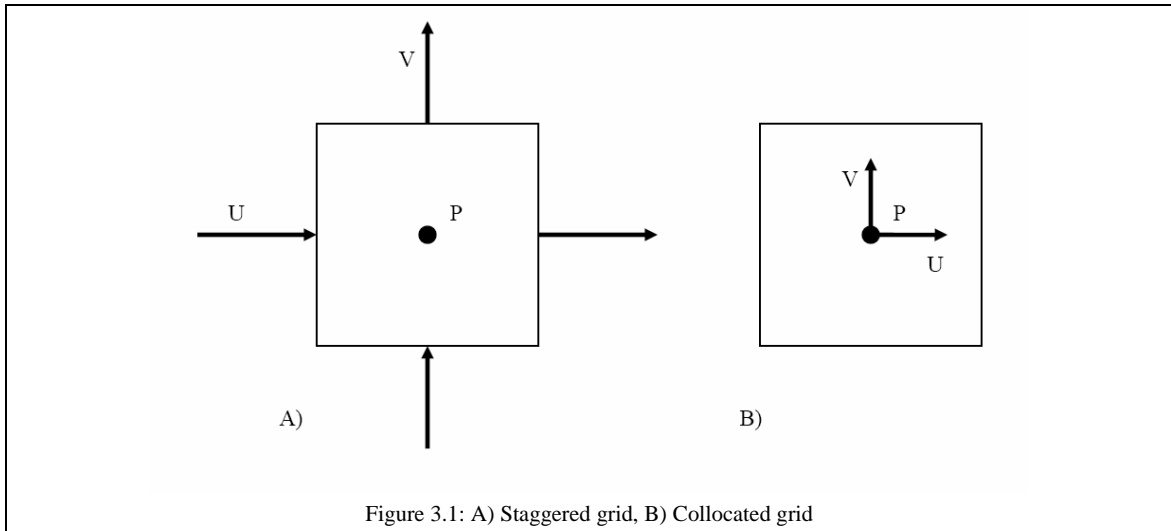
As the emphasis of this research is on atmospheric boundary layer flows, next we make some adjustments and alterations in the code to simulate the boundary layer. In the next chapter, differences between the channel and the boundary layer flow are discussed. We also describe how the code was modified to perform boundary layer simulation and the simulation results are presented.

3. Boundary Layer Flow over Smooth Wall

3.1. Boundary Flow

In the previous chapter, the channel flow was discussed. In channel flow, as the fluid is in touch with the walls at both sides, the velocity has to become zero close to the solid walls and the velocity is maximal in the middle of the channel.

In our simulation a staggered grid was used. In general there are two types of grid being used in computational fluid dynamics (Anderson, 1995). The first one is called collocated grid, where the velocities and pressures are defined at the same location and in the actual grid points, and the second one is the staggered grid, where the velocities are defined at the middle point of the faces of a grid cell and the pressure is defined at the centre of a grid cell. Therefore none of the u , v , w , and p are collocated; see Figure 3.1A for the staggered grid and Figure 3.1B for the collocated grid.



As a staggered grid is used in the simulations, the first component of the velocity, u , is located at $\frac{\Delta y_1}{2}$ away from the wall, where $\Delta y_1 = y_1 - y_0$. Therefore the first computed u is non-zero but very close to zero, Figure 3.2. The same applies for the spanwise velocity w while the wall-normal velocity component is defined at the wall.

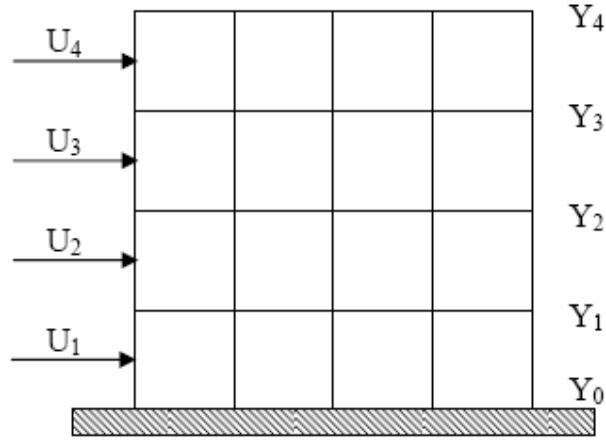


Figure 3.2: Location of the First 'u' Relative to the Wall

The implementation of the no-slip boundary condition requires that the velocities are extended in the dummy-locations inside the wall, using an odd function of the wall normal coordinate. In other words, they are extended negative-equal with respect to the wall; see Figure 2.2. However, the boundary layer flow, such as the atmospheric flow, is not confined by walls at both sides. Compared to the channel application, the far-field boundary condition has to be altered to accommodate a boundary layer. We consider this next.

The lower part of the flow is indeed in touch with the wall, or the earth surface in atmospheric flows, and therefore the same boundary condition can be applied as in the channel flow, i.e., no-slip boundary conditions:

$$\begin{cases} u_{wall} = 0 \\ v_{wall} = 0 \\ w_{wall} = 0 \end{cases} \quad (3.1)$$

The upper part of the boundary layer is not confined by any walls; therefore, the no-slip boundary condition cannot be applied. Instead we adopt the free-slip boundary condition to approximate the flow behaviour far away from the earth's surface. The free slip boundary condition implies that the derivatives in the wall normal direction, of the streamwise and spanwise velocity components are equal to zero (Masson et al., 2006). In addition, the wall-normal velocity-component needs to be set to zero at the upper layer. The free-slip boundary condition can be summarized as:

$$\begin{cases} \left(\frac{du}{dy} \right)_{y=y_N} = 0 \\ v_{y_N} = 0 \\ \left(\frac{dw}{dy} \right)_{y=y_N} = 0 \end{cases} \quad (3.2)$$

Here y_N denotes the location of the upper boundary. Remember that u is the streamwise velocity component, w the spanwise velocity component and the v is the vertical velocity component. Thus, the same numerical formulation is used for the v component as in the no-slip condition, but u and w have to be implemented differently.

To implement the free-slip boundary condition, an even function is used for both u and w components to set the wall normal derivative of u and w to zero. This is illustrated in Figure 3.3. As can be seen, for the free-slip boundary condition, this means:

$$\begin{cases} u_{n+1} = u_n \\ u_{n+2} = u_{n-1} \\ u_{n+3} = u_{n-2} \end{cases} \quad (3.3)$$

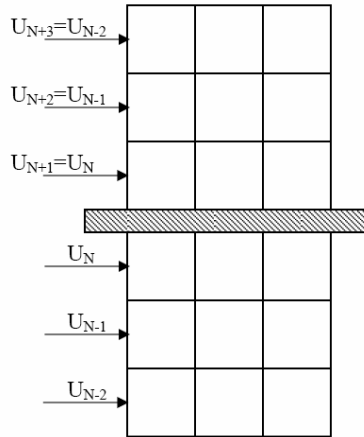


Figure 3.3: Numerical Implementation of Non-Slip Boundary Condition

3.2. Simulation Domain Description

To perform boundary layer simulations, a channel with its lower boundary confined to the wall, or the earth, was chosen. The streamwise length was chosen to be 4π and the spanwise length was chosen to be $\frac{4\pi}{3}$. The main extension, compared to the channel flow, is that the boundary height was increased to be able to represent an ‘infinitely’ thick boundary layer, not confined by an ‘upper-lid’. In the vertical direction at the lower wall, where the flow is in touch with the wall or the earth, the no-slip boundary condition was used and at the top boundary the free-slip boundary condition was chosen.

The Reynolds number was chosen to be equal to 10000, and the time-step was set to $dt = 1.25 * 10^{-2}$. In the rest of this subsection we describe the choices made for the numerical parameters and variations considered during validation of the approach.

Simulation Height and Grid Generation

Different heights of the simulation domain were checked. The stream-wise and span-wise length was chosen to be the same in all cases but the height was chosen to be 2, 10, 20 or 30, which are referred as L2, L10, L20, and L30 in this text.

To be able to compare the results of these different simulations, the grid was generated for the L20 case. Then for the simulation case L10, some nodes were taken out and for the simulation case L30 some nodes were inserted manually in the y direction without changing the distance between them. After this construction, all the y -values match each other perfectly; see Figure 3.4. The grid for simulation for case L2 was generated separately using the same equation as noted in chapter 2, i.e. equation 2.4, but without mirroring. As in the case of channel flow, all generated grids, in the x and z directions, are equidistant but in the y direction a finer mesh is used closer to the wall. Farther from the wall the grid becomes coarser.

It is known that if a finer mesh is used in the vertical direction, a better result can be obtained. To study the effect of the horizontal mesh size a grid with the same vertical resolution was generated but with coarser horizontal resolutions. Three different horizontal resolutions were selected, 1) $32*32$, 2) $16*16$, 3) $8*8$, Figure 3.5. It was seen that with $16*16$ horizontal resolution almost the same results can be obtained.

Time Step

The time step was chosen to be $dt = 1.25 * 10^{-3}$ for the channel flow. As a rule of thumb, the minimum grid length divided by the maximum velocity gives an indication of the maximum allowable size of the time step. This condition implies that the signals will not travel over more than the distance of one grid-cell, the numerical scheme will not crash, and the results follow the solution in time quite accurately. To be on the safe side, a fraction f of 0.7 or 0.8 of this maximum time step is used. The maximum allowable time step size can be written as:

$$dt_{\max} = f \frac{\min(\text{Grid.Length})}{\max(\text{velocity})} \quad (3.4)$$

Expression (3.4) represents the well-known CFL condition; it is the basis for deciding to have $dt = 1.25 * 10^{-2}$. To investigate how the simulations behave with a larger time step, two different simulations were performed. All the parameters were the same in these simulations except for the time step. In one simulation the time step was set to $dt = 1.25 * 10^{-3}$ and in the other one it was set to $dt = 1.25 * 10^{-2}$. Both simulations were run for 200 dimensionless time units. Of course, the number of time steps that have to be passed to reach 200 is also different in these two simulations. In

the first one, the number of time steps is $n_t = \frac{200}{1.25 * 10^{-3}} = 160000$ and in the second simulation it is

$$n_t = \frac{200}{1.25 * 10^{-2}} = 16000 .$$

It was seen that the u velocity follows the same average and pattern in both simulations. Therefore, after increasing the time step by a factor of 10 compared to the time step used originally in the channel flow, the results are still quite accurate but can be obtained much faster. It also seems that the simulation with a larger time step is going to converge much faster. As in second simulation, the time step is chosen to be larger; thus, small phenomenons, that have shorter life time than the selected time step, can not be seen in the results. In other words, the simulation results are somehow averaged over the time step.

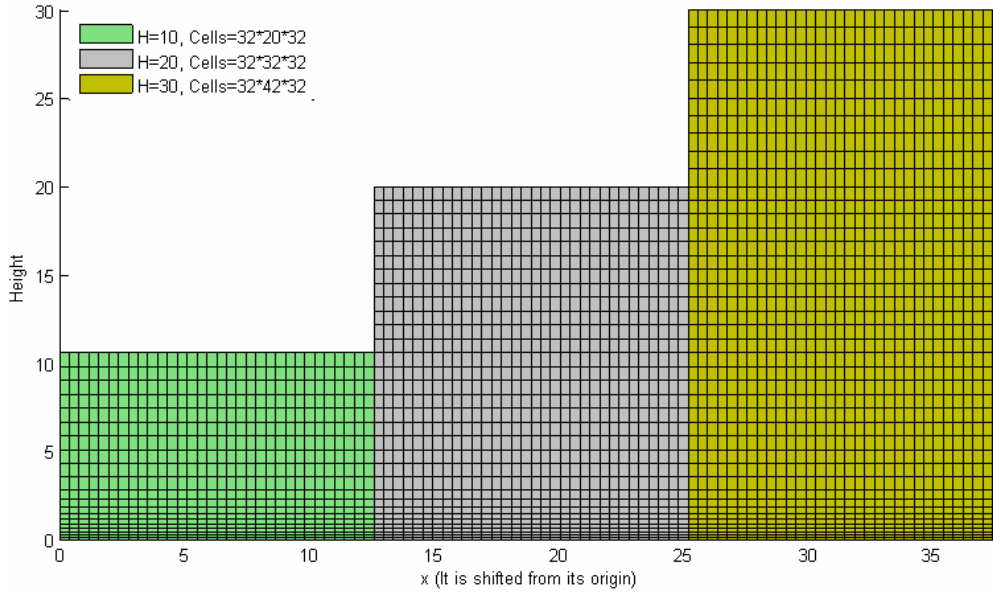


Figure 3.4: L10, L20, L30 grid

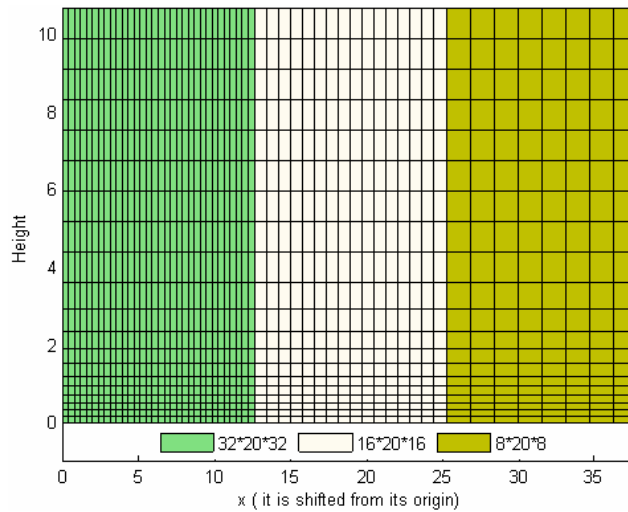


Figure 3.5: Grid in different horizontal resolution

3.3. Qualitative and Quantitative Inspection of Simulation Result

In this section one case is selected for a detailed study. Several other cases were investigated as well, globally confirming the findings presented here. The Reynolds number and the height of selected boundary layer are as follows:

1. $Re = 10000$
2. $H = 10$

To monitor the convergence of long time averaging the variation of the running average of the Reynolds number based on u_r was considered, the same way as in chapter 2. This parameter and its time derivative are shown in Figures 3.6 and 3.7. Other factors, which can help to decide whether a simulation has converged or not, are the L2 norm of the U_{rms} , V_{rms} , W_{rms} , and uv , as it was considered in the channel flow. It was seen that in all cases the convergence of the long-time averaging was quite satisfactory; however, they are not shown here. It was decided that at least 8000 dimensionless time unit has to be passed before the numerical scheme is converged. The jump in all Figures around 5000 is due to restarting, as in chapter 2.

Figure 3.8 shows the variances of the u , v , and w profiles. It can be seen that the variance of u starts to increase again near the top layer (right hand side of Figure 3.8). It can be also seen in Figure 3.11 that the profile starts to deviate from the theoretical line, which can also justify the increase of variance in Figure 3.8

The spanwise velocity profile is shown in figure 3.9. It can be seen that this value is very small; therefore, as expected, the dominant flow is in x direction.

The mean vertical velocity should be zero, as shown also by measured data, and it can be seen that it is indeed a very small value of the order of $O(10^{-11})$, Figure 3.10. Figure 3.11 shows the simulated result against the logarithmic and linear theoretical line, discussed in chapter 2. They have perfect match except at the top. This deviation is partly due to coarse resolution at top of the boundary layer. This test is the only quantitative test that has been done and shows that the simulation results are matching the theoretical predictions quite well.

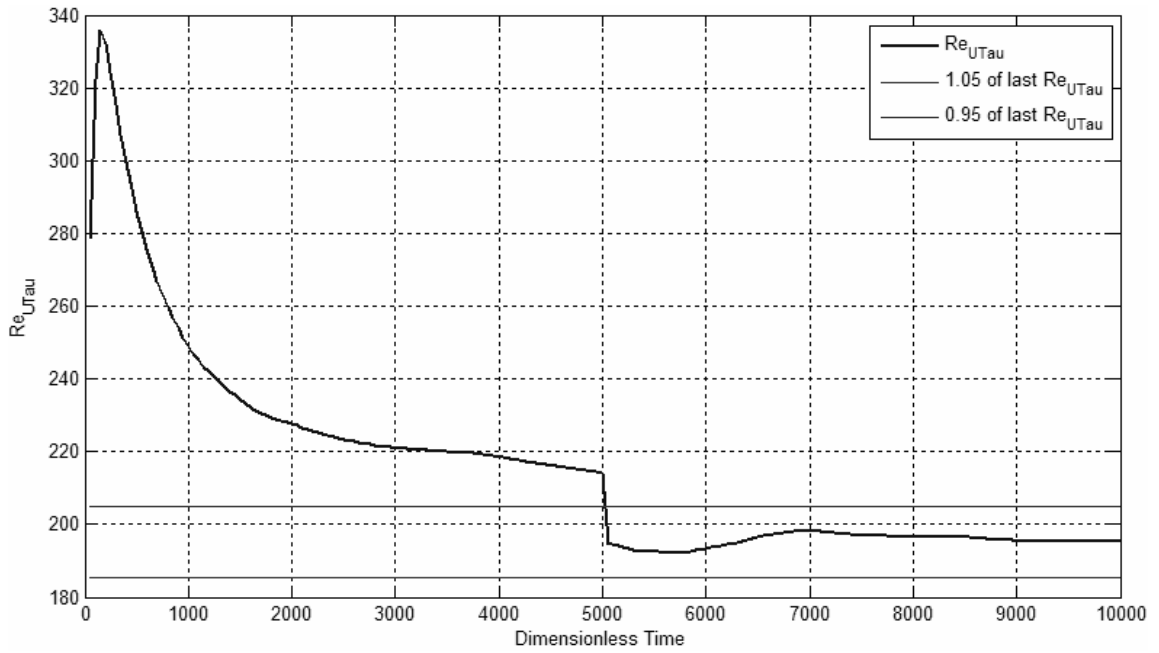


Figure 3.6: Variation of Reynolds number based on friction velocity

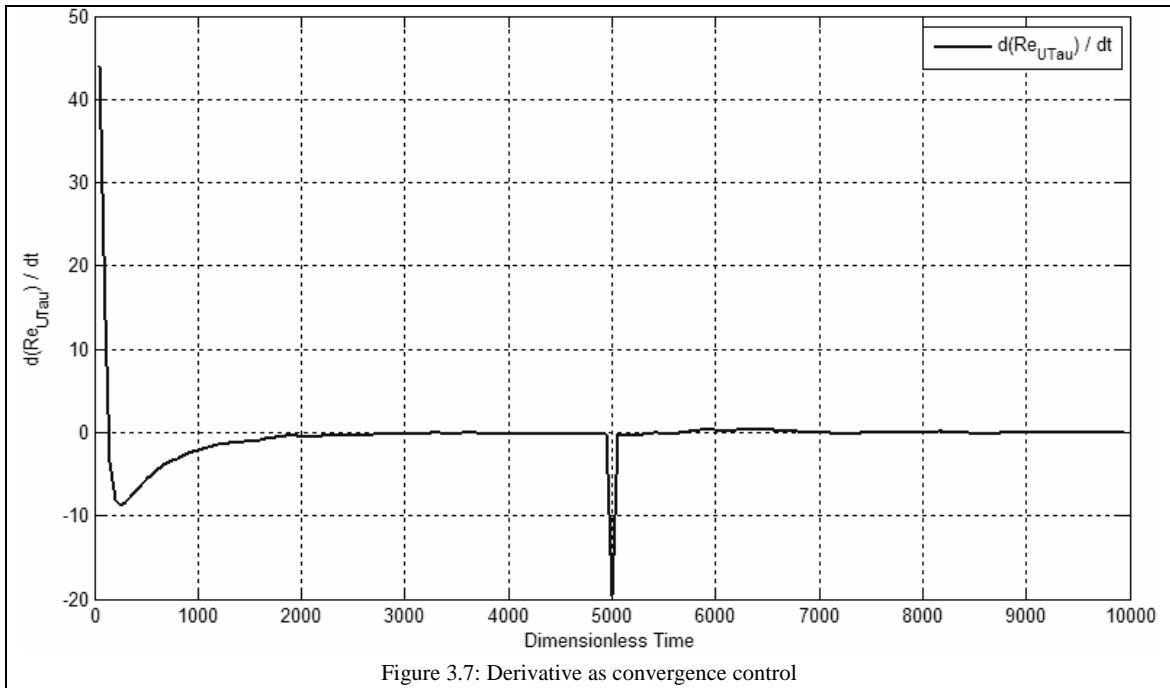


Figure 3.7: Derivative as convergence control

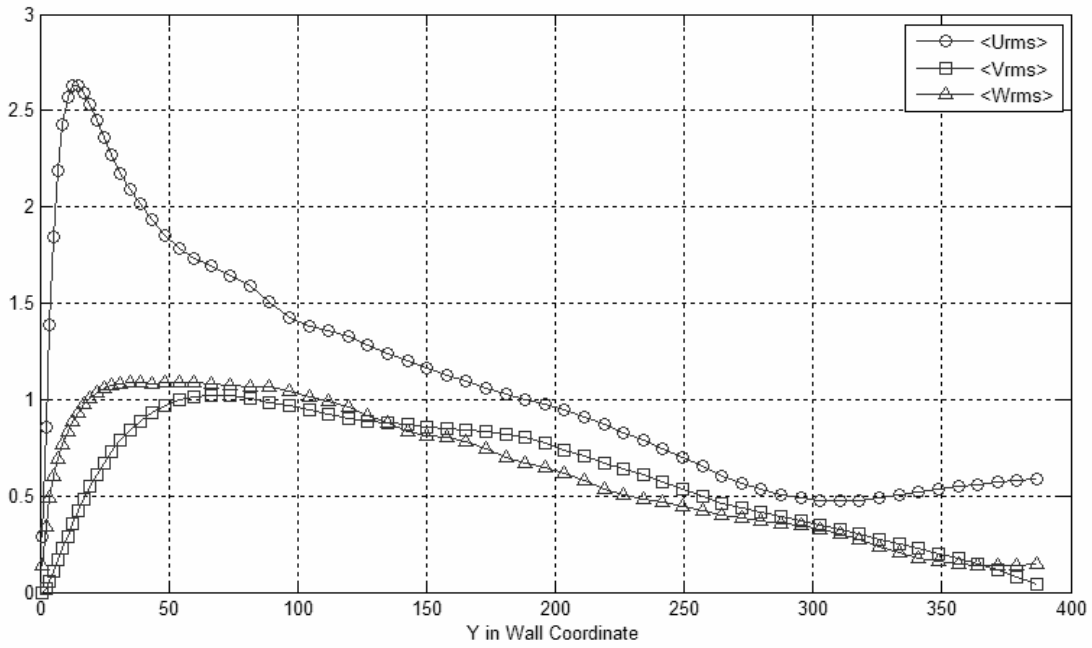


Figure 3.8: RMS profile for three components of the velocity

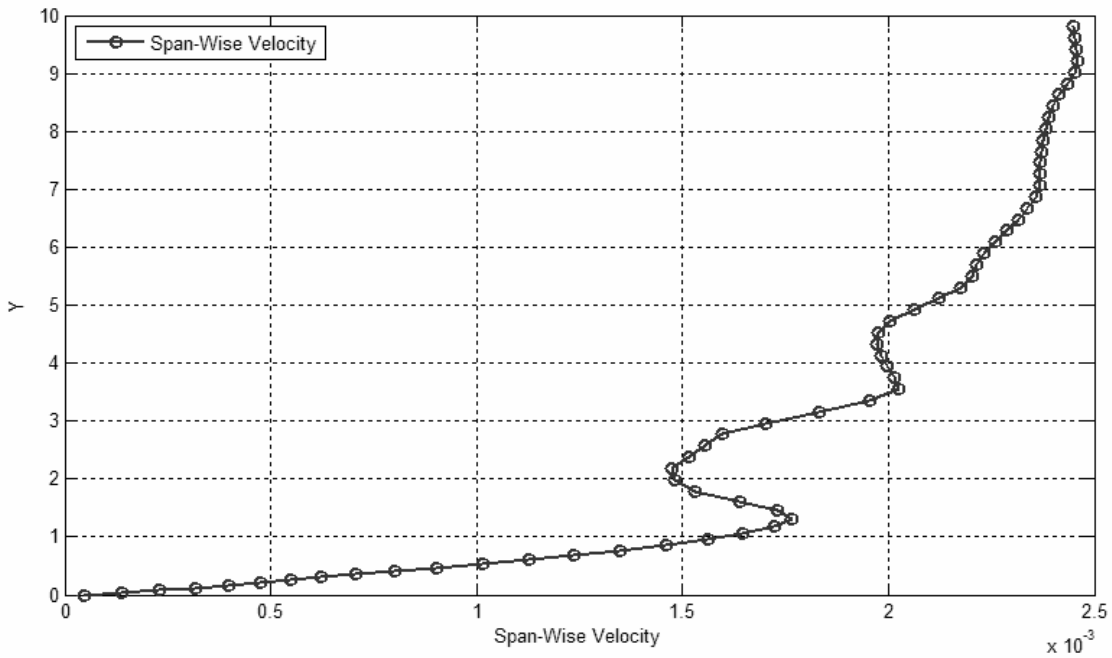


Figure 3.9: Spanwise velocity profile

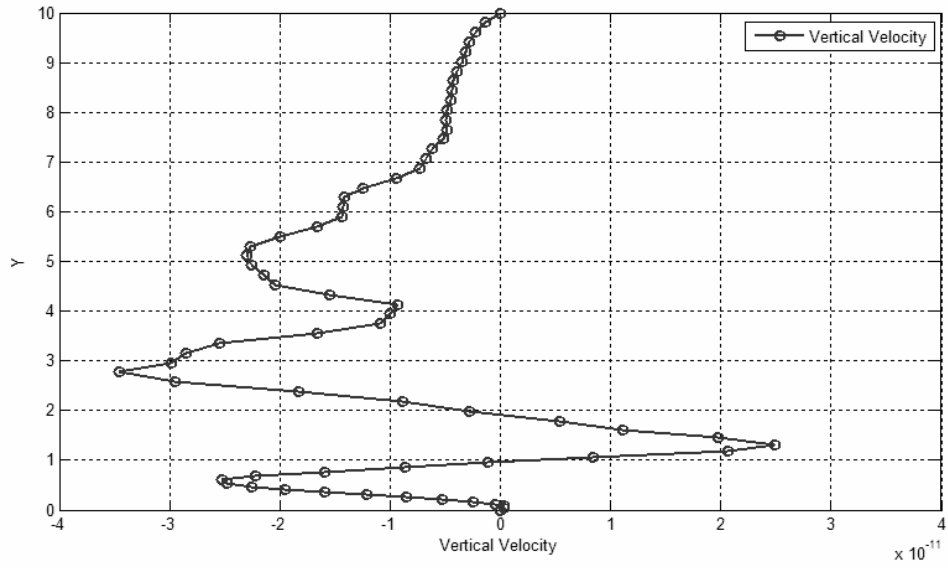


Figure 3.10: Vertical velocity profile

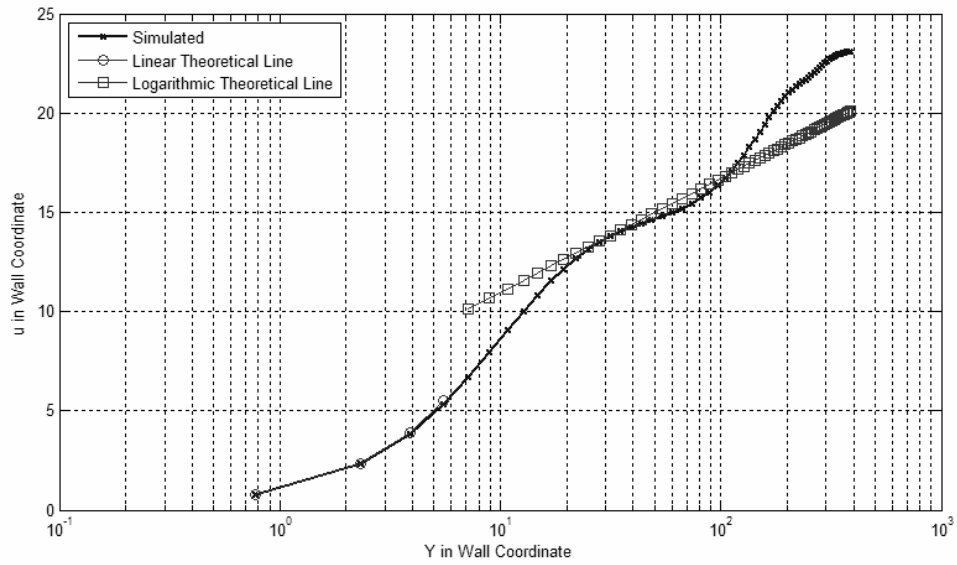


Figure 3.11: Comparison of simulation result with theoretical lines

3.4. Summary

In this chapter the code-adjustments required for boundary layer simulations were introduced and validated. For simplicity, among several cases that have been simulated, just one case was represented. The results were checked mostly in a qualitative manner but the simulation results matched theoretical results quite well. It can be concluded that the code is also able to simulate boundary layer flows.

But the code is still solving all scales of motion and therefore rather costly. Each simulation took about 6 days on a supercomputer located in Amsterdam. More than 20 simulations were performed and if the supercomputer was not able to run them in parallel, it was impossible to finish the simulations in time.

Therefore, in the next chapter, the code is combined with a wall model. The wall model helps to decrease the amount of grid cells by eliminating the fine grids at the wall; therefore, the simulation can be done much faster. In principle, this change also allows increasing the time-step. Besides, the requirements for the memory will decrease and it becomes possible to use regular PCs for such simulations.

4. Wall Modelling for Turbulent Flow over Smooth Wall

4.1. Wall Modelling Methods - History

In section 2.6 the law of the wall was introduced and wall modelling was defined. In this section, a brief review of wall modelling is given. Wall modelling, which is sometimes referred to as “wall-layer models” or “approximate boundary conditions”, can be categorized in two groups as (Piomelli et al., 2003):

1. Equilibrium Laws
2. Zonal Models

The main features of these approaches are discussed next.

4.1.1. Equilibrium Laws

In this category of models, it is assumed that a universal ‘law-of-the-wall’ applies for the dynamic properties of the wall layer. This law is based on an averaged velocity over a large enough patch. Piomelli (Piomelli and Balaras, 2002) has proposed that the grid spacing in horizontal direction should have a minimum size to capture some of the near wall streaks in the solution. In the streamwise direction it is suggested that the grid spacing should be of the order of 100 in wall units and in spanwise direction it should be of the order of 50 in wall units. The grid itself should be of the order of 1500 in wall units in streamwise direction and 700 in wall units in spanwise direction. If the grid and/or grid spacing is much finer than the above proposed sizes, the statistics on which the wall modelling is based, will usually fail. Therefore, when wall models are used, finer grids do not necessarily mean better and more accurate results.

This is an important factor which has to be taken into account even in some surface energy balance models, since they are also using these wall models intrinsically. Thus, using a satellite image with finer pixel size does not necessarily mean that better results will be achieved.

“Equilibrium Laws” finally leads to a linear equation relating mean wind velocity to the height near the earth surface and a logarithmic one farther away from the surface, discussed fully in section 2.6. These equations were successfully used in some simple cases and especially in meteorological applications. But it is very hard to apply them in cases for which the geometry of the domain is complicated, (Balaras et al., 1996).

4.1.2. Zonal Models

The zonal models, proposed by Balaras (Balaras and Benocci, 1994; Balaras et al., 1996) use the advantage of the RANS method, i.e. requiring less memory and processing power and being fast, in the near wall region or, more precisely, they use unsteady RANS equations in the near wall region. This means that the simulation extends to the wall where the simple and classical no-slip boundary condition is used.

Two different techniques can be employed here. One of them uses a completely different grid in the near wall region and in the outer region and is known as Two Layer Model or TLM and the other one uses a single grid but the turbulence model is changed in different regions. The latter is mostly known as Detached Eddy Simulation or DES invented and proposed by Spalart (Spalart et al., 1997).

Although the “Zonal Model” requires more memory and processing power than the “Equilibrium Law” approach, it is still much faster than methods without wall modelling. Besides, “Zonal Models” can be applied in complex geometries, where “Equilibrium Law” models can not be used.

Here due to lack of time this method is not studied further. Therefore, in the rest of this chapter and all over this research/thesis the equilibrium law is used.

4.2. Developed Wall Model

In this research several model, based on Groetzbach approach,(Groetzbach, 1987), which is an extended version of Schumann approach (Schumann, 1975), and can be classified under “Equilibrium Law” models, are presented. The basic equations are discussed in section 2.6. Several different averaging operators were studied in this thesis as follow:

1. Averaging over time and both streamwise and spanwise directions (M1).
2. Averaging over streamwise and spanwise directions (M2).
3. Averaging over time and spanwise direction (M3).
4. Averaging over time (M4).

We admit that the best wall model is the one which does not use any averaging. But since the log law is established on averaged values, if the instantaneous values are used, sometimes the log law fails and the model becomes unstable. The M1 model uses too much averaging; therefore, it is not able to show the surface changes properly. As proceeding to the bottom of the above list, i.e. the M4 model, the model depends less on averaging; thus, more spatial changes of the surface can be examined. In this section, each model is discussed in more detail.

4.2.1. Averaged Value over Time and Space (M1)

In this method, the velocities are averaged over time and space, i.e., averaged over a horizontal plane at each y-level and also averaged over time to yield the averaged velocities that were used in the equilibrium law equation to have an estimation of the friction velocity at the surface. Using

this estimated friction velocity and the equilibrium law, the velocity at the bottom of the outer layer is calculated and fed into the model as boundary condition, Figure 2.19.

Using previous simulation results, it was seen that the friction velocity can be calculated within one percent error using averaged velocities over time and space. As an example, for the simulation case in which the Reynolds number was set to be 10000 in chapter 3, the friction velocity was first calculated using the computed near wall solution. Then, using the same data set and (2.24) in the outer region the friction velocity was estimated again. It was seen that the model was able to calculate the friction velocity with 0.02 percent, i.e., much less than one percent error.

Now this plane-time averaged value has to be used for each grid-cell to have an estimate for local friction velocities. The mean friction velocity is related to its local value by:

$$u_\tau = \frac{u_r}{u_r} u_\tau \tag{4.1}$$

To have a directional local friction velocity, Schuman (Schumann, 1989) suggested that the ratio of velocities' component to the velocity can be used, i.e.:

$$\begin{aligned} u_{\tau x} &= \frac{u}{u_r} u_\tau \\ u_{\tau z} &= \frac{w}{u_r} u_\tau \end{aligned} \tag{4.2}$$

Now that the friction velocity for each grid cell is calculated, the velocity at the boundary, where the log-layer or outer region starts, was obtained using the estimated friction velocities and the log law equation. This method was tested with available data. It was seen that the estimated velocities were in the same range as they were expected to be. This proves that the output range of this method is perfect. The results and ranges are given in Table 4.1. As it can be seen, the model produces more error for minimum value; but, still the error is less than ten percent. For maximum values the model produces less error and the error is confined to less than three percent, except for estimating w , spanwise velocity, which the error is still less than ten percent.

Table 4.1: Modelled and simulated velocities

Velocity	Modelled	Simulated	<i>Simulated</i> <i>Modeled</i>	
u_r	Min	0.0315	0.0339	1.0749
	Max	0.0738	0.0725	0.9818
u	Min	0.0315	0.0338	1.0741
	Max	0.0738	0.0724	0.9815
w	Min	-0.0087	-0.0091	1.0537
	Max	0.0172	0.0157	0.9145

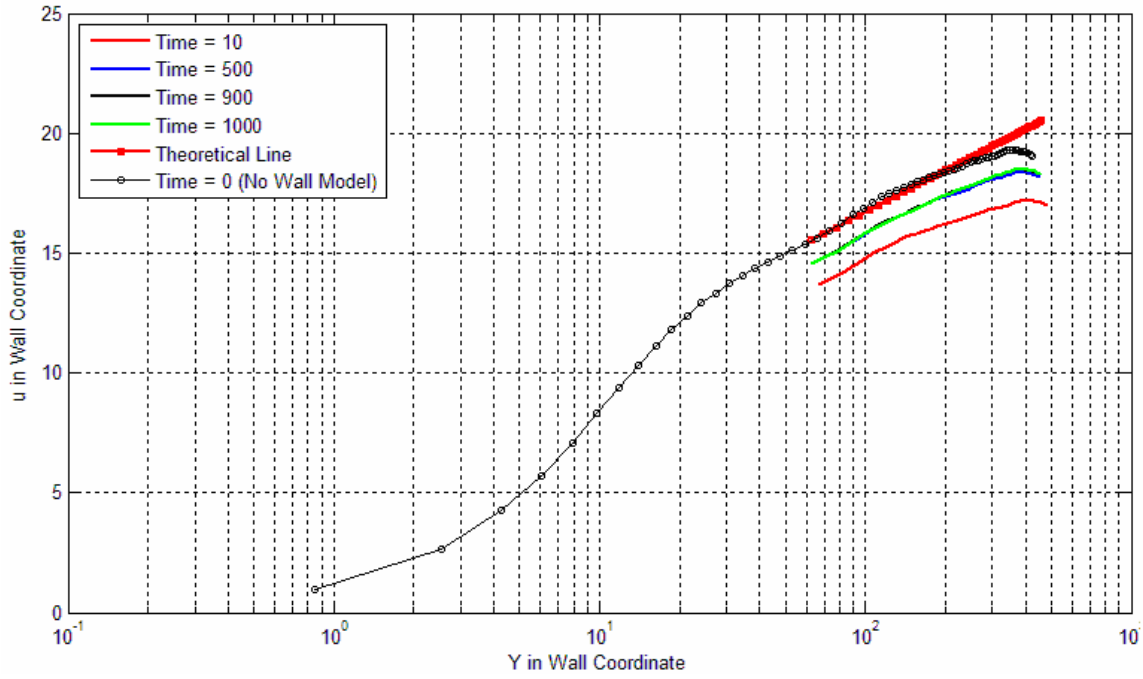


Figure 4.1: Wall modelled results

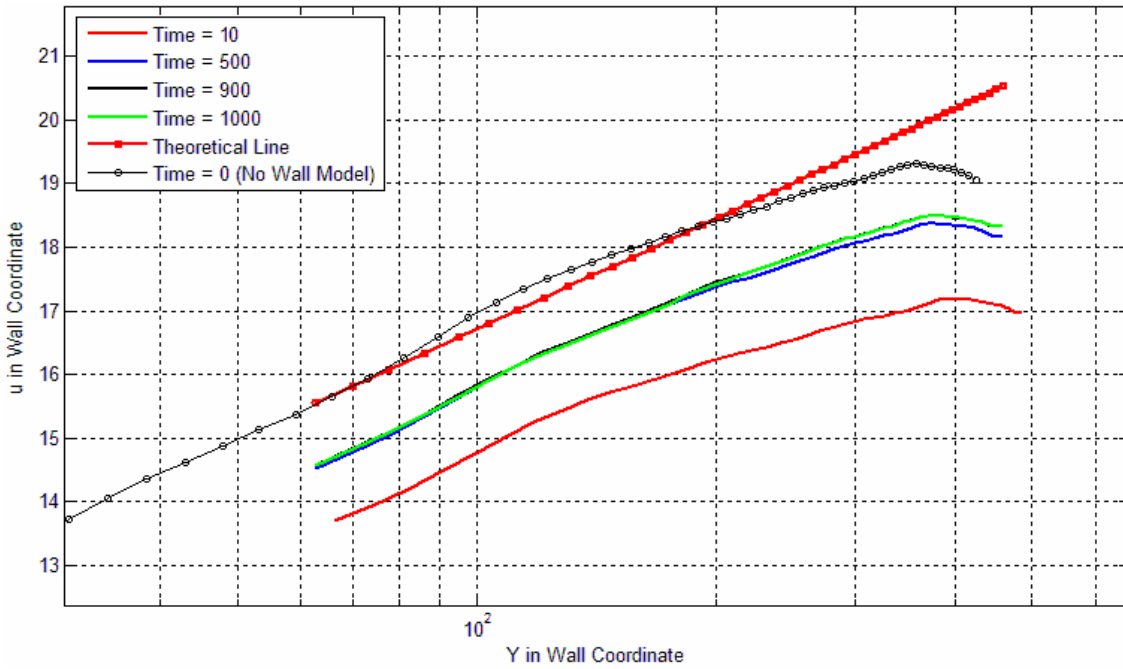


Figure 4.2: Wall modelled results (zoomed)

One more comment about this method is that it was decided not to use logarithmic law for spanwise velocities and it was seen that the following expression, whose outputs are shown in Table 4.1, gives better results:

$$w = \sqrt{u_r^2 - u^2} \tag{4.3}$$

Once the velocity at the boundary was calculated, i.e. at $y_{j=0}$ and by taking into account that a staggered grid is used, the velocity at the dummy-nodes was set by using a linear extrapolation as:

$$\begin{aligned} u_{i,0,k} &= 2u_0 - u_{i,1,k} \\ u_{i,-1,k} &= 2u_0 - u_{i,2,k} \\ u_{i,-2,k} &= 2u_0 - u_{i,3,k} \end{aligned} \quad (4.4)$$

The same method is applied for spanwise direction with this difference that $w_{j=0}$ was calculated using equation 4.3. The simulation result is shown in Figures 4.1 and 4.2.

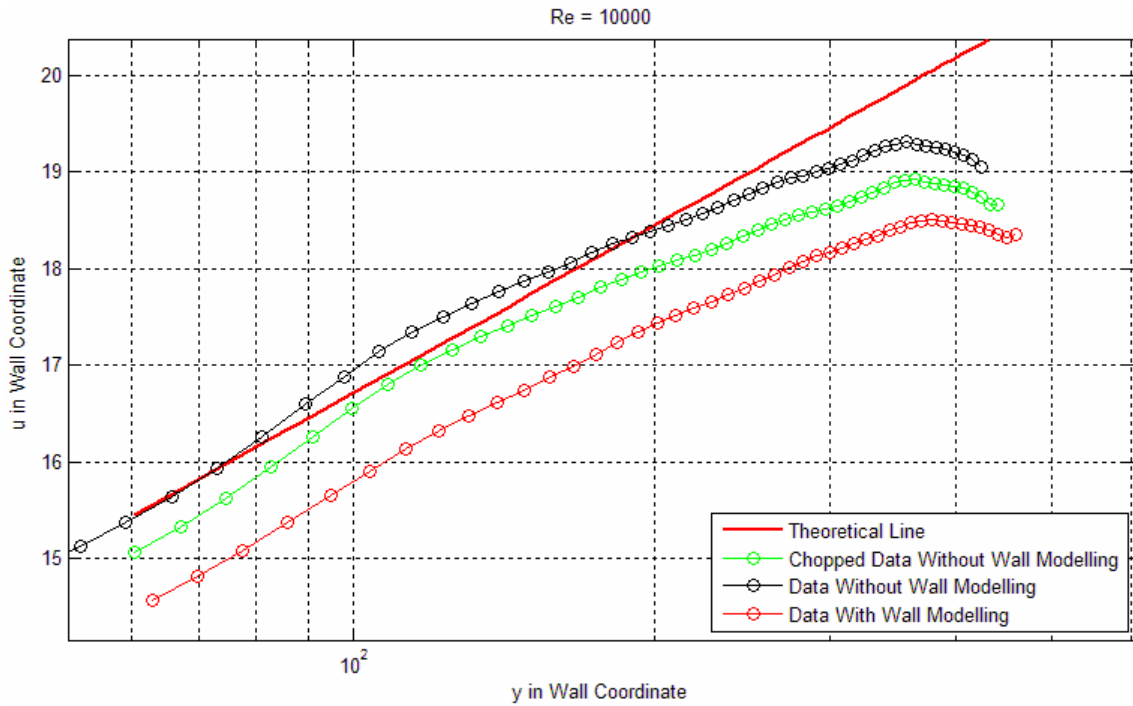


Figure 4.3: Exaggerated error

It can be seen that in general, there is a uniform drop in velocity all over the line but the pattern in the solution has been preserved very well. This could be a sign for a systematic error. Part of this drop is due to the modelling effect, but it has to be noted that to transform the velocity and the y values into the wall coordinate system, the friction velocity must be calculated. When the flow is resolved down to the wall the expression (2.22) is used to calculate the friction velocity, but when the lower region is removed and the wall model is used the expression (2.24) is used; therefore, there would be a little difference in calculated friction velocity. Although, this difference is very little (much less than 1%), but still wall coordinates are very sensitive to it and during visualization it is shown exaggerated. In Figure 4.3, the green line is produced with the same data that the black line was produced. While transforming the black line in wall coordinate the expression (2.22) was used to calculate friction velocity and in case of the green line the expression (2.24); the difference between the two friction velocities is just 0.02%. But it is obvious that this small error is shown exaggerated.

Therefore, it was decided to show the wind profile without using wall coordinates, Figure 4.4. It can be seen in figure 4.4 that at the lower part the two lines match each other perfectly. At higher elevations they differ a little from each other. But the difference is still within acceptable range. It is worth to mention that in this simulation the same mass flow was used; although, the domain was slightly smaller. This can also justify the increase of velocity in upper height. It can be decided that this wall model is working ok.

Enhancing Boundary condition

By examining the previous algorithm one may wonder why a linear extrapolation was used to define the velocity at the dummy nodes. We studied the changes that arise when the logarithmic profile continues even to the dummy nodes and the dummy node's velocities were computed using the same logarithmic law, i.e.:

$$\begin{aligned} u_0 &= \frac{u_\tau}{\kappa} \ln\left(\frac{y_0^+}{y_o^+}\right) \\ u_{-1} &= \frac{u_\tau}{\kappa} \ln\left(\frac{y_{-1}^+}{y_o^+}\right) \\ u_{-2} &= \frac{u_\tau}{\kappa} \ln\left(\frac{y_{-2}^+}{y_o^+}\right) \end{aligned} \quad (4.5)$$

After comparing the two versions, it was seen that there is practically no noticeable difference between the two methods. Definitely these differences can be neglected without any further discussion.

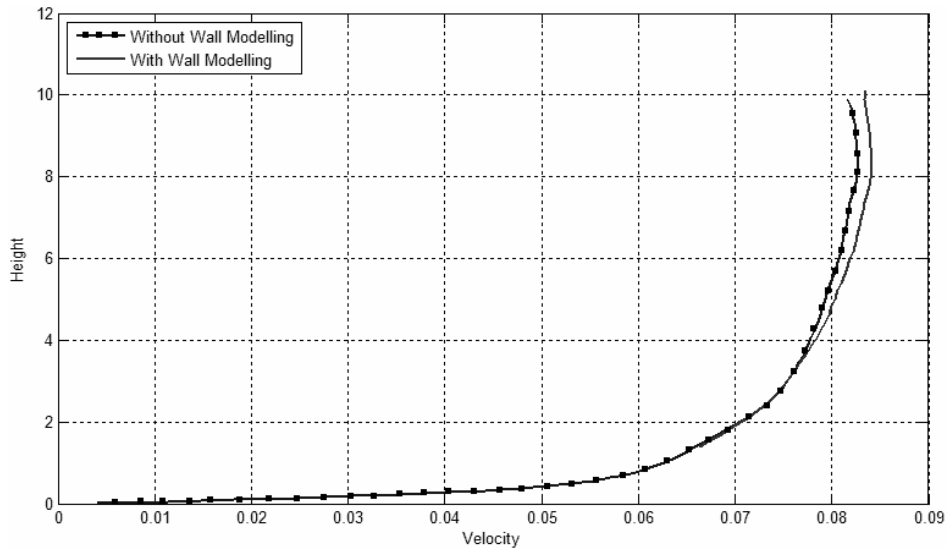


Figure 4.4: Simulation result with and without wall modelling

4.2.2. Averaged Value over Space (M2)

Although the wall modelled proposed in previous subsection was perfect; but, it uses the averaged values over both time and space. It was decided to eliminate averaging over time and just

perform plane averaging each time the boundary condition is applied. Instead of using the numerical solution at two y-levels to calculate the friction velocity just the first grid point was used, i.e.:

$$u_{\tau} = \frac{\kappa u_1}{\ln\left(\frac{y^+}{y_o^+}\right)} \quad (4.6)$$

As discussed before, y_o^+ is the roughness length scale. Now by calculating the derivative of equation (2.23) it can be written:

$$\frac{du}{dy} = \frac{u_{\tau}}{\kappa y} \quad (4.7)$$

Hence, the derivative of u is known at the first grid point and the velocity at the boundary can be calculated simply by:

$$u_0 = u_1 - \frac{du}{dy}(y_{m1} - y_0) \quad (4.8)$$

where $y_{m1} = \frac{y_1 + y_0}{2}$ is the elevation of first velocity node (since the staggered grid is used, the velocity is defined at the middle of each grid cells). Now that the velocity at the boundary is known, using (4.4) the dummy nodes are set. When this method was applied it was seen that the profile was damped heavily. Albertson (Albertson, 1996) has proposed some modification in calculating the velocity derivative. He has proposed that (4.7) can be used for mean values. But some additional terms have to be added as well to correct the derivative. Albertson uses equation (4.6) to calculate the friction velocity in all grid-cells located at the bottom of simulation domain, i.e. in the boundary. Then these friction velocities are averaged over horizontal planes. Similarly, velocities are averaged and the difference between the local and mean velocity derivatives is added to the result. In other words:

$$\frac{du}{dy} = \frac{\overline{u_{\tau}}}{\kappa y} + \frac{(\overline{u_1} - u_1) - (\overline{u_2} - u_2)}{\Delta y} \quad (4.9)$$

It was claimed that (4.9) gives better result and our simulation shows that indeed it does.

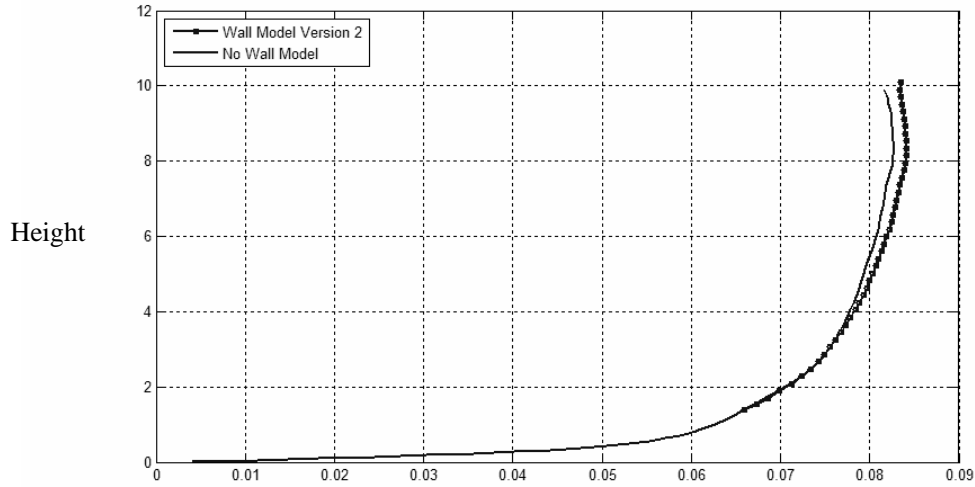


Figure 4.5: Wall model version 2

After visualizing the result, it was amazingly seen that this approach also behaves much the same as the previous model discussed in section 4.3.1 and there is just a little and ignorable amount of difference.

Enhancing the method

To have more accuracy, it was decided to distribute the velocities over dummy nodes using second order derivative approximate, but it did not enhanced the result by a noticeable amount.

4.2.3. Averaged Value over Time and Spanwise Direction (M3)

In previous models, velocities which are averaged over the entire horizontal plane were used. This procedure imposes no restriction as long as the surface roughness is homogenous. But as the goal of this research is to simulate boundary layer flow over surfaces with variable roughness, a model is needed which does not depend on values that are averaged in this way. It would not allow to truthfully representing any details of the underlying surface. Instead a more localized averaging would be desired and here we first turn to averaging over the spanwise direction only.

It was decided to use friction velocity values which are based on the solution averaged over time and only the spanwise direction. First, the code was modified to allow for this possibility. The simulation result is given in figure 4.6. The performance of this model is given in figure 4.7. The performance is defined here as $\frac{\text{Modelled.Value}}{\text{Simulated.Value}}$. It can be seen that the error is less than ten percent everywhere, relative to the values that no wall model was used.

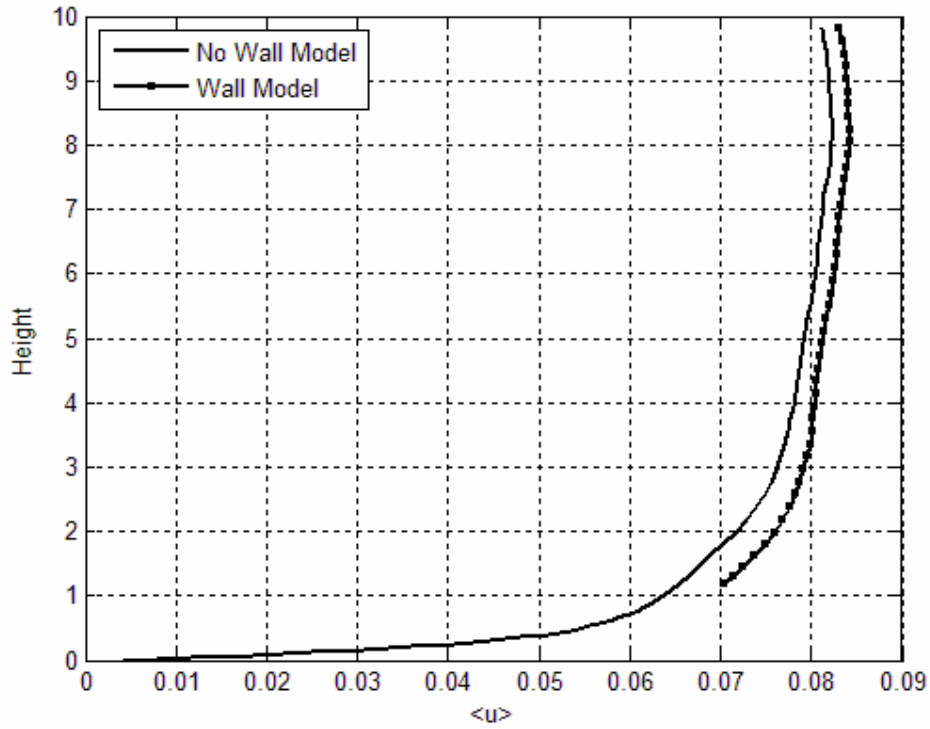


Figure 4.6: Wall modelling using averaged values over time and spanwise direction

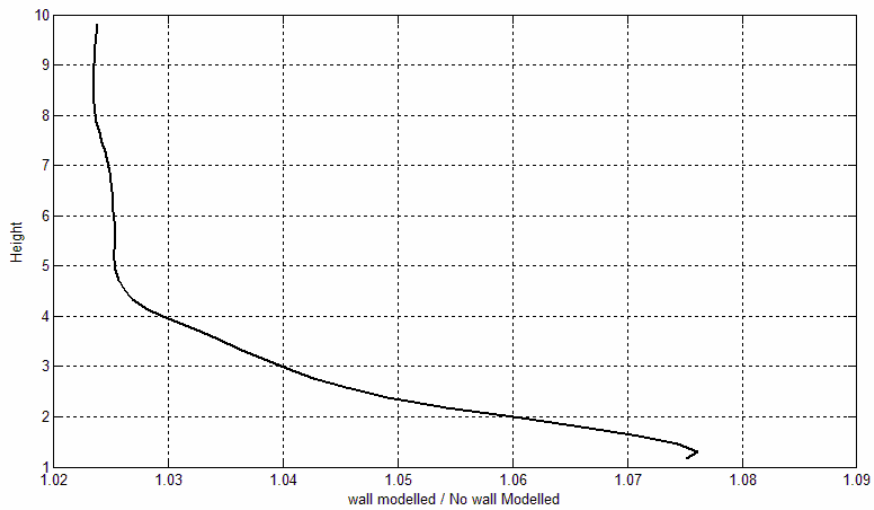


Figure 4.7: Model performance using averaged values over time and spanwise direction

4.2.4. Averaged Value over Time

The model in the previous subsection uses spanwise and time averaging only. However, it is more convenient, if no spatial averaging is used at all; thus, it is possible to unveil more of the surface structures. Hence, the code was modified again to have it calculate the average of the velocity over time. Later, these values were used in the same wall model as previously. The simulation results are

given in Figure 4.8 and how the result of this model compares to the case, which uses no wall modelling, is given in Figure 4.9.

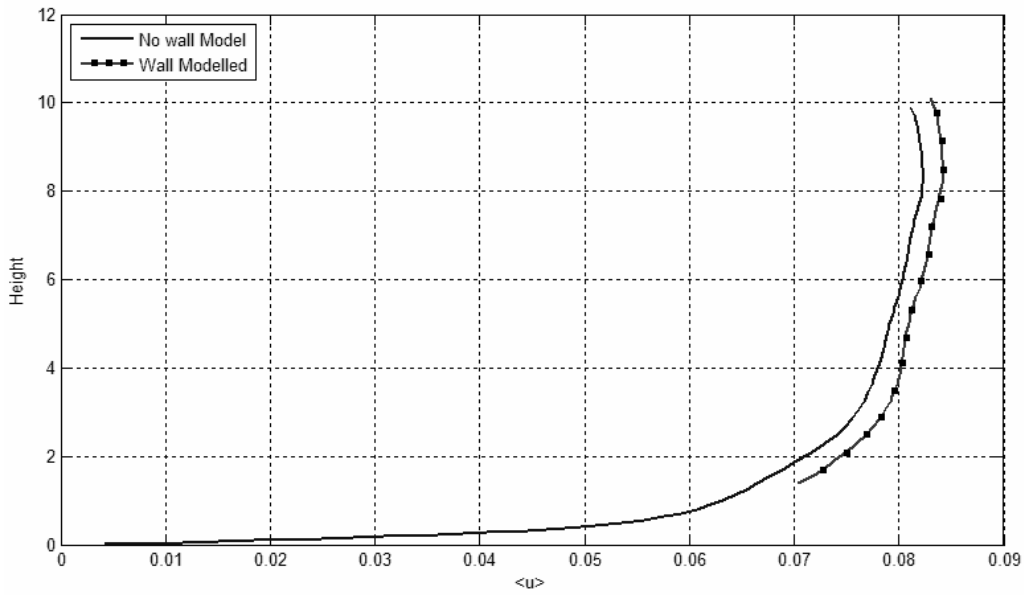


Figure 4.8: Wall model using averaged value over time

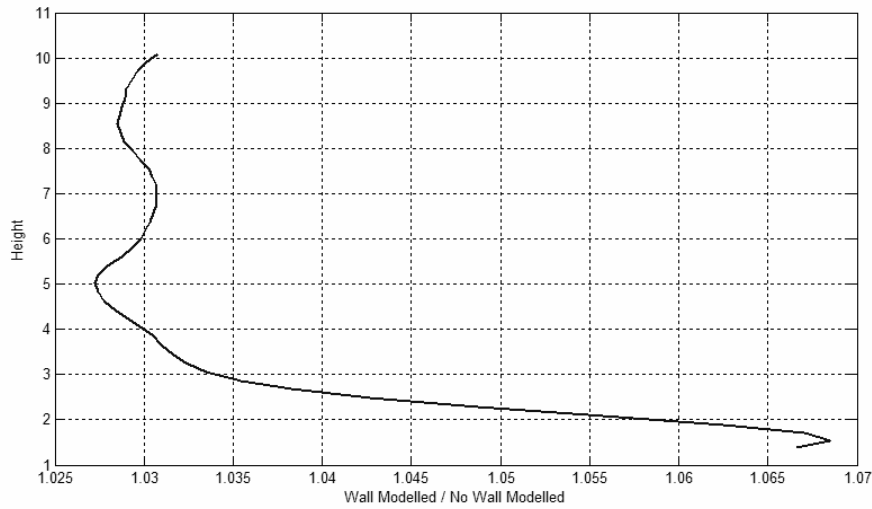


Figure 4.9: Wall model performance using averaged value over time

4.3. Summary

In this chapter several wall models were developed and they were tested against previous DNS simulation results that no wall model was used. It was seen that all developed models were able to perform the same simulation requiring much less memory and processing power and the error introduced by the modelling was within acceptable limit.

In this chapter all simulations were performed concerning boundary layer flow over homogenous smooth surfaces. In the next chapter the last model, which involves only time-averaging, is tested in an environment where the surface is rough. Thus, spatial variations in the roughness will be investigated in chapter 5.

5. Boundary Flow over Rough Surfaces

5.1. Introduction

In the previous chapter, different wall models were developed and tested for boundary-layer flow over smooth walls. The modelling introduces a certain error but:

1. The error was always less than 10 percent relative to the case when no wall model was used.
2. The current code requires much less memory; since, all small grid cells near the wall region are removed. According to Piomelli's studies (Piomelli and Balaras, 2002) it should reduce memory usage by 90 percent.
3. And it also needs much less processing power, due to reduced amount of grid cells and the fact that a greater time step can be used.

In this chapter the wall model which just uses time-averaged values, is tested over rough surfaces. The other developed models were also tested in the same condition but for the sake of simplicity the results are not represented here. They compare quite closely to the results presented.

The code was modified to allow for a variable surface roughness parameter, and a subroutine was added to read the roughness value from an external file as one of the model inputs. The roughness has to be stored in a file called 'y0.dat' and this file must be placed in the './dat/' directory (Linux addressing scheme is used here). When the code is compiled and executed, this file will be read by the code and the roughness value is stored in memory while the code is running.

As explained in previous chapter, the rougher the terrain is, the lower the wind velocity should be. Therefore, it is expected that the wind velocity over rough terrains be lower than those in smooth terrains. As no data related to rough terrain was available, the only possible way to check the simulation/modelling result is using the theoretical equations and also inspecting the behaviour of the code qualitatively.

In the following sections, first the code is tested in a homogeneous rough terrain, i.e. the terrain which has the same roughness scale everywhere. Later, the code is tested over a generally smooth terrain consisting of one rough patch in the middle. A surface with more number of rough patches is also tested.

5.2. Homogenous Rough Surface

Although, several cases were tested; but, the result of just one case is represented here. Other cases also show the same behaviour. A simulation domain with streamwise length of 4π and spanwise length of $\frac{4\pi}{3}$ and height of 10 was chosen. A uniformly rough surface with roughness of

$y_0^+ = 0.4996$ was chosen for the simulation, remember that for a smooth terrain this value has to be set to $y_0^+ = 0.1239$; thus, the terrain is chosen to be four times more rougher.

The rough-wall simulation result is given in Figure 5.1. This result is compared with the smooth wall case. It can be seen from Figure 5.1 that the model is able to perfectly simulate the homogenous rough wall as in previous chapter but for the smooth wall. By inspecting Figure 5.1 closer, it can also be seen that the error, difference between the simulation and theoretical formula, is less in homogenous rough wall. The theoretical lines, shown in Figure 5.1, are the same as the one explained in section 2.6. The difference is just in the value chosen for the roughness parameter.

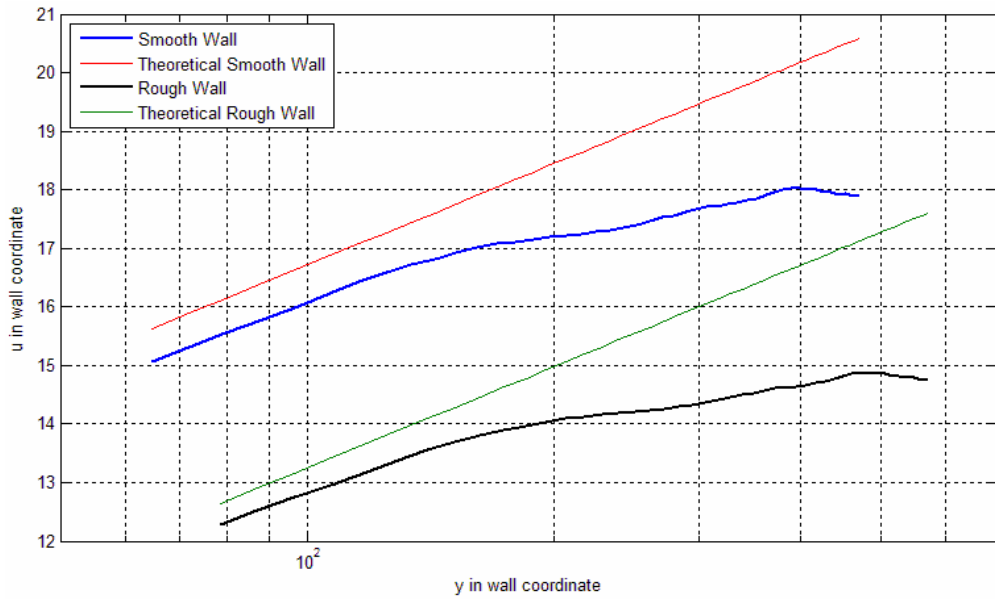


Figure 5.1: Wall model result over homogenous rough wall

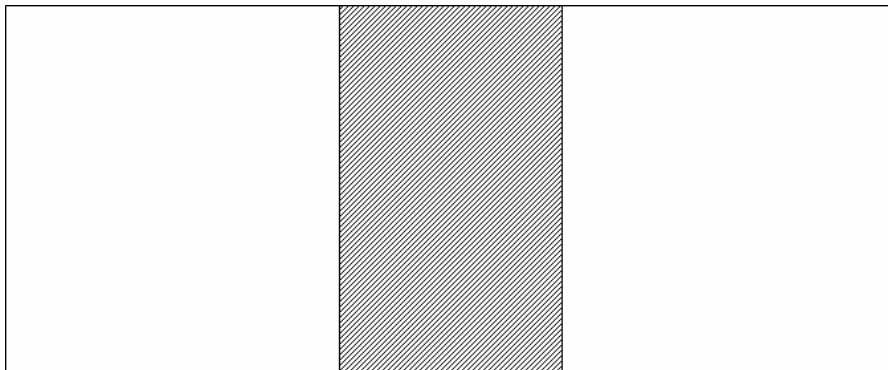


Figure 5.2: Patchy rough surface

5.3. Patchy Rough Surface

After assuring that the code is able to simulate a homogenous rough surface, a patchy surface was chosen to test the code. It was decided to choose a surface which one quarter of it is 4 times rougher than the rest, Figure 5.2.

As no data is available for this case from literature, the only possible method to validate the result is by inspecting the velocity behaviour qualitatively. As the fluid approaches the rough patch, it is expected to slow down and when it passes the rough patch and enters the smooth section again, it is expected to accelerate. We should be able to detect this cross-over in the mean velocity. This behaviour is indeed observed in the simulation result, Figure 5.3. For i index refer to Figure 5.2.

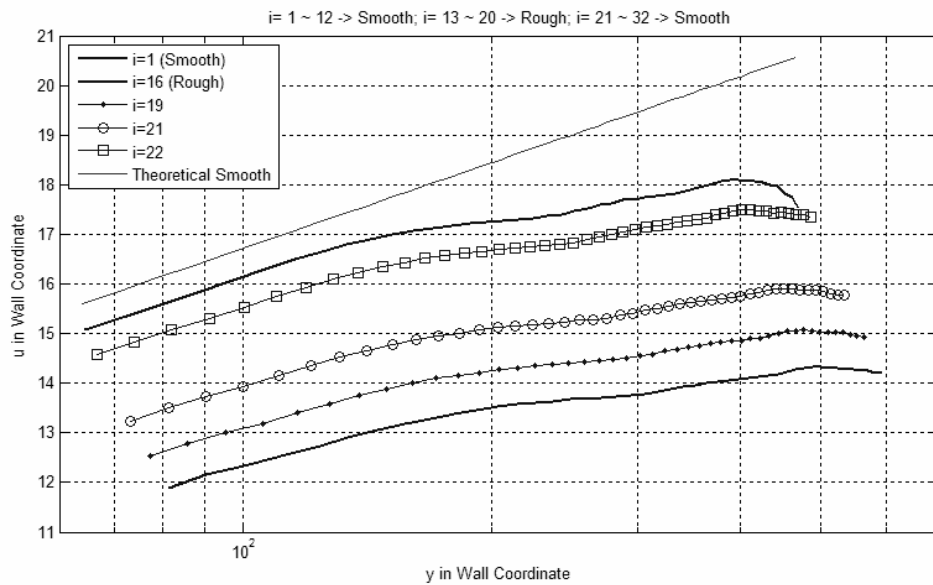


Figure 5.3: Velocity profile in different locations over patchy rough wall

Other wall models, discussed in previous chapter, also behave in much the same manner. The next test is to increase the number of patches and check how the velocity is responding. For this test, a surface with 4 patches was selected; see Figure 5.4. The velocity profile at $y_{j=1}$ is shown in figure 5.5. The dashed lines in figure 5.5 show the border of the patches. It can be seen that the velocity responds to roughness patches quite well in this case. It is worth mentioning that the velocity is averaged over spanwise direction and then this averaged velocity is shown in Figure 5.5.

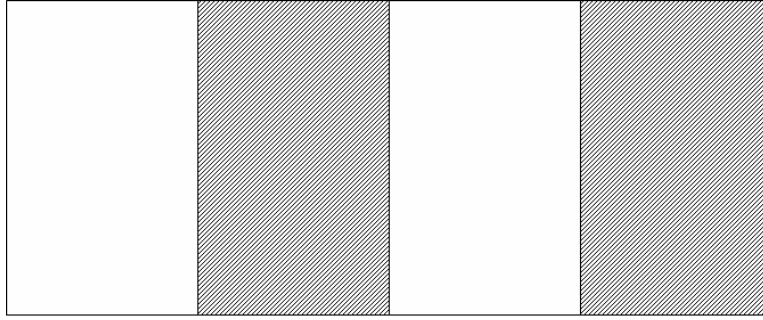


Figure 5.4: Surface with Four Patches

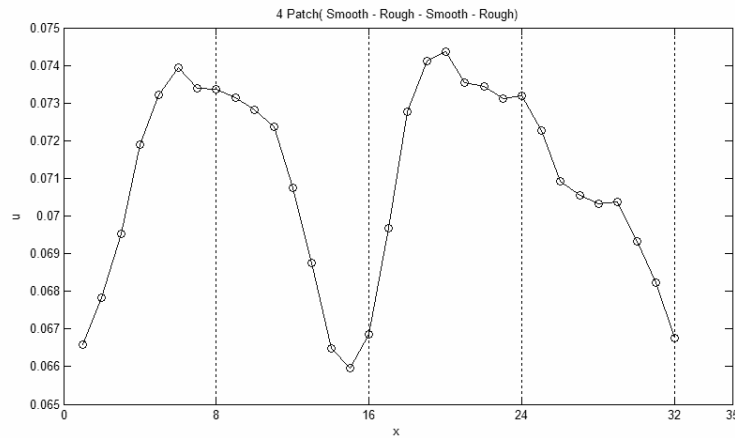


Figure 5.5: Velocity Variation

5.4. Summary

In this final chapter, the developed wall model during this thesis was tested over homogenous and patchy surfaces. It was seen that the error is decreased over more rough surfaces and the code was able to perfectly respond to surface roughness changes. Due to lack of data, the result was mostly validated qualitatively. Only in homogenous rough surface it was possible to compare the result with the theoretical lines; hence, performing a more quantitative test.

The developed code requires much less memory and processing power; thus, much faster and less expensive simulation is possible. By removing very fine grids near the wall, a greater time step can be selected; causing even a faster simulation.

Further validation is recommended. Also including a sub grid scale model (SGS) may improve the result.

6. Conclusions and Recommendations

6.1. Conclusions

During this research, a code based on direct numerical simulation using FORTRAN programming language was developed, which with the help of proper boundary condition, can be used to simulate atmospheric flow over rough terrains. By coupling the code with a wall model, the memory and processing power requirements were reduced drastically and much faster, less expensive simulations became possible. Moreover, by removing very fine grids near the wall due to using a wall model, a greater time step can be selected; causing even a faster simulation.

In chapter two, a fully developed channel was modelled using the DNS code. It was also discussed how the boundary has to be implemented in this case. It is known that in channel flow the fluid is in touch with the walls or boundaries, which causes the fluid velocity to be zero. Based on this research it can be concluded that by applying no-slip boundary conditions and using an odd extrapolation to extend the velocity to the dummy nodes, channel flow boundary condition can be simulated.

The code was completely tested in a fully developed channel flow and the simulation results were compared with theoretical formulas. Later, the simulation results were tested against other available data. Although, the grid resolutions, which were used in this research, were rather coarse, still the results were perfectly matching available data through CFD databases, which use much finer grids. This shows that the symmetry-preserving discretization, used in this research, leads to a much more economical simulation with an acceptable accuracy.

The main objective, i.e. studying how the CFD techniques can be used in atmospheric flow studies, was addressed in chapter 3. As sub-objective, the boundary condition requirements and how they have to be numerically implemented was also studied in this chapter. To use the Navier-Stokes equations to simulate boundary layer flow, at the bottom, the same no-slip boundary condition as in channel flow has to be used, but at top layer the free-slip boundary condition can be applied. To implement the free-slip boundary condition an even function is used to distribute the velocity over dummy nodes.

To make the simulation even faster, it was decided to model the small eddies, existing close to the surface. These small eddies have great impact on DNS simulations costs and by removing them the DNS can be performed much more economically. This approach is called wall modelling and its theory was discussed fully in section 2.6. Several developed wall models were explained in chapter 4. All the developed wall models were tested in a homogenous smooth surface environment and it was shown that the error due to modelling effect is always less than 10 percent. In chapter 5, one of the developed wall models was validated in homogenous rough surface and also in patchy rough surface. It was shown that the developed model was behaving within the acceptable margin for both surfaces.

The current code coupled with a wall model is much faster and requires much less memory; therefore, it is suitable for real case simulations.

6.2. Implementation to Real Problems

Since the time was so limited, it was not possible to test the code for a real case in nature. But all variables were used in non-dimensional form, so that in future the code can be fed with real data much easier.

While simulating a real case, the physical domain of the problem has to be converted into computational domain. The physical domain is the real geometry of the problem representing the real world. The computational domain is a transformed version of physical domain. This transformation is being performed in such a way that both the calculation and coding become much easier, Figure 6.1. This transformation is usually performed via calculating Jacobi matrices. Even commercial software packages perform this transformation intrinsically, although they may not report that to the user.

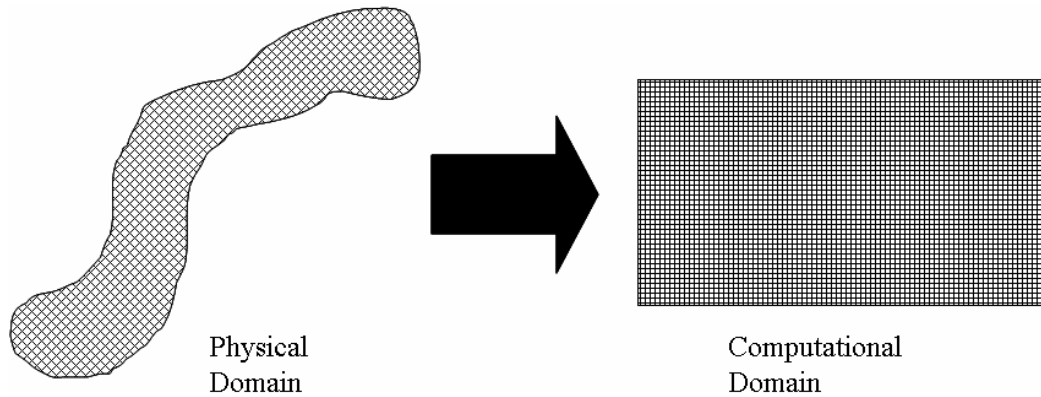


Figure 6.1: Physical Domain Versus Computational Domain

All the variables and the parameters have to be transformed into a non-dimensional form. Therefore, a length scale and velocity scale is selected to make all lengths and all velocities non-dimensional. When the velocity and length scale are selected, it is no longer possible to select the time scale arbitrary. In boundary layer simulation (Albertson, 1996), it is very common to choose half the boundary height as length scale. Therefore, the boundary height becomes 2 in non-dimensional form. Due to the requirements of the numerical algorithms used in horizontal directions, the non-dimensional length of the horizontal directions should be a factor of π . Using the length scale, which is already selected, the dimensional horizontal lengths of the physical domain have to be properly selected; somehow they are forced. The friction velocity is usually chosen as the velocity scale; thus, the time scale is simply the length scale over the velocity scale, Table 6.1.

Therefore, before using this code in real cases, a pre-processing unit has to be developed which behave as an interface between the current code and the data collected in the field. As the results of the current code are also in non-dimensional form, a post-processing unit has to be used to convert everything back in its dimensional form. Moreover, by introducing a sub-grid scale model (SGS), the effect of small eddies can be taken into account; hence, the simulation results may become more accurate.

Table 6.1: Common scales

Parameter / Variable	Description	Example (Taken from Albertson, 1996)
u_τ or u^*	Velocity Scale	0.45 ms^{-1}
l^* or z^*	Length Scale	637 m
$t^* = \frac{u^*}{z^*}$	Time Scale	1415 s

6.3. Interactions with RS-GIS

ITC's main focus is on Geo-Information Science and Earth Observation, which consist of a combination of tools and methods for the collection - through aerospace survey techniques - storage and processing of geo-spatial data, for the dissemination and use of these data and of services based on these data (http://www.itc.nl/about_itc/mission_statement.asp).

The focus of this thesis lies on developing the numerical tools required in atmospheric boundary studies; therefore, all of the data were theoretical and it was not derived from the satellite imagery or remote sensing techniques. In real cases however, it is possible to extract data from satellite imagery and use them as inputs of the code. As an example, the roughness parameter, which was selected arbitrary in this thesis, can be extracted from the satellite images or by processing LIDAR data. The velocity field can be initialized via data collected in the field or by processing satellite images with a high temporal resolution. These temporal data can be also used in updating the deferent parameters or variables used in the code.

6.4. Applications

Air pollution transport simulation

When the velocity fields are resolved and the air movement is simulated, it is possible to study how the different materials, such as air pollutants, are transported by this fluid. For example, it is possible to study how the smoke of a chimney belonging to a factory is transported by the air and whether it is going to affect the city or not. Therefore, the best location for the factory can be selected. Or, in cases with the possibility of explosion producing dangerous smokes, this code can help to locate the safe location, such as Chernobyl nuclear power plant explosion.

Water vapour transport modelling combined with remote sensing

The water vapour can be derived using algorithms such as SEBS via satellite images. This water vapour can be introduced into the code as scalar field and it is possible to simulate how the water vapour is transported via atmospheric flow.

Foot print analysis

Using measuring devices, based on Eddy Correlation (EC) theory, is very common among the meteorologists. As an example, meteorologists are interested to measure the amount of the carbon dioxide produced or used by a certain region to study the carbon cycle, one of the key factors in bio-

environmental and climate changes studies. The location of carbon dioxide measuring device is crucial. If installed on wrong location it would measure the carbon exchange of the wrong area. The area, where each device collects data from, is called foot print of that device. Defining this area is a very cumbersome procedure and is called foot print analysis. The current developed code can be used in foot print analysis, Figure 6.2.

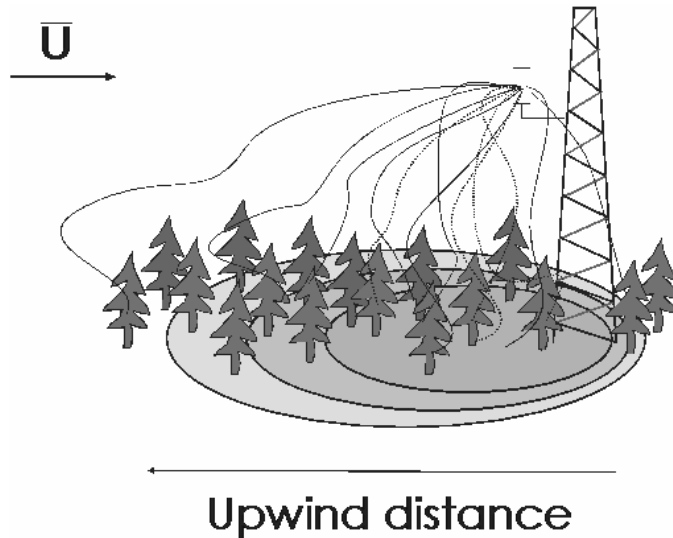


Figure 6.2: Foot Print of a Measuring Device

Cloud formation and movement simulation

By adding few more software routines to analyse the multiphase materials, it is also possible to study the cloud formation. Furthermore, it is also possible to study how clouds are being transported by the atmospheric flow.

The above cases are examples to indicate the potential use of the developed code for number of practical problems. Therefore, it is recommended to test and apply the current code to real world situations.

A. Mathematical Description of Fluid Flow (Navier-Stokes Equation)

A.1. Modelling methods for the fluid flow

To scientifically study natural phenomena, first we need to define them in mathematic language. In other words, we are not able to study the behaviour of an event prior to having a mathematical abstraction of that natural phenomenon. The mathematic is the language for abstracting the reality into numbers. Then we are able to process the numbers using available numerical method and large computers, such as super computers. And then the results in numerical form are converted back again to their natural and physical meaning for further interpretation.

In order to convert the real world into numbers and mathematical equation, we need to follow some logical and rational protocols or rules, which can be called physics or physical concept given by physicist. Fluid mechanics and turbulence or in general computational fluid dynamics, CFD, is based on some fundamental physical concept, such as:

1. There is no production of mass (conservation of mass), which leads to continuity equation.
2. Second law of motion proposed by Newton, $\vec{F} = m\vec{a}$. Where \vec{F} is the total or net body and surface forces applied on fluid, m is the mass of fluid, and \vec{a} is the acceleration or $\vec{a} = \frac{d\vec{r}}{dt}$. This leads to momentum equation.
3. Energy is conserved. (Usually that part of fluid energy which is converted into heat is called dissipated energy, but the some of these two is constant.). This leads to energy equation.

To derive an equation for each of these physical principles, first we have to choose a model for the fluid and later apply these physical principles to the selected model and later, by the mathematical rules, derive the appropriate equation. There are two models for the fluid, commonly used by scientist(Anderson, 1995):

1. **Finite Control Volume:** In this model a control volume, Ω , of fluid and a closed imaginary surface bounding this volume, S , is drawn in a finite region of the fluid. This volume may be fixed in space and fluid particles pass through the volume, i.e. Eulerian Frame, or it may flow by the fluid so that always the same fluid particles are inside the volume, i.e. Lagrangian Frame.
2. **Infinitesimal Fluid Model:** In this method, a very small portion of fluid volume is imagined, $d\Omega$. This element is small enough and differentiable but at the same time it is large enough to hold many molecules, so that it can be considered as a continuum. This Element can also be fixed in space or flow by the fluid.

A.2. Conservative Versus Non-conservative Form of the Equations

In previous section four models of fluid were introduced. There is another naming convention usually used. Those equations resulted from a stationary fluid model are usually called ‘Conservative’ and those derived from a moving model are well-known as ‘Non-Conservative’.

The conservative and non-conservative forms are just two different approaches in mathematical method of describing the flow. And they can be converted back and forth to each other, (Anderson, 1995). Thus, general speaking conservative and non-conservative form of the equations do not differ from each other and indeed scientist did not pay too much attention to it. Even in the text books the equations were not labelled by these two terms, until around 1980s (Anderson, 1995). 1980s can be determined as the period when the modern CFD was taking birth. After inventing modern computers and super computers, the numerical algorithm become so popular and programmers tried to developed computer codes using different languages to perform numerical analysis and CFD enters its new era which can be called modern CFD. Although the conservative and non-conservative form of the fluid equations have no difference in physical and conceptual aspect; but, they do differ in coding effort (Anderson, 1995).

A.3. Continuity Equation

Table A.1: Continuity Equation

Non-Conservative Form

$$\frac{D\rho}{Dt} + \rho \vec{\nabla} \cdot \vec{V} = 0 \quad (\text{A.1})$$

Conservative Form

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{V}) = 0 \quad (\text{A.2})$$

A.4. Momentum Equation

Table A.2: Momentum Equation

Non-Conservative Form

$$\rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x \quad (\text{A.3})$$

$$\rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y \quad (\text{A.4})$$

$$\rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z \quad (\text{A.5})$$

Conservative Form

$$\frac{\partial(\rho u)}{\partial t} + \vec{\nabla} \cdot (\rho u \vec{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x \quad (\text{A.6})$$

$$\frac{\partial(\rho v)}{\partial t} + \vec{\nabla} \cdot (\rho v \vec{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y \quad (\text{A.7})$$

$$\frac{\partial(\rho w)}{\partial t} + \vec{\nabla} \cdot (\rho w \vec{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z \quad (\text{A.8})$$

A.5. Energy Equation

Table A.3: Energy Equation

Non-Conservative Form

$$\begin{aligned}
 \rho \frac{D}{Dt} \left(e + \frac{|\vec{V}|^2}{2} \right) &= \rho \dot{q} + \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) \\
 &- \frac{\partial(u\bar{p})}{\partial x} - \frac{\partial(v\bar{p})}{\partial y} - \frac{\partial(w\bar{p})}{\partial z} \\
 &+ \frac{\partial(u\tau_{xx})}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} \\
 &+ \frac{\partial(v\tau_{xy})}{\partial x} + \frac{\partial(v\tau_{yy})}{\partial y} + \frac{\partial(v\tau_{zy})}{\partial z} \\
 &+ \frac{\partial(w\tau_{xz})}{\partial x} + \frac{\partial(w\tau_{yz})}{\partial y} + \frac{\partial(w\tau_{zz})}{\partial z} + \rho \vec{f} \cdot \vec{V}
 \end{aligned} \tag{A.9}$$

Conservative Form

$$\begin{aligned}
 \frac{\partial}{\partial t} \left[\rho \left(e + \frac{|\vec{V}|^2}{2} \right) \right] + \vec{\nabla} \cdot \left[\rho \left(e + \frac{|\vec{V}|^2}{2} \right) \vec{V} \right] &= \rho \dot{q} \\
 &+ \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) \\
 &- \frac{\partial(u\bar{p})}{\partial x} - \frac{\partial(v\bar{p})}{\partial y} - \frac{\partial(w\bar{p})}{\partial z} \\
 &+ \frac{\partial(u\tau_{xx})}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} \\
 &+ \frac{\partial(v\tau_{xy})}{\partial x} + \frac{\partial(v\tau_{yy})}{\partial y} + \frac{\partial(v\tau_{zy})}{\partial z} \\
 &+ \frac{\partial(w\tau_{xz})}{\partial x} + \frac{\partial(w\tau_{yz})}{\partial y} + \frac{\partial(w\tau_{zz})}{\partial z} + \rho \vec{f} \cdot \vec{V}
 \end{aligned} \tag{A.10}$$

A.6. Poisson's Equation to Solve Pressure Term

To solve the momentum equations, pressure must be a priori known variable, which is not; thus, the resulted velocities component does not satisfy the continuity equation. Therefore, it is not allowed to use equation of state:

$$P = \rho_{air} \mathfrak{R} T_v \tag{A.11}$$

Where:

\mathfrak{R} is the gas constant

T_v is the virtual absolute temperature.

Therefore, usually Poisson's equation is used to solve pressure field. The idea is that (Anderson, 1995):

- 1) Using initial value of P^* , denoted with star to mention that it is not a correct pressure, the momentum equation is solved for components of velocities. The calculated velocities are not correct at this stage and they do not satisfy the continuity equation.
- 2) Then the Poisson's equation is solved using the available velocity to find the correction term for pressure field or P' , denoted by prime to indicate that it is a correction term.
- 3) Now the Pressure field is corrected by:

$$P = P^* + P' \quad (\text{A.12})$$

- 4) Now the velocity field is corrected:

$$\begin{aligned} u &= u^* + u' \\ v &= v^* + v' \\ w &= w^* + w' \end{aligned} \quad (\text{A.13})$$

- 5) Then the step 2, 3, and 4 are repeated again until the corrected components of velocities satisfy the continuity equation.

The Poisson's equation is named after the French mathematician "Simon Denis Poisson" (Wikipedia3) and it can be written as:

$$\nabla^2 \varphi = f \quad (\text{A.14})$$

Where:

∇^2 is known as Laplace Operator
 φ and f are any scalar real or complex function

Here, φ is the pressure field and in the developed FORTRAN code f is defined in its discrete form as:

$$\begin{aligned} f_{i,j,k} &= \frac{3^{2+d}}{\Delta t} \left[(u_{i,j,k} - u_{i-1,j,k}) d_{yj}^1 d_{zk}^1 + (v_{i,j,k} - v_{i,j-1,k}) d_{xi}^1 d_{zk}^1 + (w_{i,j,k} - w_{i,j,k-1}) d_{xi}^1 d_{yj}^1 \right] \\ &- \frac{3^{2+d}}{\Delta t} \left[(u_{i+1,j,k} - u_{i-2,j,k}) d_{yj}^3 d_{zk}^3 + (v_{i,j+1,k} - v_{i,j-2,k}) d_{xi}^3 d_{zk}^3 + (w_{i,j,k+1} - w_{i,j,k-2}) d_{xi}^3 d_{yj}^3 \right] \end{aligned} \quad (\text{A.15})$$

The parameters $d_{xi}^1, d_{yj}^1, \dots$ are defined as before.

B. Validation with Taylor-Green Class of Vortices

B.1. Taylor-Green Class of Vortices

In turbulent flow, the large scale eddies are divided into smaller ones and the smaller ones are divided further into even smaller ones. This procedure continues until eddies are so small that the energy will dissipate via viscosity. Taylor (Taylor, 1935; Taylor and Green, 1937) has shown that the dissipation of energy and creation of turbulence is not the same all over the field. He has shown that near the wall the energy dissipation is less than the turbulent generation.

To better simulate the process of energy dissipation, via generation of smaller eddies from larger ones, a special class of vortices are tested, which are referred to as Taylor-Green class of vortices; since, these two scientists have found a closed form mathematical solution for the Navier-Stokes equations while the turbulent field is initialized by these type of vortices. For thorough and complete explanation refer to (Taylor and Green, 1937). Here just a summary of the procedure that these two scientists used is given.

The velocity field for Taylor-Green vortices is initialized by:

$$\begin{aligned} u &= A \cos(ax) \sin(by) \sin(cz) \\ v &= B \sin(ax) \cos(by) \sin(cz) \\ w &= C \sin(ax) \sin(by) \cos(cz) \end{aligned} \quad (\text{B.1})$$

These equations are consistent, i.e. they do not contain contradiction, only if:

$$Aa + Bb + Cc = 0 \quad (\text{B.2})$$

Now by considering the equation of motion, the above equation, and a little bit of integration the velocity can easily be derived. For the integration techniques refer to (Abouali, 2002). Here just the solution for u is given:

$$u = A(1 - \theta vt) \cos(ax) \sin(by) \sin(cz) + \frac{A_3}{a} t \sin(2ax) \cos(2by) - \frac{A_2}{a} t \sin(2ax) \cos(2cz) \quad (\text{B.3})$$

And:

$$\theta = a^2 + b^2 + c^2 \quad (\text{B.4})$$

$$A_1 = \frac{b}{4} \left(A^2 b \frac{a^2}{b^2 + c^2} + ABa \right) = -\frac{c}{4} \left(A^2 c \frac{a^2}{b^2 + c^2} + ACa \right) \quad (\text{B.5})$$

The formula for A_2 and A_3 can be derived by cyclical permutation of a , b , and c . Now the same procedure can be followed to derive the second, third, and successive approximations for u , v , and w ; but this time by using equation (B.3) as initial value. (Taylor and Green, 1937) have found that these equations follow a special pattern and they have proceeded to find a few approximations. The second approximation, calculated by Taylor, is given here, the third one contains 518 terms and it would not be mentioned here.

$$\begin{aligned}
 u &= \delta_1 \cos(ax) \sin(by) \sin(cz) + \frac{x_3}{a} \sin(2ax) \cos(2by) - \frac{x_2}{a} \sin(2ax) \cos(2cz) \\
 &+ \frac{\alpha_1}{3} \cos(3ax) \sin(by) \sin(cz) + \gamma_2 \cos(ax) \sin(3by) \sin(cz) \\
 &+ \beta_3 \cos(ax) \sin(by) \sin(3cz) + \frac{y_3}{a} \cos(3ax) \sin(3by) \sin(cz) \\
 &- \frac{y_2}{a} \cos(3ax) \sin(by) \sin(3cz) + \frac{L_1}{2} \sin(4ax) \cos(2by) \cos(2cz) \\
 &+ N_2 \sin(2ax) \cos(4by) \cos(2cz) + M_3 \sin(2ax) \cos(2by) \cos(4cz)
 \end{aligned} \tag{B.6}$$

The coefficients used in above equation are defined as follow:

$$\begin{aligned}
 \delta_1 &= A \left(1 - \theta v t + \frac{1}{2} \theta^2 v^2 t^2 \right) - \frac{1}{2a} ((CcA_3 - BbA_2) \left(\frac{1}{2} t^2 - \frac{1}{3} \theta v t^3 \right)) \\
 x_1 &= A_1 \left(t - \theta v t^2 + \frac{1}{3} \theta^2 v^2 t^3 \right) - 2A_1 (b^2 + c^2) v t^2 - \frac{A_2 A_3}{3} \left(\frac{b^2 - c^2}{b^2 + c^2} \right) t^3 \\
 y_1 &= \frac{1}{2} \left(\frac{1}{2} t^2 - \frac{1}{3} \theta v t^3 \right) A_1 A a \\
 \frac{\alpha_1}{3} &= - \left(\frac{1}{2} t^2 - \frac{1}{3} \theta v t^3 \right) \left(\frac{6a(A_3(Aa - Bb) + A_2(Cc - Aa))}{9a^2 + b^2 + c^2} - AA_3 + AA_2 - \frac{1}{2} A_2 \frac{Cc}{a} + \frac{1}{2} A_3 \frac{Bb}{a} \right) \\
 \beta_1 &= - \left(\frac{1}{2} t^2 - \frac{1}{3} \theta v t^3 \right) \left(\frac{2b(A_3(Aa - Bb) + A_2(Cc - Aa))}{9a^2 + b^2 + c^2} + \frac{1}{2} \frac{a}{b} AA_3 + \frac{1}{2} BA_2 \right) \\
 \gamma_1 &= - \left(\frac{1}{2} t^2 - \frac{1}{3} \theta v t^3 \right) \left(\frac{2c(A_3(Aa - Bb) + A_2(Cc - Aa))}{9a^2 + b^2 + c^2} - \frac{1}{2} \frac{a}{c} AA_2 - \frac{1}{2} CA_3 \right) \\
 \frac{L_1}{2} &= - \frac{1}{3} t^3 \left(\frac{4a}{4a^2 + b^2 + c^2} - \frac{2}{a} \right) A_2 A_3 \\
 M_1 &= - \frac{1}{3} t^3 \left(\frac{2b}{4a^2 + b^2 + c^2} + \frac{1}{b} \right) A_2 A_3 \\
 N_1 &= - \frac{1}{3} t^3 \left(\frac{2c}{4a^2 + b^2 + c^2} + \frac{1}{c} \right) A_2 A_3
 \end{aligned} \tag{B.7}$$

The coefficient with indices of 2 and 3 can be obtained again by cyclical permutation of a , b , and c . it is worth to mention that these coefficients are not independent, but related to each other by:

$$\begin{aligned}
 a\delta_1 + b\delta_2 + c\delta_3 &= 0 \\
 a\alpha_1 + b\beta_1 + c\gamma_1 &= 0 \\
 aL_1 + bM_1 + cN_1 &= 0
 \end{aligned} \tag{B.8}$$

B.2. Special Case of Taylor-Green Class of Vortices:

For simplification, a special case of Taylor-Green vortices is considered here. This special case is obtained when:

$$\begin{aligned} a &= b = c \\ A &= -B \\ C &= 0 \end{aligned} \tag{B.9}$$

By making this assumption, all the equations can be simplified as:

$$\begin{aligned} \theta &= 3a^2 \\ A_1 &= -A_2 = -\frac{1}{8}A^2a^2 \\ A_3 &= 0 \end{aligned} \tag{B.10}$$

Also the coefficient can be summarized as:

$$\begin{aligned} \delta_3 &= x_3 = y_3 = \alpha_3 = L_1 = M_1 = N_1 = L_2 = M_2 = N_2 = 0 \\ \delta_1 &= -\delta_2 = A \left(1 - 3a^2vt + \frac{9}{2}a^4v^2t^2 \right) + \frac{1}{2}AA_1 \left(\frac{1}{2}t^2 - a^2vt^3 \right) \\ x_1 &= x_2 = A_1 \left(t - 7a^2vt^2 + 3a^4v^2t^3 \right) \\ \frac{1}{2}\alpha_1 &= -\frac{1}{2}\alpha_2 = -\beta_1 = -\gamma_1 = \beta_2 = \gamma_2 = \frac{15}{22}AA_1 \left(\frac{1}{2}t^2 - a^2vt^3 \right) \\ -\beta_3 &= \gamma_3 = \frac{y_1}{a} = \frac{y_2}{a} = \frac{1}{2}AA_1 \left(\frac{1}{2}t^2 - a^2vt^3 \right) \\ -\frac{1}{2}L_3 &= M_3 = N_3 = \frac{4}{9} \frac{A_1^2}{a} t^3 \end{aligned} \tag{B.11}$$

B.3. Skew-Symmetry Preserving Simulation Result

To validate the DNS code the same special case of Taylor-Green class of vortices was used. Before proceeding with this section, it should be noted that if time becomes larger than a certain value, the above equation is no more valid and their results are prone to errors. Usually the results of the above equations are not trustable for large t .

To initialize the velocity field at time zero, it was assumed that:

$$\begin{aligned}
 a &= b = c = 1 \\
 A &= -B = 1 \\
 C &= 0
 \end{aligned}
 \tag{B.12}$$

Therefore the velocity field is initialized at $t = 0$ with:

$$\begin{aligned}
 u &= \cos(x) \sin(y) \sin(z) \\
 v &= \sin(x) \cos(y) \sin(z) \\
 w &= 0
 \end{aligned}
 \tag{B.13}$$

But as the current DNS code uses a staggered grid, the formula was altered a little bit in order to correct for the correct position of u , v , and w as follow:

$$\begin{aligned}
 u_{i,j,k} &= \cos(x_i) \sin\left(\frac{y_{j-1} + y_j}{2}\right) \sin\left(\frac{z_{k-1} + z_k}{2}\right) \\
 v_{i,j,k} &= -\sin\left(\frac{x_{i-1} + x_i}{2}\right) \cos(y_j) \sin\left(\frac{z_{k-1} + z_k}{2}\right) \\
 w_{i,j,k} &= 0.0
 \end{aligned}
 \tag{B.14}$$

The simulation space was chosen to be $2\pi * 2\pi * 2\pi$ and this space was divided into $64*64*64=262144$ cells. The grid spacing was equidistant in all directions and the periodic boundary condition was applied in all direction. Therefore the flow is not bounded. Although the flow is not bounded, there is no mass flow from any boundaries which applies in this special case of vortices.

The vortices' evolution is shown in figure B.1 to figure B.6. In these figures the horizontal axis is x and the vertical axis is y . The coordinates on these axes is the cell number.

The process of energy dissipation is monitored in figure B.7 and B.8. This result is compared with other simulation like the one from (Kuczaj, 2003), figure B.9 and B.10. It should be noted that the Reynolds number is different between (Kuczaj, 2003) and the one here.

The same simulation is performed with $32*32*32=32768$ cells to check the sensitivity of the code to coarse meshes. The results are given in figure B.11 and B.12. It is evident that the two energy graphs are matching each other very well in the early stage of time, but later, when time becomes larger, the difference is noticeable in energy. This is exactly due to coarse meshes. As the time advances, eddies are grinded down to the smaller one, until they become much smaller to be simulated by large grid cells. Therefore, the finer grid, can track this process longer in time.

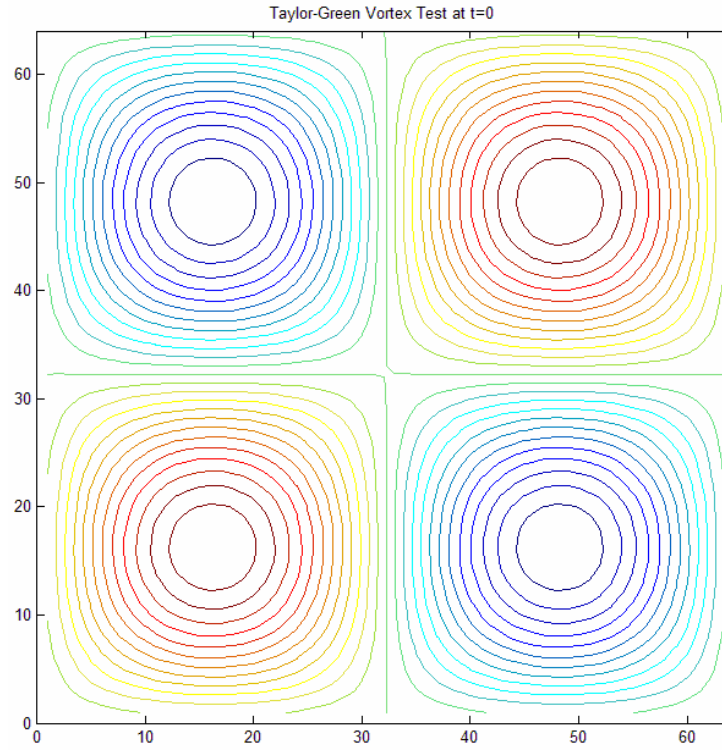


Figure B.1: Special Case of Taylor-Green Class of Vortices at time 0

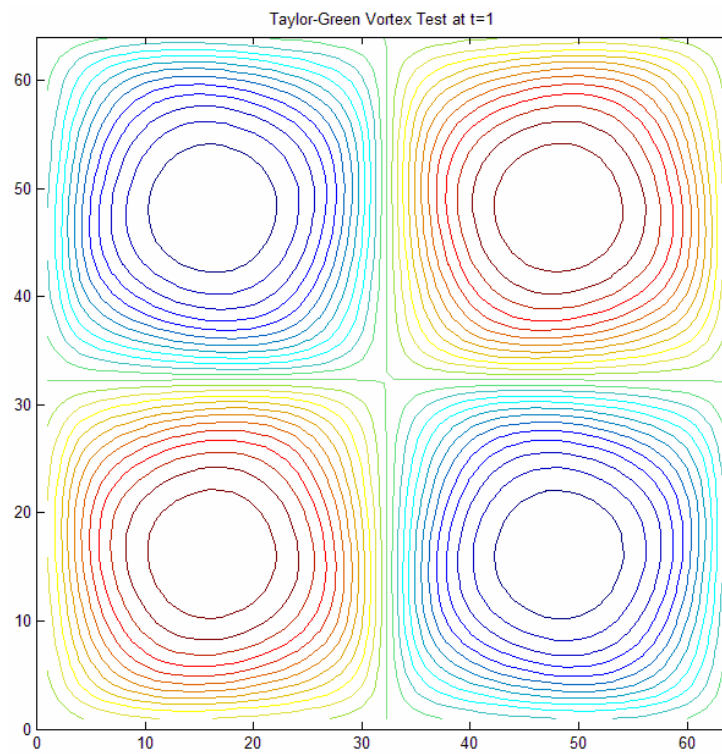


Figure B.2: Special Case of Taylor-Green Class of Vortices at time 1

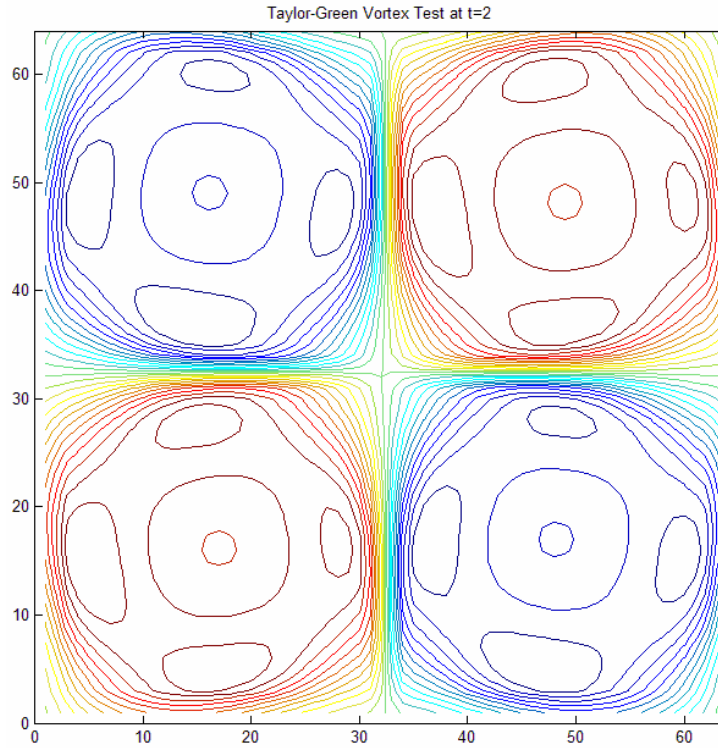


Figure B.3: Special Case of Taylor-Green Class of Vortices at time 2

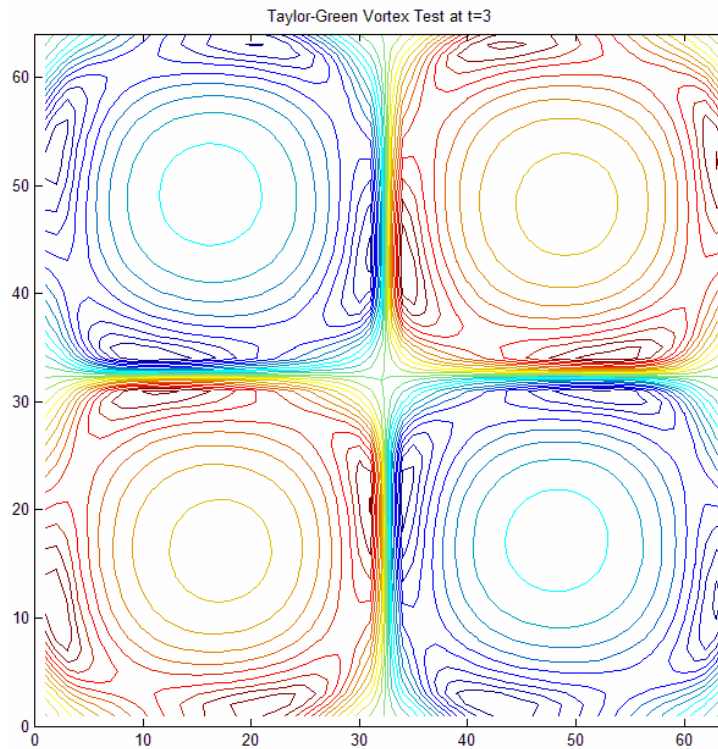


Figure B.4: Special Case of Taylor-Green Class of Vortices at time 3

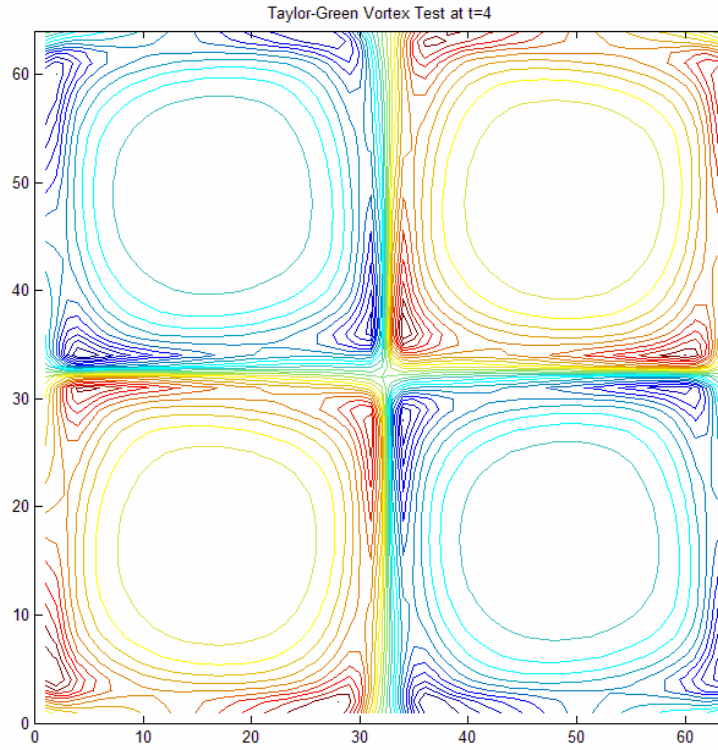


Figure B.5: Special Case of Taylor-Green Class of Vortices at time 4

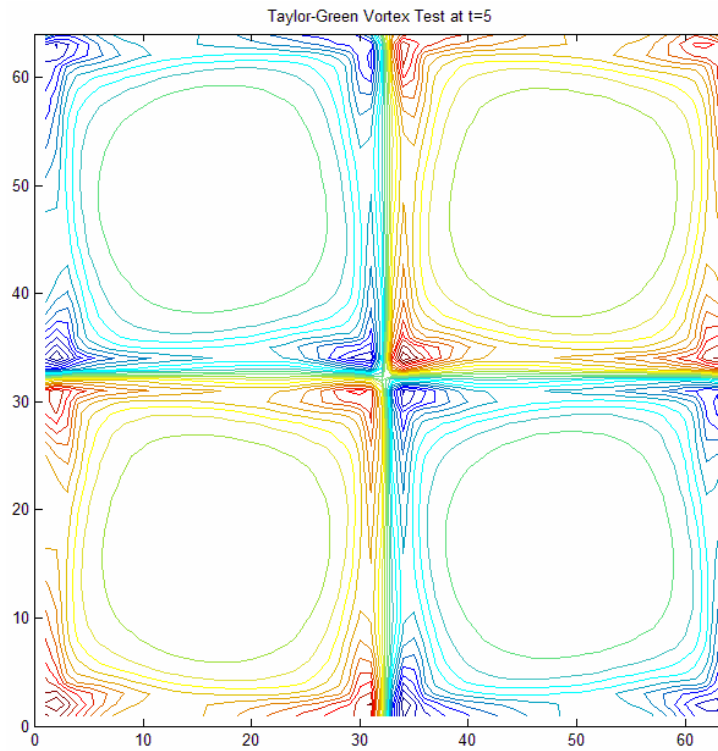


Figure B.6: Special Case of Taylor-Green Class of Vortices at time 5

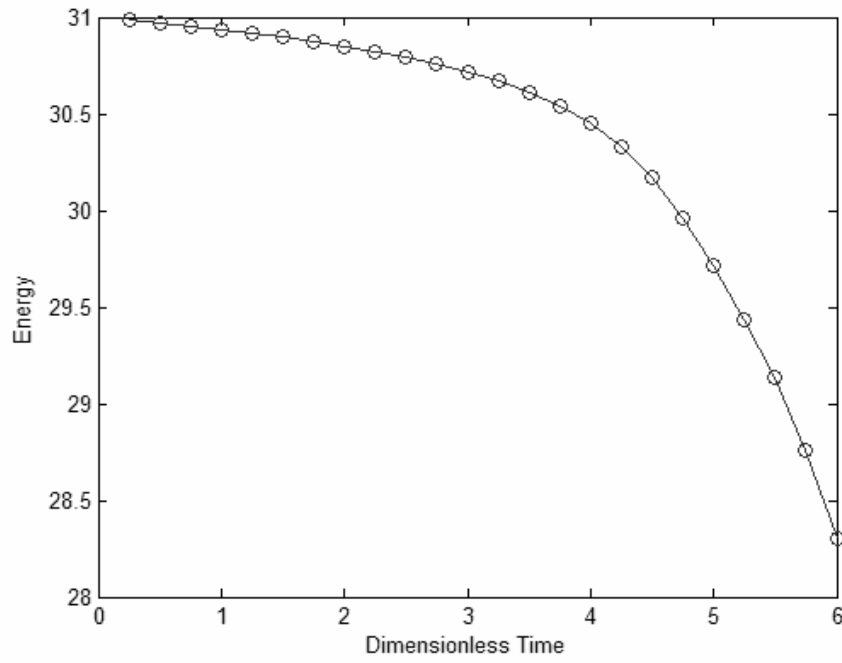


Figure B.7: Bulk Energy Variation

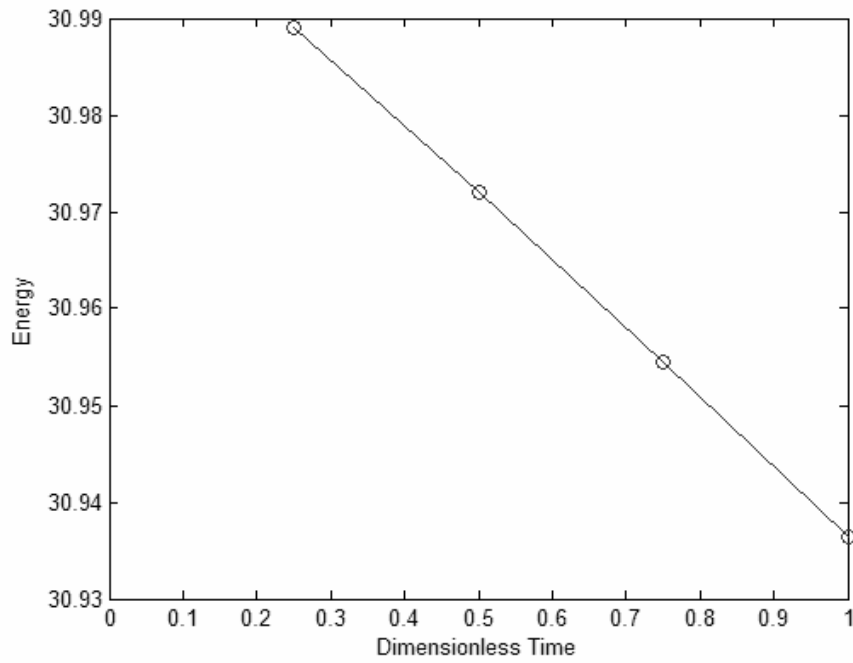


Figure B.8: Bulk Energy Variation (zoomed)

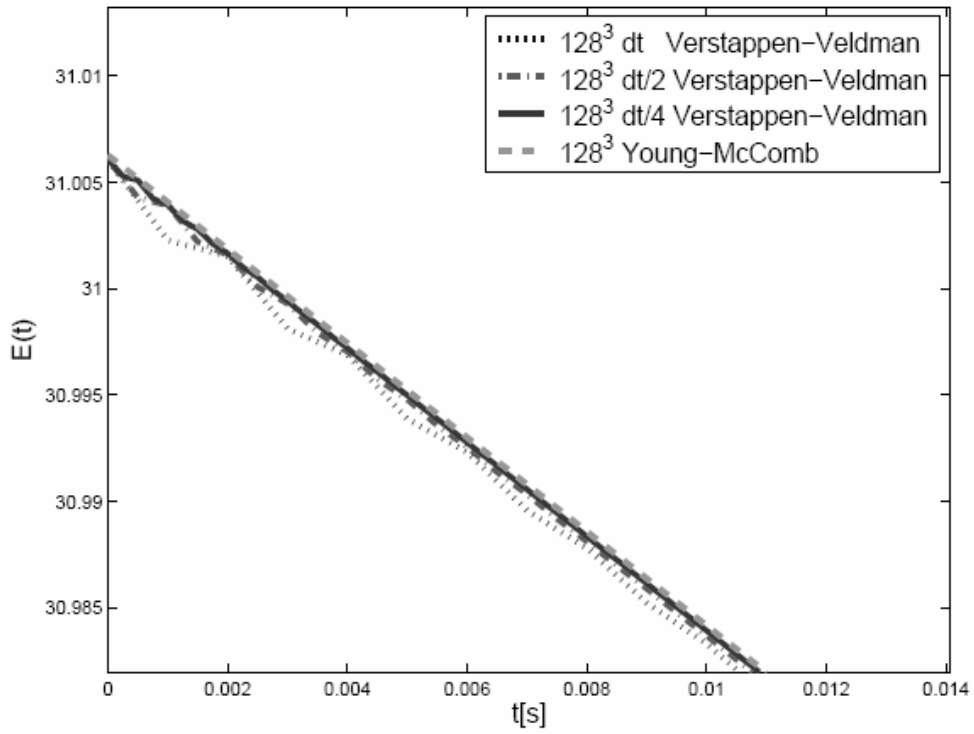


Figure B.9: Energy Dissipation (Kuczaj, 2003)

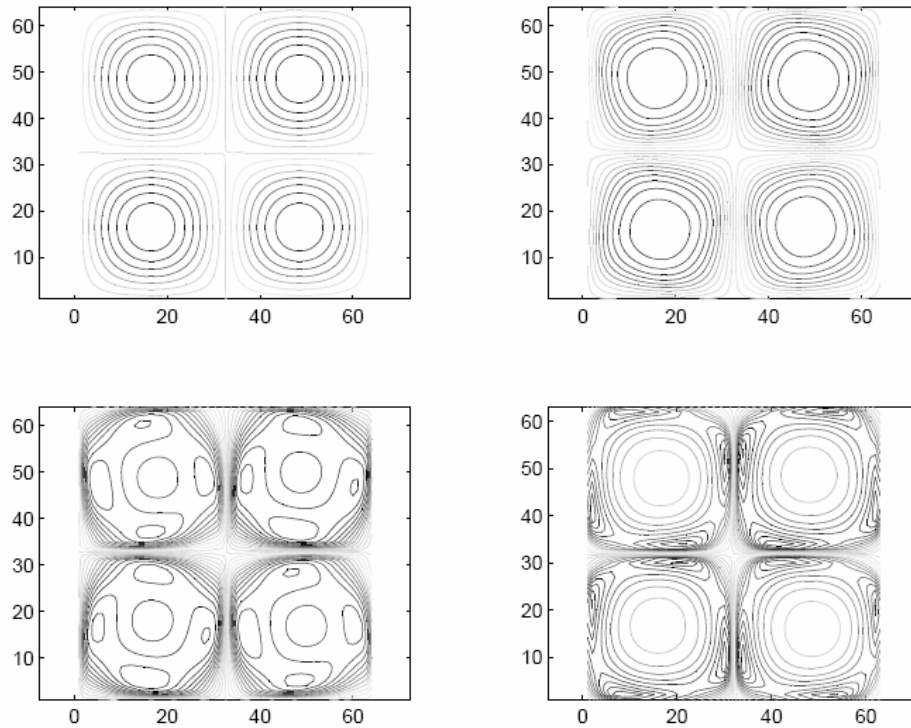


Figure B.10: Vorticity Vectors at $t=0, 1, 2, 3$ (Kuczaj, 2003)

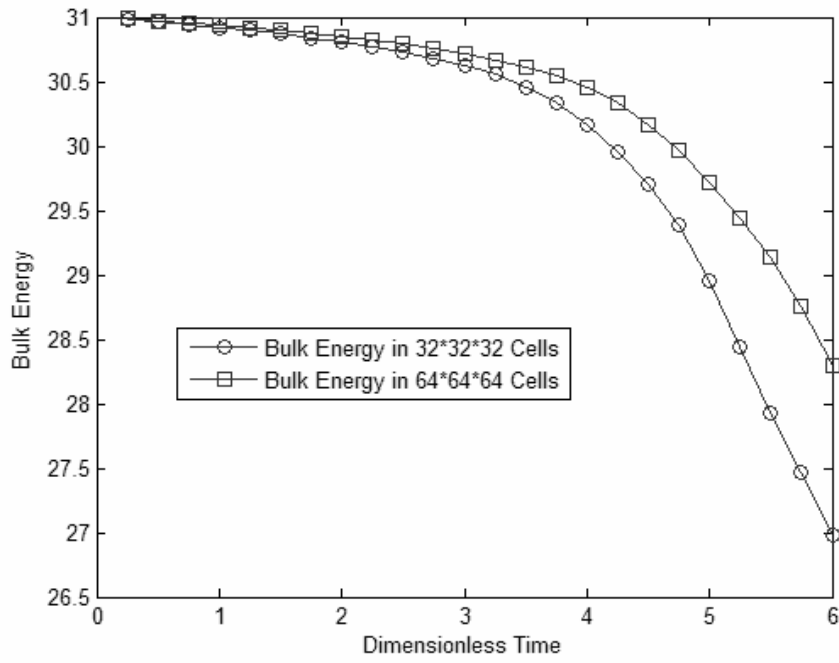


Figure B.11: Bulk Energy in 64^3 and 32^3 cells

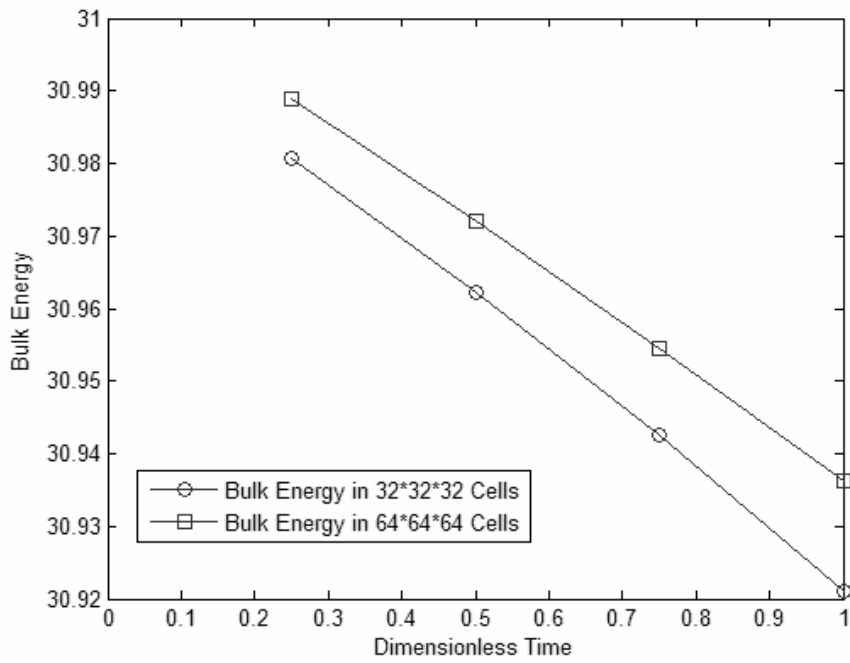


Figure B.12: Bulk Energy in 64^3 and 32^3 cells (zoomed)

B.4. Reynolds Number Effect on Energy Dissipation

The Reynolds number is a measure of the ratio between the forces trying to keep the flow in a laminar condition to those making flow regimes turbulent. In non-dimensional field it is also a measure for viscosity, or to be more precise $Re = \frac{1}{\nu}$. To study how the Reynolds number is affecting the energy, the same simulation was performed with three different Reynolds Numbers:

- 1) Low, $Re = 2000$,
- 2) Medium, $Re = 5600$,
- 3) Very High, $Re = 10^{25}$.

It is seen in figure B.13 that at higher Reynolds number, the fluid tries to retain its energy for a longer period of time. It can be also concluded that if $Re \rightarrow \infty$ the energy will stay constant for all time.

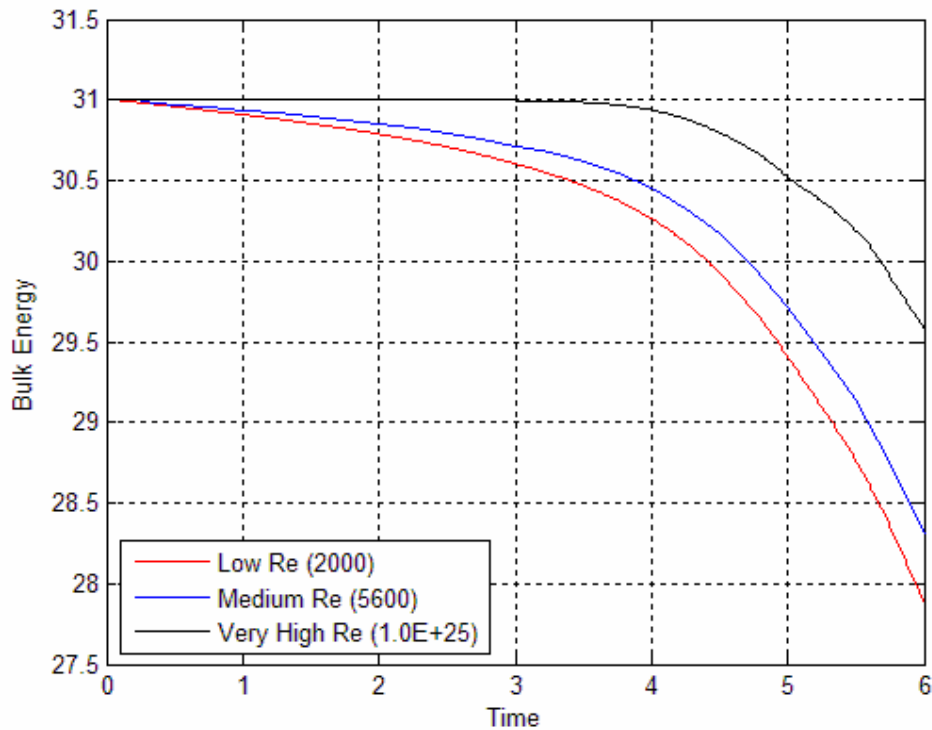


Figure B.13: Reynolds Number Effect on Bulk Energy

C. Connecting to Super Computers and Downloading/Uploading Data

C.1. Connecting

UNIX/Linux Operating System

While UNIX or Linux Operating System (OS) is used, connecting to super computer is very straight forward; because, the OS installed on supercomputers is always either Linux or UNIX with no exception, as far as I know. The only thing that has to be done is issuing the “SSH” command to establish a secure shell between your PC and the super computer. It is called secure shell; since, all the communications between the super computers and the PC are encrypted. Using “SSH” command is easy. One have to type “ssh” and in front of this command the web address or internet address (IP) of the supercomputer. In our case, we were using ASTER and TERAS supercomputers located in the Amsterdam and the command takes the form:

```
“ssh TERAS.sara.nl”  
“ssh ASTER.sara.nl”
```

Apple Operating System

Under Apple OS, still the procedure is straight forward. First a terminal has to be started. Then in the opened terminal, instead of using “SSH” command, one must use the “Login” command. The rest is as previous.

Microsoft Windows Operating System

The Microsoft Windows operating system does not directly support establishing secure shell terminal. The only possible shell possible using Microsoft Windows is “Telnet” session. This session is the same as secure shell. It is actually the older version of ssh session and it does not support any encryption.

the first solution is to download “Windows Services for UNIX” developed by Microsoft. This software is add-on component that can be installed on windows 2000, and XP, (XP Home Edition is not supported). This package simulates a UNIX/Linux Kernel, which allows the user to run and execute software and services developed for UNIX/Linux environment under Windows. Therefore, it should be possible to issue “SSH” command. This package is downloadable via following web address:

<http://www.microsoft.com/technet/interopmigration/unix/sfu/default.mspix>

Another solution is using the third party software packages. The most famous one is PUTTY. This software is the windows implementation of many UNIX/Linux commands. This software is very light and it can be downloaded from the following link for free:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

When it is downloaded and installed on Windows it can be executed as regular Windows packages. When this software is executed, in “Hostname (or IP address):” the web address or internet address of the super computer has to be provided. The “SSH” radio button must be selected and the “open” push button is pressed, Figure C.1.

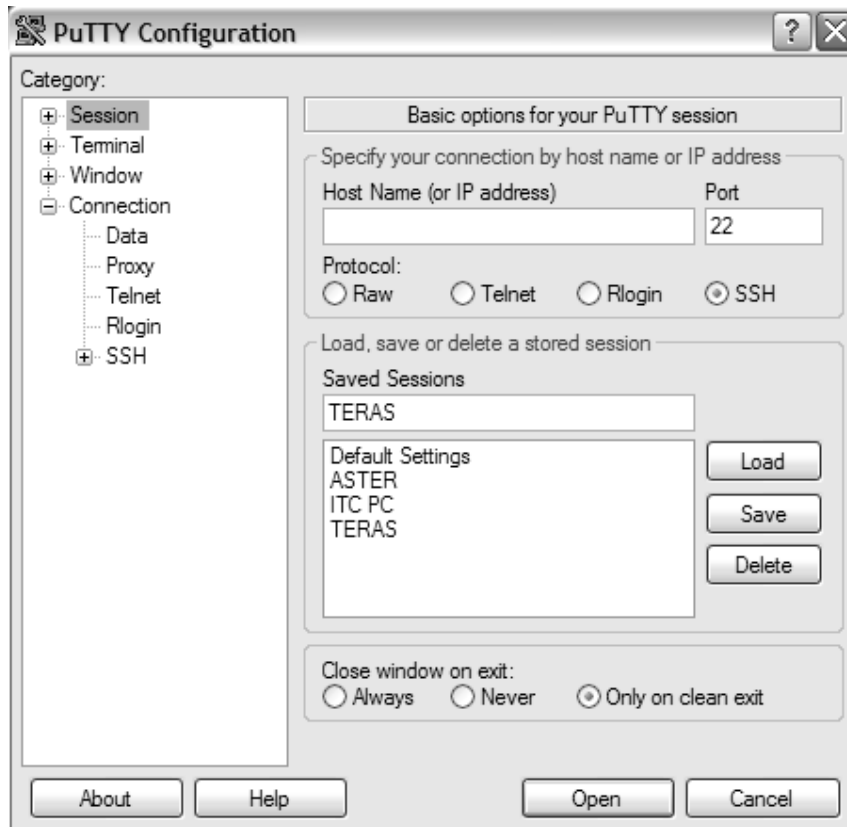


Figure C.1: Putty Screen

Mobile Windows Available on PDAs and Phones

A special version of Windows, called Mobile Windows (the older version is called Windows CE), is available for PDA and mobile phones. Fortunately a version of PUTTY is available for this Windows called “Pocket Putty”. It can be downloaded from:

<http://www.svpocketpc.com/reviews/pocketputty/PocketPuTTY.html>

for a screen shot of this software refer to Figure C.2.

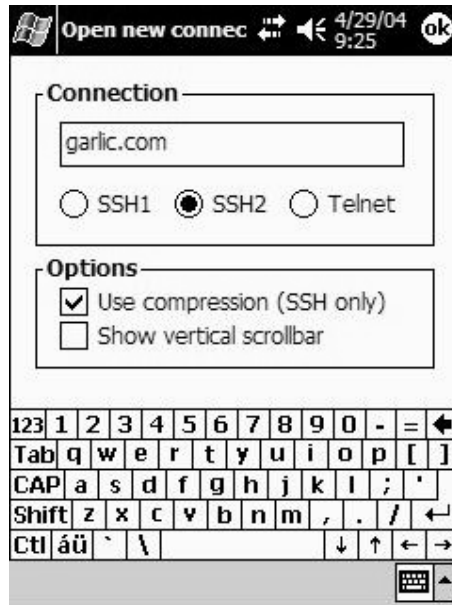


Figure C.2: Pocket Putty

C.2. Downloading and Uploading Data

Downloading and uploading data is easy and straight forward. The “SFTP” command must be used to connect to super computers. The SFTP stands for Secure FTP. This is the upgraded version of FTP, which stands for File Transfer Protocol. The only difference is that the SFTP encrypts everything transmitted in-between; thus, the usernames and passwords are secured. This command is directly available on UNIX/Linux operating systems. Windows does not support this command again. But if “UNIX Services for Windows” is installed or if PUTTY is downloaded and installed, this command becomes available. The SFTP command is not available for PDA, pocket PCs, and Mobile phones.

Once the connection to super computer is established using SFTP, by issuing GET or MGET command files and data can be downloaded and by issuing PUT or MPUT command the files and data can be uploaded. M stands for multiple and it is used when multiple files has to be uploaded or downloaded by one command.

D. Variable Used in the code

Table D.1: Variables used in Direct Numerical Simulation

Dimensions				
parameter	Name	Type	Array Range	Description
Dimensions	Nx	Integer	1	Number of Cell in X direction
	Ny	Integer	1	Number of Cell in Y direction
	Nz	Integer	1	Number of Cell in Z direction
	LogNx	Integer	1	Equals base 2 logarithm of Nx
	LogNy	Integer	1	Equals base 2 logarithm of Ny
	LogNz	Integer	1	Equals base 2 logarithm of Nz
Global Variables				
Velocities	u	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	x-component of Velocity
	v	double precision	(-2:Nx,-2:Ny,-2:Nz)	y-component of Velocity
	w	double precision	(-2:Nx,-2:Ny,-2:Nz)	z-component of Velocity
	uex	double precision	(1:Nx,1:Ny,1:Nz)	Old x-component of Velocity
	vex	double precision	(1:Nx,1:Ny,1:Nz)	Old y-component of Velocity
	wex	double precision	(1:Nx,1:Ny,1:Nz)	Old z-component of Velocity
Pressure	p	double precision	(0:Nx+3,0:Ny+3,0:Nz+3)	at time level n+1 & is defined at the centre of the cell
Temperature	T	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	at time level n+1 & is defined at the centre of the cell
Reynolds	Re	double precision	1	Reynolds Number
prandtl	pr	double precision	1	Prandtl Number
Boundary conditions	ip	integer	1	If equals to 1 the velocity is periodical in x direction and if equals to 0 it is not. Other values generates errors
	jp	integer	1	If equals to 1 the velocity is periodical in y direction and if equals to 0 it is not. Other values generates errors
	kp	integer	1	If equals to 1 the velocity is periodical in z direction and if equals to 0 it is not. Other values generates errors
	ux0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
	ux1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
	uy0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.

uy1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
uz0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
uz1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
vx0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
vx1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
vy0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
vy1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
vz0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
vz1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
wx0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
wx1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
wy0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
wy1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
wz0giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
wz1giv	logical	1	If it's true the Dirichlet BC is used, otherwise the Neumann.
ux0	double precision	1	
ux1	double precision	1	
uy0	double precision	1	
uy1	double precision	1	
uz0	double precision	1	
uz1	double precision	1	
vx0	double precision	1	
vx1	double precision	1	
vy0	double precision	1	
vy1	double precision	1	
vz0	double precision	1	
vz1	double precision	1	
wx0	double precision	1	
wx1	double precision	1	
wy0	double precision	1	
wy1	double precision	1	

	wz0	double precision	1	
	wz1	double precision	1	
mass flow	ipdrop	logical	1	
	jpdrop	logical	1	
	kpdrop	logical	1	
	mflow	double precision	1	
	s	double precision	(0:Nx+2,0:Ny+2,0:Nz+2)	
	mflows	double precision	(0:Nx+2,0:Ny+2,0:Nz+2)	
time integrate	time	double precision	1	Present Time
	dt	double precision	1	time step length
	beta	double precision	1	Parameterizes the one-leg time integration method
	n	Integer	1	time step count
	nt	Integer	1	number of time step of the present run
finite volume	order	Integer		Order of spatial discretization
	x	double precision	(0:Nx)	x coordinate of the grid
	y	double precision	(0:Ny)	y coordinate of the grid
	z	double precision	(0:Nz)	z coordinate of the grid
	dx	double precision	(-2:Nx+4)	
	dy	double precision	(-2:Nx+4)	
	dz	double precision	(-2:Nx+4)	
	dxs	double precision	(-2:Nx+3)	
	dys	double precision	(-2:Nx+3)	
	dzs	double precision	(-2:Nx+3)	
	dx3	double precision	(-1:Nx+3)	
	dy3	double precision	(-1:Nx+3)	
	dz3	double precision	(-1:Nx+3)	
	dxs3	double precision	(-1:Nx+2)	
	dys3	double precision	(-1:Nx+2)	
dzs3	double precision	(-1:Nx+2)		
Poisson Solver	d	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	e	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	lx1	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	lx2	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	lx3	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	ly1	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	ly2	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	ly3	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	lz1	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	lz2	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	lz3	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	ibx	Integer	(0:Nx-1)	
	iby	Integer	(0:Ny-1)	
	ibz	Integer	(0:Nz-1)	
	Cx	double precision	(0:Nx-1,1:LogNx)	real part of Fourier transform
Sx	double precision	(0:Nx-1,1:LogNx)	Imaginary part of the Fourier transform	
Cy	double precision	(0:Ny-1,1:LogNy)	real part of Fourier transform	
Sy	double precision	(0:Ny-1,1:LogNy)	Imaginary part of the Fourier transform	
Cz	double precision	(0:Nz-1,1:LogNz)	real part of Fourier	

				transform
	Sz	double precision	(0:Nz-1,1:LogNz)	Imaginary part of the Fourier transform
	Mass	Integer	1	
	nsf	Integer	1	Number of step after which the velocity & pressure field are written to a file.
	nsb	Integer	1	Number of step after which the bulk values are written to a file.
Post Processes	u1mn	double precision	(1:Ny)	
	u2mn	double precision	(1:Ny)	
	u3mn	double precision	(1:Ny)	
	u4mn	double precision	(1:Ny)	
	v1mn	double precision	(0:Ny)	
	v2mn	double precision	(0:Ny)	
	v3mn	double precision	(0:Ny)	
	v4mn	double precision	(0:Ny)	
	w1mn	double precision	(1:Ny)	
	w2mn	double precision	(1:Ny)	
	w3mn	double precision	(1:Ny)	
	w4mn	double precision	(1:Ny)	
	uvmn	double precision	(1:Ny)	
	uwmn	double precision	(1:Ny)	
	vwmn	double precision	(1:Ny)	
	umnt	double precision	(-2:Nx,-2:Ny,-2:Nz)	
	vmnt	double precision	(-2:Nx,-2:Ny,-2:Nz)	
wmnt	double precision	(-2:Nx,-2:Ny,-2:Nz)		
umntz	double precision	(-2:Nx,-2:Ny)		
vmntz	double precision	(-2:Nx,-2:Ny)		
wmntz	double precision	(-2:Nx,-2:Ny)		
mntime	double precision	(1:Ny)		
Work Space	r	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	a	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	b	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
	c	double precision	(-2:Nx+3,-2:Ny+3,-2:Nz+3)	
Logging Units	inpars	Integer	1	Paramaters
	ingrid	Integer	1	dat/grid.dat
	infld	Integer	1	dat/fld***.dat
	inbcs	Integer	1	bdnconds
	intmp	Integer	1	dat/tmp***.dat
	inmns	Integer	1	dat/mns***.dat
	outfld	Integer	1	dat/fld***.dat
	outtmp	Integer	1	dat/tmp***.dat
	outmns	Integer	1	dat/mns***.dat
	outdiv	Integer	1	dat/maxdiv***.dat
	outcfl	Integer	1	dat/maxcfl***.dat
	outbulk	Integer	1	dat/bulk***.dat
outerr	Integer	1	errors	

Table D.2: Variables and parameters used in the text

Parameter/ Variable	Description	Formula or Value
u	Stream wise velocity	-
v	Vertical velocity	-
w	Spanwise velocity	-
p	Pressure	-
Re	Reynolds number	$\frac{uh}{\nu}$, variable, 5600 & 10000
Re_{u_τ}	Reynolds number based on friction velocity	$\frac{u_\tau h}{\nu}$
ν	Viscosity	variable ($\sim \frac{1}{Re}$)
h or δ	Half the channel height	1
H	Boundary Layer Height	Variable
g	Gravity acceleration constant	-
G	General filter kernel function	-
u^+	U in wall coordinate	$\frac{u}{u_\tau}$
y^+	Y in wall coordinate	$\frac{yu_\tau}{\nu}$
u_τ	Friction velocity	$\sqrt{\frac{\tau_w}{\rho}}$
$u_{\tau x}$	Friction velocity in x direction	-
$u_{\tau z}$	Friction velocity in z direction	-
τ_w	Shear stress	$\rho \nu \left(\frac{d\langle u \rangle}{dy} \right)_{y=0}$
u_{ij}	Total velocity	$\sqrt{u_j^2 + w_j^2}$
dt	Time step	-
n_t	Number of time step	$t = n_t dt$
y_0^+	Roughness in wall coordinate	$\frac{y_0 u_\tau}{\nu}, \left\{ \begin{array}{l} smooth.wall = 0.1249 \\ rough.wall = 0.4996 \end{array} \right\}$
y_0	roughness	-
κ	Von Karman constant	0.4

Reference:

- Abouali, M., 2002. Integration Techniques, The Complete Reference, 1. Sara, 222 pp.
- Albertson, J.D., 1996. Large eddy simulation of land-atmosphere interaction, University of California, 185 pp.
- Albertson, J.D. and Parlange, M.B., 1999. Surface length scales and shear stress: Implications for land-atmosphere interaction over complex terrain. *Water Resources Research*, 35(7): 2121-2132.
- Anderson, J.D., 1995. *Computational Fluid Dynamics: The Basics with Applications*, 1. McGraw Hill, 545 pp.
- Balaras, E. and Benocci, C., 1994. Subgrid-scale models in finite-difference simulations of complex wall bounded flows. *AGARD CP*, 2.1: 551.
- Balaras, E., Benocci, C. and Piomelli, U., 1996. Two-layer approximate boundary conditions for large-eddy simulations. *AIAA J.*, 34: 1111.
- Businger, J.A., 1982. *Atmospheric Turbulence and Air Pollution Modelling*. Reidel, 358 pp.
- Chapman, D., 1979. Computational aerodynamics, development and outlook. *AIAA J.*, 17: 1293-313.
- Dutton, J.A. and Fichtl, G.H., 1969. Approximate Equations of Motion for Gases and Liquids. *Journal of the Atmospheric Sciences*, 26(2): 241-254.
- Duynkerke, P.G. and Driedonks, A.G.M., 1987. A Model for the Turbulent Structure of the Stratocumulus-Topped Atmospheric Boundary Layer. *Journal of Atmospheric Sciences*, 44(1): 43-64.
- Geurts, B.J., 2004. *Elements of Direct and Large-eddy Simulation*, 1. Edwards, 344 pp.
- Groetzbach, G., 1987. Direct numerical and large eddy simulation of turbulent channel flows. *Encyclopedia of Fluid Mechanics*, V6, ed. NP Cheremisinoff: 1337-91.
- Holtslag, A.A.M. and Ek, M., 1996. Simulation of surface fluxes and boundary layer development over the pine forest in Hapex-Mobilihy. *Journal of Applied Meteorology*, 35: 202-213.
- Kim, J., Moin, P. and Moser, R., 1987. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177: 133 - 166.
- Kuczaj, A., 2003. Validation of 3D DNS Solvers for incompressible Navier-Stokes equations - description of codes from the University of Edinburgh and the University of Groningen, Twente University, Enschede.

- Lee, X., Massman, W. and Law, B., 2005. Handbook of Micrometeorology: A Guide for Surface Flux Measurement and Analysis (Atmospheric and Oceanographic Sciences Library), 1. Springer, 268 pp.
- Masson, W., Turnbull, R. and Freeman, J., 2006. Free Slip Boundary Condition, <http://www.mcc.monash.edu.au/twiki/view/Software/UnderworldBoundaryFreeSlip>. Monash University Web Site.
- Oke, T.R., 1988. Boundary Layer Climates, 1. Routledge, 450 pp.
- Parlange, M.B., Eichinger, W.E. and Alberson, J.D., 1995. Regional scale evaporation and the atmospheric boundary layer. In: Reviews of Geophysics, 33(1995)1, pp. 99-124.
- Piomelli, U. and Balaras, E., 2002. Wall-layer models for large-eddy simulations. Annual Review of Fluid Mechanics, 34: 349-374.
- Piomelli, U., Balaras, E., Pasinato, H., Squires, K.D. and Spalart, P.R., 2003. The inner-outer layer interface in large-eddy simulations with wall-layer models. Int. J. Heat Fluid Flow, 24: 538-550
- Piomelli, U., Ferziger, J., Moin, P. and Kim, J., 1989. New Approximate Boundary-Conditions for Large Eddy Simulations of Wall-Bounded Flows. Physics of Fluids a-Fluid Dynamics, 1(6): 1061-1068.
- Polyanin, A.D., Kutepov, A.M., Vyazmin, A.V. and Kazenin, D.A., 2002. Hydrodynamics, Mass and Heat Transfer in Chemical Engineering. Taylor & Francis, London.
- Pope, S.B., 2000. Turbulent Flows, 1. Cambridge University Press, 770 pp.
- Raupach, M.R., Antonia, R.A. and Rajagopalan, S., 1991. Rough-Wall Turbulent Boundary Layers. Applied Mechanics Reviews, 44(1): 1 - 26.
- Sagaut, P. and Meneveau, C., 2004. Large Eddy Simulation for Incompressible Flows 1. Springer, 426 pp.
- Schumann, U., 1975. Subgrid-scale model for finite difference simulation of turbulent flows in plane channels and annuli. J. Comput. Phys., 18: 376-404.
- Schumann, U., 1989. Large-eddy simulation of turbulent diffusion with chemical reactions in the convective boundary layer. Atmos. Env. , 23: 1713 - 1727.
- Spalart, P., Jou, W., Strelets, M. and Allmaras, S., 1997. Comments on the feasibility of LES for wings and on a hybrid RANS/LES approach. Advances in DNS/LES(ed. C Liu, Z Liu,): 137-48.
- Stull, R.B., 1999. An Introduction to Boundary Layer Meteorology, 1. Springer, 684 pp.
- Taylor, G.I., 1935. Statistical Theory of Turbulence. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences
- 151: 421-478.

- Taylor, G.I. and Green, A.E., 1937. Mechanism of the Production of Small Eddies from Large Ones. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 158(895): 499-521.
- Verstappen, R. and Van der Velde, R.M., 2006. Symmetry-preserving discretization of heat transfer in a complex turbulent flow. Journal of Engineering Mathematics, 54(4): 299-318.
- Verstappen, R. and Veldman, A.E.P., 1997. Direct numerical simulation of turbulence at lower costs. Journal of Engineering Mathematics, 32(2-3): 143-159.
- Verstappen, R. and Veldman, A.E.P., 2003. Symmetry-preserving discretization of turbulent flow. Journal of Computational Physics, 187(1): 343-368.
- Wikipedia1, Navier-Stokes equations. http://en.wikipedia.org/wiki/Navier_Stokes.
- Wikipedia2, No-Slip Condition. http://en.wikipedia.org/wiki/No-slip_boundary_condition.
- Wikipedia3, Siméon Denis Poisson. <http://en.wikipedia.org/wiki/Poisson>.
- Wooding, R.A., Bradley, E.F. and Marshal, J.K., 1973. Drag due to regular arrays of roughness elements of varying geometry. Boundary-Layer Meteorology, 5: 285 - 308.
- Zanoun, E.S., Durst, F. and Nagib, H., 2003. Evaluating the law of the wall in two-dimensional fully developed turbulent channel flows. Physics of Fluids, 15(10): 3079-3089.

