EVALUATING THE EFFECT OF TOPOGRAPHIC ELEMENTS ON WIND FLOW: A COMBINED NUMERICAL SIMULATION - NEURAL NETWORK APPROACH

Girma T. Bitsuamlak

A Thesis

In

The Department of Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

September 2004

Your file Votre référence
ISBN: 0-612-96950-9
Our file Notre référence
ISBN: 0-612-96950-9

Abstract

Evaluating the effect of topographic elements on wind flow: A combined numerical simulation - neural network approach

Girma T. Bitsuamlak, Ph.D.
Concordia University, 2004

Wind pressures on buildings and other structures, pedestrian level winds, and wind-induced dispersion of pollutants in urban locations depend, among other factors, on the velocity profile and turbulence characteristics of the upcoming wind. These, in turn, depend on the roughness and general configuration of the upstream topography. Consequently, wind standards and codes of practice typically assume simplified upstream topography conditions of homogeneous roughness or provide explicit corrections only for a limited number of specific topographies such as single hills, valleys, or escarpments. For all other more complex situations, the practitioner is referred to physical simulations in a boundary layer wind tunnel (BLWT).

This thesis evaluates the effect of upstream topography on wind flow using two mathematical approaches: Computational Fluid Dynamics (CFD) and Neural Network (NN) chosen in lieu of traditional BLWT test. CFD-based numerical simulation usually consists of discretizing and solving a set of partial differential equations (the so called Reynolds-averaged Navier-Stokes equations and standard k-ε turbulence model) describing the wind flow over a number of different topographic elements. For this purpose a robust Grid Generator, suitable for geometries characterized by curves and slopes, and a specialized CFD tool have been designed using an object-oriented approach

and implemented in C++ programming language. Emphasis has been given to several numerical details and to the incorporation of influential parameters such as ground roughness. The associated accuracy of the proposed CFD tool has been quantified and validated against the results of the extensive review carried out.

Despite continuous progress in hardware/software technology resulting in fewer resource requirements, the practical utilization of CFD-based numerical simulations to predict design wind load parameters is rather limited. To address this issue, a new combined CFD-NN approach in which the NN model is trained with CFD-generated data has been proposed and developed. Since the NN approach is defined on the basis of connections between system state variables (input, internal and output variables) with only limited knowledge of the "physical" behavior of the system, output variable values (speed-up ratio in the present case) are produced following input of simple geometrical parameters describing the topography and roughness of the ground.

In this combined approach, the domain expert first carries out the complex numerical simulations and validations, then trains the NN model and makes it available for use by the end user who avoids therefore dealing with complex numerical simulations. The results compare well with independent sets of experimental data gathered from literature for a wide spectrum of terrain geometries, which includes multiple hills that are not covered by wind design standards and codes of practice. Ultimately, the new approach requires fewer resources from the end user compared to running BLWT tests, using BLWT-NN combination or CFD simulations alone, while producing results of comparable quality.

Acknowledgment

Most of all I would like to express my deepest gratitude to my supervisors Dr. Theodore Stathopoulos and Dr. Claude Bédard for their excellent guidance, inspiring wisdom, unparalleled support, encouragement, dedication and understanding during the course of this study. In fact, their influence has started long time before I joined Concordia through their excellent publications as well as through the kindness and commitment extended to me while joining their research group. Further I am grateful for the chance given to me to attend various international conferences.

The valuable feedback from the members of my examining committee Drs. H. Tanaka, R. Zemeureanu, A. Hammad, Clement Lam and H. Rivard is acknowledged with thanks.

Very special thanks to the faculty and staff in the department of Building, Civil and Environmental Engineering and all my friends in Montreal.

I would like also to thank Drs. W. Ghaly and F. Nebebe for their useful discussions on general aspects of my research.

The financial support received from my supervisors and Concordia University through a Graduate Fellowship, International Tuition Fee Remission and Partial Tuition Scholarship for International Students is fully acknowledged.

Finally, I would like to thank my parents, sisters and brother for their continuous support and encouragement. A very special note of thank to my wife as well as my friend, Martha, for her endless love and support. I really appreciate her organizational skills, commitment to proof read the manuscript, and fruitful discussions.

To my parents

Table of Contents

List of Figures

List of Tables

Nomenclature

A_E, A_W, A_N, A_S, A_P	coefficients in the discretized equations
$A(\|Pe\|)$	a hybrid scheme function defined by $A(\|Pe\|) = \max[0,(1-0.5\|Pe\|)]$
b	constant part in the standard discretized equation
B	control volume boundary
B^U, B^W	coefficients in the discretized momentum equations
C_1, C_2, C_μ	constants in the k-ε turbulence model equations
Cp	mean pressure coefficient
D	Diffusion
E	mean square error
$\overline{E}$	mean square error averaged over all data points
f	distortion function
F	Convection
G_1, G_2	convective terms normal to CV boundaries
$h(\xi)$, $h(\eta)$	scale factors in ξ - and η- direction respectively
H	hill height
I	input to the neurons
J	Jacobean of the transformation

K	turbulent kinetic energy
L	distance down wind of the crest of a hill to where the ground elevation is half the height of a hill
m_p	mass source of the control volume
max[a, b]	a function that returns maximum value of its arguments a and b
p	pressure field
P	augmented pressure field
P	CV center grid point
Pe	Peclet number
r	relative residual
R	correlation coefficient
S	source term
S_c	constant source term
S_p	variable source term
T	target output
U, W	mean velocity components in x, z direction
u(t), w(t)	instantaneous velocity components in x, z direction
u',w'	fluctuating velocity in x, z direction
U^*	friction velocity
$U^*,W^*,p^*,k^*,\varepsilon^*$	estimated values for U, W, p, k, ε respectively

U_o	upstream reference velocity
W_{ji}	value of the weight connecting neuron j with neuron i
X	net input at each neuron
x, z	space coordinates
Y	output from processing neuron
z_p	z-coordinate of the node adjacent to the wall
z^+	non-dimensional boundary layer coordinate
z_o	roughness length

Greek Symbols

$\alpha_1, \beta_1, \gamma_1$	coordinate transformation parameters
α	exponent of the power-law wind speed profile
θ	neural network learning rate
β	decay coefficient for the decrease in the speed up with the height
Γ	transportation coefficient
δ	error signal used to correct connection weights in the neural network
δ_{ij}	kronecker delta , $\delta_{ij}=1$ if $i=j$ and $\delta_{ij}=0$ if $i\neq j$
Δ_ξ, Δ_η	grid point spacing in ξ and η-direction respectively
$\delta\xi_w, \delta\xi_e$	distance between central point *P* and its east and west neighboring grid points in ξ-direction respectively
$\delta\eta_s, \delta\eta_n$	distance between central grid point *P* and its south and east neighboring points in η-direction respectively
ε	turbulent kinetic energy dissipation
ε_p	turbulent energy dissipation of the node adjacent to the wall
ν	kinematic viscosity
ν_t	turbulent eddy viscosity
ρ	density of air

σ_k, σ_ε	constants in the k-ε turbulence model equations
Φ	general dependent variable in the governing equations
ΔS	speed-up ratio
ΔS_{max}	represents the relative speed-up ratio at the crest of a hill near the surface
$\Delta\xi$, $\Delta\eta$	dimensions of the control volume in ξ and η-direction respectively
κ_1	constant used in the speed-up ratio expression recommended by NBCC
κ	constant in the law of the wall
ξ,η,ζ	transformed space coordinates
τ_w	wall shear stress
Ω	neural network weight maximum growth factor
$\partial/\partial x$	partial differentiation with respect to variable x

Subscripts

e, w, n, s	east, west, north, south faces of the control volume
E, W, N ,S	east, west, north, south neighboring grid points of the reference grid point
i, j, k	subscripts denoting x, y and z direction
Nb	neighbors
no	number of output neurons
nd	neural network training data size
NE, NW, SE, SW	north-east, north-west, south-east, and south-west neighboring grid points of the reference grid point
P	data point index
P	value at a node adjacent to the wall

Abbreviations

2D	two-dimensional
3D	three-dimensional
NN	neural network
BLWT	boundary layer wind tunnel
CCNN	cascade correlation neural network
CD	computational domain

CFD	computational fluid dynamics
CV	control volume
CWE	computational wind engineering
DH	distance between two hills
FSUR	fractional speed-up ratio
LHS	left hand side
NBCC	national building code of Canada
NURB	Non-Uniform Rational B-spline curves
PC	personal computer
RANS	Reynold's averaged Navier- Stoke's
RHS	right hand side
RNG k-ε	renormalization group k-ε turbulence model
RSM	Reynold's stress turbulence model
SIMPLE	semi implicit method for pressure liked equations
SIMPLEC	SIMPLE consistent
TDMA	tri-diagonal matrix algorithm

1 Introduction

Wind pressures on buildings and other structures, pedestrian level winds, and wind-induced dispersion of pollutants in urban locations depend, among other factors, on the velocity profile and turbulence characteristics of the upcoming wind. These, in turn, depend on the roughness and general configuration of the upstream terrain. Consequently, wind standards and codes of practice typically assume simplified upstream terrain conditions of homogeneous roughness or provide explicit corrections only for a limited number of specific topographies such as isolated symmetrical hills, valleys, or escarpments. For all other more complex situations, the practitioner is referred to physical simulation in a boundary layer wind tunnel (BLWT).

Canada's National Building Code (NBCC 1995) is one of the most progressive in the world in its treatment of wind loads. It provides simple exponential equation for estimating "speed up" factors for flows over isolated symmetrical 2D escarpments, hills and valleys. However no provision is available in NBCC for non-symmetrical isolated 2D hills and valleys, multiple hills of any kind, or complex terrain. Even for some types of geometries (for example very steep hills) covered in the NBCC (1995), the provided values are slightly different from the BLWT experimental values mainly because of the shortcomings of the underlying theoretical approaches that were developed without consideration of non-linearity. Hence there is room for improvement and a need for a simple and efficient approach to determine the effect of topography on wind loads for the cases that are not covered by the NBCC provisions.

The evaluation of topographic effect on wind flow is often made using BLWT tests. BLWT procedures are relatively reliable and also have wide acceptance in the engineering community. However they are very expensive, they require the use of a specialized wind tunnel, the time-consuming manufacturing of physical models and different sensitive pressure and velocity reading instruments. Recent advances in computational hardware and software technology coupled with their cost-effectiveness have resulted in the evolution of Computational Wind Engineering (CWE). This makes computational approaches attractive for the evaluation of wind flow over different types of topography. In the present study, a computational approach using a Computational Fluid Dynamics (CFD) technique has been adopted instead of expensive BLWT tests.

1.1 Application Areas

The ability to simulate wind flow over different types of topography is of significance in the following fields.

Wind Loading: To assess wind-loading estimates for buildings and other structures with upstream complex terrain. Such estimates are especially useful for buildings in mountainous areas, hilltop installation of microwave receiver stations, for power lines in complex terrain, industrial projects, siting of airports and windmills. These applications constitute the main motivation for the present study.

The design of some facilities, optical and radio telescopes and communication towers, typically built in complex mountainous terrain requires consideration of detailed operating conditions in addition to the structural aspects. Hence, information regarding the turbulence of the wind, its variation with height and the spectrum of wind is required for their performance and operation (Zarghamee, 2001).

Built Environment: To provide realistic wind flow characteristics for numerical studies related to built environment such as pedestrian level wind flow, wind driven rain, natural ventilation and indoor air quality.

Wind Energy: To investigate wind power availability over complex terrain as well as to find the optimum location for wind turbines and wind farms.

Environmental: Wind field simulation is becoming a prerequisite in order to develop a systematic strategy for air pollution control in complex terrain. The wind environment can be influential in determining local human exposures to environmental pollution. It is also important in the study of bed load and suspended load transport and redistribution of materials (sand, dust and snow) over complex terrain

Meteorology: Site-specific wind forecasts are another potential application. In this context, wind output is taken from large-scale, regional or mesoscale forecast models and then the numerical models with high-resolution topography can be used for specific sites within the large-scale grid. In other words the numerical models can be used to "interpolate" for specific small sites based on the output from regional models.

Forestry: The prediction of flow structure over and through the vegetation covering the terrain is very important for studying forest fire propagation. It is also important in forest risk analysis, and pollutant deposition in forest (Ferreira et al., 1995).

1.2 Objectives and scope of study

The purpose of this study is therefore to enable the end-user to generate wind loads computationally for a wide spectrum of terrain geometries that includes single and multiple hills, escarpments and valleys accurately and efficiently. Emphasis has been

given to the generation of wind design parameters for additional topographic cases such as multiple hills that are not covered in standards and codes of practice, as well to enhance the accuracy of prediction for cases where the code values deviate from the BLWT results as in the case of very steep hills.

With the objective of bringing into perspective the significance of the problem as well as past and present research efforts in this area, data collected from literature has been analyzed to identify common points of agreement and areas of concern. This further facilitates prioritizing of numerical details and influential parameters for incorporation into the proposed numerical model. As part of the analysis, results obtained using different methods have also been compared mainly to validate the quality and accuracy of the present work. Applications of computational wind engineering are in their infancy (Stathopoulos, 1997) therefore their use must be preceded with rigorous validations by comparing computed results with experimental data. This should be happening until such time that confidence is built on the numerical approaches.

Current utilization of numerical simulations to predict wind speed-up in practical applications is rather limited. This is due to, among other reasons, the unavailability of specialized and cost-effective numerical tools targeted to give solutions for wind flow over different topographies. Hence, a second objective has been the design of a specialized numerical tool fine-tuned for the application of wind flow simulation over different types of terrain. To this end, special emphasis has been given to incorporate influential parameters such as the ground roughness and to develop grids that represent the terrain accurately.

Due to the complexity generally associated with the CFD based numerical method, its use requires background knowledge and experience, fast hardware with large memory, and availability of ample computational time. Thus in parallel to developing these numerical tools, simple and efficient ways to convey the results produced by such simulations to the engineering profession (end-user) must be devised. This constitutes the third objective of the present study. To this effect, a new computational approach has been developed and validated. The proposed and developed architecture combines the aforementioned complex simulations with less computationally intense approach based on Neural Networks (NN).

The scope of the present study covers therefore the numerical evaluation of the steady state wind flow over different types of topographies including isolated hills, multiple hills, valleys and escarpments in two-dimensional cases. The numerical simulations are based on standard CFD techniques and turbulence models. The study also devised simple and efficient ways of conveying complex numerical results and further expanding them to new cases by designing a new combined CFD-NN approach. For this purpose, novel numerical techniques and tools have been designed and implemented following an object-oriented approach using C++ language.

1.3 Plan of thesis

This thesis consists of six chapters, a list of references and an appendix. Chapter 2 describes the necessary theoretical background of the two numerical approaches, CFD and NN, used in the present study. It reviews the state-of-the-art in numerical evaluation of wind flow over different topographies and overviews general neural network applications in wind engineering. Comparison between five different methods: CFD,

BLWT, analytical, full scale measurements and NBCC (1995) provisions, are also made wherever applicable. Chapter 3 describes the numerical methodology including the CFD tool details and fine tunings made for the present application as well its design and analysis process using object-oriented approach. Further similar description is given regarding the design and development of the grid generator tool. Numerical simulations result for different topographies including single and multiple hills, escarpments and valleys is presented in Chapter 4. Also in the same chapter, numerical simulations result validation through comparison with BLWT experimental data and full-scale measurements available in literature is presented. Chapter 5 presents the analysis, design, and development of NN tool and NN models trained with CFD-generated data by means of the combined CFD-NN approach. CFD-NN results and validation are also presented in this chapter. Finally, Chapter 6 summarizes the findings of this study, presents its original contributions as well as recommendations for future work.

2 Background knowledge and literature review

2.1 Introduction

This chapter reviews the state-of-the-art in the numerical evaluation of wind flow over different types of topographies and the application of NN in wind engineering. To facilitate comprehension of the literature review, terms that are frequently used such as speed-up ratio are defined, NBCC provisions (1995) are discussed, and wind flow governing equations are described. Following this, a literature review dealing with numerical evaluation and NN applications is presented.

2.2 National Building Code User's Guide (1995) – structural commentaries (part 4)

The increase in wind velocity found to occur on the crests of topographic features such as hills and ridges is termed speed-up. This speed-up is usually defined in terms of speed-up ratio, ΔS, which is given by

$$\Delta S = \frac{U(z) - U_0(z)}{U_0(z)} \tag{2.1}$$

where $U(z)$ is the velocity at height z above the local hill surface and $U_o(z)$ represents upstream reference velocity at the same height z as shown in Figure 2.1.

The NBCC recommends the following speed-up ratio expression:

$$\Delta S = \Delta S_{max}\left(1 - \frac{|x|}{\kappa_1 L}\right)e^{(-\beta z/L)} \tag{2.2}$$

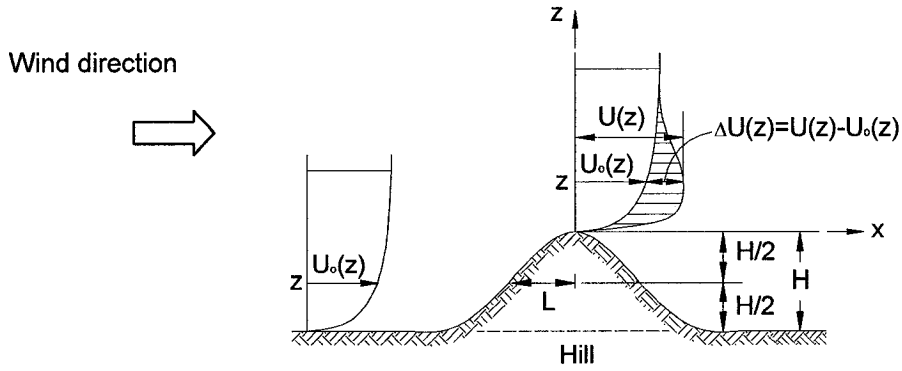


Figure 2.1 Definition of wind speed-up over hills (NBCC 1995)

where ΔS_{max} represents the relative speed-up ratio at the crest near the surface, β represents a decay coefficient for the decrease in the speed-up with height, ΔS represents speed-up ratio, L represents the distance upwind of the crest to where the ground elevation is half the height of the hill, H represents height of the hill, κ_1 represents a constant given in Table 2.1 and x represents the horizontal location of a position under consideration with respect to the crest of the hill (Figure 2.1). Values of β and ΔS_{max} are given in Table 2.1. Their values depend on the shape and steepness of the hill as given.

Table 2.1 Parameters for maximum speed-up over low hills (NBCC 1995)

Hill Shape	ΔS_{max} [1]	β	κ_1	
			x<0	x>0
2-dimensional ridges (or valleys with H negative)	2.2 H/L	3	1.5	1.5
2-dimensional escarpments	1.3 H/L	2.5	1.5	4
3-dimensional axisymmetrical hills	1.6 H/L	4	1.5	1.5

[1] For H/L > 0.5, assume in the formula that H/L = 0.5 and substitute 2H for L in equation (2.2)

2.3 Governing wind flow equations

For fluid flow in general and wind flow over topography in particular, the governing equations consist of a set of Partial Differential Equations (PDE). These PDEs include the so-called continuity and momentum equations (White 1974). Continuity for incompressible flow (wind flow is considered to be incompressible due to relatively small wind speeds) is given by:

$$\frac{\partial u(t)_j}{\partial x_i} = 0 \tag{2.3}$$

Conservation of momentum, expressed by the so-called Navier-Stokes equations, is given by

$$\frac{\partial u(t)}{\partial t} + u(t)_j \frac{\partial u(t)_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u(t)_i}{\partial x_j} + \frac{\partial u(t)_j}{\partial x_i}\right)\right] \tag{2.4}$$

Wind flow fluctuates highly with time. For numerical simulation the fluctuating part is usually separated from the mean part, i.e. u(t) = U+u' where U and u' are the mean and the fluctuating velocities respectively. Replacing u(t) in equation (2.4), by U+u' and assuming steady flow provides the so-called Reynolds-Averaged Navier-Stokes (RANS) equations (2.5). RANS has been used in the present study.

$$U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \overline{u_i' u_j'}\right] \tag{2.5}$$

Thus it is possible to express the momentum equations using mean flow values; however this averaging has created a new parameter that is the last term in equation (2.5), which requires extra closure equations. Fluid mechanics specialists such as Boussinesq (1877) prefer to express this new term in a similar fashion with dynamic viscosity times the

gradient term, i.e. the last term in equation (2.4). Therefore, the product of turbulent viscosity and the velocity gradient replaces the new term as follows:

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3}\delta_{ij} k \tag{2.6}$$

where ν_t termed "eddy viscosity" is given by

$$\nu_t = C_\mu \frac{k^2}{\varepsilon} \tag{2.7}$$

Substituting equation (2.6) in (2.5) gives

$$U_j \frac{\partial U_i}{\partial x_j} = -\frac{\partial P}{\partial x_j} + \frac{\partial}{\partial x_j}\left((\nu + \nu_t) \frac{\partial U_i}{\partial x_j} \right) + \frac{\partial \nu_t}{\partial x_j} \frac{\partial U_j}{\partial x_i} \tag{2.8}$$

where P is the augmented pressure field. In order to have fewer terms in equation (2.8), the k and p terms are augmented in the following manner

$$P = \frac{p}{\rho} + \frac{2}{3} k \tag{2.9}$$

The turbulent (eddy) viscosity is purely flow property whereas dynamic viscosity is fluid property. The extra equations required to determine the turbulent viscosity constitute the so called turbulence models. There are different kinds of turbulent models such as k-ε, Large Eddy Simulation (LES) etc. The most widely used and appropriate for the case of homogeneous isotropic turbulence is the k-ε turbulence model. It has two turbulence transport equations, one for the turbulent kinetic energy (k) and one for the dissipation rate (ε), given by equations (2.10) and (2.11) respectively. Model constants C_1, C_2, C_μ, σ_k, and σ_ε are equal to 1.44, 1.92, 0.09, 1.0 and 1.3 respectively (Launder and Spalding 1974).

$$U_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + \nu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\frac{\partial U_i}{\partial x_j} - \varepsilon \tag{2.10}$$

$$U_j \frac{\partial \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + C_1 \frac{\varepsilon}{k}\nu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\frac{\partial U_i}{\partial x_j} - C_2\varepsilon^2 \tag{2.11}$$

Most common terrain geometries upon which wind flow has been simulated include: two-dimensional single and multiple hills (hills with very large width compared to their length, L); three-dimensional hills (hills with cone type shapes); two-dimensional valleys (valleys with very large width compared to their length, L); and two-dimensional escarpments (escarpments with very large width compared to their length, L).

2.4 Numerical evaluation

This section reviews the state-of-the-art in the numerical evaluation of wind flow over different types of topographies. Numerical simulations differing from one another by the type of numerical formulation followed, the turbulence model used, the type of boundary conditions applied, the type of grids adopted and the type of terrain considered are summarized. Comparison has been made between existing numerical and experimental (both BLWT and field) works establishing wind flow modifications over hills, escarpments, valleys and other complex terrain configurations. Software capable of extracting graph values from scanned figures has been used to establish comparisons among values found in literature. As a result most of the figures are redrawn and reformatted by the author in order to highlight a particular advantage or disadvantage of the numerical model under consideration. Comparisons are also made with provisions from the current wind standards and codes of practice with the intention of clarifying the

scope and applicability of requirements. Whenever applicable, comparisons are also made with speed-up values calculated using guidelines derived from theoretical models.

2.4.1 Two-dimensional single hill and escarpments

The energy shortage since the 70's in Europe has forced engineers to re-consider the potential of wind energy in more depth. In response to this demand Jackson and Hunt (1975) developed an analytical model that describes wind flow over a single hill. Their model used an analytical perturbation theory valid for low hills without flow reversal hence their predictions cannot extend to steeper terrain where separation has large effect on the flow. However despite its limitations, their model shed light on the structure of the interaction of the viscous layer near the hill with the inviscid region and inspired researchers such as Mason and Sykes (1979), Taylor et al. (1983), Mason and King (1985), Beljaars et al. (1987) and Karpik (1988) to work in that direction. Inevitably, the underlying assumptions of the theoretical models have limited their applicability to non-separating flow over gentle-sloped low hills.

Numerical solution of strongly non-linear governing equations has much greater potential than analytical methods that may not be applicable to non-linear problems of this nature. As a result, a number of research projects have been carried out in this field. Deaves (1980) simulated flow over a single hill numerically with a turbulence closure based on Prandtl mixing length theory, strictly applicable to hills with no separation. Other researchers whose work is considered in this group have used one of the variants of k-ε turbulence model. Bergeles (1985), Paterson and Holmes (1993), Kim et al. (1997) and Carpenter and Locke (1999) used the standard k-ε model. Quinn et al. (1998) used three different variants of k-ε models (standard, Mochida and Murakami, ReNormalization

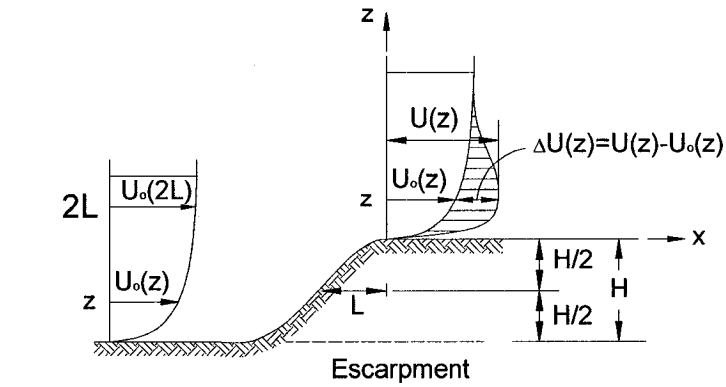
Group - RNG). Regarding the type of grid most researchers used body fitted coordinates that are fitted to the contour of the ground surface. The body fitted coordinates used by Bergeles (1985) and Kim et al. (1997) were orthogonal and generated by solving elliptical-type differential equations. The use of body-fitted coordinates was shown to be advantageous for accurate ground boundary application since the grid fits the ground contour exactly, which is clearly better than staircase type of ground surface approximation applied with Cartesian coordinates.

Field measurement results and experimental data from BLWT simulations have been compared with Jackson's analytical prediction (1979), Paterson and Holmes (1993) and Quinn et al. (1998) numerical evaluations and provisions of the National Building Code of Canada (NBCC 1995) for a number of escarpment geometries and roughnesses. It should be noted that these code provisions have also been used by the American Wind Loading Standard ASCE 7 in all its recent editions as well (ASCE 2002). Quinn et al. (1998) results based on the RNG k-ε were selected and used for the present comparison due to slightly better agreement with field data when compared with other k-ε models used in their study. Figure 2.2 shows typical comparative results for the variation of velocity profiles above an escarpment. Data has been gathered from literature but recalculated and reformatted in order to fit in the same graph for comparison purposes. Similarly CFD simulation of wind flow over single hill by Jackson (1979), Bergeles (1985), Paterson and Holmes (1993), Kim et al. (1997) and Carpenter and Locke (1999) have been compared to field and BLWT data as well as NBCC provisions, as shown Figure 2.3. For different geometries of escarpments (H/L=0.2 to 1.2) and hills (H/L = 0.2 to 0.6), as well as different values of upstream roughness expressed by Jensen's number

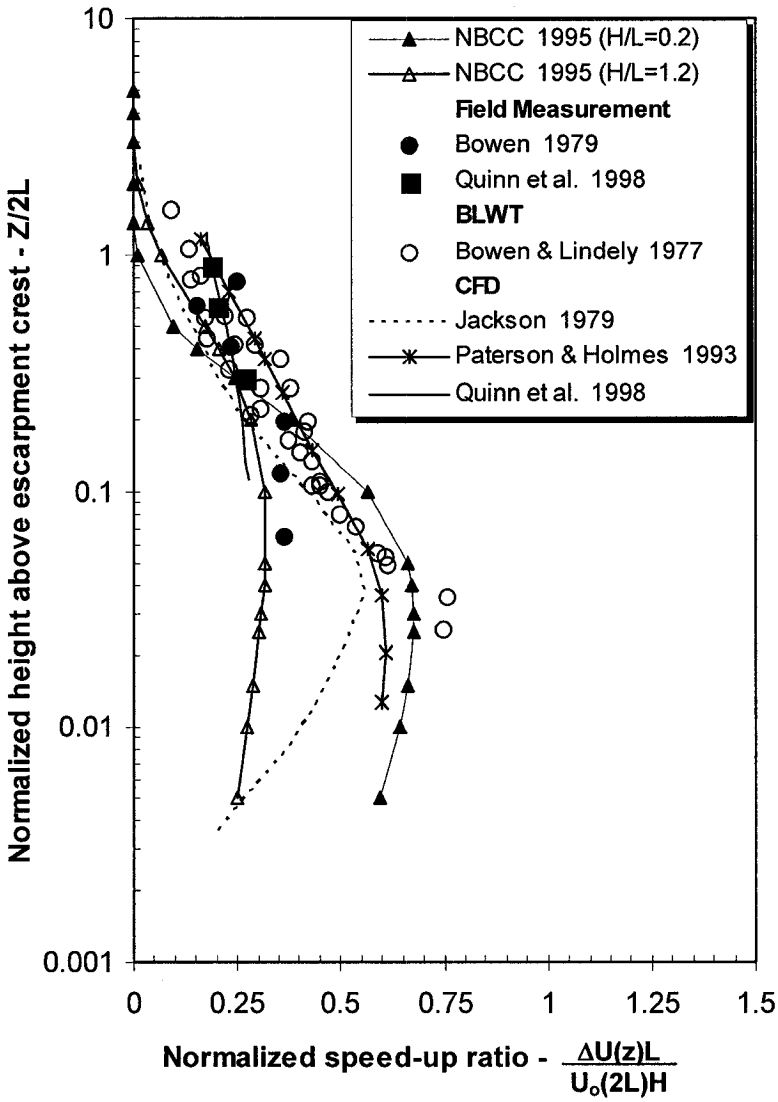
(H/z_o) ranging from 400 to 40,000, normalized speed-up ratio values given by $\Delta U(z)L/U_o(2L)H$ for escarpments and $\Delta U(z)L/U_o(L)H$ for hills are quite similar. In these expressions H represents the height of the hill/escarpment, L represents the horizontal distance from the crest to where the ground elevation is half the height of the hill/escarpment, z_o represents roughness of the ground, $\Delta U(z)$ represents increase in velocity i.e. $U(z)-U_o(z)$ at height z above the local escarpment/hill surface, $U_o(2L)$ and $U_o(L)$ represent upstream reference velocity at the height 2L and L above the ground respectively; similarly $U_o(z)$ and U(z) represent upstream reference velocity and mean velocity at the local hill surface at height z above the ground respectively. Normalized speed-up ratio values based on NBCC (1995) in Figures 2.2 and 2.3 have been calculated for the steepest and the shallowest hill/escarpment in the group of isolated hills/escarpments considered.

When significantly different hill geometries are considered separately, the agreement between CFD, BLWT, analytical and NBCC Fractional Speed-Up Ratio (FSUR) values expressed by the ratio $U(z)/U_o(z)$ is better for shallow hills than for steep hills as can be seen in Figure 2.4. The sizable discrepancy among values near the steep hilltop as shown in Figure 2.4 (b) is attributed to the presence of flow separation and recirculation behind steep hills for which the isotropic turbulence models used in most of the studies may not be suitable.

Similarly, significant difference in roughness of the ground surface affects the wind flow pattern over hills considerably. This was illustrated by the studies of Kobayashi et al. (1994) regarding airflow above forested and non-forested two-dimensional hills as well



(a)



(b)

Figure 2.2 Comparison between previous works for wind velocities above the crest of an escarpment (a) parameters definition, and (b) normalized speed-up ratios.

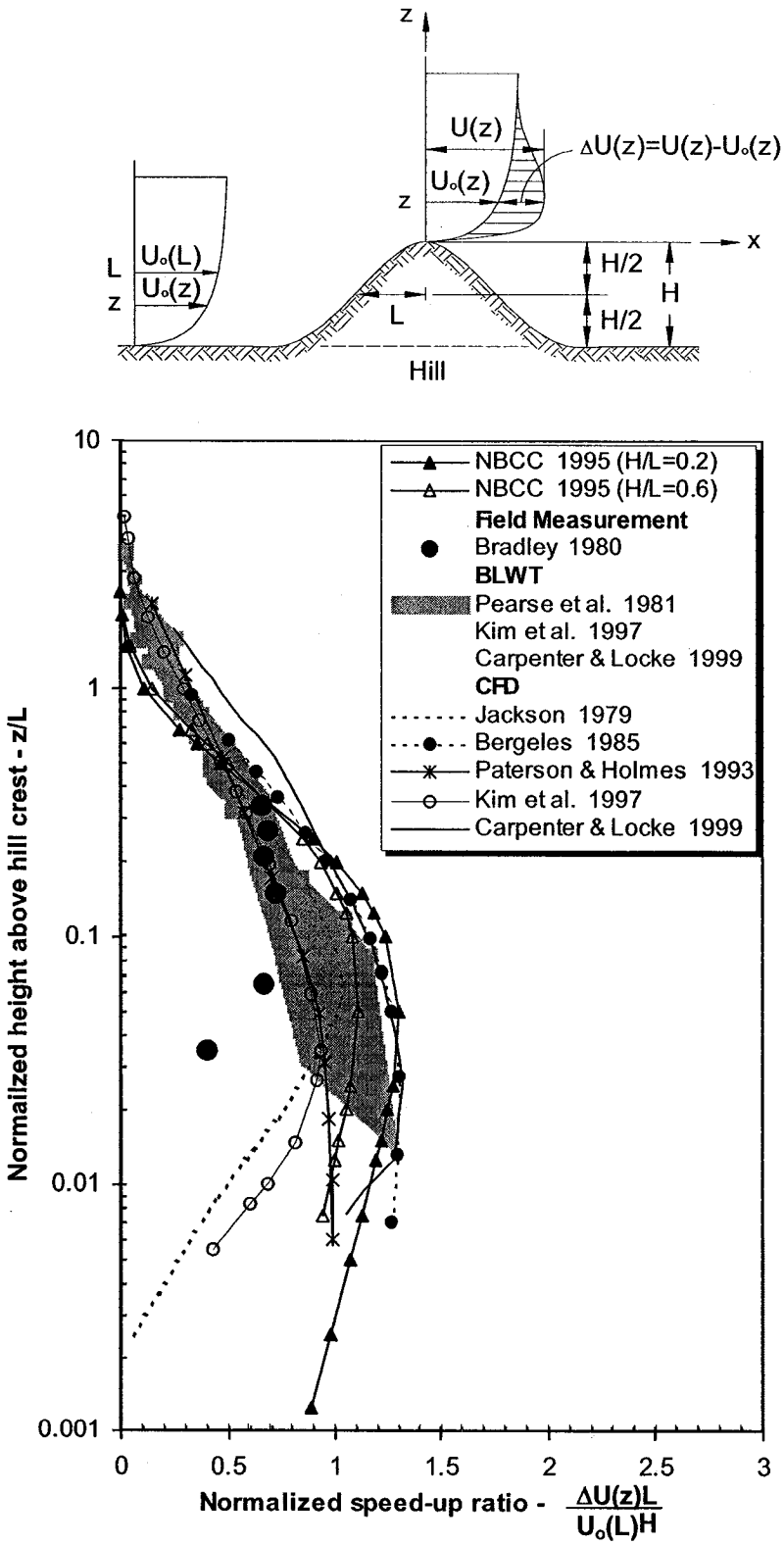
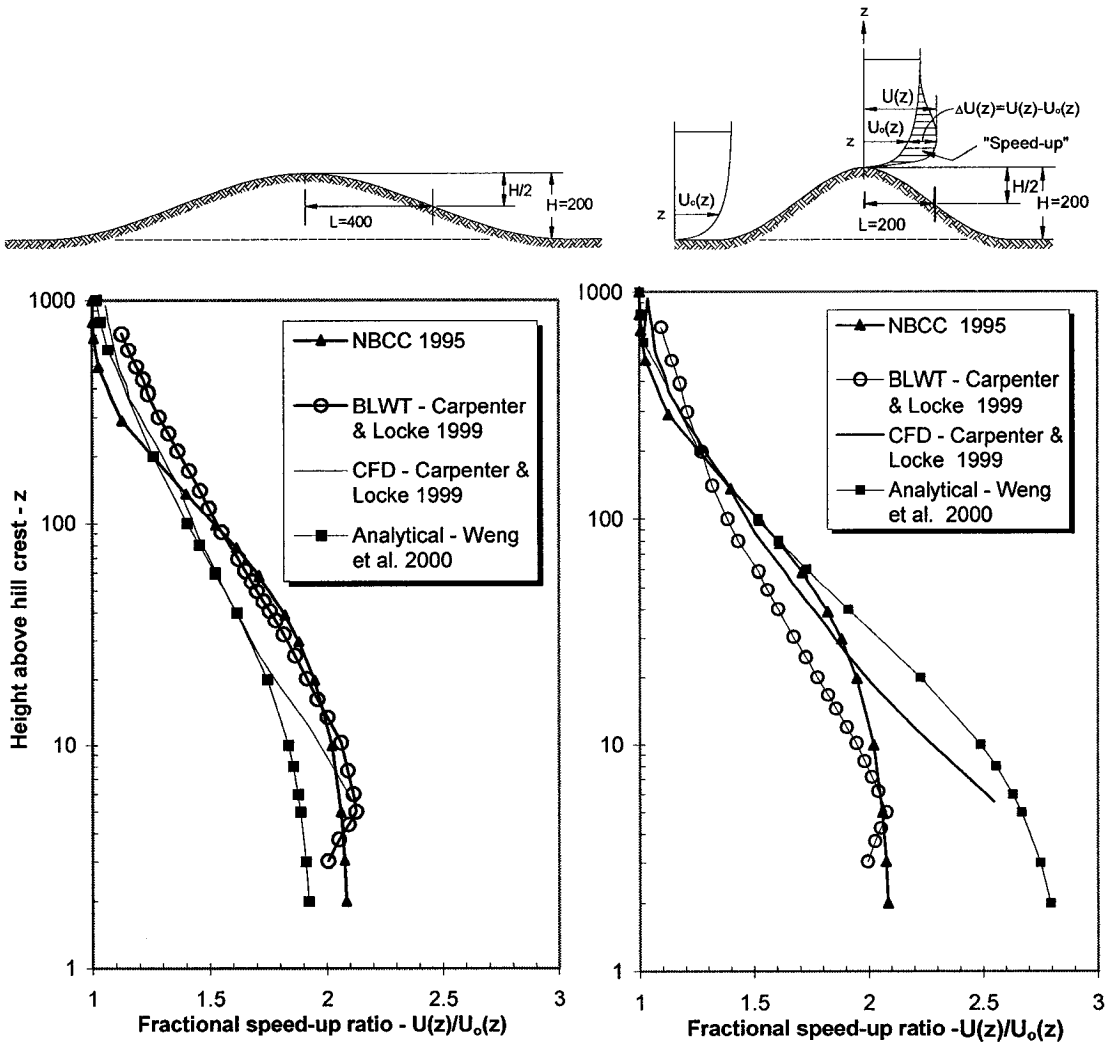


Figure 2.3 Comparison between previous works for wind velocities above the crest of a single hill (a) parameters definition, (b) normalized speed-up ratios.

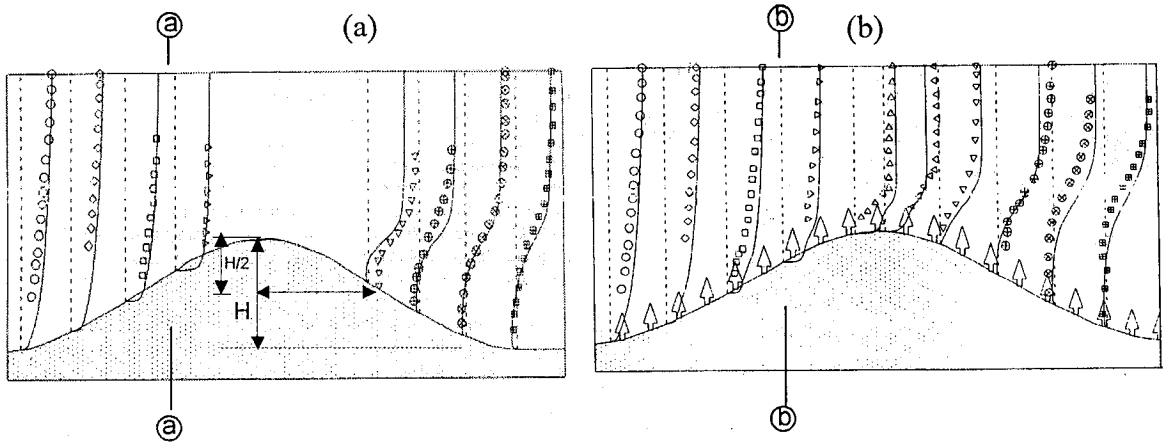


(a) (b)

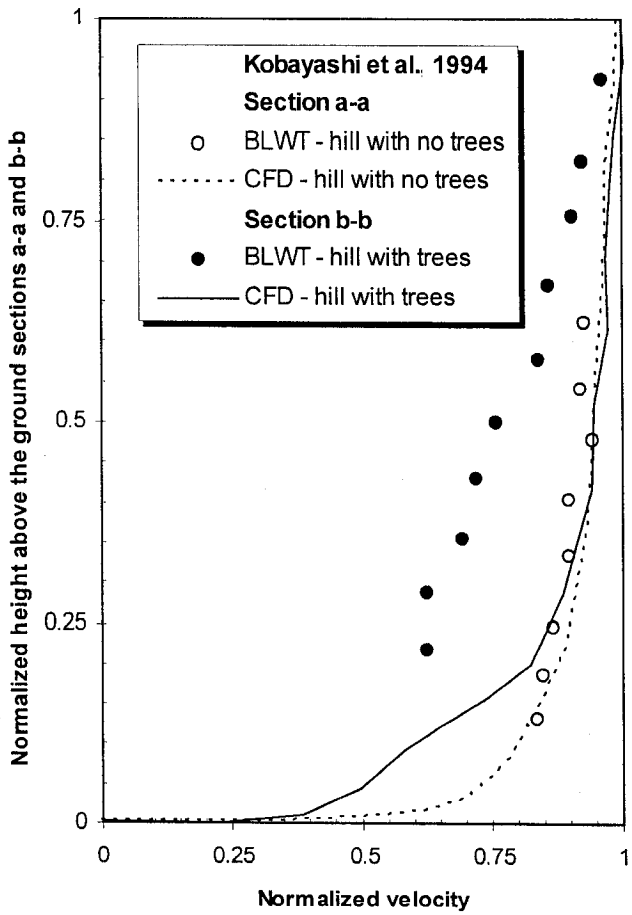
Figure 2.4 Comparison between BLWT, CFD, Analytical and NBCC fractional speed–up ratios at the crest of (a) shallow, and (b) steep sinusoidal hill (Reynolds number $Re(H)=1.3x10^5$, roughness $z_0=0.02m$). Scale 1:1000. All dimensions are in [mm].

as the research of Lun et al. (2003) for flow around smooth and rough two-dimensional hills. Kobayashi et al. (1994) take the effect of vegetation cover on airflow and turbulence into account by considering additional drag forces in the RANS equations and the k-ε model respectively. In the study of Lun et al. (2003) the surface roughness was incorporated through the application of a wall function that contains z_o at the ground boundary; three turbulence models were used: standard, revised k-ε model by Durbin (1996) and Reynolds stress algebraic equation model by Shih et al. (1995). Comparisons between numerical simulations and BLWT measurements for the mean flow characteristics are shown in Figure 2.5 for two cases, without trees and with trees (tree height = 50mm), on a steep hill (H/L=1 and H=200 mm). The same comparisons have been enlarged by the present author at sections a-a, and b-b for hill with trees and without trees respectively as shown in Figure 2.5 (c). Since data at the crest of the hill with trees is not given, the comparison could not be done at the crest. Numerical simulations agree with the experimental measurements satisfactorily, as shown in Figures 2.5 (a) and (b).

The agreement between velocity profiles obtained by CFD and BLWT experiments is better for the hill with no trees than the hill with trees, as shown in Figure 2.5 (c), possibly indicating the requirement of fine-tuning the roughness parameter application in the numerical models. The separated flow on the leeward side of the hill for the non-forested case has a profile much thinner than for the forested case, leaving more momentum available to overcome the adverse pressure gradient; consequently separation occurs further down the crest as shown in Figure 2.6 (a). Lun et al. (2003) have also obtained similar results by using the nonlinear turbulence model of Shih et al. (1995)



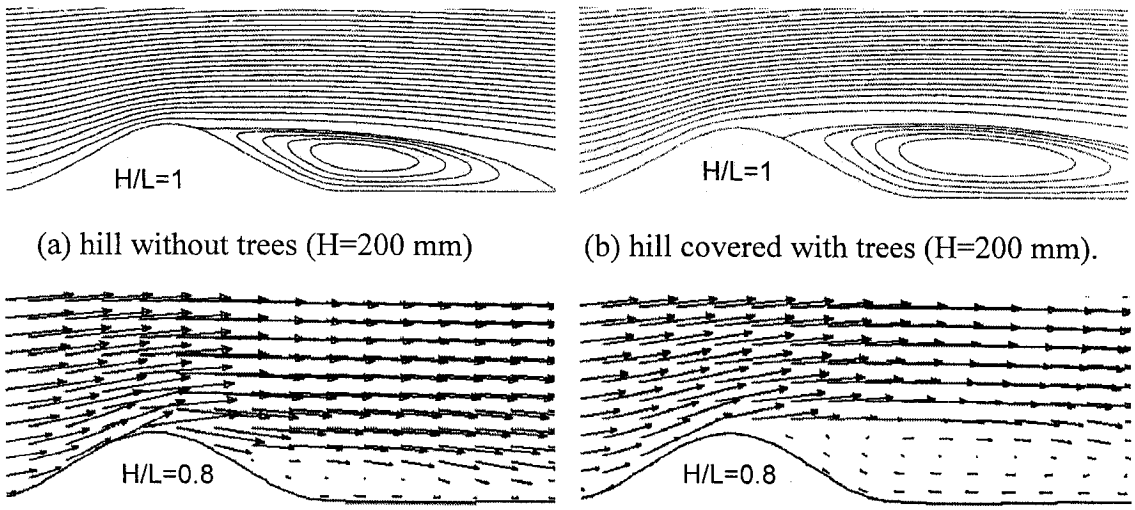
Note: Symbols O◇□▷▽⊕ ⊗△◁ show BLWT result and the solid lines show CFD results



(c)

Figure 2.5 Numerical and experimental velocity profiles at different horizontal positions for a hill (a) with no trees (b) with trees (after Kobayashi et al. 1994) and (c) with trees and with no trees at sections a-a and b-b respectively. (H/L=1, H=200 mm).

as shown in Figure 2.6 (c) and (d). Overall numerical simulations based on the k-ε model appear very good when the differences between wind tunnel and field data are taken into account. It is worth noting that the fractional speed-up ratios derived from the NBCC (1995) compare well with the computational results. The guidelines (Weng et al. 2000) used in the comparison shown in Figure 2.4 were derived from the Multi Spectral Finite Difference (MSFD) model of Beljaars et al. (1987) and its nonlinear extension NLMSFD by Xu et al. (1994). These are theoretical formulations similar to that of Jackson and Hunt (1975), in which the equations of motion are linearized by expressing solution variables as being equal to an undisturbed part and a small perturbation. Fourier Transform techniques are used on the resulting equations and allow a solution to be obtained by numerical means. Provision of simple guidelines that can convey the result of complex numerical simulation to the end user by using a simple algebraic equation is an attractive approach. Lemelin et al. (1988) have also used very simple exponential



(a) hill without trees (H=200 mm) (b) hill covered with trees (H=200 mm).

(c) smooth hill (z_0=0.01mm, H=40 mm) (d) rough hill (z_0=0.3mm, H=40 mm)

Figure 2.6. Numerically obtained streamlines (a) without trees and (b) with trees (tree height=50 mm, after Kobayashi et al. 1994). Velocity field over (c) smooth hill, and (d) rough hill (after Lun et al. 2003)

equations to represent results obtained through complex theoretical calculations. The same approach has been used by NBCC (1995).

2.4.2 Two-dimensional multiple hills

Carpenter and Locke (1999) investigated the variation of wind speeds over steep and shallow multiple two-dimensional sinusoidal hills. The hills have uniform geometry as shown in Figure 2.7. Computational results were obtained by using the finite element approach with the standard high Reynolds number k-ε turbulence model.

Fractional speed-up ratios for hill geometries used by Carpenter and Locke (1999) have also been calculated based on guidelines by Weng et al. (2000) and NBCC provisions (1995) for the present study. A typical comparison of fractional speed-up ratios (i.e. $U(z)/U_o(z)$) among CFD, BLWT, calculated values using Weng et al.'s (2000) guidelines and NBCC provisions, is shown in Figures 2.8 and 2.9. The agreement between CFD and BLWT results is slightly better for shallow hills than for steep hills. In general, the agreement between CFD and BLWT results is somewhat better for downstream hills than upstream hills especially 40m above the crest. The BLWT experiments indicated a decreasing pattern of fractional speed-up ratios for the downstream hills when compared with the upstream hill. It is interesting to observe that similar pattern is captured by the CFD simulations as shown in Figures 2.8 and 2.9. NBCC values for the isolated hill cases agree with both BLWT and CFD predictions as discussed in the previous section; however NBCC fractional speed-up ratios are found to be conservative for use in most cases of multiple hills. In particular NBCC, if used for triple hills, over predicted the fractional speed-up ratios by 47% and 43% for hill-2 and hill-3 respectively at 40 m above the crest (Figure 2.9).

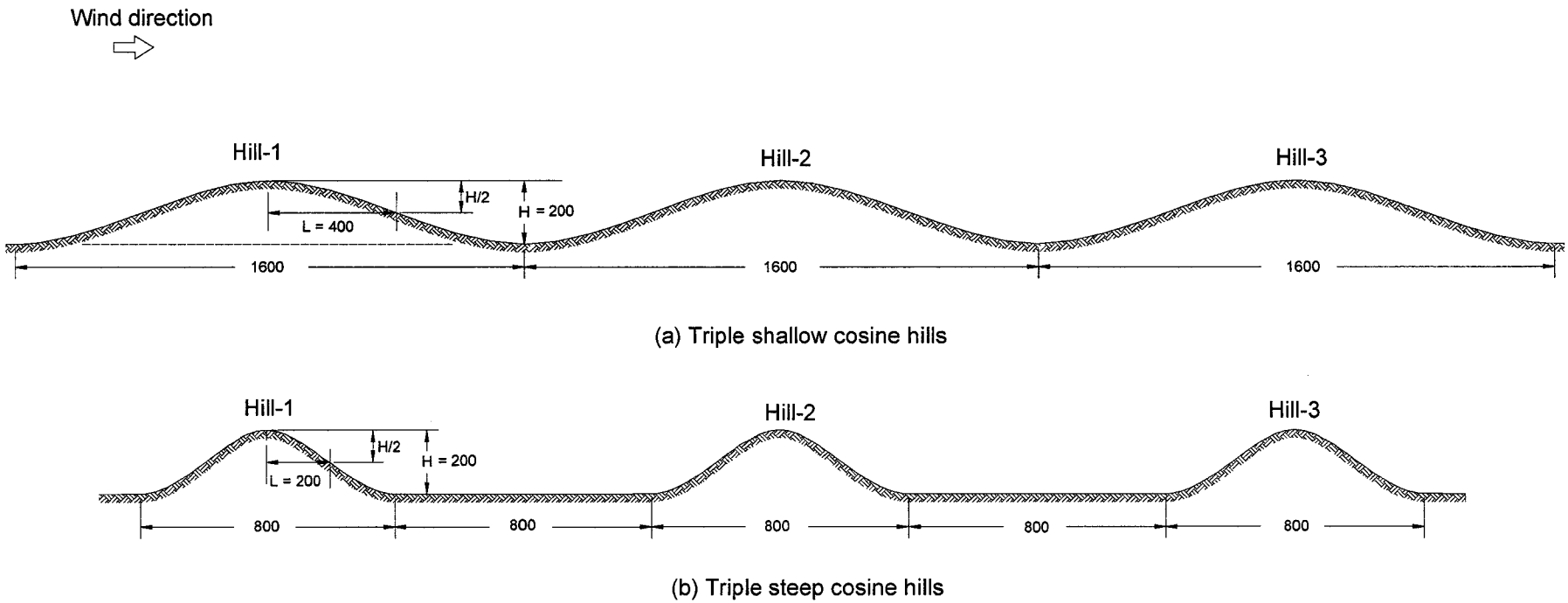


Figure 2.7 Geometry of shallow and steep sinusoidal hills used in Carpenter and Locke (1999). Scale 1:1000. All dimensions are in [mm]

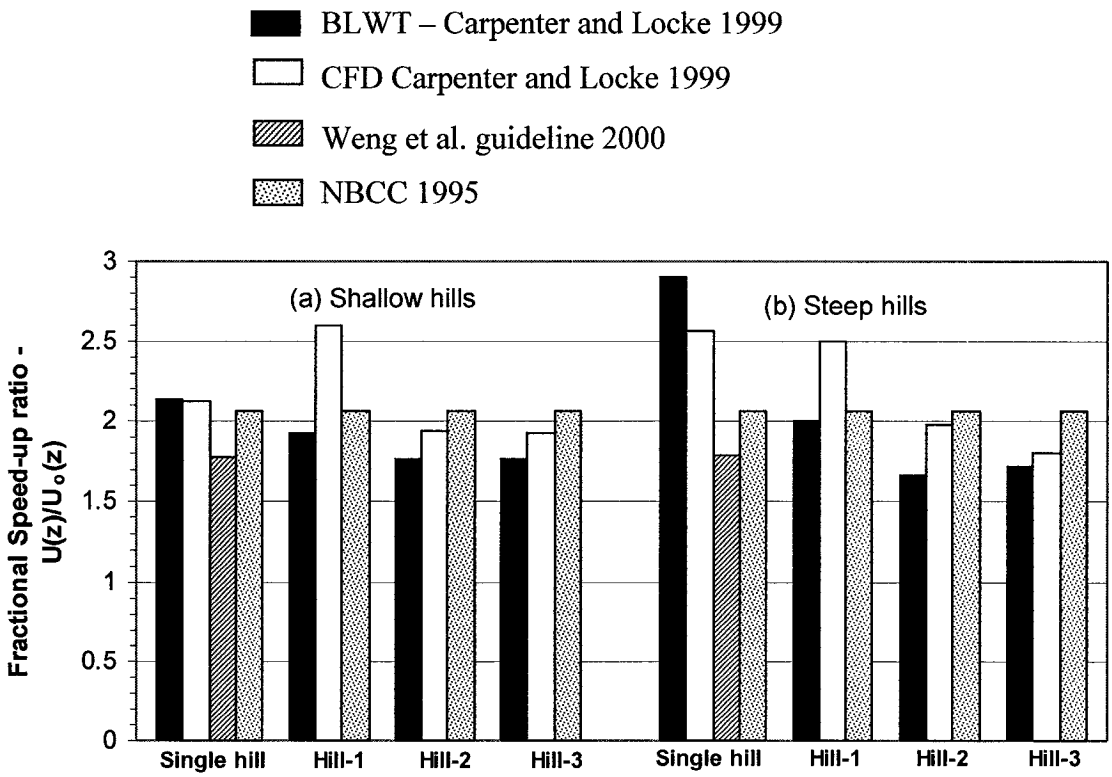


Figure 2.8 Comparison between fractional speed-up ratio values obtained from BLWT, CFD, NBCC and analytical model at 5m above the crest of the multiple (a) shallow, and (b) steep hills. Values from a single hill are also given.

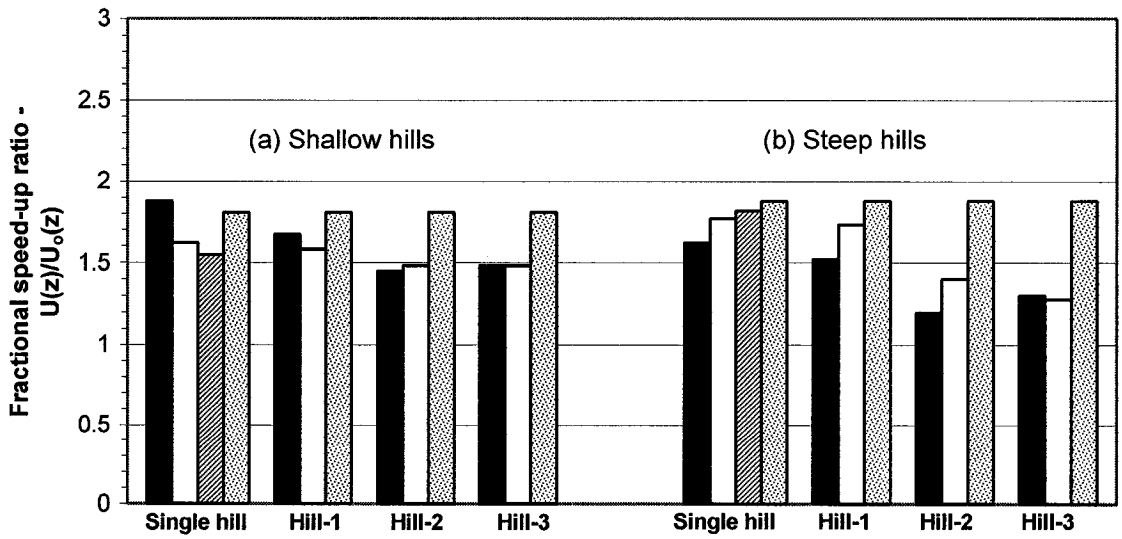
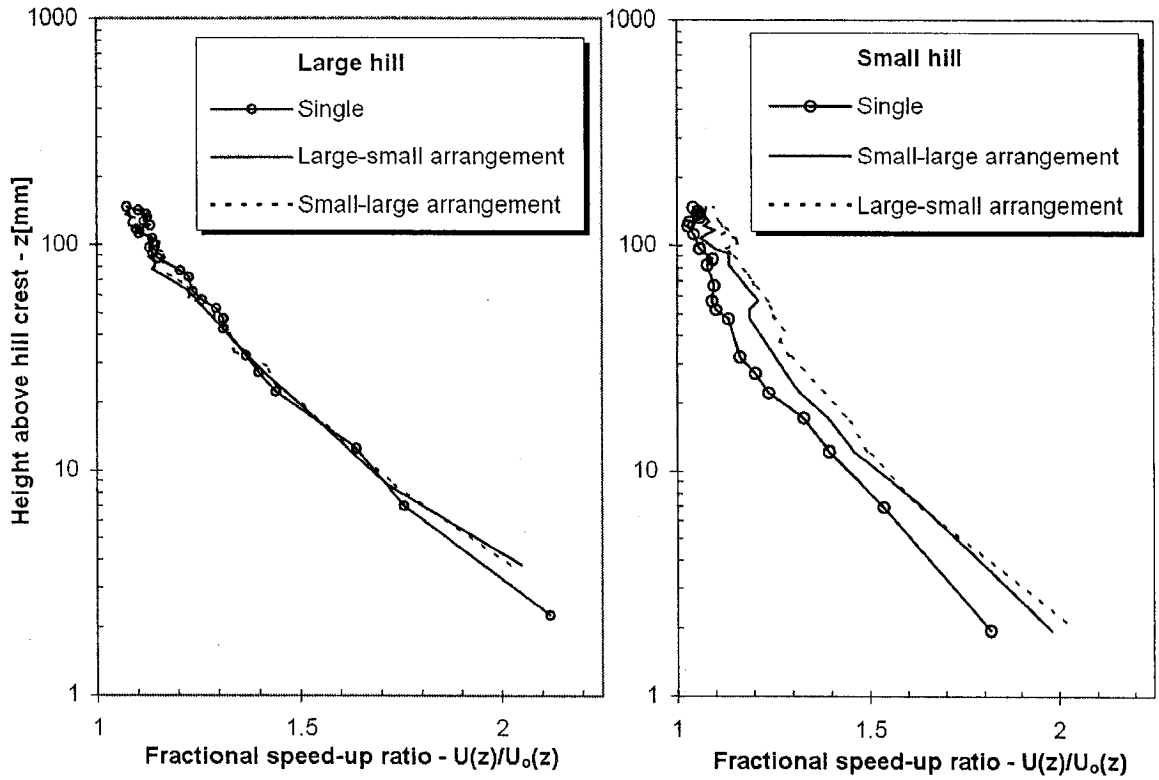


Figure 2.9 Comparison between fractional speed-up ratio values obtained from BLWT, CFD, NBCC and analytical model at 40m above the crest of the multiple (a) shallow, and (b) steep hills. Values from a single hill are also given.

The relative size of neighboring hills also plays a significant effect on fractional speed-up ratio values. For example, Kim et al. (1997) studied this effect on the mean velocity profile at the top of a small hill caused by a neighboring higher hill located at the same horizontal distance either upwind or downwind of the small hill. Kim et al. (1997) solved the RANS equations using body fitted coordinates and the standard k-ε turbulence model closure. Using Kim et al.'s (1997) velocity profile, fractional speed-up ratio values have been calculated at the crest of small (H/L=0.6, H=40 mm) and large (H/L=0.6, H=70 mm) hills for four different cases: single small hill, single large hill, upstream small hill-downstream large hill (small-large), and upstream large hill-downstream small hill (large-small). Figure 2.10 shows fractional speed-up ratio values that have been calculated based on Kim et al. (1997) results for mean velocity distributions at the hilltop of the small hill and of the large hill. Clearly the hill top fractional speed-up ratio profiles of the large hill are not much influenced by the neighboring small hill as shown in Figure 2.10 (a), whereas the lower hill is somewhat influenced by the neighboring higher hill as shown in Figure 2.10 (b). It is interesting to note that the fractional speed-up ratio on the small hill decreases by about 5% to 10% in the presence of the large hill compared with that of an isolated case. The influence of a neighboring hill has also been observed in related experimental studies by Ferreira et al. (1991), Miller and Davenport (1998), and Carpenter and Locke (1999).



(a) values on the large hill (b) values on the small hill

Figure 2.10 Comparison of the fractional speed-up ratio values obtained from CFD models at the hilltops of (a) large hill: H/L=0.6 and H=70 mm, (b) small hill: H/L=0.6 and H=40 mm.

2.4.3 Two-dimensional valleys

Maurizi (2000) presented useful computational results dealing with two-dimensional valleys of various slopes and compared them with the wind tunnel experimental data from Busuoli et al. (1993). Out of the three different k-ε models utilized - the standard k-ε model, the one described by Chen and Kim (1987), and the Re-Normalization Group (RNG) version (Yakhot et al. 1992) - RNG was reported to give better results, hence this is used to calculate speed-up ratio values in the present comparisons. Speed-up ratio values have been calculated from computational and experimental velocity profile data taken from Maurizi (2000) and Busuoli et al. (1993) respectively. Further speed-up ratio

values have been calculated for valley geometries used by Maurizi (2000) applying the guidelines of Weng et al. (2000) and the NBCC provisions (1995). Comparisons of speed-up ratio values have been made for both shallow (H/L=0.125) and deep (H/L=0.333) valleys at four different longitudinal locations. Figure 2.11 shows good speed-up ratio agreement between all numerical results and the BLWT data at four longitudinal locations in the case of a gently sloping shallow valley. The NBCC provisions for the case of two-dimensional valleys seem to agree well with other values for the shallow valley except at the downstream edge of the valley (section-d) as shown in Figure 2.11 (d). The comparisons of NBCC values with numerical and experimental data become less satisfactory for the deeper valley as shown in Figures 2.12 (b) and (d), since flow recirculation occurs. Particularly poor agreement with the NBCC is observed at the downstream edge of the deep valley (section d) as shown in Figure 2.12 (d). Values obtained using the guidelines of Weng et al. (2000) are also compared at the center of the valley (section-b), where the Weng et al. (2000) model is only applicable. Although the guideline values of Weng et al. (2000) are in good agreement with CFD, BLWT and NBCC for shallow valley as shown in Figure 2.11 (b), they overestimate speed-up ratio values for deeper valley as shown in Figure 2.12 (b). The overestimation may be caused by the underlying theoretical model's inability to simulate separating flow.

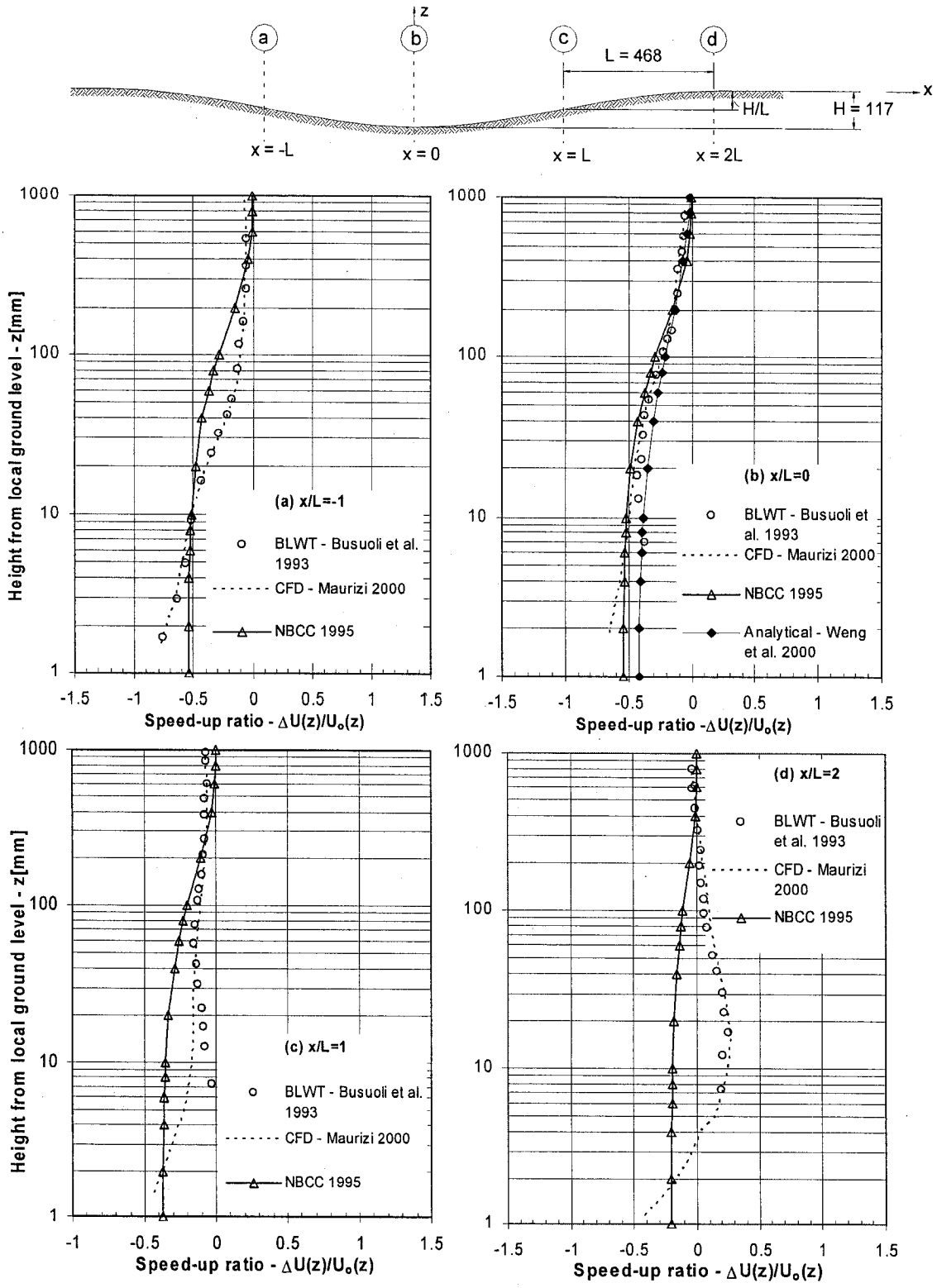


Figure 2.11 Vertical profiles of horizontal mean velocity for shallow valley with H/L=0.25 at our longitudinal locations. The axis origin x=0 is in the valley center while z=0 is on the local ground surface. All dimensions are in [mm].

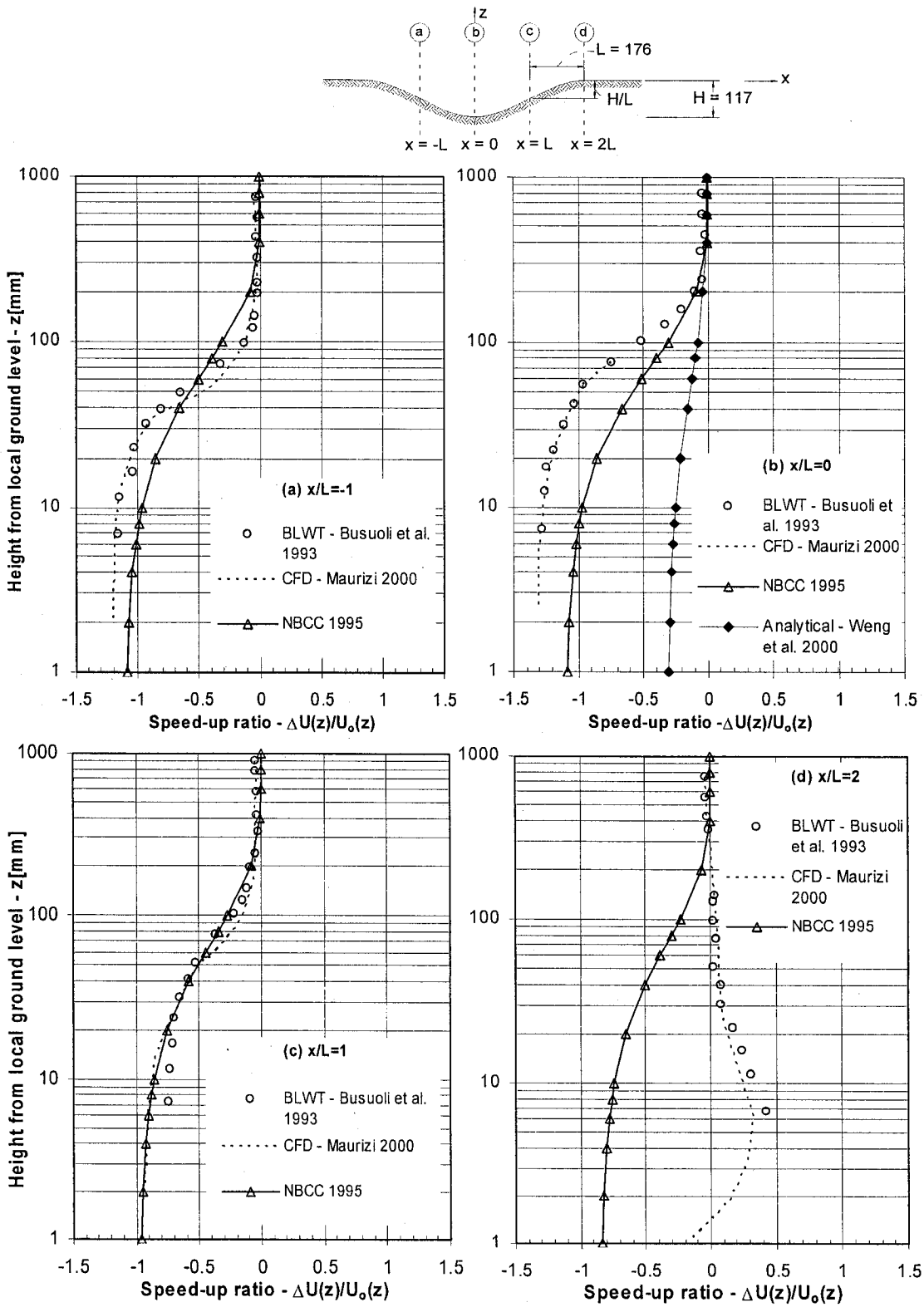


Figure 2.12 Vertical profiles of horizontal mean velocity for deep valley with H/L=0.666 at four longitudinal locations. The axis origin x=0 is in the valley center while z=0 is on the local ground surface. All dimensions are in [mm].

2.4.4 Three-dimensional complex terrain

Kim et al. (2000) simulated wind flow over the Askervein Hill in Scotland by solving the RANS equations using body-fitted coordinates. Standard and RNG-based k-ε models are used together with wall functions to account for surface roughness. The field measurements were available along lines A-A and B-B at a height of 10 m above the hill surface as shown in the contour map of Askervein hill in Figure 2.13.

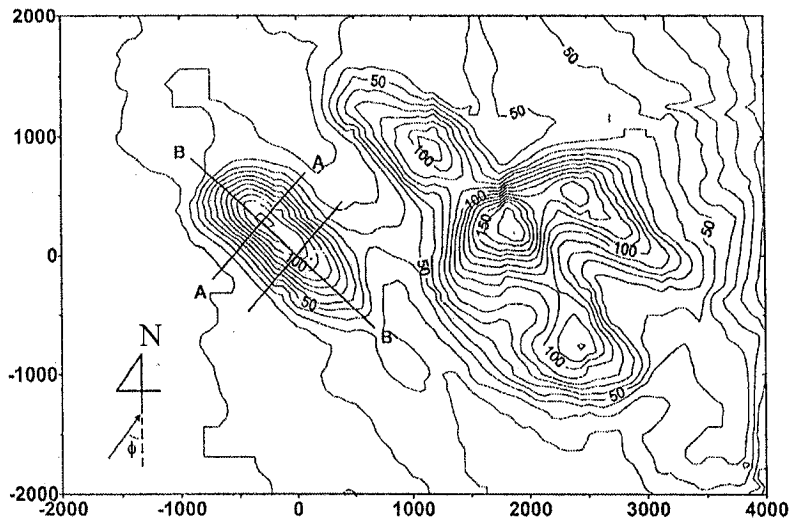
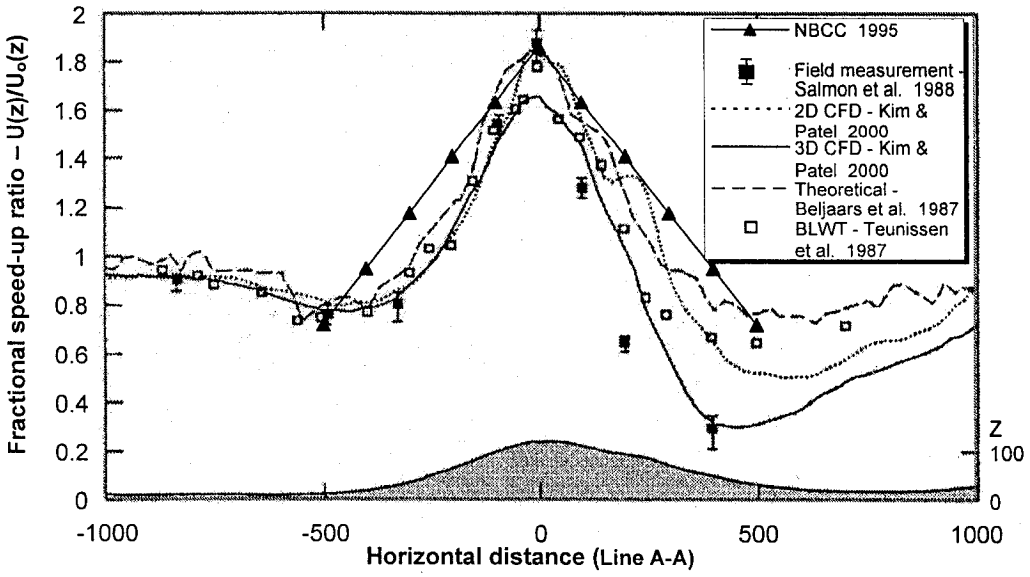


Figure 2.13. Topographic map of Askervein hill and surrounding area. All dimensions are in [m] (after Kim et al. 2000).

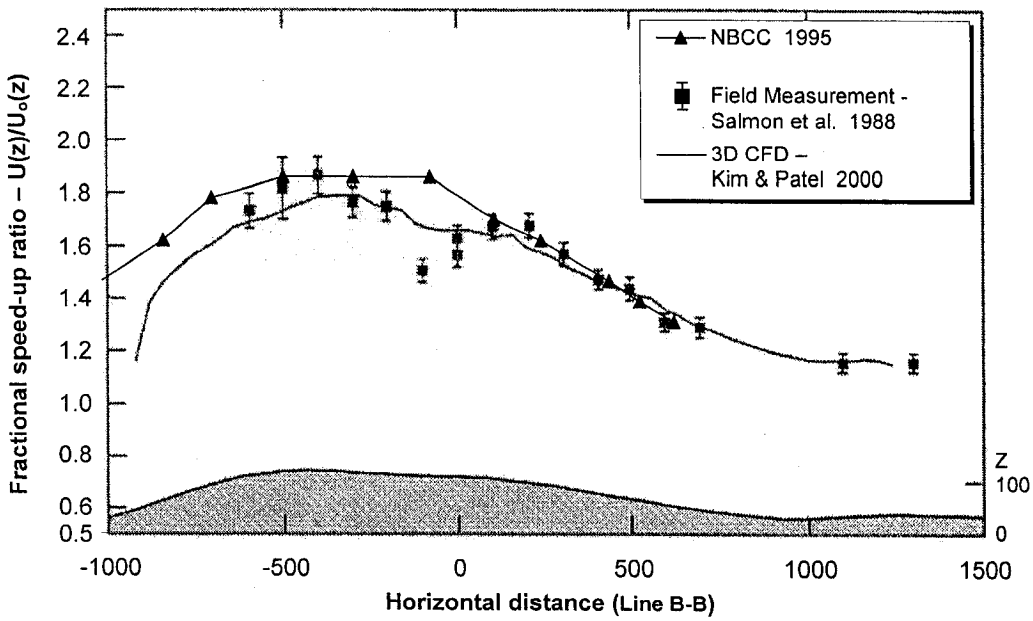
Following this work, fractional speed-up ratio values have been generated for Askervein hill using NBCC provisions for the same positions. Note that NBCC provisions are meant for either two-dimensional hills or axisymmetric three-dimensional hills and not for a complex three-dimensional terrain. Since section A-A is approximately symmetrical and aligned along the wind direction, fractional speed-up ratio values have been calculated using NBCC provisions for two-dimensional hills. In case of section B-B however, this type of approximation cannot be used since the section is not aligned in the wind direction; instead a number of two-dimensional sections along the wind direction and

perpendicular to section B-B have been considered systematically. Fractional speed-up ratio values using NBCC have then been calculated at the points where these two dimensional cross-sections intersect section B-B and used in the subsequent comparisons.

Figures 2.14 (a) and (b) show comparisons made among the BLWT experiments, full-scale measurements, two- and three-dimensional numerical computations, and NBCC provisions. There is substantial agreement among all the results on the windward slope of the hill along line A-A, but significant differences are found on the lee side where the full-scale measurements show rapid decrease of fractional speed-up ratio down to 0.3. Kim et al. (2000) conjectured that this trend arises from the three-dimensional flow separation on the lee-slope due to the influence of neighboring hills. They also reported that flow separation on the lee side of Askervein hill was not predicted by the three-dimensional computation when the neighboring hills were excluded. In that case the fractional speed-up ratio distributions were quite similar to those predicted by the two-dimensional computation. The wind-tunnel experiment, which employed a smooth surface model, failed also to reproduce the decrease of fractional speed-up ratios on the lee slope. Only the three-dimensional numerical simulation predicts this behavior although there is some deterioration of agreement with data at the peak. For section A-A, the agreement between NBCC provisions, field measurement, and three-dimensional CFD values is relatively better in windward than leeward direction. For the case of section B-B, NBCC predictions represent the experimental data well, as shown in Figure 2.14 (b). Often comparisons are made between numerical results and BLWT measurements. To put these comparisons into the right perspective one should also further consider the differences that exist between field and BLWT measurements.



(a)



(b)

Figure 2.14 Fractional speed-up ratios for Askervein hill at 10 m above the ground level for wind angle ϕ=30° at (a) line A-A, and (b) line B-B. All dimensions are [m].

2.4.5 Closure

For single two-dimensional hills, general agreement between different numerical speed-up ratio predictions and experimental data (including the commentaries of NBCC) has been observed. The agreement is better for a single shallow hill compared to a single steep hill. There are no provisions for multiple hill cases. The use of speed-up ratios based on studies for single hill to multiple hill cases from codes and standards is clearly a conservative approach with generally over-predicted speed-up ratios. In the comparisons made by the author, NBCC over predicted the speed-up ratios by more than 40% (see section 2.1.4), which can translate into over 90% increase in wind load. This indeed is very conservative. The economical impact of accurate wind load predictions has been studied by Horesfield et al. (2002). It was shown that with the use of more accurate wind loads, 19% of the concrete lateral load-resisting system (around 6500 m^3) and 2.1% of usable floor area had been saved. Therefore there is a strong need for developing guidelines that work for a wide spectrum of terrain geometries and roughness conditions. This will be one of the goals of the present study.

At present, utilization of numerical simulations to predict wind speed-up in practical applications is rather limited and designers still have to rely on physical simulations for complex terrain situations. This is mainly due to the unavailability of specialized and cost-effective numerical tools targeted to give solutions for wind flow over different topographies.

Another problem is the complexity generally associated with CFD-based numerical tools. Normally these require background knowledge, fast hardware with large memories, and availability of ample computational time. Thus in parallel to developing such numerical

tools, simple ways of conveying the results produced by complex CFD simulations to the engineering profession (end-user) must be devised. To this effect, the author has proposed and developed a neural network model trained with CFD-generated data, which is capable of producing speed-up ratio values following input of simple geometrical parameters describing the topography and roughness of the ground. The details are discussed in chapter 5.

2.5 Neural network

Artificial Neural Networks (ANN or NN for short) that have proven of tackling many engineering problems (Flood and Kartam 1994 I and II, Mathew et al. 1999, Shigidi and Garcia 2003, Tsai and Hsu 2002 etc.) operate analogously to biological nervous systems. NN have the ability to imitate the brain in making decisions and drawing conclusions when presented with representative data and they are tolerant to noise and/or partial information. NN representations are capable of developing functional relationships from discrete values of input-output quantities obtained from experimental results or complex computational approaches. This generalization property makes it possible to train a network on a representative set of input-output examples and get good results for new input without further training the network. NN are considered as powerful computational tools.

2.5.1 Basics of NN

The NN approach adopts the brain metaphor, which suggests that "intelligence" emerges through a large number of neurons connected together, each performing a simple operation. In this context, it can be said that an artificial neuron (processing element) depicts behavior of the biological neuron. Each processing element is interconnected to

others following a specific interconnection scheme. Usually three layers of Processing Elements (PE) form the architecture of a NN: the input layer, the hidden layer and the output layer.

The input layer: input neurons are used to encode data presented to the network for processing. The number of parameters in the input data dictates number of input neurons.

The hidden layer: the neurons in it are called hidden neurons since they are not directly accessible to the user. They provide non-linearity in the network behavior. The number of hidden neurons is usually determined by trial and error or can be estimated from the statistics of the training data. The size depends on the level of non-linearity in the training data. Generally for a high level of non-linearity and a large number of data, a larger number of hidden neurons will be required. But one has to be cautious not to use a needlessly large number of hidden neurons as this may lead to poor generalization.

The output layer: output neurons store possible output values for the problem under consideration. The number of output neurons in the output layer is dictated by the problem.

Each input neuron is connected to all neurons in the hidden layer, and each hidden neuron is connected in turn to all neurons in the output layer, as shown in Figure 2.15. Some network architectures such as cascade correlation-learning network (Fahlman and Lebiere 1990) allow direct connections between input and output neurons and between hidden and other hidden neurons. A weight is associated with each connection. The

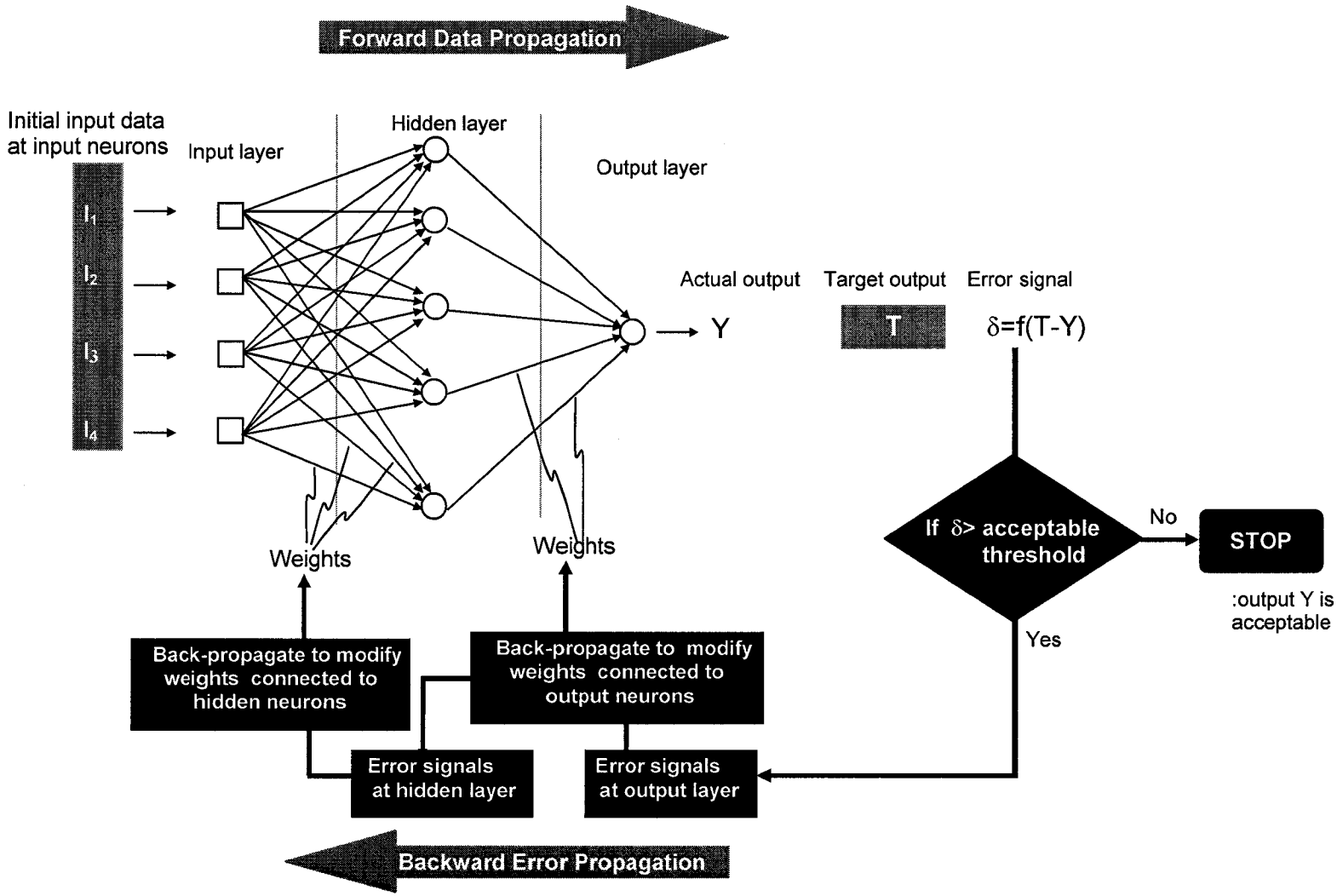


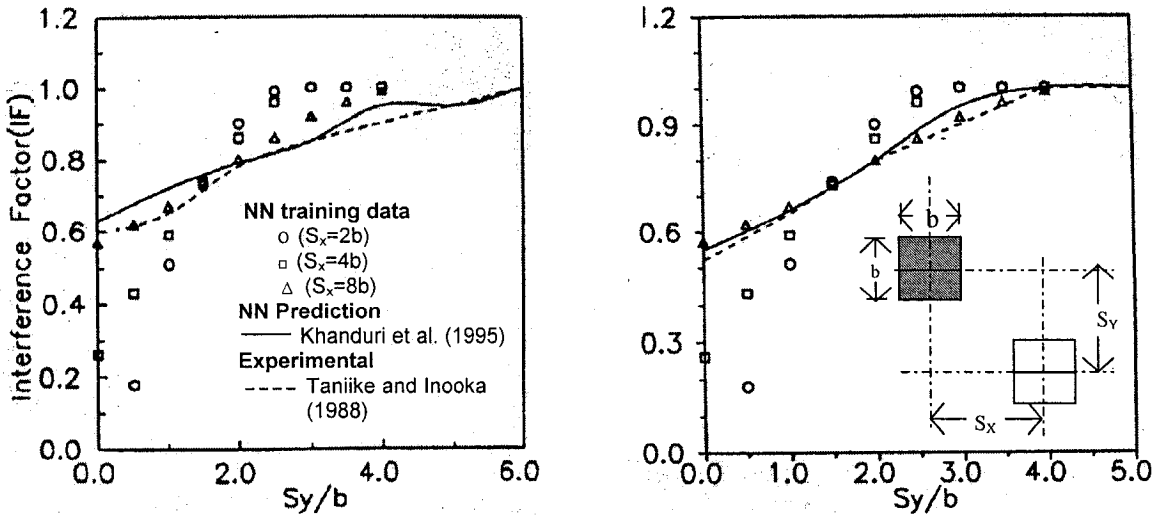
Figure 2.15 Architecture of neural network and error back-propagation

knowledge required to map input pattern into an appropriate output pattern is embedded in the weights. Initially the weights are unknown. Until a set of appropriate weights is found the network has no ability to solve the problem. Different mathematical calculations and error optimization are involved in obtaining a useful set of weights. Mathematical derivations and detailed algorithms can be found in Khanduri et al. (1995) and Eberhart and Dobbins (1990). The process of obtaining these useful weights is called training. A training set is composed of sample input/output (target output) pairs defining the desired system behavior.

As already mentioned, NN learning is achieved by adjusting connection weights iteratively until a desired output set is obtained with an acceptable error. In practice, training iterations are stopped when the error calculated during learning stage reaches below a fixed threshold error specified by the user. Once the learning is over, the NN can be used for predicting the parameters of interest. However, before a NN can be used with any degree of confidence, it must be validated. Usually, validation involves evaluating the network performance on a set of test problems (test set) that were not used for training, but for which solutions are available. Comparison between the correct solutions and those produced by the network can be established in a qualitative manner (such as a visual comparison of plotted points) or in a quantitative manner using statistical tests such as correlation coefficient and/or root mean square error. In the present study both qualitative and quantitative methods have been used. The test problems are selected randomly in order to reduce the likelihood of significant bias in results.

2.5.2 Neural network applications in wind engineering

Though many applications of Neural Network in Civil Engineering have been reported in literature, only few have been reported in Wind Engineering. Sandri and Mehta (1995) first demonstrated the use and adaptability of NN to wind-induced damage predictions. Khanduri et al. (1995, 1997) suggested a neural network approach for the assessment of wind-induced interference effects on design wind loads for buildings. To test the applicability of neural network methodology for modeling interference effects, they developed a back-propagation neural network example to predict the mean and dynamic along-wind and across-wind interference factors when two square buildings interact. The Interference Factors (IF) is a ratio of base overturning moment on a building in the presence of an upstream building to that on an isolated building. The input parameters considered were the center-to-center spacing between the two square buildings in the along wind (S_x) and across wind (S_y) directions. The output parameter was the corresponding value of IF. Twenty-seven training sets from relevant experiments were used for training. Their predictions agreed well with the experimental data as shown in Figure 2.16 where b is the building width.

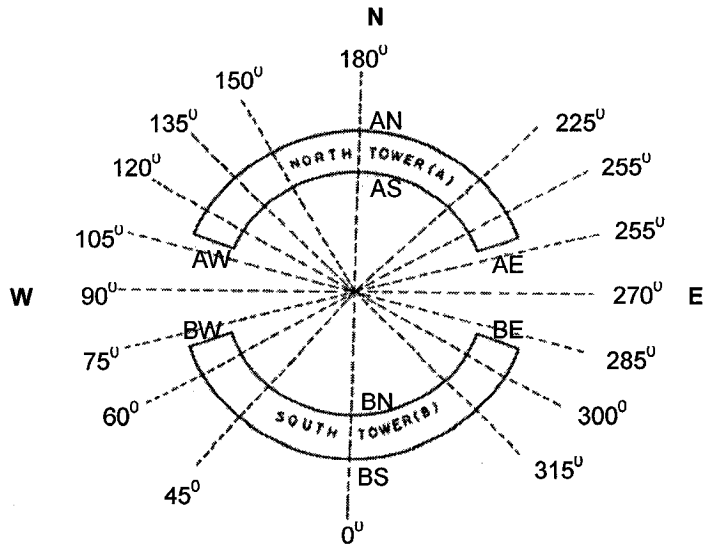


(a) S_x=10b (b) S_x =7b

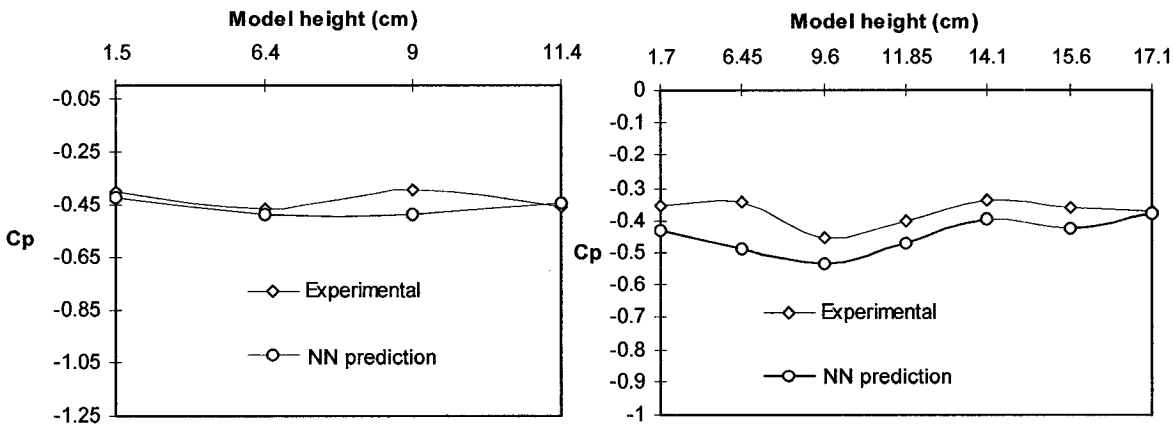
Figure 2.16 Comparison of NN predicted and experimental values for mean along-wind IF for (a) S_x=10b and (b) S_x =7b.

Bitsuamlak and Godbole (1999) used NN to predict wind pressure coefficients for a high-rise building with complex shape after training the neural network with wind tunnel data. The input parameters consist of the location of each pressure point on the wall of the building (height from the ground). The corresponding output parameter is the dimensionless pressure coefficient Cp. Typical prediction comparisons in which Cp values corresponding to wind direction of 90^0,which was not considered during training the NN, are shown in Figure 2.17.

Other works are reported in the literature. English and Fricke (1999) described a NN application for analysis of interference index. Kwatra et al. (1999) used a neural network for predicting wind-induced pressures in various zones of a gable building. More recently, Chen et al. (2003) applied NN to predict pressure coefficients on roofs of low buildings.



(a)



(b) (c)

Figure 2.17 (a) Definition of wind directions. NN prediction comparisons for pressure coefficients Cp at (b) face BE, and (c) face AN.

2.5.3 Closure

Neural network applications in wind engineering described briefly in the previous section prove the ability of NN to predict useful design parameters reasonably accurately once trained with limited available experimental data. So far no work has been done in integrating NN and CFD approaches. Also NN normally depend on experimental data. In the present study however, the NN will be trained with numerically obtained data (CFD data) thus making the whole process computational. The objective is to develop a combined CFD-NN approach that will therefore require less resource as compared to the BLWT-NN approach. In addition, it will also be easier than using direct CFD approach for the end-user since in the CFD-NN approach, the end-users will not have to deal with the complexity associated with CFD approach. In effect a trained NN requires only simple input such as terrain geometries and roughness to estimate design wind load parameters.

3 Numerical methodology

As highlighted in the literature review, there is a need for having a numerical tool for estimating the effect of topography on design wind load parameters. Such a numerical tool is required to evaluate wind flow over a wide spectrum of terrain geometries and roughness conditions. Although previous numerical work exists in the literature, the present study focuses mainly on: efficient incorporation of influential parameters such as roughness of the terrain in the numerical model and quantification of the improvements due to their incorporation, better terrain geometry representation and quality of grid, software development based on object oriented design, extensive validation of numerical results and finally generation of useful design wind load parameters (such as speed-up ratios) for cases that are not covered by codes and standards. A numerical approach has been selected instead of physical simulations mainly for economical reasons. This proposition appears particularly advantageous when recent improvements in hardware/ software technologies are taken into account together with better cost effectiveness.

One of the strategies followed in the present study is that the mathematical model adopted should enable the end-user to get a reasonably accurate result by using personal computers. According to the information technology survey conducted in Canada by Rivard (2000), 99% of the target engineering community uses personal computers, 90 % out of which uses Microsoft operating systems. The CFD tool developed has been designed and implemented to run on personal computers with such operating systems. In the adopted wind flow governing equations U, W, k and ε represent mean values. Thus solutions can be obtained within reasonable time frame by using a standard PC. It is to be

noted that the present study is limited only to wind flow over two-dimensional single and multiple hills (hills with very large width compared to their length L), two-dimensional valleys (valleys with very large width compared to their length L), and two-dimensional escarpments (escarpments with very large width compared to their length L). These shapes are believed to cover a wide spectrum of actual terrain geometries.

In the following sections, CFD methodology is presented including governing equations in a general form, details of their discretization, solution procedures and application of boundary conditions. Object-oriented design and analysis of a new CFD tool developed is presented. Finally, details of the proposed nearly orthogonal grid generation procedure are given.

3.1 CFD-methodology

3.1.1 Governing equations in the general form

The governing equations employed are the RANS equations together with the standard (Launder and Spalding 1974) and/or renormalization group (Yakhot et al. 1992) k-ε turbulence models. The governing wind flow equations used for the present study in Cartesian coordinates are shown in Section 2.3. They include continuity equation (2.3), RANS equation (2.8), k-ε turbulence model equations (2.10) and (2.11) respectively. These equations can be rewritten and represented by using a single equation in a general form as shown below,

$$\underbrace{\frac{\partial(\rho U\Phi)}{\partial x}+\frac{\partial(\rho W\Phi)}{\partial z}}_{\text{advection terms}}=\underbrace{\frac{\partial}{\partial x}\left[\Gamma\frac{\partial\Phi}{\partial x}\right]+\frac{\partial}{\partial z}\left[\Gamma\frac{\partial\Phi}{\partial z}\right]}_{\text{diffusion terms}}+\underbrace{S}_{\text{source term}} \tag{3.1}$$

where Γ is an effective diffusion coefficient and $S = S_c + S_p\Phi$ where S_c and S_p are called constant and variable source terms respectively. The source term represents all other terms in the governing equations that are not included in the advection or diffusion terms. Table 3.1 defines the appropriate expressions for the diffusion and source terms corresponding to each dependent variable (i.e. U, W, k or ε) represented by Φ in the general equation (3.1). The recognition that all the relevant differential equations can be considered as particular cases of the general Φ equation is an important time saving step and makes it possible to concern ourselves only with the numerical solution of general equation (3.1). Hence, in the present study a general computer program has been written to solve equation (3.1) and repeatedly used for different meanings of Φ along with appropriate expressions for diffusion coefficient Γ and source terms S_c and S_p as given in Table 3.1, as well as appropriate initial and boundary conditions. The concept of the general Φ equation has enabled the formulation of a general numerical method and the preparation of general-purpose computer program.

Table 3.1 Appropriate expressions for diffusion coefficient and source terms in the general equation

Equation	Φ	Γ	S_c	S_p
Continuity	1	0	0	0
x-momentum	U	$\nu_t+\nu$	$\frac{\partial}{\partial x}\left(\nu_t\frac{\partial U}{\partial x}\right)+\frac{\partial}{\partial z}\left(\nu_t\frac{\partial W}{\partial x}\right)-\frac{\partial P}{\partial x}$	0
z-momentum	W	$\nu_t+\nu$	$\frac{\partial}{\partial x}\left(\nu_t\frac{\partial U}{\partial z}\right)+\frac{\partial}{\partial z}\left(\nu_t\frac{\partial W}{\partial z}\right)-\frac{\partial P}{\partial z}$	0
Turbulence energy	k	$\frac{\nu_t}{\sigma_k}+\nu$	$\nu_t\left\{2\left\{\left(\frac{\partial U}{\partial x}\right)^2+2\left(\frac{\partial W}{\partial z}\right)^2\right\}+\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)^2\right\}$	$\frac{\varepsilon}{k}$
Energy dissipation	ε	$\frac{\nu_t}{\sigma_k}+\nu$	$\frac{\varepsilon C_1\nu_t}{k}\left\{2\left\{\left(\frac{\partial U}{\partial x}\right)^2+2\left(\frac{\partial W}{\partial z}\right)^2\right\}+\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)^2\right\}$	$-\frac{C_2\varepsilon}{k}$

3.1.2 Transformation of the governing equations

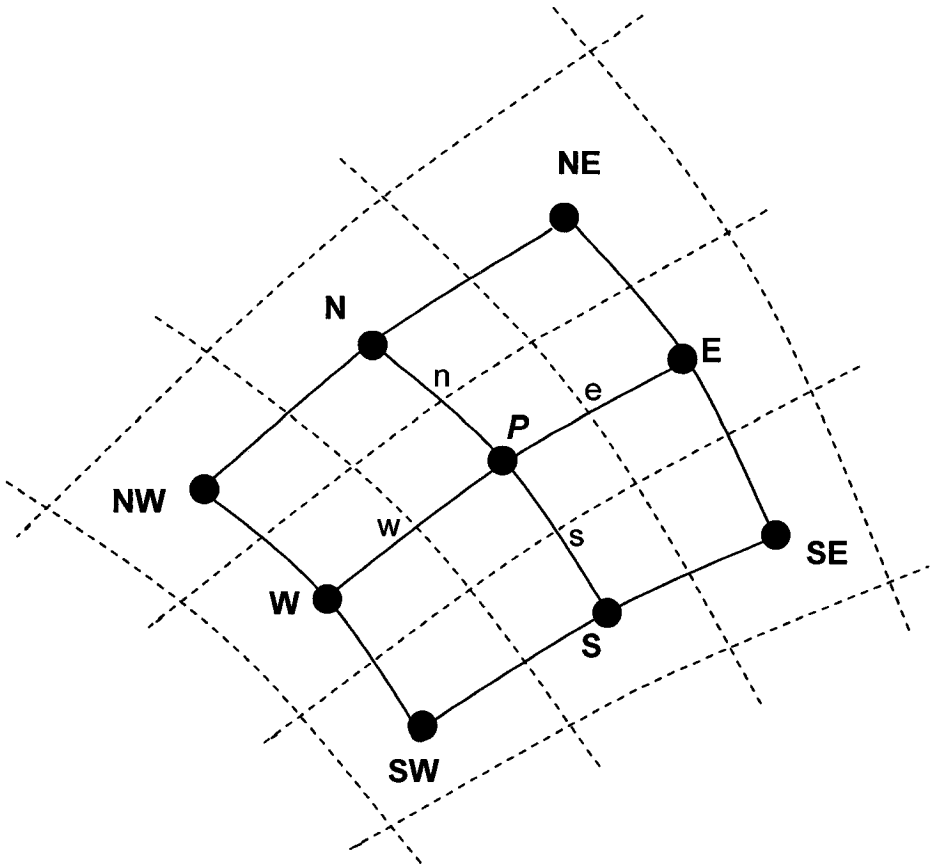
The physical feature of the representative topographies considered in the present study are best described by body-fitted curvilinear coordinates in preference to Cartesian coordinate systems which used staircase type of approximation to represent the slopes of the cross-section of the terrain. The latter introduce errors particularly when the ground boundary conditions are applied. In contrast, in body-fitted curvilinear coordinate system the boundary coordinates are fitted to the cross-section of the terrain thus representing the exact shape of the terrain. As demonstrated in section 4.3.2, the numerical solutions are highly sensitive to geometrical representation of the terrain. Use of the body-fitted grids has necessitated utilization of general curvilinear coordinates. The objective of this section is therefore to transform the governing equations given in Cartesian form in equations (3.1) into new independent variables $\xi(x,z)$ and $\eta(x,z)$. The grid points in the physical plane and the transformed plane are shown in Figure 3.1 (a) and (b) respectively.

Partial derivatives are transformed using the chain rule of partial differentiation:

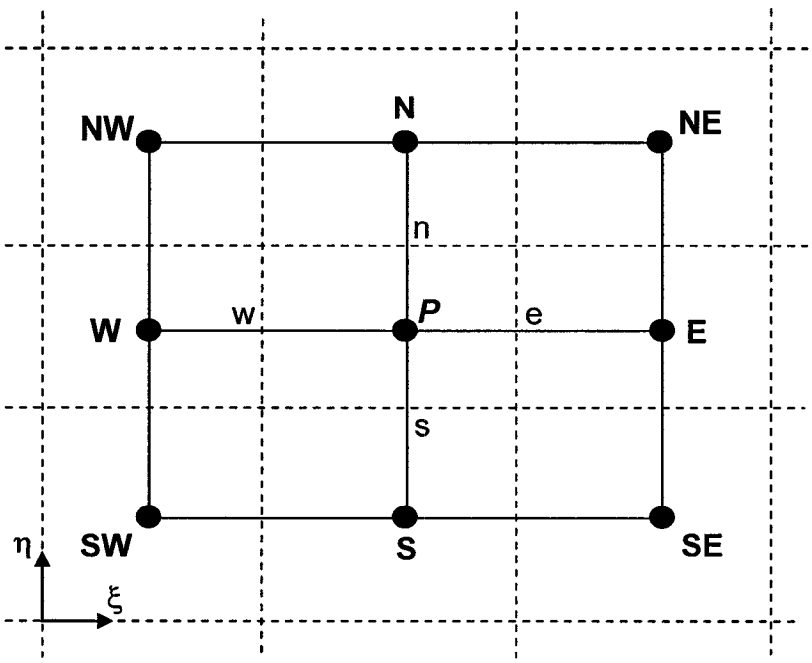
$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} \\ \frac{\partial \xi}{\partial z}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial z}\frac{\partial}{\partial \eta} \end{bmatrix} \tag{3.2}$$

The metrics appearing in these equations, i.e. $\partial\xi/\partial x$, $\partial\eta/\partial x$, $\partial\xi/\partial z$ and $\partial\eta/\partial z$ are determined in the following manner. Recalling the differential expressions as

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial z} \end{bmatrix} \begin{bmatrix} dx \\ dz \end{bmatrix} \tag{3.3}$$



(a) Physical plane.



(b) Transformed plane.

Figure 3.1 Description of physical plane and transformed plane.

and,

$$\begin{bmatrix} dx \\ dz \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{3.4}$$

Substitution of equation (3.4) into equation (3.3) provides

$$\begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} \end{bmatrix}^{-1} = \frac{1}{\frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \xi}} \begin{bmatrix} \frac{\partial z}{\partial \eta} & -\frac{\partial x}{\partial \eta} \\ -\frac{\partial z}{\partial \xi} & \frac{\partial x}{\partial \xi} \end{bmatrix} \tag{3.5}$$

Therefore, using equations (3.2) and (3.5), transformation of partial derivatives for any function f is given by

$$\frac{\partial f}{\partial x} = \frac{1}{J}\left(\frac{\partial z}{\partial \eta}\frac{\partial f}{\partial \xi} - \frac{\partial z}{\partial \xi}\frac{\partial f}{\partial \eta}\right) \tag{3.6}$$

$$\frac{\partial f}{\partial z} = \frac{1}{J}\left(-\frac{\partial x}{\partial \eta}\frac{\partial f}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial f}{\partial \eta}\right) \tag{3.7}$$

where J is the Jacobian of the transformation given by

$$J = \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \xi} \tag{3.8}$$

When this type of transformation is applied to the governing equations (3.1), the following transformed equations are obtained

$$\frac{1}{J}\frac{\partial}{\partial\xi}\left\{\left(\rho U\frac{\partial z}{\partial\eta}-\rho W\frac{\partial x}{\partial\eta}\right)\Phi\right\}+\frac{1}{J}\frac{\partial}{\partial\eta}\left\{\left(\rho W\frac{\partial x}{\partial\xi}-\rho U\frac{\partial z}{\partial\xi}\right)\Phi\right\}$$
$$=\frac{1}{J}\frac{\partial}{\partial\xi}\left[\frac{\Gamma}{J}\left\{\left(\left(\frac{\partial x}{\partial\eta}\right)^2+\left(\frac{\partial z}{\partial\eta}\right)^2\right)\frac{\partial\Phi}{\partial\xi}-\left(\frac{\partial x}{\partial\xi}\frac{\partial x}{\partial\eta}+\frac{\partial z}{\partial\xi}\frac{\partial z}{\partial\eta}\right)\frac{\partial\Phi}{\partial\eta}\right\}\right]+ \tag{3.9}$$
$$\frac{1}{J}\frac{\partial}{\partial\eta}\left[\frac{\Gamma}{J}\left\{\left(\left(\frac{\partial x}{\partial\xi}\right)^2+\left(\frac{\partial z}{\partial\xi}\right)^2\right)\frac{\partial\Phi}{\partial\eta}-\left(\frac{\partial x}{\partial\xi}\frac{\partial x}{\partial\eta}+\frac{\partial z}{\partial\xi}\frac{\partial z}{\partial\eta}\right)\frac{\partial\Phi}{\partial\xi}\right\}\right]+S$$

Using new notations defined below as used by Rhie and Chow (1983),

$$G_1=\frac{\partial z}{\partial\eta}U-\frac{\partial x}{\partial\eta}W,$$

$$G_2=\frac{\partial x}{\partial\xi}U-\frac{\partial z}{\partial\xi}W,$$

$$\alpha_1=\left(\frac{\partial x}{\partial\eta}\right)^2+\left(\frac{\partial z}{\partial\eta}\right)^2,$$

$$\beta_1=\frac{\partial x}{\partial\xi}\frac{\partial x}{\partial\eta}+\frac{\partial z}{\partial\xi}\frac{\partial z}{\partial\eta}, \text{ and}$$

$$\gamma_1=\left(\frac{\partial x}{\partial\xi}\right)^2+\left(\frac{\partial z}{\partial\xi}\right)^2 \tag{3.10}$$

Equations (3.9) reduce to the following compact transformed equation:

$$\frac{1}{J}\frac{\partial}{\partial\xi}\{\rho G_1\Phi\}+\frac{1}{J}\frac{\partial}{\partial\eta}\{\rho G_2\Phi\}$$
$$=\frac{1}{J}\frac{\partial}{\partial\xi}\left\{\frac{\Gamma}{J}\left(\alpha_1\frac{\partial\Phi}{\partial\xi}-\beta_1\frac{\partial\Phi}{\partial\eta}\right)\right\}+\frac{1}{J}\frac{\partial}{\partial\eta}\left\{\frac{\Gamma}{J}\left(\gamma_1\frac{\partial\Phi}{\partial\eta}-\beta_1\frac{\partial\Phi}{\partial\xi}\right)\right\}+S \tag{3.11}$$

At this point, it is worthwhile to consider the physical interpretation associated with definitions given by equation (3.11). For orthogonal ξ and η coordinates, β_1 vanishes and

$\sqrt{\alpha_1}$ and $\sqrt{\gamma_1}$ represent the scale factors. As the coordinates ξ and η deviate from orthogonality, β_1 increases accordingly. Therefore, β_1 is interpreted as the degree of orthogonality of the local grid. G_1 and G_2 have also important physical meaning. In fact $G_1/\sqrt{\alpha_1}$ and $G_2/\sqrt{\gamma_1}$ represent velocity components normal to the lines of constant ξ and η respectively in the physical plane.

3.1.3 Discretization of the governing equations

In applying the wind flow governing equations in the present study, first the computational domain consisting of the topographical element of interest is separated from the entire domain as shown in Figure 3.2. This is due to the fact that effect of the topography is more pronounced in the vicinity of a topographic element and as well to minimize the computational load that would be otherwise incurred by treating needlessly a large computational domain. The size of the computational domain is decided based on preliminary numerical experimentation and values given in literature has been used starting point as discussed later in Chapter 4. Typical computational domain dimensions are shown in Figure 3.2.

Once the computational domain is chosen, the effect of the entire domain on it is considered through proper application of the boundary conditions at each boundary of the computational domain. Details of boundary conditions are given in section 3.2.4. The computational domain is then discretized into many space elements called control volumes (CV). Upon integrating the transformed governing equations (3.11) over a control volume in the ξ- and η- plane, the following integral conservation relation is obtained

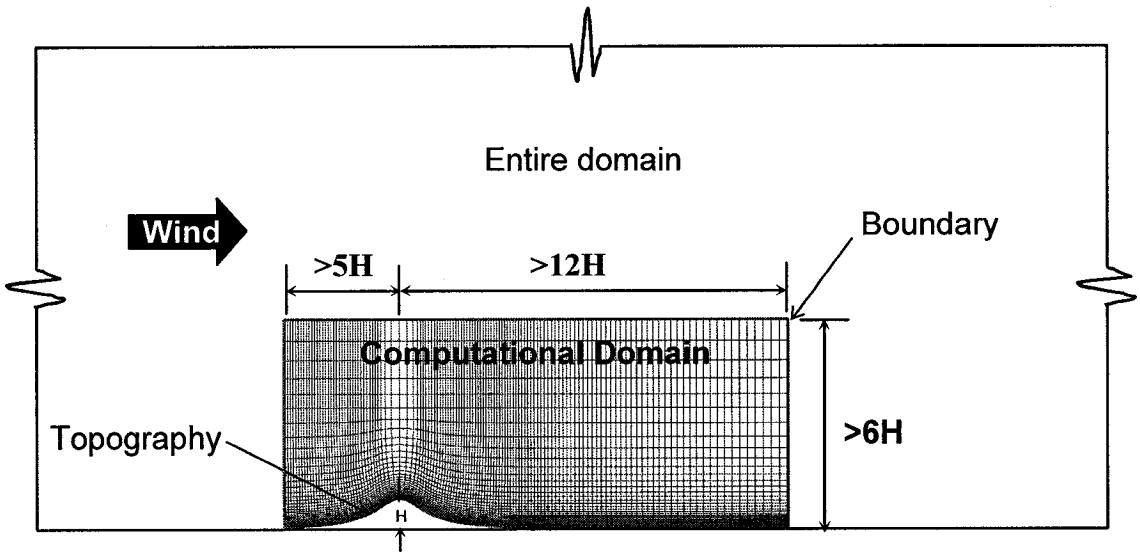


Figure 3.2 Description of whole domain and computational domain.

$$\int_B \{\rho G_1 \Phi d\eta + \rho G_2 \Phi d\xi\} = \int_B \left\{ \frac{\Gamma}{J}\left(\alpha_1 \frac{\partial \Phi}{\partial \xi} - \beta_1 \frac{\partial \Phi}{\partial \eta}\right) d\eta + \frac{\Gamma}{J}\left(\gamma_1 \frac{\partial \Phi}{\partial \eta} - \beta_1 \frac{\partial \Phi}{\partial \xi}\right) d\xi \right\} + \iint_{CV} SJ d\xi d\eta \tag{3.12}$$

where B is the boundary of the CV .

Using the notations of Figure 3.3, for a typical center grid point *P* enclosed in the CV and surrounded by its neighbors E, W, N and S, the finite-difference approximation to the integral conservation relation equation (3.12) over the CV is written as:

$$(\rho G_1 \Phi \Delta\eta)_w^e + (\rho G_2 \Phi \Delta\xi)_s^n = \left\{ \frac{\Gamma}{J}\left(\alpha_1 \frac{\partial \Phi}{\partial \xi} - \beta_1 \frac{\partial \Phi}{\partial \eta}\right) \Delta\eta \right\}_w^e + \left\{ \frac{\Gamma}{J}\left(\gamma_1 \frac{\partial \Phi}{\partial \eta} - \beta_1 \frac{\partial \Phi}{\partial \xi}\right) \Delta\xi \right\}_s^n + \iint_{CV} SJ d\xi d\eta \tag{3.13}$$

Denoting the flow terms such as $\rho G \Delta\xi$ by F and utilizing linear interpolation for the dependent variable Φ at the faces of the CV, for example at east face $\Phi_e=(\Phi_E+\Phi_P)/2$, the left hand side of equation (3.13) can be written as

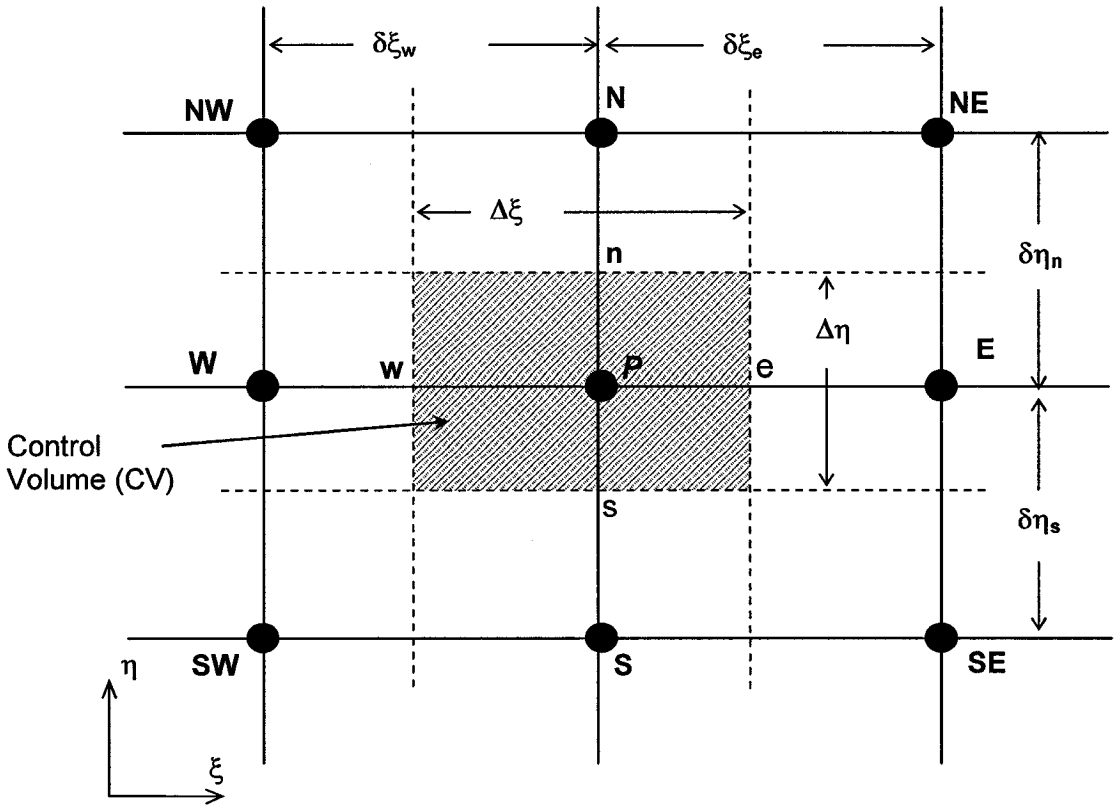


Figure 3.3 Definitions of control volume and related geometrical parameters

$$F_e\left(\frac{\Phi_E+\Phi_P}{2}\right)-F_w\left(\frac{\Phi_P+\Phi_W}{2}\right)+F_N\left(\frac{\Phi_N+\Phi_P}{2}\right)-F_S\left(\frac{\Phi_P+\Phi_S}{2}\right) \tag{3.14}$$

Similarly, using first order finite 1-$\delta\xi$ difference approximations for the derivatives, the derivative of Φ at the east face is approximated by

$$\left(\frac{\partial\Phi}{\partial\xi}\right)_e=\frac{\Phi_E-\Phi_P}{\delta\xi_e} \tag{3.15}$$

This process is repeated to obtain the derivatives of Φ at w, n, and s faces of the CV. Denoting the diffusion terms such as $(\Gamma\alpha\Delta\eta)/(J\delta\xi)$ and $(\Gamma\gamma\Delta\xi)/(J\delta\eta)$ by D, the right hand side of equation (3.12) is written as

$$D_e(\Phi_E-\Phi_P)+D_w(\Phi_P-\Phi_W)+D_n(\Phi_N-\Phi_P)+D_s(\Phi_P-\Phi_S)$$
$$+SJ\Delta\xi\Delta\eta-\left\{\left(\frac{\Gamma}{J}\beta_1\Delta\eta\frac{\partial\Phi}{\partial\eta}\right)_w^e+\left(\frac{\Gamma}{J}\beta_1\Delta\xi\frac{\partial\Phi}{\partial\xi}\right)_s^n\right\} \tag{3.16}$$

the last terms within the curly brackets in equation (3.16) are the results of the cross derivatives in the diffusion terms due to non-orthogonal coordinate. These terms are usually very small if nearly orthogonal coordinates are used as in the present case. Therefore, these terms are combined into the source term and they are denoted as S_d henceforth and treated as known quantities (rather estimated from values in the previous iterations). Putting the LHS, i.e. equation (3.14), and the RHS, i.e. equation (3.16), together and rearranging some terms and introducing some other new terms defined below provides the following discretized form of the governing equations.

$$A_P\Phi_P=A_E\Phi_E+A_W\Phi_W+A_N\Phi_N+A_S\Phi_S+b$$

or in compact form

$$A_P\phi_P=\sum_{EWNS}A\Phi+b \tag{3.17}$$

where

$$A_E=D_e-\frac{F_e}{2},$$

$$A_N=D_n-\frac{F_n}{2},$$

$$A_S=D_s+\frac{F_s}{2},$$

$b=S_C+S_d$, and

$$A_P = A_E + A_W + A_N + A_S - S_p \Delta V \tag{3.18}$$

where F represents the flow terms and D represents the diffusion terms. Recall

$$F_e = (\rho G_1 \Phi)_e,\ F_w = (\rho G_1 \Phi)_w,\ F_n = (\rho G_2 \Phi)_n, \text{ and } F_s = (\rho G_2 \Phi)_s \tag{3.19}$$

$$D_e = \left(\frac{\Gamma \alpha \Delta \eta}{J \delta_\xi}\right)_e,\ D_w = \left(\frac{\Gamma \alpha \Delta \eta}{J \delta_\xi}\right)_w,\ D_n = \left(\frac{\Gamma \alpha \Delta \xi}{J \delta_\xi}\right)_n, \text{ and } D_s = \left(\frac{\Gamma \alpha \Delta \eta}{J \delta_\xi}\right)_s \tag{3.20}$$

The coefficients in equation (3.17) are updated according to a hybrid-scheme of Spalding (1972) during the iterative solution procedure to be discussed later. Hybrid scheme uses central differences when the CV-Peclet number is less than 2 and upwind differences (Patankar 1980) when the CV-Peclet number is greater than or equal to 2. CV-Peclet number is defined as the ratio of the flow term (F) over the diffusion (D) term as given by equation (3.23). If central difference scheme alone is used, some of the coefficients A_E, A_W, A_N, and A_S can become negative when the convective terms are large. Physically this is unrealistic (Patankar 1980) and under this condition the equation ceases to be diagonally dominant. The typical technique used to solve it, i.e. Tri-Diagonal Matrix Analysis (TDMA), becomes unstable and will not converge.

Using a more general function notation $A(|Pe|)$ of Patankar (1980), $A(|Pe|)$ for a hybrid scheme is given by

$$A(|Pe|) = \max[0, (1 - 0.5|Pe|)] \tag{3.21}$$

The function max [a, b] returns the maximum of its two arguments a and b. Using this notation the coefficients can be rewritten as

$$A_E = D_e A(|Pe|) + \max[-F_e, 0],$$

$$A_W = D_w A(|Pe|) + \max[F_w, 0],$$

$$A_N = D_n A(|Pe|) + \max[-F_n, 0],$$

$$A_S = D_s A(|Pe|) + \max[F_s, 0],$$

$b = S_C + S_d$, and

$$A_P = A_E + A_W + A_N + A_S - S_p \Delta V \tag{3.22}$$

where Pe is Peclet number given by

$$|Pe| = \left|\frac{F}{D}\right| \tag{3.23}$$

substituting the appropriate expressions for the flow (F) and the diffusion (D) terms as given in equations (3.19) and (3.20), the Peclet number at east face of the control volume is

$$|Pe|_e = \left|\frac{\rho G_1 J \delta_\xi}{\Gamma \alpha}\right|_e \tag{3.24}$$

Peclet numbers at w, n, and s CV faces are obtained similarly. In these equations e, w, n, and s represent the east, west, north, and south faces of the control volume respectively. E, W, N, and S represent east, west, north, and south grid point respectively. *P* represents the center grid point. S_c is the constant source term and S_d represents diffusion terms resulting from cross derivatives and placed in the source terms. Source terms for each of the dependent variables are obtained in the following manner.

3.1.3.1 Source term in the x- and z - momentum equations

The source term in x-momentum equation consists of S_c (the constant source term) and S_d (the diffusion term resulting from cross derivatives and placed in the source terms). S_c as given in Cartesian coordinates in Table 3.1 is also written as:

$$S_C = \frac{\partial}{\partial x}\left(\nu_t \frac{\partial U}{\partial x}\right) + \frac{\partial}{\partial z}\left(\nu_t \frac{\partial W}{\partial x}\right) - \frac{\partial P}{\partial x} \tag{3.25}$$

Prior to transforming this S_c expression into a general coordinate, the terms $\nu_t \partial U/\partial x$ and $\nu_t \partial W/\partial x$ are represented by λ and ψ respectively. Then transformation equations (3.6) and (3.7) are applied and the source terms are integrated over the CV as:

$$\iint S_C J \Delta\xi \Delta\eta = \left\{\left(\frac{\partial z}{\partial \eta}\frac{\partial \lambda}{\partial \xi}\right)\left(\frac{\partial z}{\partial \eta}\frac{\partial \lambda}{\partial \xi}\right) - \left(\frac{\partial z}{\partial \xi}\frac{\partial \lambda}{\partial \eta}\right) + \left(\frac{\partial x}{\partial \xi}\frac{\partial \psi}{\partial \eta}\right) - \left(\frac{\partial x}{\partial \eta}\frac{\partial \varphi}{\partial \xi}\right) + \left(\frac{\partial z}{\partial \xi}\frac{\partial P}{\partial \eta}\right) - \left(\frac{\partial z}{\partial \eta}\frac{\partial P}{\partial \xi}\right)\right\}_P \Delta\xi\Delta\eta \tag{3.26}$$

Subsequently, the λ and ψ's are transformed in the following manner

$$\lambda = \frac{\nu_t}{J}\left(\frac{\partial z}{\partial \eta}\frac{\partial U}{\partial \xi} - \frac{\partial z}{\partial \xi}\frac{\partial U}{\partial \eta}\right) \tag{3.27}$$

$$\psi = \frac{\nu_t}{J}\left(\frac{\partial z}{\partial \eta}\frac{\partial W}{\partial \xi} - \frac{\partial z}{\partial \xi}\frac{\partial W}{\partial \eta}\right) \tag{3.28}$$

Derivatives such as $\{\partial z/\partial x\}_P$ in equations (3.26) to (3.28) are discretized as

$$\left(\frac{\partial z}{\partial \eta}\right)_P = \frac{Z_N - Z_S}{\delta\eta_n + \delta\eta_s} \tag{3.29}$$

The other component of source term in the x-momentum equation, S_d, which is given as part of equation (3.15) is also written as

$$S_d = \left(\frac{\Gamma}{J}\beta_1\Delta\eta\frac{\partial\Phi}{\partial\eta}\right)_e - \left(\frac{\Gamma}{J}\beta_1\Delta\eta\frac{\partial\Phi}{\partial\eta}\right)_w + \left(\frac{\Gamma}{J}\beta_1\Delta\xi\frac{\partial\Phi}{\partial\xi}\right)_n - \left(\frac{\Gamma}{J}\beta_1\Delta\xi\frac{\partial\Phi}{\partial\xi}\right)_s \tag{3.30}$$

The first term in RHS of equation (3.30) is discretized as

$$\left(\frac{\Gamma}{J}\beta_1\Delta\eta\frac{\partial\Phi}{\partial\eta}\right)_e = \left(\frac{\Gamma}{J}\beta_1\Delta\eta\right)_e\left(\frac{\Phi_{NE} - \Phi_{SE} + \Phi_N + \Phi_S}{4}\right) \tag{3.31}$$

where $(\Gamma\beta_1\Delta\eta/J)_e$ is obtained using linear interpolation between $(\Gamma\beta_1\Delta\eta/J)_E$ and $(\Gamma\beta_1\Delta\eta/J)_P$. The remaining terms can be discretized similarly.

A similar discretization procedure is also used to obtain discretized form of the sources in the z-momentum equation.

3.1.3.2 Other source terms

The source in k-equation consists of S_c (the constant source term) as given in Cartesian coordinates in Table 3.1 and also written as:

$$S_C = \nu_t\left\{2\left(\frac{\partial U}{\partial x}\right)^2 + 2\left(\frac{\partial W}{\partial z}\right)^2 + \left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right)^2\right\} \tag{3.32}$$

Prior to transforming this S_c expression into a general coordinate, the terms $\partial U/\partial x$, $\partial W/\partial x$, $\partial U/\partial z$ and $\partial W/\partial z$ are represented by λ_1, λ_2, λ_3 and λ_4, respectively. Then transformation equations (3.6) and (3.7) are applied and the source terms are integrated over the CV as:

$$S_C = \nu_t\left\{2\left(\lambda_1^2 + \lambda_4^2\right) + \left(\lambda_2 + \lambda_3\right)^2\right\}J\Delta\xi\Delta\eta \tag{3.33}$$

Subsequently, the λ terms are transformed in the following manner

$$\lambda_1 = \frac{1}{J}\left(\frac{\partial z}{\partial\eta}\frac{\partial U}{\partial\xi} - \frac{\partial z}{\partial\xi}\frac{\partial U}{\partial\eta}\right) \tag{3.34}$$

Similar transformations for λ_2, λ_3 and λ_4 can be obtained.

Following a similar discretization procedure, the discretized form of the source terms in the ε-equation is obtained. Thus S_c for ε-equations is given by

$$S_c = \frac{\varepsilon C_1 \nu_t}{k}\left\{2\left(\lambda_1^2 + \lambda_4^2\right) + \left(\lambda_2 + \lambda_3\right)^2\right\} J \Delta\xi \Delta\eta \,. \tag{3.35}$$

3.1.4 Boundary conditions

The governing equations are basically a group of partial differential equations and appropriate boundary conditions are required in order to solve them. The boundary conditions are expressed as conditions on the velocities, turbulent kinetic energy and turbulent kinetic energy dissipation at the boundaries of the computational domain. The boundary conditions are discussed individually below and summarized also as shown in Figure 3.4.

3.1.4.1 Upstream boundary condition on velocity

Log-law velocity distribution corresponding to the upstream terrain condition is applied at the upstream boundary as follows

$$U = \frac{U^*}{\kappa} \ln\left(\frac{z}{z_0}\right) \tag{3.36}$$

$$W = 0 \tag{3.37}$$

3.1.4.2 Upstream boundary conditions of turbulence properties

Upstream boundary conditions of k and ε are considered the same as those of the oncoming wind flow, which is the flow undisturbed by the terrain. This flow is the so-called atmospheric boundary layer flow. For a horizontal wind, W=0 and the variation

along the x direction is also supposed to be zero $\partial/\partial x = 0$. Imposing these conditions the following simplified k-ε equations are obtained

k-equation:

$$\frac{\partial}{\partial z}\left(\frac{\nu_t}{\sigma_k}\frac{\partial k}{\partial z}\right) + \nu_t\left(\frac{\partial U}{\partial z}\right)^2 - \varepsilon = 0 \tag{3.38}$$

ε-equation:

$$\frac{\partial}{\partial z}\left(\frac{\nu_t}{\sigma_k}\frac{\partial k}{\partial z}\right) + C_1 C_\mu\left(\frac{\partial U}{\partial z}\right)^2 - C_2\frac{\varepsilon^2}{k} = 0 \tag{3.39}$$

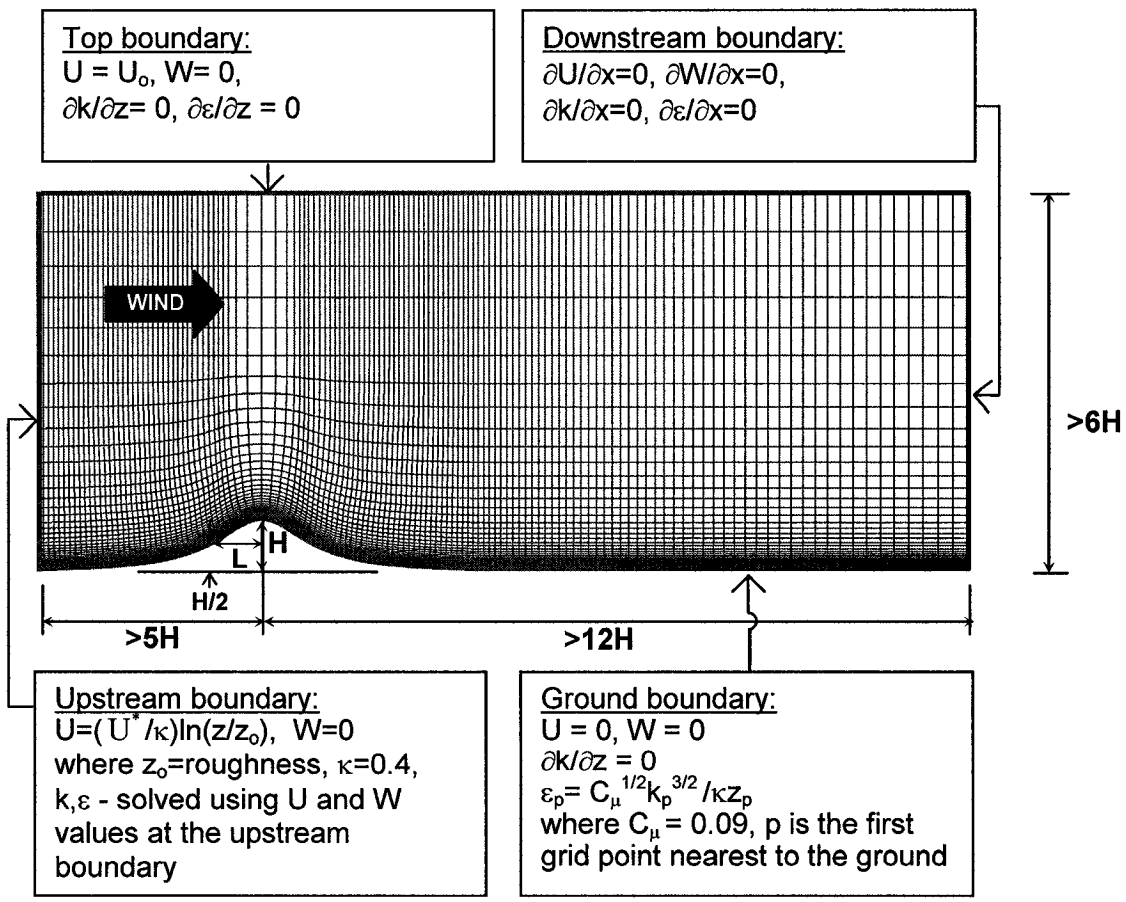


Figure 3.4 Computational domain dimensions and its boundary conditions.

These two simplified k-ε equations are transformed into curvilinear coordinates and are solved using the upstream velocity profile given by equations (3.36) and (3.37). The solutions to the simplified k-ε equations are then used as upstream boundary conditions for k and ε at the upstream boundary.

3.1.4.3 Downstream boundary conditions

By taking the computational domain large enough compared to the terrain obstruction and taking the downstream boundary far enough from the terrain to be simulated, it is assumed that all the dependent variables do not change in the x-direction. Thus $\partial\Phi/\partial x = 0$ is applied as a downstream boundary condition for all the dependent variables.

3.1.4.4 Ground boundary conditions

No slips condition on velocity (i.e. U=0 and W=0) are applied on the ground boundary; also roughness of the surface is introduced through the application of log-law to the control-volumes that are adjacent to the ground. In related previous studies the effect of roughness is taken into account only through the upstream boundary velocity profile (Bergeles 1985) by using log-law wind velocity profile that reflects the roughness condition at the upstream region. In the present study, the effect of roughness throughout the ground surface of the computational domain has been considered in addition to incorporating its effect at the upstream boundary using log-law velocity profile. The tangential velocity at the first grid point closer to the ground (i.e. z_p) is calculated by using log-law:

$$U_P = \frac{U^*}{\kappa} \ln\left(\frac{z_P}{z_o}\right) \tag{3.40}$$

The shear stress is then obtained as:

$$\tau_w = \frac{U_P}{z_P} \frac{\mu z_P^+}{\kappa \ln(z_P/z_0)} \tag{3.41}$$

Incorporating roughness of the topography within the computational domain into the governing equations is made through the computation of the shear stress given by equation (3.41) throughout the ground surface (and then incorporating it into the momentum equations). The shear stress given by equation (3.41) is acting tangent to the ground surface in the direction opposite to that of the local flow. Thus, its x-component $(\tau_w)_x$ decomposed using the local ground slope is incorporated in the x-momentum equation (i.e. U-equation) and the z-component $(\tau_w)_z$ is incorporated in the z-momentum equation (i.e. W-equation). Note that only the shear stress term in the momentum equations related to CV's adjacent to the ground are replaced by equation (3.41).

3.1.5 Convergence criteria

Numerical solution of non-linear equations involves use of estimated values of unknown dependent variables to calculate the coefficients in the discretization equations at every step in order to get new estimated values of the unknowns. This process is repeated until further repetitions (iterations) cease to produce any further significant changes in the estimated values of the unknowns. A common criterion of convergence is to evaluate the changes in the unknowns between two successive iterations until it is smaller than a certain threshold set by the user. However in most nonlinear problems the change in the dependent variable between successive iterations is intentionally slowed down (i.e. under relaxation). When heavy under relaxation is used, which is the usual case in most solution procedures for highly nonlinear equations, this may create an illusion of convergence

although the computed solution may be far from having converged. Hence care must be taken in interpreting the convergence criteria.

In the present study, two types of relative residue criteria have been used. A residue is the difference between the RHS and LHS value of the discretization equations after the computed value is substituted into the equation for each grid point. In the first criterion, a ratio of residue difference between successive iterations with respect to the residue of the previous iteration (i-1) defined by equation (3.42) has been used. As a second criterion, the ratio of the i^{th} iteration residue over the residue of the first iteration given by equation (3.43) has been used. These computed relative residual values are compared with accepted tolerance thresholds to stop the iteration.

$$\frac{\left|r^{i}\right| - \left|r^{i-1}\right|}{\left|r^{i-1}\right|} < \vartheta \tag{3.42}$$

$$\frac{\left|r^{i}\right|}{\left|r^{1}\right|} < \chi \tag{3.43}$$

According to Van Doormal and Raithby (1984) the optimal value of the relative residue (χ) ranges from 0.25 to 0.05. In the present study, the computation is considered convergent if the relative residues given by χ and ϑ for all equations are less than 0.25 and $1x10^{-6}$ respectively. In addition to the above criteria, the magnitude of the absolute values of the residues every 100 iterations are also plotted and inspected visually during the solution procedure. Based on this inspection in most cases maximum number of 5000 iterations has been set.

3.1.6 Solution Procedure

The procedure SIMPLEC (Semi-Implicit Method for Pressure-Linked Equations Consistent) of Van Doormal and Raithby (1984) is used for solving the discretized equations in order to obtain the flow field. The method is iterative and the calculations are done sequentially since other scalars such as k and ε are coupled to the momentum equations. In this procedure, the momentum equations are solved based on estimated pressure field denoted by P^*. The corresponding approximate velocity components thus obtained are U^* and W^*. However this velocity field does not satisfy the continuity equation until the correct pressure field is obtained and used in the momentum equations. Thus P^*, U^*, and W^* fields have to be corrected. In this solution procedure, a method to improve the estimated pressure as well as the velocity field is obtained by combining the continuity and momentum equations as explained below.

A preliminary set of velocity components, $U^*$and W^*, is obtained using the estimated P^* from equations (3.17) and the new terms B^U, C^U, B^W and C^W defined below in the following manner

$$U_P^* = \frac{\sum A_{nb} U_{nb}^*}{A_P} + \frac{S_d^U}{A_P} + \left(B^U \frac{\partial P^*}{\partial \xi} + C^U \frac{\partial P^*}{\partial \eta} \right) \tag{3.44}$$

$$W_P^* = \frac{\sum A_{nb} W_{nb}^*}{A_P} + \frac{S_d^W}{A_P} + \left(B^W \frac{\partial P^*}{\partial \xi} + C^W \frac{\partial P^*}{\partial \eta} \right) \tag{3.45}$$

where

$$B^U = \frac{-1}{A_P} \frac{\partial z}{\partial \eta} \Delta\xi\Delta\eta,$$

$$C^U = \frac{1}{A_P}\frac{\partial z}{\partial \xi}\Delta\xi\Delta\eta,$$

$$B^W = \frac{1}{A_P}\frac{\partial x}{\partial \eta}\Delta\xi\Delta\eta, \text{ and}$$

$$C^W = \frac{-1}{A_P}\frac{\partial x}{\partial \xi}\Delta\xi\Delta\eta \tag{3.46}$$

The corrected flow fields (U, W and P) are denoted as

$$U = U^* + U',$$

$$W = W^* + W', \text{ and}$$

$$P = P^* + P' \tag{3.47}$$

where P', U', and W' are called pressure and velocity corrections respectively. By subtracting equation (3.44) and (3.45) with the estimated variables from the same equation with the corrected variables, the following equations are obtained:

$$U'_P = \frac{\sum A_{nb}U'_{nb}}{A_P} + \left(B^U\frac{\partial P'}{\partial \xi} + C^U\frac{\partial P'}{\partial \eta}\right) \tag{3.48}$$

$$W'_P = \frac{\sum A_{nb}W'_{nb}}{A_P} + \left(B^W\frac{\partial P'}{\partial \xi} + C^W\frac{\partial P'}{\partial \eta}\right) \tag{3.49}$$

At this point an approximation is introduced. The first terms in equations (3.48) and (3.49) that contain the summation terms are dropped and equations (3.50) and (3.51) are obtained. Omission of these terms is the main approximation of the SIMPLEC algorithm. In fact the assumption made does not affect the final solution since in the final converged solution these omitted terms vanish.

$$U'_P = B^U \frac{\partial P'}{\partial \xi} + C^U \frac{\partial P'}{\partial \eta} \tag{3.50}$$

$$W'_P = B^W \frac{\partial P'}{\partial \xi} + C^W \frac{\partial P'}{\partial \eta} \tag{3.51}$$

Now the corrected velocity components are obtained from equations (3.47) as

$$U = U^* + \left(B^U \frac{\partial P'}{\partial \xi} + C^U \frac{\partial P'}{\partial \eta} \right) \tag{3.52}$$

$$W = W^* + \left(B^W \frac{\partial P'}{\partial \xi} + C^W \frac{\partial P'}{\partial \eta} \right) \tag{3.53}$$

Similarly the correction equations for G_1 and G_2 are obtained from equations (3.52) and (3.53).

$$G_1 = G_1^* + \left(B^U \frac{\partial z}{\partial \eta} - B^W \frac{\partial x}{\partial \eta} \right) \frac{\partial P'}{\partial \xi} + \left(C^U \frac{\partial z}{\partial \eta} - C^W \frac{\partial x}{\partial \eta} \right) \frac{\partial P'}{\partial \eta} \tag{3.54}$$

$$G_2 = G_2^* + \left(C^W \frac{\partial x}{\partial \eta} - C^U \frac{\partial z}{\partial \eta} \right) \frac{\partial P'}{\partial \eta} + \left(B^W \frac{\partial x}{\partial \xi} - B^U \frac{\partial z}{\partial \xi} \right) \frac{\partial P'}{\partial \xi} \tag{3.55}$$

It has to be noted that the last terms of equations (3.54) and (3.55) are negligible if the grid system is nearly orthogonal. One may, therefore, further simplify the above expressions by neglecting these terms, and obtain the following

$$G_1 = G_1^* + B \frac{\partial P'}{\partial \xi} \tag{3.56}$$

$$G_2 = G_2^* + C \frac{\partial P'}{\partial \eta} \tag{3.57}$$

where

$$B = B^U \frac{\partial z}{\partial \eta} - B^w \frac{\partial x}{\partial \eta} \tag{3.58}$$

$$C = C^W \frac{\partial x}{\partial \eta} - C^U \frac{\partial z}{\partial \eta} \tag{3.59}$$

Recalling the continuity equation in Cartesian coordinate

$$\rho \frac{\partial U}{\partial x} + \rho \frac{\partial W}{\partial z} = 0 \tag{3.60}$$

Transforming it into ξ-η plane using equations (3.6) and (3.7) provides

$$\left(\frac{\partial z}{\partial \eta} \frac{\partial U}{\partial \xi} - \frac{\partial z}{\partial \xi} \frac{\partial U}{\partial \eta} \right) + \left(- \frac{\partial x}{\partial \eta} \frac{\partial W}{\partial \xi} + \frac{\partial x}{\partial \xi} \frac{\partial W}{\partial \eta} \right) = 0 \tag{3.61}$$

Integrating equation (3.61) over the CV and using the notations G_1 and G_2 as defined in equation (3.10) the following form of continuity equation is obtained

$$\left(\rho G_1 \Delta\eta\right)_e - \left(\rho G_1 \Delta\eta\right)_w + \left(\rho G_2 \Delta\xi\right)_n - \left(\rho G_2 \Delta\xi\right)_s = 0 \tag{3.62}$$

Combining this continuity equation and equation (3.56) provides

$$\left(\rho B \frac{\partial P'}{\partial \xi} \Delta\eta \right)_e - \left(\rho B \frac{\partial P'}{\partial \xi} \Delta\eta \right)_w + \left(\rho C \frac{\partial P'}{\partial \eta} \Delta\xi \right)_n - \left(\rho C \frac{\partial P'}{\partial \eta} \Delta\xi \right)_s + m_P = 0 \tag{3.63}$$

where

$$m_P = \left(\rho G_1^* \Delta\eta\right)_e - \left(\rho G_1^* \Delta\eta\right)_w + \left(\rho G_2^* \Delta\xi\right)_n - \left(\rho G_2^* \Delta\xi\right)_s \tag{3.64}$$

The pressure gradient appearing in equation (3.63) are approximated on the CV faces by second order central difference, for example

$$\left(\frac{\partial P'}{\partial \xi} \right)_e = \frac{P'_E - P'_P}{\Delta\xi} \tag{3.65}$$

Substituting this type of pressure gradient approximation into equations (3.63) and rearranging it into a standard form of algebraic equations provides the so called pressure correction equation:

$$A_P P_P' = A_E P_E' + A_W P_W' + A_N P_N' + A_S P_S' + b \tag{3.66}$$

where

$$A_E = \left(\frac{\rho B \Delta\eta}{\delta_\xi}\right)_e,$$

$$A_W = \left(\frac{\rho B \Delta\eta}{\delta_\xi}\right)_w,$$

$$A_N = \left(\frac{\rho C \Delta\xi}{\delta_\xi}\right)_n,$$

$$A_S = \left(\frac{\rho C \Delta\xi}{\delta_\xi}\right)_s,$$

$A_E = A_E + A_W + A_N + A_S$, and

$$b = m_P \tag{3.67}$$

In the present study, collocated grid arrangement in which all five dependent variables U, W, P, k and ε are solved on the same grid points has been adopted, as opposed to a staggered grid that is commonly used with Cartesian coordinates – (see Stathopoulos and Zhou 1993). Staggered grids cannot be applied to general coordinates effectively since the velocity components U and W are no longer related to the direction of constant ξ and η lines. The collocated grid arrangement has the advantage of easy book keeping since all

the variables are computed at the same grid points in addition to their suitability for use with general curvilinear coordinates.

It is well known that oscillation in pressure and velocity may be produced when the $2\Delta\xi$-central difference scheme is used for the computations of pressure gradients at the CV centers formed with collocated grids (Zhou 1995). In other words, the momentum equations discretized at the CV centers using $2\Delta\xi$-central difference schemes will be insensitive to pressure variations at $\Delta\xi$ scale. To avoid this problem also known as "checker board oscillation" associated with the use of collocated grids, Rhie and Chow's (1983) interpolation technique for pressure correction is adopted. It should be noted that the convective terms G_1 and G_2 are always located on the CV faces and are obtained by the interpolations between grid points at the CV center in this study, according to

$$G_1 = G_1^* + B\frac{\partial P'}{\partial \xi} \tag{3.68}$$

Even though G_1 is related to $\partial P'/\partial \xi$ in the form of the above equation, the value of G_1 on the face of the CV is actually obtained from the interpolation between the grid points at the center of the CV. Thus, the value of G_1 determined in this manner cannot detect the difference of pressure at $\Delta\xi$ scale on the faces of the CV. To detect the pressure at $\Delta\xi$ scale, Rhie and Chow (1983) suggested an effective form of interpolating convective terms (G_1 and G_2) at the faces of the CV's as given by equation (3.69). The effective form interpolates convective terms at the faces using second order difference scheme while interpolating the pressure gradient locally using first order difference. For example, the effective form at the east face is

$$G_{1e}^{*} = \overline{G_1^{*}} + \overline{B}\left(\frac{P_E^{*} - P_P^{*}}{\delta\xi} - \overline{\frac{\partial P^{*}}{\partial \xi}}\right) \tag{3.69}$$

where the over bar denotes the regular results obtained from linear interpolation between grid points E and P. Here the interpolated pressure gradients term $\overline{P_\xi^*}$ is based on the $2\Delta\xi$ -central differences on the face of the CV and has been eliminated from the convection term and replaced by $(P_E^* - P_P^*)/\partial\xi$. This procedure ensures strong velocity-pressure coupling. The boundary condition required to solve the pressure correction equation is that the normal derivatives of the pressure correction P' vanishes on the boundaries in the computational domain.

To summarize, the SIMPLEC algorithm consists of the following procedures

i. Start with the estimated velocity and pressure fields U^*, W^*, and P^*;

ii. Solve for the velocity fields using the momentum equations (3.17);

iii. Solve the other scalar transport equations, k and ε sequentially ;

iv. Solve the pressure correction equation (3.66);

v. Correct the velocity and pressure fields applying under-relaxation factors;

vi. Return to step ii and repeat the procedure until a converged solution is obtained.

The procedure of operations in a CFD that employs this SIMPLEC algorithm is also shown schematically in Figure 3.5. Details of the CFD tool development using object-oriented approach are discussed in the next section. Finally, a very large set of numbers will be obtained as a solution of the algebraic equations that consequently will be passed into a post processor for graphical presentation in order to assist in the interpretation of the results. For this purpose, commercial software named TECPLOT ® (version 9.2) has been used. Results and discussions are presented in Chapter 4.

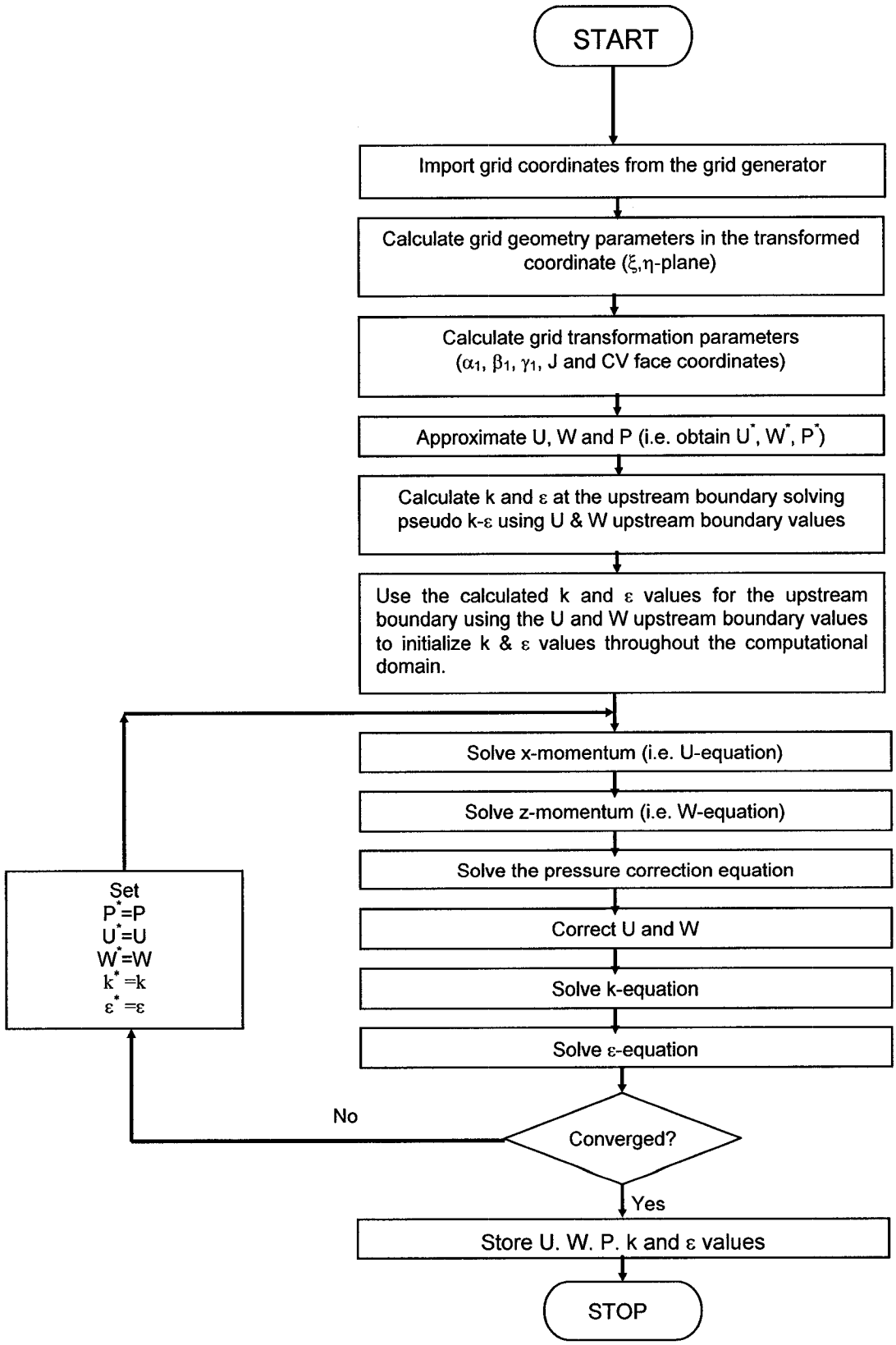


Figure 3.5 SIMPLEC solution procedures.

3.2 CFD tool development: object-oriented approach

3.2.1 Why object-oriented?

The present type of research spans over a long period of time and requires the effort of many researchers due to its complexity. Therefore, the selected paradigm should allow easy reusability (can be assembled from pre-written components with minimal efforts) and extensibility (the assembled system can be easily extended without having to modify significantly the reused components). Previous CFD programs such as TWIST (Turbulent WInd SimulaTion) developed by Baskaran (1990), Zhou (1995), and Li (1998) are written in a procedural programming language, usually FORTRAN, and consist of several thousand lines of procedural code containing many data structures accessed throughout the program. This type of global accessibility decreases the flexibility of the system. It is difficult to modify such existing codes and to extend or adapt it to new uses, models, and solution procedures. The inflexibility is demonstrated in several ways:

- a high degree of knowledge of the entire program is required to work on even a minor portion of the code,
- reuse of code is difficult,
- a small change in the data structures can affect the entire system,
- the numerous interdependences between the components of the design are hidden and difficult to establish.

In the present CFD tool development, an object-oriented methodology has been used to address these issues. The object-oriented paradigm provides fundamental concepts such as objects, classes, inheritance, and polymorphism. Software is organized into objects that store both data and operators on the data (i.e. member functions). This allows

abstraction of the essential properties of an object and those that will be used by other objects. The abstraction of the data into objects limits the knowledge of the system required to work on the code to only the object of interest and the services other objects provide to the objects of interest. This abstraction allows the details of the implementation of each object to be hidden from unintentional uses and allows access to the designed objects only, thus easing further modification as well management of the software. Encapsulating the data and operations together isolates the classes and prompts the reuse of the code.

Objects are instances described by a class definition. Classes are related by inheritance. A subclass inherits behavior through the attributes and operators of the superclass. Placing attributes common to several subclasses into the superclass, which is implemented once for all, enhance code reuse further. Changes to a class affect only the class in question. Polymorphism allows the same operation to behave differently in different classes and thus allows objects of one class to be used in place of those of another related class. Interdependences between the classes are explicitly laid out in the class interface. The number of dependences is minimized and easily determined. Object-oriented languages enforce the encapsulation of the classes. The language assures the integrity of the data structure.

In addition to these inherent advantages of the object-oriented approach, it is believed that object-oriented technology is important for parallel programming. There is a noticeable trend that large-scale numerical simulations in similar research work are heading in the direction of using parallel computers. Object-oriented technology in a parallel environment helps to hide the complexity of the programming from users,

thereby "facilitating" parallel programming. It also provides a modular structure background over which large computational tasks can be subdivided and distributed to different processors.

3.2.2 Design and Implementation

The procedure in object-oriented tool development includes detailed description of the problem statement (analysis), identifying potential classes from the problem statements, designing of each class and their member functions, and implementation of the design using a programming language that supports object-oriented approach such as C++.

3.2.2.1 Problem statement

The main objective here is the development of CFD tool that uses the wind flow governing equations described in previous section. The tool will discretize the computational domain into small control volumes using orthogonal grid meshes generated by the nearly orthogonal grid generator, which is described in section 3.3. Following this, it changes the governing PDE's into a set of non-linear algebraic equations at each control volume. Finally it solves the discretized equations for each of the dependent variables at each control volume sequentially and iteratively. For this purpose the following tasks are necessary:

- Separate the computational domain from the whole domain (Figure 3.6);
- Discretize the computational domain into small control volumes;
- Calculate appropriate geometric transformation parameters;
- Integrate the governing equations over the control volumes and apply finite difference techniques to obtain sets of algebraic equations;
- Estimate initial conditions;

- For each dependent variable;
  - Obtain the pressure gradients;
  - Set gamma (dynamic viscosity plus turbulent viscosity);
  - Set convection terms;
  - Set flow terms;
  - Set diffusion terms;
- Calculate the east, west, north and south coefficients at each control volume for each dependent variable;
- Calculate the sources at each control volume for each dependent variable;
- Apply boundary conditions;
- Under relax dependent variables;
- Iteratively solve the algebraic equations for each of the dependent variables sequentially;
- Store the results;
- Format the results for post processing.

After examining the problem statements, two alternatives are considered to define different classes and their relationships. The first alternative is to base the classes and their relationships on physical or geometrical entities that exist in the problem, while the other option is to base the relationship on mathematical entities. The latter has been used by Weller et al. (1998) for a general purpose CFD code developed for use in different areas of specialization. In their approach, mathematical objects such as divergence, laplacian operator and different tensorial notations were designed as objects. This approach might be suitable for very large general-purpose application software. For

specialized applications as in the case of the present study, however, this thesis supports an approach in which the designed objects expose useful geometrical relationships that exist in the problem domain as well as the physics rather than the mathematics. After all the purpose of object-oriented design philosophy is to make the system as real and apparent as possible. Also, if parallel implementation is required, the physical and geometrical entities can be used conveniently as a base to divide the tasks and submit them to different processing units. As a result physical/geometrical relationships have been given preference over the mathematical relationships. Examining the problem carefully helps to identify physical/geometrical objects such as control volume and computational domain. More precisely CompDomain and ControlVolume are the two class names used. The geometrical meaning of these classes is shown in Figure 3.6.

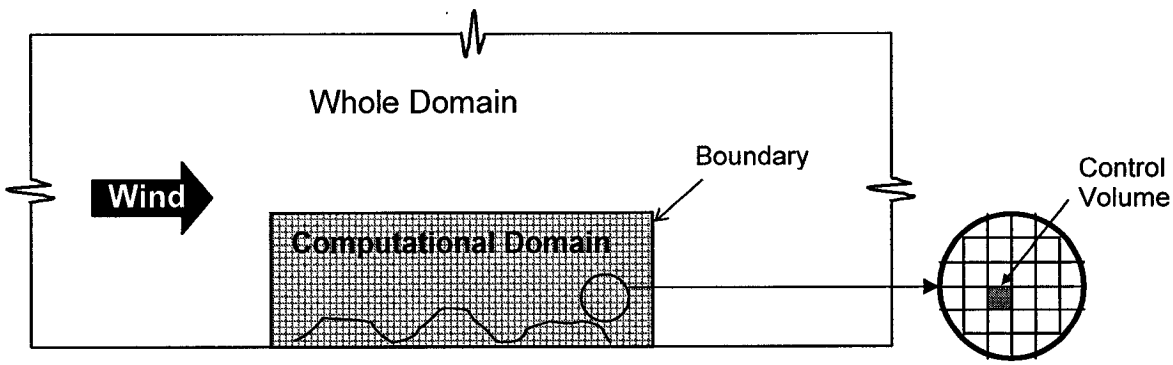


Figure 3.6 Descriptions of Whole Domain, Computational Domain, and Control Volume.

The relationship between the two classes is "CompDomain has a number of ControlVolumes". An entity relationship diagram is shown in Figure 3.7 for the two classes. Entity relationship diagrams show the relationships among classes in the design. Classes are shown as boxes around the class name. The links between the boxes establish the relationship between the classes. Links are labeled with the type of association that exists between the classes. In the present case, ControlVolume is part of CompDomain.

Thus, the classes ControlVolume and CompDomain are linked together with the "part of" aggregation. Links that end in a diamond shape represent aggregation, which describes a relationship in which one object is "part of" another in an assembly. In fact, many ControlVolume are part of a CompDomain. A darkened circle at the end of the link represents this multiplicity, i.e. of more than one. Links with no special ending represent a multiplicity of exactly one.

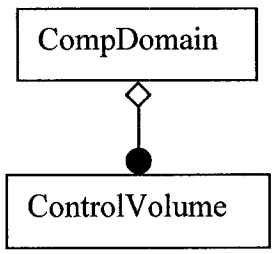


Figure 3.7 Entity relationship diagram

3.2.2.2 Interface definition of classes

Once the relationships the object has to the rest of the system are established, the specific services the object must provide can be defined. The Interface Definition Table provides a formal description of the publicly available member functions the object provides, including the argument and return types. Interface definitions for CompDomain and ControlVolume classes are given in Tables 3.2 and 3.3 respectively. The arguments and return types are given as a non-language specific description of the object type that is required. It is intended to be used directly to implement the design in a programming language.

3.2.2.3 Implementation in C++

Once the object design is complete, the next task is implementation in a computer language. The natural choice for an object-oriented design is an object-oriented

programming language. C++ is the most widely used object-oriented programming language for a wide range of applications. Such a large base of users has led to many high quality compilers and different types of class libraries. Thus C++ was chosen for the present work. The objects described, using the design formalisms from the previous section, lend themselves to direct implementation as classes in C++. The Entity Relationship Diagram gives the dependencies the class has with other classes. The Interface Definition Table defines the member functions the class must provide. What remains for the implementation is identifying the specific C++ data types used as the arguments and return values of the member functions, as well as the data types used for the class instance variables.

The implementation of the object design requires the use of specific C++ classes to serve as the arguments and return values of the methods. The classes in the present study are designed from scratch. Intrinsic C++ types, such as Boolean, ints (integers) and doubles (real numbers) have been used. Often a class makes use of data structures such as Arrays. An array is a fixed-length collection of objects of the same type. Value access is through an integer index. Individual members of the collection are accessed directly. Access is rapid, but the number of members in the collection must be known ahead of time and this is not a desirable feature for large programs. As a result in the present study, a storage that can store objects or any other data types using dynamic memory allocation has been designed and implemented by using pointers and "new" commands available in C++ programming language. The complete C++ code for the implemented object design consists of over 10,000 lines. Obviously it is far too bulky to be presented in this document. The C++ interfaces for the classes are given in Appendix A.

Last but not least, high level of documentation at each level of computer code development has been given major emphasis for sake of better clarity/efficiency either during the present research, future research or operation stage. As a result, a significant percentage of the whole source code represents comments either defining variables, or explaining the task of each of member functions or objects.

Table 3.2 CompDomain class interface definition

Member functions	Arguments	Returns	Purpose
Construction	Size of CD and number of grid points		Creates an instance of a specific type of CD
SetGrid		Coordinates of CD grid point coordinates	Discretize the CD into small CVs
SetGridTransform Parameters		Parameters used in transforming the physical domain to CD	Provides appropriate geometric transformation parameters
InitializeDepVar	Type of dependent variable	Estimated U, W, P, k and ε values	Provides initial estimated values for each of the dependent variables during the first iteration
SetPressCorrParas		Pressure correction parameters	Provides appropriate geometric transformation parameters
SetPressureGradients		Pressure gradient at grid point and face of CV	Provides the pressure gradients
SetConvectiveTerms			Provides convection terms
SetFlow		Flow parameters	Provides flow terms
SetDiffusion		Diffusion parameters	Provides diffusion terms
SetSources	Type of dependent variable	Source terms	Provides the sources at each CV for each dependent variable

SetTurbulentVisc	Type of dependent variable	Turbulent viscosity	Provides turbulent Viscosity
SetGamma	Type of dependent variable	Sum of dynamic viscosity and turbulent viscosity	Provides gamma which is a sum of dynamic viscosity and turbulent viscosity
SetCoefficients	Type of dependent variable	East, west, north and south coefficients at each CV for U, W, P, k and ε	Provides the east, west, north and south coefficients at each CV for each dependent variable
SolverManager	Type of dependent variable		Selects 1D, 2D or 3D solvers
Solve			Solves 1D, 2D or 3D system of linear equations
CorrectUand W		Corrected U and W after each iteration	Corrects velocity field
CorrectCovectiveCoef		Corrected Convective terms after each iteration	Corrects convective coefficient terms
Plot		Inputs to TECPLOT	Formats the final results for TECPLOT
Output			Setup output of desired results
CalculateSpeedUp		Speed up	Extracts FSUR values from the velocity field
ConvergCri			Provides the convergence criteria
IntializeFromFile	File names		Provides option of initializing dependent variables from files (pre-computed results)

Table 3.3. ControlVolume class interface definition.

Member functions	Arguments	Returns	Purpose
Construction			Creates an instance of a specific type of CD
DefaultValues		Default values of member variables	Assigns default values to member variables, eg. g=9.81 m/sec^2
EastFace	East coefficient value		Assigns known east coefficient value
EastFace	Flow and diffusion variables		Calculates east coefficient value
WestFace	West coefficient value		Assigns known west coefficient value
WestFace	Flow and diffusion variables		Calculates west coefficient value
NorthFace	North coefficient value		Assigns known north coefficient value
NorthFace	Flow and diffusion variables		Calculates north coefficient value
SouthFace	South coefficient value		Assigns known south coefficient value
SouthFace	Flow and diffusion variables		Calculates south coefficient value
CenterPoint			Calculates the center point coefficient
SourceTerm	Constant term value in the in the discretized equation		Assigns a known value to the constant term in the discretized equation

SourceTerm	Type of dependent variable, constant source value, variable source value	Constant term value in the linear equation	Calculates constant term value in the discretized equation
SourceTerm	Dependent variable type	East, west, north and south coefficients at each CV for U, W, P, k and ε	Provides the east, west, north and south coefficients at each CV for each dependent variable
CenterPointStore	Center point coefficient value		Assigns known center point coefficient value
CenterPoint	Difference of flow at faces of the CVs		Adds the argument to the center point coefficient
UnderRelax	Dependent variable type and value		Under relaxes the dependent variables
DownStrBound	Dependent variable type and its value at the downstream boundary		Assigns downstream boundary value
UpStrBound	Dependent variable type and its value at the upstream boundary		Assigns upstream boundary value
GroundBound	Dependent variable type and its value at the ground boundary		Assigns ground boundary value
NextToGround	Dependent variable type and its value at the grid point next to the ground		Assigns values at the first grid points next to the ground
UnderRelax	Dependent variable type and its value		Under relaxes the dependent variable
PrintCV			Prints CV values such as coefficients during

	run time
Ae	Returns value of east coefficient
aw	Returns value of west coefficient
an	Returns value of north coefficient
as	Returns value of south coefficient
ap	Returns value of center point coefficient
apu	Returns value of center point coefficient for U-equation
apw	Returns value of center point coefficient for W-equation
con	Returns value of constant term in the discretized equation
conu	Returns value of constant term in the discretized U-equation
conw	Returns value of constant term in the discretized W-equation

3.3 Nearly orthogonal grid generation

One of the first steps in numerical evaluation of wind flow is generation of a good quality grid. The physical domain must be covered with a mesh so that discrete control volumes, where the conservation laws are applied, are identified. The success of a numerical wind flow evaluation is dependent on grid characteristics such as orthogonality, smoothness, skewness and spatial distribution. As a result one of the objectives of the present study has been the generation of a good quality grid that takes into account the aforementioned characteristics.

Two alternative grid-generating systems have been considered: Cartesian coordinate system that approximates the terrain geometry by staircase type of approximation and body-fitted curvilinear coordinate system adjusted to the terrain geometry. In the present study the most accurate body-fitted coordinate system has been adopted since the staircase type approximation of terrain geometry would have introduced some numerical errors. The transformed governing equations however appear longer and more complicated in the ξ-η coordinate system.

In the following section, terrain geometries used in the numerical simulation are described. The differential equations used to generate the physical location of orthogonal grids are discussed, their discretized forms are derived and the solution procedures are elaborated. The grid density control mechanism is highlighted. The new features of the nearly orthogonal grid-generating tool are described.

3.3.1 Description of terrain geometries

Simulation of wind flow over a wide spectrum of representative terrain geometries is of main interest in this study. Representative terrain geometries including escarpments, single and multiple hills and valleys are shown in Figure 3.8. Normally in BLWT studies, the shapes of valleys and hills are approximated by Jackson or Cosine function, given by equations (3.70) and (3.71) respectively, whereas escarpments and embankments are represented by their slope. The dimensions are scaled-downed from the actual topographic element they represent according to the dynamic similarity to the actual flow field. Similar strategy is adopted in this study so as to validate the computational results by comparing them with their BLWT counterparts.

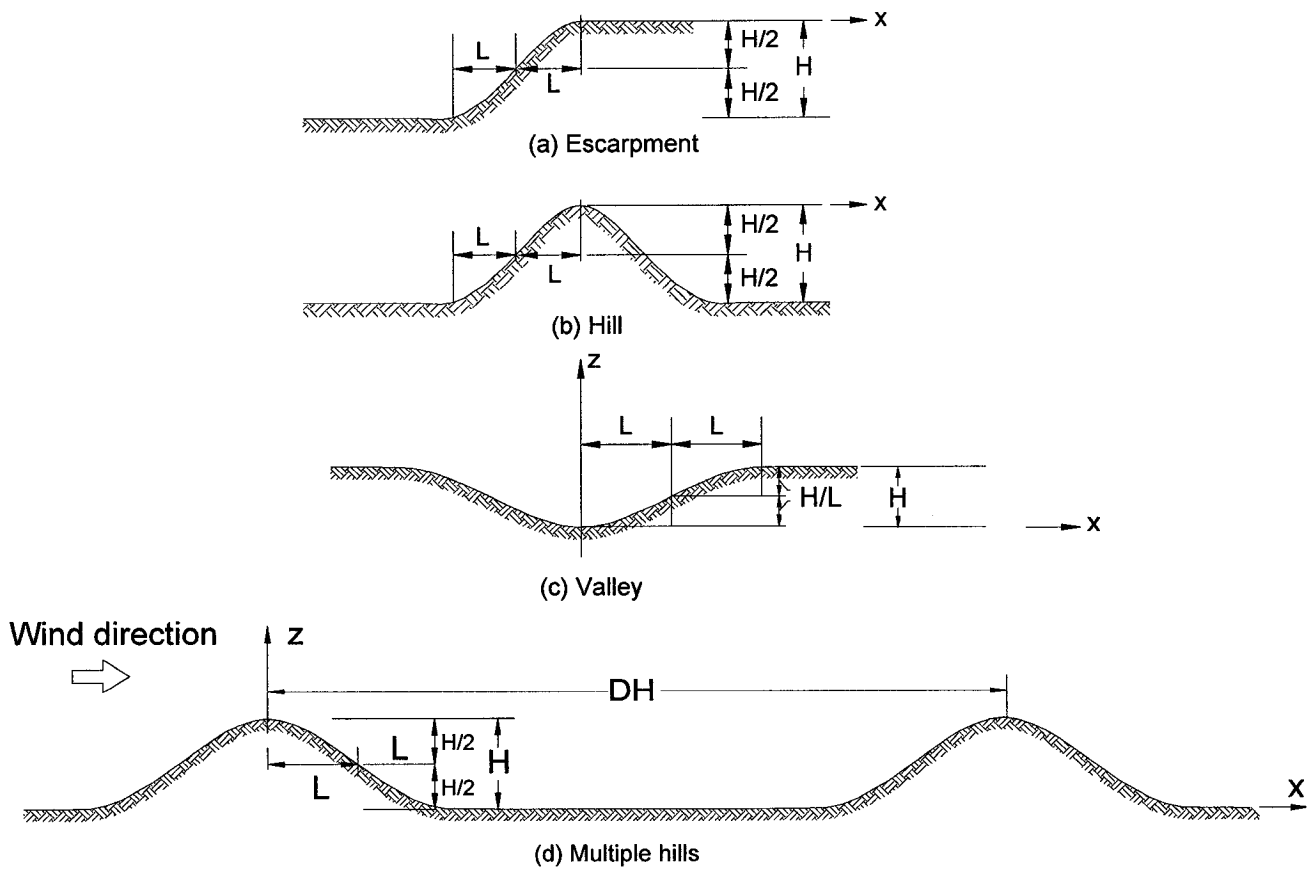


Figure 3.8 Description of terrain geometries used in the present study.

$$z = \frac{H}{1+(x/L)^2} \tag{3.70}$$

$$z = H\left(\frac{\cos(\pi x)}{4L}\right)^2 \tag{3.71}$$

where x and z represents the x- and z-coordinates of the terrain cross-section in the physical domain respectively.

3.3.2 Required capability of the grid generator system

At the outset the following goals are targeted while developing the nearly orthogonal grid generator tool:

- Robustness;
- Preventing grids from collapsing on one another;
- Ability to calculate the distortion functions directly from their definitions numerically as opposed to using constant values or using approximate interpolations that may not ensure generation of orthogonal grids;
- Ability to control the spatial grid distribution over the computational domain easily.

These goals have been set after examining the limitations of the grid generation systems used in the area of computational wind engineering.

3.3.3 The grid generation system

Systems of partial equations are solved with appropriate orthogonal boundary conditions to generate the physical location of each grid points. For the present study, not only curvilinear coordinates fitted to the physical domain are used but also the grid points form a nearly orthogonal mesh. With non-orthogonal grids the coefficients multiplying

mixed derivatives, i.e. β_1 and γ_1 in equation (3.11), can be larger than the diagonal coefficients in the discretized equations, which can lead to poor convergence and oscillations in the solution. The physical location of each grid point is obtained as a solution of the following set of partial differential equations:

$$\frac{\partial}{\partial\xi}\left(f\frac{\partial x}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\frac{1}{f}\frac{\partial x}{\partial\eta}\right)=0 \tag{3.72}$$

$$\frac{\partial}{\partial\xi}\left(f\frac{\partial z}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\frac{1}{f}\frac{\partial z}{\partial\eta}\right)=0 \tag{3.73}$$

where f represents the distortion function given as the ratio of scale factors in the η-direction to that in the ξ-direction. In a similar previous study, Bergeles (1985) approximated the shape function by a constant value. The constant distortion function was taken as the ratio of the size of the computational domain in vertical direction over the size of the computational domain in the horizontal direction. This approach limited the robustness and in some cases did not ensure orthogonality. It also makes the system sensitive to slight geometry changes. Theodoropoulos (1985) calculated the distortion function from predefined grid distributions at the boundary of computational domain and further interpolated the interior computational domain from the calculated shape function values at the boundary following the suggestion by Ryskin and Leal (1983). Preliminary numerical experimentation with this type of approach resulted in large skewness and collapsing of grids for some geometries with large slopes and large number of grids. Similar problems have also been documented by Ackelic et al. (2001).

To avoid these problems, the distortion function throughout the computational domain has been calculated numerically directly from its definition during nonlinear iterations as

initially suggested by Eca (1996). This is as opposed to using constant distortion functions throughout the computational domain or using interpolation function approximations. Thus

$$f = \frac{h(\eta)}{h(\xi)} \tag{3.74}$$

where the scale factors are defined by

$$h(\eta) = \sqrt{\left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial z}{\partial \eta}\right)^2} \tag{3.75}$$

$$h(\xi) = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\right)^2} \tag{3.76}$$

Moreover, the new grid generation tool has implemented additional functions, suggested by Ackelic et al. (2001), which prevent scale factors from approaching zero or excessive values in some part of the computational domain that would otherwise cause the grids to collapse on one another. The last two terms in equations (3.77) and (3.76) are the newly introduced functions that control the scale factors in both ξ- and η-directions. Therefore the two –dimensional grid generating system with the added new functions R and Q takes the following form:

$$\frac{\partial}{\partial \xi}\left(f\frac{\partial x}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{1}{f}\frac{\partial x}{\partial \eta}\right) + R(h(\xi)) + Q(h(\eta)) = 0 \tag{3.77}$$

$$\frac{\partial}{\partial \xi}\left(f\frac{\partial z}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{1}{f}\frac{\partial z}{\partial \eta}\right) + R(h(\xi)) + Q(h(\eta)) = 0 \tag{3.78}$$

The functions R and Q are defined by

$$R(h(\xi)) = c\left(h(\xi) - \frac{\overline{h(\xi)}^2}{h(\xi)} \right)$$

$$Q(h(\eta)) = c\left(h(\eta) - \frac{\overline{h(\eta)}^2}{h(\eta)} \right) \tag{3.79}$$

where c is the so-called force constant. By changing its magnitude, it is possible to change the intensities of R and Q. The mean scale factors $\overline{h(\xi)}$ and $\overline{h(\eta)}$ are defined as

$$\overline{h(\xi)} = \left. \frac{\int h(\xi) d\xi}{\int d\xi} \right|_{\eta=\text{const.}} \tag{3.80}$$

$$\overline{h(\eta)} = \left. \frac{\int h(\eta) d\eta}{\int d\eta} \right|_{\xi=\text{const.}} \tag{3.81}$$

Here R and Q are homogeneous source terms that alter the solution (x, z) in such a way as to control favorably the scale factors $h(\xi)$ and $h(\eta)$.

3.3.4 The discretized grid generating system

The integration of equations (3.77) and (3.78) over a control volume as shown in Figure 3.9 on the ξ- and η-plane and applying the finite difference approximations yields

$$A_P \Phi_P = A_E \Phi_E + A_W \Phi_W + A_N \Phi_N + A_S \Phi_S \tag{3.82}$$

where the coefficients are calculated, using the notation shown in Figure 3.9, as follows.

$$A_E = \frac{f_e \Delta\eta}{\Delta\xi_{EP}},$$

$$A_W = \frac{f_w \Delta\eta}{\Delta\xi_{PW}},$$

$$A_N = \frac{f_n \Delta\xi}{\Delta\xi_{NP}},$$

$$A_S = \frac{f_s \Delta\xi}{\Delta\xi_{PS}},$$

$S_C = P\Delta\xi\Delta\eta + Q\Delta\xi\Delta\eta$, and

$$A_P = A_E + A_W + A_N + A_S + S_C \tag{3.83}$$

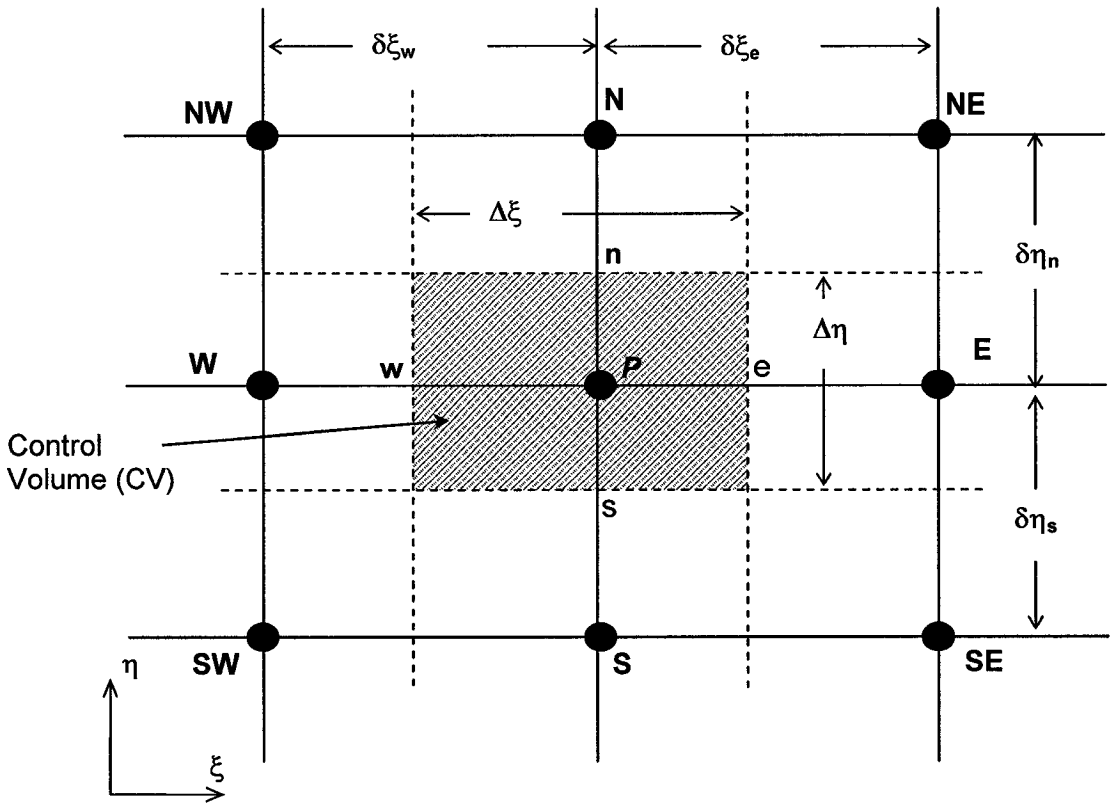


Figure 3.9 Control Volume and its parameter definitions

At each one of the four faces of the control volume the distortion functions are approximated by the following finite difference approximations using the notations of Figure 3.9.

$$f_e = \frac{\delta\xi_e}{2\Delta\eta}\sqrt{\frac{\left((x_{NE}+x_N)-(x_S+x_{SE})\right)^2+\left((z_{NE}+z_N)-(z_S+z_{SE})\right)^2}{(x_E-x_P)^2+(z_E-z_P)^2}}$$

$$f_w = \frac{\delta\xi_w}{2\Delta\eta}\sqrt{\frac{\left((x_{NW}+x_N)-(x_S+x_{SW})\right)^2+\left((z_{NW}+z_N)-(z_S+z_{SW})\right)^2}{(x_P-x_w)^2+(z_P-z_W)^2}}$$

$$f_n = \frac{2\Delta\xi}{2\delta\eta_n}\sqrt{\frac{(x_N-x_P)^2+(z_N-z_P)^2}{(x_{NE}+x_E-x_{NW}+x_W)^2+(z_{NE}+z_E-z_{NW}-z_W)^2}}$$

$$f_s = \frac{2\Delta\xi}{2\delta\eta_n}\sqrt{\frac{(x_P-x_S)^2+(z_P-z_S)^2}{(x_{SE}+x_E-x_W-x_{SW})^2+(z_{SE}+z_E-z_{SW}-z_W)^2}} \qquad (3.84)$$

Scale factors h_ξ and h_η are discretized similarly. For example the scale factors at the east face of the control volume will look like

$$h_{\xi_e} = \frac{1}{\Delta\xi}\sqrt{(x_E-x_P)^2+(z_E-z_P)^2}$$

$$h_{\eta_e} = \frac{1}{\Delta\eta}\sqrt{(x_E-x_P)^2+(z_E-z_P)^2} \qquad (3.85)$$

The average scale terms in the functions P and Q are obtained as follows

$$\overline{h_{\xi i}} = \frac{1}{M-1}\sum_{i=1}^{M-1} h_{\xi_{i,j}}^{i,i+1}$$

$$\overline{h_{\eta j}} = \frac{1}{N-1}\sum_{j=1}^{N-1} h_{\eta_{i,j}}^{j,j+1} \qquad (3.86)$$

where M and N are number of grid lines in the ξ- and the η-directions, respectively.

3.3.5 Solution procedure

The systems of equations defined in equations (3.82) are highly non-linear and as a result, the following iterative algorithm has been used to solve them:

i. Choose four corner points of the physical domain that serve as the corner points of the grid in computational domain. Get the $x(\xi,\eta)$, $z(\xi,\eta)$ values of the boundary grid points by dividing physical boundaries into equal segments.

ii. Get the $x(\xi,\eta)$, $z(\xi,\eta)$ values at the interior grid points by bilinear interpolation.

iii. Get the distortion function $f(\xi,\eta)$ from equations (3.74).

iv. Substituting these $f(\xi,\eta)$, solve equation (3.82) to get new $x(\xi,\eta)$, $z(\xi,\eta)$. Equation (3.82) is solved using a few iterations sequentially for $x(\xi,\eta)$, $z(\xi,\eta)$. Get R and Q from equations (3.79).

v. Adjust boundary conditions. If Dirichlet boundaries are applied nothing is done. For sliding (Neuman-Direchlet) boundaries, relocate boundary nodes to satisfy orthogonality.

vi. Go to step iii if convergence criteria on orthogonality and aspect ratio are not satisfied.

3.3.6 Grid generation illustration

As an illustration, an orthogonal grid has been generated for a half-hill geometry and has been plotted using TECPLOT® (TEPLOT 2000) as shown in Figure 3.10. The grid of Figure 3.10 has been generated by using the following boundary conditions: sliding and orthogonal boundary condition at the upstream, downstream and top boundaries; non-sliding and orthogonal at the ground boundary. Sliding boundary condition is the case in

which the grid points are allowed to slide on the respective boundary. By symmetrical reflection of the half-hill grid system, a grid system for a full-hill is obtained. By repeating this process, grid system for double or triple hills can be obtained as required. When non-orthogonal boundary condition both at the upstream, downstream and top boundaries together with constant distortion, representing example of application of improper boundary condition and distortion function, are used it resulted a typical case of collapsing grids as shown in Figure 3.11.

The grid generation tool also has the utility that can extend the computational domain in the required direction. For example larger area is usually required downstream the hill in order to incorporate the recirculation area and to allow the wind flows to fully develop after the disturbance by the hill. Hence the "extend" utility developed is used to enlarge the computational domain in the downstream direction. In addition, it has a utility to populate the grid when more grid points are required. Of course this could have also been done by assigning the required large number of nodes while generating the grid initially. However the populating utility is advantageous since it does not have to go through solving the algebraic equation iteratively thus reducing computation time. Figure 3.12 shows orthogonal grid generated for a half grid mirror-reflected to form a full single hill, populated throughout the computational domain to double the number of grid points both in the vertical and horizontal direction, extended in the downstream direction and clustered using a utility explained below, and drawn using TECPLOT® (TECPLOT 2000).

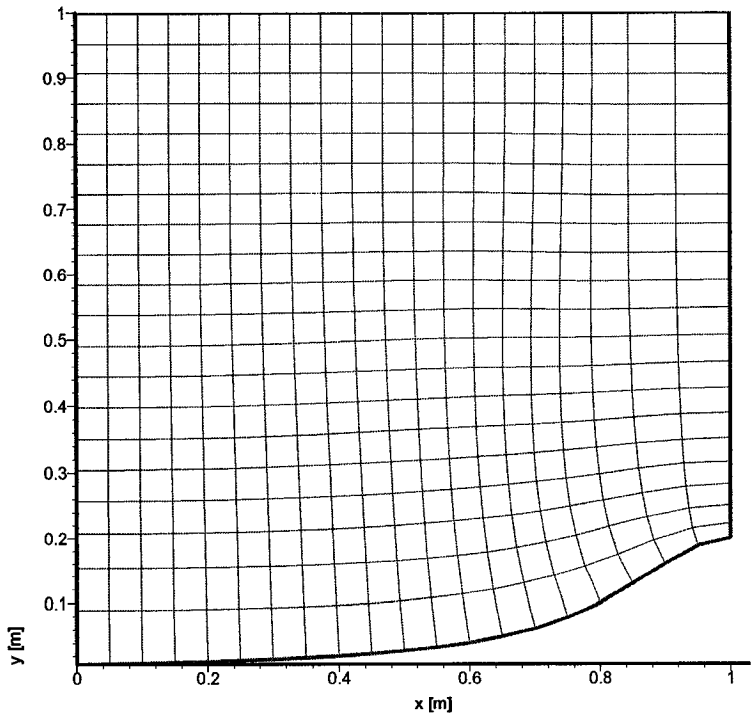


Figure 3.10 Orthogonal grid.

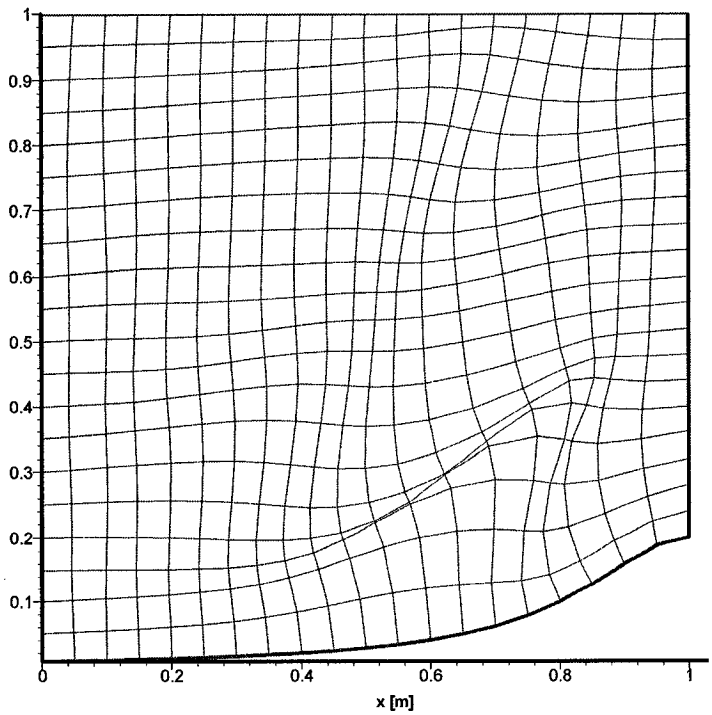


Figure 3.11 Example of inadequate grid.

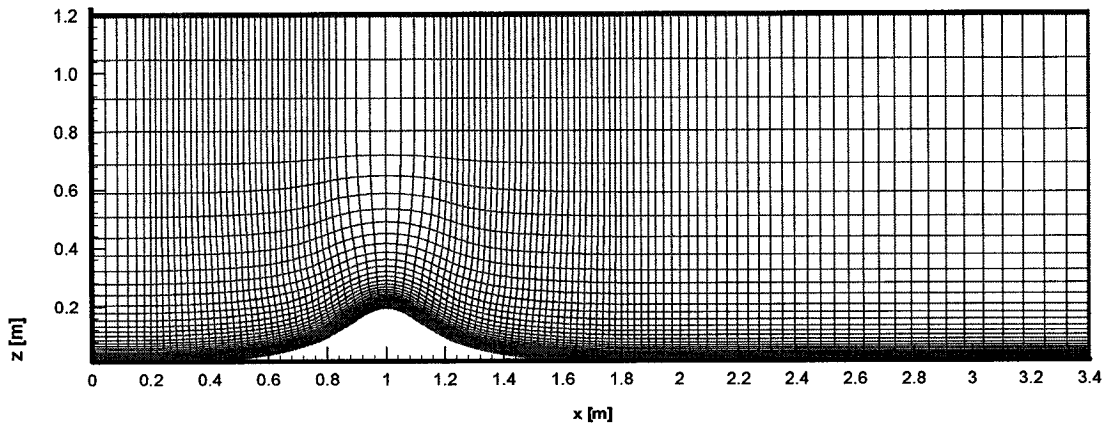


Figure 3.12 Reflected, populated, extended and clustered mesh.

3.3.7 Grid spatial distribution adjustment

Solving the partial differential equations provides uniformly spaced grids as shown in Figure 3.10. In order to alter the spatial distribution to any desired spatial grid distribution dictated by the physics of the flow, the method discussed below has been implemented. For example, dense grids suitable for capturing the steep gradients of the dependent variables closer to the ground can be produced in the following manner. Initially the first grid closer to the ground is moved to a new location along the vertical line connecting the grids based on the requirements of log-law of the wall. Subsequently, the remaining grid points are redistributed along their respective vertical lines using an exponential recursive function given by equation (3.87). The recursive function basically generates chord lengths of the relocated grid points that are along the vertical line (i.e. constant ξ - line). Their x- and z-coordinates are yet to be determined.

$$S_j = \begin{cases} 0 & \text{if } j = 1 \\ S_{j-1} + S_{min}(1+d)^{j-1} & 2 < j \le L \end{cases} \tag{3.87}$$

where S_j is arc length up to the j^{th} grid point along constant ξ - line and S_{min} is assigned a value equal to the distance of the first grid point from the local ground surface(i.e. z_P) that is determined from the log-law requirement. L represents the number of grid points in vertical direction while d represents a parameter so determined (by using Newton-Raphson method) to satisfy conditions S_{min} is equal to z_p and S_L is equal to the total curve length of the vertical line that is determined using the grid points before relocation. Note that relocation of the grid points will not alter the total length of the curve.

At this stage the chord length corresponding to the new relocated grid points has been found, what remains is to find their x and y coordinates. In order to find the coordinates of the new relocated grid points, lines connecting the grids in the vertical direction are parametrically represented by Non-Uniform Rational B-spline curves (NURB). Generally NURB curve is defined by control points, curve's order, and curve parameter. The curve defined by the NURB is a series of polynomials, the parameters determining where in the parameter range these polynomials start and stop. These parameters are called "knots" because the values "tie together" the polynomials. The collection of knots for a particular curve is called the knot vector. The values in the knot vector must always be in ascending order. Mathematically the NURBS curve is given by:

$$\vec{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \vec{P}_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{3.88}$$

where $\vec{P}_i$ are the x- and z-coordinates of control point i, w_i is the corresponding weight, $N_{i,p}$ is the p^{th} degree B-spline basis function, and $\vec{C}(u_i)$ are the x- and z-coordinates of

point i on the curve, which corresponds to u_i, the i^{th} element of the knot vector $\bar{u}$. The B-spline basis functions are defined using the following recursive formula:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i < u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u_i}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.89}$$

The basis functions $N_{i,p}$ vanish everywhere except in the vicinity of point i, where the size of this vicinity depends on the order p. The knots can be considered as division points that subdivide the interval $[u_0, u_m]$ into knot spans. All B-spline basis functions are supposed to have their domain on $[u_0, u_m]$. In the present study, the u's are determined using the chord length method calculated by using the recursive equation (3.86) described earlier. Thus the vertical lines can be represented as:

$$N\vec{P}_i = \vec{C}(u_i) \tag{3.90}$$

where N is the NURBS coefficients, $\vec{C}(u_i)$ represents the required grid points after relocation and vector P_i is the set of control points that are determined using a relationship similar to equation (3.90) using grid points before relocation.

4 Results of numerical simulations and discussion

Wind flow modified by topography has been evaluated using the developed CFD tool described in Chapter 3. For this purpose representative geometries are selected to cover a wide spectrum of terrain conditions. These include steep escarpment, shallow and steep isolated hills, shallow and steep multiple hills, and shallow and deep valleys. To validate or highlight the advantage and/or disadvantages of the currently developed numerical method, detailed comparisons are made with results obtained from five other methods: field measurements, BLWT experiments, analytical methods, NBCC provisions (1995), and existing CFD simulations. For the purpose of validating the proposed CFD tool, the specific dimensions of the terrain geometries used in the present CFD simulations are chosen in most cases to be similar to those used in the BLWT experiments reported in literature. In BLWT experiments, boundary conditions are generally better controlled, thus producing reliable results widely acceptable by the engineering community at large.

In the following sections, details of the numerical simulation of wind flow over representative terrain geometries are discussed arranged according to the type of terrain. The numerical set up is defined. Extensive validation of the numerical evaluation carried out during the present study is presented as well. Improvements on numerical results due to the fine-tuning of parameter applications such as ground surface roughness and grid quality are discussed.

4.1 Numerical setup

Prior to numerical simulation of wind flow over a specific type of terrain geometry, a common numerical setup is defined for all types of terrain in terms of computational

domain size, convergence criteria and hardware requirement. The following sections discuss each one of these issues separately.

4.1.1 Computational domain size

A preliminary numerical experimentation has been conducted to select the size of the computational domain, i.e. height, upstream length and downstream length, so that the location of the boundaries has minimum influence on the numerical results. To study the effect of the computational domain height on the flow field for instance, simulations are carried out for low and high computational domain size with height equal to 6H and 12H respectively, where H represents the height of the hill under consideration. The numerical results do not show significant difference both in terms of FSUR values above the crest of hill or mean wind flow velocity field throughout the computational domain as shown in Figures 4.1 and 4.2 respectively. This can be explained by examining the upstream velocity profiles used in the numerical evaluation, which have the following characteristics: closer to the ground the velocity increases with height and above a height (called gradient height) the velocity does not vary further with increase in height. The top boundary is set higher than the gradient height for both low and high computational domains, encompassing the boundary layer thickness with a good clearance. This is the reason why similar numerical results have been obtained for both cases.

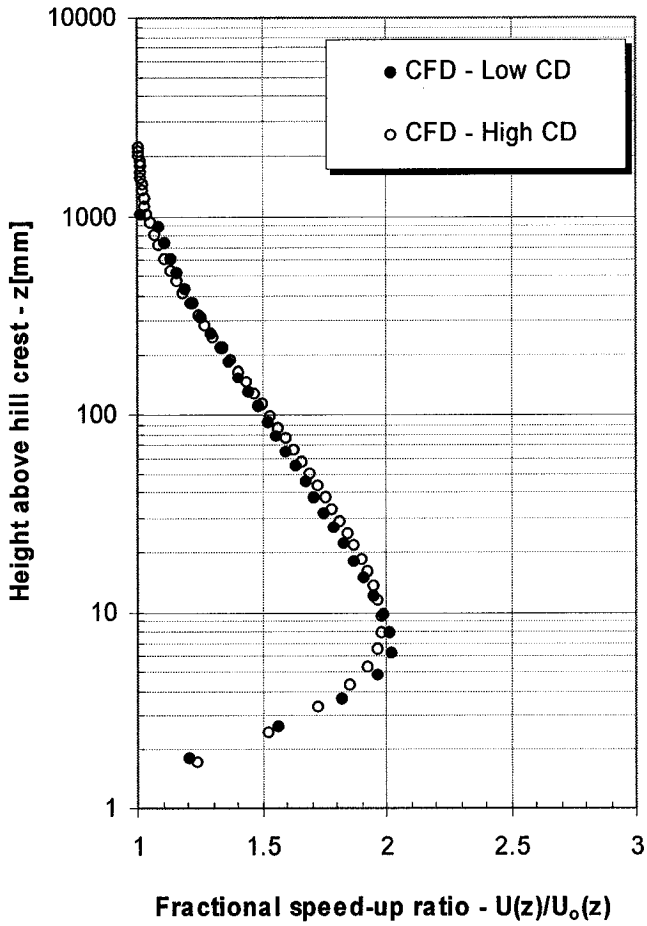
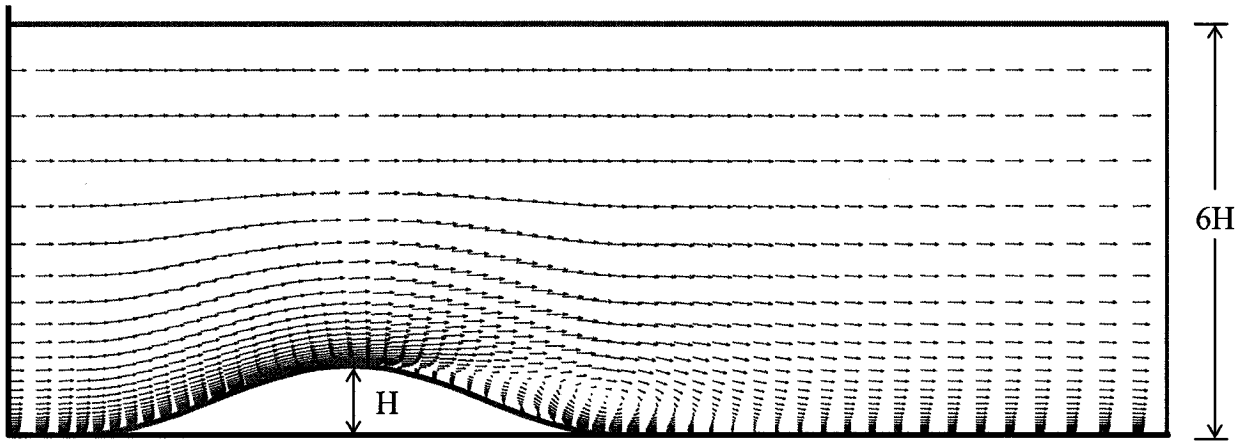
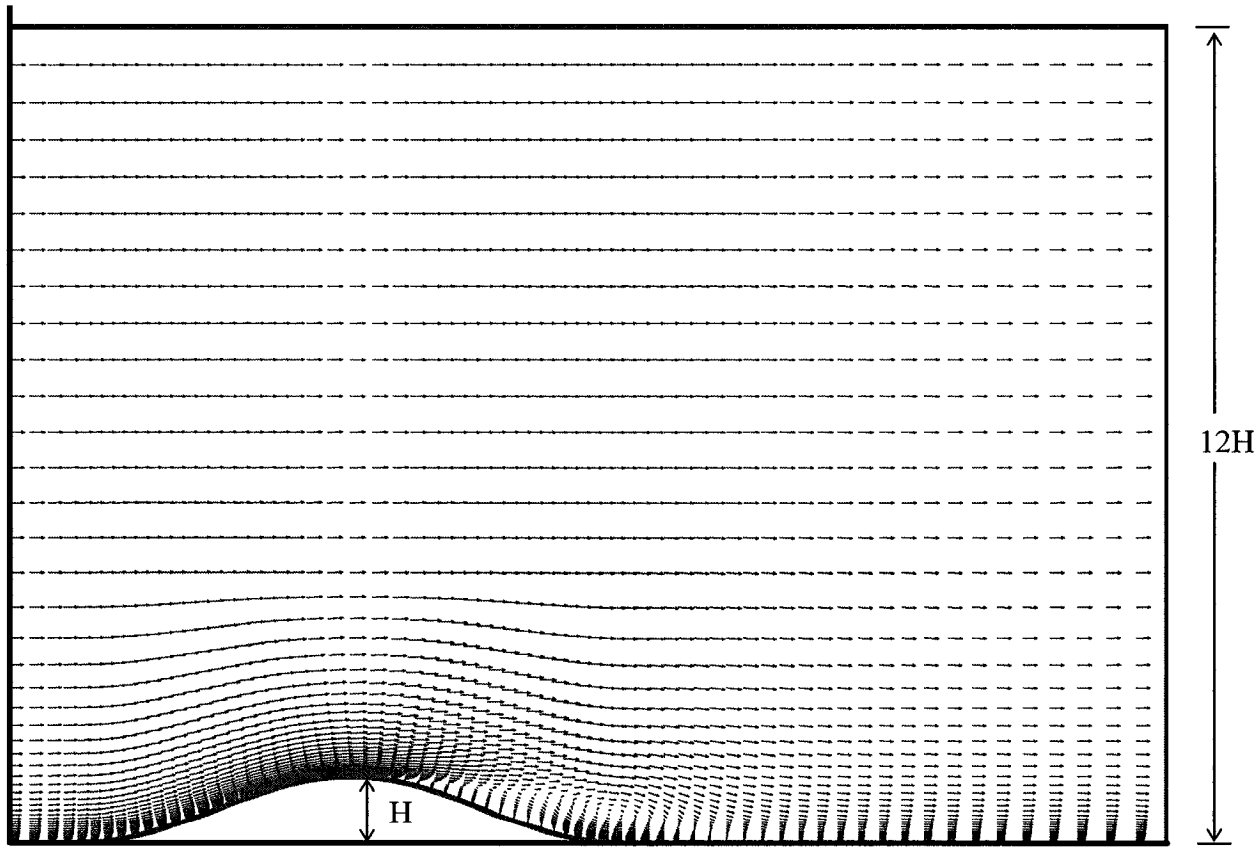


Figure 4.1 Comparison between FSUR values obtained by using low and high computational domains.



(a) Low computational domain



(b) High computational domain

Figure 4.2 Wind velocity vector fields obtained by using (a) low and (b) high computational domain.

To reduce computational time, therefore, the smaller computational domain size (i.e. 6H) is used in all the present CFD evaluations unless otherwise mentioned. The upstream and downstream lengths of the computational domain have also been selected in a similar manner as 5H and 12H respectively. Figure 4.3 shows the size of the computational domain used in the present study. For the case of multiple hills the upstream length (5H) is measured from the crest of the first upstream hill and the downstream length (12H) is measured from the crest of the last downstream hill. It may be noted that for high speed flows, use of large CD size is necessary.

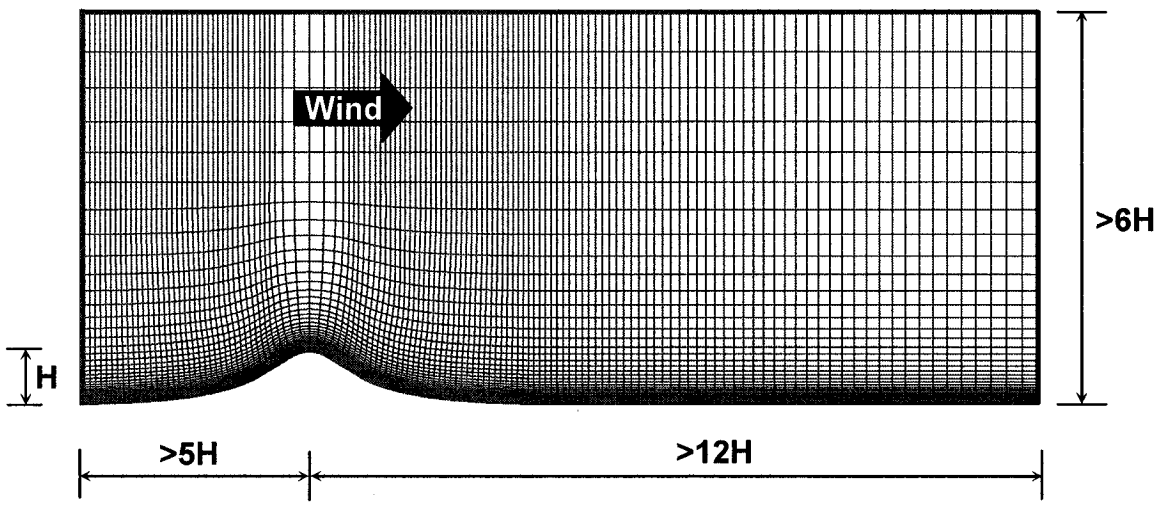


Figure 4.3 Size of the computational domain.

4.1.2 Convergence criteria

Relative residual convergence criteria as discussed in section 3.1.5 have been used in all numerical evaluations of the present study. Typical plots showing the decay of the relative residuals as a function of the number of iterations are shown in Figure 4.4. In most cases the maximum iteration has been set to 5000. The solution of the dependent variables is stored in a file every 100 iterations (or any user specified number). This type

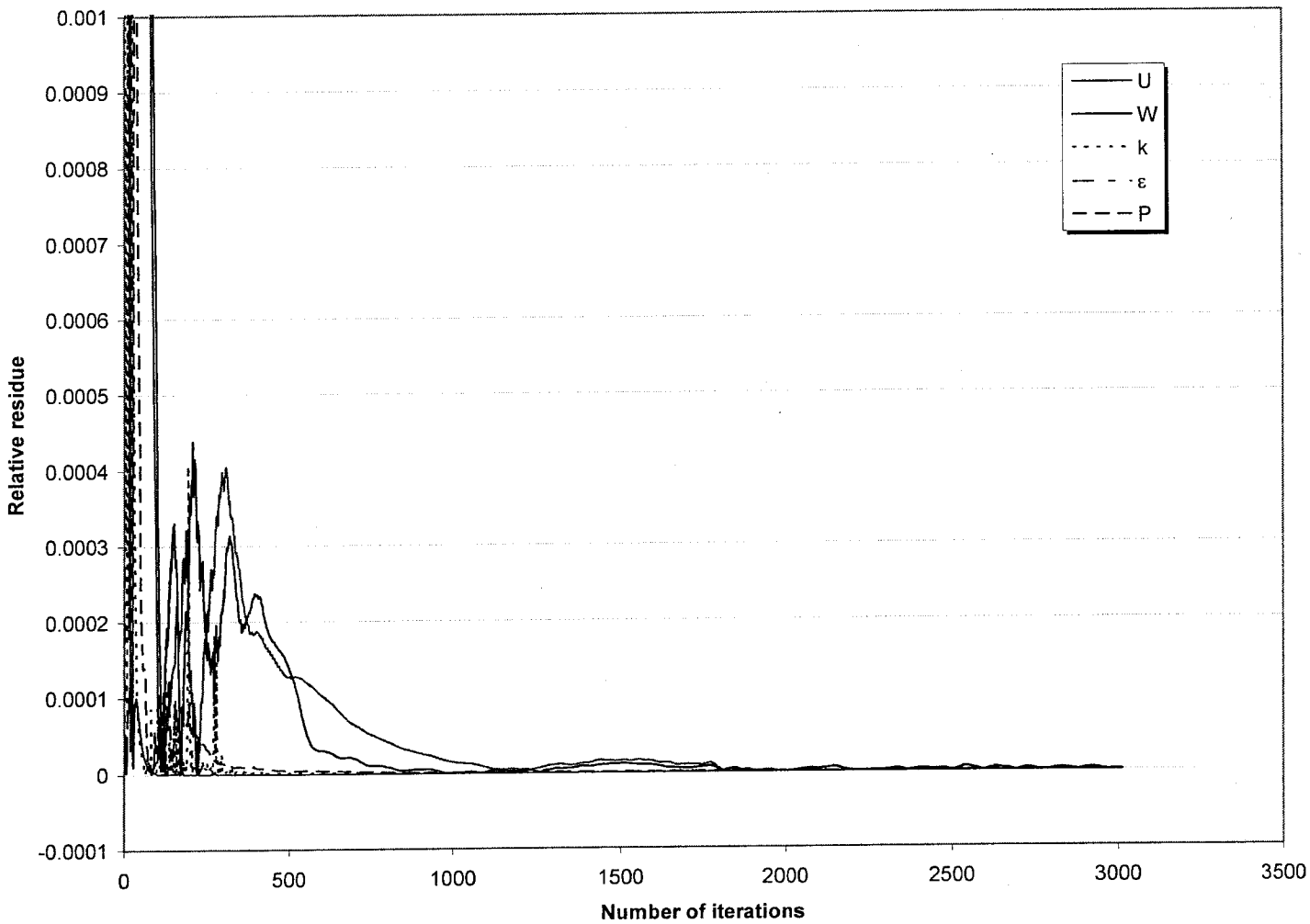


Figure 4.4 Decay of relative residue (i.e. r^i/r^1).

of filing enables to restart the iteration from where it stopped and facilitates the use of stored values as a starting point for other similar cases, thus reducing computational time.

4.1.3 Hardware

A standard personal computer with Intel Pentium IV processor, 1.6 GHz clock speed and 500 MByte RAM specifications has been used. The author believes this type of modest hardware used to obtain the results reported in the present study will inspire engineers in considering a computational approach alternative to evaluate design wind load parameters.

4.2 Two-dimensional single hills and escarpments

4.2.1 Isolated Hill

Under this category, numerical evaluation of wind flow over shallow and steep single cosine hills is presented. The geometries and roughness parameter used are similar to those used in Carpenter and Locke's (1999) and Chung and Bienkiewicz' (2004) BLWT experiments. Thus similar to Carpenter and Locke's hills, the shallow hill has H=200 mm and L=400 mm and the steep hill has H=200 mm, and L=200 mm, where H represents the height of the hills and L represents the horizontal distance from the crest to where the ground elevation is half the height of the hill. The terrain geometries are shown in Figure 4.5. Scaling up these models by 1000 gives the actual hill they represent. A roughness of 0.02 m (at full scale) equivalent to open terrain category is used.

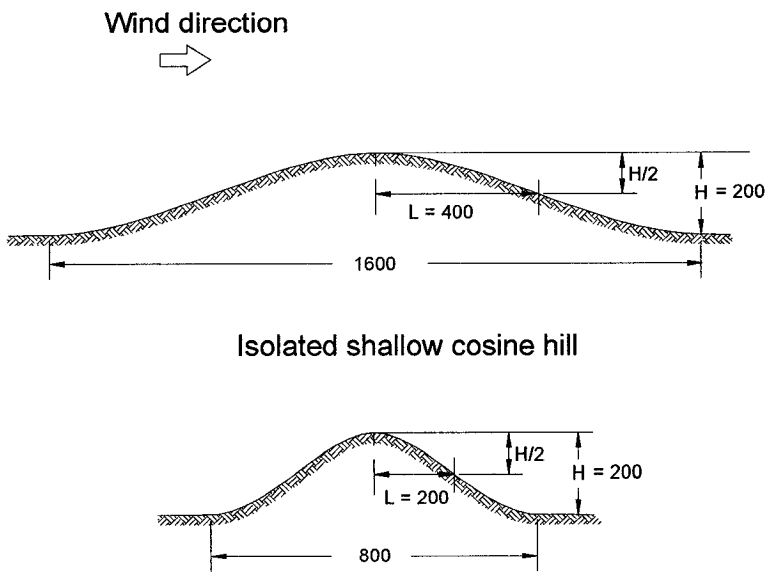


Figure 4.5 Details of hill geometries used by the present as well as Carpenter and Locke's (1999) studies. Scale 1:1000. All dimensions are in [mm].

A 115 x 36 mesh is used both for the shallow and the steep hills as shown in Figures 4.6 (a) and 4.7 (a) respectively. The nearly orthogonal grid system is generated through the procedure presented in section 3.4. In both cases the grid extends over a computational domain of 3.4 m x 1.2 m. The grid points are clustered densely closer to the ground boundary in order to capture the varying gradients of the dependent variables in the region. The first grid point is located closest to the ground boundary at z_p = 0.5 mm so that it satisfies the log-law requirement $z^+>11$. Where z^+ is the so called dimensionless height defined as:

$$z^+ = \frac{U^* z_p}{\nu}, \tag{5.1}$$

where U^* is the friction velocity given by $U^*=(\tau_w/\rho)^{1/2}$, ν is the dynamic viscosity and z_P is distance from the wall. This requirement has been imposed in consistent with the

ground boundary condition applied in the present study which uses log-law as discussed in section 3.1.4.4. It is well documented that for near wall flow the log-law profile holds true for z+>11 (Bradshow 1976) whereas for lower values it takes a linear form.

Numerical results are shown and compared in Figures 4.6 to 4.9. Figures 4.6 (a) and (b) show the grid used and the numerically evaluated mean velocity vectors respectively for the isolated shallow hill. Figures 4.7 (a) and (b) show the same for the isolated steep hill. In both cases, the gradual increase in velocity as the wind approaches the hill and the recirculation region behind the hill have been clearly predicted. The recirculation region for the steep hill is much larger compared to the shallow hill. For reasons of clarity in all velocity field vector figures, such as Figure 4.6 (b) and 4.7 (b), the velocity field vectors are plotted skipping at every other grid point in the x-direction.

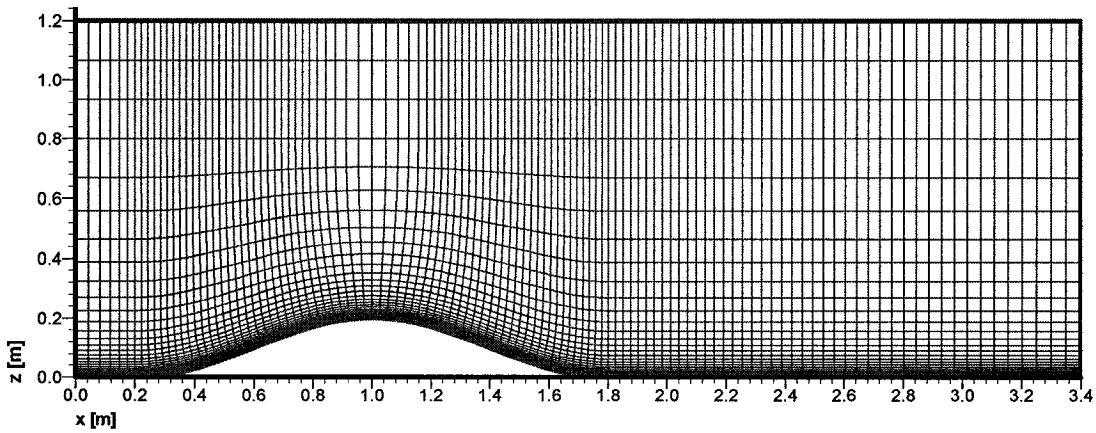
For validation purposes, FSUR values are compared above the crest of the hills where they are critical. The agreement between the present CFD and BLWT results is excellent for both shallow and steep hills as shown in Figures 4.8 and 4.9 respectively. For both shallow and steep hill cases, the present CFD simulation results agree better with the BLWT data than Carpenter and Locke's (1999) with the same BLWT data. Amongst other factors, the close agreement is believed to be primarily due to the incorporation of roughness in the present study through the application of log law throughout the ground boundary that depicts exactly the roughness conditions used by Carpenter and Locke (1999) BLWT experiments. Moreover, the body-fitted grid used in the present study, which allows representation of the true geometry of the BLWT model may have also contributed to the improvement, together with better spatial distribution of grid points

over the computational domain that allows allocation of dense grid points where they are most needed.

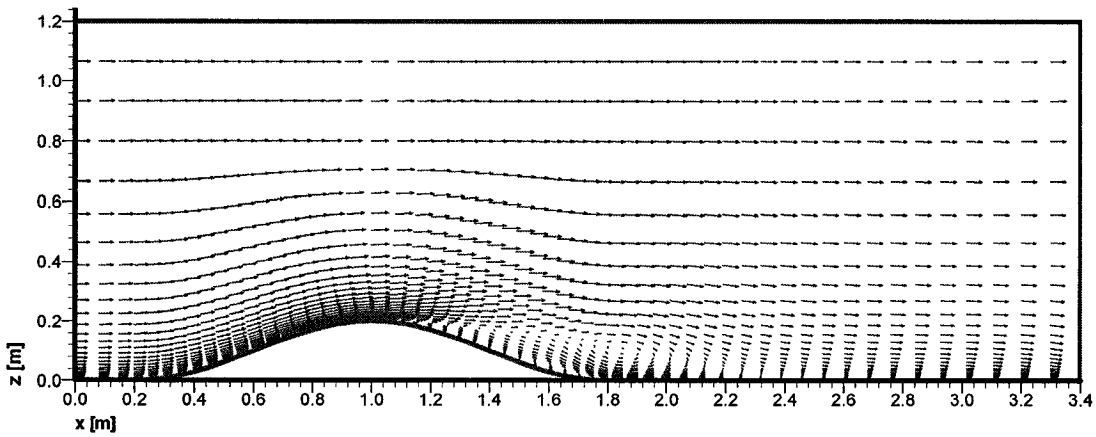
FSUR values at the crest of the hill are also generated using NBCC provisions and Weng et al. (2000) guidelines. Weng et al (2000) guidelines, which were derived from Multi Spectral Finite Difference (MSFD) model of Beljaars et al. (1987) and its nonlinear extension NLMSFD by Xu et al. (1994). Theoretical formulation details are given in the literature review. A typical comparison of FSUR values between CFD, BLWT, calculated values using Weng et al. (2000) guidelines and NBCC provisions (1995) is shown in Figures 4.8 and 4.9 for shallow and steep hill respectively. The agreement between CFD, BLWT, NBCC (1995) and Weng et al. (2000) is good for the shallow hill. However there is sizable discrepancy among Weng et al. (2000) values and the BLWT data near the steep hilltop as shown in Figure 4.9. Weng et al. (2000) guidelines overpredict the FSUR values. This is attributed to the inadequacy of the underlying linear theory used by the guidelines especially for wind flow over steep hills that are characterized by the presence of flow separation and large recirculation regions.

To examine the improvement in the accuracy of the present computational results due to the newly incorporated ground surface roughness, wind flow over shallow and steep single hills is simulated with and without roughness incorporation at the ground surface. For both cases, the FSUR values produced are plotted and compared with BLWT and Carpenter and Locke (1999) CFD data as shown in Figure 4.10. The CFD-without-roughness shows some deviation from BLWT experiments while the CFD-with-roughness produces results closer to that of the BLWT data. This clearly shows that CFD result accuracy is enhanced by incorporating roughness both at the upstream velocity

profile by using log law and throughout the ground boundary by enforcing roughness while calculating the shear stress for the control volumes close to the ground as explained in section 3.1.4.

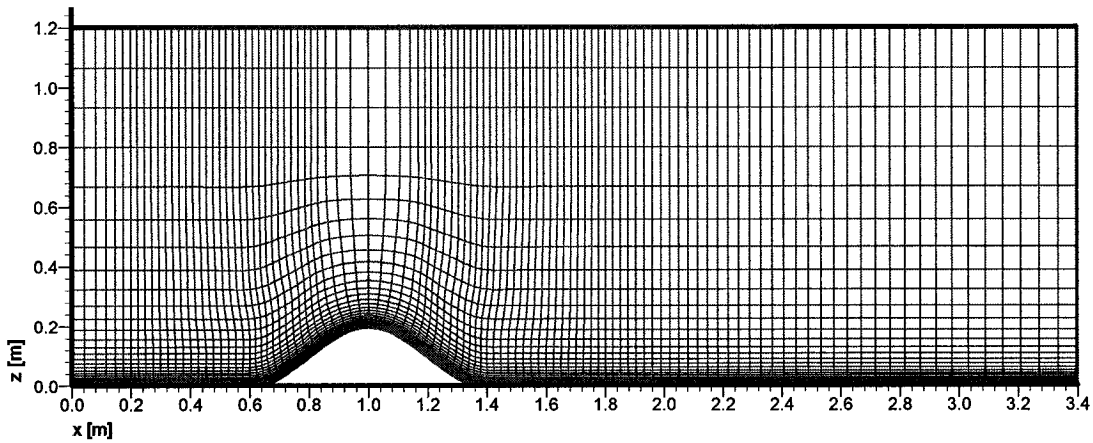


(a)

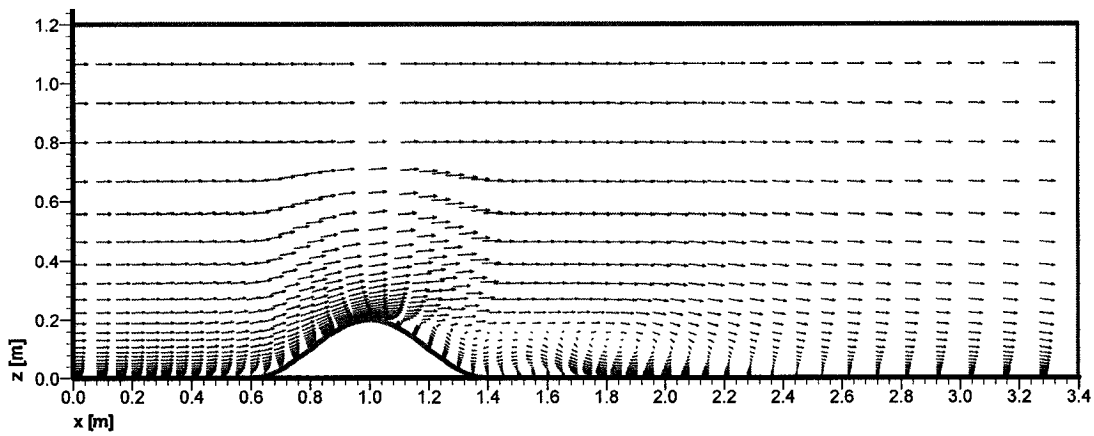


(b)

Figure 4.6 Numerical simulation of wind flow over a single shallow hill: (a) nearly orthogonal grid and (b) velocity field vectors.



(a)



(b)

Figure 4.7 Numerical simulation of wind flow over a single steep hill: (a) nearly orthogonal grid and (b) velocity field vectors.

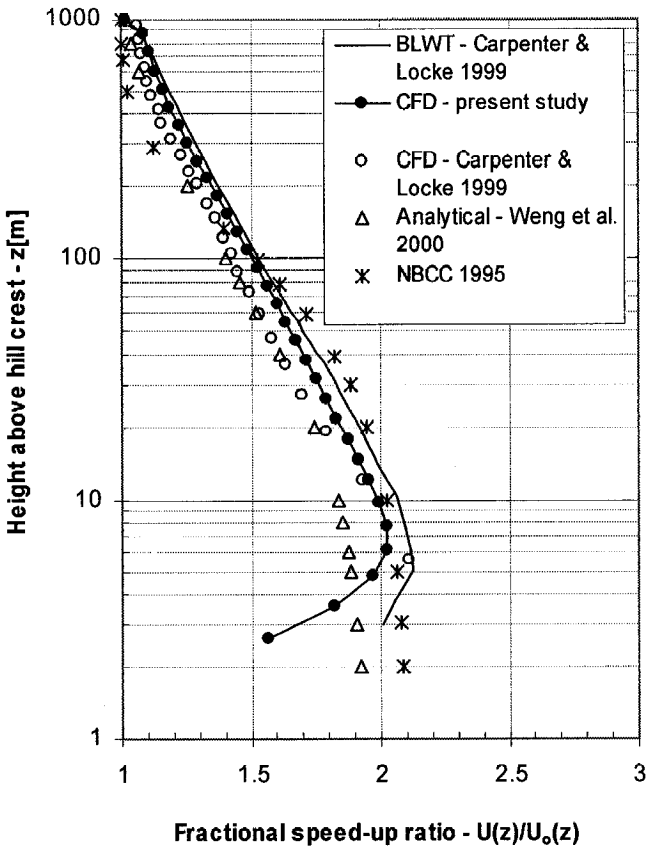


Figure 4.8 Comparison between FSUR values obtained from BLWT, CFD, NBCC and analytical model above the crest of a single shallow hill.

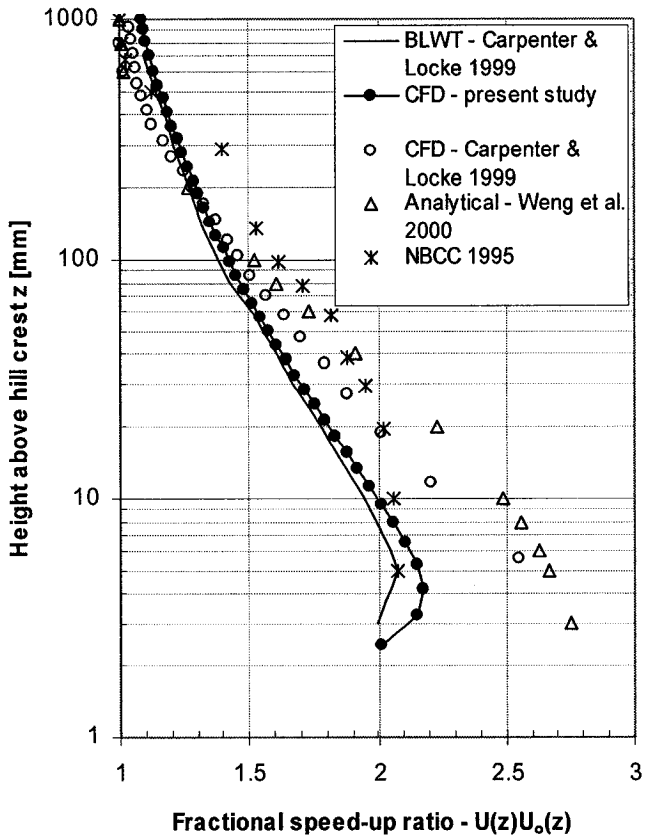


Figure 4.9 Comparisons between FSUR values obtained from BLWT, CFD, NBCC and analytical model above the crest of a single steep hill.

It is also noteworthy that FSUR values obtained by using the present CFD-without-roughness simulations are similar to those of Carpenter and Locke (1999) CFD simulations, that used similar boundary conditions, for shallow and steep cases, as shown in Figures 4.10 (a) and (b). Although both results deviate from BLWT values, the similarity observed in the results among each other under similar boundary conditions illustrates the repeatability of the CFD results. Carpenter and Lock (1999) CFD results were obtained by using a finite element formulation while in the present study finite volume formulation has been adopted.

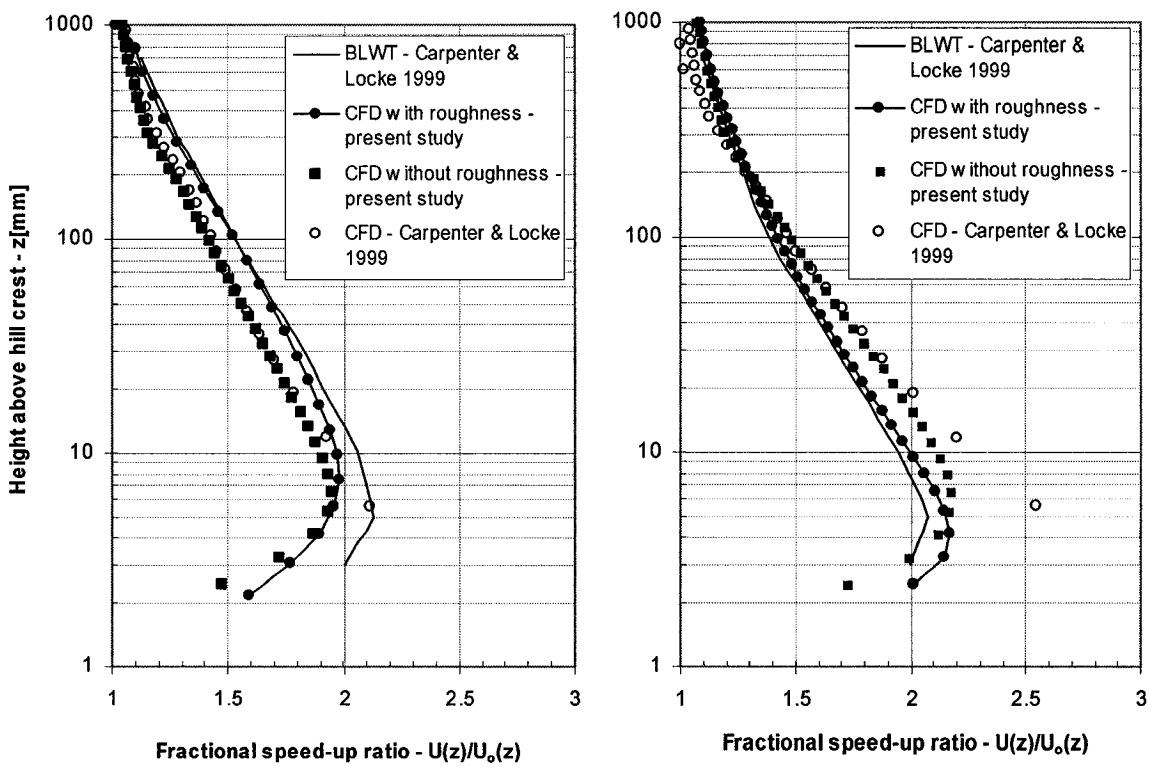


Figure 4.10 Comparison between FUSR values obtained from BLWT and CFD, with and without ground roughness considerations, above the crest of single (a) shallow hill, (b) steep hill.

More recently Chung and Bienkiewicz (2004) have simulated wind flow over the steep hill, as shown in Figure 4.11, numerically and experimentally by using BLWT. Comparison of FSUR values at three different longitudinal locations is shown in Figures 12 (a) to (c). The present CFD results compared very well with the experimental data both at the upstream location (i.e. at x = -L) and at the crest (i.e. at x = 0) of the hill. However, comparison between the two CFD studies for the downstream location (i.e. at x = 10L) is not so good. In the present study, a large recirculation zone has been observed behind the hill when compared to the Chung and Bienkiewicz (2004) CFD simulation resulting in a rather weak agreement between the two numerical evaluations.

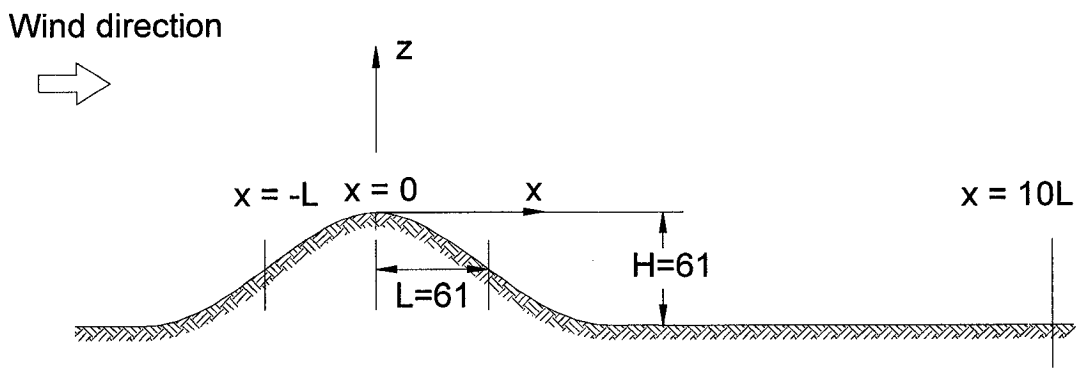


Figure 4.11 Terrain geometry and location of 3 longitudinal sections used by the present as well as Chung and Bienkiewicz (2004) studies. All dimensions are in [mm].

4.2.2 Escarpment

Numerical evaluation of wind flow over an escarpment has been validated using field measurements (Holmes et al. 1997) and BLWT experiments (Glanville and Kwok 1997). A television broadcasting tower near the peak of the escarpment was used for the field measurements of wind speeds at heights up to 69 m. For the CFD simulation, the geometrical and roughness parameters are similar to those used by Glanville and Kwok

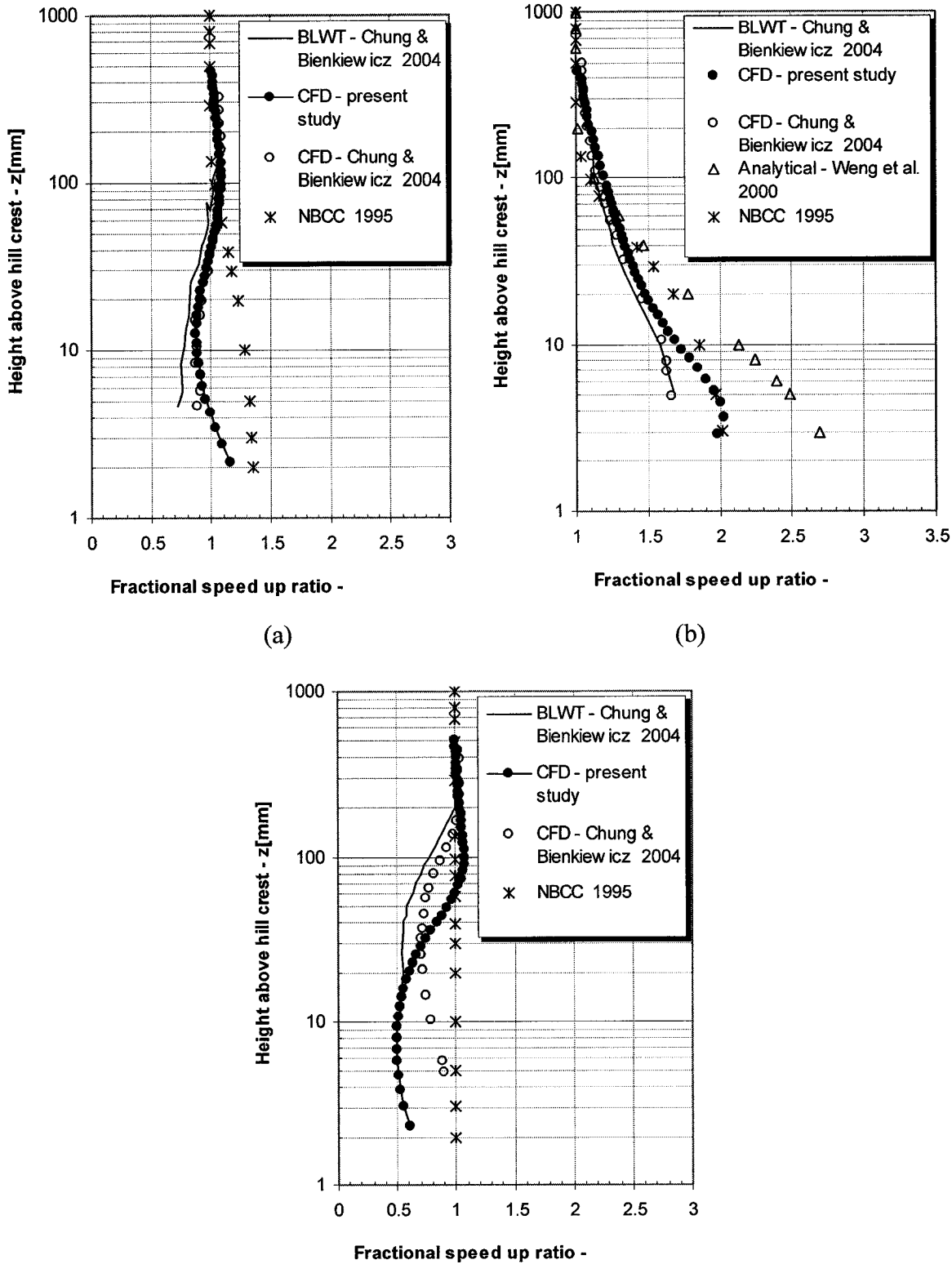


Figure 4.12.Comparison between BLWT and CFD FSUR values at (a) upstream (x=-L), (b) crest of the hill (x =0), and (c) downstream (x =10L) longitudinal locations.

(1997) BLWT experiments. Thus, the escarpment has H=400 mm and L=400 mm and ground roughness of 1.5 mm corresponding to open terrain category. The geometries used for the simulation are shown in Figure 4.13. Scaling up these models by 1000 gives the actual escarpment (Holmes et al. 1997). A 73 x 35 mesh that extends over a computational domain of 4.2 m x 2.4 m. is used as shown in Figure 4.14 (a). The first grid point closest to the ground boundary, z_p=0.5mm, is determined by using the z^+>11 requirement.

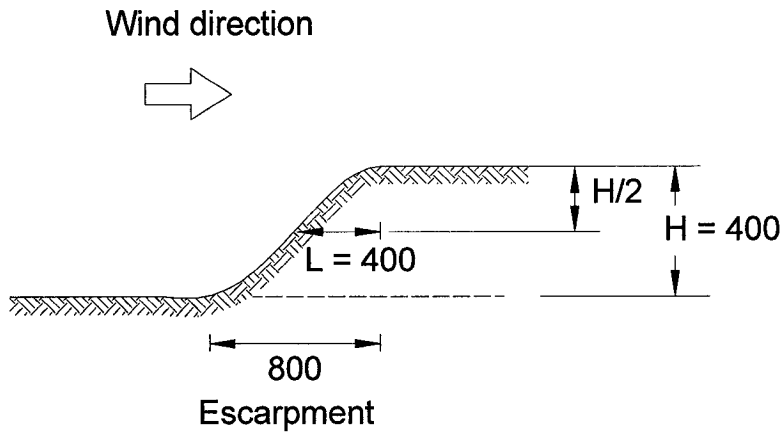
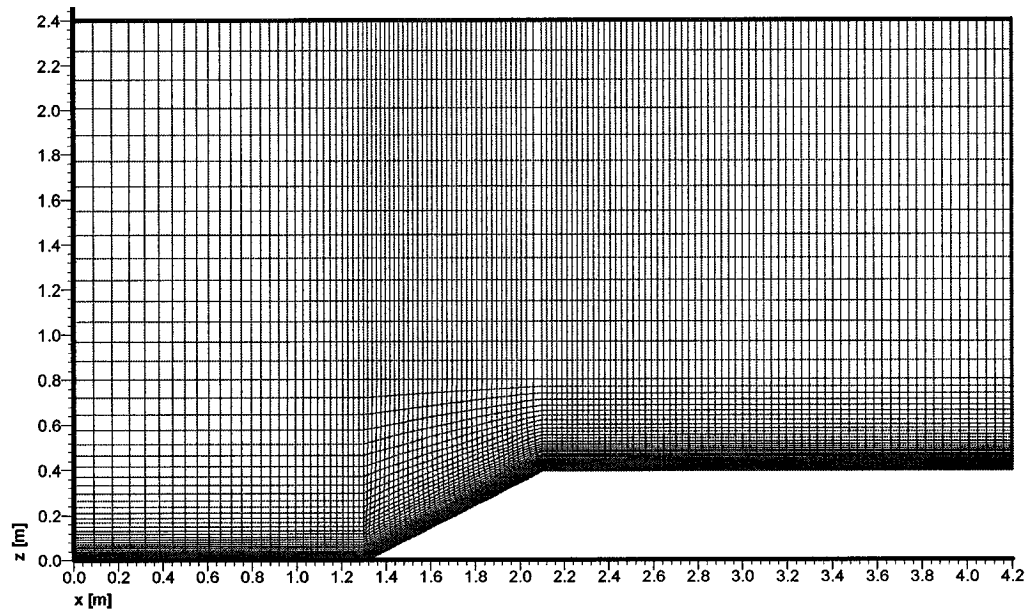
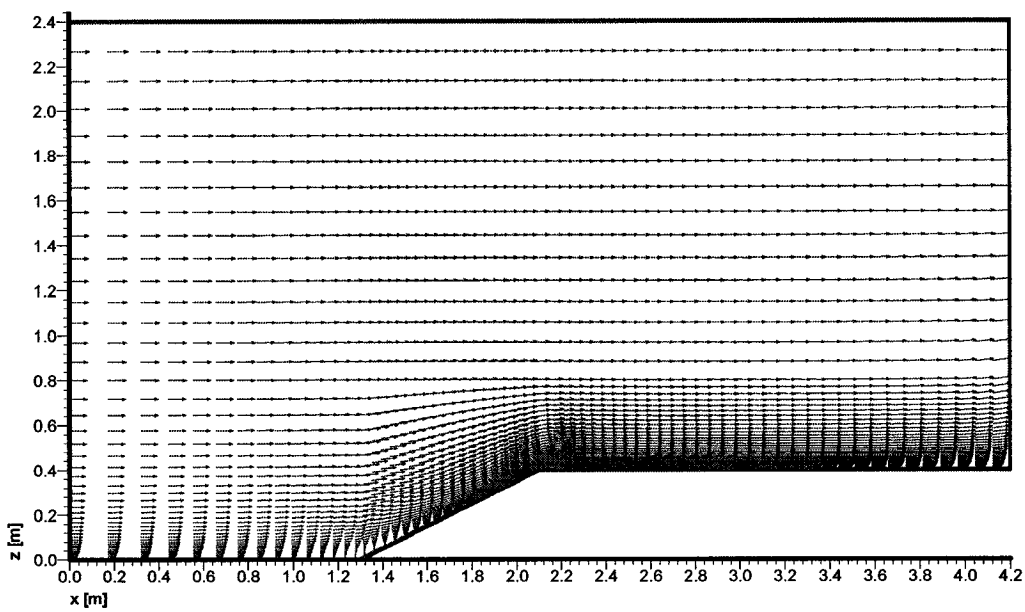


Figure 4.13 Dimensions used by the present CFD as well as Glanville and Kwok (1997) BLWT and Holmes et al. (1997) field study. Scale 1:1000. All dimensions are in [mm].

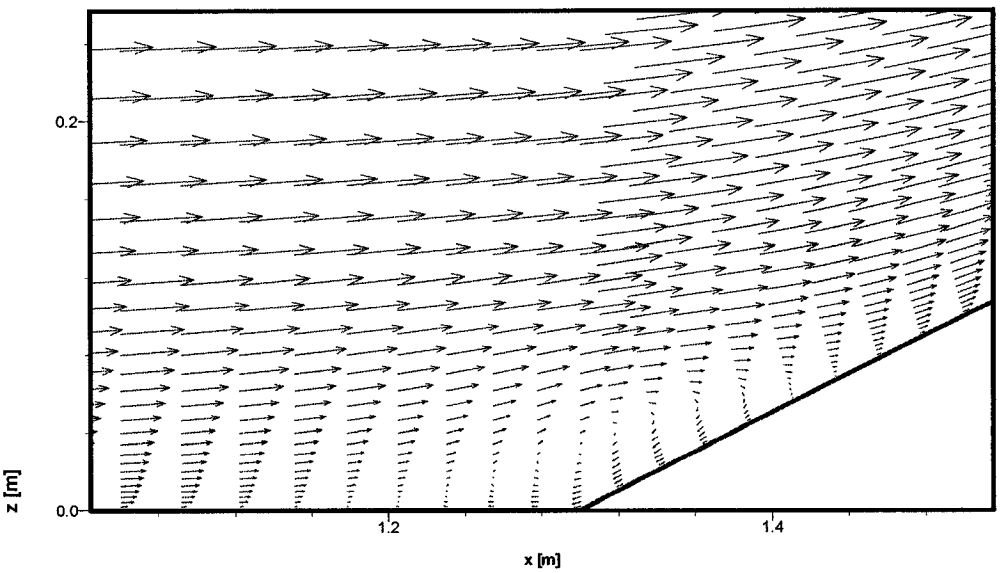
Figures 4.14 (b) and (d) show numerically evaluated mean velocity field vectors for different sections on the escarpment. It is interesting to note the wind flow recirculations both at the upstream foot of the escarpment and at the top surface of the escarpment as shown in Figures 4.14 (c) and (d) respectively. Similarly, Glanville and Kwok (1997) also reported flow circulation at these two places.



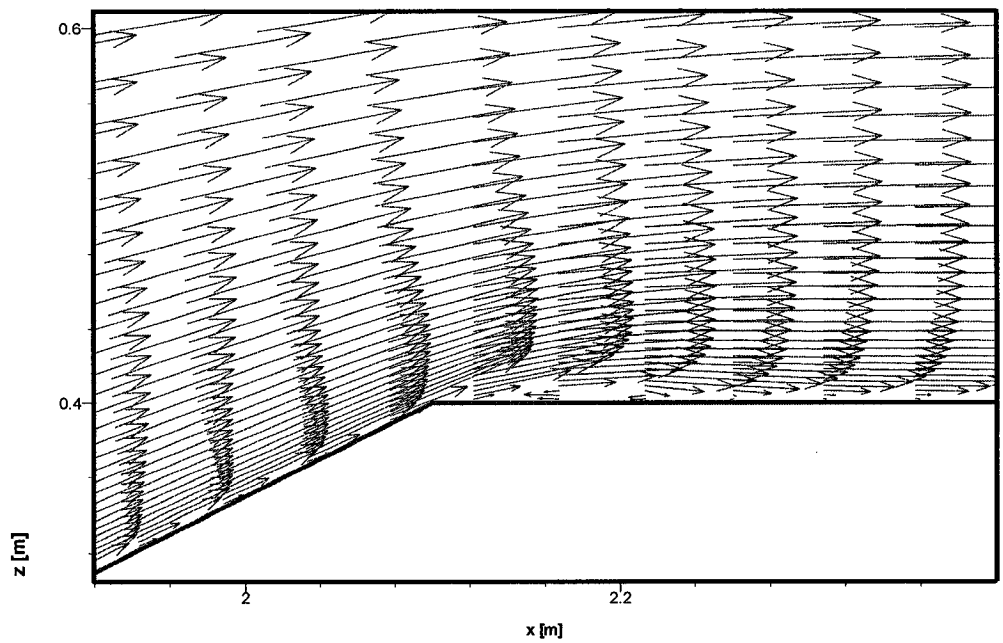
(a)



(b)



(c)



(d)

Figure 4.14 Numerical simulation of wind flow over an escarpment: (a) grid, (b) velocity field vectors, and magnified velocity field vectors (c) at the foot of the escarpment, and (d) at the corner crest of the escarpment.

Figure 4.15 shows comparison between the present CFD, field, NBCC and BLWT FSUR values. The agreement between CFD and BLWT is good except for values very close to the ground. Holmes et al. (1997) field measurements only at three points (i.e. at 32, 45 and 65 meters from the ground) are located between the CFD and the BLWT values. NBCC provisions (1995) underpredicted the FSUR values.

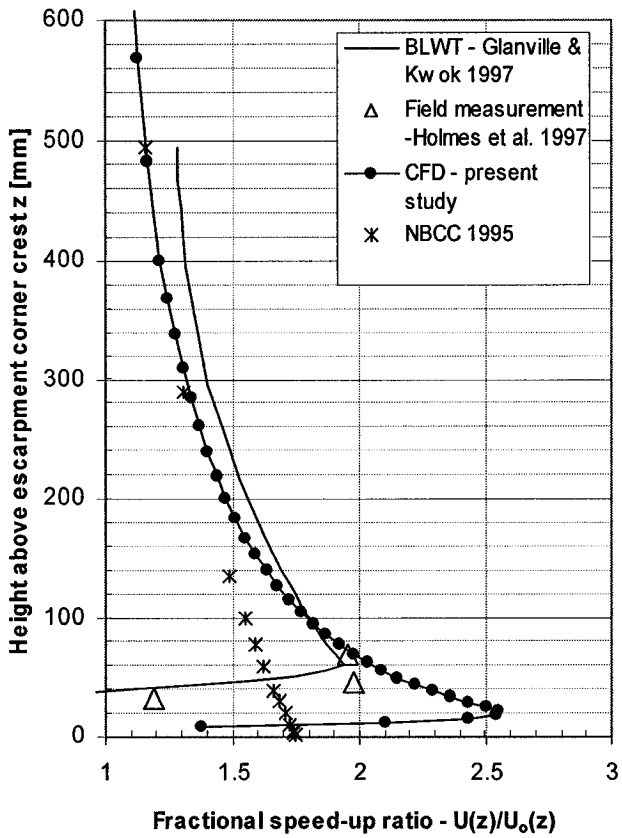


Figure 4.15 FSUR value comparisons above the corner crest of an escarpment among BLWT, field, CFD and NBCC data.

4.3 Two-dimensional multiple hills

4.3.1 Two-dimensional double hills

The geometry and roughness parameters used under this category of the present study are similar to those used in Carpenter and Locke's (1999) BLWT experiments. Thus a roughness of 0.02 m (at full scale) corresponding to open terrain category is used. The shallow hills have H=200 mm and L=400 mm, and the steep hills have H=200 mm, and L=200 mm. The geometries used for the numerical evaluation are shown in Figure 4.16 at a scale of 1:1000. Distance between the crests of the hills is 8H as shown in Figure 4.16. A 179 x 36 mesh covering a computational domain of 5.2 m x 1.2 m is used both for the shallow and steep hills as shown in Figures 4.16 (a) and 4.17 (a) respectively. The first grid point closer to the ground boundary is placed at 0.5 mm from the ground satisfying the log law requirement.

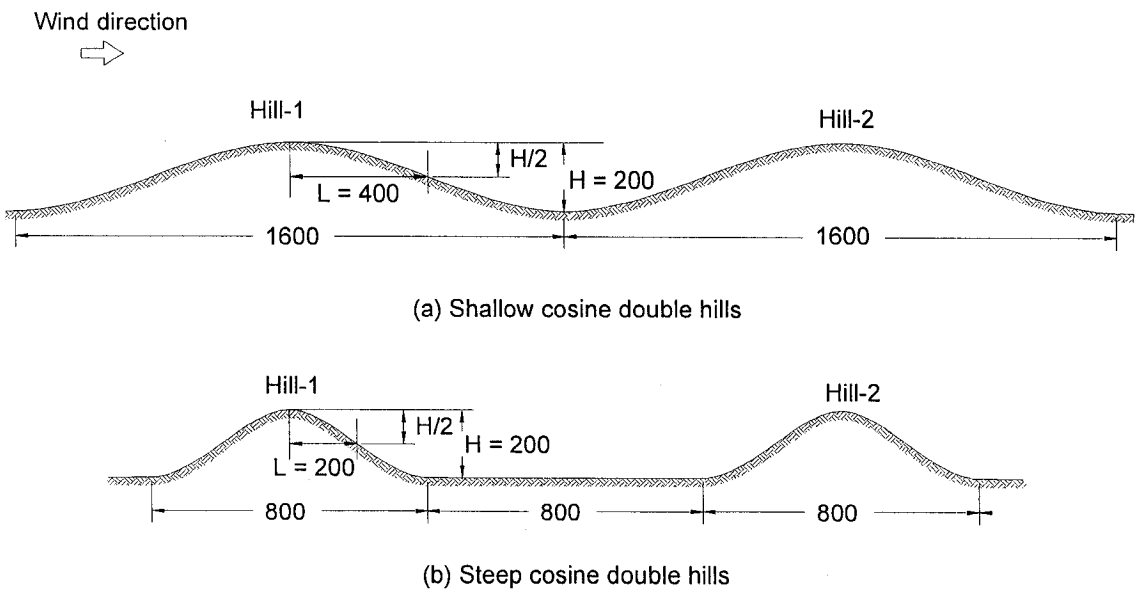


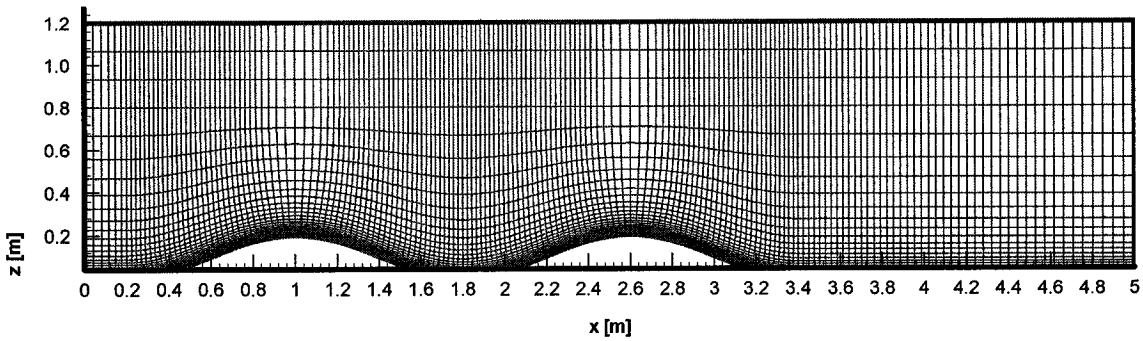
Figure 4.16 Geometry of double shallow and steep cosine double hills used by the present as well as Carpenter and Locke's (1999) studies. Scale 1:1000. All dimensions are in [mm].

Numerical results are shown in Figures 4.17 to 4.20. Figures 4.17 (a) and (b) show the grid used and the numerically evaluated mean wind velocity field vectors respectively for the double shallow hills. Figures 4.18 (a) and (b) show the same for the double steep hills. In both shallow and steep double hill cases, the gradual increase in velocity as the wind approaches the hills and the recirculation region behind the hills are clearly predicted. As the isolated hill case, the recirculation region behind the steep hills is much larger compared to the shallow case as shown in Figures 4.17 (b) and 4.18 (b) respectively. The mean flow characteristic for double hill case however fundamentally differs from that of an isolated hill case especially for the downstream hill. This is because the wind flow separation from the upstream hill (Hill-1) results in an increased turbulence on the downstream hill. This in turn causes the flow to attain more uniformity resulting in a reduction of the peak FSUR values just above the crest of the downstream hill (Hill-2). Hence the FSUR values for the downstream hill (Hill-2) are smaller than the FSUR values for the upstream hill (Hill-1), as shown in Figures 4.19 and 4.20 for shallow and steep double hills respectively. At the crest of the respective hills and for the height range of most engineering interest (5-100 m), upstream hill FSUR values are 20 and 30 % larger than the downstream FSUR values for the shallow and steep double hills, respectively (as shown in Figures 4.19 and 4.20). This percentage in terms of FSUR values can be translated to approximately 45% and 70% increase in wind load for the shallow and steep cases respectively.

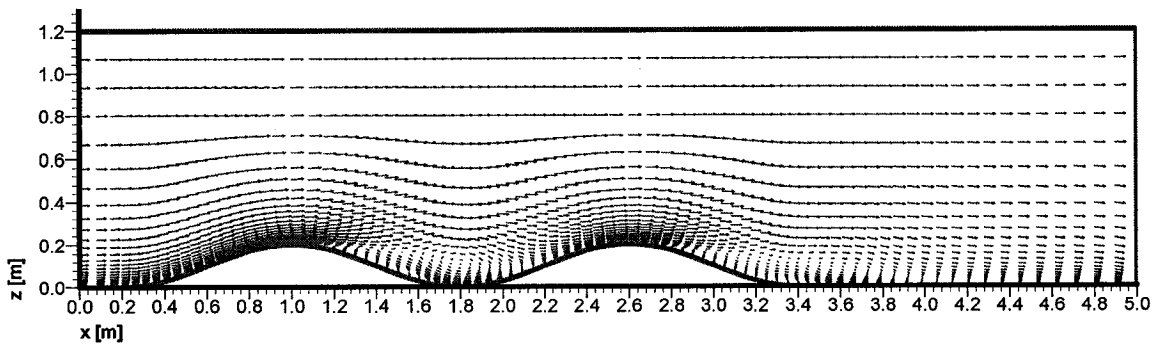
In NBCC (1995) there is no provision for double hills. In practice, NBCC provisions for single hills apply to the double hill case. However, when NBCC provisions are used for the downstream hill case, FSUR values are over predicted approximately by 25% and

44% for the height range of 5-100 m for the shallow and steep hill cases respectively. The wind load is therefore over predicted by 55% and 100% respectively. The economical impact of accurate wind load predictions has been studied by Horesfield et al. (2002) on a residential development consisting of seven tower varying in height from 32 to 37 storeys. It was shown that with the use of more accurate wind loads 19% of the concrete lateral load-resisting system (around 6500 m^3) and 2.1% of usable floor area had been saved. The practice of using identical FSUR values for both the upstream and downstream hills usually obtained from the single hill investigations is indeed a conservative approach. Hence, in areas characterized by chains of mountains, the wind load has to be evaluated based on studies of multiple hills, instead of single hills.

In addition, the detailed wind field evaluations carried out in the present study can be of great importance in environmental planning. In an industrialized area for example, it is apparent from Figures 4.17 (b) and 4.18 (b) that the recirculation zone between the hills can entrap pollutants. Furthermore, considering the increasing trend of using wind energy alternatives in Canada and around the world, the present numerical simulation tool can also be useful in wind farm planning.

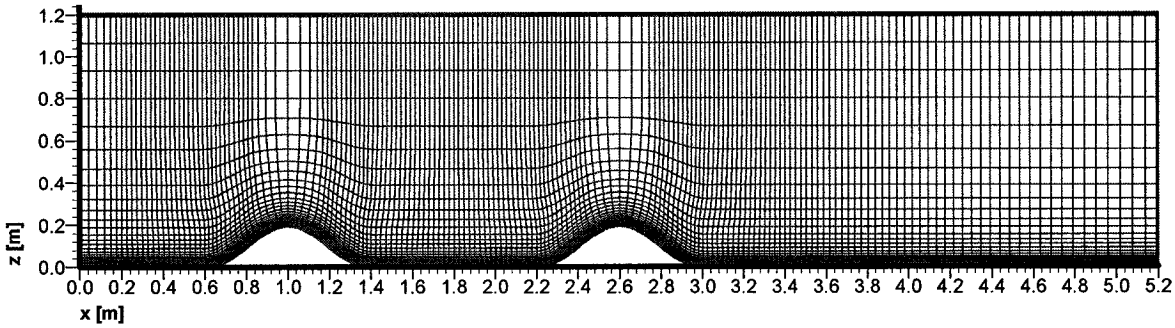


(a)

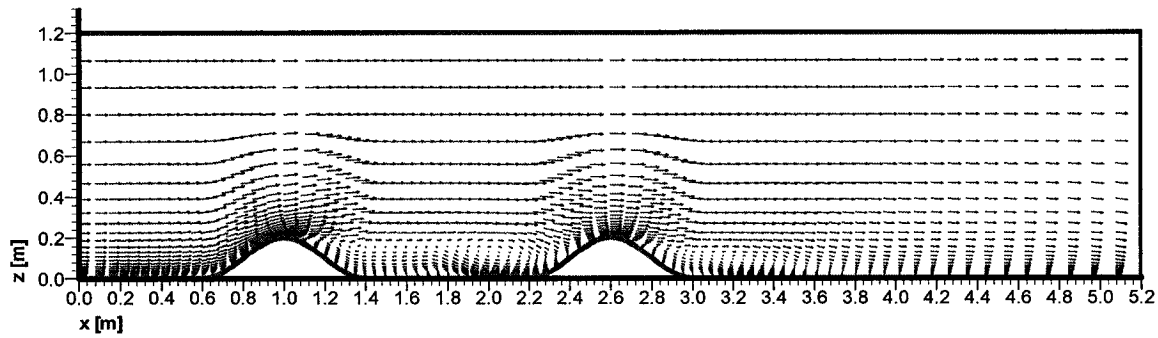


(b)

Figure 4.17 Numerical simulation of wind flow over double shallow hills: (a) nearly orthogonal grid and (b) velocity field vectors.



(a)



(b)

Figure 4.18 Numerical simulation of wind flow over double steep hills: (a) nearly orthogonal grid and (b) velocity field.

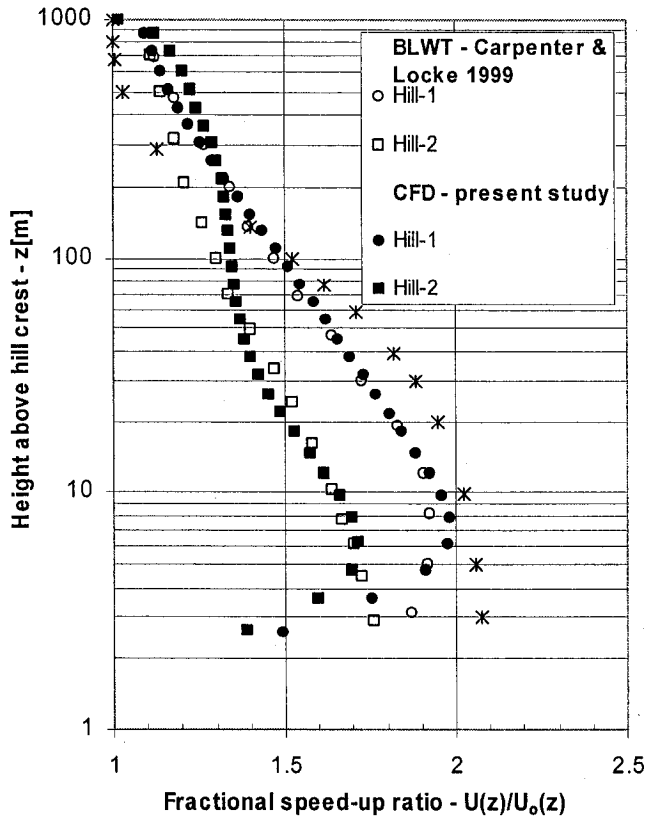


Figure 4.19 Comparison between FSUR values obtained from BLWT, CFD, NBCC and analytical model above the crests of double shallow hills.

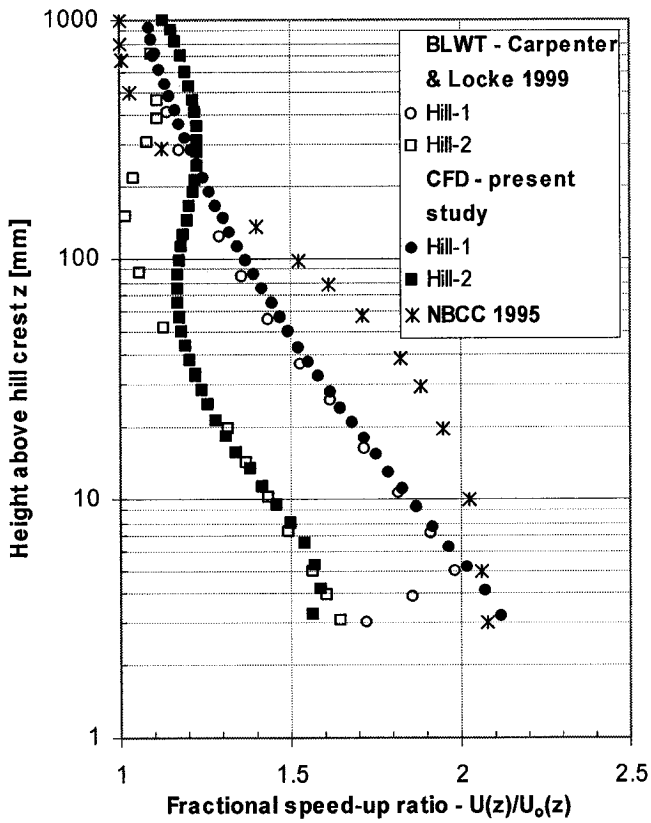


Figure 4.20 Comparisons between FSUR values obtained from BLWT, CFD, NBCC and analytical model above the crests of double steep hills.

4.3.2 Two-dimensional triple hills

The geometry and roughness parameters used under this category are similar to those used in Carpenter and Locke's (1999) BLWT experiments. Thus ground roughness of 0.02 m (full scale) corresponding to open terrain category has been used. The shallow hills have H=200 mm and L=400 mm, and the steep hills have H=200 mm and L=200 mm. The geometries used for the simulation are shown in Figure 4.21. Scaling up these models by 1000 gives the actual hill dimensions. Distance between the crests of hills equal to 8H is used for the case of triple hills as shown in Figure 4.21. A 243 x 36 mesh covering a computational domain of 6.5 m x 1.2 m is used both for the triple shallow and steep hills as shown in Figures 4.22 (a) and 4.23 (a) respectively. The first grid point closest to the ground boundary is placed at 0.5 mm from the ground satisfying the log law requirement.

Numerical results are shown in Figures 4.22 to 4.25. Figures 4.22 (a) and (b) show the grid used and the numerically evaluated mean wind velocity field vectors respectively for the shallow triple hills. Figures 4.24 (a) and (b) show the same for the steep triple hills. FSUR comparisons between CFD, Carpenter and Locke (1999) BLWT, and NBCC provisions (1995) are shown in Figures 4.24 and 4.25 for the case of shallow and steep triple hills respectively. In Carpenter and Locke (1999), BLWT experiments show the highest crest FSUR values were observed at the crest of upstream hill (Hill-1), and the smaller values at the crests of the down stream hills (Hill-2 and Hill-3) for both shallow and steep triple hill cases. It is interesting to see the same patterns of FSUR values as observed in the BLWT experiments being captured by the CFD simulations, as shown in

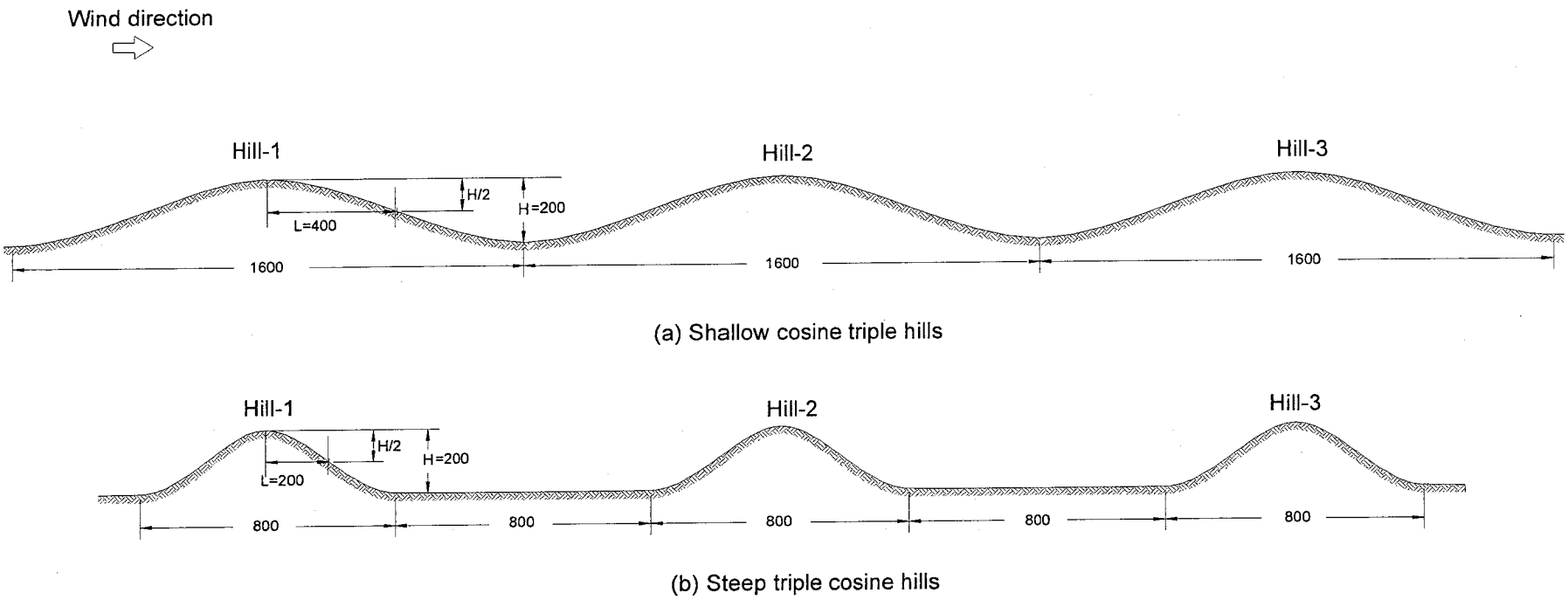
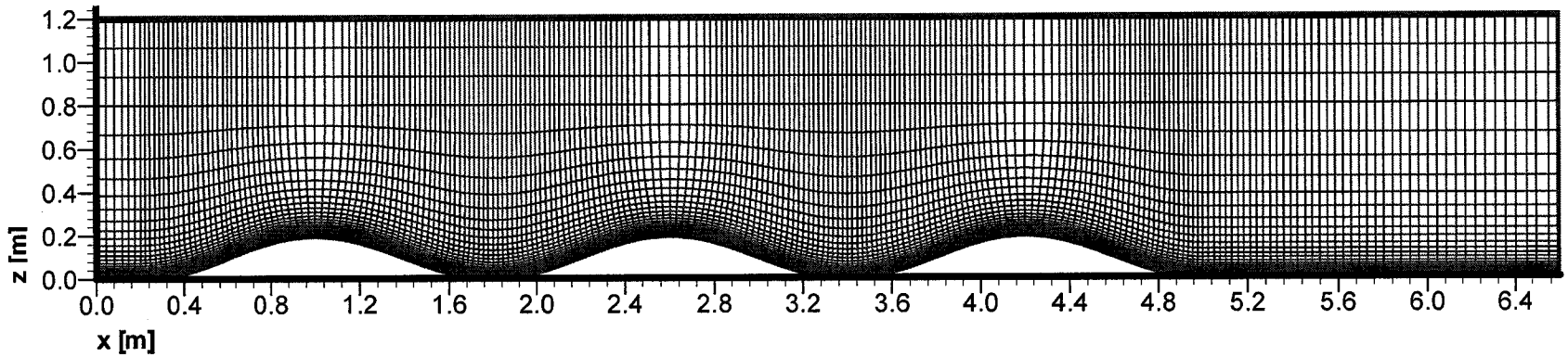


Figure 4.21 Geometry of shallow and steep triple hills used by the present as well as Carpenter and Locke's (1999) studies. Scale 1:1000. All dimensions are in [mm].

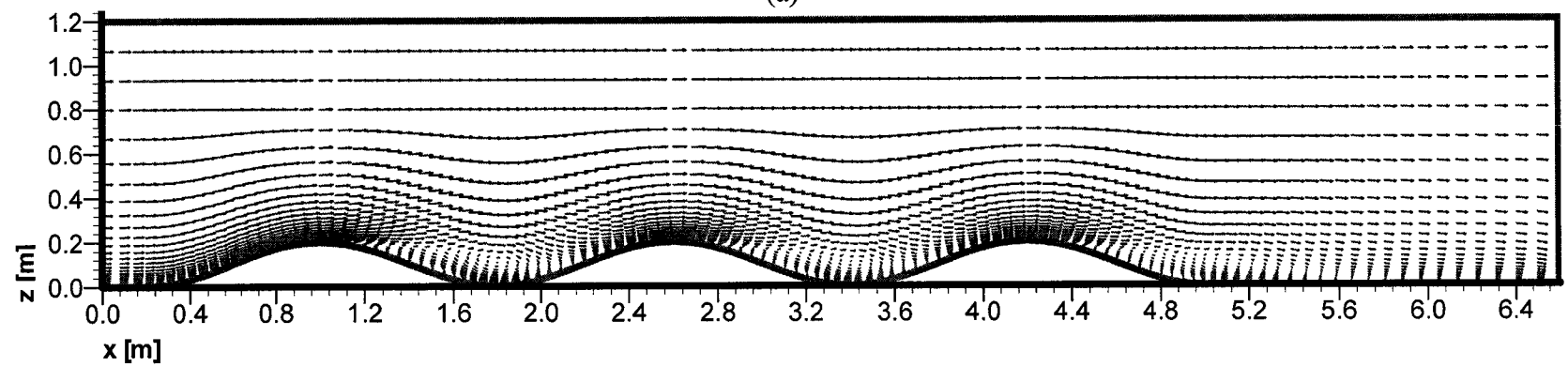
Figures 4.24 and 4.25 for shallow and steep triple hills respectively. This is with the exception of FSUR values at the crest of Hill-2 of the steep hills where there is a sizable difference between the numerical result and the BLWT data as shown in Figure 4.25. This type of variation of FSUR values over the three hills is due to the flow separation from the first hill, resulting in low wind speeds and increased turbulence at the second hillcrest. This increased turbulence at the second hillcrest helps to reduce the flow separation from the second hill and so the mean speed at the third hillcrest increases relative to that at the second hillcrest.

Overall there is good agreement between CFD and BLWT experimental data for triple shallow hills as shown in Figure 4.24. Comparison of FSUR values for the steep triple hill case with BLWT data at the hillcrests shown in Figure 4.25 indicates that CFD predicted FSUR values are in good agreement for the upstream hill (Hill-1) and middle hill (Hill-2), whereas for the last downstream hill (Hill-3), ratios are relatively smaller than BLWT values but still in better agreement than NBCC provisions (1995). Note that the maximum of the FSUR values at Hill-1 for steep hill case did not occur exactly at the hillcrest but one grid point upwind (15 mm) of the crest due to the effect of flow separation closer to the hillcrest. In similar situations, these FSUR values have been used in the present study.

Generally NBCC FSUR values for the isolated hill cases agree with both BLWT and CFD predictions, as discussed in the previous sections, however no provisions are given in NBCC (1995) for the cases of multiple hills. The application of NBCC FSUR values specified for single hills directly to the case of multiple hills is a conservative approach.

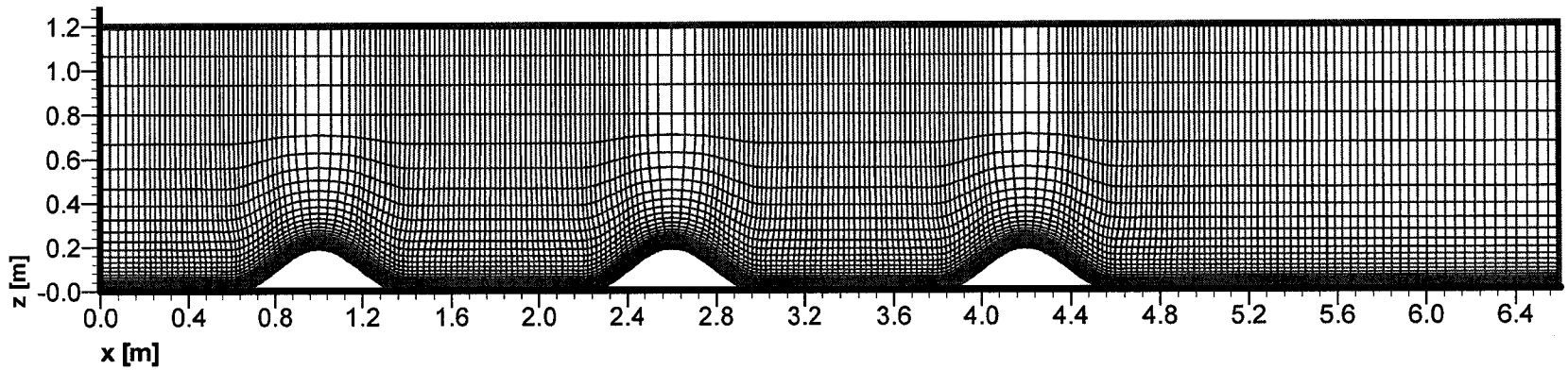


(a)

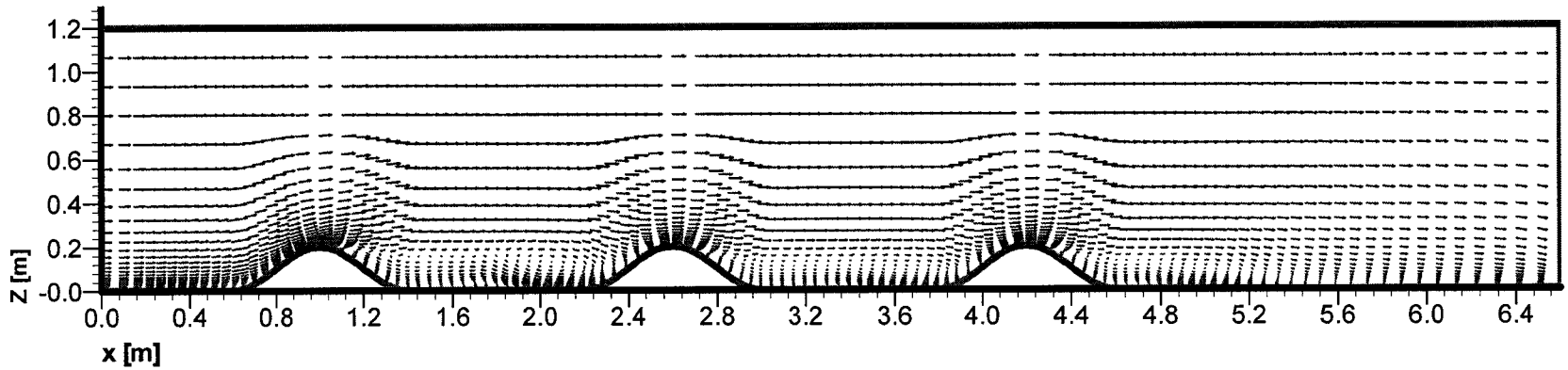


(b)

Figure 4.22 Numerical simulation of wind flow over triple shallow hills: (a) nearly orthogonal grid and (b) velocity field vectors.



(a)



(b)

Figure 4.23 Numerical simulation of wind flow over triple steep hills: (a) nearly orthogonal grid and (b) velocity field vectors.

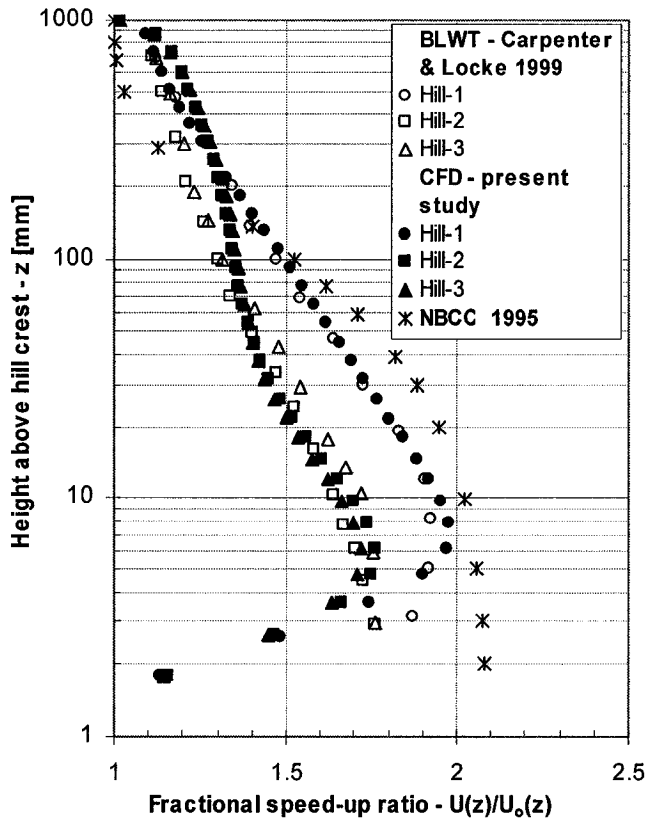


Figure 4.24 Comparisons between FSUR values obtained from BLWT, CFD and NBCC above the crest of shallow triple hills.

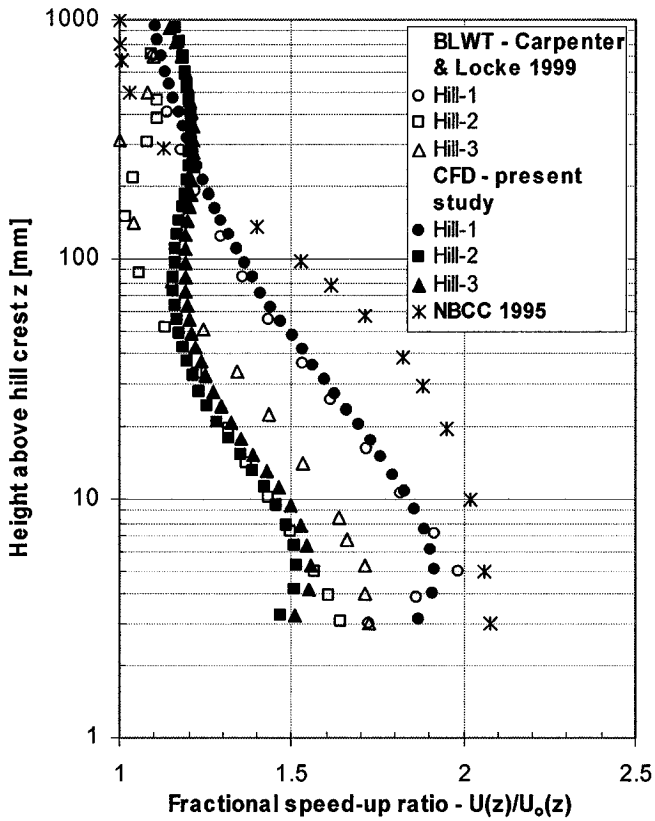


Figure 4.25 Comparisons between FSUR values obtained from BLWT, CFD and NBCC above the crest of steep triple hills.

For example, NBCC (1995) would have over-predicted the FSUR values by 47% and 43% for shallow Hill-2 and Hill-3 respectively at 40 m above the hillcrest. This can be translated to an increase of 100% wind load approximately. As pointed out previously, the economical impact of wind load prediction accuracy is very significant. In areas characterized by chains of mountains, the wind load has to be evaluated based on studies of multiple hills instead of single hills. Thus the proposed CFD-NN tool can be used to complement the NBCC provisions (1995) for such cases.

Last but not least, sensitivity of wind flow simulations to geometrical variations/ approximations has been studied. For this purpose numerical evaluation of wind flow has been carried out by using two most widely used hill shapes in literature, Jackson and cosine hill, represented by equations (3.64) and (3.65) respectively. The hills represented by these equations show slight difference in the area covered under the hill equations as shown in Figure 4.26. This difference is more evident when the hill has large dimensions, such as the shallow hill case in this study. Recall, the shallow hills have larger width than the steep hills while having the same height.

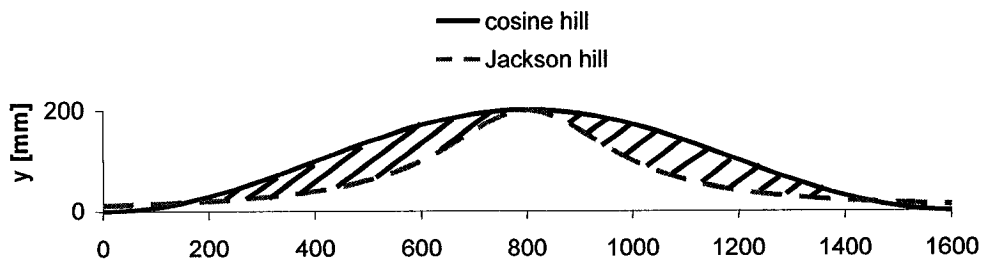


Figure 4.26 Differences between Jackson and Cosine hills.

Figures 4.27 and 4.28 show comparisons between the CFD results with Carpenter and Locke (1999) BLWT data for Jackson and cosine shallow triple hills respectively. Overall, the CFD result for the cosine hills agrees with Carpenter and Locke (1999)

BLWT data, since the latter have also used cosine hills. However, the CFD results for Jackson hills deviate from Carpenter and Locke (1999) BLWT data, especially for the upstream hill, i.e. Hill-1. This indicates that CFD simulations are very sensitive to geometric approximations. Clearly care has to be taken to represent the hill geometry as accurately as possible. The effort and time spent in developing a robust body-fitted grid generator tool which represents the ground surface accurately is thus well justified in the present study.

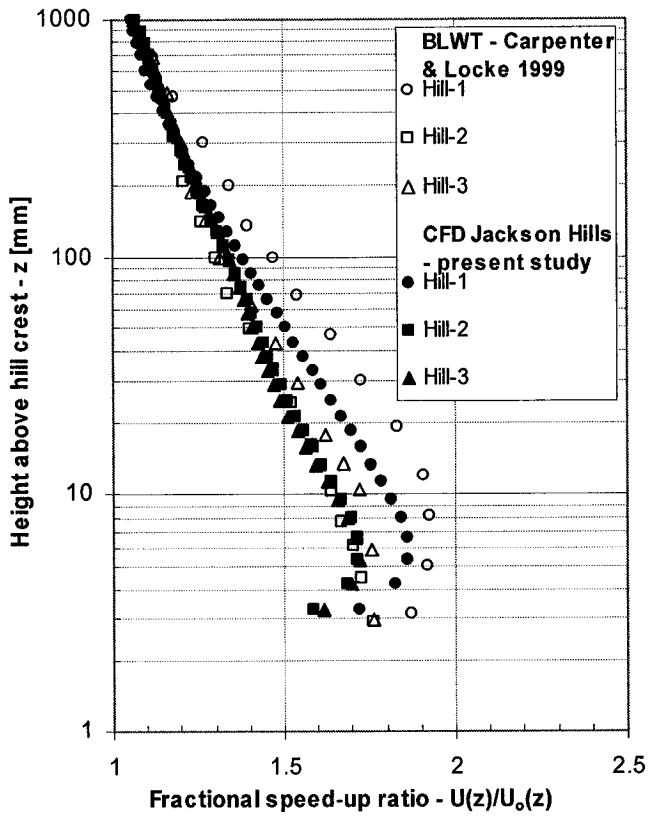


Figure 4.27 Comparisons between FSUR ratio values obtained using BLWT experiments and CFD simulation at the crest of shallow Jackson triple hills.

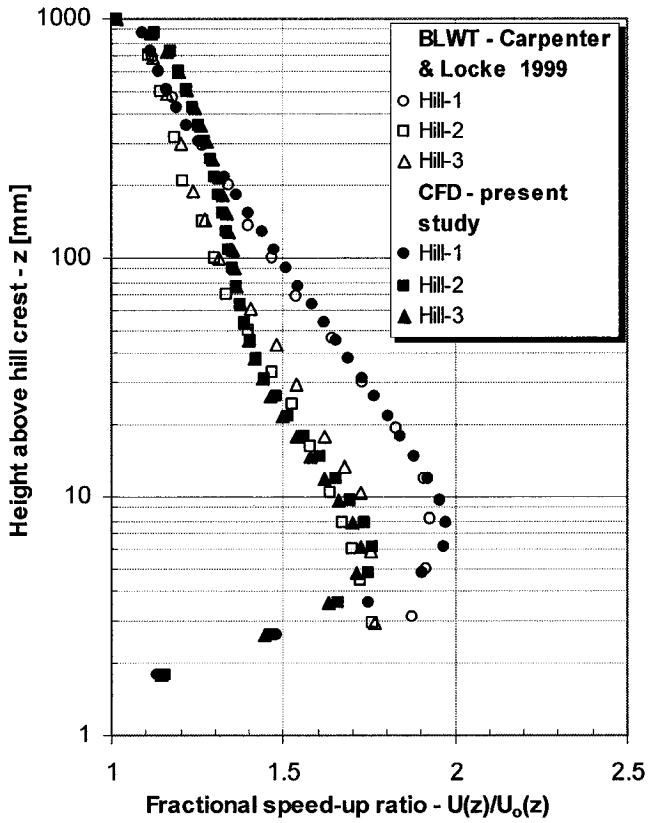


Figure 4.28 Comparisons between FSUR values obtained using BLWT experiments and CFD simulation at the crest of shallow cosine triple hills.

4.4 Two-dimensional valley

In this category, numerical evaluations of wind flow over shallow and deep valleys are presented. The geometry and roughness parameters used for this category are made similar to those used in Busuoli et al.(1993) BLWT experiments. The ground roughness is 0.016 m (full scale). The shallow valley has H=117 mm and L=468 mm, whereas the deep valley has H=117 mm and L=175.5 mm. The geometries are shown in Figure 4.29.

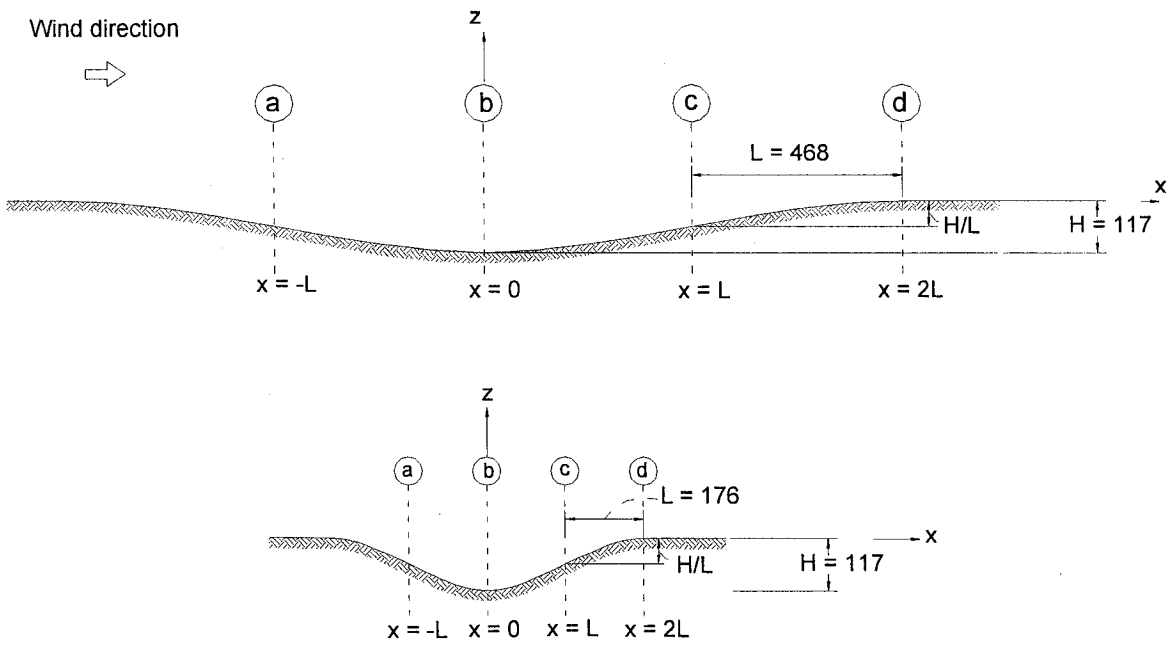


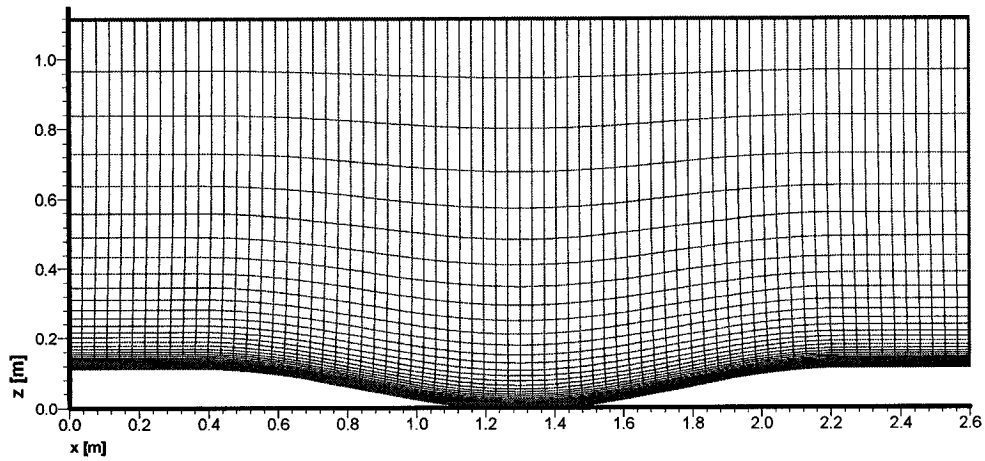
Figure 4.29 Geometry of shallow and deep cosine valleys used by the present as well as Maurizi' (2000) studies. Scale 1:1000. All measurements are in [mm].

4.4.1 Shallow valley

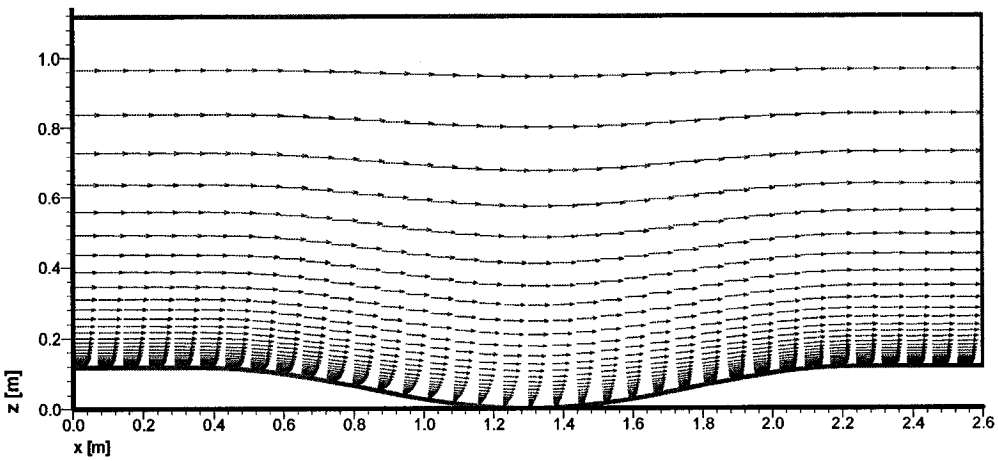
In this section both the standard and the RNG k-ε turbulence models have been used to evaluate the wind flow over the valley. A 73 x 37 mesh extending over a computational domain of 2.6 m x 1.12 m is used as shown in Figure 4.30 (a). The first grid point closest to the ground boundary z_p=0.5mm is determined to satisfy the z^+>11 requirement. Figure 4.30 (b) shows the numerically evaluated mean velocity field vectors for the shallow

valley obtained by using the standard k-ε turbulence model. On the other hand Figure 4.31 shows the same result obtained by using the RNG k-ε turbulence models. In contrast to the hill cases, the wind velocity decreases as it approaches the valleys. This gradual decrease in wind velocity inside the valley has been clearly predicted by the numerical simulation. The FSUR values obtained using the standard k-ε model are in a good agreement with the BLWT data at four different longitudinal locations, as shown in Figure 4.32. However, the RNG k-ε has exaggerated the flow recirculation inside the valley as shown in Figure 4.31. As a result, the FSUR values obtained using the RNG k-ε turbulence model deviates considerably from the BLWT values especially inside the recirculation region as shown in Figures 4.32 (a) and (b). Outside the recirculation region (sections d and c), however, similar values with FSUR values obtained using the standard k-ε turbulence and the BLWT data have been found as shown in Figures 4.32 (c) and (d).

Further FSUR values are calculated for the shallow valley applying the guidelines of Weng et al. (2000) and the NBCC provisions (1995). Comparisons of FSUR values have been made at the same four different longitudinal locations. The NBCC FSUR values agree well with other FSUR values for the shallow valley except at the downstream edge of the valley (section-d) as shown in Figure 4.32 (d). FSUR values obtained using the guidelines of Weng et al. (2000) are also compared at the center of the valley (section-b), where the Weng et al. (2000) model is only applicable. The guideline values of Weng et al. (2000) are also in good agreement with CFD, BLWT and NBCC at the center of the shallow valley.



(a)



(b)

Figure 4.30 Numerical simulation of wind flow over a single shallow valley using Standard k-ε turbulence model: (a) nearly orthogonal grid and (b) velocity field vectors.

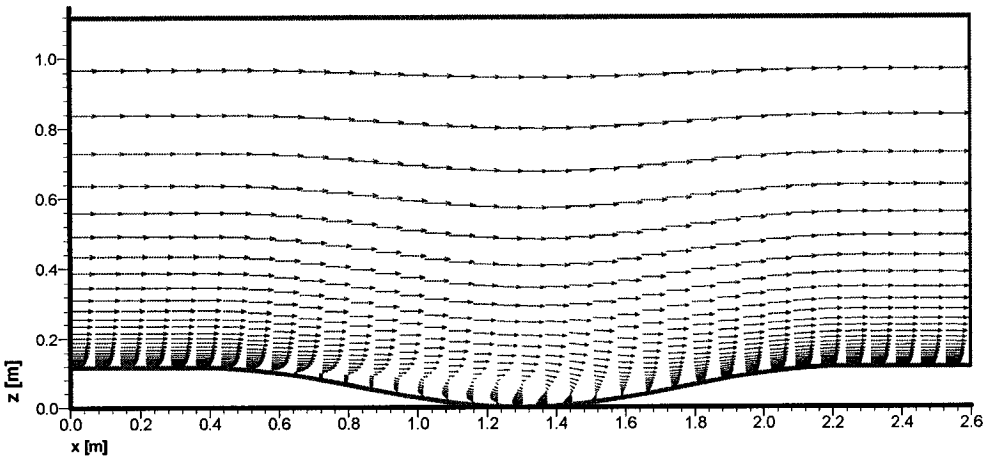


Figure 4.31 Numerically obtained velocity field vectors over a single shallow valley using RNG k-ε turbulence model.

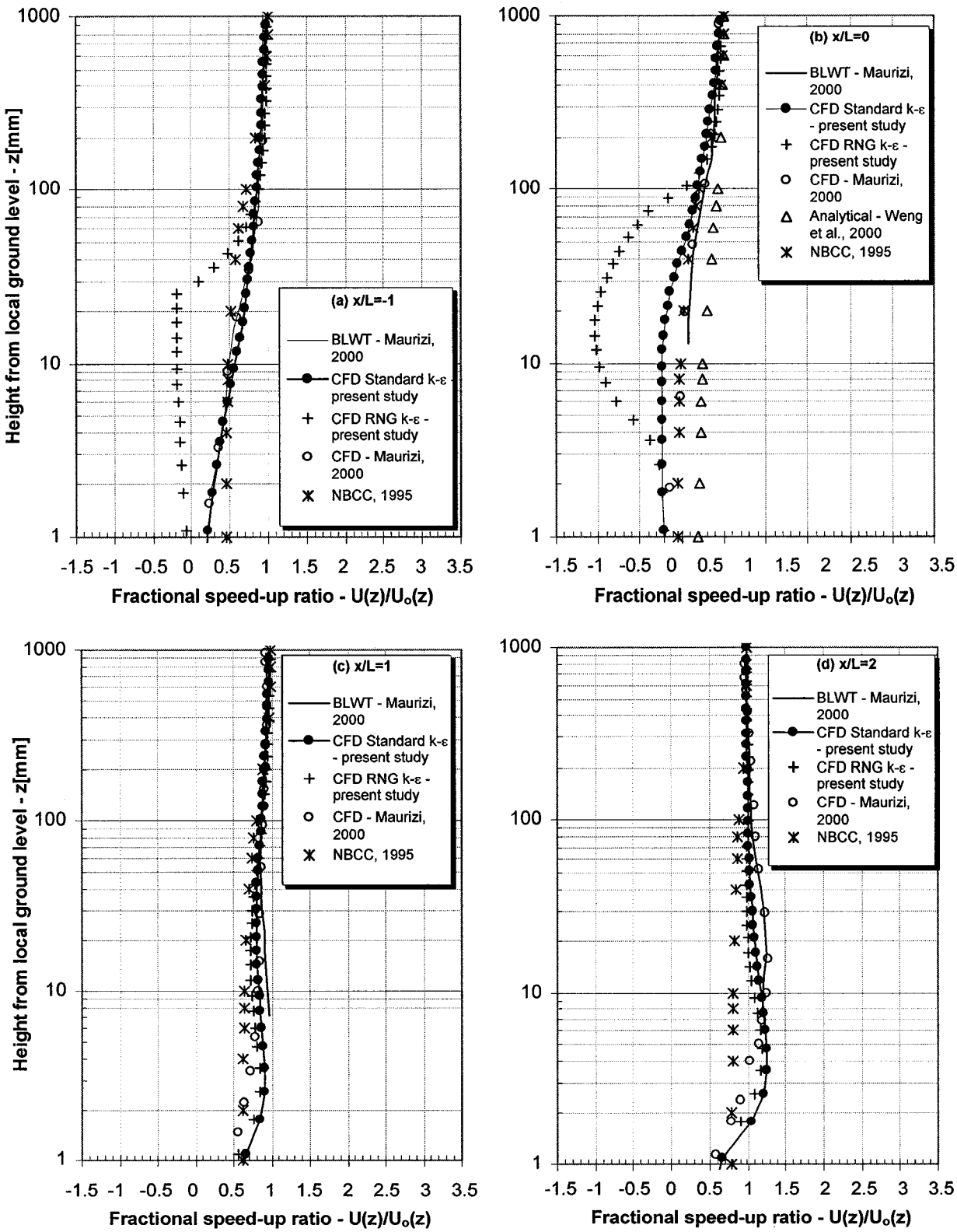
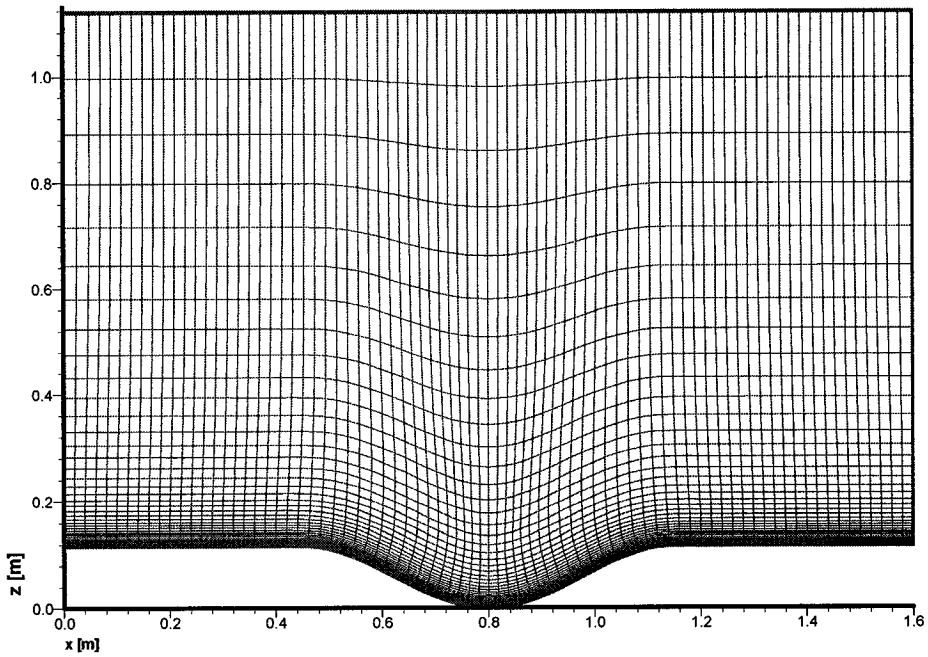


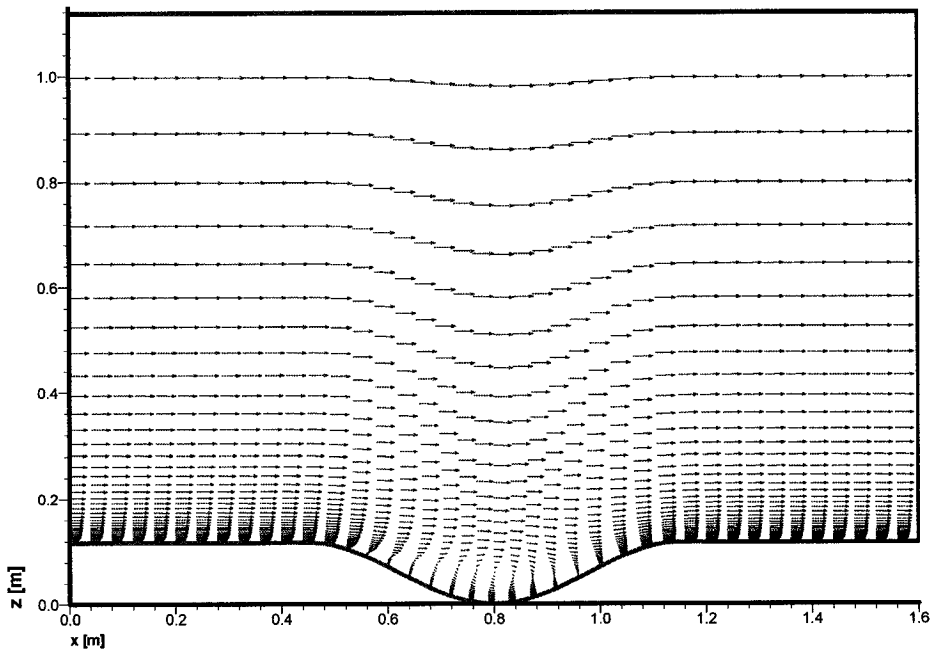
Figure 4.32 Comparison between CFD, BLWT, and NBCC FSUR values for the shallow valley (H/L = 0.125) at four longitudinal locations.

4.4.2 Deep valley

Here an 81 x 45 mesh extending over a computational domain of 1.6 m x 1.12 m is used as shown in Figure 4.33 (a). The first grid point closest to the ground boundary z_p=0.5mm is determined to satisfy the z^+>11 requirement. For the deep valley case, both the standard and RNG k-ε turbulence models have shown similar recirculation zones as shown in Figures 4.33 (b) and 4.34 respectively. To better visual comparison the magnified views of the velocity field vectors are also shown in Figures 4.35 (a) and (b) for the RNG and standard k-ε turbulence models respectively. Good FSUR value agreement between the two numerical evaluations (i.e. based on standard and RNG k-ε turbulence models) and BLWT data have been found in all the four longitudinal locations as shown in Figure 4.36. The comparisons of NBCC FSUR values with the numerical and the BLWT data become less satisfactory for the deeper valley as shown in Figures 4.36 (b) and (d). Particularly poor agreement is observed at the downstream edge of the deep valley (section d) as shown in Figure 4.36 (d). FSUR values obtained using the guidelines of Weng et al. (2000) are also compared at the center of the deep valley (section-b), where such guidelines can only be applied. Although the guideline values of Weng et al. (2000) are in good agreement with CFD, BLWT and NBCC for shallow valley as discussed in the previous section, they overestimate FSUR values for the deeper valley as shown in Figure 4.36 (b). The deviations of NBCC provisions (1995) and overestimation of Weng et al. (2000) guidelines might have originated from the underlying theoretical model's inability to simulate separating flow.



(a)



(b)

Figure 4.33 Numerical simulation of wind flow over a single deep valley using standard k-ε model: (a) nearly orthogonal grid and (b) velocity field vectors.

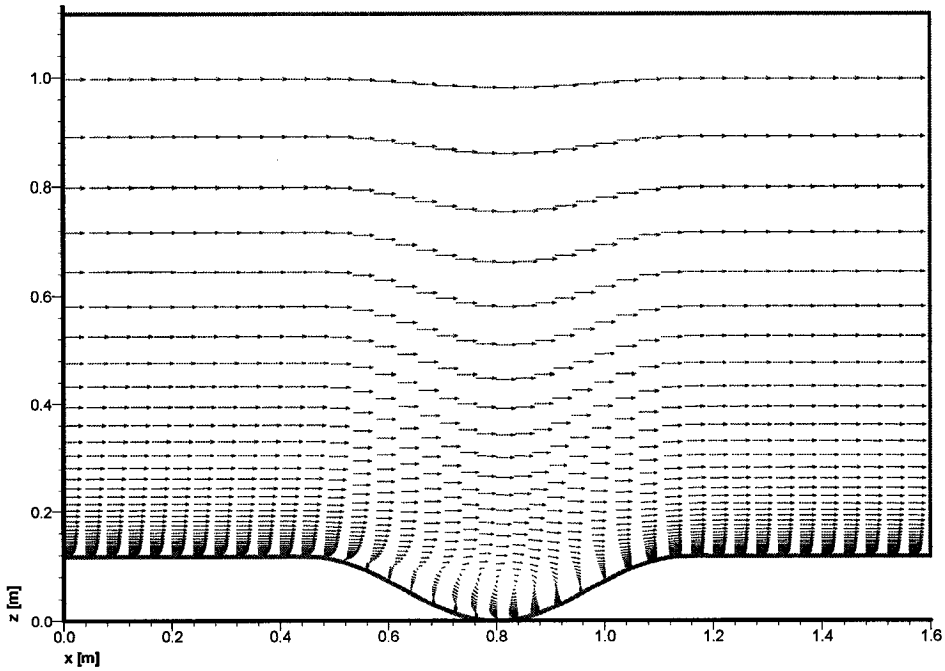


Figure 4.34 Numerically obtained velocity vector field over a single deep valley using RNG k-ε turbulence model.

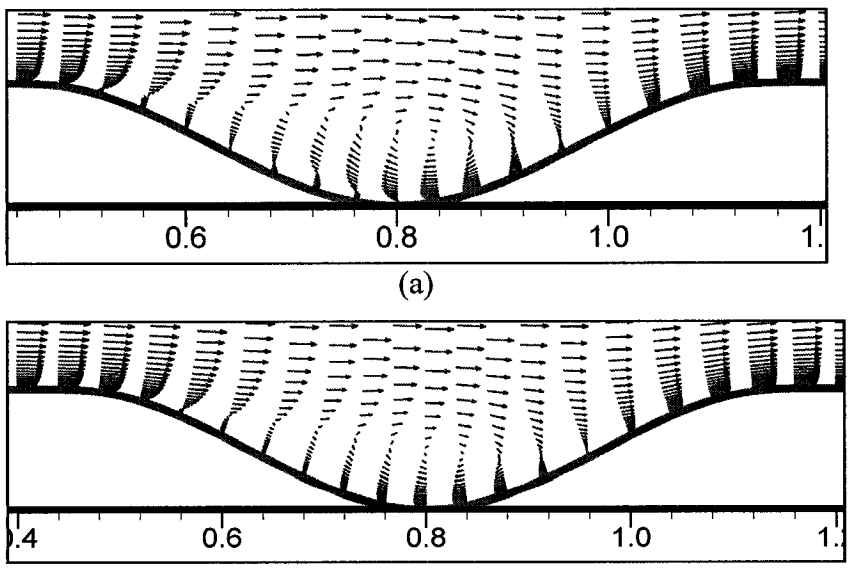


Figure 4.35 Magnified view of the numerically obtained velocity vector field over a single deep valley using (a) RNG and (b) standard k-ε turbulence model.

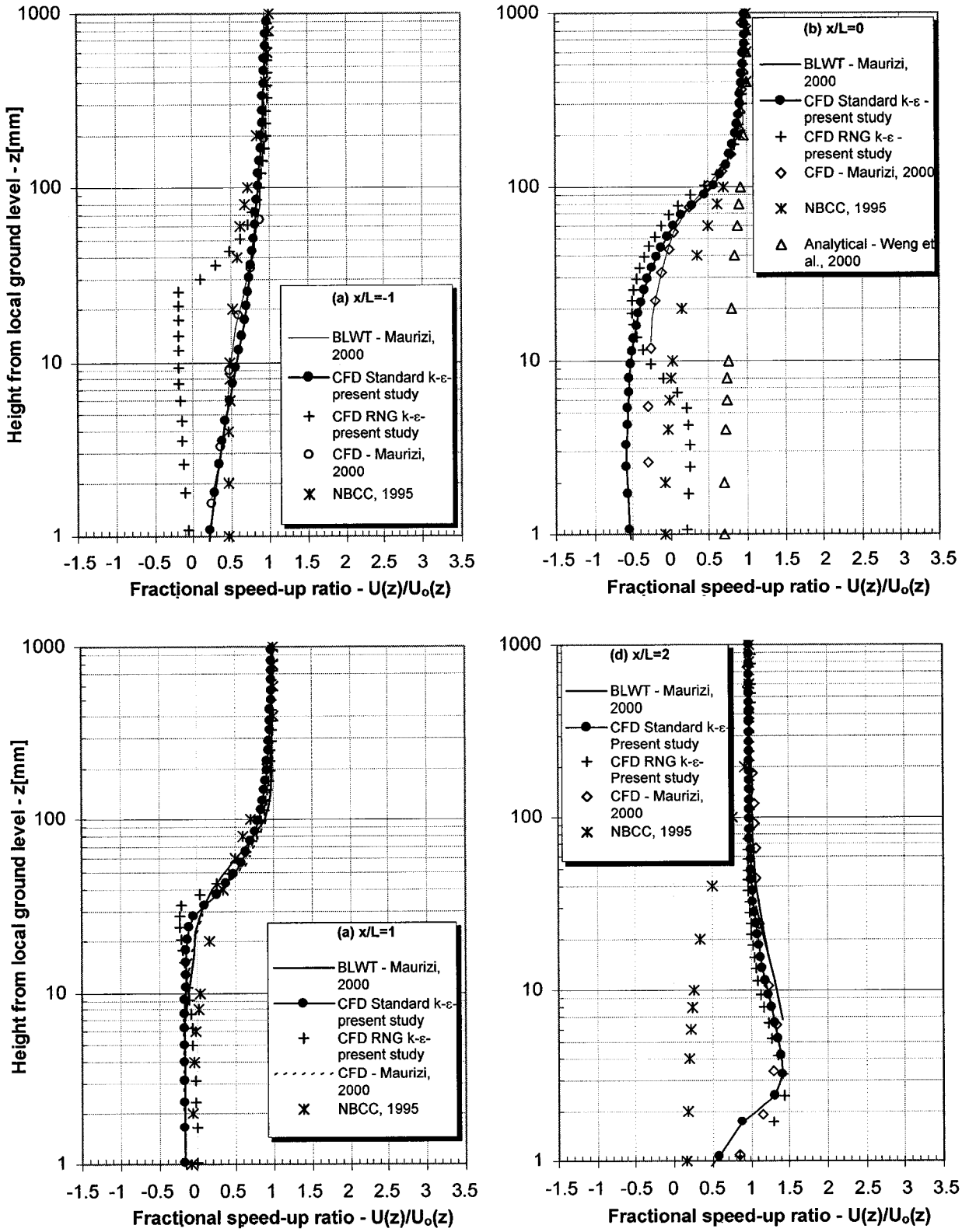


Figure 4.36 Comparisons between CFD, BLWT, and NBCC FSUR values for the deep valley (H/L =0.667) at four longitudinal locations.

5 Combined CFD-NN system for the prediction of FSUR values

5.1 Combined CFD-NN approach

Despite fewer resource requirements and favorable progress in hardware/software technology, present practical utilization of CFD-based numerical simulations to predict wind speed-up is rather limited. This is due to, among other reasons, the complexity generally associated with the CFD-based numerical methods which require background knowledge and experience, fast hardware with large memory, and availability of ample computational time. CFD simulations involve a number of tasks such as: finding an accurate geometrical representation of the terrain by means of good quality grid generation, dealing realistically with boundary conditions for each dependent variable, solving tens of thousands of equations (for two dimensional cases) iteratively for each dependent variable, using appropriate flow governing equations, making the selection of a suitable turbulence model and optimal relaxation parameters for each dependent variable.

Thus in parallel to developing such numerical tools, simple ways must be devised to convey the results produced by complex numerical simulations to the engineering profession (end-user). To this effect, this thesis proposes an integrated CFD-NN approach in which NN model is trained with CFD-generated data. Since the NN approach is defined on the basis of connections between the system state variables (input, internal, and output variables) with only limited knowledge of the "physical" behavior of the system, output variable values (speed-up ratio in the present case) can be produced following input of simple geometrical parameters describing the topography and roughness of the ground.

Neural Networks are data-driven models useful for mapping non-linear relationships that exist in wind flow data. In NN modeling two processes, i.e. learning and prediction/testing, are performed. During the learning process, the NN basically learns from a representative training dataset containing both influential input parameters and an output parameter. The NN "learns" how to map input parameters into an output parameter through an iterative process that involves optimization and other mathematical operations as shown later. For successful NN modeling, representative training data is compulsory. In the present work training data is generated using the developed CFD tool as shown in Figure 5.1, making the whole system less expensive. Thus CFD replaces the traditional role of BLWT experimental studies.

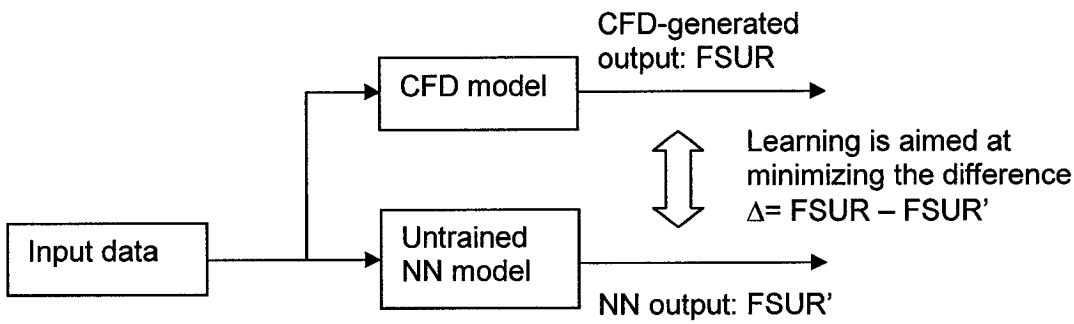


Figure 5.1 Learning in a combined CFD-NN approach as performed by domain expert.

The proposed and developed combined CFD-NN approach uses the following steps in order to produce a trained NN:

i. Training data is generated using the proposed and developed CFD tool. Terrain geometries similar to those used in previous CFD and BLWT studies are adopted.

ii. CFD-generated values are validated by comparison primarily with published BLWT and CFD data.

iii. NN is trained using CFD-generated training data.

iv. Results generated by the trained NN are validated by comparing with existing BLWT and CFD data.

Once the trained NN is validated, it is used to generate wind flow parameters for new cases either through interpolation or extrapolation. Note that, finally it is this trained NN that will be provided to the end user. The process flow during use of NN model is shown in Figure 5.2.

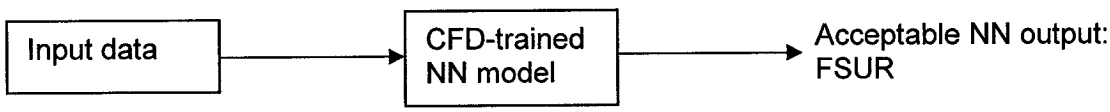


Figure 5.2 Using the combined approach by the end user.

In this combined approach, the domain expert carries out the complex numerical simulations and validations, thus ensuring the quality of results. Following this, the NN model is trained with CFD data generated once again by the domain expert and is available for use by the end user who avoids therefore dealing with complex numerical simulations. The proposed and developed combined CFD-NN approach is less expensive than the combined BLWT-NN approach for the end-user. The combined approach is also less expensive than the CFD simulations alone while producing results of comparable quality as shown later in this chapter. Thus combining these two approaches will result in significant practical benefits. The schematic shown in Figure 5.3 summarizes the development of the combined CFD-NN system. For this purpose nearly-orthogonal grid generator, CFD and NN tools have been designed and developed using an object-oriented approach and implemented using C++ programming language. Details of the grid

generator, CFD model and CFD tool development have been the subject of chapters 3 and 4. The next main component, namely NN model and NN tool development is discussed in detail in the following sections divided into two parts, development and performance evaluation.

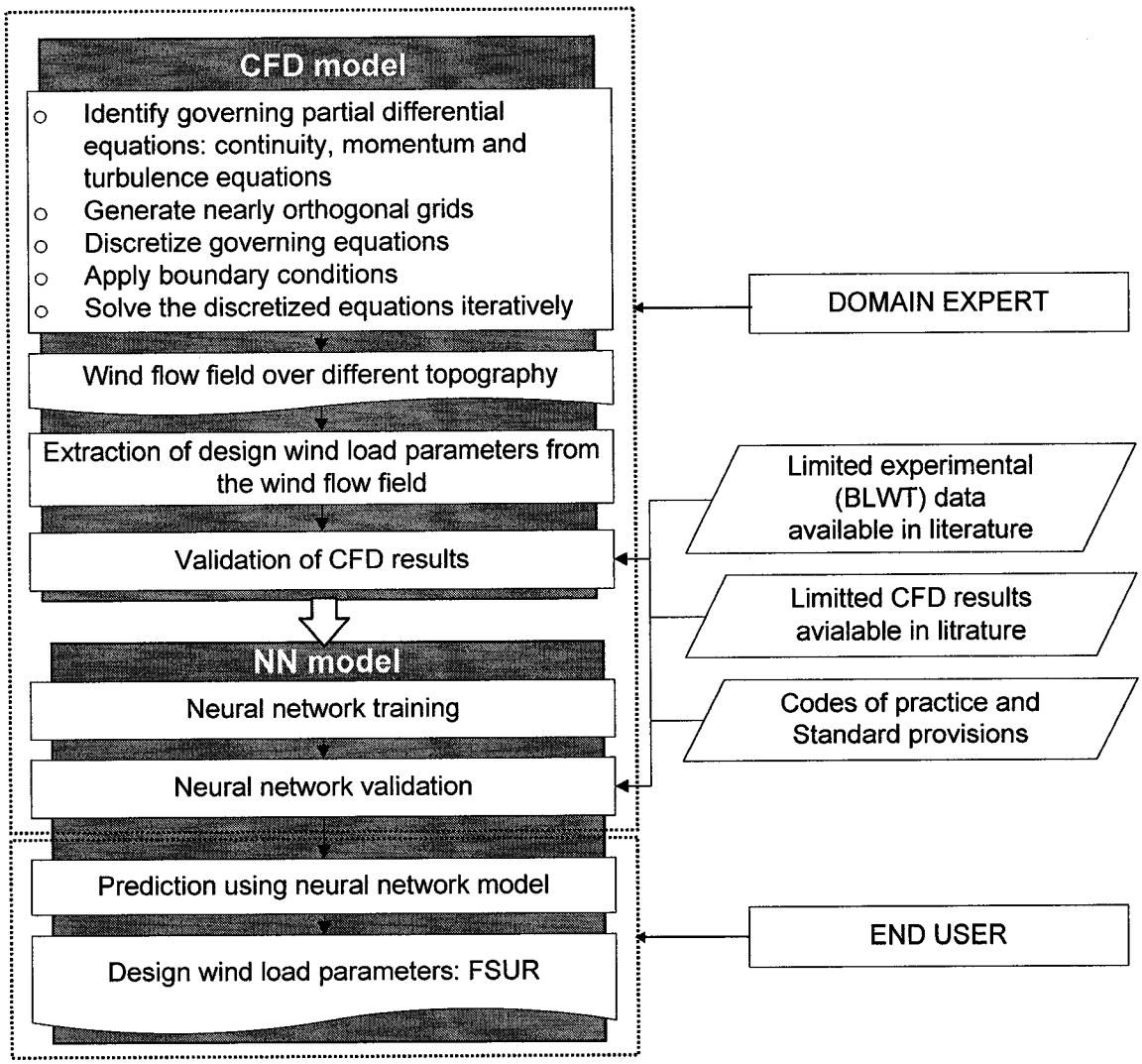


Figure 5.3 Schematic development of the proposed CFD-NN system.

5.2 Development of NN model for generating design wind load parameter

The most practical design criteria to build and train a neural network include: selection of internal error criterion, efficiency of learning as well as choice of network topology. Therefore care and time need to be taken to consider and make appropriate choice of internal error criteria, learning algorithms and optimum stopping criterion for maximum performance. In the present work, the neural network tool for prediction of FSUR values is developed based on cascade correlation algorithm of Fahlman and Lebiere (1990), using the object-oriented approach. This algorithm is chosen because of the following advantages:

- **Higher level of interconnection:** Besides the regular input to hidden layer, and hidden to output layer connections that exist in the standard back-propagation NN, connections exist also between hidden neuron to other hidden neurons as well as direct connections between input and output neurons. Thus each hidden neuron acts as a hidden layer, receiving inputs from input neurons and pre-existing hidden neurons. For instance, a standard back-propagation NN with architecture of 20 hidden, 5 input and 1 output neurons has a total 120 connections, whereas 315 connections exist in a Cascade-Correlation NN (CCNN) for the same number of neurons. This high level of interconnections in CCNN leads to the creation of very powerful higher order non-linear models. The knowledge of any type of NN is represented by the weights of the connections. The fact that input neurons are directly connected to output neurons enables to "project activities" forward by bypassing hidden layers. The result is that the training of the layers closer to the output becomes much more efficient.

- **Automatic determination of CCNN topology:** The most attractive capability of CCNN is determination of its own size by installing new hidden neurons during run-time as required. The architecture of a CCNN is shown in Figure 5.4. In this algorithm new hidden neurons are installed one at time during run-time as required from a pool of candidate hidden neurons that are trained separately in the background. Thus, for each new hidden neuron, the present algorithm tries to maximize the magnitude of the correlation between the new neurons output and the residual error signal of the CCNN (the mathematical details behind CCNN are discussed later). Installation of the new hidden neurons is automatically stopped when the network meets the error criteria or exceeds the maximum number of hidden neurons set by the user. This is unlike standard back-propagation algorithm in which the number of hidden neurons is determined by trial and error. Khanduri et al. (1995) reported that determining the architecture a priori by trial and error was one of the major problems faced while building a neural net work adviser for the effect of interference of buildings on wind load using standard back-propagation NN.
- **Incremental learning:** Another inherent characteristic of this algorithm is its capability to "freeze" some of the neurons during training. This prevents the neurons from behaving identically, thus producing a "moving target" problem. Connections shown with solid lines in Figure 5.4 are frozen once the hidden neurons are installed in the active network. The broken line connections however are trained repeatedly. This strategy allows the NN to learn faster as well.

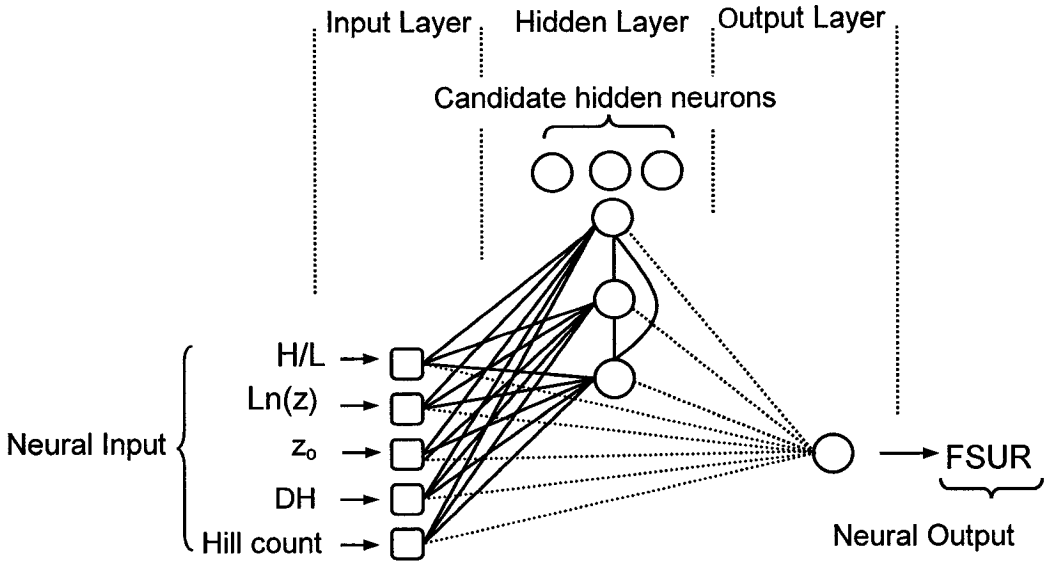


Figure 5.4 A NN with cascade-correlation architecture after three hidden neurons have been added. Squares represent non-processing neurons and circles represent processing neurons.

5.3 The CCNN training algorithm

CCNN have two types of neurons: processing (shown as squares in Figure 5.4) and non-processing (shown as circles in Figure 5.4). Input neurons are non-processing, i.e. their task is simply to distribute the input vector to the hidden and output neurons, whereas all other neurons (hidden and output neurons) are processing neurons. The processing neurons make use of two mathematical operations: summation of net weighted inputs (X given by equation 5.1) and execution of activation function (Y given by equation 5.2). The input to each hidden or output neuron is the sum of all its incoming connection weights multiplied by their respective connecting neural values. The result is fed into the activation function of the neuron to yield the desired output as shown in Figure 5.5. The activation function may have many forms such as linear, threshold, sigmoid, or hyperbolic. In the present study sigmoid activation function has been used, as given by

equation (5.2), mainly due to its continuous differentiability, squashing and normalizing effect. The average firing frequency of biological neurons as a function of excitation follows sigmoid characteristics (Schalkoff 1997) as well.

$$X_j = \sum_{i=1}^{N} I_i W_{ji} \tag{5.1}$$

$$Y_j = \frac{1}{1+e^{-X_j}} \tag{5.2}$$

where X_j is net weighted input for neuron j, W_{ji} is the connection weight between neurons i and j, Y_j is calculated output at neuron j, I_i is the input vector i.e. neural value of the i^{th} connecting neuron, N is the number of connecting neurons, j is the destination neuron for which the activation is being computed, i represents the connecting neurons. A detailed representation of a typical processing neuron is shown in Figure 5.5

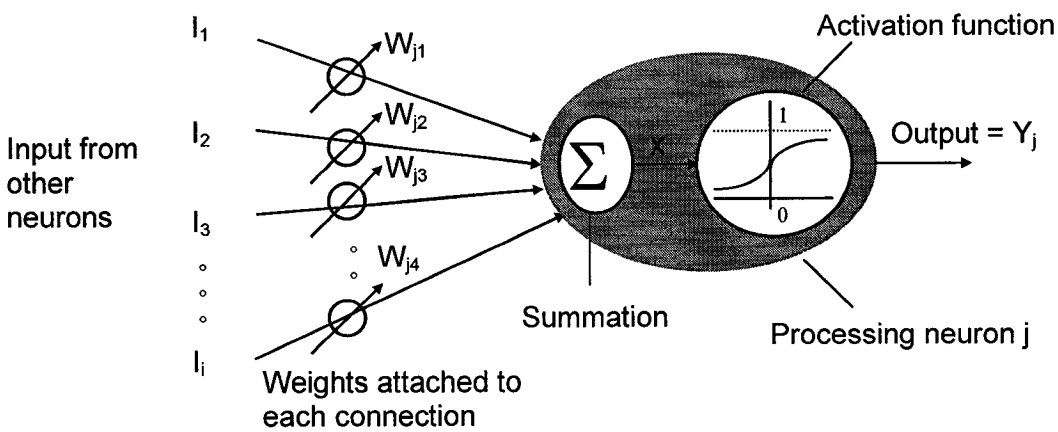


Figure 5.5 A typical processing neuron.

The first step of NN model development is learning (or training). Learning is a crucial step in neural network application. The learning procedure extracts the information from the input data with the help of the desired response. In doing so the size of the neural network (number of hidden neurons) and the value of the associated connection weights

(W_{ji}) are determined. While the CCNN algorithm determines the size automatically during run-time, the connection weights to the output and hidden neurons are adjusted iteratively using the following mathematical equations.

The weights connected to the output neuron are adjusted by minimizing the sum squared error E_{pk} given as:

$$E_{pk} = 0.5\sum_{k=1}^{no}(T_{pk} - Y_{pk})^2 \tag{5.3}$$

where T_{pk} is the target output (given as part of the training data set), Y_{pk} is the calculated output at output neuron k for a training data set p and "no" is number of output neurons. The error (E_{pk}) is minimized by gradient decent method:

$$\Delta W_{kj} \propto -\frac{\partial E_{pk}}{\partial W_{kj}} \tag{5.4}$$

where ΔW_{kj} represents the adjustment in the weight connecting neuron j and destination neuron k. Application of this method requires slope calculation and further adjustments of the weights using delta rule (based on the slope).

Slope calculation:

To find usable difference form of the slope of the error function in equation 5.4 first chain rule is applied as follows:

$$\frac{\partial E_{pk}}{\partial W_{kj}} = \frac{\partial E_{pk}}{\partial X_k}\frac{\partial X_k}{\partial W_{kj}} \tag{5.5}$$

$$\frac{\partial E_{pk}}{\partial W_{kj}} = \frac{\partial E_{pk}}{\partial X_{pj}} * \frac{\partial \sum Y_{Pj} W_{kj}}{\partial W_{kj}} \tag{5.6}$$

$$\frac{\partial E_{pk}}{\partial W_{kj}} = \frac{\partial E_{pk}}{\partial X_{pj}} * Y_{pj} \tag{5.7}$$

Defining error signal δ_{kj} as

$$\delta_{kj} = -\frac{\partial E_{pk}}{\partial X_{pj}} \tag{5.8}$$

Once again applying chain rule gives

$$\delta_{kj} = -\frac{\partial E_{pk}}{\partial Y_{pk}} * \frac{\partial Y_{pk}}{\partial X_{pj}} \tag{5.9}$$

The two derivatives in equation (5. 9) are obtained as follows:

$$\frac{\partial E_{pk}}{\partial Y_{pk}} = \frac{\partial \sum (T_{pk} - Y_{pk})^2}{\partial Y_{pk}} \tag{5.10}$$

$$\frac{\partial E_{pk}}{\partial Y_{pk}} = -(T_{pk} - O_{pk}) \tag{5.11}$$

The second derivative in equation (5.9) is obtained using the selected activation function (given in equation 5.2) as follows

$$\frac{\partial Y_{pk}}{\partial X_{pk}} = \left(\frac{1}{1+e^{-X_{pk}}}\right)\left(1 - \frac{1}{1+e^{-X_{pk}}}\right) = Y_{pk}(1 - Y_{pk}) \tag{5.12}$$

Thus the error signal has the following form

$$\delta_{pk} = (Y_{pk} - T_{pk}) Y_{pk}(1 - Y_{pk}) \tag{5.13}$$

Finally the slope of the error function is written as

$$\frac{\partial E_{pk}}{\partial W_{kj}} = (Y_{pk} - T_{pk}) Y_{pk}(1 - Y_{pk}) Y_{pj} \tag{5.14}$$

where Y_{pj} and Y_{pk} are the neural values of the connecting neuron j and output neuron k for a training data set p respectively, T_{pk} is the target output (given as part of the training data set) and W_{kj} represents the weight of the connection between a connecting neuron k and an output neuron j.

Similarly the weights connected to the hidden neuron are adjusted by maximizing the correlation (R) between the output neural value Y_p of a hidden neuron and the error (E_{pk}) of an output neuron. The correlation is given by Fahlman and Lebiere (1990) with:

$$R_p = \sum_{k}^{no} \left| \sum_{p}^{nd} Y_p \left(E_{pk} - \overline{E_k} \right) \right| \tag{5.15}$$

where Y_p is the activation of a hidden candidate neuron, $\overline{E_k}$ is the average error of an output neuron k over all patterns p, "no" is the number of output neurons and "nd" is the size of training data set. The maximization of R_p proceeds by gradient ascent. A similar derivation procedure as used for the error function slope and error signal of output neuron is also used to obtain the following forms of the correlation function slopes and error signals for the hidden neurons.

$$\frac{\partial R_p}{\partial W_i} = \sum_{p} \delta_p Y_{pi} \tag{5.16}$$

where

$$\delta_p = \sum_{k} \sigma_k \left(E_{pk} - \overline{E_k} \right) Y_p (1 - Y_p) \tag{5.17}$$

Y_{pi} is the neural value of the connecting neuron i for a training data set p, Y_p is neural value of the hidden neuron for a training data set p, σ_k is the sign of the correlation between the hidden neuron's output and the residual error at output k.

Delta rule:

During the training phase the weights, both for hidden and output neuron connections, have been adjusted (adopted) using delta rule given as

$$W_{ji}(n) = W_{ji}(n-1) + \Delta W_{ji}(n) \tag{5.18}$$

where n represents the iteration number. The $\Delta W_{ji}(n)$ in equation (5.17) is estimated using the QUICKPROP learning algorithm developed by Fahlman and Lebiere (1990). QUICKPROP is a second order method based loosely on Newton's method given by:

$$\Delta W_{ji}(n) = \begin{cases} \theta S_{ji}(n) & \text{if } \Delta W_{ji}(n-1) = 0 \\ \dfrac{S_{ji}(n)\Delta W_{ji}(n-1)}{S_{ji}(n-1) - S_{ji}(n)} & \text{if } \Delta W_{ji}(n-1) \neq 0 \text{ and } \dfrac{S_{ji}(n)}{S_{ji}(n-1) - S_{ji}(n)} < \Omega \\ \Omega \Delta W_{ji}(n-1) & \text{otherwise} \end{cases} \tag{5.19}$$

where $S_{ji} = \partial E / \partial W_{ji}$ (given by equation 5.14) for adjusting weights connected to the output neurons and $S_{ji} = \partial R / \partial W_{ji}$ (given by equation 5.16) for adjusting weights connected to the hidden neurons, θ represents learning rate that governs the distance traveled in the direction of the negative gradient (or positive gradient while training hidden neurons) when a step in weight space is taken, and Ω represents maximum growth factor. The difference of first partial derivatives has been used in equation (5.19) in an attempt to approximate the second partial derivative of $\partial^2 E / \partial W_{ji}^2$. In addition $|\Delta W_{ji}(n)|$ has been controlled not to exceed $\Omega |\Delta W_{ji}(n-1)|$ in order to prevent single, large weight changes that could negate many epochs of previous learning.

To summarize, training the CCNN model involves the following steps:

i. Once the training data is prepared and the input and output parameters are identified, start training CCNN with a minimal network consisting only of input and output neurons, as shown in Figure 5.6 (a).

ii. Train all the connections ending at an output neuron with the QUICKPROP learning algorithm (given by equation 5.19) until the error of the CCNN no longer decreases or is less than a threshold value specified by the user.

iii. Generate the so-called candidate hidden neurons. Every candidate-hidden neuron is connected with all input neurons and with all existing hidden neurons. The difference among them lies in their weights, which have been initialized to different randomly generated values $(W_{ij})_1$, $(W_{ij})_2$, $(W_{ij})_3$, for candidate hidden neuron-1, -2 and -3 respectively. Between the pool of candidate neurons and the output neurons there are no weights. An example of three candidate neurons and their connections is shown in Figure 5.6 (b). These candidate neurons are trained just before installing a second neuron in the active CCNN.

iv. Maximize the correlation between the activation of the candidate neurons and the residual error of the CCNN, equation (5.15), by training all the links leading to a candidate neuron. Once again the learning takes place with the QUICKPROP learning algorithm. The training is stopped when the correlation score no longer improves.

v. Choose the candidate neuron with the maximum correlation, freeze its incoming weights and add it to the CCNN as shown in Figure 5.6 (c). To change the candidate neuron into a hidden neuron, generate links between the selected

neuron and all output neurons, as shown in Figure 5.6 (d). Since the weights leading to the new hidden neuron are frozen, a new permanent non-linear model is obtained. Loop back to step ii until the overall error of the CCNN falls below a user-specified error threshold, or the maximum hidden neurons criteria is satisfied.

During testing/use, only a single forward pass through the CCNN using the trained weights will generate the desired output.

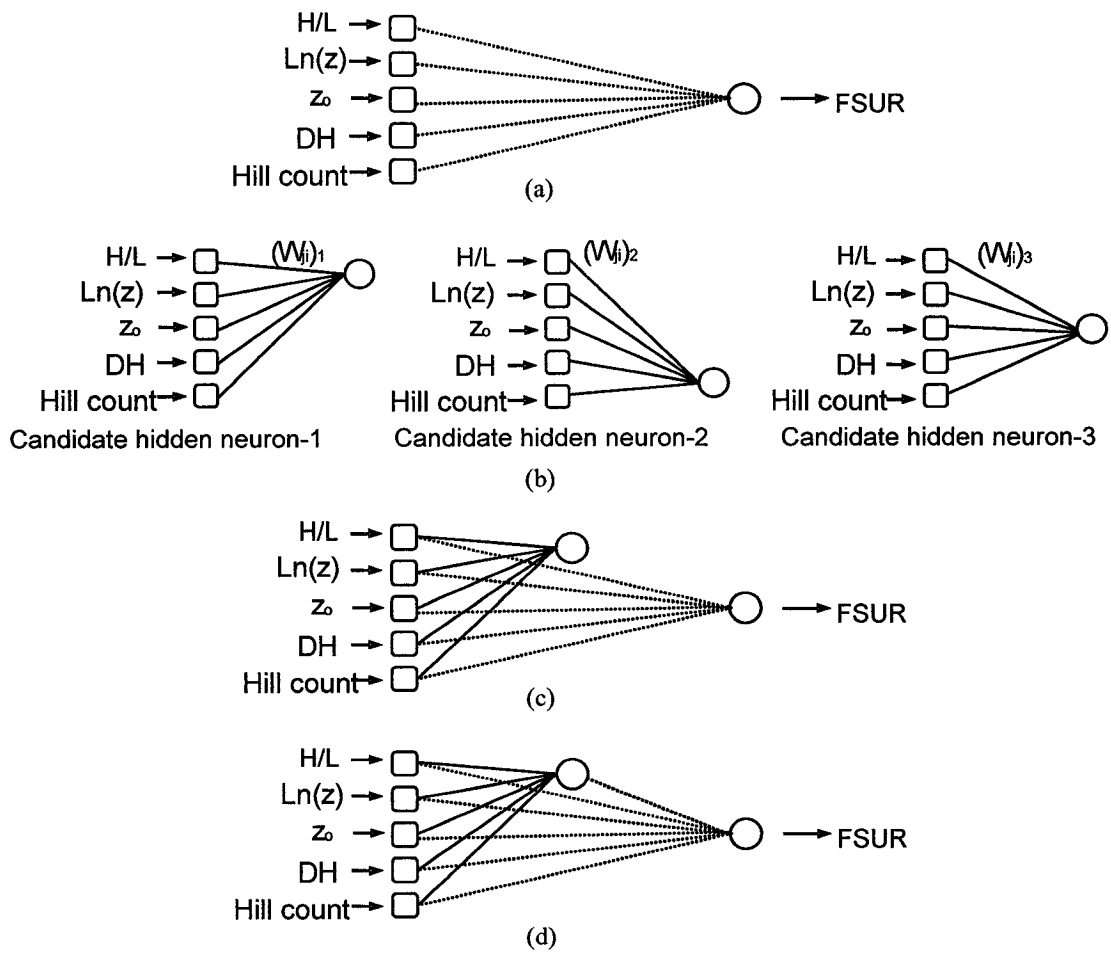


Figure 5.6 CCNN learning process: (a) train weights from input to output, (b) few candidate hidden neurons trained separately from the active CCNN, (c) best candidate hidden neuron installed in the active network, (d) retrain output layer weights.

5.4 NN tool development in object-oriented computing environment

Object-oriented approach has been adopted for the development of the NN tool. As previously mentioned the procedure in object-oriented tool development includes detail description of the problem statement (analysis), identifying potential classes from the problem statement, designing of each class and their member functions, and implementation of the design using a programming language that supports object-oriented approach such as C++. In the following paragraphs, therefore, such details are presented as well as process flow during training and testing.

5.4.1 Problem statement

The objective is the development of NN tool that will be used to create CCNN models to predict FSUR values based on simple geometrical and roughness parameters of the terrain. As discussed in the previous sections, the following major tasks are necessary:

- create input neurons and output neurons and connect them;
- generate weights on the connections;
- obtain the activation functions;
- estimate the error;
- adjust the weights so that the activation is similar to the target output of the training data;
- create a pool of candidate hidden neurons;
- find the correlation between the hidden neuron activation and the estimated error;
- train the hidden neurons to give maximum correlation with the error using QUICKPRO learning algorithm (equation 5.9);
- install the most correlated candidate hidden neuron into the active CCNN

- architecture; train the output neuron using QUICKPRO learning algorithm;
- estimate the new error;
- if the error is larger than a user-specified threshold error, continue installing a new hidden neuron from a pool of candidate neurons and continue the whole process;
- store the connection weights.

5.4.2 Class interface definition

The architecture of CCNN has different layers, as shown in Figure 5.4, consisting of a number of neurons on each layer. However in cascade-correlation architecture, a hidden neuron acts more like a hidden layer than a single neuron, receiving inputs from connecting input neurons and all previously created hidden neurons. Therefore in the present study layers have been selected as the basis and two classes named OutputLayer and HiddenNeuron, have been adopted. With regard to the input layer, there are no mathematical operations carried out since the input neurons simply distribute the input parameter to the output and hidden neurons. As a result, input neurons have been represented by simple two dimensional arrays (to store the number of neurons and input data) as opposed to user-defined objects like HiddenNeuron and OutputLayer objects. The interface definition tables for the HiddenNeuron and OutputLayer classes are shown in Tables 5.1 and 5.2 respectively. Also the C++ implementation is given in Appendices A-5 and A-6 for the HiddenNeurons and OutputLayer classes respectively.

5.4.3 Sequence diagrams

A sequence diagram describes the interaction between the HiddenNeuron and OutputLayer objects as well as the execution of sequence of each member function. This is shown in Figures 5.7 and 5.8 for training and testing/use cases respectively. The Loop-

1 shown in Figure 5.7 relates to the installation of new hidden neuron. It terminates either when the maximum number of hidden neurons is exceeded or the minimum threshold error specified by the user is satisfied whereas Loop-2 and Loop-3 in Figure 5.7 represent the iteration required for training the hidden neuron and output neuron weights respectively. Loop-1 in Figure 5.8 relates to the number of hidden neurons.

Table 5.1 HiddenNeuron class interface definition

Member functions	Arguments	Returns	Purpose
Construction	Input and output neuron number, data size, output activation		Creates an instance of an output layer class and initializes the parameters indicated as arguments for the object
randWminus			Randomize weights connected to the hidden neuron
readWminus	File name		Reads weights connecting to the hidden neuron from file
saveWminus	File name		Saves weights connected to the hidden neuron in a file
displayWminus			Displays weights connected to the hidden neuron
computeO			Computes output neural value
adjustWminus			Computes slopes and adjusts weights connected to the output neuron using Quickprop to minimize the error
displayH			Displays hidden neural value during run time
displayT			Displays T during run time
computeE			Computes the mean square error
returnH		Hidden neural value	Returns hidden neural value
returnWminus		Wminus*	Return hidden output activation

* Wminus represents weights connected to hidden neurons

Table 5.2 OutputLayer class interface definition.

Member functions	Arguments	Returns	Purpose
Construction	Training data, input and output neuron size		Creates an instance of an output layer class and initializes the parameters indicated as arguments for the object
randWplus			Randomize weights connected to the output neuron
getoldWplus	Previous weights connected to the output neuron		Assign weights connected to the output neuron with previously trained weights to retrain them together with new Wplus formed after addition of a new hidden node to the network.
displayWplus			Displays weights connected to the output neuron
computeO			Computes output neuron activation
adjustWplus			Computes error signals and adjusts weights connected to the output neuron using QUICKPROP
displayO			Displays output neural value during run time
returnO		Output neural value	Returns output neural value
computeE			Computes the error
returnavgE		Average error	Returns average error to be used in main
displayE			Displays the error during run time
returnWplus		Wplus*	Return Wplus to be stored globally for use after addition of a newly created hidden node to the network
StoreO			Stores output neural values

* Wplus represents weights connected to output neurons

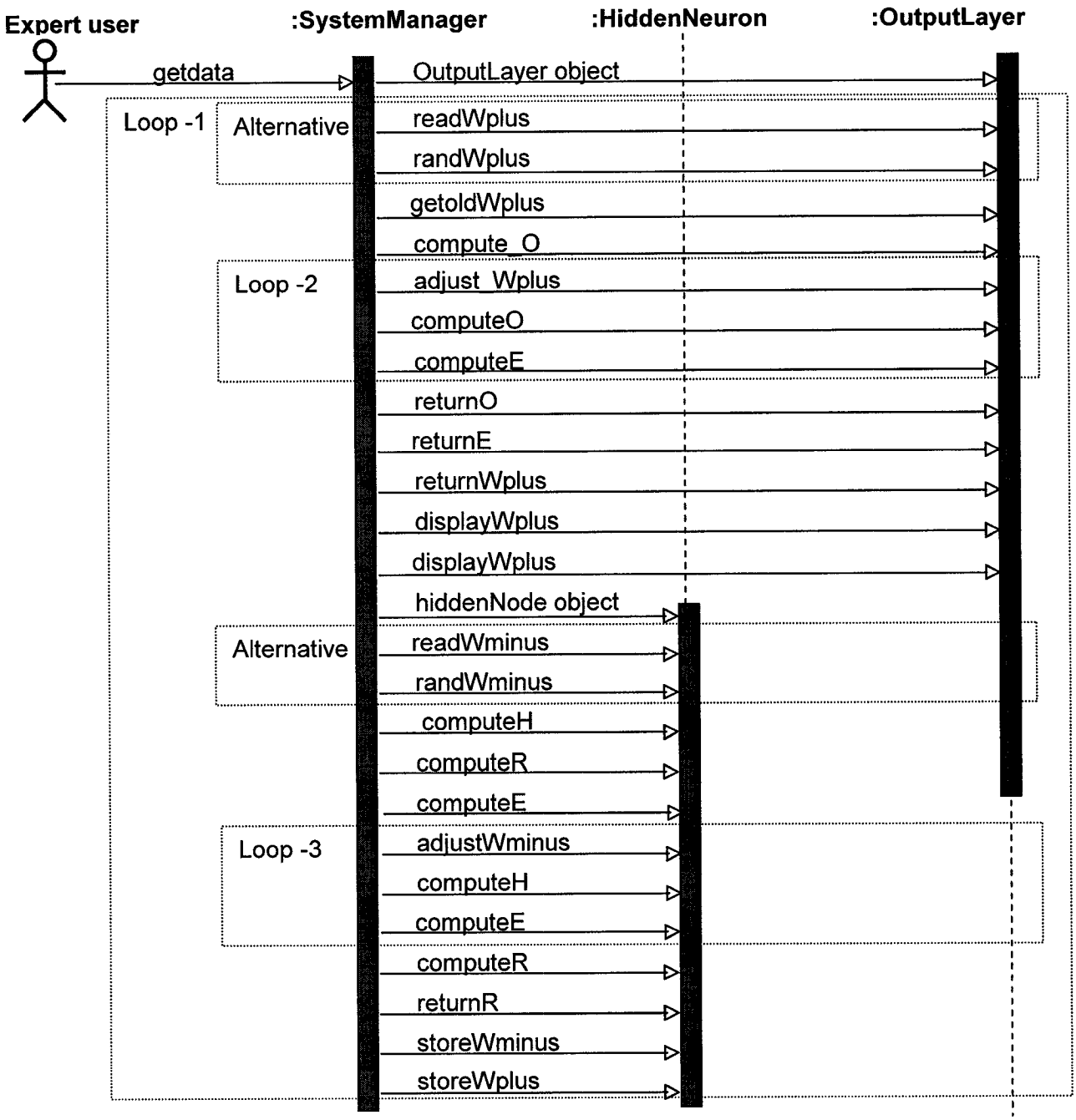


Figure 5.7 CCNN training sequence diagram.

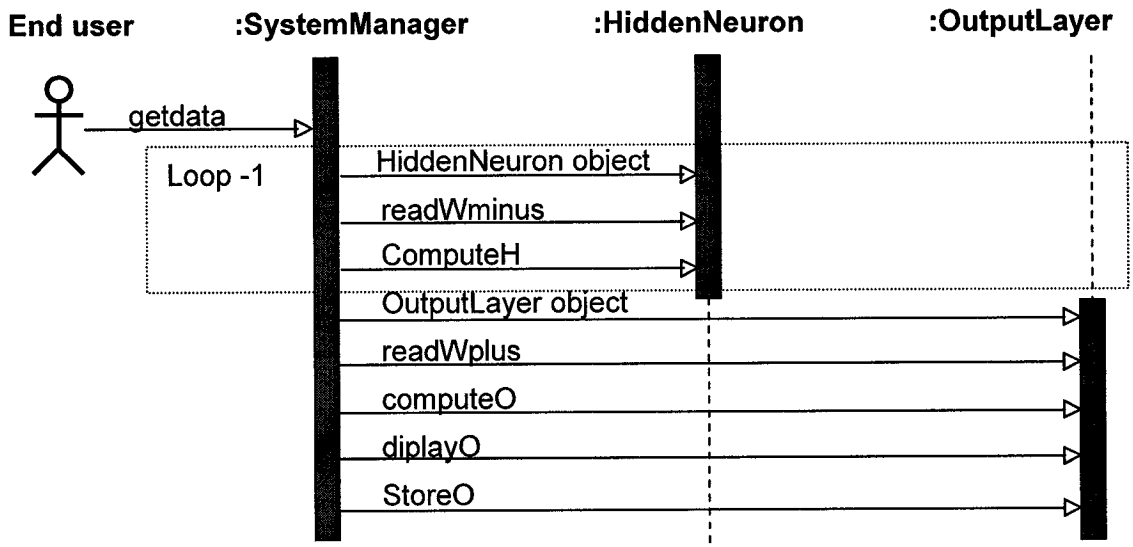


Figure 5.8 CCNN testing/use sequence diagram.

5.5 Training of NN model

5.5.1 Training data description

Representative training data is crucial for successful application of CCNN models. Finding data that represents the different scenarios in a problem is the major challenge in CCNN applications for engineering purposes in general and wind engineering in particular. This requires considerable efforts as well as resources. Since the CCNN learns relationships directly from training data, one has to ensure the quality prior to their use. In the present study, training data is generated using the CFD tool as described in the previous chapters, thus making the whole procedure a numerical process. Generally, this approach requires fewer resources as compared to generating training data using BLWT experiments. In fact for the end user, it is also faster, simpler and less expensive than running the CFD simulations alone. With the combined CFD-NN approach, the end user is provided with a trained CCNN.

5.5.2 Input/Output parameter identification

Sets of input and output parameters are required to train the neural network. As much as possible the input variables should include all the parameters that are known to influence the output parameter or assist the CCNN model in differentiating between different scenarios. In the present case, the output parameter is the FSUR value which depends on several parameters. In order to train and test the CCNN, it is therefore necessary to develop data sets consisting of FSUR values in relation to the various relevant input parameters. After thorough investigation and with the help of the knowledge gained from CFD simulations and from experience, the following parameters are selected as input to the CCNN:

i. Geometrical parameters (see Figure 5.9):
    - Windward slope of the hill: H/L.
    - Height from the crest of the hill at which FSUR values are to be determined: z. Since variations are significant closer to the ground, a logarithmic scale has been adopted instead of a normal scale.
    - Longitudinal location (only for the case of valleys where FSUR values at longitudinal locations other than the crest of the hill are required): x/L
    - Distance between hills in case of multiple hills: DH.

ii. Roughness length of the ground: z_o.

iii. Hill count. In the case of multiple hills, a parameter that enables the CCNN to identify each hill is required. For this purpose an arbitrary hill count coefficient has been defined in the following manner: (1,0,0), (0,1,0) and (0,0,1) for upstream, middle and downstream hills of the triple hills case respectively.

Input and output parameters are also shown in Figure 5.4. Typically, three input neurons for receiving the three input parameters (H/L. z_o, ln(z)) have been used in the input layer (in case of multiple hills five input neurons for receiving (H/L. z_o, ln(z), DH, hill count) have been used. The expected output from the CCNN is only the FSUR value; hence a single output neuron has been used in the output layer. The number of hidden neurons is determined during run time automatically. In the present case the maximum number of hidden neurons has been set to 15. This has been imposed in order to prevent over-fitting problems associated with large size CCNN that will lead to poor generalization capability during use. Simplifying the information presented to the CCNN enhances its performance, and helps it to learn the relation ships among the data more effectively.

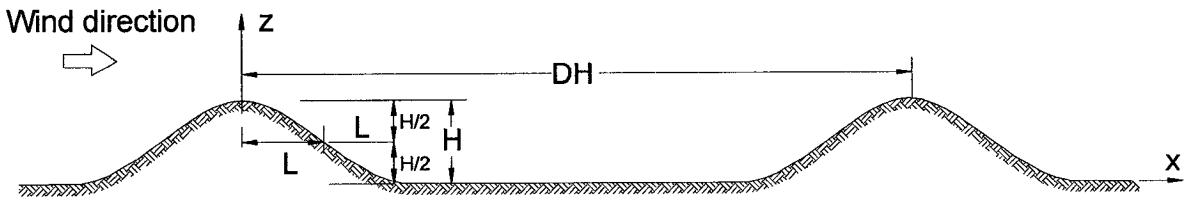


Figure 5.9 Basic geometrical parameters of hills used as input during training the CCNN.

5.5.3 CCNN model preparation

Several CCNN models have been prepared for prediction of design wind load parameter, i.e. FSUR value, pertaining to different kinds of terrains. As mentioned before, the CCNN model has been trained with data obtained from the proposed CFD code described in the previous chapters. The available limited BLWT data in literature regarding flow over terrains has been used for testing the neural network model as discussed in the results and discussion section. The CCNN model begins learning with direct input-output connections and continues installing new hidden neurons during run-time until the learning threshold is satisfied. The learning threshold stipulates the allowable mean square error specified by the user to stop training. In most cases of CCNN model training, a learning threshold of 0.00001 has been satisfied after installation of 11 hidden nodes. Training usually takes less than couple of minute using a 3.6 GHz Pentium III processor with 500 MB RAM under Windows 98. Bank of CCNN models trained for isolated hill, multiple hills, valley, and escarpment separately, shown in Figure 5.10, are stored in file. During testing or use, the end user will be queried to identify the geometry so that the proper CCNN model will be recalled and used from the multiple CCNN bank.

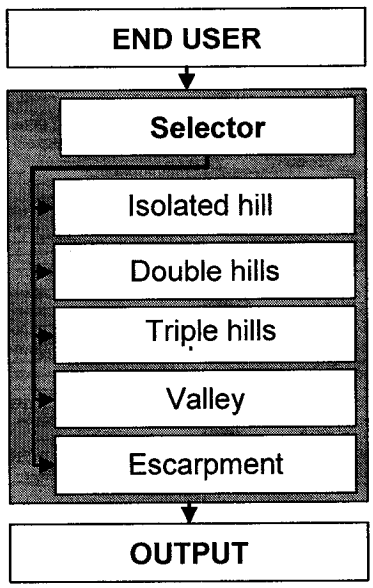


Figure 5.10 Schematic of the trained CCNN model types proposed to the end user.

5.6 CFD-NN results and discussions

Once the training job is completed, the trained CCNN has been tested first for reproducing the CFD results of the same case, and then for predicting FSUR values for new terrain geometries that have not been considered during the training phase. This reproduction test has been carried out for hills, valleys and escarpments. As one may expect this exercise was not challenging to the CCNN, which reproduced the CFD data exactly with R-square value of 1. It has also shown good agreement with BLWT experimental data (detail results are not shown in this paper). However this simple exercise proves that a CCNN can reproduce, to the end user, CFD-like results using simple input parameters and mathematical computations. In the following paragraphs, the developed CCNN generated FSUR values for different scenarios are compared with the BLWT data thus enabling the performance evaluation of CCNN models. Only the results for new cases are presented, i.e. whose geometries have not been used during training.

5.6.1 Isolated hill

In this category a CCNN model trained with CFD-data for hills with slopes H/L=0.5 and 0.75, and roughness length (z_o = 0.02m at full scale), has been used to predict FSUR values for new steep hill with slope H/L=1 and similar roughness length. The predicted FSUR ratio values are consequently compared with Carpenter and Lock (1999) BLWT data. Figures 5.11 (a) and (b) show a comparison of FSUR values produced by the CCNN model with BLWT experimental data and NBCC (1995) provision respectively. Although the agreement in both cases is satisfactory, relatively better agreement has been observed between the FSUR values produced by CCNN and BLWT experiments than NBCC (1995) and BLWT experiment. This case underlines the extrapolation capability of

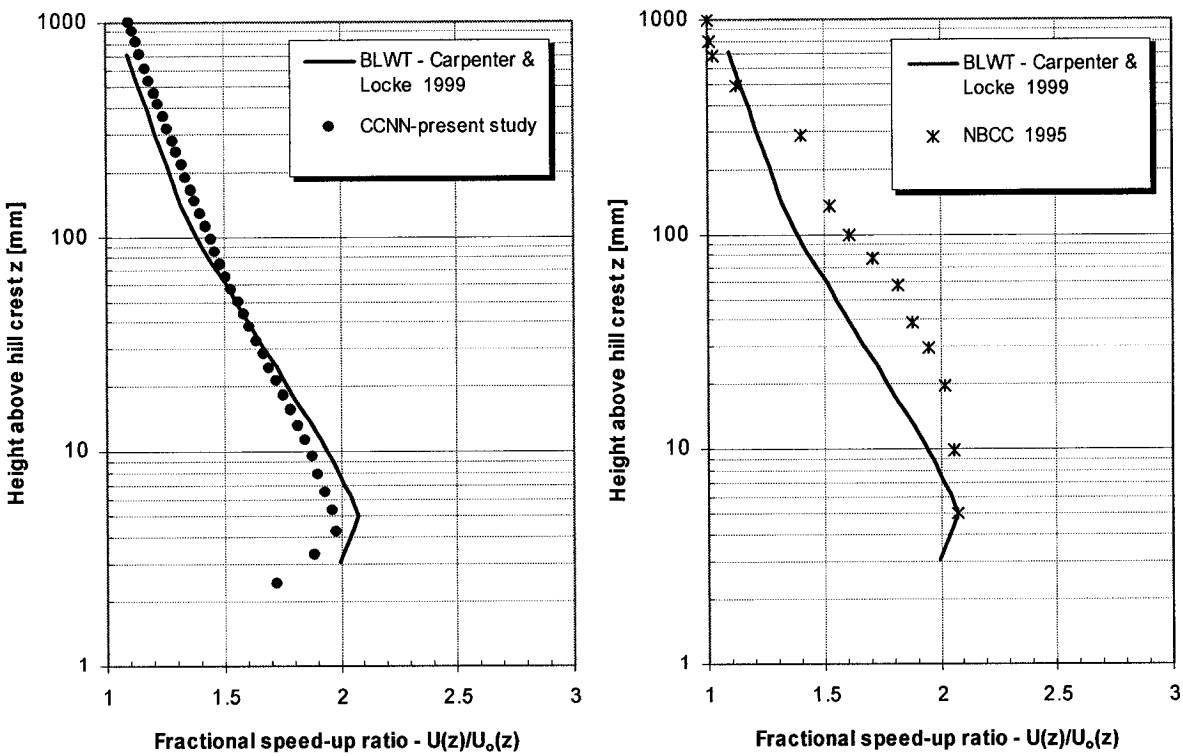


Figure 5.11 Isolated hill - Comparisons of FSUR values at the crest of steep hill (H/L=1) obtained using (a) CCNN and BLWT, (b) NBCC and BLWT.

CCNN as well. Note that all evaluations in the present study have been made at the BLWT model scale. In this case the MSE error criterion (0.00001) has been achieved after installing 7 hidden neurons.

5.6.2 Triple hills

In this category, the evaluation is carried out for a new case of intermediate triple hills (H/L=0.66, distance between hills 6H) that has not been considered during the training phase, as shown in Figure 5.12. CFD data generated and validated for shallow (H/L=0.5) and steep (H/L=1) triple hills has been used to train the CCNN. Distances between hills equal to 8H and 6H have been used in the training data. Both smooth (z_o=0.02m) and rough (z_o=0.16m) hill surfaces are considered in the training data as well. CCNN with 7 hidden neurons has been used. The predicted FSUR values are consequently compared with Miller (1996) BLWT data as shown in Figure 5.12. The CCNN FSUR values agree very well with the BLWT data and they capture as well the decrease in FSUR values for the downstream hills as shown in Figure 5.13 (a). The decrease in FSUR values for the downstream hills is due to the wind flow separation from the upstream hill, which results in increased turbulence on the downstream hills. This in turn causes the flow to attain more uniformity resulting in a reduction of the peak FSUR values just above the crests of the downstream hills. R-square values between CCNN and BLWT FSUR values equal to 0.99, 0.99 and 0.96 have been obtained for Hill-1, Hill-2 and Hill-3 respectively. Figure 5.13 (b) shows comparisons between NBCC and BLWT FSUR values for the same geometries. Although the NBCC FSUR values agree with the BLWT data for the upstream hill (Hill-1), their use for the downstream hills over-predicts FSUR values. The use of NBCC-1995 provisions yields R-square

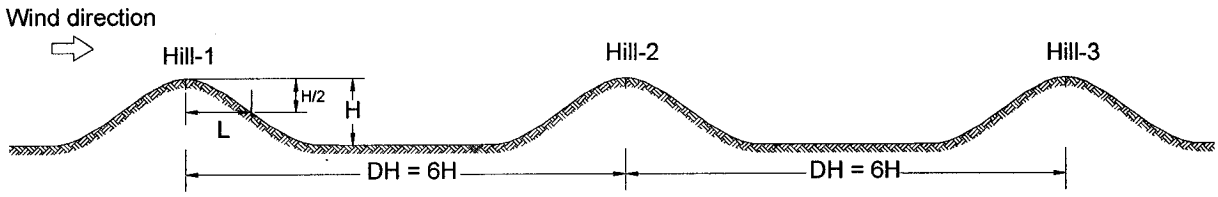


Figure 5.12 Triple hills - Geometry used by the present as well as Miller's (1996) studies (H/L=0.66).

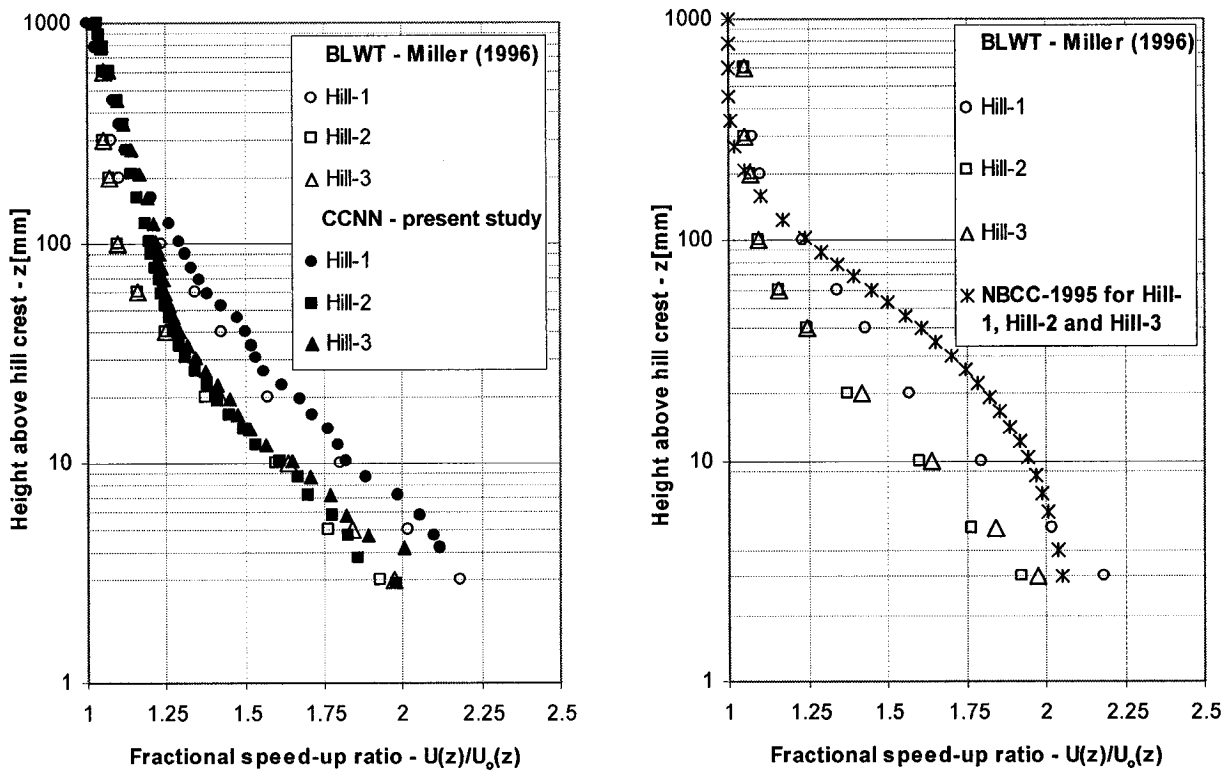


Figure 5.13 Comparisons of FSUR values at the crest of triple hills (H/L=0.66) obtained using (a) CCNN and BLWT, (b) NBCC and BLWT.

values equal to 0.92, 0.85 and 0.86 for Hill-1, Hill-2 and Hill-3 respectively. The CCNN-BLWT R-square values are better than NBCC-BLWT R-square values by 7.5%, 14% and 11% for Hill-1, Hill-2 and Hill-3 respectively. This shows that a significant improvement in the accuracy of FSUR values can be achieved by using the combined CFD-NN approach. It also indicates that CCNN model, if trained properly, not only reproduces the CFD result accurately but also expands the information available for the end user by generating reliable FSUR values for new hill geometries.

5.6.3 Escarpment

For this category, CCNN results are validated using field measurement (Holmes et al. 1997) and BLWT experiments (Glanville and Kwok 1997). The geometrical and roughness parameters are similar to those used by Glanville and Kwok (1997) BLWT study, i.e. H = 400 mm, L = 400 mm and full-scale roughness length of 0.02 m. The CCNN model trained with CFD-data for escarpments with slopes H/L=0.25 and 0.75, and roughness length (z_o = 1.5 mm), has been used to predict FSUR values for the new escarpment with slope H/L=1 with similar roughness length. Figure 5.14 shows comparisons between the present CCNN, field study, and BLWT FSUR values. The agreement between CCNN and BLWT is satisfactory except for values very close to the ground. Holmes et al. (1997) field measurements only at three points close to the ground are located between the CCNN and the BLWT values. NBCC (1995) provisions underpredict the FSUR values.

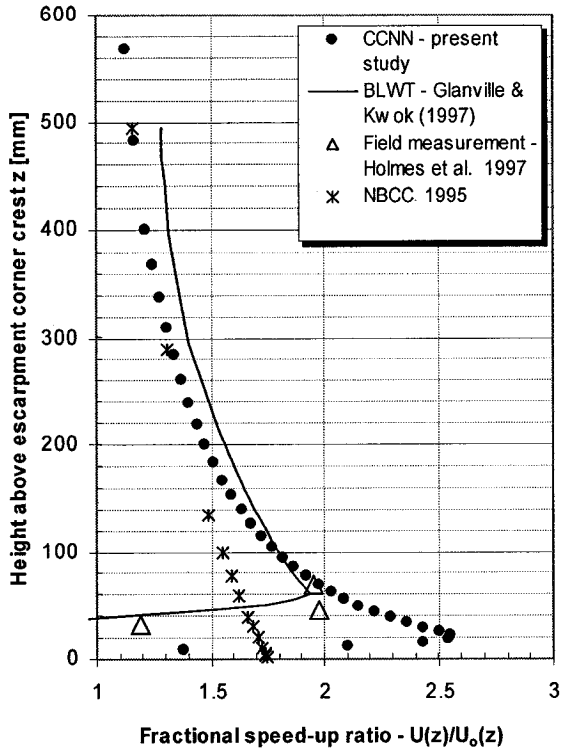


Figure 5.14 Escarpment - Comparison between FSUR values obtained from CCNN, BLWT, field study and NBCC above the corner crest of an escarpment.

5.6.4 Valley

In this category, CCNN model predictions for a deep valley at different longitudinal locations are presented. The geometry and roughness parameters are similar to those used in Busuoli et al. (1993) and Chung and Bienkiewicz (2004) BLWT experiments. The geometry is shown in Figure 5.15. The ground roughness used is 0.016 m in full scale. The CCNN model trained with CFD-data for valleys with slopes H/L=0.25, 0.5 and 1, and roughness length (z_0 = 0.016m at full scale) has been used to predict FSUR values for the new valley with slope H/L=0.667 and similar roughness length at different longitudinal locations shown in Figure 5.15. For this case, an additional input parameter (x/L) has been used, in addition to those used previously (H/L, ln(z), z_0), in order to

enable the CCNN to distinguish between the different longitudinal locations shown in Figure 5.15. The CCNN FSUR values are in a good agreement with the BLWT data as well as Maurizi (2000) CFD results as shown in Figures 5.16 (a) to (d). Further FSUR values are calculated for the valleys applying the NBCC provisions (1995). Although the comparison of NBCC FSUR values with numerical and experimental data is satisfactory at longitudinal locations x/L = 1 and x/L = 2 as shown in Figures 16 (c) and (d), it is less satisfactory at the center of the deep valley (x/L= 0) as well as at x/L = -1 as shown in Figures 5.16 (b) and (a) respectively. The overestimation of NBCC (1995) may originate from the theoretical models on which it is based and their inability to represent separating flow.

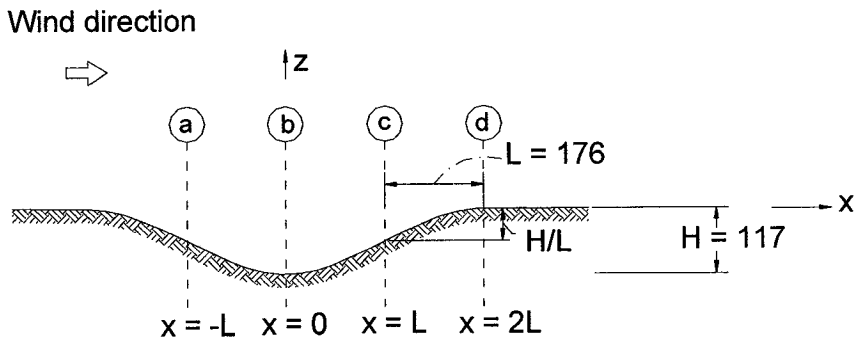


Figure 5.15 Valley - Geometry of deep cosine valley used by the present as well as Busuoli, (1993), Maurizi (2000), and Chung (2004) studies. Scale 1:1000. All measurements are in [mm].

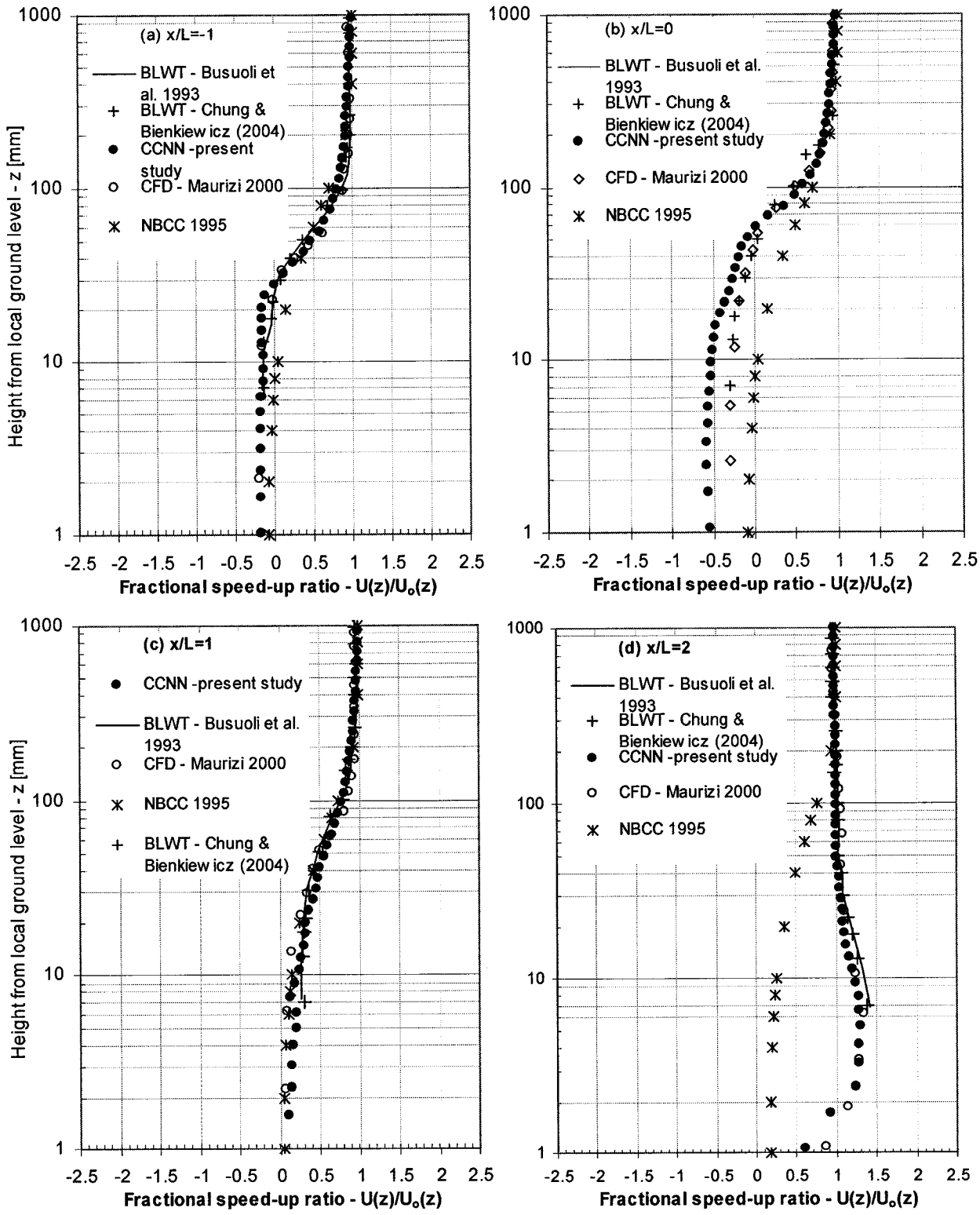


Figure 5.16 Valley - Comparison between CCNN, BLWT, and NBCC FSUR values (for H/L =0.667) at four longitudinal locations (a)x/L = -1, (b)x/L = 0, (c)x/L = 1, and (d)x/L= 2

5.7 Closure

A new combined CFD-NN approach useful for estimating design wind loads on buildings in the vicinity of topographic features such as hills, valleys, and escarpments has been developed and validated. In comparison with the CFD alone approach, the proposed approach is not computationally intensive and requires only provision of simple terrain geometry and roughness parameters to predict the corresponding FSUR values. This makes the approach less expensive than CFD simulations alone while producing results of comparable quality. It is also less expensive than BLWT-NN approach for the end-user.

The validated CFD-NN FSUR predictor can be distributed among practicing engineers and used to estimate the variation in wind loads due to upstream topography. In addition to its ease to use, the proposed combined CFD-NN has been more useful by generating FSUR results for multiple hill geometries not currently addressed in wind codes and standards.

6 Conclusions

6.1 Concluding remarks

Using a computational approach, a comprehensive evaluation of topography effects on design wind flow conditions has been carried out. Research developments in this area have been analyzed and results of various studies based on five different methods have been compared to stress the need to model these topographic effects, to identify influential parameters as well as to highlight some computational details that are necessary to enhance the accuracy of prediction. Such influential parameters include ground roughness and accuracy of geometrical representation of the terrain.

To this effect the roughness of the terrain is incorporated into the proposed and developed numerical model both through the upstream boundary conditions with the use of log-law and throughout the ground boundary using shear in the momentum equation. Numerical simulations with and without roughness consideration have been carried out to quantify the improvements made due to the incorporation of roughness. The results of this analysis clearly indicate the enhancement associated with the inclusion of roughness values.

To study systematically the effect of accurate geometrical representation of the terrain on the quality of the numerical results, numerical simulations with different topographical element representations have been carried out. The observed sensitivity of the numerical results thus justifies the time spent to develop a robust grid generation tool. To further achieve numerical simulations on a reasonable time frame, it is necessary to control the number of grid points and their distribution in the computational domain. In this regard, the proposed grid generator allows easy control. Thus dense grid point distributions have

been allocated where they are more needed, for example close to the ground and to the topographic elements, where dependent variables are expected to show significant variations.

Wind flow over different types of topography including isolated hills, multiple hills, valleys and escarpments has been considered. The results of this analysis indicate good agreement with experimental data. The comparison of FSUR values at the crest of single and multiple hills shows better agreement with the experiment than previous numerical studies. This is mainly due to the emphasis given on the computational details and the incorporation of roughness in the analysis.

The simulation of wind flow over multiple hills has also shown the impact of a neighboring upstream hill on the wind load. In fact the FSUR values at the crest of a downstream hill are lower by 20 % and 30 % for shallow and steep hills respectively for the heights of general engineering interest, i.e. 5 to 100 m. This percentage in terms of FSUR values can be translated to approximately 45% and 70% reductions in wind loads for the shallow and steep hills, respectively. In NBCC (1995) there is no provision for multiple hills. In practice, NBCC provisions for single hills apply to the multiple hill case which means for the downstream hill the FSUR values are over predicted approximately by 25% and 44% for the height range of 5 to 100 m for the shallow and steep hills respectively. A study on the economical impact of accurate wind load predictions has shown that, with the use of more accurate wind loads, 19% of the concrete lateral load-resisting system (around 6500 m^3) and 2.1% of usable floor area had been saved (Horesfield et al. 2002). The practice of using identical FSUR values for both the upstream and downstream hills usually obtained from the single hill investigations can be

a very conservative approach. Hence, in areas characterized by chains of mountains, the wind load has to be evaluated based on studies of multiple hills, instead of a single hill. The present tools can be beneficial to complement the code provisions in the future.

To further facilitate the practical use of numerical simulations, a new combined CFD-NN approach has been designed, implemented and validated. The combined approach has also been used for producing flow parameters useful for estimating design wind loads on buildings in the presence of considerable terrain topographic features. Consistent FSUR values are generated using the developed CFD-NN approach. Their comparison with experimental values shows a good agreement over a range of different terrain geometries. Compared to CFD simulations alone, the application of a combined CFD-NN model does not demand from the end-user significant experience in numerical simulations nor any extensive computing resources. In addition, the new approach requires only simple geometrical parameters and roughness of the ground to produce good quality FSUR results. The proposed combined CFD-NN approach has been applied successfully for multiple hills geometries not currently addressed in wind codes and standards.

The goals set for the present study have been achieved through the development of three novel tools namely:

- CFD for wind flow over topographic elements;
- Nearly-orthogonal grid generator;
- Combined CFD-NN FSUR predictor.

These three tools are designed using the object-oriented approach and implemented using C++ programming language. This gives an additional advantage in terms of ease of reuse

and modification tasks in the future. Considering the trend of using parallel computers, it facilitates parallel programming tasks as well.

It is believed that future applications of the combined CFD-NN tool will constitute a practical design support system. More specifically the CFD-trained NN can be distributed among practicing engineers enabling them to assess wind load modifications due to the presence of upstream topographic elements. This approach will save time and cost, especially pertinent when scale models are to be built for subsequent wind-tunnel testing. In addition, this approach will also save costs incurred because of conservative estimates of wind loads provided by guidelines based on studies with only limited terrain geometries.

Although the present study focuses on evaluation of wind flow over different topographic elements, the developed methodologies can be of significant use in built environments and environmental planning in general. Furthermore, considering the increasing trend of using wind energy alternatives in Canada and around the world, the present numerical simulation tool can also be useful in wind farm planning. The new combined CFD-NN approach is very efficient time-wise (it takes only 1/300 of the computing time required by CFD alone) and it requires simple input from the end user. These characteristics are very attractive in several areas where real-time solutions are necessary (complex numerical simulations do not satisfy this need). Examples of such civil engineering applications are automated structural health monitoring and operation of large drinking water and wastewater treatment plants. In this regard, complex numerical models based on physical principles generate the knowledge to train the NN in the background while the trained NN supports real-time decisions during the operation of the facilities.

6.2 Research Contributions

The contributions of this study towards a better overall understanding and efficient estimation of topography effects on wind flow, by using integrated computational approaches, can be summarized as follows:

i. Provide a comprehensive review of the state-of-the-art in the domain of numerical evaluation of topography effects on wind flow. A thorough analysis and comparison of results obtained through five different methods namely - BLWT experiments, full scale measurements, analytical methods, NBCC (1995) and CFD - have been made. This review provides a benchmark to evaluate the performance of mathematical models, stresses the promise of computational approaches and emphasizes the requirement of specialized computer tools to model these effects.

ii. Expand the knowledge base: The combined CFD-NN approach has been applied and has successfully generated FSUR values for multiple hill geometries not currently addressed in wind codes and standards. For geometries specified by the code such as the isolated steep hill cases, valleys and escarpments better quality FSUR values have been generated.

iii. Increase accuracy by emphasizing computational details and fine-tuning. A robust grid generator tool, which is suitable for arbitrary surfaces characterized by slopes and curves, has been designed. This has enabled more accurate terrain geometry description, which subsequently allows more accurate ground boundary application. An efficient spatial distribution of grid points (more points are allocated in areas of the computational domain where there is

considerable change in the dependent variable) has also been achieved. Effective incorporation of influential parameters such as ground roughness has contributed to accuracy enhancements in the CFD simulation. In addition, quantitative studies of parameters affecting FSUR values have been carried out, such as geometry representation, ground roughness and a neighbouring hill.

iv. Integrate the computational fluid dynamics approach with neural network generalization leading to a novel and efficient combined CFD-NN for the prediction of FSUR values, which represents added advantage for the end user, i.e. to avoid the complexity associated with the use of CFD approach alone.

v. Design, implementation and validation of three tools namely nearly-orthogonal grid generator, CFD and a combined CFD-NN technique using an object oriented-approach that eases reuse, future expansion, as well as parallel programming, if the need arises.

6.3 Recommendations for future work

The following could enhance the present contributions if succeeded by research in the future:

- Expansion of the present work for 3D complex terrain cases.
- Considerations of unsteady flow to enable estimation of the dynamic components of wind loads.
- Integration of the structure (for example a building) in the computational domain for the purpose of determining the wind loads. This will allow evaluation of the variation on the wind flow induced by presence of the structure in addition to the topographical elements.

- Incorporation of more complex turbulence models such as RSM or LES to enhance the accuracy of wind flow properties estimation in the wake region.
- Integration of the present numerical model with mesoscale numerical meteorological models.

7 References

Akcelik V., Jaramaz, B., Ghattas G. (2001), "Nearly orthogonal two-dimensional grid generation with aspect ratio control", Jnl. of Comput. Physics, Vol. 171, 805–821.

ASCE 7-02, (2002), "ASCE Standard: Minimum Design Loads for Buildings and Other Structures", American Society of Civil Engineers.

Baskaran, A. (1990), "Computer simulation of 3D turbulent wind effects on buildings", PhD thesis, Concordia University, Montreal, Quebec, Canada.

Bergeles, G.C. (1985), "Numerical computation of turbulent flow around two-dimensional hills", Jnl. of Wind Eng. Ind. Aero, Vol 21, 307-321.

Beljaars, A.C., Walmsley, J.L., Taylor, P.A. (1987), "A mixed spectral finite-difference model for neutrally stratified boundary-layer flow over roughness changes and topography", Boundary-Layer Meteorology, Vol 38, 273-303.

Bitsuamlak, G.T., Godbole, P.N. (1999), "Application of cascade-correlation learning network for determination of wind pressure distribution in buildings", Wind Engineering into the 21st Century, Balkema, Rotterdam.

Bitsuamlak, G. T., Stathopoulos, T., Bedard, C. (2002), " Neural network predictions of wind flow over complex terrain", 4th Structural Specialty Conference of the Canadian Society for Civil Engineering, Montréal, Québec, Canada, June 5-8, 2002.

Bradshow (1976)), "Topics in applied physics-Turbulence", Springer-Verlag, NewYork, 2nd Ed., Vol. 12.

Boussinesq. J. (1877) "Théorie de l'écoulement tourbillant", Mém. prés. par div. savants à l'Acad. Sci., Vol 23, 46-50, Paris.

Bowen A.J. (1979), "Full scale measurements of the atmospheric turbulence over two escarpments", Proceedings 5th International Conference on Wind Engineering, 1, 161-172, Pergamon.

Bowen, A.J., Lindely, D. A. (1977), "Wind tunnel investigation of the wind speed and turbulence characteristics close to the ground over various escarpment shapes", Boundary-Layer Meteorology, Vol 12, 259-271.

Bradley, E.F. (1980), "An experimental study of the profiles of wind speed, shearing stress and turbulence at the crest of a large hill", Quart. Jnl. Roy. Met. Soc., Vol 106, 121-124.

Busuoli, M., Trombetti, F., Tamperi, F. (1993), "Data sets for studies of flow and dispersion in complex terrain: (II) the "RUSVAL" wind tunnel experiment (flow data)", Technical paper No. 3, FISBAT-TP-93/1, 129.

Carpenter, P., Locke, N. (1999), "Investigation of wind speed over multiple two-dimensional hills", Jnl. of Wind Eng. Ind. Aero, Vol 83, 109-120.

Chen, Y., Kopp, G.A., Surry, D. (2003), "Prediction of pressure coefficients on roofs of low buildings using artificial neural networks", Jnl. of Wind Eng. Ind. Aero, Vol 91, 423-441.

Chen, Y.S., Kim, S.W. (1987), "Computation of turbulent flows using an extended k-ε turbulence closure model", NASA, CR – 179204.

Chung, J., Bienkiewicz, B. (2004), "Numerical simulation of flow past 2D hill and valley", Jnl of Wind and Structures, Vol.7(1),1-12.

Deaves, D.M. (1980), "Computations of wind flow over two-dimensional hills and embankments", Jnl. of Wind Eng. Ind. Aero, Vol 6, 89-111.

Durbin P.A. (1996), "Technical note: on the k–ε stagnation point anomaly", Int. Jnl. Heat Fluid Flow Vol 17, 89–90.

Eberhart, R.C., Dobbins, R.W. (1990), "Neural network P.C. tools: A practical guide", Academic press, USA.

Eca, L. (1996), "2D orthogonal grid generation with boundary point distribution control", Jnl. Comput. Physics, Vol. 125, 440-453.

English, E. C. and Fricke, F. R. (1999), "The interference index and its prediction using a neural network analysis of wind-tunnel data", Jnl. of Wind Eng. Ind. Aero, Vol 83, 567-575.

Fahlman, S.E. and Lebiere C. (1990), "The Cascade-Correlation Learning Architecture", in Advances in Neural Information Processing Systems 2, Touretzky D. S. (ed.), Morgan Kaufmann Publishers, Los Altos CA, 524-532.

Ferreira, A.D., Silva, M.C.G., Viegas, D.X., Lopes, A.G. (1991), "Wind tunnel simulation of the flow around two-dimensional hills", Jnl. of Wind Eng. Ind. Aero, Vol 38, 109-122.

Ferreira, A.D., Lopes, A.M.G., Viegas, D.X., Sousa, A.C.M. (1995), "Experimental and numerical simulation of flow around two-dimensional hills", Jnl. of Wind Eng. Ind. Aero, Vol 54/55, 173-181.

Flood, I., Kartam, N. (1994), "Neural Networks in Civil Engineering. I: Principles and Understanding", Jnl. Comp. in Civ. Engrg., Vol. 8(2),131-148.

Flood, I., Kartam, N. (1994), "Neural Networks in Civil Engineering. II: Systems and Application", Jnl. Comp. in Civ. Engrg., Vol. 8(2), 149-162.

Glanville, M.J., Kwok, K. C. S. (1997), "Measurements of topographic multipliers and flow separation from a steep escarpment. Part II. Model-scale measurements", Jnl. of Wind Eng. Ind. Aero, Vol 69-71, 893-902.

Holmes, J. D., Banks, R.W., Paevere, P. (1997), "Measurements of topographic multipliers and flow separation from a steep escarpment. Part I. Full scale measurements", Jnl. of Wind Eng. Ind. Aero, Vol 69-71, 885-892.

Hohfeld M. and Fahlman, S.E. (1992), "Learning with Limited Numerical Precision Using the Cascade-Correlation Learning Algorithm", IEEE Transactions on Neural Networks, Vol 3, No 4, 602-611.

Horsfield, J.N., Chan, C.M., Denoon R.O. (2002), "Towards sustainable development through innovative engineering", Housing conference, Wanchai, Hong Kong.

Jackson, P.S. (1979), "The influence of local terrain features on the site selection for wind energy generating systems, Faculty of Engineering Science", University of Western Ontario, Report BLWT-1-1979, 83.

Jackson, P.S., Hunt, J.C.R. (1975), "Turbulent wind flow over a low hill", Quart. Jnl. Roy. Met. Soc., Vol 101, 929-955.

Khanduri, A.C., Bedard, C., Stathopoulos, T. (1995), "Neural network modelling of wind-induced interference effects", Proceedings of the Ninth International Conference on Wind Engineering, New Delhi, India, 1341-1352.

Khanduri, A.C., Bedard, C., Stathopoulos, T. (1997), "Modelling wind-induced interference effects using backpropagation neural networks", Jnl. Wind Engr. & Ind. Aerodyn, Vol. 72, 71-79.

Karpik, S.R. (1988), "An improved method for integrating the mixed spectral finite difference (MSFD) model equations", Boundary-Layer Meteorology, Vol 43, 273–286.

Kim, H.G., Patel, V.C., Lee, C.M. (2000), "Numerical evaluation of wind flow over hilly terrain", Jnl. of Wind Eng. Ind. Aero, Vol 87, 45-60.

Kim, H.G., Choung, M.L., Lim, H. C., Kyong, N. H. (1997), "An experimental and numerical study on the flow over two-dimensional hills", Jnl. of Wind Eng. Ind. Aero, Vol 66, 17-33.

Kobayashi, M.H., Pereira, J.C.F., Siqueira, M.B.B. (1994), "Numerical study of the turbulent flow over and in a model forest on a 2D hill", Jnl. of Wind Eng. Ind. Aero, Vol 53, 357-375.

Kwatra, N., Godbole, P.N., Krishna, P. (1999), "Application of artificial neural network for determination of wind induced pressures on gable roofs", Wind Engineering into the 21st Century, Balkema, Rotterdam.

Launder, B.E. and Spalding, D.B. (1974), "The numerical computation of Turbulent flows", Comput. Methods Appl. Mech. Eng., Vol 3, 269-289.

Lemelin, D.R., Surry, D., A.G. Davenport, A.G. (1988), "Simple approximations for wind speed-up over hills", Jnl. of Wind Eng. Ind. Aero, Vol 28, 117–127.

Li, Y. (1998) "Numerical Evaluation of wind-induced dispersion of pollutants around buildings" PhD thesis, Concordia University, Montreal, Quebec, Canada.

Lun, Y.F. Mochida, A., Murakami, S., Yoshino, H., Taichi Shirasawa, T. (2003), "Numerical simulation of flow over topographic features by revised k–ε models", Jnl. Wind Eng. Ind. Aero, Vol 91, 231-245.

Mason, P.J., Sykes, R.I. (1979), "Flow Over an Isolated Hill of Moderate Slope", Quart. Jnl. Roy. Met. Soc., Vol 105, 383–395.

Mason, P.J. King, J.C. (1985), "Measurements and predictions of flow and turbulence over an isolated hill of moderate slope", Quart. Jnl. Roy. Met. Soc., Vol 111, 617–640.

Mathew, A. Kumar, B., Sinha, B.P., Pedreschi, R. F. (1999), "Analysis of Masonry Panel under Biaxial Bending Using ANNs and CBR", Jnl. Comp. in Civ. Engrg. Vol. 13(3), 170-177.

Maurizi, A. (2000), "Numerical simulation of turbulent flows over 2D valleys using three versions of the k-ε closure model", Jnl. Wind Eng. Ind. Aero, Vol 85, 59-73.

Miller, C.A. (1996), "Turbulent boundary layers above complex terrain", PhD thesis, University of Western Ontario, London, Ontario, Canada.

Miller, C.A., Davenport, A.G. (1998), "Guidelines for the calculation of wind speed-ups in complex terrain", Jnl. Wind Eng. Ind. Aero, Vol 74-76, 189-197.

NBCC (1995) National Building Code User's Guide – Structural Commentaries (Part 4), Canadian Commission on Building and Fire Codes, National Research Council of Canada, Ottawa.

Patankar, S.V. (1981), "Numerical heat transfer and fluid flow", Hemisphere Publishing.

Paterson, D.A., Holmes, J.D. (1993), "Computation of wind flow over topography", Jnl. of Wind Eng. Ind. Aero, Vol 46&47, 471-478.

Pearse, J.R., Lindely, D., Stevenson, D.C. (1981), "Wind flow over ridges in simulated atmospheric boundary layers", Boundary-Layer Meteorology, Vol 21, No. 1, 77-92.

Quinn, A.D., Robertson, A.P., Hoxey R.P., Burgess L.R. (1998), "The effect of small embankments on wind speeds", Wind and Structures, Vol 1, No. 4, 303-316.

Rhie, C.M., Chow, W.L. (1983), "Numerical study of the turbulent flows past an airfoil with trailing edge separation", AIAA Journal, 21, 1532-1525.

Rivard, H. (2000), "Surveying the IT Landscape", Canadian Architect, publié par Southam inc., Vol. 45, No. 7, p. 29.

Ryskin, G., Leal L.G. (1983), "Orthogonal mapping", Jnl. of Comput. Physics, Vol. 50, 71–81.

Sandri P, Mehta, K.C. (1995), "Using backpropagation neural network for predicting wind-induced damage to building", Proceedings of the Ninth International Conference on Wind Engineering, New Delhi, India, 1989-1999.

Salmon, J.R., Bowen, A.J., Hoff, A.M., Johnson, R., Mickle, R.E., Taylor, P.A. (1988), "The Askervein Hill Experiment: Mean wind variations at fixed heights above the ground", Boundary-Layer Meteorology, Vol 43, 247-271.

Shigidi, A., Garcia, L.A. (2003), "Parameter Estimation in Groundwater Hydrology Using Artificial Neural Networks", Jnl. Comp. in Civ. Engrg. Vol. 17(4), 281-289.

Shih, T.H., Zhu, J., Lumley, J.L. (1995), "A new Reynolds stress algebraic equation model", Comput. Methods Appl. Mech. Eng. Vol 125, 287–302.

Schalkoff, R. (1997),"Artificial Neural Networks", Book News, Inc., Portland, Or.

Spalding D. B. (1972), "A novel finite-difference formulation for differential expressions involving both first and second derivatives", Int. Jnl. Num. Methods Eng., Vol 4, 551-559.

Stathopoulos, T. (1997), "Computational wind engineering: Past achievements and future challenges" Jnl. of Wind Eng. Ind. Aero, Vol 67-68, 509-532.

Stathopoulos, T. and Zhou, Y.S. (1993), "Numerical simulation of wind-induced pressures on buildings of various geometries" Jnl. of Wind Eng. Ind. Aero, Vol 46-47, 419-430.

Taylor, P.A., Walmsley, J.L., Salmon, J.R. (1983), "A simple model of neutrally stratified boundary-layer flow over real terrain incorporating wave number dependent scaling", Boundary-Layer Meteorology, Vol 26, 169–189.

TECPLOT® (2000) Plotting and data visualization software, Tecplot, Inc., Bellevue, WA, USA.

Teunissen, H.W., Shokr, M.E., Bowen, A.J., Wood, C.J., Green, D.W.R. (1987), "Askervein Hill Project: Wind-tunnel simulations at three length scales", Boundary-Layer Meteorology, Vol 38, 1-29.

Theodoropoulos, T., Bergeles G., Atanassiadis N. (1985), "Orthogonal grid generation in two dimensional space", Proceeding 4th international conference in numerical methods in laminar and turbulent flow, Swansea, U.K.

Tsai, C. H, Hsu, D.S. (2002), "Diagnosis of Reinforced Concrete Structural Damage Base on Displacement Time History using the Back-Propagation Neural Network Technique", Jnl. Comp. in Civ. Engrg. Vol. 16(1), 49-58.

Van Doormal, J.P and Raithby, G.D. (1984), "Enhancements of the SIMPLE method for predicting incompressible fluid flows", Numer. Heat Transfer, Vol 7, 147-163.

Xu, D., Ayotte, A.W., Taylor, P.A. (1994), "Development of a non-linear mixed spectral finite difference model for turbulent boundary-layer flow over topography", Boundary-Layer Meteorology, Vol 70, 341–367.

Yakhot, V., Orzag, S.A. Thamgams, S., Gatski, T.B., Speziale, C.G., (1992), "Development of turbulence models for shear flows by a double expansion technique", Physics of Fluids: A: Fluid Dynamics, 4(7), 1510-1520.

Weller, H.G., Tabor, G., Jasak H., Fureby, C. (1998), "A tensorial approach to computational continuum mechanics using object orientated techniques", Computers in Physics Vol 12(6), 620 – 631.

Weng, W., Taylor, P.A., Walmsley, J.L. (2000), "Guidelines for airflow over complex terrain: model developments", Jnl. of Wind Eng. Ind. Aero, Vol 86, 169-186.

White, F.M. (1991), "Viscous fluid flow", New York, McGraw-Hill.

Zarghamee, M.S. (2001), "Wind characterization problems at remote sites", ASCE structures exposition, Washington DC.

Zhou, Y. (1995), "Numerical evaluation of wind effects on buildings" PhD thesis, Concordia University, Montreal, Quebec, Canada.

Appendix A

A-1 C++ implementation of class CompDomain definition used by the CFD tool

```
// compdomain.h
// definition of class compdomain

#ifndef COMPDOMAIN_H
#define COMPDOMAIN_H
#include "controlvolume.h"
#include "parameters.h"
#include "CFDInterfaceClass.h"

class CompDomain
{
   public:

        CompDomain(int L1, int M1);
        CFDInterfaceClass CFDInterface;
        ControlVolume **CV;
        void ReadInputParas();
        void Start();
        void InitializeDepVar(int DepVarIndex, bool StartFileIndex);
        void SetGrid();
        void SetGrid(int KEinitialization);
        void SetGridTransformParameters();
        void SetPressCorrParas();
        void SetPressureGradients();
        void SetConvectiveTerms();
        void SetFlow();
        void SetDiffusion();
        void SetSources(int DepVarIndex);
        void SetTurbulentVisc(int DepVarIndex);
        void SetGamma(int DepVarIndex);
        void SetCoefficients(int DepVarIndex);
        void ApplyBound(int DepVarIndex);
        void UnderRelax(int DepVarIndex);
        void PressureCont(int DepVarIndex);
        void SolverManager(int DepVarIndex);
        void SolverManager(int iNumberOfIterations, int DepVarIndex);
        void Solve(bool b1Dsolver,
                   int iNumberOfIter,
                   int DepVarIndex,
                   long double **Phi);
        void Solve(int iNumberOfIter, int DepVarIndex, long double **Phi);
        void SolveP(int iNumberOfIter, int DepVarIndex, long double **Phi);
```

```
    void CorrectUandV();
    void CorrectCovectiveCoef();
    void StoreDepVar(int DepVarIndex);
    void Output(int DepVarIndex);
    void Plot();
    void Plot(char * cPFName);
    void CalculateSpeedUp();
    void ConvergCri();
    void CallingFree();
    bool Covergence();
    char * PlotFileName();
    void IntializeFromFile(char * ptrDataFileN);

private:

    long double x[NX]; // x - coordinates of nodes [i][j]
    long double y[NY]; // x - coordinates of nodes [i][j]
    long double xf[NX+1]; // x - coordinates of CV faces (Note: xf[0]=x[0] and
                          xf[l1+1]=x[l1])
    long double yf[NY+1]; // y - coordinates of CV faces (Note: xf[0]=y[0] and
                          yf[l1+1]=y[l1])
    long double xcv[NX]; // x - coordinates of adjacent CV faces in the x - direction
    long double ycv[NY]; // y - coordinates of adjacent CV faces in the y - direction
    long double xdif[NX+1]; // distance between adjacent grid nodes in the x - direction
    long double ydif[NY+1]; // distance between adjacent grid nodes in the x - direction
    long double wav;
    long double xl; // xl - total length of computation domain in the x-direction
    long double dHalfXl;
    long double yl; // yl - total length of computation domain in the y-direction
    long double xpow; // used to produce an even nodal distribution in the y-direction
    long double ypow; // used to produce an even nodal distribution in the y-direction
    long double h; // ratio of scale factor in two orthogonal direction
    long double HillH; // hill height
    long double HillL; // width of hill from center to a point where hill  height = HillH/2
    long double **CurviX; // x - coordinates in the orthogonal curvilinear coordinate system
    long double **CurviXFaceWE; // CV face x - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double **CurviXFaceSN; // CV face x - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double CurviY[NX][NY];
    long double **CurviY; // y - coordinates in the orthogonal curvilinear coordinate system
    long double **CurviYFaceWE;// CV face y - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double **CurviYFaceSN;// CV face y - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double **dDxDeta;// dx/deta at grid points
    long double **dD2xDeta2;// d2x/deta2 at grid points
    long double **dDyDeta;// dy/deta at grid points
    long double **dD2yDeta2;// d2y/deta2 at grid points
    long double **dDxDzai;// dx/dzai at grid points
    long double **dD2xDzai2;// d2x/dzai2 at grid points
```

```
long double **dDyDzai;// dy/dzai at grid points
long double **dD2yDzai2;// d2y/dzai2 at grid points
long double **dJacobian;// Jacobian at @ cell point P to be used in the source terms
long double **dAlpha;  // (dx/deta)2*(dy/deta)2 at the faces of the CV's to be used in
                          the diffssion coefficients
long double **dBeta; // Beta {(dx/dzai)*(dx/deta)+(dy/deta)*(dy/dzai)} at the nodal
                        points  to be used in the diffussion coefficients and the source terms
long double **dGamma;  // Gamma {(dx/dzai)2*(dy/zai)2} at the nodal points to be used
                          in the diffssion coefficients
long double dPe, deE, dPn, dnN; // dPe is distance betwen nodal point P and CV east face
long double **dPhi; // dependent variable of interest
long double **sp; // coefficient of Phi[i][j] in the linearized source term
long double **sc; // constant in the linearized source term
long double **dLamda1;
long double **dLamda2;
long double **dLamda3;
long double **dLamda4;
long double **dLamda1WE;
long double **dLamda2WE;
long double **dLamda1SN;
long double **dLamda2SN;
long double **dEtaRNG;
long double **dSource1;
long double **dSourceP;
long double **gamma;  // diffusion coefficient
long double **gammaWE; // diffusion coefficient at the east-west faces
long double **gammaSN; // diffusion coefficient at south-north faces
long double **dTurbulentVisc;
long double dPstar; //parmater used in the sources of RNG ke model
long double dConst1_RNG_ke;//variable in RNG ke model
long double dGamW; // temporary storages for gammma at the respective CV faces
long double dGamE, dGamN, dGamS, DynVisc, Gravity, Cond, cp, reftra;
long double resm[MDEP_VAR];
long double resmo[MDEP_VAR];
long double ResIteration1[MDEP_VAR];
long double RelativeResidue[MDEP_VAR];
long double ResMin[MDEP_VAR];
long double RefTrn[MDEP_VAR];
long double ratio;
long double cri[MDEP_VAR];
long double CriMax;
long double qt[MNXNY]; // variables particular to the solve member function TDMA
                          solver
long double pt[MNXNY]; // variables particular to the solve member function TDMA
                          solver
long double b; // variables particular to the solve member function TDMA solver
long double denom; // denominator used in the solve member function TDMA solver
long double AbsRes; // variable used to store the difference between LHS and RHS of
                       descritized equations
long double duGradient; // Parameters related to power law u distribution for u/s bc
long double dyGradient, dAlphaPowerLaw, duPower;
```

```
long double dinletU[NY], dinletV[NY], dinletK[NY], dinletE[NY];
long double dX, dY, dL, dS, dYp, uground, vground;
long double dShearAtWall;//[NX];
long double **u; // horizontal velocity component
long double **dDuDeta;
long double **dDuDzai;
long double **v;  // vertical velocity component
long double **dDvDeta;
long double **dDvDzai;
long double **k;
long double kus[NY];
long double **e;
long double eus[NY];
long double **dG1;
long double **dG1WE;
long double **dG1SN;
long double dG1W, dG1E;
long double **dG2;
long double **dG2WE;
long double **dG2SN;
long double dG2S, dG2N;
long double **dBu;  // Bu, Bv, Cu, Cv are paramaters in G1 and G2 (convective terms)
long double **dBv;
long double **dCu;
long double **dCv;
long double **dB;
long double **dC;
long double **dBWE;
long double **dCSN;
long double dMassSource;
long double **p;
long double **dPprime;// newly calculated pressure correction
long double PGX,PGY, PGXHV,PGYHV;
long double **dDpDeta;// dp/deta at nodal points
long double **dDpDzai;// dp/dzai at nodal points
long double **dDpPrimeDeta;// dpPrime/deta at nodal points
long double **dDpPrimeDzai;// dpPrime/dzai at nodal points
long double **dDpDetaSN;// dp/deta at west east vertical faces of the CV
long double **dDpDetaWE;// dp/deta at west east vertical faces of the CV
long double **dDpDzaiWE;// dp/dzai at south north horizontal faces of the CV
long double **dDpDzaiSN;// dp/dzai at south north horizontal faces of the CV
long double dPressureSource; // the source term in the momentum eqns due to
                                the pressure gradient
long double **FlowX;
long double **FlowY;
long double **FlowWE; // mass flow rate in the x-direction at xf[i] locations
long double **FlowSN; // mass flow rate in the y-direction at yf[i] locations
long double **dDiffusion; // temporary storage for part of the diffusion term
                             (gamma*alpha(or gamma)/Jacobian)
long double **dDiffusionWE; // diffusion terms at west-east faces of the CV
                               (dDiffusion*delataEta/ZaiDifference)
```

```
        long double **dDiffusionSN; // diffusion terms at south-north faces of the CV
                                    // (dDiffusion*delataEta/ZaiDifference)
        long double **dDiffusionSourceWE; //diffusion contributions to the source terms
                                    //at CV faces (gamma*beta*delataEta/Jacobian)
        long double **dDiffusionSourceSN;
        long double dDiffSource, dPhiDetaW, dPhiDetaE, dPhiDzaiS, dPhiDzaiN, PsuFlow;
        long double **APL;
        long double **APR;
        long double **AEL;
        long double **AER;
        long double **AWL;
        long double **AWR;
        long double **ANL;
        long double **ANR;
        long double **ASL;
        long double **ASR;
        long double **CONL;
        long double **CONR;
        long double AEVL[NY], AEVR[NY], AWVL[NY],AWVR[NY];
        long double ANVL[NY], ANVR[NY], ASVL[NY], ASVR[NY];
        long double AEVT[NX], AEVB[NX], AWVT[NX], AWVB[NX];
        long double ANVT[NX], ANVB[NX], ASVT[NX], ASVB[NX];
        long double AEEL[NY], AEER[NY], AWEL[NY], AWER[NY];
        long double ANEL[NY], ANER[NY], ASEL[NY], ASER[NY];
        long double AEET[NX], AEEB[NX], AWET[NX], AWEB[NX];
        long double ANET[NX], ANEB[NX], ASET[NX], ASEB[NX];
        long double ANV[NY], ASV[NY]; // to be used in intialization of k and e
        long double SumAnbU, SumAnbV;
        long double Beta, RhoW, RhoS,,UhatW, VhatS, DuW, DvS;
        long double PgradW, PgradS;
        long double FluxAbs; // value of the sum of the abs. rate of mass flow across
                             // all CV faces along xf[(l1+2)/2]
        long double FluxAbsOld; // old value of FluxAbs
        int   l; // number of nodes in x-direction
        int   m; // number of nodes in y-direction
        int   l1; // l1 - last node in the x-dir.
        int   iHalfL1; // iHalfL1 - the midel node in the x-dir.
        int   l2; // l2 - second to last node in the x-dir.
        int   m1; // m1 - last node in the y-dir.
        int   m2; // m2 - second to last node in the y-dir.
        int   OuterIter, iref, jref, iresm, jresm, ires, jres;
        bool  lsolve, lprint, convge, StartFileI, b1Dsolver;
        bool bPowerLaw, bLogLaw;
        bool bStandard, bRNG, bLowRE, bPatersonKE, bLunKE;
        bool bJackson, bCosine;
        char  cGridFileName[SIZE_PFILENAME];
        char  PlotFName[SIZE_PFILENAME];
        char * cptrPFileName;
};
#endif
```

A-2 C++ implementation of class ControlVolume definition used by the CFD tool

```
// controlvolume.h
// definition of class ControlVolume

#ifndef CONTROLVOLUME_H
#define CONTROLVOLUME_H

class ControlVolume
{
  public:

        ControlVolume();
        void DefaultValues();
        void EastFace(long double ae);
        void EastFace(long double flowE, long double diffussionE);
        void WestFace(long double aw);
        void WestFace(long double dflow, long double dDiffusion);
        void NorthFace(long double an);
        void NorthFace(long double dflow, long double dDiffusion);
        void SouthFace(long double as);
        void SouthFace(long double dflow, long double dDiffusion);
        void SourceTerm(long double SC);
        void SourceTerm(int DepVarIndex, long double dConstSourcesV);
        void SourceTerm(int DepVarIndex,
                        long double dConstSourcesV,
                        long double dSourcesP_V);
        void CenterPoint();
        void CenterPointStore(long double ap);
        void CenterPoint(long double dFlowDiff);
        void DownStrBound(int DepVarIndex, long double dBoundValue);
        void UpStrBound(int DepVarIndex, long double dBoundValue);
        void TopBound(int DepVarIndex, long double dBoundValue);
        void GroundBound(int DepVarIndex, long double dBoundValue);
        void NextToGround(int DepVarIndex, long double dBoundValue);
        void UnderRelax(int DepVarI, long double phi);
        void PrintCV();
        long double ae();
        long double aw();
        long double an();
        long double as();
        long double ap();
        long double apu();
        long double apv();
        long double con();
        long double conu();
        long double conv();
```

```
  private:

        long double AP, APU, APV, AE, AW, AN, AS, CON, dSC_dVcv, dSP_dVcv, APEC,
                    CONU, CONV, CONK, CONE, PeGrid, Apec;
        long double g, DiffE, DiffN, FlowE, FlowW, FlowN, FlowS;
        long double Max(long double a,long double b)
                {
                        if(a<b) return b;
                        else return a;
                }
};
#endif
```

A-3 C++ implementation of class ControlVolume definition used by the grid generator tool

```
// controlvolumeGrid.h
// definition of class ControlVolumeGrid

#ifndef CONTROLVOLUMEGRID_H
#define CONTROLVOLUMEGRID_H

class ControlVolumeGrid
{
  public:

        ControlVolumeGrid();
        void EastFace(long double ycvJ,long double xdiflp1,long double gam);
        void WestFace(long double ycvj,long double xdifi,long double gam);
        void NorthFace(long double xcvI,long double ydiflp1,long double gam);
        void SouthFace(long double xcvi,long double ydifj,long double gam);
        void CenterPoint(long double xCVi,long double yCVj,long double SourceP);
        void ConstSource(long double xcvi,long double ycvj,long double SourceC);
        void DownStrBound(int DepVarIndex);
        void DownStrBound(int DepVarIndex, long double dXl, long double dY);
        void DownStrBound(int DepVarIndex,
                          long double dXl,
                          long double dCurvYl1POld,
                          long double dCurvXl2j,
                          long double dCurvXl1jp1,
                          long double dCurvXl1jm1,
                          long double dCurvYl2j,
                          long double dCurvYl1jp1,
                          long double dCurvYl1jm1,
                          long double dXdifl1,
                          long double dYcvj);
        void UpStrBound(int DepVarIndex);
        void UpStrBound(int DepVarIndex, long double dCon);
        void UpStrBound(int DepVarIndex,
                          long double dX0,
                          long double dCurvY0Pold,
                          long double dCurvX1j,
                          long double dCurvX0jp1,
                          long double dCurvX0jm1,
                          long double dCurvY1j,
                          long double dCurvY0jp1,
                          long double dCurvY0jm1,
                          long double dXdif1,
                          long double dYcvj);
        void TopBound(int DepVarIndex, long double dYl); //TopBound(int DepVarIndex);
        void TopBound(int DepVarIndex, long double dX, long double dYl);
```

```
        void TopBound(int DepVarIndex,
                      long double dYm1,
                      long double dCurviXm1Pold,
                      long double dCurvXip1m1,
                      long double dCurvXim1m1,
                      long double dCurvXim2,
                      long double dCurvYip1m1,
                      long double dCurvYim1m1,
                      long double dCurvYim2,
                      long double dXcvi,
                      long double dYdifm1);
        void GroundBound(int DepVarIndex,
                      long double curvX,
                      long double H,
                      long double L,
                      long double dXl);
        void GroundBound(int DepVarIndex, long double curvX, long double curviYground);
        void GroundBound(int DepVarIndex,
        long double curvX,
        long double curvXip1,
        long double curvYip1,
        long double curvXim1,
        long double curvYim1,
        long double curvXjp1,
        long double curvYjp1,
        long double xCVi,
        long double ydifJp1,
        long double H,
        long double L,
        long double dXl);
        void PrintCV();
        void UnderRelax(long double dPhi);
        long double ae();
        long double aw();
        long double an();
        long double as();
        long double ap();
        long double con();

    private:

        long double AP,AE,AW,AN,AS,CON;

};
#endif
```

A-4 C++ implementation of class CompDomainGrid definition used by the grid generator tool

```
// compdomainGrid.h
// definition of class compdomainGrid

#ifndef COMPDOMAINGRID_H
#define COMPDOMAINGRID_H

#include "controlvolumeGrid.h"
#include "GridInterfaceClass.h"
#include "parameters.h"

class CompDomainGrid
{
  public:

        CompDomainGrid();
        ControlVolumeGrid CV[NX][NY];
        GridInterfaceClass GridInteface;

        void ReadGrid();
        void Start();
        void InitializeCurviXY();
        void InitializeDepVar(int DepVarIndex);
        void SetGrid();
        void SetGamSource(int DepVarIndex);
        void SetCoeff();
        void ApplyBound(int DepVarIndex);
        void Solve();
        void StoreDepVar(int DepVarIndex);
        void Output();
        char * PlotFileName(int iii);
        void Plot();
        bool DidDFConverge();
        void UnderRelax();
        void ExtendCD();
        void PopulateGrid();
        long double  RecursiveArcLength(int,long double,long double);
        long double  DerivativeOfRecursiveArcLength(int,long double,long double);

        void initial_bspline(int,
                             int,
                             long double **,
                             long double **,
                             long double **,
                             long double **,
```

```
                    long double **);

    void basis_function(int,int,long double,long double*,long double **);
    int find_span(int,long double,long double*);
    void basis(int,long double,long double*,long double*);
    void gauss(int,long double**,long double**,int,long double*);
    void nodal(int,int,long double**,long double**,long double*,long double*);
    void nodal_new(int,int,long double**,long double*,long double*);
    void compute_bspline(int,
                         int,
                         long double*,
                         long double*,
                         long double**,
                         long double*,
                         long double**,
                         long double**);
    void cluster();
    void FaceCoordinates();
    bool clustering();
    void MirrorReflection();
    void FreeMem();

private:

    int         Nl,Nm;
    long double **CurviX;
    long double **CurviY;
    long double **CurviXclustered;
    long double **CurviYclustered;
    long double **dS;
    long double **dPointArcLength;
    long double dArcLength[NY],dTotalArcLength[NX];
    long double dDNew, dDTemp, dDold, dDeltaSmin;
    long double **CurviXFaceWE; // CV face x - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double **CurviXFaceSN;// CV face x - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double **CurviYFaceWE;// CV face y - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double **CurviYFaceSN;// CV face y - coordinates in the orthogonal curvilinear
                                   coordinate system
    long double **CurviXcornerPts; // CV corner coordinates
    long double **CurviYcornerPts; // CV corner coordinates
    long double **CurviXpopulated;
    long double **CurviYpopulated;
    long double **dDxDeta;   // dx/deta at grid points
    long double **dD2xDeta2;  // d2x/deta2 at grid points
    long double **dDyDeta;   // dy/deta at grid points
    long double **dD2yDeta2;  // d2y/deta2 at grid points
    long double **dDxDzai;          // dx/dzai at grid points
    long double **dD2xDzai2;  // d2x/dzai2 at grid points
```

```
long double **dDyDzai;              // dy/dzai at grid points
long double **dD2yDzai2;  // d2y/dzai2 at grid points
int iBoundBTemp=0;  //parameters related to extendCD
char PlotFNameExtend[20];
char PFNameExtend[20];
long double **CurviXnew;
long double **CurviYnew;
long double x[NX]; // x - coordinates of nodes [i][j]
long double y[NY]; // x - coordinates of nodes [i][j]
long double xf[NX+1]; // x - coordinates of CV faces (Note xf[0]=x[0] and
                          xf[l1+1]=x[l1])
long double yf[NY+1]; // x - coordinates of CV faces (Note xf[0]=y[0] and
                          yf[l1+1]=y[l1])
long double xcv[NX]; // x - coordinates of adjucent CV faces in the x - direction
long double ycv[NY]; // y - coordinates of adjucent CV faces in the y - direction
long double xdif[NX+1]; // distance between adjacent grid nodes in the x - direction
long double ydif[NY+1]; // distance between adjacent grid nodes in the x - direction
long double **dYBilinear;
long double wav;
long double xl; // xl - total length of computation domain in the x-direction
long double yl; // yl - total length of computation domain in the y-direction
long double xpow; // used to produce an even nodal distibution in the y-direction
long double ypow; // used to produce an even nodal distibution in the y-direction
long double h; // ratio of scale factor in two orthogonal direction
long double HillH; // hill height
long double HillL; // width of hill from center to a point where hill height = HillH/2
long double dCurviYground;
long double dF2[NY];
long double **Phi;
long double **sp;
long double **sc;
long double **dDistortFunctionW;
long double **dDistortFunctionS;
long double dAvgHx[NY+1];
long double dAvgHy[NX+1];
long double **dHx;
long double **dHy;
long double **dP;
long double **dPofHx;
long double **dPofHy;
long double **dQ;
long double **dQofHx;
long double **dQofHy;
long double **gamma;
long double DynVisc, rattol, restol, reftra, resm, resmo, ResMin, reftrn, rati, relax;
long double qt[MNXNY]; // variables particular to the solve member function TDMA
                          solver
long double pt[MNXNY]; // variables particular to the solve member function TDMA
                          solver
long double b; // variables particular to the solve member function TDMA solver
long double denom;// denominator used in the solve member function TDMA solver
```

```
        long double AbsRes; // variable used to store the difference between LHS and RHS of
                                descritized equations
        long double dAbsResDF;
        long double dAbsResDFPerecent;
        long double dOldDF;
        int   l; // number of nodes in x-direction
        int   m; // number of nodes in y-direction
        int   l1; // l1 - last node in the x-dir.
        int   l2; // l2 - second to last node in the x-dir.
        int   m1; // m1 - last node in the y-dir.
        int   m2; // m2 - second to last node in the y-dir.
        int   iDoubleL1, iDoubleM1, iTripleL1, iTripleM1;
        int   maxit; // maxit - Maximum number of outer iteration
        int   iter, maxTDMA, iref, jref, iresm, jresm;
        int   ires, jres, iResDF, jResDF;
        int iBoundBTemp;
        int iBsplineDegree;
        bool  lsolve, lprint, convge, bConvergeDF;
        bool  dJackson, dCosine;
        bool  bDownS_BCType,bUpS_BCType,bTopBCType,bGroundBCType;
        bool  Sliding, Non_Sliding;
        char  PlotFName[20],FileName[20],PFName[20];
        char * cptrPFileName;
        char * ptrPFName;
        bool  bClustered;
};
#endif
```

A-5 C++ implementation of class OutputLayer definition used by the CFD-NN tool

```
// OutpuLyer.h
// definition of class OutputLayer

#ifndef OUTPUTLAYER_H
#define OUTPUTLAYER_H
#include "parameter.h"

class OutputLayer
{
  public:
        OutputLayer(int in,int out,int data);
        void randWplus();
        void getoldWplus(long double wp11[MAX_NO][MAX_NI]);
        void DisplayWplus();
        void saveWplus(char fw11[12]);
        void readWplus(char fw11[12]);
        void computeO();
        void StoreO(char fn_O[20]);
        void quickprop1();
        void returnO(long double OO[MAX_NDAT][MAX_NO]);
        void displayO();
        void displayT();
        void computeE();
        long double returnE();
        void displayE();
        void returnavgE();
        void returnWplus(long double WP[MAX_NO][MAX_NI]);

   private:
         int nni, nno, nndat;
         long double INPUT[MAX_NDAT][MAX_NI];
         long double O1[MAX_NDAT][MAX_NO];
         long double newslope, SumSqError[MAX_NDAT], avgSumSqError;
         long double dWW, ddWplus[MAX_NO][MAX_NI],
                    Wplus[MAX_NO][MAX_NI];
        long double slope[MAX_NO][MAX_NI];

};

#endif
```

A-6 C++ implementation of class HiddenNeuron definition used by the CFD-NN tool

```
//      HiddenNeuron.h
//      definition of class HiddenNeuron

#ifndef HIDDENNEURON_H
#define HIDDENNEURON_H
#include "parameter.h"

class HiddenNeuron
{
  public:

      HiddenNeuron(int nin,
                   int nout,
                   int ndata,
                   long double OUT1[MAX_NDAT][MAX_NO]);
      void randWminus();
      void displayWminus();
      void getWminus(long double wm1[MAX_NI]);
      void returnWminus(long double WWMM[MAX_NI]);
      void displayT();
      void computeH();
      void returnH(long double HH[MAX_NDAT]);
      void quickprop2();
      void computeR();
      void computeE();
      long double returnR();

   private:

      int nii,noo,ndatt;
      double Wminus[MAX_NI],dWminus,ddWminus[MAX_NI],newslope;
      long double O2[MAX_NDAT][MAX_NO];
      long double H[MAX_NDAT],avgH;
      long double SqError[MAX_NDAT], SumSqError, E[MAX_NDAT][MAX_NO],
                  avgE[MAX_NO], candcor[MAX_NO],R;
      long double slope2[MAX_NI];
      long double direction;

};
#endif
```