SHEAR STRESS CONDITIONS ON COMPLEX TERRAIN SURFACES: BASIC PHYSICS AND IMPLEMENTATION

A Dissertation

by

YI LI

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2022

Major Subject: Atmospheric Science

ABSTRACT

The behavior and dynamics of the surface stress boundary condition are explored, both in terms of the basic physics of the condition and the associated implementation in finite-difference models.

Numerical experiments are presented to illustrate the impact of the stress condition on flows past a region of complex terrain, with particular emphasis on the dependence of the condition on terrain geometry. Arguments are presented to show that the surface stresses depend on the terrain geometry in two ways: (i) a dependence on slope, as represented by a normal gradient term; and (ii) a dependence on terrain curvature, which appears in the condition as a Dirichlet term. This dependence on terrain geometry is illustrated through a series of experiments in which simulations using the full form of the stress condition are compared to companion simulations using one of two widely used approximations: (a) the normal gradient condition, which accounts for the terrain slope but neglects curvature; and (b) the flat boundary assumption, which neglects both slope and curvature. The results show that for realistic flows, the terrain geometry plays an important role in the behavior of the surface stresses, and that the associated approximate conditions fail to capture important aspects of the flow over complex terrain.

Previous implementations of the stress condition in numerical models (including the experiments described above) have largely relied on a direct discretization of the stresses at the boundary, ultimately resulting in a global sparse matrix inversion. However, such methods are difficult to implement in highly parallelized models, in which domain decomposition strategies are generally employed. To simplify the implementation, a new method is presented in which the drag condition is recast into a form allowing a straightforward local implementation, thus eliminating the need for a global inversion. As an example, the new approach is implemented in the context of the widely used Weather Research and Forecasting (WRF) model. Verifications are presented showing that for sufficiently high resolution, the new method as implemented in WRF produces essentially identical results to the previous matrix approach.

DEDICATION

I dedicate my dissertation work to my parents and my grandparents, who supported me, and will support me in their second half of lives while in my whole life.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Craig Epifanio, and all my committee member, Dr. John Nielsen-Gammon, Dr. Istvan Szunyogh, and Dr. Robert Hetland for their guidance and acdemic advisoring. And I would like to thank Joseph Olson from NOAA for his help with my WRF learning. I appreciate Candace Renaud and Julie Barnum for their help with data searching at the begining of my project.

I will also give my thanks to my family and all my friends, who have been supporting and encouraging me all the way.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by Professor Craig Epifanio[advisor], Julie Barnum and Candace Renaud [intern undergraduate students] who searched for the terrain data and the sounding data used in Chapter 3.

Funding Sources

Graduate study was supported by the NOAA Global Systems Laboratory, the teaching assistantship from the department of atmospheric science of Texas A&M University

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGMENTS iv

CONTRIBUTORS AND FUNDING SOURCES v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES xi

1. INTRODUCTION 1

2. BACKGROUND 3

   2.1 Basic Physics 3
      2.1.1 The stress condition 3
      2.1.2 Terrain following coordinates 5
      2.1.3 Vorticity at the boundary 7
      2.1.4 Approximate forms 8
   2.2 Implementation 10
      2.2.1 Early attempts 10
      2.2.2 A time-lagged implementation 13
      2.2.3 The matrix implementation 14
      2.2.4 The drag parameterization 15

3. SEMI-IDEALIZED MODEL EXPERIMENTS 19

   3.1 Introduction 19
   3.2 Experimental design 21
      3.2.1 Physical framework 21
      3.2.2 Test case and data sources 22
      3.2.3 Numerical description 23
   3.3 Implementation and dependence on terrain geometry 24
      3.3.1 The normal gradient condition: matrix implementation 24
      3.3.2 The flat boundary assumption 25
      3.3.3 Dependence on terrain geometry 25

3.4 Overview of cases 28
3.4.1 Error analysis 29
3.4.2 Spatial patterns 32
3.4.3 Dependence on surface roughness 34
3.4.4 The outlier cases 36
3.5 Resolution dependence and terrain smoothing 38
3.5.1 Dependence on horizontal grid resolution 38
3.5.2 An outlier case 43
3.5.3 Dependence on terrain smoothing 43
3.6 Implications for wind energy forecasting 47
3.7 Summary 49

4. WRF IMPLEMENTATION 53

4.1 Introduction 53
4.2 The stress condition in local form 54
4.3 Numerical implementation 56
4.3.1 3D local full condition 56
4.3.2 The normal gradient and flat boundary approximations 58
4.3.3 Finding the intersection point 58
4.3.4 Implementation in the WRF code 60
4.4 Idealized tests 63
4.4.1 Verification 63
4.4.1.1 2D verifications 63
4.4.1.2 3D verifications 65
4.4.2 Flat boundary condition in the default WRF 67
4.4.3 WRF Resolution tests 69
4.5 Summary 72

5. CONCLUSIONS 77

REFERENCES 79

APPENDIX A. WRF CODING 82

A.1 Registry 82
A.2 get_norm_intersection and get_surf_winds 82
A.3 cal_deform_and_div 93

LIST OF FIGURES

FIGURE Page

2.1 Schematic representation of the normal and tangent vectors, as described in the text. 3

3.1 (a) Topography on the simulation domain (contour interval = 250m), showing Mts. Ngauruhoe (to the north) and Ruapehu (to the south). Pink shading surrounding Mt. Ngauruhoe shows the region used for quantitative error calculations, as described in the text. (b) Three-dimensional rendering of the region surrounding Mt. Ngauruhoe, as viewed from the direction indicated by the red arrow in (a). Solid lines show terrain contours, as in (a). 22

3.2 Rotated Cartesian coordinates at the point of interest. 26

3.3 Geometry of the tangent direction on a curved surface. 26

3.4 Errors in the low-level winds for simulations using the flat boundary [(a) and (c)] and normal gradient [(b) and (d)] approximations. Red dots indicate afternoon or evening cases, while blue dots show morning cases. All errors are root mean square values, as averaged over the subdomain indicated in Fig. 3.1. (See text for details.) (a) and (b) show errors normalized by the mean disturbance wind speed on the error averaging domain, while (c) and (d) show net errors in units of m/s. Dashed lines in (c) and (d) show the linear least-squares regression (two outliers excluded). 29

3.5 As in Figs. 3.4c and d, but including only the largest 5% of errors on the averaging domain. Shown are mean errors in m/s for the (a) flat boundary and (b) normal gradient cases. 31

3.6 Spatial distributions of the low-level wind errors for simulations using the flat boundary approximation, as sampled for four representative cases from Fig. 3.4. Colors show the magnitude of the vector wind difference between simulations using the full stress condition and the flat boundary approximation, as normalized by the maximum value of the error on the subdomain shown in Fig. 3.1. (See text for details.) Green vectors show the wind difference $\mathbf{u} - \mathbf{u_f}$ in vector form, while thick black arrows show the low-level upstream wind direction. Solid black lines show contours of constant terrain height (c.i. = 250m). Shown are results for (a) Aug 19th, 08Z; (b) Sep 08th, 05Z; (c) Nov 29th, 09Z; and (d) Dec, 16th 06Z. The view in all cases is roughly from the north, as indicated by the red arrow in Fig. 3.1. 33

3.7 As in Fig. 3.6, but for simulations using the normal gradient approximation. Shown are results for (a) Aug 19th, 08Z; (b) Sep 08th, 05Z; (c) Nov 29th, 09Z; and (d) Dec 16th, 06Z. 35

3.8 As in Fig. 3.4, but for simulations using a reduced roughness length of 0.0003 m. .. 36

3.9 Low-level wind errors for the two outlier cases shown in Fig. 3.4, as viewed from a vantage point above and slightly to the north. Colors show the magnitude of the low-level vector wind difference between simulations using the full stress condition and the flat boundary approximation, normalized and plotted as described in Fig. 3.6. Thick black arrows show the direction of the low-level upstream winds. ... 37

3.10 Error patterns in different horizontal resolution for the Sep 08th case between full-stress and flat-boundary conditions: (a) 90m; (b) 120m; (c) 180m; (d) 240m. 39

3.11 Error patterns in different horizontal resolution for the Sep 08th case between full-stress and normal-gradient conditions: (a) 90m; (b) 120m; (c) 180m; (d) 240m. 40

3.12 Error dependence on resolution in the Sep 08 case. Circles: errors produced by flat boundary approximation; triangles: errors produced by normal gradient approximation. 41

3.13 Error dependence on the resolution in Mar 03 case 44

3.14 Error patterns of different filters for the Sep 08th case between full-stress and approximated conditions: (a),(d) 2^{nd} order 2 passes; (b),(e) 4^{th} order 7 passes; (c),(f) 4^{th} order 3 passes. The top three panels show error patterns between full-stress and flat-boundary conditions, while the bottom three show error patterns between full-stress and normal-gradient conditions. 46

3.15 Errors in the lower 5 levels winds for simulations using the flat boundary [(a) and (c)] and normal gradient [(b) and (d)] approximations. (a) and (b) show errors in units of m/s of wind speed on the error averaging domain, while (c) and (d) show the 5% averaging errors. 47

4.1 The intersection of the terrain normal with the first interior coordinate surface 58

4.2 Code map showing parts of the WRF model code relevant to the surface stress implementation. Parts of the code requiring changes (beyond simple subroutine calls, etc) are colored reddish. 62

4.3 Numerical simulations for free-slip flow with $Re = 50$ past a 2D ridge at time $Ut/L = 24$. (a), (d) The semi-idealized model with the matrix full-stress condition and the WRF model with the local full-stress condition; (b), (e) The semi-idealized model with the matrix normal-gradient condition and the WRF model with the local normal-gradient condition; and (c), (f) The semi-idealized model with the flat-boundary condition and the WRF model with the flat-boundary condition. All the plots show x-component velocity (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity). 64

4.4 Numerical simulations for free-slip flow with Re = 50 past a 3D mountain at time Ut/L = 16.8. (a), (b) The idealized model with the matrix full-stress condition and the WRF model with the local full-stress condition; (c), (d) The semi-idealized model with the matrix normal-gradient condition and the WRF model with the local normal-gradient condition; (e) The semi-idealized model with the flat-boundary condition; and (f) WRF model with the default condition. All the plots show x-component velocity (c.i. = 2.5 m/s; darker color indicates higher horizontal velocity). 66

4.5 Numerical simulations for flat-boundary flow with (a) flat boundary condition described in Chapter 3, but applied in WRF code; (b) default version WRF code. (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity). 68

4.6 Numerical simulations for free-slip flow with full stress condition (matrix method used in the left column, local method used in the right column) past a 2D ridge at time Ut/L = 24. (a), (b) $\Delta x = 1/40L_0$; (c), (d) $\Delta x = 1/20L_0$; (e), (f) $\Delta x = 1/10L_0$; (g), (h) $\Delta x = 1/5L_0$. (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity). 70

4.7 Same as 4.6, but with normal gradient conditions 71

4.8 Same as 4.6, but with flat boundary conditions 73

4.9 Numerical simulations for free-slip flow with full stress condition (matrix method used in the top panels, local method used in the lower panels) and fixed vertical grid spacing past a 2D ridge at time Ut/L = 24. (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity). 74

4.10 Same as 4.9 but with fixed horizontal grid spacing 75

LIST OF TABLES

TABLE Page

3.1 Errors in low-level winds for simulations using the flat boundary condition with different filters 45

3.2 Errors in low-level winds for simulations using the normal gradient condition with different filters 45

1. INTRODUCTION

As computational power increases, high-resolution atmospheric models are increasingly being used to simulate flows in regions of steep and complex terrain. A common example is atmospheric transport and dispersion modeling (e.g., Stockie 2011), which has been used extensively in the study of emissions from industrial operations, forest fires, and other applications, such as ash release from volcanic eruptions (Turner & Hurst, 2001), as well as seed, pollen and insect dispersal (Yang et al. 1998; Levin et al. 2003; Loos et al. 2003). When the source of these various constituents are near the ground, terrain can play an important role in how the constituents are transported. Terrain features are also important for wind energy modeling. The wind industry has become increasingly important as the installed capacity of wind energy has grown (Haupt et al., 2019). The wind industry in the United States has grown by a factor of 4.6 from 2008 to 2018 (Weissman, Sargent, & Fanshaw, 2018), and many wind turbines are placed in the mountainous regions. So we might expect that higher resolution and a better representation of the terrain may improve wind energy forecasts, as well as help decide the location of wind turbines.

The need to account for complex terrain has led to a number of well-known challenges for atmospheric models, from both numerical and physical aspects. One of the best known examples is the problem of computing horizontal pressure gradients using data on sloped coordinate surfaces, which is considered one of the most important disadvantages of using terrain-following coordinates. Considerable effort has been devoted to reducing the resulting errors (Corby et al. 1972; Mesinger 1982; Mihailović & Janjić 1986; Janjić 1989). A more physical example is the problem of radiative effects associated with terrain shading. Topography can significantly modify radiation on the Earth's surface through the effects of slope orientation and shadowing, which can affect the surface energy balance (Scherer & Parlow 1994; Dubayah & Loechel 1997; Oliphant et al. 2003). A third example is the problem of choosing an appropriate model resolution. Increased resolution does not always reduce model errors, due to the limitations in physical parameterizations and input data, as well as numerical errors introduced by grid interactions with steep terrain (Mass et al.

2002; Zängl 2007).

A problem that has received somewhat less attention is the impact of steep and complex terrain on the drag exerted on the atmosphere by the ground. In atmospheric models, the interaction of the flow with the ground is generally represented by specifying a turbulent flux of momentum (or a turbulent stress) across the lower boundary. In most current models, this turbulent flux is imposed as if the lower boundary were flat, in which case the implementation of the stress is relatively simple. However, as discussed by Epifanio (2007, hereinafter E07), the implementation of a surface drag over steep and complex terrain is considerably more involved. Furthermore, the terrain is expected to have an important physical impact on the behavior of the stresses and their interaction with the flow, a topic that has received relatively little attention.

The present thesis explores the role of complex terrain in the specification and physics of surface drag conditions at the lower boundary. Broadly speaking, the thesis consists of two parts. The first part considers the impact of terrain on the basic physics of the the surface drag and how the drag interacts with the flow. It is shown that the behavior of the drag has an explicit dependence on both the slope and curvature of the terrain, which produces significant differences from the behavior of stress over flat ground. The second part of the thesis considers the implementation of surface drag conditions in numerical models. A general method for imposing the drag was introduced by E07. However, the method of E07 is difficult to implement, particularly in parallelized models making use of domain decomposition. In the present thesis, an alternative method for imposing the surface stress condition is presented, which is much simpler to implement.

The following chapter provides background on both the physics and implementation of the surface drag condition. Chapter 3 discusses numerical experiments designed to explore the basic physics of the surface stresses, with a particular focus on the role of terrain geometry. Chapter 4 presents an alternative to the method of E07 for imposing the surface stress condition in models, with an implementation in the Weather Research and Forecasting model presented as an example. The final chapter provides conclusions and suggests directions for future work.

2. BACKGROUND

The present chapter reviews the basic physics and mathematical specification of the surface drag condition over complex terrain, as well as some existing numerical methods for imposing the drag condition in the context of numerical (particularly finite-difference) models.

2.1 Basic Physics

2.1.1 The stress condition

The drag condition describes the transfer of momentum across the lower boundary, suggesting the shape of the terrain plays an important role. Let the terrain height in 3D Cartesian coordinates be described by $h(x, y)$, and assume h is second-order differentiable. The local unit normal to the terrain pointing towards to the model interior is then defined uniquely at each terrain boundary point by

$$\mathbf{n} = (n_1, n_2, n_3) = \frac{(-\partial h/\partial x,\ -\partial h/\partial y,\ 1)}{\sqrt{(\partial h/\partial x)^2 + (\partial h/\partial y)^2 + 1}}\,. \tag{2.1}$$

We assume that the turbulent stress can be expressed in terms of the mean viscous stress tensor as

$$\tau_{ij} = -\kappa \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\,\delta_{ij}\,\boldsymbol{\nabla} \cdot \mathbf{u} \right) \tag{2.2}$$

where $\mathbf{u} = (u, v, w) = (u_1, u_2, u_3)$ is the fluid velocity, δ_{ij} is the Kronecker delta, and where κ is the kinematic viscosity, which is taken to be the sum of a small background viscosity plus a turbulent eddy viscosity, thus allowing a contribution from subgrid-scale turbulent fluxes to be included. We assume that $\kappa \neq 0$ at the lower boundary.

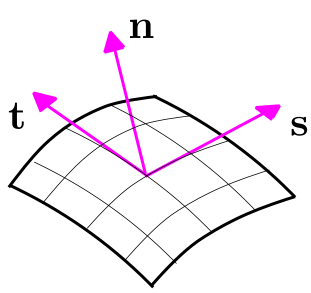


Figure 2.1: Schematic representation of the normal and tangent vectors, as described in the text.

As described by Epifanio (2007, hereinafter E07), applying the drag condition at the boundary amounts to projecting the stress tensor (2.2) across the topographic surface and then taking the tangential component of the resulting stress vector

$\boldsymbol{\tau} \cdot \mathbf{n}$. To be concrete, let the unit tangent vectors to the terrain in the xz and yz-planes be denoted by s and $\mathbf{t}$, respectively, as illustrated schematically in Fig. 2.1. We can then specify the tangential part of $\boldsymbol{\tau} \cdot \mathbf{n}$ by giving its projections in the directions of s and $\mathbf{t}$. If the stress to be imposed on the flow is given by $\mathbf{D}$, then the resulting boundary condition takes the form

$$\tau_{ij} n_j s_i = D_x \qquad \text{and} \qquad \tau_{ij} n_j t_i = D_y \tag{2.3}$$

where $D_x = \mathbf{D} \cdot \mathrm{s}$ and $D_y = \mathbf{D} \cdot \mathbf{t}$ are the projections of the imposed stress into the xz and yz tangent directions, respectively.

Note than in terms of $\mathbf{n}$, the tangent vectors s and $\mathbf{t}$ can be written as

$$\mathrm{s} = \frac{(n_3, 0, -n_1)}{\sqrt{n_1^2 + n_3^2}} \qquad \text{and} \qquad \mathbf{t} = \frac{(0, n_3, -n_2)}{\sqrt{n_2^2 + n_3^2}} \,. \tag{2.4}$$

Substituting into (2.3) then allows the boundary condition to be written in terms of τ_{ij} and the components of $\mathbf{n}$; specifically

$$\tau_{1j} n_j n_3 - \tau_{3j} n_j n_1 = D_x \left(n_1^2 + n_3^2\right)^{1/2} \qquad \text{and} \tag{2.5a}$$

$$\tau_{2j} n_j n_3 - \tau_{3j} n_j n_2 = D_y \left(n_2^2 + n_3^2\right)^{1/2} \,. \tag{2.5b}$$

The goal is then to specify the flow so that both (2.5a) and (2.5b) are satisfied at all points on the terrain surface.

Note that as written, (2.5a) and (2.5b) [or equivalently, (2.3)] seem to be boundary conditions on the components of the stress. However, the stress tensor has six unique components, so that (2.5a) and (2.5b) are insufficient to determine all six components. Instead, to have a solvable problem, (2.5) should be seen as a boundary condition on the velocity components. To be specific, substituting for τ_{ij} in terms of (2.2) and combining with the condition of no flow through the boundary gives the system

$$\left(\frac{\partial u_1}{\partial x_j}+\frac{\partial u_j}{\partial x_1}-\frac{2}{3}\,\delta_{1j}\,\boldsymbol{\nabla}\cdot\mathbf{u}\right)n_jn_3 - \left(\frac{\partial u_3}{\partial x_j}+\frac{\partial u_j}{\partial x_3}-\frac{2}{3}\,\delta_{3j}\,\boldsymbol{\nabla}\cdot\mathbf{u}\right)n_jn_1 = -\frac{D_x}{\kappa}\left(n_1^2+n_3^2\right)^{1/2} \tag{2.6a}$$

$$\left(\frac{\partial u_2}{\partial x_j}+\frac{\partial u_j}{\partial x_2}-\frac{2}{3}\,\delta_{2j}\,\boldsymbol{\nabla}\cdot\mathbf{u}\right)n_jn_3 - \left(\frac{\partial u_3}{\partial x_j}+\frac{\partial u_j}{\partial x_3}-\frac{2}{3}\,\delta_{3j}\,\boldsymbol{\nabla}\cdot\mathbf{u}\right)n_jn_2 = -\frac{D_y}{\kappa}\left(n_2^2+n_3^2\right)^{1/2} \tag{2.6b}$$

$$\mathbf{u}\cdot\mathbf{n}=0 \tag{2.6c}$$

which is three equations for the three unknown components of $\mathbf{u}$ at the boundary. The boundary condition is applied by specifying u, v, and w at the boundary so that the three components of (2.6) are satisfied together.

Finally, we note that in many idealized models, a free-slip condition is applied on the terrain surface to simplify the dynamics. From the above, the free-slip condition is just a special case of (2.6a) and (2.6b), with $D_x = D_y = 0$.

2.1.2 Terrain following coordinates

A common approach to deal with the irregular geometry of the terrain is to introduce terrain following coordinates. We consider the specific class of computational-coordinate transformations defined by

$$X = x, \qquad Y = y, \qquad q = q(x,y,z) \tag{2.7}$$

where (x,y,z) are the physical-space Cartesian coordinates and where the mapping is assumed to be one-to-one. The coordinate is said to be terrain following if in the transformed coordinate, the surface of the terrain is a surface of constant q.

The use of terrain following coordinates for terrain modeling was first proposed by Gal-Chen

& Somerville (1975), who introduced the specific coordinate

$$q = z_T \, \frac{z(x,y) - h(x,y)}{z_T - h(x,y)} \tag{2.8}$$

where z_T is the top of the model domain. In the coordinate (2.8), the terrain is defined by $q = 0$, while the upper boundary of the model domain is defined by $q = z_{top}$. Applying (2.7) with (2.8) then maps the irregular physical domain to the rectangular computational coordinates

$$x_{min} \leq x \leq x_{max}, \qquad y_{min} \leq y \leq y_{max}, \qquad 0 \leq q \leq z_T \; .$$

Since the lower boundary is a level surface in the transformed coordinate, the Cartesian gradient of q at the boundary must be parallel to $\mathbf{n}$; i.e.,

$$\boldsymbol{\nabla} q \,|_{z=h} = \gamma \mathbf{n} \tag{2.9}$$

where γ is a known function of horizontal position. For example, for the specific terrain-following coordinate defined by (Gal-Chen & Somerville, 1975)

$$\gamma = \frac{z_T}{z_T - h} \sqrt{\left(\frac{\partial h}{\partial x}\right)^2 + \left(\frac{\partial h}{\partial y}\right)^2 + 1} \; . \tag{2.10}$$

The horizontal and vertical derivatives (of u as an example) in the terrain following coordinate can then be rewritten as

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial X} + \frac{\partial u}{\partial q}\frac{\partial q}{\partial x} = \frac{\partial u}{\partial X} + \gamma n_1 \, \frac{\partial u}{\partial q} \tag{2.11a}$$

$$\frac{\partial u}{\partial z} = \frac{\partial u}{\partial q}\frac{\partial q}{\partial z} = \gamma n_3 \, \frac{\partial u}{\partial q} \tag{2.11b}$$

As shown by E07, using (2.9) and (2.11) in (2.6) and simplifying then puts the drag boundary condition eventually in the form

$$\begin{aligned}2n_1n_3\frac{\partial u}{\partial X} + n_2n_3\left(\frac{\partial u}{\partial Y}+\frac{\partial v}{\partial X}\right) + (n_3n_3 - n_1n_1)\frac{\partial w}{\partial X} \\ - n_1n_2\frac{\partial w}{\partial Y} + \gamma n_3\frac{\partial u}{\partial q} - \gamma n_1\frac{\partial w}{\partial q} &= -\frac{D_x}{\kappa}\left(n_1^2+n_3^2\right)^{1/2} \end{aligned} \tag{2.12a}$$

$$\begin{aligned}2n_2n_3\frac{\partial v}{\partial Y} + n_1n_3\left(\frac{\partial u}{\partial Y}+\frac{\partial v}{\partial X}\right) + (n_3n_3 - n_2n_2)\frac{\partial w}{\partial Y} \\ - n_1n_2\frac{\partial w}{\partial X} + \gamma n_3\frac{\partial v}{\partial q} - \gamma n_2\frac{\partial w}{\partial q} &= -\frac{D_y}{\kappa}\left(n_2^2+n_3^2\right)^{1/2} \end{aligned} \tag{2.12b}$$

$$w = u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y} \tag{2.12c}$$

2.1.3 Vorticity at the boundary

As shown by E07, for the case of 2D flow, the drag boundary condition can be rewritten more simply in terms of the vorticity at the boundary. Specifically, starting with the 2D form of (2.12a)

$$2n_1n_3\frac{\partial u}{\partial X} + \gamma n_3\frac{\partial u}{\partial q} + (n_3n_3 - n_1n_1)\frac{\partial w}{\partial X} - \gamma n_1\frac{\partial w}{\partial q} = -\frac{D_x}{\kappa} \tag{2.13}$$

E07 show that by combining with (2.1), (2.9), and (2.12c), the 2D drag condition can be rewritten as

$$\eta + 2\,u\left[1+\left(\frac{\partial h}{\partial x}\right)^2\right]^{-1}\frac{\partial^2 h}{\partial x^2} = -\frac{D_x}{\kappa}$$

where η is the vorticity, and its form in the terrain following coordinate can be written as

$$\eta = \frac{\partial u}{\partial q}\frac{\partial q}{\partial z} - \frac{\partial w}{\partial X} - \frac{\partial w}{\partial q}\frac{\partial q}{\partial x} = \gamma n_3\frac{\partial u}{\partial q} - \frac{\partial w}{\partial X} - \gamma n_1\frac{\partial w}{\partial q} \tag{2.14}$$

Recalling that for 2D flow $n_1^2 + n_3^2 = 1$, combining the first of (2.4) with (2.6c) shows that the tangential component of $\mathbf{u}$ at the boundary has the form

$$\mathbf{u}\cdot\mathbf{s} = u\left[1+\left(\frac{\partial h}{\partial x}\right)^2\right]^{1/2} \tag{2.15}$$

in which case the 2D stress condition (2.13) can be expressed as

$$\eta + 2\,\frac{\mathbf{u}\cdot\mathbf{s}}{R} = -\frac{D_x}{\kappa} \tag{2.16}$$

where

$$R = \left[1 + (\partial h/\partial x)^2\right]^{3/2} \Big/ \left(\partial^2 h/\partial x^2\right) \tag{2.17}$$

is the radius of curvature of the terrain profile. Note that as defined, R is positive for concave terrain (valleys) and negative for convex terrain (ridges).

It is worth noting that with a drag condition, the vorticity at the ground depends not just on the drag, but also on the curvature of the terrain. In particular, for flow in the positive x direction, flow past ridges ($R < 0$) produces positive vorticity due to the curvature term, while flow past valleys ($R > 0$) produces negative vorticity. This is true even for free-slip flow ($D_x = 0$), implying that even for a free-slip condition, vorticity can be produced at the boundary.

The condition (2.16) also provides a method for implementing the stress condition in 2D vorticity-streamfunction models, such as that described by E07 and Viner et al. (2013). In a vorticity-streamfunction model, the updated interior vorticity is inverted at each time step to give the updated streamfuction, and by extension, the updated velocity fields. Given the updated u, imposing the stress condition then amounts to specifying the updated vorticity at the lower boundary points according to (2.16).

2.1.4 Approximate forms

Implementing the drag condition as described above can be somewhat involved numerically, and as described in chapter 4, this is particularly true for highly parallelized models using domain decomposition strategies. As a result, existing atmospheric models almost exclusively apply the drag condition in an approximate form, thus simplifying the implementation. Broadly speaking, there are two approximations in widespread use:

- *The flat boundary approximation.* In the flat boundary approximation, the terrain is assumed to be locally horizontal for purposes of specifying the stress, in which case the drag condition (2.3) reduces to

$$\tau_{13} = D_x \qquad \text{and} \qquad \tau_{23} = D_y \tag{2.18}$$

  or equivalently

$$\frac{\partial u}{\partial z} = -\frac{D_x}{\kappa} \qquad \text{and} \qquad \frac{\partial v}{\partial z} = -\frac{D_y}{\kappa}\ . \tag{2.19}$$

  If the model grid is such that the vertical grid lines are parallel to the z-axis, then (2.19) and (2.18) are trivial to implement numerically. For this reason, most current generation atmospheric models employ some version of (2.19) or (2.18) at the boundary.

- *The normal gradient approximation.* In the normal gradient approximation, the stresses are applied as if the boundary was locally given by an inclined plane—specifically, by the tangent plane to the boundary at the point of interest. The analogous condition to (2.19) for an inclined plane is then

$$\kappa \frac{\partial \mathbf{V}}{\partial n} = -\mathbf{D} \tag{2.20}$$

  where $\partial / \partial n$ is a derivative in the direction normal to the boundary, and where

$$\mathbf{V} = \mathbf{u} - (\mathbf{u} \cdot \mathbf{n})\, \mathbf{n} \tag{2.21}$$

  is the tangential wind. Projecting (2.20) into the $\mathbf{s}$ and $\mathbf{t}$ directions then reduces the drag condition to

$$\frac{\partial u_s}{\partial n} = -\frac{D_x}{\kappa} \qquad \text{and} \qquad \frac{\partial u_t}{\partial n} = -\frac{D_y}{\kappa} \tag{2.22}$$

  where $u_s = \mathbf{u} \cdot \mathbf{s}$ and $u_t = \mathbf{u} \cdot \mathbf{t}$ are the projections of the tangential wind into the xz and yz planes, respectively. As suggested by the assumption of a tangent plane, the normal gradient condition captures the effect of terrain slope, but fails to account for the curvature of the terrain.

Further discussion of the geometric dependence of these two approximations is given in section 3.3.3.

2.2 Implementation

Below, we describe some of the attempts to implement the drag boundary condition described above and its approximations as used in previous atmospheric models. For concreteness, we focus on the implementation in finite-difference models, since most mesoscale atmospheric models have been developed in the finite-difference framework.

2.2.1 Early attempts

In many cases, a drag condition was implemented during the initial development of a model (or a family models) and then has not changed much through the subsequent versions. As such, it is useful to review the treatment of the boundary condition in some of the early modeling studies, particularly since the formulation of the condition has led to a certain amount of confusion.

Gal-Chen & Somerville (1975, hereinafter GS75): One of the first studies to consider modeling over complex terrain was GS75, which introduced a formalism for handling complex terrain of arbitrary terrain slope and curvature, referred to in subsequent studies as terrain-following coordinates (cf. section 2.1.2). To be specific, GS75 considered methods for transforming a domain of complex shape with terrain into a simpler rectangular domain, in which the equations could be discretized using standard methods. The specific coordinate transformation suggested by GS75 has formed the basis of many generations of terrain models.

GS75 were mainly concerned with defining the properties of the coordinate transformation and re-deriving the equations of motion in the new coordinate system. However, the treatment of the boundary conditions was also considered, with attention given to two alternatives: the no-slip condition, and the free-slip condition, which in principle can be considered a specific case of (2.3). Unfortunately, the free slip condition was in fact specified incorrectly (and without justification) as

$$\boldsymbol{\zeta} \times \mathbf{n} = 0 \tag{2.23}$$

where $\boldsymbol{\zeta}$ is the 3D vorticity vector.

To see that (2.23) is incorrect, we can refer to section 2.1.3. According to (2.16), free-slip flow past a long ridge will inevitably have a y-component of vorticity, due to the curvature of the terrain. Since the y-direction is perpendicular to $\mathbf{n}$, this in turn requires $\boldsymbol{\zeta} \times \mathbf{n} \neq 0$. In general, it can be shown that the free-slip version of (2.3) reduces to (2.23) only if the curvature of the terrain is neglected.

Clark (1977, hereinafter C77): Probably the first model to make use of the coordinate transformation of GS75 was the early anelastic terrain model of C77. As part of their model formulation, C77 specify a simple drag parameterization at the boundary. However, unlike the discussion in section 2.1.1, C77 treat the drag condition as boundary condition directly on the stress, rather than the velocity, which eventually leads to a problem that is not well posed. To be specific, C77 specify the τ_{13} and τ_{23} components of the stress using the flat boundary assumption (2.18). However, the remaining parts of the stress are simply set to zero at the boundary, namely

$$\tau_{11} = \tau_{22} = \tau_{33} = \tau_{12} = 0 \tag{2.24}$$

which is formally inconsistent with (2.18). To see this, note that even for a flat boundary, (2.18) puts no constraint on the horizontal derivatives of u and v at the boundary, or on $\partial w/\partial z$, meaning (2.24) will not generally apply. And clearly if the boundary is not flat, (2.24) constrains the stress in an unrealistic way.

In addition to the conditions on the stress, C77 also assume a condition on the velocity as part of their anelastic pressure solver, namely

$$\frac{\partial u}{\partial z} = \frac{\partial v}{\partial z} = 0 \tag{2.25}$$

which is a free-slip version of the flat boundary condition (2.19). However, under any more general conditions, (2.25) is again inconsistent with (2.18). It is worth noting that the condition (2.25) is used only in the pressure solver and is never applied directly to the velocity. But the inconsistency of the condition nonetheless remains.

Durran & Klemp (1983, hereinafter DK83): Another early terrain model based on the GS75 coordinate but using a somewhat different numerical framework is the time-split, compressible model of DK83. In documenting their model, DK83 did not explicitly address the drag condition at the boundary, except to mention that mixing normal to the lower boundary vanished, presumably as an implementation of the free-slip condition. However, inspection of the code shows the stresses are again implemented using the flat boundary form (2.18).

Like C77, DK83 treat the surface drag condition as a boundary condition on the stress, rather than on the velocity, so that τ_{13} and τ_{23} are specified directly. However, unlike C77, no assumptions are made about the remaining components of the stress tensor. Instead, through the combination of grid staggering and one-sided differencing, DK83 avoid any reference to the remaining components at the ground, leaving these terms unspecified (and thus avoiding the inconsistency in the C77 approach).

Tests using our own models show that for high enough resolution, the DK83 approach produces results similar to (2.19), suggesting the approach is a reliable implementation of the flat boundary condition. However, the method makes no attempt to account for terrain slope or curvature.

Subsequent models: As best we know, all the later atmospheric community models have either followed or simply inherited the approaches for the drag condition first implemented by C77 and DK83. For example, as discussed in chapter 4, the public version of the WRF model specifies the surface stresses essentially as in C77, and until the study of E07, our own model specified stresses as in DK83. The number of models implementing the full version of the stress condition, valid for arbitrary terrain slopes and curvatures, would appear to be limited to the few cases described below.

2.2.2 A time-lagged implementation

The first attempt to apply the full version of the stress condition in an atmospheric model without restrictions on terrain geometry was presented in an unpublished manuscript by V. Grubišić and P. K. Smolarkiewicz (1999, unpublished manuscript; hereinafter GS99). The method of GS99 is based on ideas borrowed from iterative elliptic solvers, in which a complex boundary condition is sometimes split over two iterations so as to simplify the application (e.g., Smolarkiewicz & Margolin, 1994). As long as the solver converges, the splitting of the boundary terms is expected to have negligible influence on the final solution.

In the context of the drag condition, the GS99 method involves splitting the boundary condition over two time steps. To be concrete, suppose the method is applied as part of a forward in time step advancing from t to $t + \Delta t$. In that case, the fields at the interior grid points are first advanced using the model dynamics, and the drag condition is then used to determine the fields at the ground at $t + \Delta t$. In the approach of GS99, the boundary condition is discretized so that only the vertical (or more precisely, $\partial/\partial q$) terms in (2.12) are evaluated at $t + \Delta t$, while the remaining terms are lagged to time t. That is, the boundary conditions (2.12) are evaluated as

$$\gamma n_3 \frac{\partial u}{\partial q}^{t+\Delta t} - \gamma n_1 \frac{\partial w}{\partial q}^{t+\Delta t} = \text{known terms} \tag{2.26}$$

$$\gamma n_3 \frac{\partial v}{\partial q}^{t+\Delta t} - \gamma n_2 \frac{\partial w}{\partial q}^{t+\Delta t} = \text{known terms} \tag{2.27}$$

$$w^{t+\Delta t} - u^{t+\Delta t}\frac{\partial h}{\partial x} - v^{t+\Delta t}\frac{\partial h}{\partial y} = 0 \tag{2.28}$$

where the right side in (2.26) and (2.27) includes the specified drag along with terms evaluated at time t. The vertical derivatives in (2.26) and (2.27) are then discretized using one-sided differences and known terms at interior points are moved to the right side, leaving a simple algebraic system for the boundary velocity at time $t + \Delta t$.

The GS99 method has an advantage of being straightforward to implement. However, a significant limitation of the method is that in order for (2.26)–(2.28) to provide an accurate boundary condition, the terms on the right side must be accurately represented at time t, implying the fields at time t must already satisfy the boundary conditions. Going back, this means the fields at the initial model time must satisfy the boundary conditions *a priori*—if not, the chain of boundary conditions will be broken for all times going forward. While this limitation may be acceptable for some idealized applications (e.g., an acceleration from rest), it generally cannot be satisfied for realistic applications.

Despite its limitations, the GS99 method has been an important part of the EULAG model and its more recent offshoots (e.g., Smolarkiewicz et al., 2007). It is worth noting that the method is easiest to implement on unstaggered grids, since staggering requires the terms in (2.26)–(2.27) to be averaged in space, thus coupling terms at different grid points and complicating the algebraic solution.

2.2.3 The matrix implementation

A general method for implementing the drag boundary condition in finite difference models, fully accounting for terrain geometry, was introduced by E07. In the E07 method, the fields are first advanced to time $t + \Delta t$ at interior grid points, and the terrain following conditions (2.12a) and (2.12b) are then discretized directly on the model grid at time $t + \Delta t$. Horizontal derivatives are differenced using second order centered differences at the boundary, for example

$$\frac{\partial u}{\partial X} = \frac{u_{i+1,j,0} - u_{i-1,j,0}}{2\triangle x} \tag{2.29}$$

where i and j are the horizontal indices, while the vertical index 0 refers to points on the boundary. Vertical derivatives are differenced using one sided differences, for example

$$\frac{\partial u}{\partial q} = \frac{u_{i,j,1} - u_{i,j,0}}{\triangle q} \tag{2.30}$$

where the fields at the first interior level in the vertical are known.

Discretizing (2.12a) and (2.12b) as in (2.29) and (2.30) and grouping the known interior values of velocity on the right side gives a pair of linear equations for the velocity components at the boundary. Combining with (2.12c) then gives a solvable system of three equations, which can be written symbolically as

$$\mathcal{L}\,\mathbf{u_0} = \mathbf{b} \tag{2.31}$$

where $\mathcal{L}$ is a linear sparse matrix operator derived from the discretized forms of (2.12a)–(2.12c), and where $\mathbf{u_0}$ is the vector consisting of the velocity components at the ground. The right side in (2.31) includes terms involving the known interior values, along with the specified stresses D_x and D_y. Solving (2.31) for $\mathbf{u_0}$ then gives the unique values of the velocity at the boundary satisfying (2.12a)–(2.12c), and using these surface fields when computing the deformations and associated viscous / turbulent flux terms at the boundary then guarantees the required stress condition is satisfied.

As implemented by E07, (2.12a)–(2.12c) are discretized on a staggered C-grid, and the system is solved using the iterative conjugate-residual method of Smolarkiewicz & Margolin (1994). However, it is worth noting that while the resulting method is general, it does depend on a global sparse matrix inversion, which can be difficult to implement. This is particularly true for parallelized models using domain decomposition. A new local implementation that avoids this matrix inversion issue will be outlined in chapter 4.

2.2.4 The drag parameterization

In general, the momentum flux at a solid boundary is highly dependent on the characteristics of the boundary. For atmospheric flows, we usually have a hydrodynamically rough surface, in which the stress is transmitted to the lower boundary by way of the "pressure drag" on the roughness elements at the boundary. Above the boundary, the velocity profile is dominated by shear-induced turbulent eddies, at least over a thin atmospheric layer referred to as the *surface layer*. Typical depths for this turbulent surface layer are several 10s of meters, with deeper layers during the day and shallower at night.

For simplicity, the lower boundary is assumed to be flat, although similar arguments should be expected to hold for inclined terrain. The flux of momentum in the turbulent surface layer is usually represented in terms of the flux gradient hypothesis, in which the flux is assumed proportional to the shear in the mean field; specifically

$$\overline{u'w'} = -\kappa \frac{\partial U}{\partial z} \tag{2.32}$$

where U is the mean wind, primes represent turbulent fluctuations, and the overline is an average over space or time. For notation, the stress is often expressed in terms of a *friction velocity*, defined so that

$$u_* = \left|(\overline{u'w'})_s\right|^{1/2} .$$

where the subscript s indicates the surface value. From observations, the turbulent flux is typically assumed to be roughly constant with height in the surface layer. Combining with (2.32) then gives

$$\kappa \frac{\partial U}{\partial z} = u_*^2 \tag{2.33}$$

where u_* is constant and where the x axis is chosen so that $\partial U / \partial z > 0$.

The coefficient κ is given by the mixing length hypothesis, in which displaced parcels are assumed to carry the properties of the background state over a distance l, called the *mixing lenth*. In the present case, this gives (see, e.g., Holton, 1992, sec. 5.3)

$$\overline{u'w'} \simeq -l^2 \left|\frac{\partial U}{\partial z}\right| \frac{\partial U}{\partial z} \tag{2.34}$$

where the signs give fluxes opposite the gradient in U. The condition (2.32) then suggests

$$\kappa = l^2 \left|\frac{\partial U}{\partial z}\right|$$

and combining with (2.33) then gives

$$\left(l \, \frac{\partial U}{\partial z} \right)^2 = u_*^2 \tag{2.35}$$

where we again assume $\partial U / \partial z > 0$.

For a neutral surface layer, the mixing length is assumed proportional to the distance from the boundary—specifically, $l \simeq kz$, where $k \simeq 0.41$ is the von Karman constant. Substituting into (2.35) results in

$$\frac{\partial U}{\partial z} = \frac{u_*}{kz} \tag{2.36}$$

and ntegrating with respect to z gives

$$\frac{U}{u_*} = \frac{1}{k} \ln \frac{z}{z_0} \tag{2.37}$$

where the constant z_0 is the roughness length, which is the height at which U is assumed to be zero. In terms of the turbulent stress, (2.37) implies

$$\tau_0 = \rho \, u_*^2 = \rho \, U^2 \left[\frac{1}{k} \ln \frac{z}{z_0} \right]^{-2}$$

where ρ is the density.

In vector form, (2.38) is often rewritten as

$$\mathbf{D} = -C_d \, |\mathbf{V}| \, \mathbf{V} \tag{2.38}$$

where

$$C_d = \rho \, \left[\frac{1}{k} \ln \frac{z}{z_0} \right]^{-2} \tag{2.39}$$

is the drag coefficient (as determined by z_0), and where $\mathbf{V}$ is the tangential wind, as in (2.21). The negative sign is given by the fact that the drag is exerted to the opposite direction of the wind. As applied in a numerical model, the tangential wind and drag coefficient are typically evaluated at the first grid level above the ground, which is assumed to be in the surface layer.

The conditions (2.38) and (2.39) are assumed to apply for neutral surface layers. However, complications arise because the atmosphere is not often neutral within the surface layer. In cases for which the surface layer is unstable or stable, observations suggest that the wind speeds are greater than those found under neutral conditions for unstable conditions, and decreased for stable conditions. Empirical corrections are typically applied to account for the departures from the neutral case (Dyer 1974; Businger et al. 1971), with (2.37) replaced by the form

$$\frac{U}{u_*} = \frac{1}{k}\left[\ln\frac{z}{z_0} - \psi_m\left(\frac{z}{L}\right)\right] \tag{2.40}$$

where L is the Monin-Obukhov length (Obukhov, 1971) and where the function ψ_m takes different forms depending on the stratification. Given the wind profile as described above, the turbulent flux can be recovered from the friction velocity as

$$\tau_0 = \rho\, u_*^2 = \rho\, U^2 k^2 \left[\ln\frac{z}{z_0} - \psi_m\left(\frac{z}{L}\right)\right]^{-2}$$

It is worth noting that the basic form of (2.40) follows from Monin-Obukhov similarity theory (Monin & Obukhov, 1954), which was developed for turbulence over a horizontal boundary. While significant effort has been devoted to finding more accurate flux-profile relationships in (2.40) under different stability conditions (Businger, Wyngaard, Izumi, & Bradley, 1971), much less attention has been given to the effect of terrain geometry on the fluxes. Specifically, for a sloped boundary, the main shear axis of the flow is no longer aligned with gravity, which we would expect to have an impact on the turbulent eddies and associated fluxes. To our knowledge, the impact of the terrain slope on the Monin-Obukhov similarity theory remains an open question.

3. SEMI-IDEALIZED MODEL EXPERIMENTS

3.1 Introduction

As discussed in section 2.2.3, a method for implementing the general surface stress condition over terrain of arbitrary geometry was introduced by E07. However, the study of E07 was primarily about developing the method and gave limited attention to the behavior and physics of the surface stresses. In particular, the test cases considered by E07 were all highly idealized problems, consisting of neutral stability flows with constant wind past an isolated ridge or hill. Furthermore, as mentioned in section 2.1.4, current atmospheric community models all apply the drag condition as if the lower boundary were flat, without taking into account the geometry of the terrain at all. As a result, we currently have limited insight into the effect of the full, unapproximated drag conduction in more realistic flow conditions.

The goal of the present chapter is to use the method and model of E07 to begin to explore the physics of the drag condition in more realistic flows. In particular, some questions we hope to address are

- How much of an impact does the use of the full stress condition have relative to simulations using flat boundary and normal gradient approximations discussed in section 2.1.4?
- How does the physics and behavior of the flow calculated using the boundary condition depend on particular aspects of the flow, such as the terrain geometry or the upstream wind and stability profiles?

As will be shown below, the behavior of the flow calculated using the drag condition over complex terrain is found to have a strong dependence on the geometry of the terrain in particular, with an explicit dependence on both the terrain slope and curvature. To illustrate this dependence, we consider experiments in which simulations using the full stress implementation of E07 are compared to companion simulations using each of the two approximate conditions in section 2.1.4—namely, (i) the normal gradient condition, which accounts for terrain slope but neglects curvature, and (ii) the flat boundary assumption, which ignores both the curvature and the slope. Note that while these experiments highlight the role of the terrain geometry, they also address the first of the questions listed above—specifically, how much of an impact does the full stress condition have relative to the approximations in widespread use.

We note that in the present study, we address the questions above using the model described by E07, which, as described in section 2.2.3, appears to be the only current model with a general implementation of the full stress condition. (For reference, a new implementation for the WRF model is presented in the following chapter). The E07 model was developed as an idealized dynamical code, which neglects more general physical processes such as microphysics and radiation. However, to make the current simulations more realistic, the model is implemented using terrain data for a specific region—namely, a region in Tongariro National Park, New Zealand—as well as realistic wind and stability profiles, taken from from the ERA5 reanalysis. To highlight the role of the surface drag (as opposed to heat fluxes), the experiments are carried out for cases with neutral or nearly neutral surface layers, as generally occur during the evening and morning transition periods.

The following section gives a brief overview of the model E07, as well as the data sources used in the study. Section 3.2 details the implementation of the normal gradient condition in the model (which broadly follows the method described by E07 for the full condition), along with basic physical arguments to illustrate the dependence of the drag condition on the terrain geometry. Section 3.4 describes the results of experiments using randomly chosen cases, so as to give a sense of the behavior of the drag condition under a range of different flow conditions. The behavior of the drag condition under varying grid resolution is described in section 3.5. Section 3.6 discusses some general implications of the results for wind energy applications, while a summary and conclusions are presented in section 3.7.

3.2 Experimental design

3.2.1 Physical framework

We consider the non-rotating, compressible Boussinesq problem described by

$$\frac{D\mathbf{u}}{Dt} = -\boldsymbol{\nabla}P + b\,\mathbf{k} - \boldsymbol{\nabla}\cdot\boldsymbol{\tau} \tag{3.1}$$

$$\frac{Db}{Dt} + N^2 w = -\boldsymbol{\nabla}\cdot\mathbf{B} \tag{3.2}$$

$$\frac{DP}{Dt} + c_s^2\,\boldsymbol{\nabla}\cdot\mathbf{u} = 0 \tag{3.3}$$

where $\mathbf{u} = (u, v, w) = (u_1, u_2, u_3)$ is the velocity, P is the Boussinesq disturbance pressure and b the buoyancy, N is the basic-state static stability (which depends on height), and c_s is the constant sound speed. The inclusion of compressibility in (3.3) allows the system to be integrated using explicit time integration methods, as described below. However, for sufficiently small Mach numbers (as in the present case), the flow is effectively incompressible, in which case (3.1)–(3.3) reduces to the conventional Boussinesq system. For further discussion of the compressible Boussinesq problem (including scaling arguments and range of validity), see Appendix C of Epifanio & Rotunno (2005).

The viscous stress tensor $\boldsymbol{\tau}$ in (3.1) is defined by the conventional form (2.2). A Boussinesq potential temperature variable for the system (3.1)–(3.3) can be defined (to within a constant) by

$$\theta = \int_0^z N^2(z')\,dz' + b$$

in which case the diffusive heat flux in (3.2) takes the form

$$B_j = -\kappa\frac{\partial\theta}{\partial x_j} \tag{3.4}$$

where the thermal diffusivity and viscosity have been assumed equal.

The initial and boundary conditions for the problem are defined using realistic sounding and terrain data, as described further in sections 3.2.2 and 3.2.3 below.

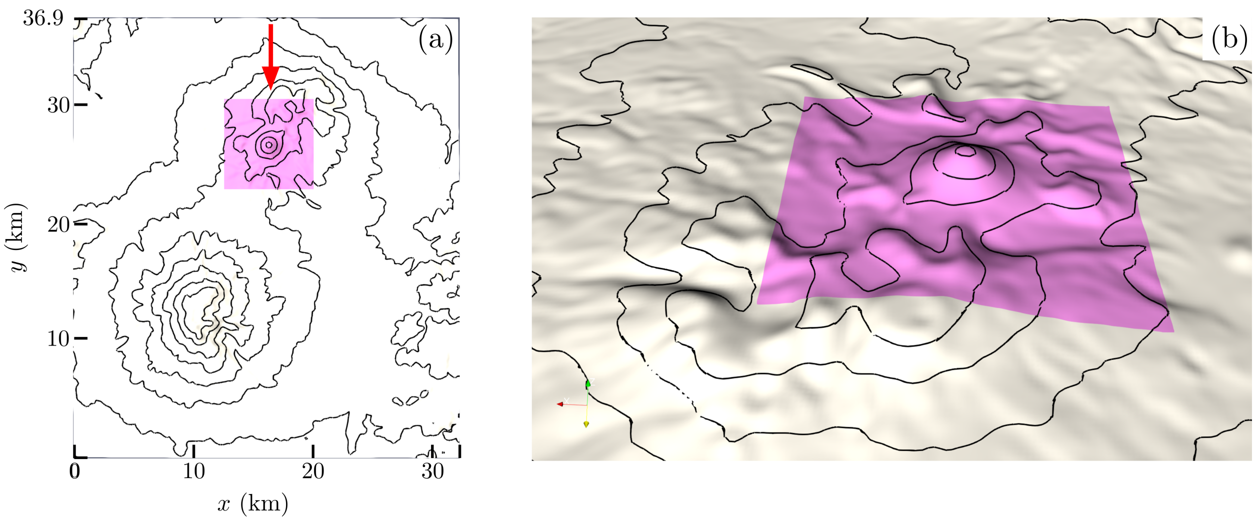


Figure 3.1: (a) Topography on the simulation domain (contour interval = 250m), showing Mts. Ngauruhoe (to the north) and Ruapehu (to the south). Pink shading surrounding Mt. Ngauruhoe shows the region used for quantitative error calculations, as described in the text. (b) Three-dimensional rendering of the region surrounding Mt. Ngauruhoe, as viewed from the direction indicated by the red arrow in (a). Solid lines show terrain contours, as in (a).

3.2.2 Test case and data sources

The test domain for our experiments consists of a pair of stratovolcanoes—specifically, Mt. Ngauruhoe and Mt. Ruapehu—and the surrounding regions of rough terrain in Tongariro National Park, New Zealand, as shown in Fig. 3.1. This region was chosen in part due to the strong (roughly 30°) slopes on the sides of Mt. Ngauruhoe but also due to the very localized nature of the areas of complex terrain. As seen in Fig. 3.1, away from the two volcanoes, the topography becomes relatively flat, thus allowing the impact of the terrain to be easily attributable to specific terrain features.

The terrain was represented in the model using data obtained from the NASA Shuttle Radar Topographic Mission (SRTM) 3 arc-second (or roughly 90 m) digital elevation database, quality controlled to account for missing data.[1] The initial state for the model was defined using data from

[1]Note that at the time, the 90 m SRTM was the highest resolution terrain dataset available.

the ECMWF $0.25^{\circ} \times 0.25^{\circ}$ reanalysis (ERA5). Given the limited domain size, a single ERA5 grid point (the nearest point) was taken to be representative of the model background state, with the sounding from this point applied uniformly in the horizontal at the initial time.

To focus attention on the effects of surface drag (as opposed to heat fluxes), cases were selected in which the surface layer was expected to be close to neutral, as predicted by the ERA5 Monin-Obukhov length. To be specific, 10 dates were first chosen randomly from the year 2012. Starting from each date, the nearest case in which the ERA5 Monin-Obukhov length exceeded 500 was then selected, implying nearly neutral surface conditions (e.g., Sathe et al., 2015). More specifically, for each date, the nearest morning and nearest afternoon / evening cases to satisfy the criteria were both selected, loosely representing the morning and evening transitions.

3.2.3 Numerical description

The numerical model used in this study is a version of the 3D compressible Boussinesq code described in Epifanio & Durran (2001). The model solves (3.1)–(3.3) using the partial time-splitting algorithm of Klemp & Wilhelmson (1978) [see also Durran (1999)] to handle the acoustic modes. Spatial discretization is done through finite differencing on a C-grid (see Durran (1999)), with fourth order centered differencing in the horizontal and second-order in the vertical. Topography is incorporated using the terrain-following coordinate transformation of Gal-Chen & Somerville (1975). Unresolved turbulence is handled through a Smagorinsky type turbulence parameterization, while surface drag is represented using a standard roughness-length formulation, as described in section 2.2.4.

Except where otherwise indicated, the horizontal grid spacing is uniform at 90 m (matching the resolution of the terrain data). The vertical grid features geometric stretching, with a grid spacing of 30 m near the lower boundary and vertical stretching factor of 1.026 between grid levels. The upper 5.7 km of the domain is implemented as a Rayleigh sponge layer, with a gravity wave radiation condition applied at the grid upper boundary. A sponge layer of width 5 km is applied at all lateral boundaries.[2] The lower boundary is assumed thermally insulating, consistent with the

[2]The damping magnitude in all cases varies as $0.5\,[1 + \cos(\pi d/L_s)]$, where L_s is the width of the damping zone

emphasis on nearly neutral surface layers.

3.3 Implementation and dependence on terrain geometry

In the present experiments, the full version of the drag condition is implemented using the matrix method of E07, as described in section 2.2.3. Below we describe the implementation of the two approximate versions of the drag condition, namely, the normal gradient and flat boundary approximations. We also present arguments to illustrate the role of the terrain geometry in the behavior the effect of the boundary condition.

3.3.1 The normal gradient condition: matrix implementation

As in chapter 2, we let $\mathbf{s}$ and $\mathbf{t}$ be the tangent vectors to the terrain the xz and yz-planes, respectively. Combining (2.22) with (2.4) then shows the normal gradient condition as projected into the $\mathbf{s}$ and $\mathbf{t}$ directions can be written in the form

$$\frac{\partial}{\partial n}(un_3 - wn_1) = n_3\frac{\partial u}{\partial n} - n_1\frac{\partial w}{\partial n} = -\frac{D_x}{\kappa}\sqrt{n_1^2+n_3^2} \tag{3.5a}$$

$$\frac{\partial}{\partial n}(vn_3 - wn_2) = n_3\frac{\partial v}{\partial n} - n_2\frac{\partial w}{\partial n} = -\frac{D_y}{\kappa}\sqrt{n_2^2+n_3^2} \tag{3.5b}$$

For any variable ψ, the normal derivative is defined by

$$\frac{\partial \psi}{\partial n} = \nabla\psi\cdot\hat{n} = n_1\frac{\partial \psi}{\partial x} + n_2\frac{\partial \psi}{\partial y} + n_3\frac{\partial \psi}{\partial z} \tag{3.6}$$

which suggests that the conditions (3.5) can be rewritten as

$$n_3n_1\frac{\partial u}{\partial x} + n_3n_2\frac{\partial u}{\partial y} + n_3^2\frac{\partial u}{\partial z} - n_1^2\frac{\partial w}{\partial x} - n_1n_2\frac{\partial w}{\partial y} - n_1n_3\frac{\partial w}{\partial z} = -\frac{D_x}{\kappa}\sqrt{n_1^2+n_3^2} \tag{3.7a}$$

$$n_3n_1\frac{\partial v}{\partial x} + n_3n_2\frac{\partial v}{\partial y} + n_3^2\frac{\partial v}{\partial z} - n_2n_1\frac{\partial w}{\partial x} - n_2^2\frac{\partial w}{\partial y} - n_2n_3\frac{\partial w}{\partial z} = -\frac{D_x}{\kappa}\sqrt{n_2^2+n_3^2} \tag{3.7b}$$

and d is the distance from the boundary.

Suppose we use (2.9) and (2.11) to transform (3.7) to the terrain following coordinate (X, Y, q). Keeping in mind $n_1^2 + n_2^2 + n_3^2 = 1$, the end result looks like

$$\begin{aligned} n_3 n_1 \frac{\partial u}{\partial X} + n_3 n_2 \frac{\partial u}{\partial Y} + \gamma n_3 \frac{\partial u}{\partial q} - n_1^2 \frac{\partial w}{\partial X} - n_1 n_2 \frac{\partial w}{\partial Y} \\ - \gamma n_1 \frac{\partial w}{\partial q} = -\frac{D_x}{\kappa}\sqrt{n_1^2 + n_3^2} \end{aligned} \tag{3.8a}$$

$$\begin{aligned} n_3 n_1 \frac{\partial v}{\partial X} + n_3 n_2 \frac{\partial v}{\partial Y} + \gamma n_3 \frac{\partial v}{\partial q} - n_2 n_1 \frac{\partial w}{\partial X} - n_2^2 \frac{\partial w}{\partial Y} \\ - \gamma n_1 \frac{\partial w}{\partial q} = -\frac{D_x}{\kappa}\sqrt{n_2^2 + n_3^2} \end{aligned} \tag{3.8b}$$

where for the GS75 coordinate, γ is described by (2.10).

The conditions (3.8) are implemented by discretizing as in (2.29) and (2.30) and applying a similar matrix approach as in section 2.2.3. In fact, comparing (2.12) and (3.8) shows that the full stress condition is the same as (3.8), but with extra terms added. (We return to this point in chapter 4.) The normal gradient condition can then be imposed using the same matrix solver as the full condition, but with the extra terms set to zero.

3.3.2 The flat boundary assumption

The flat boundary approximation can take different forms, depending on which parts of the boundary condition and the drag parameterization are approximated. In the present study, the flat boundary approximation involves the following assumptions: (i) the drag condition is implemented using (2.18), following a similar approach to that described for DK83 in section 2.2.1; (ii) the reference location for (2.38) is taken to be the first full grid level above the ground vertically (as opposed to along the terrain normal); and (iii) the reference wind in (2.38) is computed using just the horizontal part of the wind. It is believed that these approximations are representative of most current generation atmospheric models.

3.3.3 Dependence on terrain geometry

The dependence of the drag condition on the terrain geometry is most easily illustrated in the 2D case. The end result can be formally derived from (2.13) with (2.15). However, here we adopt a more basic derivation, to emphasize some of the geometric properties.

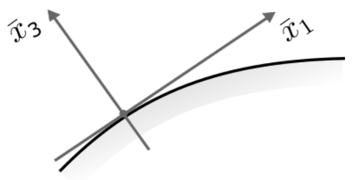


Figure 3.2: Rotated Cartesian coordinates at the point of interest.

For a given point on the terrain, suppose we define a rotated Cartesian coordinate system ($\bar{x}_1$, $\bar{x}_3$), as shown in Fig. 3.2. Note that the $\bar{x}_3$ axis is normal to the terrain and the $\bar{x}_1$ axis is tangent to the boundary (and with $\bar{x}_2$ implicitly in the uniform direction), and let the velocity vector in these coordinates be $\bar{\mathbf{u}} = (\bar{u}_1, \bar{u}_3)$. In the rotated coordinates, the tangent and normal vectors are simply

$$\bar{\mathbf{s}} = (1, 0) \qquad , \qquad \bar{\mathbf{n}} = (0, 1)$$

so that the rotated form of the stress condition (2.3) with (2.2) reduces to

$$\frac{\partial \bar{u}_1}{\partial \bar{x}_3} + \frac{\partial \bar{u}_3}{\partial \bar{x}_1} = -\frac{D_x}{\kappa} \tag{3.9}$$

Since the flow at the boundary must be tangent, the $\bar{u}_3$ component is zero at the point of interest. The $\partial \bar{u}_3 / \partial \bar{x}_1$ term in (3.9) must then depend on the change in orientation of the boundary—i.e., on the boundary curvature.

Consider two nearby points on the terrain surface separated by an angle $\delta\phi$ with respect to the osculating circle for the terrain, as shown in Fig. 3.3. In the limit of small $\delta\phi$, the separation $\delta\bar{x}_1$ between the points is then

$$\delta\bar{x}_1 \simeq R\,\delta\phi$$

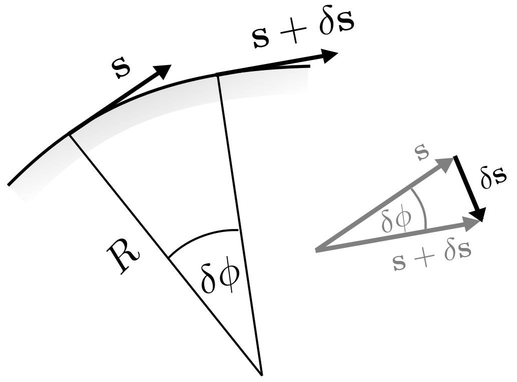


Figure 3.3: Geometry of the tangent direction on a curved surface.

where R is the radius of curvature. In terms of
signs, note that both R and $\delta\phi$ are negative for convex terrain (ridges), and positive for concave terrain (valleys).

Suppose we let the (scalar) tangential component of the wind be V (which can again be positive or negative). The difference in velocity between the two points is then

$$(V + \delta V)\,(\mathbf{s} + \delta\mathbf{s}) - V\mathbf{s} \,\simeq\, \delta V\mathbf{s} + V\,\delta\mathbf{s}\;. \tag{3.10}$$

But in the limit of small $\delta\phi$, we can write $\delta\mathbf{s} \simeq \delta\phi\,\mathbf{n}$ (see Fig. 3.3), suggesting from (3.10) that the change in the Cartesian $\bar{u}_3$ component between the points must be

$$\delta\bar{u}_3 \simeq V\delta\phi$$

which further implies that the $\partial\bar{u}_3/\partial\bar{x}_1$ term in (3.9) must look like

$$\frac{\delta\bar{u}_3}{\delta\bar{x}_1} \simeq \frac{V\,\delta\phi}{R\,\delta\phi} = \frac{V}{R}\;. \tag{3.11}$$

Since at the point of interest the tangential component of the wind is $V = \mathbf{u}\cdot\mathbf{s} = u_s = \bar{u}_1$, our rotated form of the stress condition can finally be written as

$$\frac{\partial u_s}{\partial n} + \frac{u_s}{R} = -\frac{D_x}{\kappa} \tag{3.12}$$

where as before, $\partial/\partial n$ is the derivative in the normal (in this case $\bar{x}_3$) direction.

According to (3.12), the behavior of the stress at the boundary depends on two aspects of the terrain geometry: (i) the slope of the boundary (through the normal derivative) and (ii) the local terrain curvature. As a general rule, we can expect the tangential drag $D_x = \mathbf{D}\cdot\mathbf{s}$ to act in an opposite sense to the free stream tangential wind, suggesting that the net impact of D_x is to decrease the wind speed at the ground, relative to the free stream value (or equivalently, to increase the magnitude of the normal shear). On the other hand, the effect of the curvature term in (3.12) depends on the sense of the curvature—specifically, convex terrain ($R < 0$) is expected to experience a larger wind speed deficit at the ground (or equivalently, larger wind shear), while concave terrain ($R > 0$) tends to produce to a smaller wind speed deficit. The size of this curvature effect depends both on the magnitude of the surface wind speed and on the amount of curvature in the terrain, with more strongly curved terrain (smaller R) producing a bigger impact. That said, in the atmospheric context, we generally expect to have larger wind speeds over peaks and ridges than over valleys, suggesting that all else being equal, the effects of curvature over convex features will likely outweigh those over the corresponding concave terrain.

Similar considerations apply in the three dimensional (3D) case as well, although the role of terrain curvature in that case is somewhat more involved mathematically, due to the additional degree of freedom.

3.4 Overview of cases

As described in section 3.2.2, simulations were carried out for 20 randomly chosen cases from 2012. All cases were selected to have nearly neutral surface layers (Monin-Obukhov length greater than 500), with half the cases occurring during the afternoon or evening hours (loosely representing the evening transition), and the other half during the early morning or morning hours (morning transition). For each case, simulations were carried out using three different versions of the drag condition—specifically, the full stress condition (as described in 2.2.3, along with both the normal gradient and flat boundary approximations (2.1.4).

For the purposes of analysis, we adopt a perfect model framework, in which the simulations using the full stress condition are taken to accurately represent the flow. For the simulations using the approximate drag conditions (normal gradient and flat boundary), any differences from the full stress simulations are then treated as model errors, associated with the use of the approximate boundary conditions. All results that follow are shown at a run time of 30 minutes, at which point the flow at the lower boundary was found to be roughly steady.

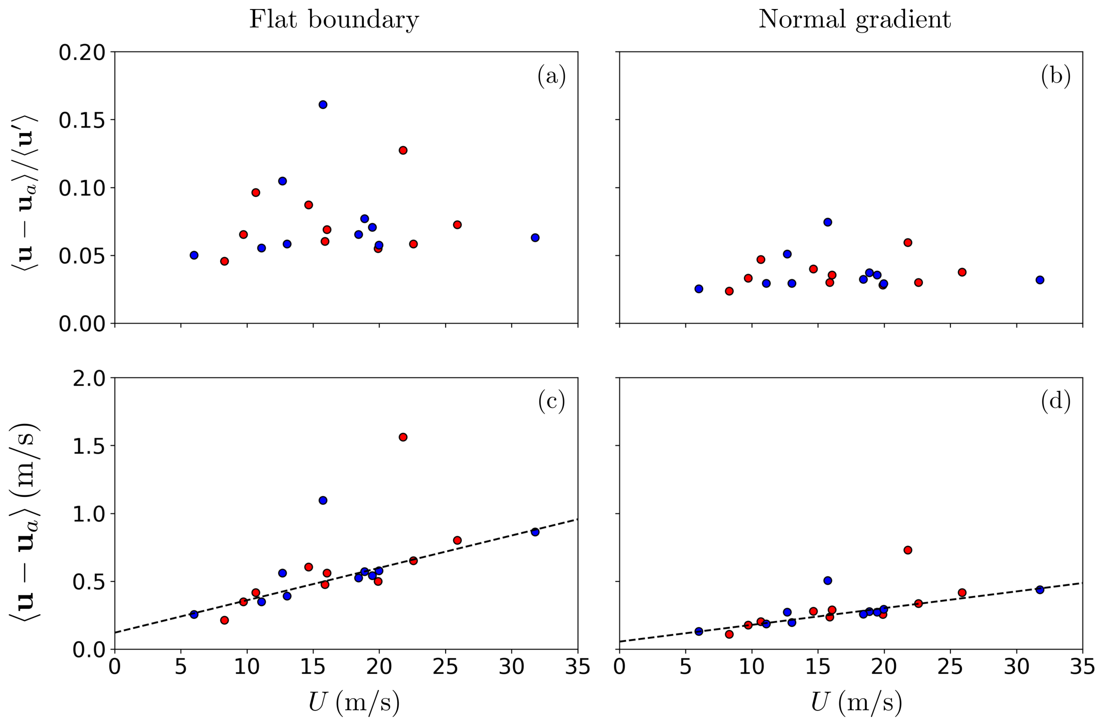


Figure 3.4: Errors in the low-level winds for simulations using the flat boundary [(a) and (c)] and normal gradient [(b) and (d)] approximations. Red dots indicate afternoon or evening cases, while blue dots show morning cases. All errors are root mean square values, as averaged over the subdomain indicated in Fig. 3.1. (See text for details.) (a) and (b) show errors normalized by the mean disturbance wind speed on the error averaging domain, while (c) and (d) show net errors in units of m/s. Dashed lines in (c) and (d) show the linear least-squares regression (two outliers excluded).

3.4.1 Error analysis

The overall errors associated with use of the flat boundary approximation in our experiments are summarized in Figs. 3.4a and c, while the associated errors for the normal gradient approximation are shown in Figs. 3.4b and d. Each data point in the figure represents the error for one particular case (i.e., one date and time), plotted as a function of the low-level upstream wind speed for each case (defined as the mean background wind speed below the peak of Mt. Ngauruhoe). Red points indicate afternoon / evening cases, while morning cases are shown as blue.

For quantitative purposes, each of the errors in Fig. 3.4 reflects the root mean square (rms) vector wind difference between two otherwise identical model runs: one using the approximate boundary condition (normal gradient or flat boundary) and the other using the full stress condition. The errors are computed for the first interior grid half level (approximately 15 m above the ground) and then averaged over a subdomain including Mt. Ngauruhoe and the surrounding area of rough terrain, as illustrated in Fig. 3.1. In Figs. 3.4a and b, the errors are normalized by the rms disturbance wind speed on the averaging domain, as given by the fractional error measure

$$E_a = \frac{\langle \mathbf{u} - \mathbf{u}_a \rangle}{\langle \mathbf{u}' \rangle} \tag{3.13}$$

where $\mathbf{u}$ and $\mathbf{u}_a$ are the winds for the full stress case and for the run with the approximate boundary condition, respectively, and where $\mathbf{u}'$ is the disturbance from the background state, as evaluated for the full stress case. The angle brackets in (3.13) represent the rms average (where the square in this case reflects the squared vector magnitude), as averaged over all grid points in the relevant subdomain. In Figs. 3.4c and d, the errors are presented without the normalization factor—i.e., showing just the numerator in (3.13).

As seen in Fig. 3.4, the normalized errors associated with use of the flat boundary approximation range from roughly 0.04 to as high as 0.17, with mean and median values of approximately 0.075 and 0.065, respectively. In dimensional (not normalized) terms (Fig. 3.4c), the corresponding errors range from approximately 0.2 m/s on the low end to roughly 1.6 m/s at the upper end, with a mean value of 0.59 m/s. As seen in Fig. 3.4c, apart from a pair of outliers, the dimensional errors in the wind are found to vary roughly linearly with the background wind speed. A linear least squares curve fit (excluding the two outliers) predicts a slope of 0.024, although the presence of a nonzero intercept suggests that additional data points are needed to capture the dependence at small windspeeds. It is worth noting that the two outlier cases in Fig. 3.4c appear to be mainly distinguished by the direction of the background winds—further discussion of these cases is given in section 3.4.4.

As shown in Figs. 3.4b and d, applying the normal gradient condition leads to a significant reduction in the low-level vector wind errors, relative to the flat boundary approximation. In quantitative terms, the errors for the normal gradient case are roughly half those of the corresponding flat boundary flows across the board, including both the mean and median error values (both normalized and dimensional), as well as the slope and intercept of the linear least squares fit. Recall from section 3.3.3 that in its full, unapproximated form, the way in which the stress at the boundary interacts with the flow depends on both the slope and curvature of the underlying terrain. Comparing the flat boundary and normal gradient results then suggests that, at least for the current case, the slope and curvature play roughly equal roles, in the sense that neglecting both the slope and the curvature (as in the flat boundary approximation) leads to twice as much error as neglecting just the curvature, while accounting for slope (normal gradient).

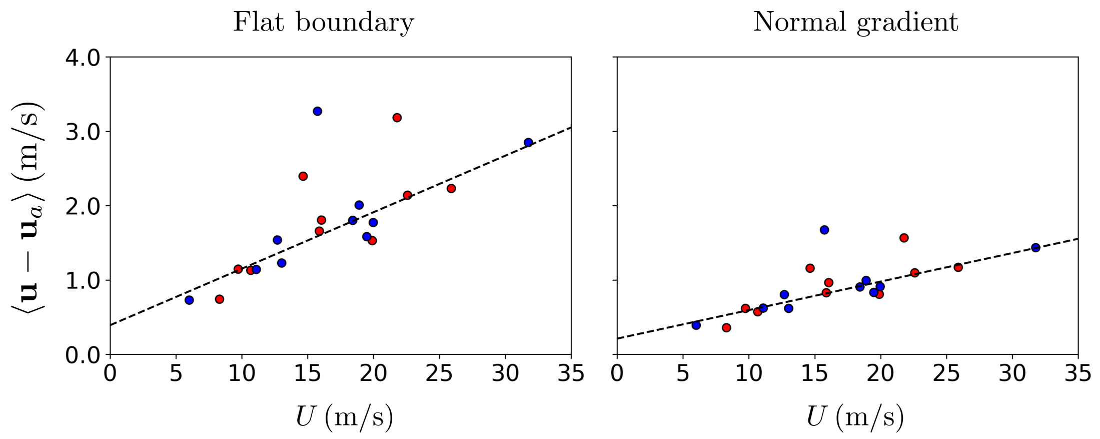


Figure 3.5: As in Figs. 3.4c and d, but including only the largest 5% of errors on the averaging domain. Shown are mean errors in m/s for the (a) flat boundary and (b) normal gradient cases.

Finally, it is worth keeping in mind that the errors shown in Fig. 3.4 are mean values taken over an averaging region, as illustrated in Fig. 3.1. However, as shown in section 3.4.2, in practice the errors tend to be fairly localized to particular points in the flow, as tied to specific terrain features, suggesting that the errors at certain points may be significantly larger than the mean values. To give a sense of these more localized errors, Fig. 3.5 shows the average magnitude of the vector wind difference, as in Figs. 3.4c and d, but with the average taken over just the largest 5% of the errors among all the grid points on the averaging domain for any given case. As shown in Fig. 3.5a, if only the top 5% of values are included, the errors for the flat boundary case increase to a range of roughly 0.8 m/s on the low end up to 3.3 m/s on the upper end. The mean value for Fig. 3.5a is 1.8 m/s, while the predicted slope is 0.076, both of which suggest that the average over the top 5% of errors is roughly three times larger than the mean values for the whole averaging domain. And as with Fig. 3.4, applying the normal gradient condition reduces the errors by roughly half, relative to the flat boundary case (Fig. 3.5b).

3.4.2 Spatial patterns

Figure 3.6 gives a sense of the spatial distribution in the errors associated with the use of the flat boundary approximation, as sampled for four representative cases from Fig. 3.4. Shown in the figure is the magnitude of the vector wind difference between runs using the full stress and flat boundary conditions, as evaluated at the first interior grid level (approximately 15 m above ground). The errors for each case are normalized by the maximum error on the averaging domain, so that the plotted values in each panel range from zero to one; specifically,

$$e_f = \frac{|\,\mathbf{u} - \mathbf{u_f}\,|}{|\,\mathbf{u} - \mathbf{u_f}\,|_{max}} \tag{3.14}$$

where $\mathbf{u}$ and $\mathbf{u_f}$ are the winds for the full stress and flat boundary simulations, respectively, and where the denominator is the maximum value on the averaging domain from Fig. 3.1. (The corresponding maximum error in each case is indicated by the figure labels.) To draw attention to regions with larger error values, the colors are defined so as to transition from bluish to reddish shades starting at a value of roughly 0.1, as illustrated by the associated colorbar. Small green arrows show the corresponding wind difference $\mathbf{u} - \mathbf{u_f}$ plotted in vector form, while thick black arrows indicate the mean upstream wind directions below the peak of Mt. Ngauruhoe.

In each of the four cases shown, the largest errors in the winds are associated with particular characteristics of the terrain, most notably with small ridge-like features whose axes are aligned roughly perpendicular to the incoming winds. The largest errors are for the most part found at or

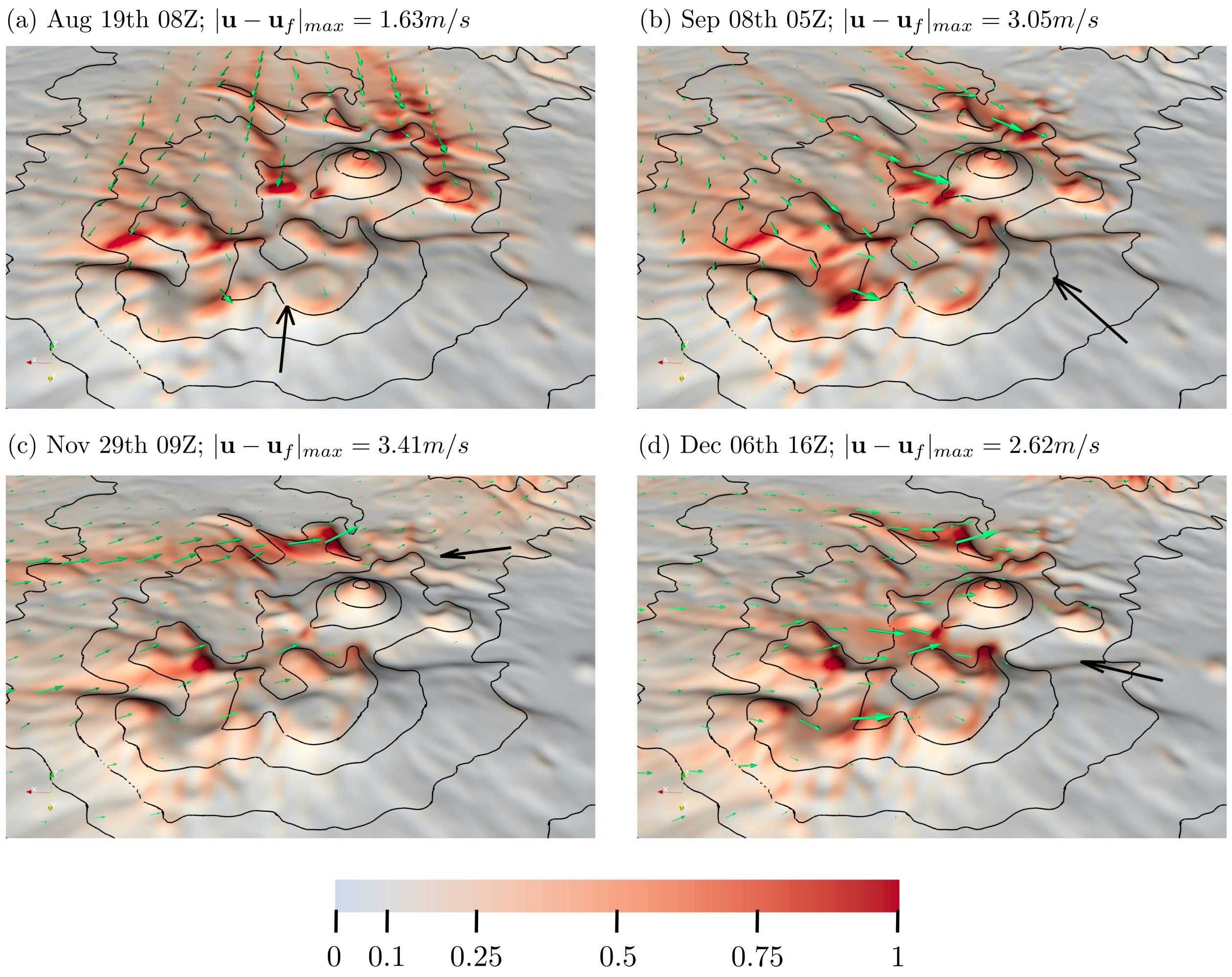

(a) Aug 19th 08Z; $|\mathbf{u} - \mathbf{u}_f|_{max} = 1.63m/s$
(b) Sep 08th 05Z; $|\mathbf{u} - \mathbf{u}_f|_{max} = 3.05m/s$
(c) Nov 29th 09Z; $|\mathbf{u} - \mathbf{u}_f|_{max} = 3.41m/s$
(d) Dec 06th 16Z; $|\mathbf{u} - \mathbf{u}_f|_{max} = 2.62m/s$
0
0.1
0.25
0.5
0.75
1

Figure 3.6: Spatial distributions of the low-level wind errors for simulations using the flat boundary approximation, as sampled for four representative cases from Fig. 3.4. Colors show the magnitude of the vector wind difference between simulations using the full stress condition and the flat boundary approximation, as normalized by the maximum value of the error on the subdomain shown in Fig. 3.1. (See text for details.) Green vectors show the wind difference $\mathbf{u} - \mathbf{u_f}$ in vector form, while thick black arrows show the low-level upstream wind direction. Solid black lines show contours of constant terrain height (c.i. = 250m). Shown are results for (a) Aug 19th, 08Z; (b) Sep 08th, 05Z; (c) Nov 29th, 09Z; and (d) Dec, 16th 06Z. The view in all cases is roughly from the north, as indicated by the red arrow in Fig. 3.1.

near the peaks of the ridges, where the effects of terrain curvature on the stress are expected to be largest, although in many cases the errors extend along the nearby slopes as well. The sense of the vectors shown in Fig. 3.6 suggests that as general rule, the simulations with the full stress condition experience stronger deceleration of the flow, relative to the flat boundary simulations, leading to difference vectors oriented more or less opposite to the upstream winds. Note that in most cases, the errors generated at the peaks and ridges are subsequently carried downstream, leading to wakes of decelerated winds in the simulations with the full stress condition. A clear example can be found in the case from November 29 (Fig. 3.6c), where a small ridge on the far side of the volcano (as viewed in the figure) is associated with a prominent wake, extending far downstream of the terrain. Similar examples can be found in each of the the remaining three cases, as well.

Figure 3.7 shows the spatial distributions of the error for simulations using the normal gradient condition, with the color ranges defined as in Fig. 3.6 (i.e., using the same normalization factors), so as to allow direct comparison between the figures. As expected, relative to the flat boundary case, the errors in the normal gradient simulations are greatly reduced, by roughly a factor of two. Comparing the two figures suggests that the largest errors for the normal gradient case are more closely tied to regions of large curvature, particularly at the peaks of small ridges. That said, the overall error patterns in Fig. 3.7 are largely similar to those seen in Fig. 3.6, with errors generated over small ridge-like features and subsequently extending downstream. As anticipated by section 3.3.3, the sense of the errors suggests that the flow is more strongly decelerated in the simulations using the full stress condition, as the inclusion of curvature effects leads to larger flow deficits near the ground.

3.4.3 Dependence on surface roughness

It is worth noting that during certain parts of the year, Tongariro National Park is largely covered by snow, leading to a significant reduction in the surface roughness. To get a sense of the effect of reduced surface roughness on the behavior of the stress, Fig. 3.8 shows a series of experiments identical to those shown in Fig. 3.4, except with the surface roughness length set to 0.0003 m, characteristic of smooth, snow-covered ground.

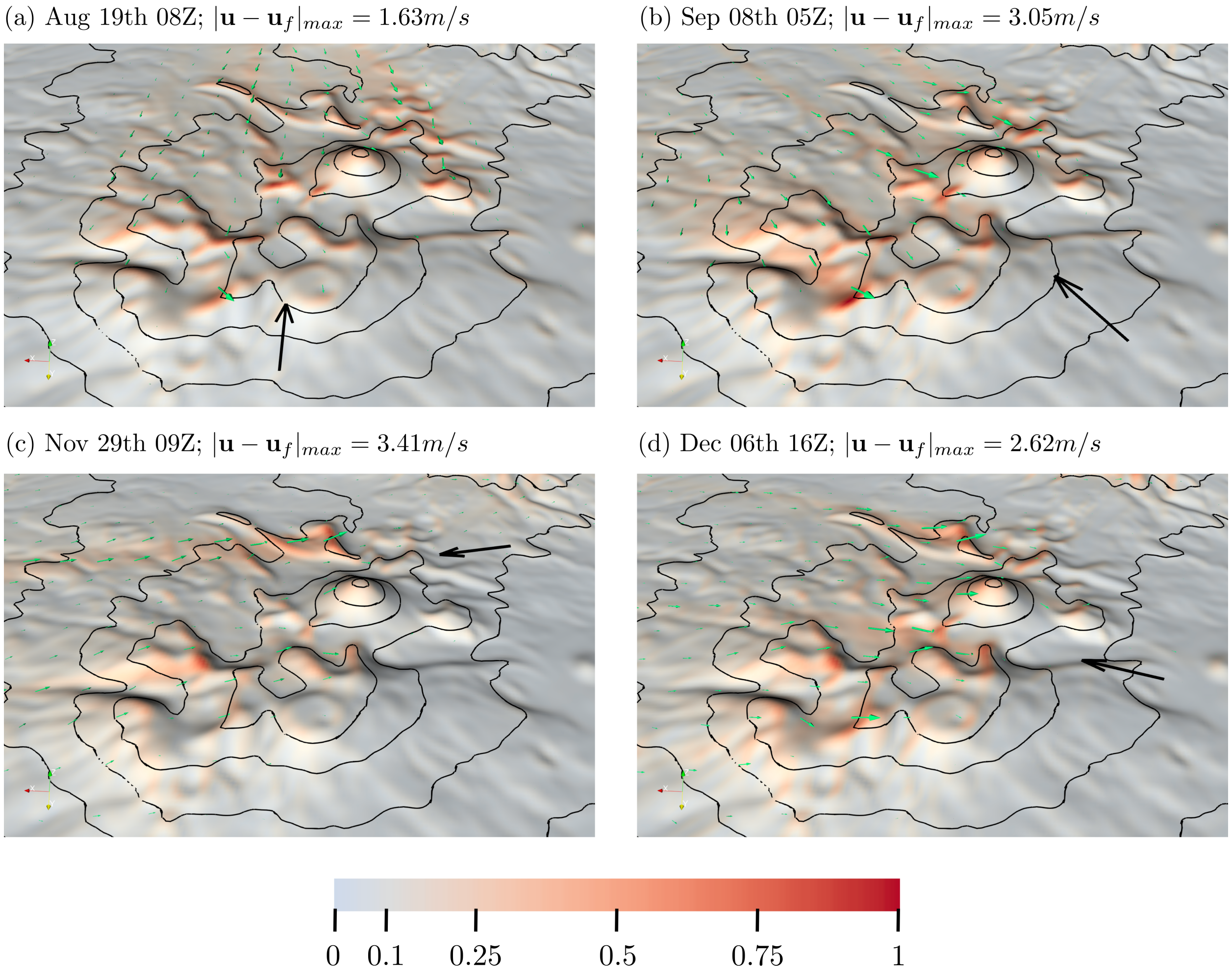

(a) Aug 19th 08Z; $|\mathbf{u}-\mathbf{u}_f|_{max} = 1.63m/s$
(b) Sep 08th 05Z; $|\mathbf{u}-\mathbf{u}_f|_{max} = 3.05m/s$
(c) Nov 29th 09Z; $|\mathbf{u}-\mathbf{u}_f|_{max} = 3.41m/s$
(d) Dec 06th 16Z; $|\mathbf{u}-\mathbf{u}_f|_{max} = 2.62m/s$
0
0.1
0.25
0.5
0.75
1

Figure 3.7: As in Fig. 3.6, but for simulations using the normal gradient approximation. Shown are results for (a) Aug 19th, 08Z; (b) Sep 08th, 05Z; (c) Nov 29th, 09Z; and (d) Dec 16th, 06Z.

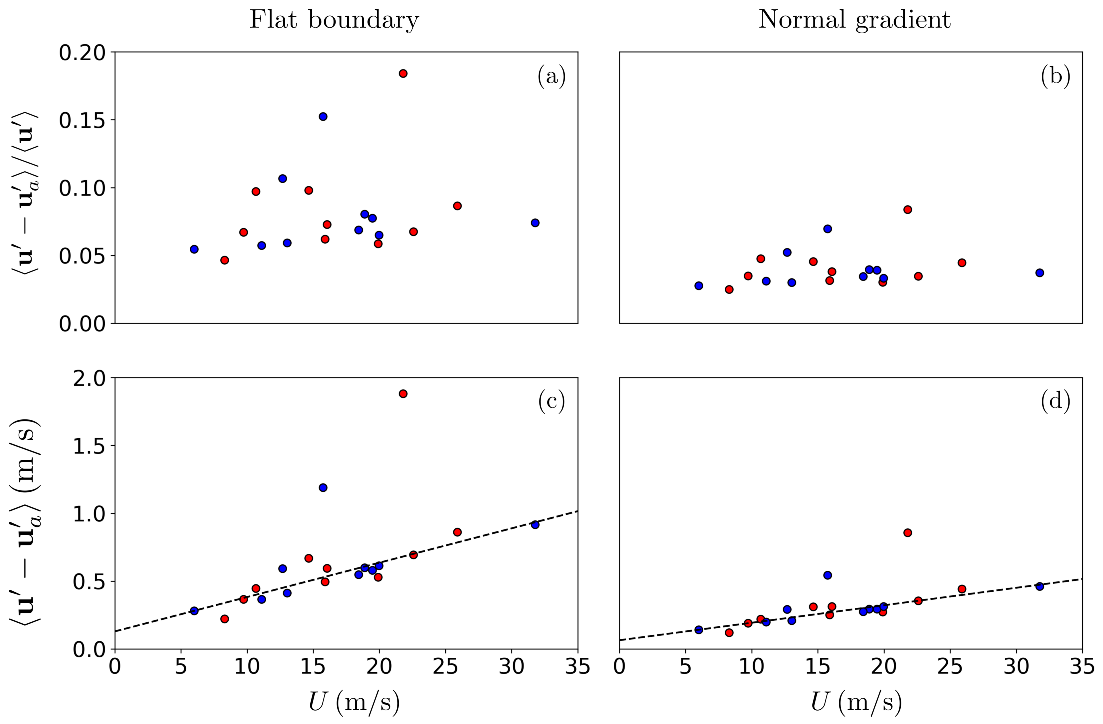


Figure 3.8: As in Fig. 3.4, but for simulations using a reduced roughness length of 0.0003 m.

Comparison of Figs. 3.4 and 3.8 shows that on average, the reduction in surface roughness has only a modest impact on the errors, both for the flat boundary and normal gradient approximations. Quantitatively speaking, the experiments with reduced surface roughness result in slightly larger errors, with the mean error for both the flat boundary and normal gradient experiments increasing by roughly 10% relative to the results in Fig. 3.4. That said, it is worth noting that certain individual cases can show a much larger sensitivity. The clearest example is the outlier case with an upstream wind of $U = 22$ m/s, for which the normalized flat boundary error increases to over 0.18 (or a dimensional error of 1.8 m/s) in the experiments with reduced surface roughness, implying an increase of nearly 45% relative to the error with roughness length 0.005 m.

3.4.4 The outlier cases

As noted above, for most of our experiments, the errors are shown to be roughly proportional to the low-level background wind speed, regardless of the direction of the background winds. That

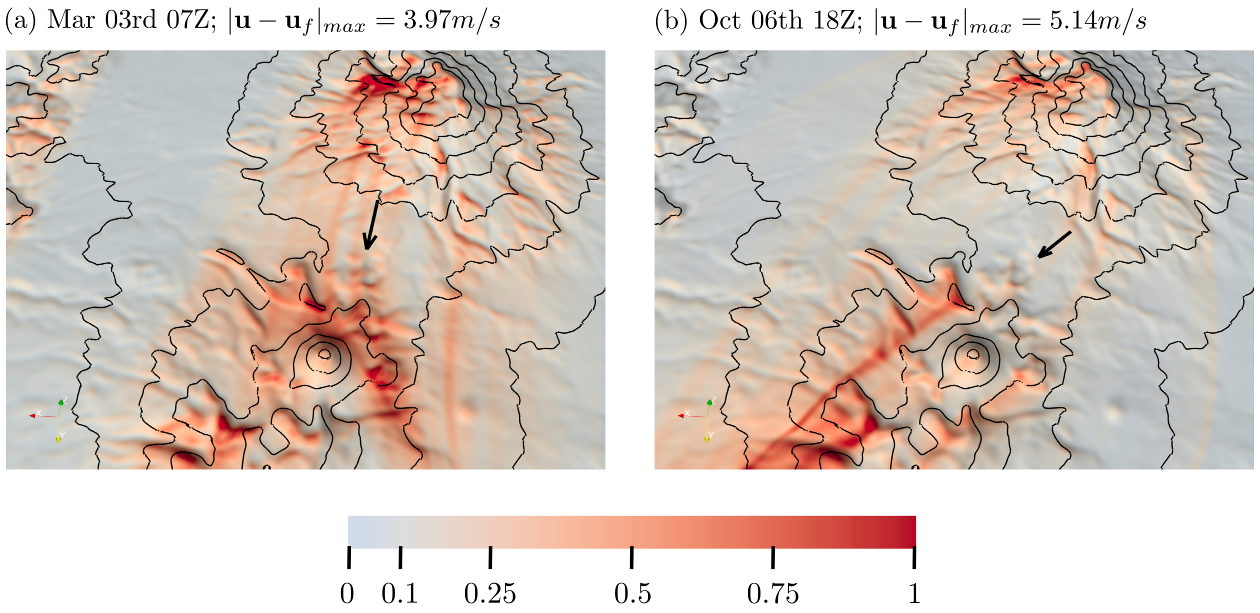


Figure 3.9: Low-level wind errors for the two outlier cases shown in Fig. 3.4, as viewed from a vantage point above and slightly to the north. Colors show the magnitude of the low-level vector wind difference between simulations using the full stress condition and the flat boundary approximation, normalized and plotted as described in Fig. 3.6. Thick black arrows show the direction of the low-level upstream winds.

said, the presence of the two outlier cases suggests that this lack of directionality may not in fact be a general result.

Figure 3.9 shows the magnitude of the errors in the flat boundary experiments for the two outlier cases, normalized and plotted as in Fig. 3.6, but showing an expanded (and somewhat more vertical) view. As seen in the figure, while the terrain surrounding Mt. Ngauruhoe is mostly gentle, there exists a range of wind directions (from roughly 0^o to 45^o) for which the flow first encounters Mt. Ruapehu to the south (cf. Fig. 3.1), with the wind for the March 03 case (Fig. 3.9a) showing a nearly direct alignment between the two mountains. As a result, there appears to be a compounding effect in the errors, as the interaction with Mt. Ruapehu leads to differences between the flat boundary and full stress simulations which extend downstream, particularly as seen for the March 03 case. These differences are then further amplified through the interaction with Mt. Ngauruhoe (and the surrounding terrain), ultimately leading to a disproportional increase in the errors on the averaging domain, as seen Fig. 3.4. A similar effect occurs (to a somewhat lesser extent) in the

Oct 06 outlier case (Fig. 3.9b), as the flow encounters the rough terrain to the northeast of Mt. Ruapehu before approaching the Mt. Ngauruhoe area.

In addition to upstream effects, a second factor which could lead to a degree of directionality in the results is simply the geometric properties of the terrain. If the terrain undulations tend to be preferentially aligned along a particular axis (as in the case of a elongated ridge), then flow directed roughly perpendicular to that axis will be expected to produce larger errors in the flat boundary and normal gradient approximations, relative to having the flow directed along the axis. This type of effect appears to play a significant role in the Oct 06 case, as the region of complex terrain surrounding Mt. Ngauruhoe features several prominent ridges aligned nearly perpendicular to the flow in that case, each of which results in significant errors.

Finally, yet another factor that might play a role in the production of outlier cases could be the presence of certain wind and stability profiles leading to an amplified response, such as a downslope windstorm or high-drag state. It is unclear whether such effects play a role in our experiments, but in principle, at least, if the disturbance flow is strongly accelerated, then the observed relationship between the stress and the upstream windspeed might not apply.

3.5 Resolution dependence and terrain smoothing

The results of section 3.3.3 and sections 3.4 show that the behavior of the surface stress has a strong dependence on the terrain geometry, particularly on the slope and curvature of the terrain. However, in the context of a numerical model, the geometry of the terrain can in turn be affected by numerical factors, such as the grid resolution or the amount of terrain smoothing applied. In the present section, we explore the impact of such factors for the case of flow past Mt. Ngauruhoe, using a typical case from Fig. 3.4 (i.e., a case with roughly average low-level wind errors) as an example.

3.5.1 Dependence on horizontal grid resolution

Figure 3.10 shows an example of the low-level wind errors for the flat boundary approximation, as simulated using horizontal grid spacings ranging from 90 m to 240 m (and with all other

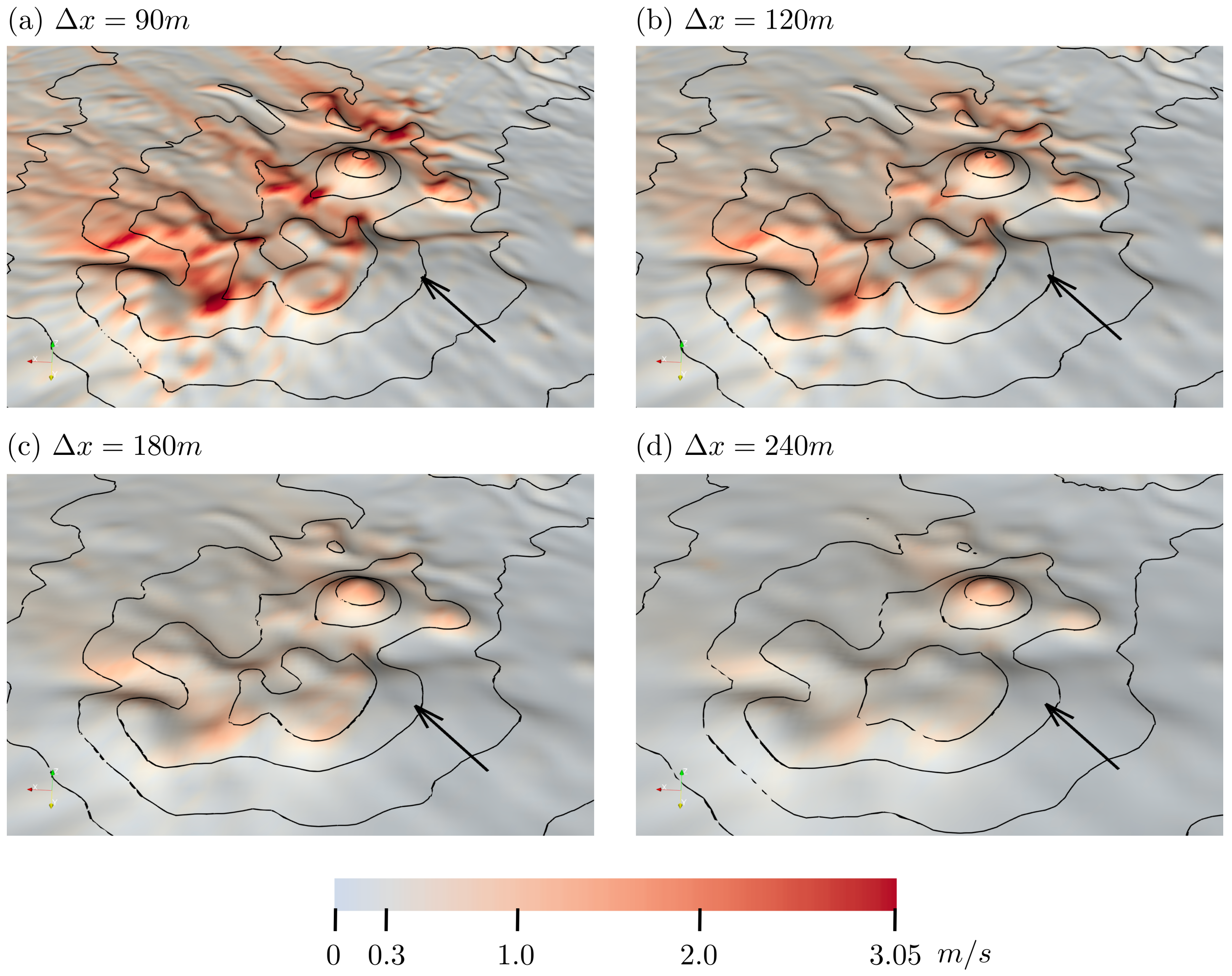


Figure 3.10: Error patterns in different horizontal resolution for the Sep 08th case between full-stress and flat-boundary conditions: (a) 90m; (b) 120m; (c) 180m; (d) 240m.

parameters as described in section 3.2). The case shown is the September 08 event illustrated in Fig. 3.6c, which is taken to be a typical case, in the sense that the normalized error for this case is roughly equal to the mean error over all twenty cases shown in Fig. 3.4a. As in Fig. 3.6, the color field in Fig. 3.10 shows the length of the vector wind difference between simulations using the flat boundary approximation and the full stress condition. However, note that unlike in Fig. 3.6, the results in Fig. 3.10 are unnormalized, allowing the size of the error to be directly compared between panels.

As can be seen in the figure, the geometry of the terrain as represented on the model grid is

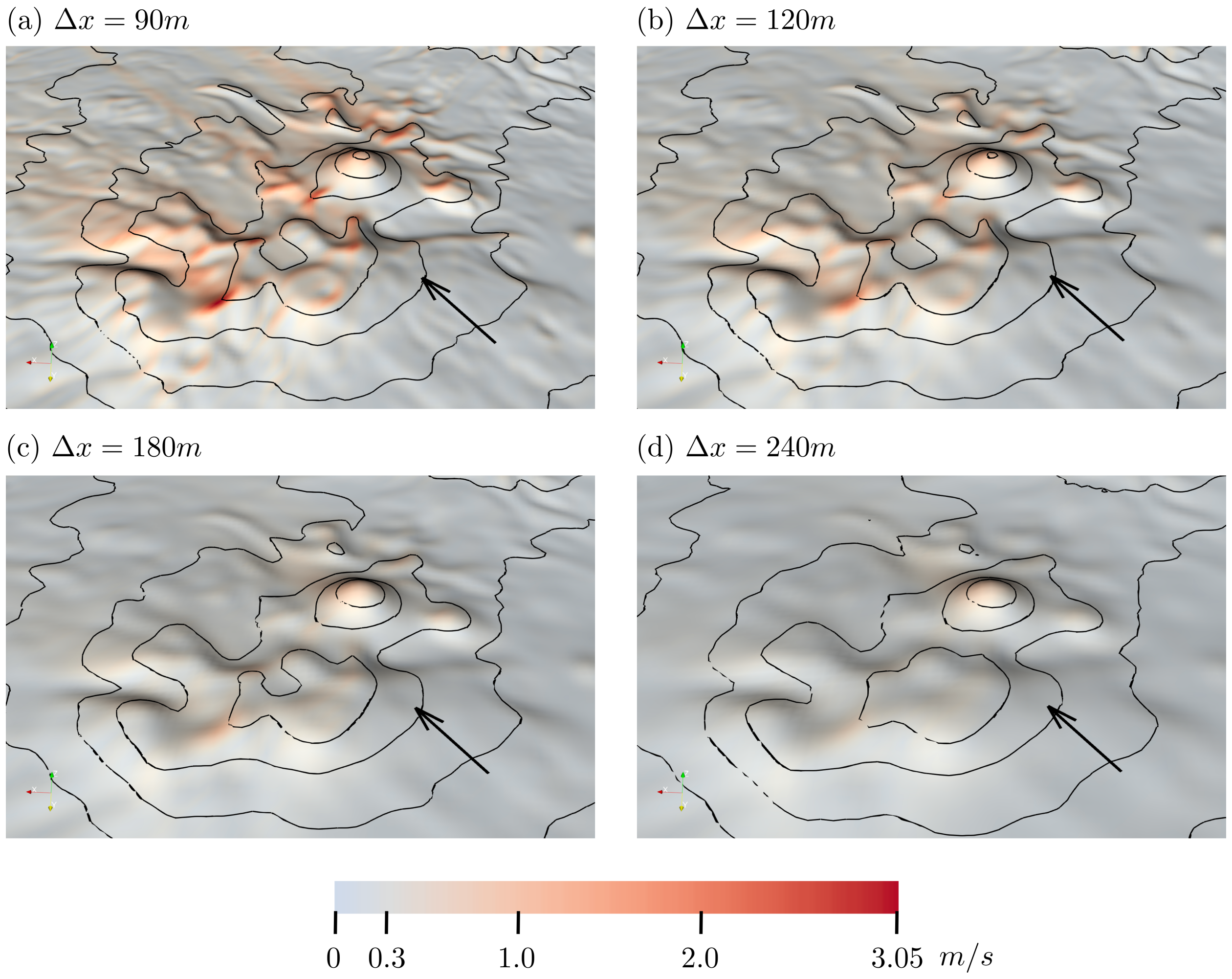


Figure 3.11: Error patterns in different horizontal resolution for the Sep 08th case between full-stress and normal-gradient conditions: (a) 90m; (b) 120m; (c) 180m; (d) 240m.

strongly dependent on the model grid spacing, with the terrain features generally becoming broader and smoother as the grid interval increases. As should be expected, this change in terrain geometry also implies a change in the errors associated with the flat boundary condition, with the approximation becoming broadly more accurate as the terrain becomes smoother. (For consistency, the blue-red transition in all panels occurs at the same value—namely, at 10% of the maximum error observed in Fig. 3.10a.) Similar results can be seen for the normal gradient condition (Fig. 3.11), for which the maximum errors in the $\Delta x = 240$ m case only barely exceed the blue-red threshold. Note that in both the normal gradient and flat boundary experiments, the decrease in the error with

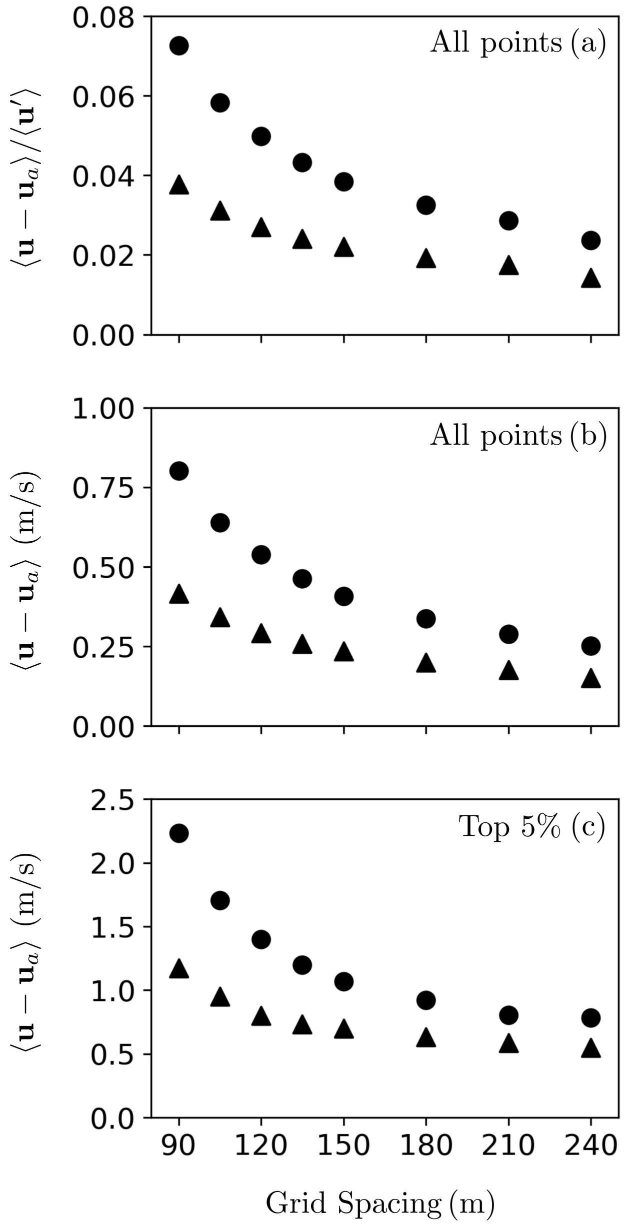


Figure 3.12: Error dependence on resolution in the Sep 08 case. Circles: errors produced by flat boundary approximation; triangles: errors produced by normal gradient approximation.

increasing Δx occurs most prominently in the regions of complex terrain surrounding Mt. Ngauruhoe, rather than over the mountain peak itself.

Figure 3.12 shows the errors for the September 08 case as averaged over the subdomain shown in Fig. 3.1, for grid spacings ranging from 90 m to 240 m. Figure 3.12a shows the errors as normalized by the RMS disturbance wind, as in Figs. 3.4a and b, while Fig. 3.12b shows the errors in dimensional form, as in Figs. 3.4c and d. In all cases, the error is seen to have a strong dependence on the grid spacing, particularly for grid intervals less than roughly $\Delta x = 150$ m. All told, the error in the flat boundary experiments changes by roughly a factor of three (from 0.8 m/s to 0.25 m/s, or from 0.073 to 0.024 in normalized terms) between $\Delta x = 90$ m and 240 m, with roughly two thirds of that difference occurring for grid intervals between 90 m and 150 m. Similar results apply for the normal gradient approximation, with the error decreasing from roughly 0.41 m/s at $\Delta x = 90$ m to roughly 0.15 m/s by $\Delta x = 240$ m.

Figure 3.12c gives a sense of the resolution dependence for the largest errors on the grid, in terms of the mean over all grid points in the top 5% of errors (as described for Fig. 3.5). As with the mean over all grid points, the errors for the top 5% case show a decrease of roughly a factor of three for the flat boundary experiments (from roughly 2.2 m/s to 0.8 m/s) over the range of grid spacings shown. However, relative to Figs. 3.12a and b, a larger fraction of the change (roughly 82%) occurs at grid spacings less than 150 m, suggesting that the top 5% errors are particularly sensitive to grid spacing at high resolution. Also apparent is that the errors for the normal gradient condition in Fig. 3.12c show somewhat less of a decrease (proportionally) than the errors for the flat boundary approximation, meaning that the two approximations become gradually more similar as the grid spacing increases.

Finally, it is worth noting that for both the flat boundary and normal gradient approximations, the errors in Fig. 3.12show little sign of leveling off as the grid spacing decreases, at least for grid spacing larger than 90 m. We might expect that once the grid spacing decreases sufficiently, the terrain features resolved by still smaller grid spacing would start to have less of impact on the overall flow, at which point the resolution dependence of the errors should begin to saturate. That said, according to Fig. 3.12, if such a saturation does occur, it clearly occurs at a grid interval smaller than 90 m.

3.5.2 An outlier case

Figure 3.13 shows the resolution dependence for one of the two outlier cases discussed in section 3.4.4, specifically, the March 03 case shown in Fig. 3.9a. Comparison to Fig. 3.12 shows that same basic trends from the September 08 case apply to the outlier case as well, only more exaggerated, with the errors decreasing by a factor of roughly five in Figs. 3.13a and b over the range of grid spacings shown. This increased resolution sensitivity would appear to be consistent with the discussion in section 3.4.4 related to the compounding effects of the upstream terrain. Specifically, as the grid interval is increased, the errors decrease not only locally, but also upstream over Mt. Ruapehu, leading to more accurate winds incident on the Mt. Ngauruhoe region. This improvement in the incident winds then tends to compound the improvement in the local errors, leading to a bigger change in the errors measured locally.

3.5.3 Dependence on terrain smoothing

When using high-resolution datasets, the terrain data as loaded into the model will inevitably have significant spatial variations on length scales close to the model grid scale, which in turn leads to poorly resolved numerical disturbances. To avoid this problem, the raw terrain data is typically smoothed as part of the model initialization phase, with the intention of removing any problematic, high wavenumber components. However, as should probably be expected, this smoothing of the terrain invariably impacts the terrain geometry, which in turn affects the behavior of the surface boundary condition.

Table 3.1 shows the low-level vector wind errors for a series of experiments in which the model terrain has been subjected to various types and degrees of smoothing. Shown in the table are the normalized, dimensional, and top 5% rms errors (defined as in Fig. 3.4and Fig. 3.5) for the flat boundary approximation, with the September 8 case chosen as the specific example case (cf. Fig. 3.6c and Fig. 3.7c). The smoothing types considered include a simple second-order (1

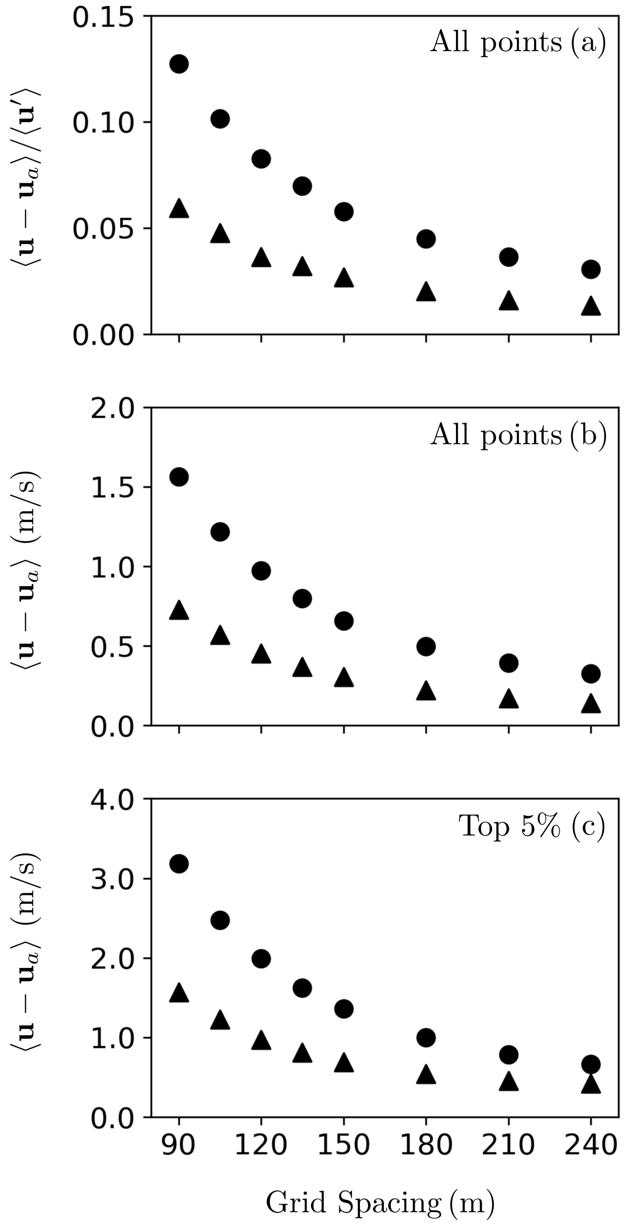


Figure 3.13: Error dependence on the resolution in Mar 03 case

Table 3.1: Errors in low-level winds for simulations using the flat boundary condition with different filters

Filter	number of passes	Normalized error	Error in m/s	Top 5% error in m/s
2^{nd} order	2	0.07267	0.8018	2.2335
4^{th} order	7	0.09201	1.0502	3.2468
4^{th} order	6	0.09697	1.1137	3.4779
4^{th} order	5	0.10315	1.1934	3.7832
4^{th} order	4	0.11066	1.2922	4.1590
4^{th} order	3	0.12071	1.4262	4.6885

Table 3.2: Errors in low-level winds for simulations using the normal gradient condition with different filters

Filter	number of passes	Normalized error	Error in m/s	Top 5% error in m/s
2^{nd} order	2	0.03775	0.4164	1.1741
4^{th} order	7	0.04568	0.5214	1.6173
4^{th} order	6	0.04795	0.5507	1.7505
4^{th} order	5	0.05046	0.5838	1.8829
4^{th} order	4	0.05386	0.6289	2.0935
4^{th} order	3	0.05780	0.6829	2.3222

-2 1) diffusive smoother, along with a more scale-selective fourth-order (-1 4 -6 4 -1) diffusive smoother, applied using varying numbers of passes (e.g., Durran, 1999, sec. 2.4.3). In each case, the smoother is applied in both the x and y directions, with the amplitude of the smoother set so as to completely damp the $2\Delta x$ and $2\Delta y$ modes within a single pass. For the sake of comparison, note that all results prior to this section were computed using the second-order smoother with two passes.

As can be seen in the table, the errors for the flat boundary simulations show a surprising degree of sensitivity to the type of terrain smoothing applied, with the errors steadily increasing as the degree of smoothing is reduced. The lowest errors are associated with the second order smoother, which is the most diffusive option considered, resulting in the smoothest terrain profiles. By comparison, the fourth order smoother is less diffusive at short wavelengths, leading to terrain features with increased slope and curvature. As a result, switching to the fourth order filter was found to have a significant impact on the errors, even for seven passes of the smoother. Reducing the number of passes leads to even sharper terrain features and larger errors, with the model eventually becoming unstable for terrain with less than three passes. Similar trends were seen for the normal gradient condition (see Table 3.2), but with the errors generally reduced by roughly half.

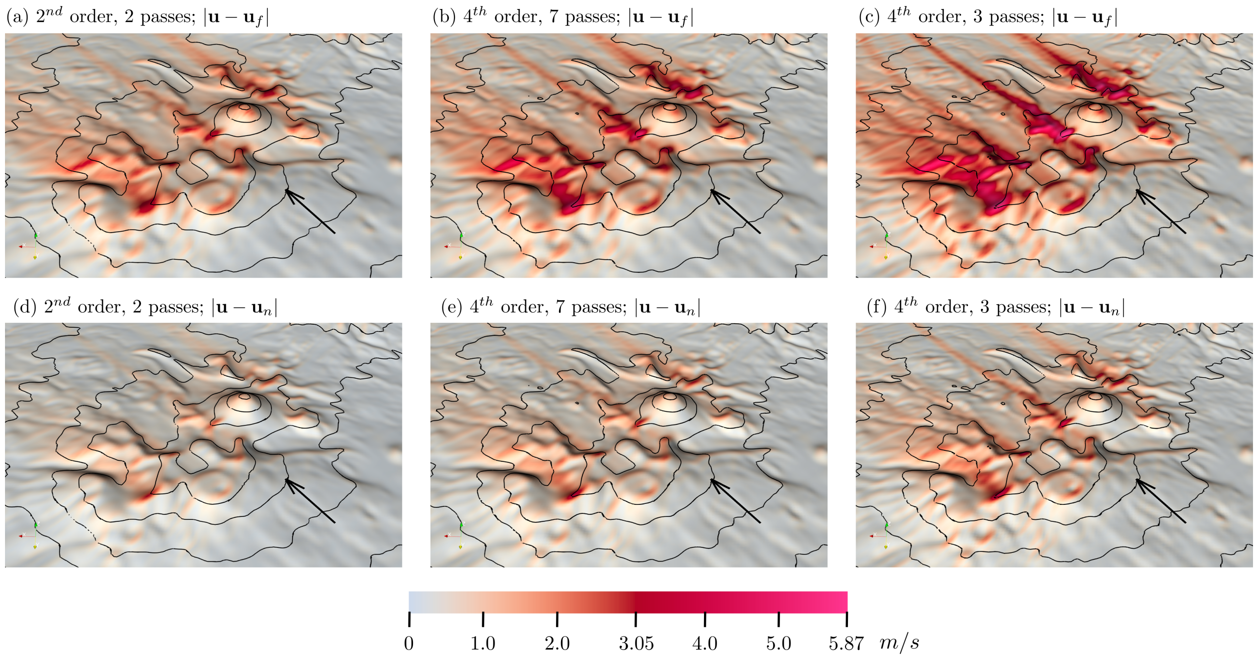

(a) 2^{nd} order, 2 passes; $|\mathbf{u} - \mathbf{u}_f|$
(b) 4^{th} order, 7 passes; $|\mathbf{u} - \mathbf{u}_f|$
(c) 4^{th} order, 3 passes; $|\mathbf{u} - \mathbf{u}_f|$
(d) 2^{nd} order, 2 passes; $|\mathbf{u} - \mathbf{u}_n|$
(e) 4^{th} order, 7 passes; $|\mathbf{u} - \mathbf{u}_n|$
(f) 4^{th} order, 3 passes; $|\mathbf{u} - \mathbf{u}_n|$
0
1.0
2.0
3.05
4.0
5.0
5.87
m/s

Figure 3.14: Error patterns of different filters for the Sep 08th case between full-stress and approximated conditions: (a),(d) 2^{nd} order 2 passes; (b),(e) 4^{th} order 7 passes; (c),(f) 4^{th} order 3 passes. The top three panels show error patterns between full-stress and flat-boundary conditions, while the bottom three show error patterns between full-stress and normal-gradient conditions.

The spatial distributions of the errors for the simulations with varying terrain smoothing are shown in Fig. 3.14 for both the flat boundary and normal gradient approximations. For reference, Figures 3.14a and d are identical to Fig. 3.6c and Fig. 3.7c, showing the Sep 08th cases with the second order filter. To illustrate the errors for the cases with reduced smoothing, the color table in Fig. 3.14 was extended beyond that used previously, to include pinkish colors for larger velocity differences. Inspection of the terrain contours in Fig. 3.14 suggests that the changes in topography

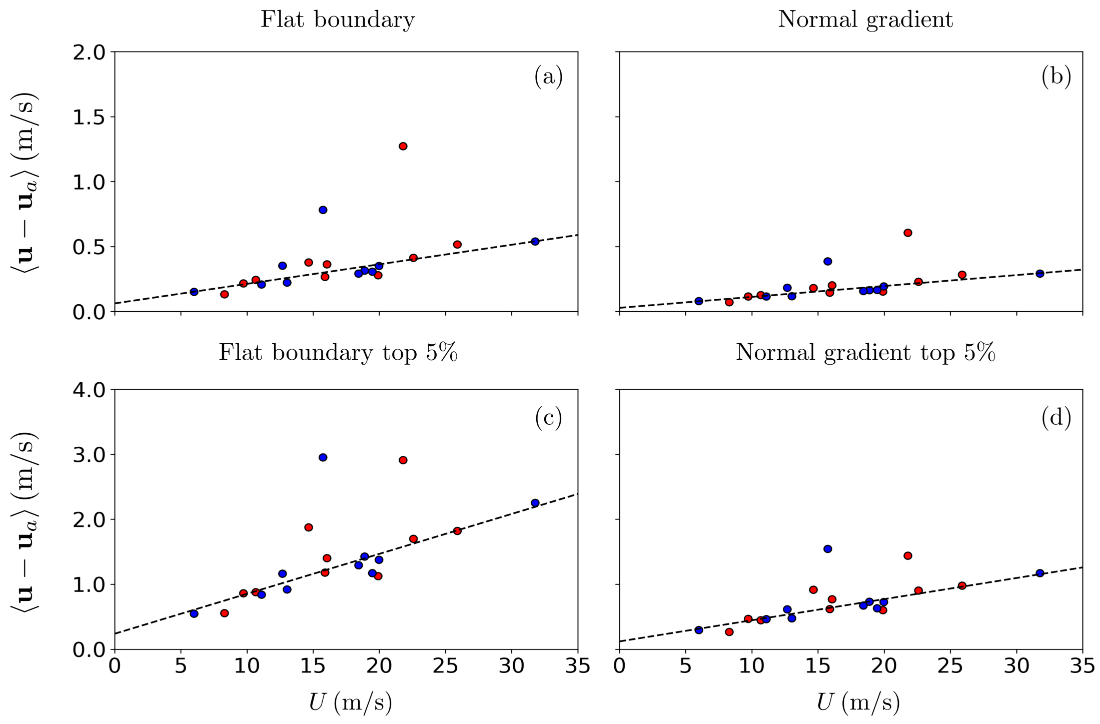


Figure 3.15: Errors in the lower 5 levels winds for simulations using the flat boundary [(a) and (c)] and normal gradient [(b) and (d)] approximations. (a) and (b) show errors in units of m/s of wind speed on the error averaging domain, while (c) and (d) show the 5% averaging errors.

for the different smoothing levels appear to be relatively subtle. Nonetheless, Fig. 3.14a–c show that decreasing the smoothing from the second order / 2 passes case to fourth order / 3 passes leads to a surprising increase in the errors in the flat boundary experiment, with the errors in Fig. 3.14c nearly doubled relative to Fig. 3.14a. The increase is particularly pronounced in regions with high terrain curvature, as well as in the wake of decelerated flow extending downstream. Similar results are seen in Fig. 3.14.d–f for the normal gradient condition.

3.6 Implications for wind energy forecasting

One of the more important applications for high resolution meteorological modeling is the prediction of low-level winds for wind energy forecasting. Broadly speaking, for a given wind turbine, the total wind power available to convert to electricity is set by the flux of wind kinetic energy across the area swept out by the turbine blades, which in turn varies with the cube of the wind speed; that is,

$$P \sim \frac{1}{2}\rho A\, U^3 \tag{3.15}$$

where in this context P is the maximum power available, A is the area swept by the turbine blades, and U is the wind speed across the turbine (keeping in mind that most turbines rotate to face the wind). The cubic dependence on the wind speed suggests that predictions of the maximum power available are strongly sensitive to errors in the predicted wind speed. Specifically, given a small wind speed error δU, the fractional change in the corresponding predicted available power can be approximated by

$$\frac{\delta P}{P} \sim 3\,\frac{\delta U}{U} \tag{3.16}$$

showing that when translated into predicted power, the fractional error in the wind speed is amplified by roughly a factor of three.

Figure 3.15 gives a sense of how the types of errors discussed in previous sections could potentially impact the prediction of the maximum power available for wind generation. Shown in the figure are the mean and top 5% errors corresponding to simulations using the flat boundary approximation for each of the 20 cases considered in section 3.4, plotted as a function of the mean low-level upstream wind speed, as in Figs. 3.4c and 3.5a. However, it should be kept in mind that while previous sections only included data from the first interior grid level (approximately 15 m above ground), the typical total height (i.e., hub height plus rotor radius) of a modern wind turbine can reach as high as 150 m. Thus, to give a better sense of the winds actually incident on the turbine blades, the results in Fig. 3.15 have been averaged all grid levels below 150 m, which corresponds in practice to the lowest five interior grid levels of the model.

As in Figs. 3.4c and 3.5a, the errors in Fig. 3.15 are seen to have a roughly linear dependence on the upstream wind speed, albeit at somewhat reduced magnitudes relative to the results at the lowest grid level. To the extent the error dependence can be treated as linear, the fractional error $\delta U/U$ can be estimated in terms of the slope of the linear regression, which for Fig. 3.15a is evaluated as 0.014. Factoring in the cubic dependence of the power as in (3.16) then gives an estimated fractional error for the available power of 4.5%, as evaluated in terms of the mean over all grid points in the averaging domain. However, it is worth keeping in mind that wind turbines are more likely to be sited in locations with higher wind speeds, which, according to section 3.4, also tend to be locations with larger errors. To give a sense of the dependence at these higher-wind locations, Fig. 3.15b shows the results as averaged over only those grid points in the top 5% of the errors (as in 3.5a), in which case the slope of the linear regression increases to 0.059, implying an error in the available power of roughly 18%. Finally, the results above are all for the flat boundary approximation. As in previous sections, applying the normal gradient condition reduces the errors by roughly half (not shown), suggesting an error in the available power of roughly 10%, as averaged over the points with higher wind speed.

It should be kept in mind the estimates described above are all framed in terms of the maximum available power, as opposed to the actual rate of electricity generation. To address the latter requires consideration of the efficiency at which a given turbine extracts the wind resource, often expressed in terms of the turbine's power curve, which varies from turbine to turbine. Broadly speaking, for the case of weak or modest wind speeds, the energy produced by a given turbine will tend to follow the same cubic dependence on U as the available power, in which case the considerations described above apply without modification. However, above a certain critical wind speed (typically in the range of 12 to 15 m/s), the energy generated by the turbine will tend to saturate, at which point the sensitivity to the wind becomes less of a concern. How this all plays out in a given situation is likely to be complicated, and is left to future studies to explore.

3.7 Summary

The behavior of surface drags on complex terrain surfaces was explored, with a particular emphasis on the way in which the associated drag condition depends on terrain geometry.

From a basic physical standpoint, specifying a surface drag (or more precisely, a surface stress) at the ground in place of the no-slip condition effectively defines an alternative lower boundary condition for the velocity at the terrain surface. Arguments were presented to suggest that this boundary condition is directly tied to the geometry of the terrain, in two respects: (i) a dependence on the slope of the terrain, as expressed through a normal gradient term; and (ii) a dependence on the local terrain curvature, appearing in the boundary condition as a Dirichlet term. The respective roles and impacts of these geometric effects were explored through a series of numerical experiments, in which simulations using the full surface-stress boundary condition [implemented as in Epifanio (2007)] were compared to simulations using one of two widely used approximations: (a) the normal gradient approximation, which accounts for the terrain slope, but ignores the local curvature; and (b) the flat boundary assumption, in which the slope and curvature of the terrain are both neglected.

The experiments were carried out using a region of complex terrain in Tongariro National Park, NZ, as an example problem, as simulated at 90 m horizontal grid spacing. Attention was limited to flows with neutral or nearly neutral surface layer conditions (as at the morning and evening transition periods, for example), but with cases otherwise chosen at random among all the corresponding events in a given year (2012). The experiments were carried out within the perfect model framework, in which flows simulated using the full form of the surface drag / stress condition were considered to be correct. For simulations using the normal gradient and flat boundary conditions, departures from the full stress simulations were treated as model errors, associated with the approximate boundary conditions and their corresponding neglect of the terrain's geometric properties.

For any given case, the error was quantified by taking the magnitude of the vector wind difference between a simulation using the full stress condition and the corresponding simulations using the normal gradient and flat boundary approximations, as evaluated at the lowest grid level (roughly 15 m) and averaged over all grid points in the domain of interest. By this measure, the flat boundary approximation was found to be associated with an error of roughly 0.59 m/s, as averaged over all 20 cases considered. That said, the impact of the boundary condition was found to be highly uneven in space, with errors several times larger than the mean value at particular locations in the flow. Applying the normal gradient condition in place of the flat boundary assumption reduced the errors by roughly half, suggesting that the slope and curvature of the terrain play roughly equal roles in determining the overall behavior of the drag.

The spatial distribution of the errors shows that the impact of the terrain geometry tends to be largest over specific features in the terrain, particularly over small ridge-like features whose long axes are oriented roughly perpendicular to the incident wind direction. When using the flat boundary assumption, the errors were largest both at the peaks of the ridges, where the curvature of the terrain is largest, as well as along the nearby slopes, while for the normal gradient condition, the errors were more closely tied to the locations of strongest curvature. In all cases, the application of the full stress condition lead to greater flow deceleration than for either of the two approximate conditions, ultimately resulting in a wake of decelerated air extending downstream of the corresponding terrain features.

As should be expected, experiments at varying horizontal grid spacing showed that the resolution of the model grid has a significant impact on the geometry of the terrain, which in turn implies an impact on the behavior of the stress condition, as well. At a grid spacing of 240 m, the errors associated with the flat boundary simulations were shown to be relatively modest (0.25 m/s, for a typical example case), suggesting that the effects of terrain slope and curvature at that resolution are relatively small. However, decreasing the grid spacing to 90 m led to a rapid increase in the errors (by roughly a factor of three), with most of the increase occurring for grid spacings less than 150 m. Similar results were found for the normal gradient condition, but with the errors at any given resolution decreased by roughly a factor of two. To the extent our experiments are representative, we thus suggest that in regions of complex terrain, geometric effects are likely to play an important role in the stress condition once the horizontal grid spacing is decreased to something roughly on the order of 100 m or so. Interestingly, the errors in our experiments showed no sign of leveling off as the resolution increased, raising the question as to how the drag condition is likely to behave as the grid spacing is reduced even further.

It is worth noting that, apart from a pair of outlier cases, the errors for both the flat boundary and normal gradient experiments were found to have a roughly linear dependence on the upstream wind speed. Given that our cases in our experiments were chosen at random, this in turn suggests that the scaling of the drag condition with the wind speed is largely independent of factors such as the wind direction or the details of the vertical wind and stability profiles. That said, a close inspection of the two outlier cases suggests this result may not in fact be general. In particular, both of the outlier cases showed significant impacts from upstream terrain features, which ultimately led to a compounding effect on the errors. Further, anisotropic terrain features with a particular axis of elongation, such as an elongated ridge, would be expected to produce greater errors for winds perpendicular to the elongation axis, as suggested to some degree by the Oct 6 outlier event. We thus expect that for many terrain flows, the behavior of the drag condition will have a significant dependence on the upstream wind direction, as well as potentially on other factors, such as the vertical wind and stability profiles.

Finally, one of the more important applications for high-resolution atmospheric modeling is the prediction of low-level winds for wind energy forecasting. Given that the total power available for wind energy depends on the cube of the wind speed, we might expect even modest errors in the winds to be amplified significantly when translated into predictions of available power. Indeed, based on the present results, a typical error associated with use of the flat boundary approximation would be expected to lead to a roughly 20% error in the available power, as evaluated at the locations with largest wind speeds, while the normal gradient condition would be associated with a roughly 10% error. Having said that, these estimates have several important caveats. First, while our experiments were carried out using real terrain data and realistic background states, we nonetheless made use of an idealized model (so as to have access to the full stress condition), meaning that factors such as microphysics, radiation, and detailed land surface properties were neglected. Furthermore, a prediction of the energy actually generated by a turbine (as opposed to the total available energy) would require consideration of the particular turbine's power curve, which was beyond the scope of the present study. Thus, while at a basic physical level, the role of the terrain geometry in the drag condition would appear to be significant, an assessment of the impact under more realistic flow conditions is left for future work.

4. WRF IMPLEMENTATION

4.1 Introduction

As described in previous chapters, a method for implementing the general surface drag condition in the context of finite-difference models was introduced by E07. However, the method of E07 is based on discretizing the boundary condition directly in the form (2.12), which ultimately results in a global sparse matrix problem to be solved simultaneously over the whole domain. As described below, this global nature of the method complicates its implementation in highly parallelized models, such as the Weather Research and Forecasting (WRF) model. The goal of the present chapter is develop a new, alternative method for implementing the drag condition, more suitable for use in models such as WRF.

As with many highly parallelized models, the WRF model achieves its high degree of scalability by means of a domain decomposition strategy, in which the full model domain is decomposed into smaller subdomains, called tiles. These subdomains are distributed across different computational nodes, with each tile assigned to an individual processor on a given node. To a certain degree, a given processor/tile is able to share information with its neighbor tiles, particularly at specific points during the time integration, such as after the completion of a model time step. But to maintain computational efficiency, the communication between tiles must be limited—that is, the computations carried out by a given processor must be largely limited to the information on the associated tile.

From a software design standpoint, the decomposition of the WRF domain is carried out by means of three different software layers, each of which has a different (virtual) window into the information distributed across the tiles. Of the three layers, the most relevant for the present discussion is the so-called model layer, where the actual numerical computations are implemented. By design, the model layer is constrained to be tile callable, meaning that the computations in this layer only have access to the information available on a single grid tile, which in turn allows the computations to be efficiently distributed across processors. Model processes that require a wider view into the data (communication between tiles, I/O, etc.) are relegated to the higher, more abstract layers of the model (namely, the driver and mediation layers).

In most respects, constraining the model layer to be tile callable tends to simplify the code, in that the numerical computations can be implemented as if the model were serial, without worrying about things like message passing, etc. However, in our particular case, this restriction presents a significant obstacle, in that the implementation of the stress condition described by Epifanio (2007) is by nature a global calculation, requiring information across the entire model domain at once. Indeed, after looking through the WRF model code, our conclusion is that the matrix method described in section 2.2.3 likely cannot be implemented in any straightforward way in WRF, at least not in a manner that respects the parallelization and efficiency of the model.

The following section describes a reformulation of the surface stress boundary condition into a form allowing the condition to be implemented locally, without the need for a global matrix solver. It is shown that the condition reduces to the sum of two terms, analogous to the terms in the 2D form of the condition described in section 3.3.3. Section 4.3 describes the basic algorithm and numerics of the method, as developed for implementation the WRF model, including an overview of associated changes to the WRF code. Section 4.4.1 presents verifications and tests of the method involving comparisons to simulations using the model of E07, as implemented using the matrix method. Some details of the default boundary condition in the standard WRF model are addressed, as well as the performance of the new method at varying grid resolution. Finally, a summary and discussion of the results is presented in section 4.5.

4.2 The stress condition in local form

The boundary condition in the form (2.12) is complicated by the presence of horizontal derivatives along the boundary, which couple nearby grid points. This coupling can be avoided by reformulating the drag conditions into a form involving the normal derivative. Specifically, by comparing (2.12a) and (2.12b) with the normal gradient condition (3.8), the full stress conditions can be rewritten in the form

$$\frac{\partial}{\partial n}(un_3 - wn_1) + n_1 n_3 \frac{\partial u}{\partial X} + n_2 n_3 \frac{\partial v}{\partial X} + n_3 n_3 \frac{\partial w}{\partial X} = -\frac{D_x}{\kappa}\sqrt{n_1^2 + n_3^2} \tag{4.1a}$$

$$\frac{\partial}{\partial n}(vn_3 - wn_2) + n_2 n_3 \frac{\partial v}{\partial Y} + n_1 n_3 \frac{\partial u}{\partial Y} + n_3 n_3 \frac{\partial w}{\partial Y} = -\frac{D_y}{\kappa}\sqrt{n_2^2 + n_3^2}\,. \tag{4.1b}$$

Given the nature of the normal gradient approximation, the terms added to the normal gradients in (4.1) presumably depend on terrain curvature. Note that by combining with (2.12c), these terms can be rewritten as

$$\begin{aligned} n_1 n_3 \frac{\partial u}{\partial X} + n_2 n_3 \frac{\partial v}{\partial X} + n_3 n_3 \frac{\partial w}{\partial X} &= \left(n_1 n_3 + n_3 n_3 \frac{\partial h}{\partial x}\right)\frac{\partial u}{\partial X} \\ &\quad + n_3 n_3 \frac{\partial^2 h}{\partial x^2} u + \left(n_2 n_3 + n_3 n_3 \frac{\partial h}{\partial y}\right)\frac{\partial v}{\partial X} + n_3 n_3 \frac{\partial^2 h}{\partial x \partial y} v \end{aligned} \tag{4.2a}$$

$$\begin{aligned} n_2 n_3 \frac{\partial v}{\partial Y} + n_1 n_3 \frac{\partial u}{\partial Y} + n_3 n_3 \frac{\partial w}{\partial Y} &= \left(n_2 n_3 + n_3 n_3 \frac{\partial h}{\partial y}\right)\frac{\partial v}{\partial Y} \\ &\quad + n_3 n_3 \frac{\partial^2 h}{\partial y^2} v + \left(n_1 n_3 + n_3 n_3 \frac{\partial h}{\partial x}\right)\frac{\partial u}{\partial Y} + n_3 n_3 \frac{\partial^2 h}{\partial x \partial y} u\,. \end{aligned} \tag{4.2b}$$

But from the definition of the normal vector (2.1), we have

$$n_3 \frac{\partial h}{\partial x} = -n_1 \qquad , \qquad n_3 \frac{\partial h}{\partial y} = -n_2$$

implying that the terms in parentheses in (4.2) are zero. Substituting (4.2) back into (4.1) then allows the full stress condition to be rewritten as

$$\frac{\partial}{\partial n}(un_3 - wn_1) + n_3^2\, \mathbf{u} \cdot \boldsymbol{\nabla} \frac{\partial h}{\partial x} = -\frac{D_x}{\kappa}\sqrt{n_1^2 + n_3^2} \tag{4.3a}$$

$$\frac{\partial}{\partial n}(vn_3 - wn_2) + n_3^2\, \mathbf{u} \cdot \boldsymbol{\nabla} \frac{\partial h}{\partial y} = -\frac{D_y}{\kappa}\sqrt{n_2^2 + n_3^2} \tag{4.3b}$$

or equivalently

$$\frac{\partial u_s}{\partial n} + \frac{n_3^2}{\sqrt{n_1^2+n_3^2}}\,\mathbf{u}\cdot\boldsymbol{\nabla}\frac{\partial h}{\partial x} = -\frac{D_x}{\kappa} \tag{4.4a}$$

$$\frac{\partial u_t}{\partial n} + \frac{n_3^2}{\sqrt{n_2^2+n_3^2}}\,\mathbf{u}\cdot\boldsymbol{\nabla}\frac{\partial h}{\partial y} = -\frac{D_y}{\kappa} \tag{4.4b}$$

where as before $u_s = \mathbf{u}\cdot\mathbf{s}$ and $u_t = \mathbf{u}\cdot\mathbf{t}$.

The first terms in (4.4a) and (4.4b) are the normal gradient terms from (2.22). The second terms involve rates of change of $\partial h/\partial x$ and $\partial h/\partial y$ along a parcel trajectory, which are related to the curvature of the terrain in the xz and yz planes as seen by a moving parcel, respectively. Note these terms involve only the components of $\mathbf{u}$ and not their derivatives, suggesting these terms are effectively Dirichlet terms. As shown below, this allows (4.4) to be implemented as a local condition, without the need for a matrix inversion.

4.3 Numerical implementation

4.3.1 3D local full condition

To compute the normal derivatives in (4.4), we consider the intersection of line normal to the terrain with the first interior model coordinate surface. To specific, suppose for a given boundary point, the velocity components at the boundary are represented by (u_0, v_0, w_0), while the velocity interpolated to the normal intersection point is denoted $(\hat{u}, \hat{v}, \hat{w})$. In the WRF model, the velocity components are staggered vertically, so that $\hat{u}$ and $\hat{v}$ are defined at the first interior half-level, while $\hat{w}$ is defined at the first full level. With that in mind, the stress condition (4.3) can be discretized as

$$n_3\frac{\hat{u}-u_0}{\Delta s} - n_1\frac{\hat{w}-w_0}{\Delta s_f} + n_3^2\,\frac{\partial^2 h}{\partial x^2}\,u_0 + n_3^2\,\frac{\partial^2 h}{\partial x\partial y}\,v_0 = -\frac{D_x}{\kappa}\sqrt{n_1^2+n_3^2} \tag{4.5a}$$

$$n_3\frac{\hat{v}-v_0}{\Delta s} - n_2\frac{\hat{w}-w_0}{\Delta s_f} + n_3^2\,\frac{\partial^2 h}{\partial x\partial y}\,u_0 + n_3^2\,\frac{\partial^2 h}{\partial y^2}\,v_0 = -\frac{D_y}{\kappa}\sqrt{n_2^2+n_3^2} \tag{4.5b}$$

where Δs is the distance to the intersection with the first half level, while Δs_f is the distance to the first full level.

Replacing w_0 in favor of u_0 and v_0 using (2.12c) converts (4.5) to

$$\begin{aligned}\left(n_1\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial x} - n_3 + n_3^2\frac{\partial^2 h}{\partial x^2}\Delta s\right)u_0 + \left(n_1\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial y} + n_3^2\frac{\partial^2 h}{\partial x\partial y}\Delta s\right)v_0 \\ = -\frac{D_x\sqrt{n_1^2+n_3^2}\,\Delta s}{\kappa} - n_3\hat{u} + n_1\frac{\Delta s}{\Delta s_f}\hat{w}\end{aligned} \tag{4.6a}$$

$$\begin{aligned}\left(n_2\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial x} + n_3^2\frac{\partial^2 h}{\partial x\partial y}\Delta s\right)u_0 + \left(n_2\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial y} - n_3 + n_3^2\frac{\partial^2 h}{\partial y^2}\Delta s\right)v_0 \\ = -\frac{D_y\sqrt{n_2^2+n_3^2}\,\Delta s}{\kappa} - n_3\hat{v} + n_2\frac{\Delta s}{\Delta s_f}\hat{w}\end{aligned} \tag{4.6b}$$

which is a linear system for u_0 and v_0 at the boundary. Denoting the coefficients in (4.6) as

$$\begin{aligned}A_1 &= n_1\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial x} - n_3 + n_3^2\frac{\partial^2 h}{\partial x^2}\Delta s \\ A_2 &= n_2\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial x} + n_3^2\frac{\partial^2 h}{\partial x\partial y}\Delta s \\ A_3 &= n_1\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial y} + n_3^2\frac{\partial^2 h}{\partial x\partial y}\Delta s \\ A_4 &= n_2\frac{\Delta s}{\Delta s_f}\frac{\partial h}{\partial y} - n_3 + n_3^2\frac{\partial^2 h}{\partial y^2}\Delta s \\ A_5 &= -\frac{D_x\sqrt{n_1^2+n_3^2}\,\Delta s}{\kappa} - n_3\hat{u} + n_1\frac{\Delta s}{\Delta s_f}\hat{w} \\ A_6 &= -\frac{D_y\sqrt{n_2^2+n_3^2}\,\Delta s}{\kappa} - n_3\hat{v} + n_2\frac{\Delta s}{\Delta s_f}\hat{w}\end{aligned} \tag{4.7}$$

the solution is given as

$$u_0 = \frac{A_4A_5 + A_3A_6}{A_1A_4 - A_2A_3} \quad , \quad v_0 = \frac{A_1A_6 + A_2A_5}{A_1A_4 - A_2A_3} \tag{4.8}$$

At any given time step, the coefficients A_1 through A_6 are known, so that (4.8) gives the velocity components u_0 and v_0 at the boundary. The vertical component w_0 is then recovered from (2.12c).

4.3.2 The normal gradient and flat boundary approximations

As described in section 4.2, the normal gradient approximation can be recovered from the full stress condition by simply setting the Dirichlet curvature terms in (4.3) to zero. In practice, this translates into setting the terms involving second derivatives in h to zero in (4.7).

A version of the flat boundary approximation was implemented using (2.19), which gives the boundary u_0 and v_0 fields simply as

$$u_0 = \frac{D_x}{\kappa}\Delta s + u_1 \qquad , \qquad v_0 = \frac{D_y}{\kappa}\Delta s + v_1$$

where in this case, u_1 and v_1 refer to values at the first interior half level directly above the point of interest. As before, w_0 is then recovered from (2.12c).

4.3.3 Finding the intersection point

As described above, our method for computing the normal derivatives in (4.3) relies on interpolating the wind fields to the intersection of the line normal to the terrain with the first interior coordinate surface. To find the intersection point, suppose the location of the point of interest on the boundary is given by (x_0, y_0, z_0).

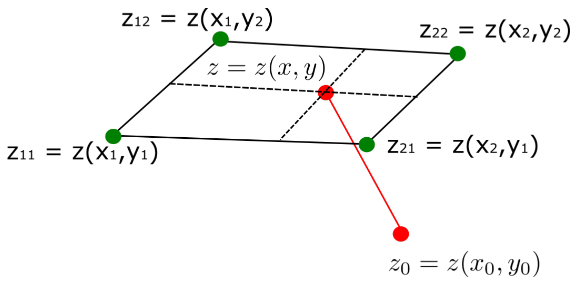


Figure 4.1: The intersection of the terrain normal with the first interior coordinate surface

Then noting that points along the normal direction are perpendicular to $\mathbf{s}$ and $\mathbf{t}$, as defined in (2.4), the coordinates of points along the line must satisfy

$$n_3(x - x_0) = n_1(z - z_0) \qquad \text{and} \qquad n_3(y - y_0) = n_2(z - z_0)\ . \tag{4.9}$$

In the WRF model, the height of the first interior coordinate surface is defined in terms of an interpolation of the geopotential field at the surrounding grid points. Suppose the line normal to the terrain intersects the coordinate surface in a grid box defined by four surrounding geopotential points, as illustrated in Fig. 4.1. Given the horizontal grid spacing and map factors, the horizontal positions of the surrounding points can be determined from the latitude and longitude coordinates. Suppose the positions of the surrounding points are given by (x_1,y_1), (x_1,y_2), (x_2,y_1), (x_2,y_2), and let the associated heights at these points be given by z_{11}, z_{12}, z_{21}, z_{22}, as shown in Fig. 4.1. The height of the coordinate surface in the grid box can then be defined through bilinear interpolation as

$$z(x, y) = \frac{\sum z_{ij}(-1)^{i-j}(x - x_i)(y - y_i)}{(x_2 - x_1)(y_2 - y_1)} \tag{4.10}$$

where the sum is over the i and j coordinates for the surrounding points.

Given the height of the coordinate surface and the line normal to the surface, the intersection is found by replacing x and y in (4.10) in terms of z using (4.9), implying that the resulting point will be both on the coordinate surface and along the normal line. The result can be written in the form

$$Az^2 + Bz + C = 0 \tag{4.11}$$

where

$$\begin{aligned} A &= n_1 * n_2 * (z_{11} + z_{22} - z_{12} - z21) \\ B &= ((z_0 - y_1) * n_1 + (z_0 - x1) * n_2) * z_{22} + ((y_1 - z_0) * n_1 + (x_2 - z_0) * n_2) * z_{12} + \\ &\quad ((y_2 - z_0) * n_1 + (x_1 - z_0) * n_2) * z_{21} + ((z_0 - y_2) * n_1 + (z_0 - x_2) * n_2) * z_{11} - \\ &\quad (y_2 - y_1) * (x_2 - x_1) \\ C &= (z_0 - x_1) * (z_0 - y_1) * z_{22} + (x_2 - z_0) * (z_0 - y_1) * z_{12} + \\ &\quad (z_0 - x_1) * (y_2 - z_0) * z_{21} + (x_2 - z_0) * (y_2 - z_0) * z_{11} \end{aligned} \tag{4.12}$$

Of the two solutions to (4.11), only one will be valid, with the other predicting a z value either higher than the highest point of the surrounding points or lower than the lowest of them.

The solution to (4.11) gives the height of the desired intersection point, with the x and y coordinates then recovered from (4.9). Given the coordinates of the point, the wind fields can be interpolated to give $\hat{u}$, $\hat{v}$ and $\hat{w}$ in (4.5).

4.3.4 Implementation in the WRF code

The implementation of the stress condition involves two steps:

1. The full stress and no-penetration conditions are applied to solve for the velocity components at the boundary, as described in previous sections.

2. The surface fields are used in the computation of the deformations and stresses at the ground, which then ultimately feed into the viscosity / turbulent mixing parameterization at the boundary.

A map of the parts of the WRF model code relevant to this implementation is shown in Figure 4.2, with parts of the code requiring modification colored red. Note that apart from minor modifications like subroutine calls, all the changes to the code are limited to the diffusion module (*module_diffusion_em*). The code that needs to be added / modified to carry out the implementation includes:

- *Registry:* We added arrays to hold the values of the surface wind fields, as well as deformation values at the ground. And we also made sure these surface arrays are included in the halo updates between tiles.

  To be specific, in the default (public) version of WRF (as well as most atmospheric models), there are no ground level values for τ_{11}, τ_{12}, τ_{22}, and τ_{33}, because they are defined on half levels. We thus added four new deformation fields (defor11, defor12, defor22, defor33) at the ground, to hold the values computed from the surface fields.

  As with the deformation terms described above, WRF also does not include surface values for the u and v components of the wind, which are again at half levels. We thus added arrays u_sfc_u, v_sfc_u, w_sfc_u, u_sfc_v, v_sfc_v, w_sfc_v to hold the velocity components at the ground provided by the stress condition. Note that in the method described above, all three components of the velocity are obtained at each surface grid point. To be specific, u_sfc_u, v_sfc_u, w_sfc_u are the velocity fields defined directly under u points at the boundary, while u_sfc_v, v_sfc_v, w_sfc_v are the components defined directly under v points.

All new arrays are added to Registry.COMMON.

- *get_norm_intersection (new routine):* Code to implement the solution for the normal intersection points for both the full level and half level, as described in section 4.3.3, as well as the interpolation of the velocity components to the intersection points.

- *get_surf_winds (new routine):* Code to implement the solution for the surface wind fields using either the full stress boundary condition or the normal gradient / flat boundary approximations, as described in sections 4.3.1 and sections 4.3.2. Note that this step makes use of the surface drags [i.e., D_x and D_y in (2.3)] generated by the surface layer module (and passed in to the diffusion routines as a friction velocity).

- *cal_deform_and_div:* The next step is to modify the computation of the deformations (which later become stresses) at the ground, so as to make use of the surface velocities determined previously. Note that we make use of the new deformation arrays described above, since some deformations do not exist in the default WRF model at the surface. This guarantees the stresses at the ground are consistent with the imposed stress condition.

- *horizontal_diffusion_2 and vertical_diffusion_2:* The final step is to use the modified deformations at the ground to define the associated stresses. The new stresses are then ultimately used in the computation of the stress convergence in the subgrid-scale turbulence parameterization, thus determining the associated time tendency.

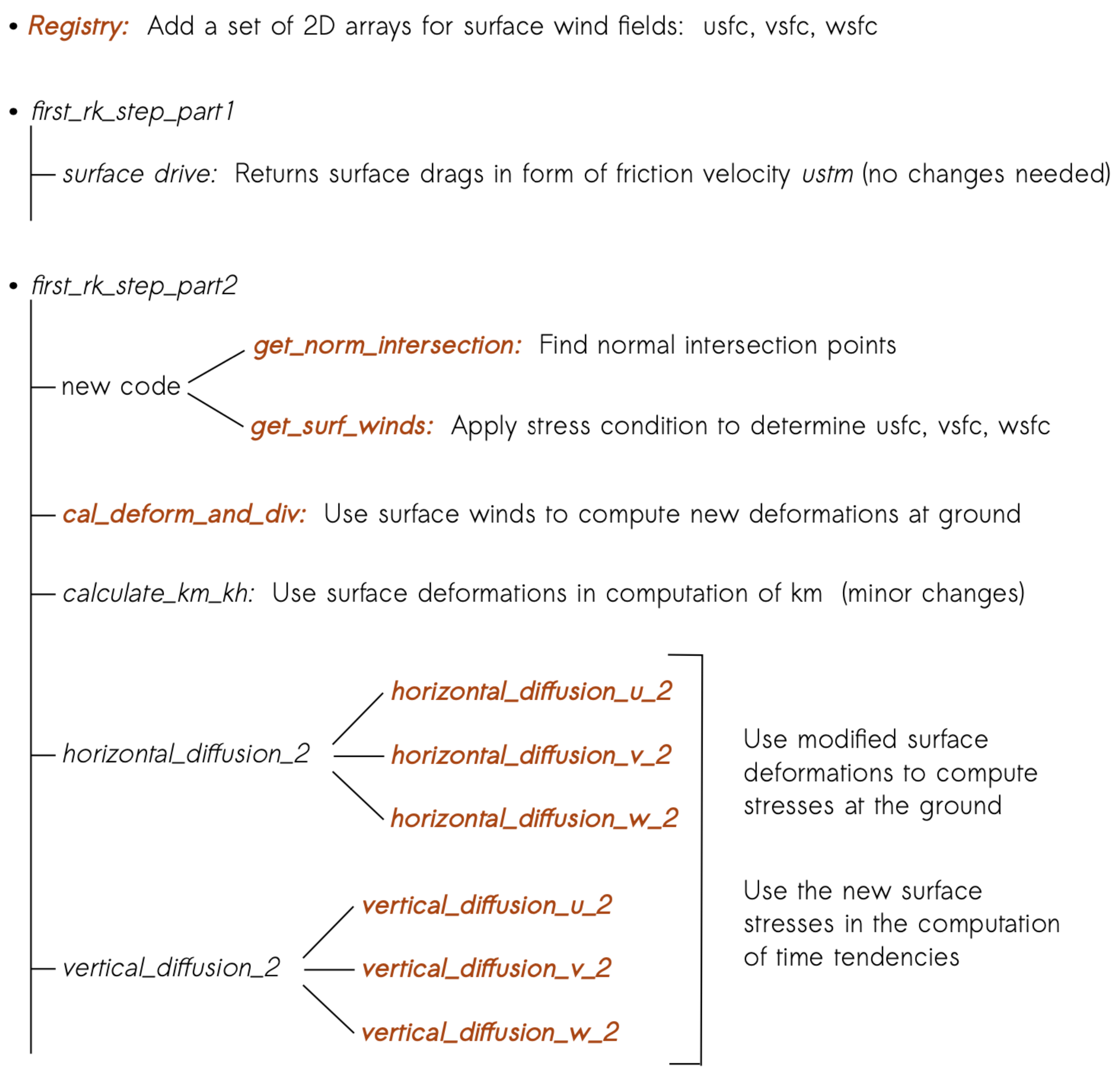

• Registry: Add a set of 2D arrays for surface wind fields: usfc, vsfc, wsfc
• first_rk_step_part1
surface drive: Returns surface drags in form of friction velocity ustm (no changes needed)
• first_rk_step_part2
new code
get_norm_intersection: Find normal intersection points
get_surf_winds: Apply stress condition to determine usfc, vsfc, wsfc
cal_deform_and_div: Use surface winds to compute new deformations at ground
calculate_km_kh: Use surface deformations in computation of km (minor changes)
horizontal_diffusion_2
horizontal_diffusion_u_2
horizontal_diffusion_v_2
horizontal_diffusion_w_2
Use modified surface deformations to compute stresses at the ground
vertical_diffusion_2
vertical_diffusion_u_2
vertical_diffusion_v_2
vertical_diffusion_w_2
Use the new surface stresses in the computation of time tendencies

Figure 4.2: Code map showing parts of the WRF model code relevant to the surface stress implementation.
Parts of the code requiring changes (beyond simple subroutine calls, etc) are colored reddish.

The changes in horizontal_diffusion_2 and vertical_diffusion_2 are somewhat subtle, but the main idea is the updated stresses can be used to determine the associated time tendencies for the winds. The remaining parts of the changed WRF code are attached in Appendix A.

4.4 Idealized tests

4.4.1 Verification

In principle, the 3D local conditions described in section 4.2 should give results equivalent to the 3D matrix full stress / normal gradient conditions introduced in Chapter 3. Verifications were thus carried by comparison to the model of E07, which implements the boundary conditions using the matrix method. The test problems considered consist of 2D and 3D flows past an isolated obstacle. Attention was limited to neutral flows with free-slip ($D_x = D_y = 0$) boundaries, so as to highlight the impact of the boundary condition itself, as opposed to the specified drag or interior dynamics.

4.4.1.1 2D verifications

For the 2D verifications, the terrain profile is the ridge with linearly sloping sides defined by E07, as specified by

$$h(x) = h_0 \times \begin{cases} \frac{1}{8}[1 + cos(\frac{\pi x}{2L_0})^2] + \frac{3\sqrt{3}\pi}{32}, & \text{if } |x| < \frac{2L_0}{3}; \\ \frac{9}{32} + \frac{3\sqrt{3}\pi}{32}(\frac{5}{3} - \frac{|x|}{L_0}), & \text{if } \frac{2L_0}{3} \leq |x| < \frac{5L_0}{3}; \\ \frac{1}{8}\left\{1 + cos[\frac{\pi x}{2L_0}]\right\}^2, & \text{if } \frac{5L_0}{3} \leq |x| < 3L_0; \\ 0, & \text{otherwise.} \end{cases} \tag{4.13}$$

where h_0 is the height of mountain, L_0 is the half width. The upstream wind velocity is constant at $U = 10$ m/s, and a rigid upper lid is at the top of the model domain at height D_0. The governing parameters are (twice of the) max slope $\delta_0 = h_0/L_0 \approx 2$, Reynolds number $Re = UL/\kappa = 50$, and $D_0/L_0 = 5 + \delta_0$. To be specific, the mountain height is set to be 1000 m, so that the rigid lid is at the height of 3500 m; $L_0 = 500$ m which is half width of the mountain. In the following cases, the simulation domain in the x direction is $-18L_0 \leq x \leq 22L_0$. Horizontal and vertical grid spacing are given by $\Delta x/L_0 = 1/40$ and $\Delta z/L_0 = 1/60$.

Fig. 4.3 shows a set of simulations with three different stress conditions in two different models. The top three panels show horizontal velocity fields in the semi-idealized model using the

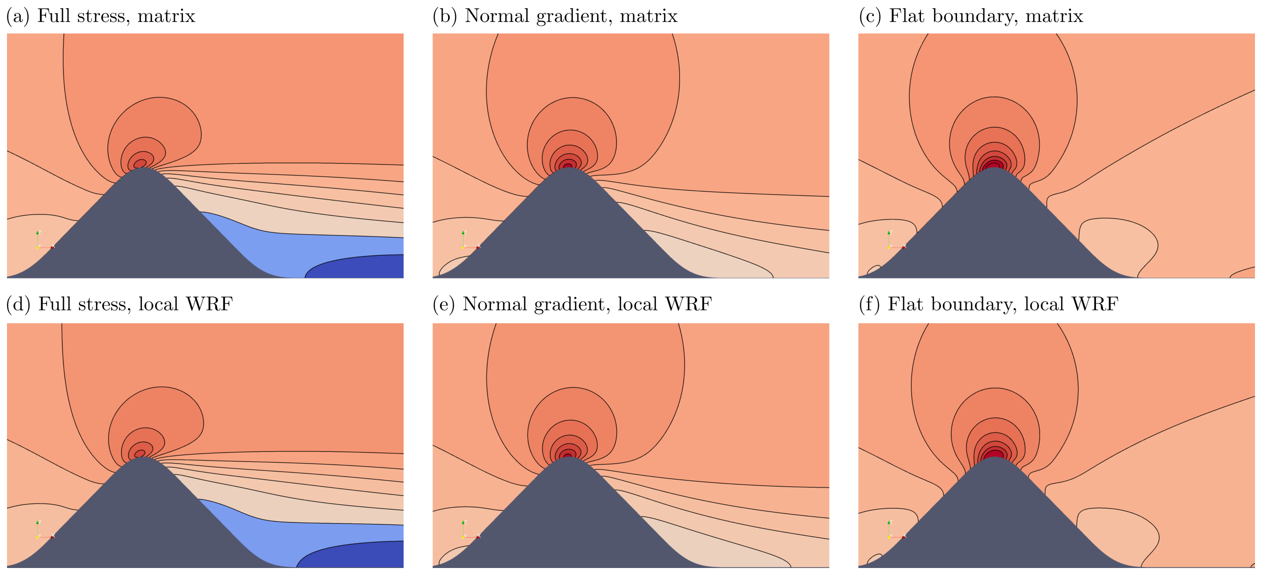


Figure 4.3: Numerical simulations for free-slip flow with $Re = 50$ past a 2D ridge at time $Ut/L = 24$. (a), (d) The semi-idealized model with the matrix full-stress condition and the WRF model with the local full-stress condition; (b), (e) The semi-idealized model with the matrix normal-gradient condition and the WRF model with the local normal-gradient condition; and (c), (f) The semi-idealized model with the flat-boundary condition and the WRF model with the flat-boundary condition. All the plots show x-component velocity (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity).

matrix method. Comparison of the different boundary conditions shows the full-stress condition (Fig. 4.3a) decelerated the wind most on the leeward side of the mountain, and it was the only case that could produce a reverse wind ($u < 0$) zone. In the normal gradient condition case (Fig. 4.3b), the obstacle decelerated the wind downstream, but obviously less than the deceleration in the full stress case. The reason might be found in sections 2.1.3: according to (2.16), the curvature term $2\mathbf{u} \cdot \mathbf{s}/R$ is negative at the top of the mountain, so that the convex terrain was expected to lead to positive vorticity $\eta > 0$ at the top of the mountain. As a result, this positive vorticity will be brought to downstream to create a negative horizontal velocity zone on the leeward side.

In Fig. 4.3.c, the flat boundary condition was adopted and the deceleration is even more moderate than the normal gradient case. The difference between the two approximate methods is described in sections 2.1.4: the normal-gradient condition ignores the curvature effect, while the flat-boundary condition ignores both the curvature and the slope effects. Note that similar effects of applying the different stress conditions were found in the Ngauruhoe test case in section 3.4 (e.g., Fig. 3.6 and Fig. 3.7).

Fig. 4.3d-f show the corresponding local conditions implemented in WRF. The very similar patterns shown by Fig. 4.3a and d, Fig. 4.3b and e, and Fig. 4.3c and f confirm that the matrix conditions and their matching local conditions produce nearly identical flows, at least for these high-resolution tests.

4.4.1.2 3D verifications

To further verify the 3D local stress conditions, another test of 3D free-slip flow past a mountain was done. The mountain in this 3D case is given by

$$h(x) = \begin{cases} \frac{h_0}{16}[1 + cos(\frac{\pi r}{4L_0})]^4; & \\ 0, & \text{otherwise.} \end{cases} \tag{4.14}$$

where r is the radius and L_0 is again the half-width of the mountain. As in the 3D case, D_0 is the model domain vertical depth; the max (twice of the) slope of the mountain is $\delta_0 = h_0/L_0 \approx 2$ and $D_0/L_0 = 4 + \delta$. Specifically, $Re = 50$, L_0 is set to be 500 m and the domain depth D_0 is 3000 m. Horizontal and vertical grid spacing are given by $\Delta x/L_0 = \Delta y/L_0 = 1/12.5$ and $\Delta z/L_0 = 1/25$.

Fig. 4.4 shows a set of 3D simulations with different stress conditions in two different models. Fig. 4.4a and b show that for the u fields produced by the two full stress conditions, the patterns are basically the same, confirming the local and matrix implementations again produce roughly equivalent results. The pattern difference between the normal-gradient conditions showed in Fig. 4.4c and Fig. 4.4d is also minor. Fig. 4.4e and Fig. 4.4f show results using the flat-boundary assumption. However, unlike in Fig. 4.3, the WRF flat boundary result in Fig. 4.4f uses the default (public) version of WRF, rather than the flat boundary implementation described in section 4.3.2. It is very clear that the E07 flat-boundary condition decelerated the flow least. Fig. 4.4f shows that in this free-slip 3D test, the default WRF stress condition gave a similar pattern as in the normal gradient cases (i.e., Fig. 4.4c and d). The difference between the default WRF stress condition and the flat

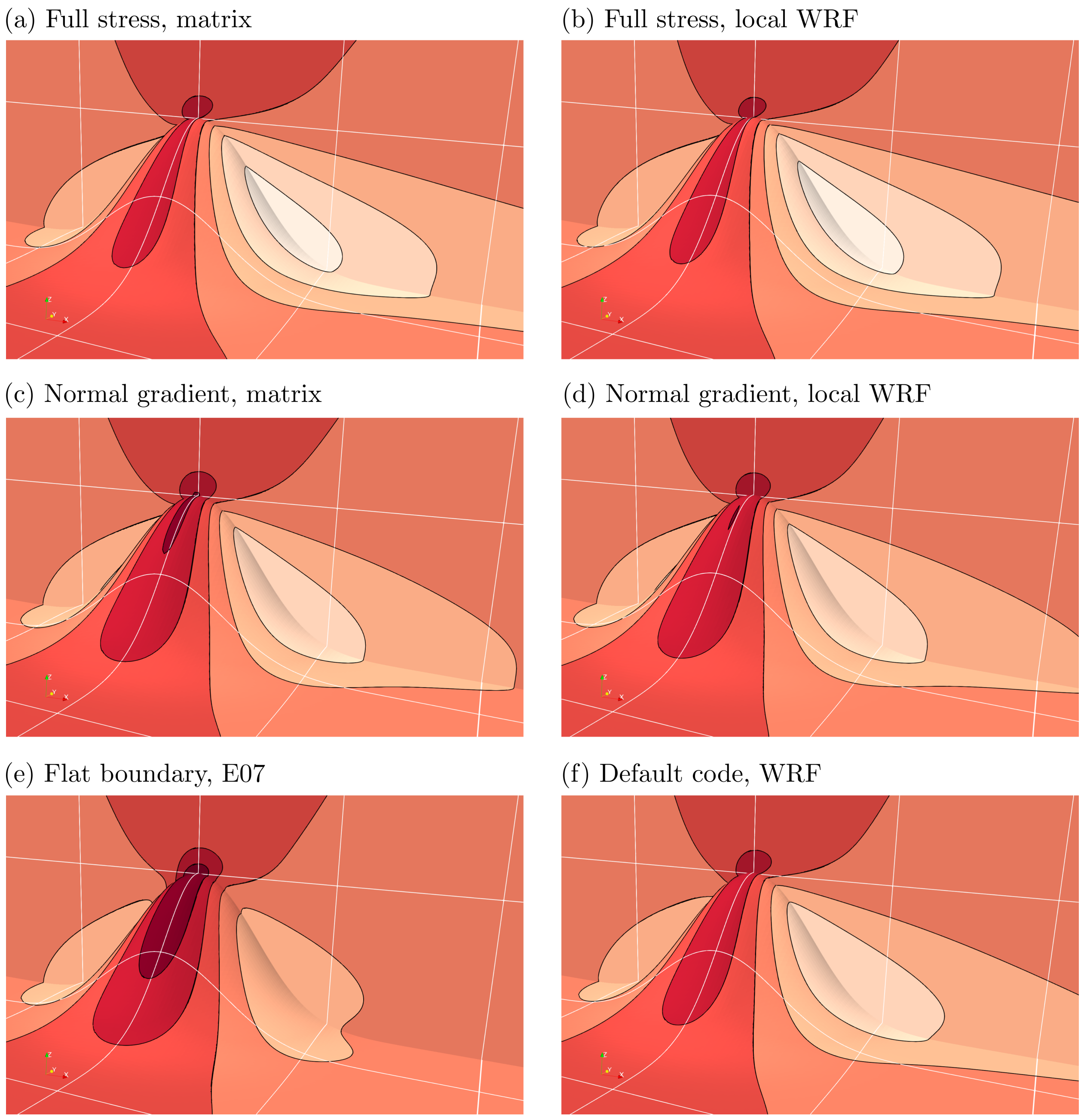

(a) Full stress, matrix
(b) Full stress, local WRF
(c) Normal gradient, matrix
(d) Normal gradient, local WRF
(e) Flat boundary, E07
(f) Default code, WRF

Figure 4.4: Numerical simulations for free-slip flow with Re = 50 past a 3D mountain at time Ut/L = 16.8. (a), (b) The idealized model with the matrix full-stress condition and the WRF model with the local full-stress condition; (c), (d) The semi-idealized model with the matrix normal-gradient condition and the WRF model with the local normal-gradient condition; (e) The semi-idealized model with the flat-boundary condition; and (f) WRF model with the default condition. All the plots show x-component velocity (c.i. = 2.5 m/s; darker color indicates higher horizontal velocity).

boundary condition in E07 will be introduced in the following section.

4.4.2 Flat boundary condition in the default WRF

As mentioned in sections 2.1.4, the flat boundary condition is employed by most current generation atmospheric models, and WRF is not an exception. In WRF, only two components at the lower boundary are specified, which are τ_{13} and τ_{23}. However, by exploring the WRF code, we found that the "flat-boundary" condition of default WRF is different from the idealized model described in Chapter 3. The details are:

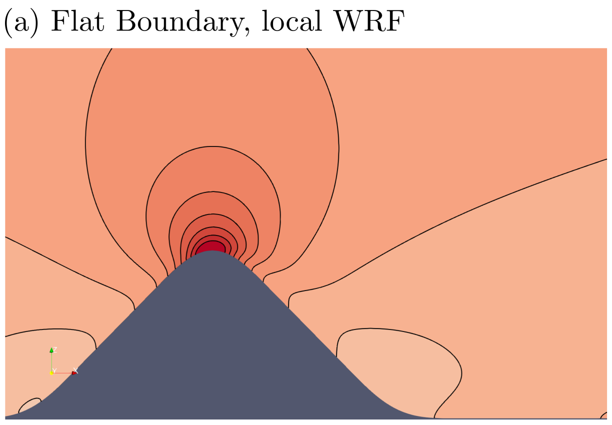
- *Surface wind fields used for the calculation of the stresses.* In the default WRF code, although the velocity fields for u and v are not stored or used (i.e., these two wind components are located on the half vertical level due to the vertical grid staggering), polynomial extrapolations of u and v from the lowest three model half levels to the surface were indeed employed for the use of stress calculation. That is, to calculate stresses at the first half level(τ_{11}, τ_{12}, and τ_{22}), vertical derivatives du/dz and dv/dz need the values for surface wind fields. Unlike what default WRF did, we just calculated the surface u and v fields by using (2.19) in the semi-idealized model described in Chapter 3. The extrapolation is obviously an approximation to specify the stress components, which may lead to inaccuracy or even errors.

  Moreover, in E07 model, u and v fields at the surface were combined with the no-penetration condition 2.6c to update the surface w field, which would contribute to the calculations of τ_{13} and τ_{23} at the surface (i.e., the dw/dx and dw/dy terms).

- *Make use of the surface stress components.* Once the stress components were obtained, they could be used to calculate the wind tendencies for the u and v fields at the first half level. For example

$$\frac{\partial u}{\partial t} = \frac{\partial \tau_{11}}{\partial x} + \frac{\partial \tau_{12}}{\partial y} + \frac{\partial \tau_{13}}{\partial z} = \frac{\partial \tau_{11}}{\partial X} + \frac{\partial \tau_{12}}{\partial Y} + \frac{\partial \tau_{11}}{\partial q}\frac{\partial q}{\partial x} + \frac{\partial \tau_{12}}{\partial q}\frac{\partial q}{\partial y} + \frac{\partial \tau_{13}}{\partial q}\frac{\partial q}{\partial z} \tag{4.15}$$

so that the stresses at the surface can be used for calculating the $\partial/\partial q$ terms when dealing with the vertical derivatives. However, the default WRF code just assign stresses to be zero at the ground except for τ_{13} and τ_{23}, which is different to E07 significantly.



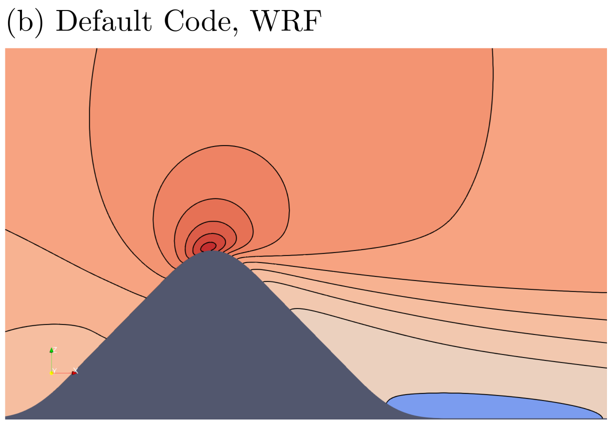


Figure 4.5: Numerical simulations for flat-boundary flow with (a) flat boundary condition described in Chapter 3, but applied in WRF code; (b) default version WRF code. (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity).

From the differences mentioned above, we can expect the resulting flow patterns shown by two different flat-boundary conditions (semi-idealized model and default WRF) might be different from each other. Fig. 4.5 shows the u component field at the same model running time as in Fig. 4.3. Fig. 4.5a is the same as Fig. 4.3f, the flat-boundary condition used in the semi-idealized model was adopted by the WRF code; While Fig. 4.5.b used the default WRF code. The difference between two conditions is clear—wind was decelerated more in the default WRF case downstream, and there is a small region with low-speed reverse flow, which never showed up in Fig. 4.5.a, or even in the normal-gradient stress runs (Fig. 4.3b and f). Actually, the default WRF stress condition in Fig. 4.5b shows a wind pattern that is somehow between the full-stress condition case pattern and the normal-gradient condition case pattern, so that if the full stress condition is considered to be the "correct" one, the performance of the default WRF "flat boundary" condition is more acceptable than the other two approximates in this test; nonetheless, the reason for its relative good representing of the wind field is hard to track, since some of the rough estimates made by the default WRF is difficult to be understood.

4.4.3 WRF Resolution tests

Although the high resolution runs in Fig. 4.3 showed identical wind patterns, inaccuracies might show up if the resolution was getting coarser by using the local normal-gradient stress condition (as well as the local full-stress condition). The process of finding the intersection points and bi-linear interpolations will be imprecise.

To test whether the local methods are still valid in lower resolution, a series of resolution tests were carried out for different stress conditions.

Fig. 4.6 shows the simulations with the full-stress condition in different resolutions, the left column gives the horizontal velocity fields by adopting the matrix method while the right one shows the local condition ones in WRF. Fig. 4.6a and b are just Fig. 4.3c and f with $\Delta x = 1/40L_0$ and $\Delta z = 1/60L_0$, these two cases are fairly close to each other. At the resolution of $\Delta x = 1/20L_0$ ($\Delta x/\Delta z$ stays proportional in this series of tests, so only Δx is specified here and later), which are shown in Fig. 4.6c and Fig. 4.6d, the difference is still minor. Larger difference showed up in the third line ($\Delta_x = 1/10L_0$), one less contour was at the top of the barrier in Fig. 4.6f, and the dark blue reversed wind region ($-5.0m/s \leq u \leq -2.5m$ was getting larger and closer to the mountain; in contrast, Fig. 4.6e was still similar to Fig. 4.6a and c. For the last pair of the full stress resolution runs, $\Delta x/L_0 = 1/5$, and obvious difference was seen by comparing the coarse WRF run Fig. 4.6g to the corresponding highest resolution case, and this difference is much smaller in the left column matrix tests. It seems that the matrix method performed better than the local WRF implementation in the lower resolution cases.

Fig. 4.7 are the runs with the exact same setting as their corresponding case except for adopting the normal-gradient conditions (matrix or local method). Basically, the matrix normal gradient method performed as good as the full stress condition in the lower resolutions; while the local normal gradient application also showed a similar magnitude of inaccuracy in the largest grid

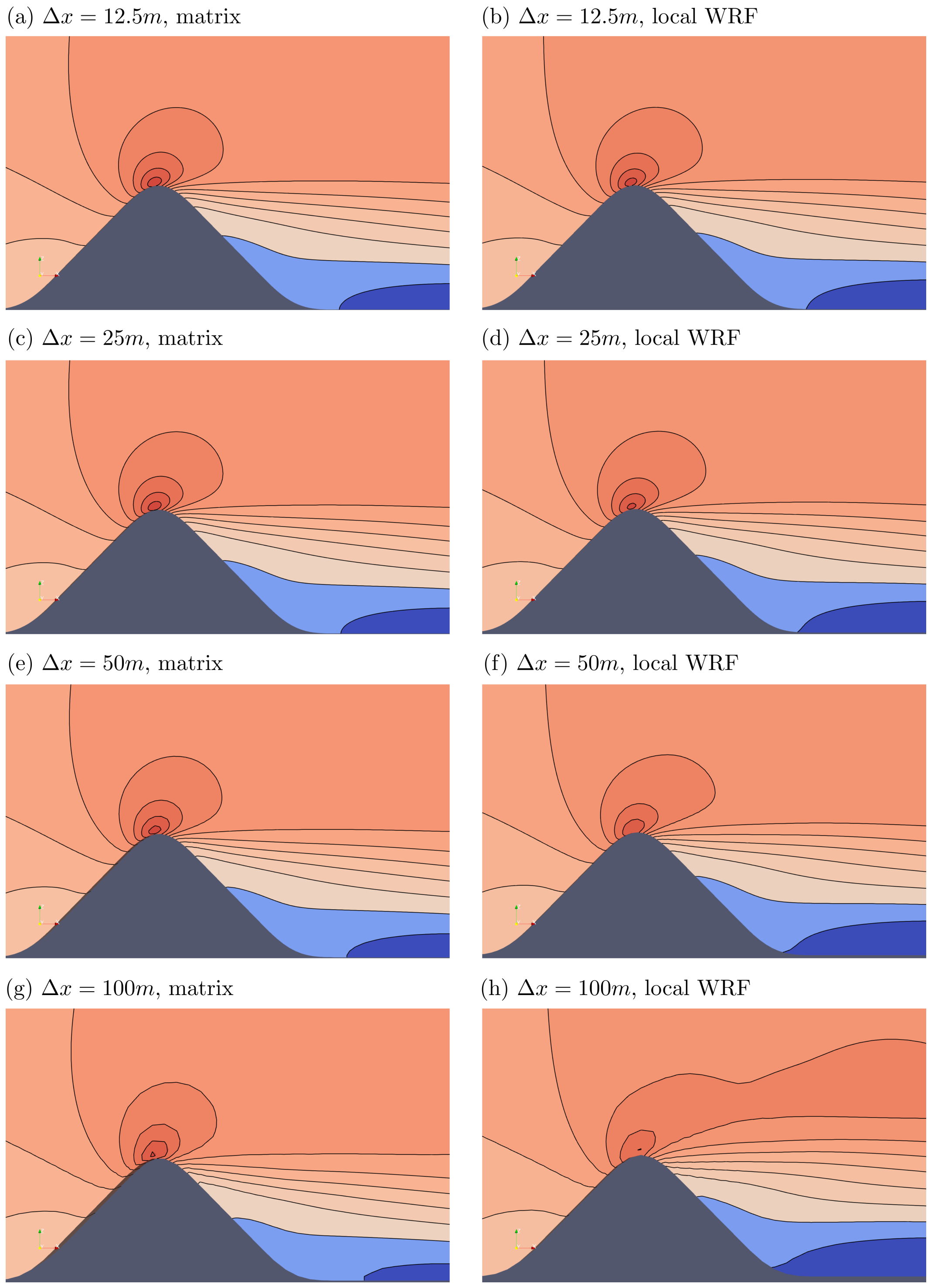
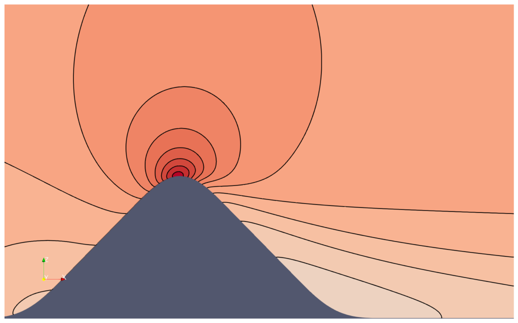
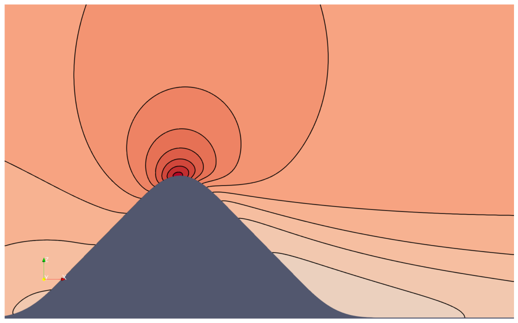


Figure 4.6: Numerical simulations for free-slip flow with full stress condition (matrix method used in the left column, local method used in the right column) past a 2D ridge at time Ut/L = 24. (a), (b) $\Delta x = 1/40L_0$; (c), (d) $\Delta x = 1/20L_0$; (e), (f) $\Delta x = 1/10L_0$; (g), (h) $\Delta x = 1/5L_0$. (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity).

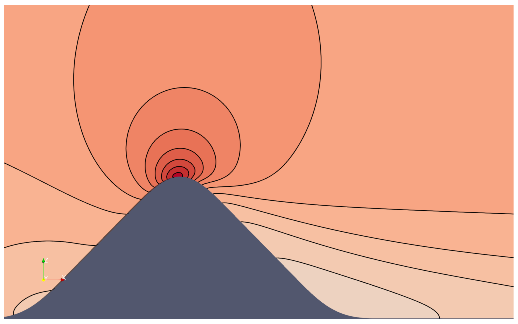
(a) $\Delta x = 12.5m$, matrix



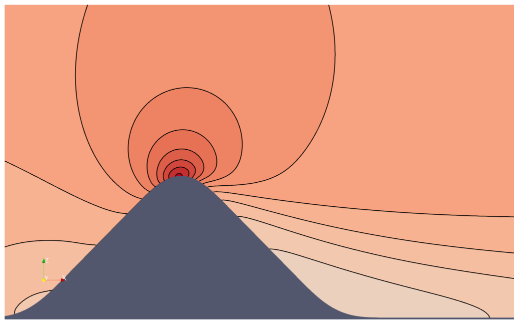
(b) $\Delta x = 12.5m$, local WRF



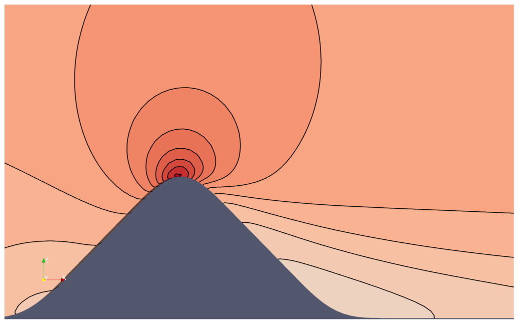
(c) $\Delta x = 25m$, matrix



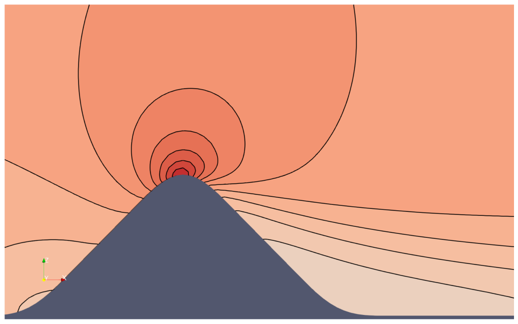
(d) $\Delta x = 25m$, local WRF



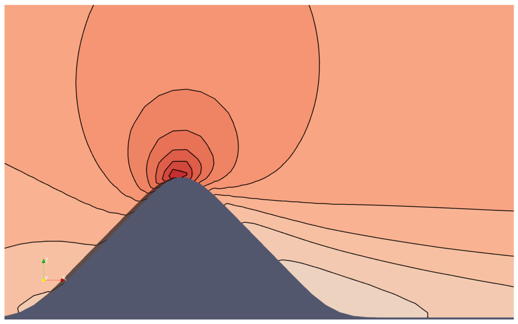
(e) $\Delta x = 50m$, matrix



(f) $\Delta x = 50m$, local WRF



(g) $\Delta x = 100m$, matrix



(h) $\Delta x = 100m$, local WRF

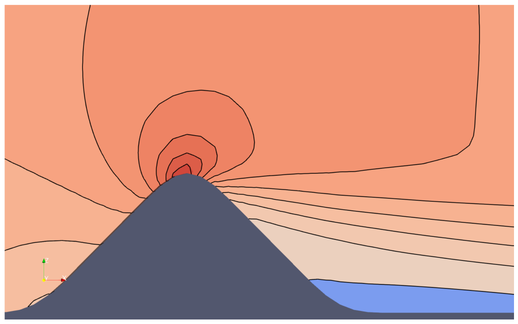


Figure 4.7: Same as 4.6, but with normal gradient conditions

spacing run as the full stress case did.

Same as Fig. 4.6 and Fig. 4.7, Fig. 4.8 exhibited the u pattern by using flat-boundary stress condition in two models. The flat boundary WRF case with the lowest resolution also showed a nearly equal inaccuracy as the full stress or normal gradient case did, which suggested that the wind field inaccuracies produced by the lower resolution cases in WRF were more likely caused by the numerical methods of WRF model instead of the boundary conditions.

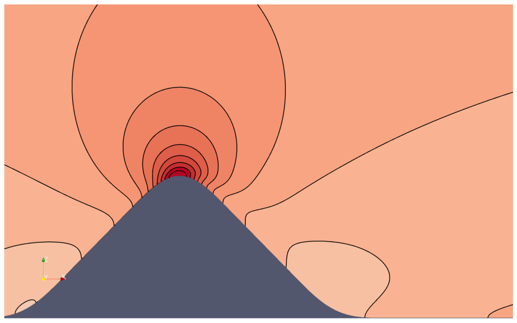
Changing the horizontal and vertical resolutions separately instead of proportionally might be helpful to better understand the inaccuracies produced by the lower resolution cases. Fig. 4.9 show the horizontal resolution tests using full stress condition by keeping $\Delta z = 1/60L_0$, while changing the horizontal resolution ($\Delta x = 1/40L_0$ for Fig. 4.9a and d, $1/20L_0$ for Fig. 4.9b and e, and $1/10L_0$ for Fig. 4.9c and f), which means Fig. 4.9a and d are same as Fig. 4.3a and d. Fig. 4.9a-c gave a similar and small u pattern change as shown in Fig. 4.6a, c and e, which suggested that the E07 matrix full stress implementation was not sensitive to the vertical resolution but was mostly affected by the horizontal resolution. However, Fig. 4.9d-f showed a different u pattern shift as in Fig. 4.6b,d and f, especially in Fig. 4.9f, the pattern differed drastically to other cases, which implied that when the grid spacing ratio $\Delta x/\Delta z$ was not close to one, the inaccuracy created by the numerics in WRF might be significant.

Fig. 4.10 show the vertical resolution tests using full stress condition by keeping $\Delta x = 1/10L_0$, while changing the vertical resolution ($\Delta z = 1/60L_0$ for Fig. 4.10a and d, $1/30L_0$ for Fig. 4.10b and e, and $1/15L_0$ for Fig. 4.10c and f). The matrix method cases showed nearly identical results, which proved that the matrix full implementation is not sensitive to the vertical resolution. And by comparing Fig. 4.10d-f, it seems that when the grid spacing ratio $\Delta x/\Delta z$ got back to close to 1, the results shown were getting normal.

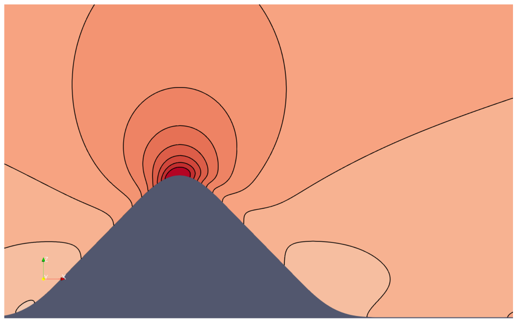
4.5 Summary

A method was developed for imposing the surface stress boundary condition over complex terrain, taking full account of the terrain slope and curvature. The method is based on a reformulation of the full stress boundary condition to an expression involving two terms: a normal gradient term

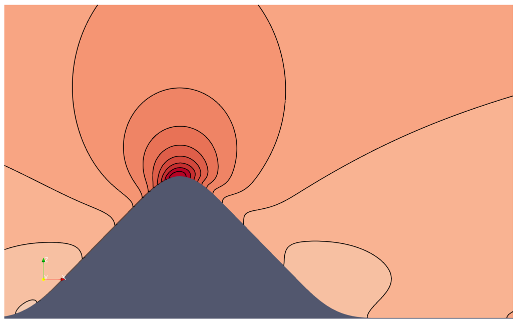
(a) $\Delta x = 12.5m$, E07



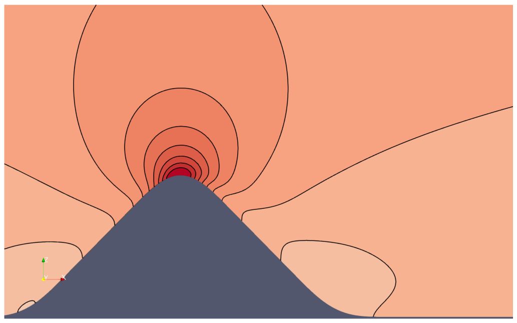
(b) $\Delta x = 12.5m$, WRF



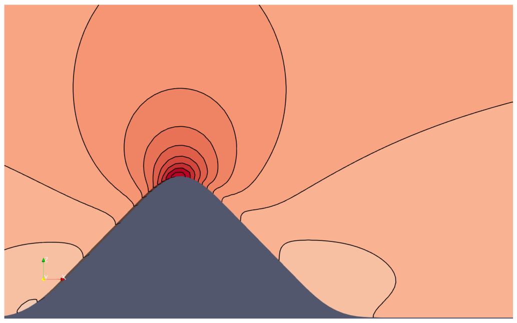
(c) $\Delta x = 25m$, E07



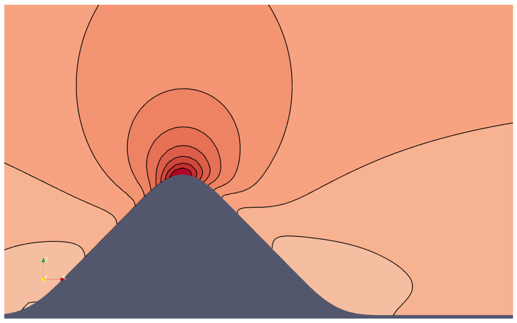
(d) $\Delta x = 25m$, WRF



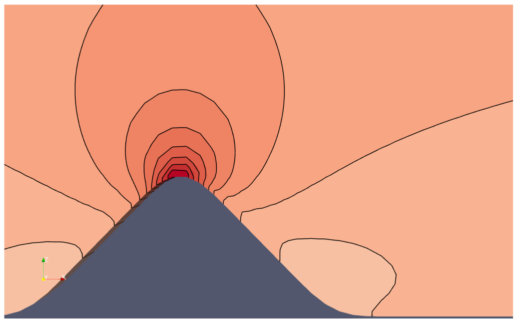
(e) $\Delta x = 50m$, E07



(f) $\Delta x = 50m$, WRF



(g) $\Delta x = 100m$, E07



(h) $\Delta x = 100m$, WRF

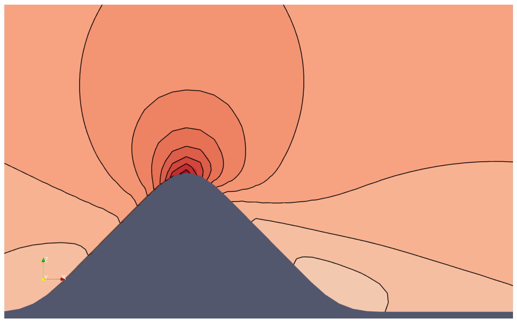


Figure 4.8: Same as 4.6, but with flat boundary conditions

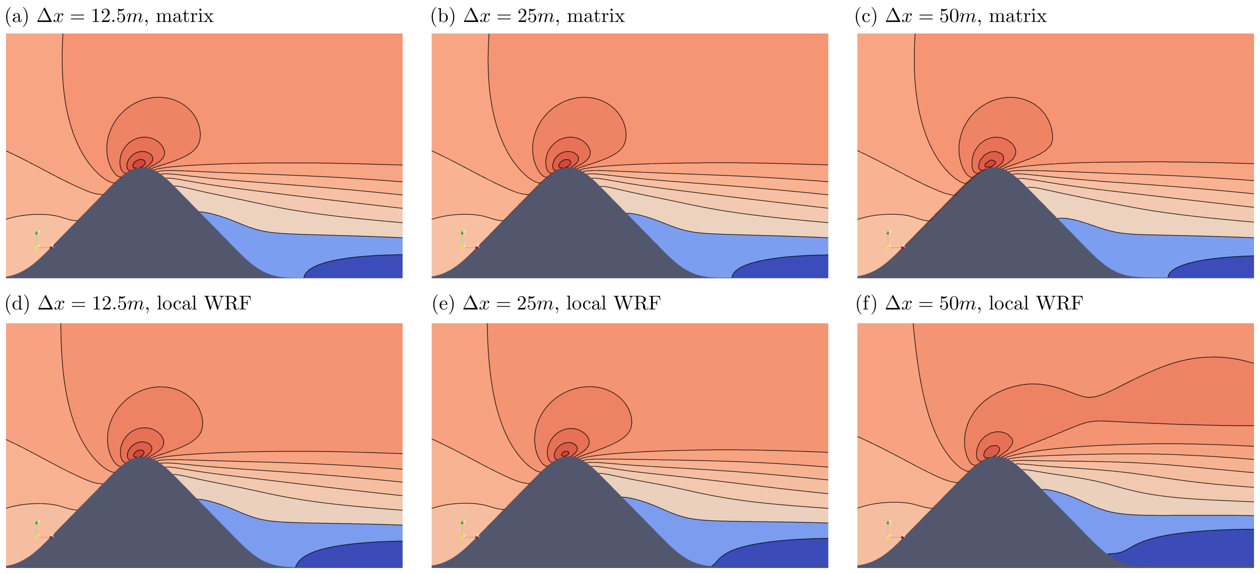


Figure 4.9: Numerical simulations for free-slip flow with full stress condition (matrix method used in the top panels, local method used in the lower panels) and fixed vertical grid spacing past a 2D ridge at time Ut/L = 24. (c.i. = 2.5 m/s; reddish indicates positive and blue means negative, darker color indicates higher absolute horizontal velocity).

and a second Dirichlet term, describing the rate of change in terrain slope along a parcel trajectory. Unlike the approach of E07, the new method avoids coupling nearby points along the boundary and can thus be implemented entirely locally, without the need for a global matrix inversion. As a result, the method is significantly more straightforward to implement, particularly in the context of parallelized models using domain decomposition, in which global methods are difficult to implement.

For the present study, the new method was implemented in the context of the widely used Weather Research and Forecasting (WRF) model. The normal derivative terms are computed by first starting from the surface data points and finding the intersection between the terrain normal direction and the first interior coordinate surface, and then interpolating the wind components to the intersection point so as to allow normal differences to be computed. Combining the normal difference terms with the Dirichlet curvature terms then gives a simple algebraic system for the surface wind fields. The surface wind fields are used to compute deformations and stresses at the boundary, which then feed into the subgrid turbulence parameterization at the lowest grid

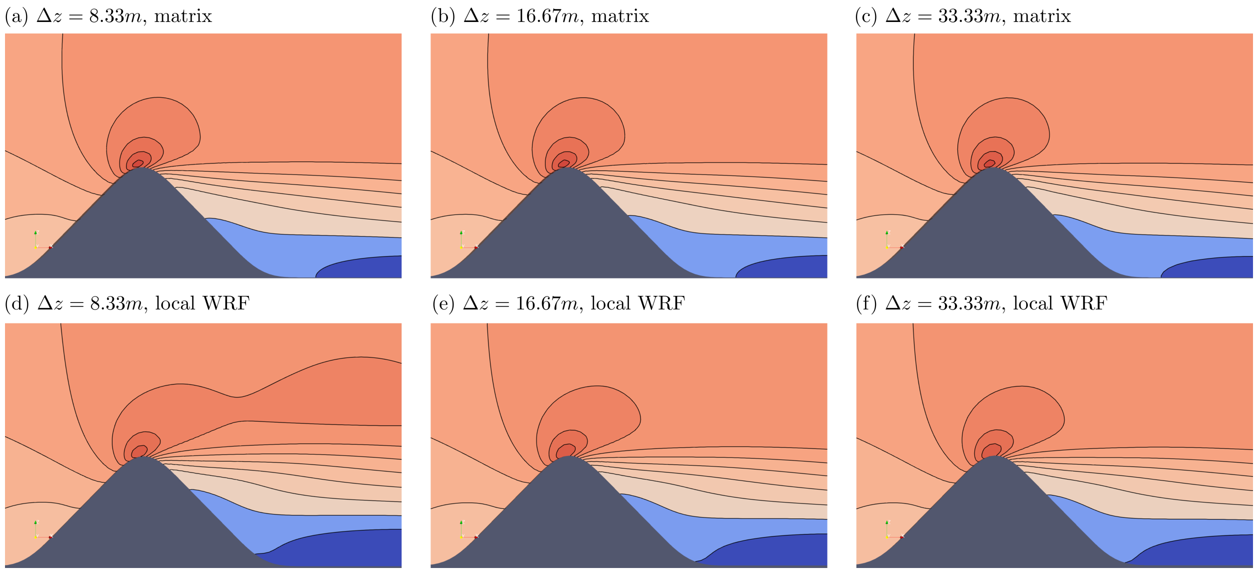


Figure 4.10: Same as 4.9 but with fixed horizontal grid spacing

level. Two approximations to the boundary condition were also implemented: a normal gradient condition, using a method similar that used for the full condition; and a flat boundary assumption, in which the vertical derivatives are specified directly.

The new method was verified by comparing to simulations using the matrix method of E07, for the case of idealized 2D and 3D flows past an isolated hill. The results show that for sufficiently high resolution tests (in this case $\Delta x = L/40$), the local and matrix implementations of the stress condition produce nearly identical results. For lower resolution, the agreement between the two implementations remains relatively strong for horizontal grid spacing as large as $L/10$, while for larger grid spacing the two models begin to show significant differences. However, a similar degree of difference was found in tests using the flat boundary approximation, which is implemented the same in the two models. This suggests the divergence of the model results for larger grid spacings is likely due to differences in the general numerics of the two models, as opposed to the boundary condition in particular. In all cases, the simulations using the full stress condition showed significant differences from simulations using the flat boundary and normal gradient approximations, highlighting the effects of terrain slope and curvature in the implementation of the stress.

Interestingly, although the default (public) version of WRF uses the flat boundary assumption, simulations using the default WRF produced results very different from the flat boundary simulations described above. Inspection of the code shows the differences are due to additional assumptions made by the WRF implementation at the boundary, similar to those described in section 2.2.1 for the Clark (1977) model. While the extra assumptions are not formally correct, the results for the present experiments showed that the default WRF produced somewhat better results than the simple flat boundary assumption. Specifically, the default WRF showed a similar level of agreement with the full stress runs as the normal gradient approximation.

While our results suggest that the method described in this study provides a valid alternative to the matrix method, it should be kept in mind our verification tests were all carried out using highly idealized flows. Further testing is needed to determine how the method performs under more realistic conditions, such as those described in chapter 3. Indeed, our new implementation should allow us to go one step further, by considering cases with the full range of physical parameterizations available in WRF.

5. CONCLUSIONS

In the previous chapters, the behavior and dynamics of the surface drag boundary condition were explored, both in terms of the basic physics and the numerical implementation of the condition. In chapter 3, a series of semi-idealized numerical experiments was presented to explore the basic physics of the condition. The results suggest that the behavior of the drag condition is strongly dependent on the geometry of the terrain, specifically on the terrain slope and curvature, and that the approximations currently in widespread use fail to capture the behavior of the condition over complex terrain. In chapter 4, a new method for implementing the condition was presented. The method is based on a reformulation of the drag condition that allows the method to be implemented locally, thus streamlining the implementation, particularly for parallelized models using domain decomposition. Taken together, these results suggest that implementing the full stress boundary condition should likely play an important role in future high-resolution forecast models.

That said, the results in this thesis come with a number of important caveats. First, as mentioned in previous chapters, our numerical experiments were carried out primarily in the context of idealized or semi-idealized modeling. Further experiments are needed to show how the full stress condition behaves in more realistic flows. Fortunately, the new implementation developed in chapter 4 should allow us to consider such experiments using the full range of physical parameterizations available in WRF. Perhaps more importantly, the present experiments were carried out entirely in the perfect model framework, in which the simulations making use of the full stress implementation were considered to be correct. However, a better representation of the physical processes in the model does not always translate into better predictions. To verify the usefulness of the condition, model experiments using the full stress condition will need to be compared with observations.

In addition to the model evaluations mentioned above, the results of chapters 3 and 4 suggest several important directions for future exploration. One important question is the behavior of the stress condition at even higher resolutions than considered in the present study. As discussed in chapter 3, the errors in the flat boundary and normal gradient approximations did not not show any sign of leveling off at the grid spacing of $\Delta x = 90m$. At some point, we might expect the errors to saturate as the grid spacing decreases, but experiments would be needed to understand how this saturation occurs.

Another more general issue for future study is the behavior of the drag parameterization itself over complex terrain. As mentioned in chapter 2, the Monin-Obukhov theory was developed for turbulence over a flat boundary. However, over a sloped boundary, the shear axis of the flow is no longer parallel to gravity, which would have an impact on the turbulent eddies and the related fluxes when buoyancy is a factor. A first step to explore this issue might involve high resolution (large-eddy simulation) modeling of the surface layer itself over boundaries of varying slope, and with varying heat fluxes.

Finally, a third potential direction of interest is the problem of parameterizing the effects of complex terrain in lower resolution models, where the slope and curvature of the resolved terrain are not well represented. Development of a parameterization of this type is likely to prove a difficult problem. Even so, one could imagine developing a statistical representation of the geometric properties of the terrain in a given region, and then using this representation to estimate the associated drag effects.

REFERENCES

Businger, J. A., Wyngaard, J. C., Izumi, Y., & Bradley, E. F. (1971). Flux-profile relationships in the atmospheric surface layer. *Journal of the atmospheric Sciences*, *28*(2), 181–189.

Clark, T. L. (1977). A small-scale dynamic model using a terrain-following coordinate transformation. *J. Comp. Phys.*, *24*, 186–215.

Corby, G. A., Gilchrist, A., & Newson, R. L. (1972). A general circulation model of the atmosphere suitable for long period integrations. *Quarterly Journal of the Royal Meteorological Society*, *98*(418), 809–832.

Dubayah, R., & Loechel, S. (1997). Modeling topographic solar radiation using goes data. *Journal of Applied Meteorology*, *36*(2), 141–154.

Durran, D. R. (1999). *Numerical methods for wave equations in geophysical fluid dynamics* (Vol. 32). Springer Science & Business Media.

Durran, D. R., & Klemp, J. B. (1983). A compressible model for the simulation of moist mountain waves. *Monthly Weather Review*, *111*(12), 2341–2361.

Dyer, A. (1974). A review of flux-profile relationships. *Boundary-Layer Meteorology*, *7*(3), 363–372.

Epifanio, C. C. (2007). A method for imposing surface stress and heat flux conditions in finite-difference models with steep terrain. *Monthly weather review*, *135*(3), 906–917.

Epifanio, C. C., & Durran, D. R. (2001). Three-dimensional effects in high-drag-state flows over long ridges. *Journal of the atmospheric sciences*, *58*(9), 1051–1065.

Epifanio, C. C., & Rotunno, R. (2005). The dynamics of orographic wake formation in flows with upstream blocking. *Journal of the atmospheric sciences*, *62*(9), 3127–3150.

Gal-Chen, T., & Somerville, R. C. (1975). On the use of a coordinate transformation for the solution of the navier-stokes equations. *Journal of Computational Physics*, *17*(2), 209–228.

Haupt, S. E., Kosovic, B., Shaw, W., Berg, L. K., Churchfield, M., Cline, J., ... others (2019). On bridging a modeling scale gap: Mesoscale to microscale coupling for wind energy. *Bulletin of* *the American Meteorological Society*, *100*(12), 2533–2550.

Holton, J. R. (1992). *An introduction to dynamic meteorology* (3^{rd} ed.). Academic Press.

Janjić, Z. I. (1989). On the pressure gradient force error in σ-coordinate spectral models. *Monthly weather review*, *117*(10), 2285–2292.

Klemp, J. B., & Wilhelmson, R. B. (1978). The simulation of three-dimensional convective storm dynamics. *Journal of Atmospheric Sciences*, *35*(6), 1070–1096.

Levin, S. A., Muller-Landau, . H. C., Nathan, . R., & Chave, . J. (2003). The ecology and evolution of seed dispersal: a theoretical perspective. *Annual Review of Ecology, Evolution, and Systematics*, *34*(1), 575–604.

Loos, C., Seppelt, R., Meier-Bethke, S., Schiemann, J., & Richter, O. (2003). Spatially explicit modelling of transgenic maize pollen dispersal and cross-pollination. *Journal of Theoretical Biology*, *225*(2), 241–255.

Mass, C. F., Ovens, D., Westrick, K., & Colle, B. A. (2002). Does increasing horizontal resolution produce more skillful forecasts? the results of two years of real-time numerical weather prediction over the pacific northwest. *Bulletin of the American Meteorological Society*, *83*(3), 407–430.

Mesinger, F. (1982). On the convergence and error problems of the calculation of the pressure gradient force in sigma coordinate models. *Geophysical & Astrophysical Fluid Dynamics*, *19*(1-2), 105–117.

Mihailović, D., & Janjić, Z. (1986). Comparison of methods for reducing the error of the pressure gradient force in sigma coordinate models. *Meteorology and Atmospheric Physics*, *35*(3), 177–184.

Monin, A. S., & Obukhov, A. M. (1954). Basic laws of turbulent mixing in the surface layer of the atmosphere. *Contrib. Geophys. Inst. Acad. Sci. USSR*, *151*(163), e187.

Obukhov, A. (1971). Turbulence in an atmosphere with a non-uniform temperature. *Boundary-layer meteorology*, *2*(1), 7–29.

Oliphant, A., Spronken-Smith, R., Sturman, A., & Owens, I. (2003). Spatial variability of surface radiation fluxes in mountainous terrain. *Journal of Applied Meteorology*, *42*(1), 113–128.

Sathe, A., Mann, J., Vasiljevic, N., & Lea, G. (2015). A six-beam method to measure turbulence statistics using ground-based wind lidars. *Atmospheric Measurement Techniques*, *8*(2), 729–740.

Scherer, D., & Parlow, E. (1994). Terrain as an important controlling factor for climatological, meteorological and hydrological processes in nw-spitzbergen. *Zeitschrift für Geomorphologie. Supplementband*(97), 175–193.

Smolarkiewicz, P. K., & Margolin, L. G. (1994). Variational solver for elliptic problems in atmospheric flows. *Appl. Math. and Comp. Sci.*, *4*, 527–551.

Smolarkiewicz, P. K., Sharman, R., Weill, J., Perry, S. G., Heist, D., & Bowker, G. (2007). Building resolving large-eddy simulations and comparison with wind tunnel. *J. Comp. Phys.*, *227*, 633–653.

Stockie, J. M. (2011). The mathematics of atmospheric dispersion modeling. *Siam Review*, *53*(2), 349–372.

Turner, R., & Hurst, T. (2001). Factors influencing volcanic ash dispersal from the 1995 and 1996 eruptions of mount ruapehu, new zealand. *Journal of applied meteorology*, *40*(1), 56–69.

Viner, K. C., Epifanio, C. C., & Doyle, J. D. (2013). A steady-state solver and instability calculator for nonlinear internal wave flows. *J. Comp. Phys.*, *251*, 432–444.

Weissman, G., Sargent, R., & Fanshaw, B. (2018). Renewables on the rise: A decade of progress toward a clean energy future. *Environment America: Research & Policy Center. Last accessed November*, *4*, 2018.

Yang, Y., Wilson, L., Makela, M., & Marchetti, M. (1998). Accuracy of numerical methods for solving the advection–diffusion equation as applied to spore and insect dispersal. *Ecological modelling*, *109*(1), 1–24.

Zängl, G. (2007). To what extent does increased model resolution improve simulated precipitation fields? a case study of two north-alpine heavy-rainfall events. *Meteorologische Zeitschrift*, *16*(5), 571–580.

APPENDIX A

WRF CODING

A.1 Registry

In RegistryRegistry.EM_COMMON, we added arrays to hold the values of the surface wind fields as well as deformation values at the ground:

```
state  real  defor11_sfc   ij   misc    1   -   r    "defor11_sfc"   "DEFORMATION␣11␣AT␣SURFACE"   "s-1"
state  real  defor22_sfc   ij   misc    1   -   r    "defor22_sfc"   "DEFORMATION␣22␣AT␣SURFACE"   "s-1"
state  real  defor33_sfc   ij   misc    1   -   r    "defor33_sfc"   "DEFORMATION␣33␣AT␣SURFACE"   "s-1"
state  real  defor12_sfc   ij   misc    1   -   r    "defor12_sfc"   "DEFORMATION␣12␣AT␣SURFACE"   "s-1"
state  real  u_sfc_u          ij   dyn_em   1   X   irh   "USFCU"  "surface␣u␣velocity␣at␣u␣vort␣pts"
state  real  v_sfc_u          ij   dyn_em   1   X   irh   "VSFCU"  "surface␣v␣velocity␣at␣u␣vort␣pts"
state  real  w_sfc_u          ij   dyn_em   1   X   irh   "WSFCU"  "surface␣w␣velocity␣at␣u␣vort␣pts"
state  real  u_sfc_v          ij   dyn_em   1   Y   irh   "USFCV"  "surface␣u␣velocity␣at␣v␣vort␣pts"
state  real  v_sfc_v          ij   dyn_em   1   Y   irh   "VSFCV"  "surface␣v␣velocity␣at␣v␣vort␣pts"
state  real  w_sfc_v          ij   dyn_em   1   Y   irh   "WSFCV"  "surface␣w␣velocity␣at␣v␣vort␣pts"
```

And we also made sure these surface arrays are included in the halo updates between tiles:

```
halo    HALO_EM_TKE_C dyn_em  8:u_sfc_u, v_sfc_v, w_sfc_u, w_sfc_v
halo    HALO_EM_TKE_D dyn_em  8:defor11_sfc,defor22_sfc,defor12_sfc
```

A.2 get_norm_intersection and get_surf_winds

I combined get_norm_intersection and get_surf_winds to one subroutine: get_norm_intersection_surf_winds. So now it is the only new subroutine we added in the WRF code, and it is placed at the beginning of dynmodule_diffusion_em.F.

```
   SUBROUTINE get_norm_intersection_surf_winds(config_flags, isotropic,        &
                                 ph, phb, z_at_u_surf, z_at_v_surf,          &
                                 n1_u, n1_v, n2_u, n2_v, n3_u, n3_v,          &
                                 xlat_u, xlat_v, xlong_u, xlong_v,           &
                                 xlat, xlong, dx, dy, msftx, msfty,          &
                                 msfux, msfuy, msfvx, msfvy,         &
                                 delta_x_u, delta_x_v, delta_y_u, delta_y_v,        &
                                 delta_h_u, delta_h_v, dsu, dsv,            &
                                 delta_h_u_f, delta_x_u_f, delta_y_u_f, &
                                 delta_h_v_f, delta_x_v_f, delta_y_v_f, &
                                 u_norm_u, v_norm_u, w_norm_u,      &
                                 u_norm_v, v_norm_v, w_norm_v,      &
                                 u_2, v_2, w_2,                     &
                                 xkmv, xkmh, xkhv, xkhh,            &
                                 rdzw, dt, mix_upper_bound,         &
                                 u_sfc_u, v_sfc_u, w_sfc_u,         &
```

```
                                          u_sfc_v ,  v_sfc_v ,  w_sfc_v ,         &
                                          D_x_u ,  D_y_u ,  D_x_v ,  D_y_v ,        &
                                          ust ,  rho ,                              &
                                          ids ,  ide ,  jds ,  jde ,  kds ,  kde ,      &
                                          ims ,  ime ,  jms ,  jme ,  kms ,  kme ,      &
                                          its ,  ite ,  jts ,  jte ,  kts ,  kte )
   IMPLICIT NONE

   TYPE( grid_config_rec_type ) , INTENT( IN )  &
   :: config_flags

   INTEGER, INTENT( IN )  &
   :: ids , ide , jds , jde , kds , kde , &
      ims , ime , jms , jme , kms , kme , &
      its , ite , jts , jte , kts , kte , isotropic

   REAL, DIMENSION( ims:ime , kms:kme , jms:jme ) , INTENT( INOUT )  &
   :: ph , phb , u_2 , v_2 , w_2 , rdzw , rho

   REAL, DIMENSION( ims:ime , kms:kme , jms:jme ) , INTENT( INOUT )  &
   :: xkmv , xkmh , xkhv , xkhh

   REAL, DIMENSION( ims:ime , jms:jme ) , INTENT( IN )  &
   :: xlat , xlong , xlat_u , xlat_v , xlong_u , xlong_v , ust , &
      msftx , msfty

   REAL, DIMENSION( ims:ime , jms:jme ) , INTENT( OUT )  &
   :: z_at_u_surf , n1_u , n2_u , n3_u , dsu ,        &
      delta_x_u , delta_y_u , delta_h_u , u_norm_u , v_norm_u , w_norm_u,&
      u_sfc_u , v_sfc_u , w_sfc_u , D_x_u , D_y_u ,           &
      msfux , msfuy , delta_h_u_f , delta_x_u_f , delta_y_u_f

   REAL, DIMENSION( ims:ime , jms:jme ) , INTENT( OUT )  &
   :: z_at_v_surf , n1_v , n2_v , n3_v , dsv ,        &
      delta_x_v , delta_y_v , delta_h_v , u_norm_v , v_norm_v , w_norm_v,&
      u_sfc_v , v_sfc_v , w_sfc_v , D_x_v , D_y_v ,           &
      msfvx , msfvy , delta_h_v_f , delta_x_v_f , delta_y_v_f

   REAL, INTENT( IN ) :: dx , dy , dt , mix_upper_bound

 ! Local variables.

   INTEGER :: i,j,k, i_start , j_start , i_end , j_end , ktf , i_dir , j_dir

   REAL :: A1, A2, z0 , x1 , x2 , y1 , y2 , &
           h11 , h12 , h21 , h22 , AA, BB, CC, h_max , h_min , deltas , &
           epsln , V0_u , V0_v , aa1 , aa2 , aa3 , aa4 , aa5 , aa6 , dx_m , dy_m , &
           u_drag , v_drag , w_drag , d2hdx2 , d2hdy2 , d2hdxy , dhdx , r_curve


   LOGICAL, EXTERNAL :: wrf_dm_on_monitor
   CHARACTER*256     :: outstring

!History:     Sep 2019  Changes by Yi Li and Craig Epifanio , TAMU
!Comments:    This SUBROUTINE is trying to find the intersection point
!             between the surface normal direction from u points and v
!               points to the half-level surface of (ph + phb) / g ,
!               also with the calculation of the wind fields near the
!               ground over complex terrain.
!

   epsln = 1.e-10
   ! To start the calculation, we have to initiliaze kv and kh (vertical and horizontal viscosities)
   ktf = min(kte,kde-1)
   i_start = its
   i_end   = MIN(ite,ide-1)
   j_start = jts
   j_end   = MIN(jte,jde-1)

    IF (isotropic .EQ. 1) THEN
      DO j = j_start, j_end
      DO k = kts, ktf
      DO i = i_start, i_end
           deltas=(dx/msftx(i,j) * dy/msfty(i,j)/rdzw(i,k,j))**0.33333333
          xkmh(i,k,j)=max(xkmh(i,k,j), 1.0E-4*deltas*deltas )
          xkmv(i,k,j)=xkmh(i,k,j)
          xkhh(i,k,j)=xkmh(i,k,j)*3
          xkhh(i,k,j)=min(xkhh(i,k,j), mix_upper_bound * MIN(dx/msftx(i,j) * dy/msfty(i,j), 1/ rdzw(i,k,j) / rdzw(i,k,j)) / dt ) !new
          xkhv(i,k,j)=xkhh(i,k,j)
      ENDDO
      ENDDO
      ENDDO
    ENDIF

    !variable values at the boundaries are tricky, we need to define values there
    IF (its == ids) THEN
        DO j = j_start, j_end
            xkmh(its-1,kts,j) = xkmh(its,kts,j)
            xkmv(its-1,kts,j) = xkmv(its,kts,j)
        ENDDO
    ENDIF

    IF (jts == jds) THEN
        DO i = i_start, i_end
            xkmh(i,kts,jts-1) = xkmh(i,kts,jts)
            xkmv(i,kts,jts-1) = xkmv(i,kts,jts)
        ENDDO
    ENDIF

    IF (ite == ide) THEN
        DO j = j_start, j_end
            xkmh(ite,kts,j) = xkmh(ite-1,kts,j)
            xkmv(ite,kts,j) = xkmv(ite-1,kts,j)
        ENDDO
    ENDIF

    IF (jte == jde) THEN
        DO i = i_start, i_end
            xkmh(i,kts,jte) = xkmh(i,kts,jte-1)
            xkmv(i,kts,jte) = xkmv(i,kts,jte-1)
        ENDDO
    ENDIF
```

```
IF (its == ids .and. jts==jds) THEN
    phb(its-1, kts, jts-1) = phb(its, kts, jts)
    u_2(its-1, kts, jts-1) = u_2(its, kts, jts)
    v_2(its-1, kts, jts-1) = v_2(its, kts, jts)
    w_2(its-1, kts, jts-1) = w_2(its, kts, jts)
ENDIF

IF (its == ids .and. jte==jde) THEN
    phb(its-1, kts, jte) = phb(its, kts, jte-1)
    u_2(its-1, kts, jte) = u_2(its, kts, jte-1)
    v_2(its-1, kts, jte+1) = v_2(its, kts, jte)
    w_2(its-1, kts, jte) = w_2(its, kts, jte-1)
ENDIF

IF (ite == ide .and. jts==jds) THEN
    phb(ite, kts, jts-1) = phb(ite-1, kts, jts)
    u_2(ite+1, kts, jts-1) = u_2(ite, kts, jts)
    v_2(ite, kts, jts-1) = v_2(ite-1, kts, jts)
    w_2(ite, kts, jts-1) = w_2(ite-1, kts, jts)
ENDIF

IF (ite == ide .and. jte==jde) THEN
    phb(ite, kts, jte) = phb(ite-1, kts, jte-1)
    u_2(ite+1, kts, jte) = u_2(ite, kts, jte-1)
    v_2(ite, kts, jte+1) = v_2(ite-1, kts, jte)
    w_2(ite, kts, jte) = w_2(ite-1, kts, jte-1)
ENDIF

!Find the heights of u & v points
DO j = MAX(jts, jds), MIN(jte, jde-1)
    DO i = MAX(its, ids), MIN(ite, ide)
        z_at_u_surf(i,j)   = ( phb(i-1,kts,j)   + phb(i,kts,j) ) /2 /g
    END DO
END DO

DO j = MAX(jts, jds), MIN(jte, jde)
    DO i = MAX(its, ids), MIN(ite, ide-1)
        z_at_v_surf(i,j)   = ( phb(i,kts,j-1)   + phb(i,kts,j) ) /2 /g
    END DO
END DO

!This is the main calculation part, find the intersection points from
!the surface u points and the corresponding wind fields near the ground
DO j = MAX(jts, jds), MIN(jte, jde-1)
DO i = MAX(its, ids), MIN(ite, ide)

    dx_m = dx / msfux(i,j) !calculate the horizontal grid spacing including mapfactors
    dy_m = dy / msfuy(i,j)

    !A1, z0, h11 .etc are all parameters used to find the intersection points
    A1 = ( phb(i-1,kts,j) - phb(i,kts,j) ) / dx_m / g  !mapfactor

    A2 = ( phb(i-1,kts,j-1) + phb(i,kts,j-1)      &
              -   phb(i-1,kts,j+1) - phb(i,kts,j+1) ) / &
                  4 / dy_m / g  !mapfactor
```

```
z0 = 0
x1 = -1 * dx_m / 2  !mapfacotr
x2 = dx_m / 2  !mapfactor
y1 = 0

h11 = (phb(i-1,kts,j) + phb(i-1,kts+1,j) + ph(i-1,kts+1,j)) / 2 / g &
    -  z_at_u_surf(i,j)
h21 = (phb(i,kts,j) + phb(i,kts+1,j) + ph(i,kts+1,j)) / 2 / g &
    -  z_at_u_surf(i,j)

IF ( A2 >= 0 ) THEN
     y2 = dy_m !mapfactor
     j_dir = 1
     h12 = (phb(i-1,kts,j+1) + phb(i-1,kts+1,j+1) + ph(i-1,kts+1,j+1)) / 2 / g &
    -  z_at_u_surf(i,j)
     h22 = (phb(i,kts,j+1) + phb(i,kts+1,j+1) + ph(i,kts+1,j+1)) / 2 / g &
    -  z_at_u_surf(i,j)
ELSE
     y2 = -1 * dy_m !mapfactor
     j_dir = -1
     h12 = (phb(i-1,kts,j-1) + phb(i-1,kts+1,j-1) + ph(i-1,kts+1,j-1)) / 2 / g &
    -  z_at_u_surf(i,j)
     h22 = (phb(i,kts,j-1) + phb(i,kts+1,j-1) + ph(i,kts+1,j-1)) / 2 / g &
    -  z_at_u_surf(i,j)
ENDIF

IF ( A1 >= 0) THEN
     i_dir = 1
ELSE
     i_dir = -1
ENDIF

AA = A1*( h22 - h12 - h21 + h11)*A2
BB = ((z0 - y1)*A1 + (z0 - x1)*A2) * h22 + &
     ((y1 - z0)*A1 + (x2 - z0)*A2) * h12 + &
     ((y2 - z0)*A1 + (x1 - z0)*A2) * h21 + &
     ((z0 - y2)*A1 + (z0 - x2)*A2) * h11 - &
      (y2 - y1) * (x2 - x1)

CC = (z0 - x1)*(z0 - y1) * h22 + (x2 - z0)*(z0 - y1) * h12 + &
     (z0 - x1)*(y2 - z0) * h21 + (x2 - z0)*(y2 - z0) * h11

!A resonable solver for the height of intersection point should be above
!the lowest height of nearby 4 points & above the highest one
h_max = MAX(h11, (h12+h11)/2, h21, (h22+h21)/2)
h_min = MIN(h11, (h12+h11)/2, h21, (h22+h21)/2)

iF &
   ( ABS(AA) <= 0.0001) THEN
     delta_h_u(i,j)  =  -1*CC / BB
ELSE IF &
   ( MAX(0.,h_min) < (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
     h_max >= (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
     delta_h_u(i,j)  =  (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA
```

```
ELSE IF &
    ( MAX(0.,h_min) < (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
      h_max >= (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
      delta_h_u(i,j)  =  (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA

ELSE
      delta_h_u(i,j)  =  -999
ENDIF

delta_x_u(i,j) = A1 * delta_h_u(i,j)
delta_y_u(i,j) = A2 * delta_h_u(i,j)
!We have delta_x, delta_y and delta_h to calculate the distance from the surface
!u point to the intersection point
dsu(i,j)   = SQRT(delta_h_u(i,j)**2 + delta_x_u(i,j)**2 + delta_y_u(i,j)**2)

!We repeat the progress, but to find the intersection points at the first full level
!above the ground instead of the first half-level
h11 = (phb(i-1,kts+1,j) + ph(i-1,kts+1,j)) / g &
    -  z_at_u_surf(i,j)
h21 = (phb(i,kts+1,j) + ph(i,kts+1,j)) / g &
    -  z_at_u_surf(i,j)

IF ( A2 >= 0 ) THEN
      h12 = (phb(i-1,kts+1,j+1) + ph(i-1,kts+1,j+1)) / g &
    -  z_at_u_surf(i,j)
      h22 = (phb(i,kts+1,j+1) + ph(i,kts+1,j+1)) / g &
    -  z_at_u_surf(i,j)
ELSE
      h12 = (phb(i-1,kts+1,j-1) + ph(i-1,kts+1,j-1)) / g &
    -  z_at_u_surf(i,j)
      h22 = (phb(i,kts+1,j-1) + ph(i,kts+1,j-1)) / g &
    -  z_at_u_surf(i,j)
ENDIF

AA = A1*( h22 - h12 - h21 + h11)*A2
BB = ((z0 - y1)*A1 + (z0 - x1)*A2) * h22 + &
      ((y1 - z0)*A1 + (x2 - z0)*A2) * h12 + &
      ((y2 - z0)*A1 + (x1 - z0)*A2) * h21 + &
      ((z0 - y2)*A1 + (z0 - x2)*A2) * h11 - &
       (y2 - y1) * (x2 - x1)

CC = (z0 - x1)*(z0 - y1) * h22 + (x2 - z0)*(z0 - y1) * h12 + &
      (z0 - x1)*(y2 - z0) * h21 + (x2 - z0)*(y2 - z0) * h11

h_max = MAX(h11, (h12+h11)/2, h21, (h22+h21)/2)
h_min = MIN(h11, (h12+h11)/2, h21, (h22+h21)/2)

iF &
    ( ABS(AA) <= 0.0001) THEN
      delta_h_u_f(i,j)  =  -1*CC / BB
ELSE IF &
    ( MAX(0.,h_min) < (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
      h_max >= (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
      delta_h_u_f(i,j)  =  (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA
ELSE IF &
    ( MAX(0.,h_min) < (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
    h_max >= (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
    delta_h_u_f(i,j)  =  (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA



ELSE
    delta_h_u_f(i,j)  =  -999
ENDIF

delta_x_u_f(i,j) = A1 * delta_h_u_f(i,j)
delta_y_u_f(i,j) = A2 * delta_h_u_f(i,j)

A1 = -A1
A2 = -A2
!Find the wind fields at the intersection points by bilinear interpolation
u_norm_u(i,j) =(ABS(delta_x_u(i,j)             * delta_y_u(i,j)            ) * u_2(i+i_dir,kts,j+j_dir) &
              + ABS((i_dir*dx_m-delta_x_u(i,j)) * delta_y_u(i,j)            ) * u_2(i       ,kts,j+j_dir) &
              + ABS(delta_x_u(i,j)             * (j_dir*dy_m-delta_y_u(i,j))) * u_2(i+i_dir,kts,j       ) &
              + ABS((i_dir*dx_m-delta_x_u(i,j)) * (j_dir*dy_m-delta_y_u(i,j))) * u_2(i       ,kts,j       ))& !mapfactor
              / dx_m / dy_m
v_norm_u(i,j) =((dx_m/2 + delta_x_u(i,j)) * (dy_m/2 + delta_y_u(i,j)) * v_2(i,kts,j+1)   &
              + (dx_m/2 - delta_x_u(i,j)) * (dy_m/2 + delta_y_u(i,j)) * v_2(i-1,kts,j+1) &
              + (dx_m/2 + delta_x_u(i,j)) * (dy_m/2 - delta_y_u(i,j)) * v_2(i,kts,j)     &
              + (dx_m/2 - delta_x_u(i,j)) * (dy_m/2 - delta_y_u(i,j)) * v_2(i-1,kts,j)  )&
              / dx_m / dy_m
w_norm_u(i,j) =(ABS((dx_m/2 + delta_x_u_f(i,j)) * delta_y_u_f(i,j))              * w_2(i,kts+1,j+j_dir)  &
              + ABS((dx_m/2 - delta_x_u_f(i,j)) * delta_y_u_f(i,j))              * w_2(i-1,kts+1,j+j_dir)&
              + ABS((dx_m/2 + delta_x_u_f(i,j)) * (j_dir*dy_m-delta_y_u_f(i,j))) * w_2(i,kts+1,j)        &
              + ABS((dx_m/2 - delta_x_u_f(i,j)) * (j_dir*dy_m-delta_y_u_f(i,j))) * w_2(i-1,kts+1,j)     )&
              / dx_m / dy_m
!Define the drags in two tangential directions
u_drag = u_2(i,kts,j)
v_drag = (v_2(i   ,kts,j   ) + v_2(i   ,kts,j+1) + v_2(i-1,kts,j   ) + v_2(i-1,kts,j+1))/4
w_drag = u_drag * A1 + v_drag * A2

n1_u(i,j) = - A1 / SQRT(A1**2 + A2**2 + 1)
n2_u(i,j) = - A2 / SQRT(A1**2 + A2**2 + 1)
n3_u(i,j) = 1. / SQRT(A1**2 + A2**2 + 1)

V0_u       = SQRT(u_drag**2 + v_drag**2 + w_drag**2 - &
                 (u_drag * n1_u(i,j) + v_drag * n2_u(i,j) + w_drag * n3_u(i,j)) ** 2) + epsln
D_x_u(i,j) =-(( ust(i-1,j) + ust(i,j) ) / 2 ) ** 2 * (rho(i,kts,j)+rho(i-1,kts,j)) / 2 * &
             ( u_drag - w_drag * A1 ) / SQRT(A1**2 + 1) / V0_u
D_y_u(i,j) =-(( ust(i-1,j) + ust(i,j) ) / 2 ) ** 2 * (rho(i,kts,j)+rho(i-1,kts,j)) / 2 * &
             ( v_drag - w_drag * A2 ) / SQRT(A2**2 + 1) / V0_u

!Full stress 2d test
!IF (i > ids+1 .AND. i < ide-1) THEN
!Full stress 3d test
!We need the slopes and curvatures in different directions to apply local full-stress condition
IF (i > ids+1 .AND. i < ide-1 .AND. j > jds+1 .AND. j < jde -1) THEN
        d2hdx2 = 0.5/dx_m/dx_m/g * (phb(i+1,kts,j) + phb(i-2,kts,j) - phb(i,kts,j) - phb(i-1,kts,j))
        d2hdy2 = 0.5/dy_m/dy_m/g * (phb(i,kts,j+1) + phb(i-1,kts,j+1) + phb(i,kts,j-1) + phb(i-1,kts,j-1) - &
                 2 * phb(i-1,kts,j) - 2 * phb(i,kts,j))
        d2hdxy = 0.5/dx_m/dy_m/g * (phb(i,kts,j+1) - phb(i-1,kts,j+1) - phb(i,kts,j-1) + phb(i-1,kts,j-1))
ELSE
        d2hdx2 = 0.
```

```
          d2hdy2 = 0.
          d2hdxy = 0.
    ENDIF



    !aa1 .etc are the parameters to calculate wind fields near the ground
    aa1        = delta_h_u(i,j) / delta_h_u_f(i,j) * n1_u(i,j) * A1 - n3_u(i,j) + n3_u(i,j)**2 * d2hdx2 * dsu(i,j)
    aa2        = delta_h_u(i,j) / delta_h_u_f(i,j) * n2_u(i,j) * A1 + n3_u(i,j)**2 * d2hdxy * dsu(i,j)
    aa3        = delta_h_u(i,j) / delta_h_u_f(i,j) * n1_u(i,j) * A2 + n3_u(i,j)**2 * d2hdxy * dsu(i,j)
    aa4        = delta_h_u(i,j) / delta_h_u_f(i,j) * n2_u(i,j) * A2 - n3_u(i,j) + n3_u(i,j)**2 * d2hdy2 * dsu(i,j)
    aa5        = -D_x_u(i,j) * SQRT(n1_u(i,j)**2 + n3_u(i,j)**2) * dsu(i,j) / (xkmh(i,kts,j) + xkmh(i-1,kts,j)) * 2 - &
                   n3_u(i,j) * u_norm_u(i,j) + n1_u(i,j) * delta_h_u(i,j) / delta_h_u_f(i,j) * w_norm_u(i,j)
    aa6        = -D_y_u(i,j) * SQRT(n2_u(i,j)**2 + n3_u(i,j)**2) * dsu(i,j) / (xkmh(i,kts,j) + xkmh(i-1,kts,j)) * 2 - &
                   n3_u(i,j) * v_norm_u(i,j) + n2_u(i,j) * delta_h_u(i,j) / delta_h_u_f(i,j) * w_norm_u(i,j)

    u_sfc_u(i,j) = (aa4 * aa5 - aa3 * aa6) / (aa1 * aa4 - aa2 * aa3)
    v_sfc_u(i,j) = (aa1 * aa6 - aa2 * aa5) / (aa1 * aa4 - aa2 * aa3)
    w_sfc_u(i,j) = u_sfc_u(i,j) * A1

END DO
END DO

!Same as u points, for v points
DO j = MAX(jts, jds), MIN(jte, jde)
DO i = MAX(its, ids), MIN(ite, ide-1)

    dx_m = dx / msfvx(i,j)
    dy_m = dy / msfvy(i,j)

    A1 = ( phb(i-1,kts,j-1) + phb(i-1,kts,j)      &
              -   phb(i+1,kts,j-1) - phb(i+1,kts,j) ) / &
                   4 / dx_m / g  !mapfactor

    A2 = ( phb(i,kts,j-1) - phb(i,kts,j) ) / dy_m / g !mapfactor

    z0 = 0
    x1 = 0
    y1 = -1 * dy_m / 2
    y2 = dy_m / 2

    h11 = (phb(i,kts,j-1) + phb(i,kts+1,j-1) + ph(i,kts+1,j-1)) / 2 / g &
        -  z_at_v_surf(i,j)
    h12 = (phb(i,kts,j) + phb(i,kts+1,j) + ph(i,kts+1,j)) / 2 / g &
        -  z_at_v_surf(i,j)

    IF ( A1 >= 0 ) THEN
         x2 = dx_m !mapfactor
         i_dir = 1
         h21 = (phb(i+1,kts,j-1) + phb(i+1,kts+1,j-1) + ph(i+1,kts+1,j-1)) / 2 / g &
        -  z_at_v_surf(i,j)
         h22 = (phb(i+1,kts,j) + phb(i+1,kts+1,j) + ph(i+1,kts+1,j)) / 2 / g &
        -  z_at_v_surf(i,j)
    ELSE
         x2 = -1 * dx_m !mapfactor
         i_dir = -1
         h21 = (phb(i-1,kts,j-1) + phb(i-1,kts+1,j-1) + ph(i-1,kts+1,j-1)) / 2 / g &
```

```
      -  z_at_v_surf(i,j)
       h22 = (phb(i-1,kts,j) + phb(i-1,kts+1,j) + ph(i-1,kts+1,j)) / 2 / g &
      -  z_at_v_surf(i,j)
ENDIF

IF ( A2 >= 0) THEN
       j_dir = 1
ELSE
       j_dir = -1
ENDIF

AA = A1*( h22 - h12 - h21 + h11)*A2
BB = ((z0 - y1)*A1 + (z0 - x1)*A2) * h22 + &
       ((y1 - z0)*A1 + (x2 - z0)*A2) * h12 + &
       ((y2 - z0)*A1 + (x1 - z0)*A2) * h21 + &
       ((z0 - y2)*A1 + (z0 - x2)*A2) * h11 - &
        (y2 - y1) * (x2 - x1)

CC = (z0 - x1)*(z0 - y1) * h22 + (x2 - z0)*(z0 - y1) * h12 + &
       (z0 - x1)*(y2 - z0) * h21 + (x2 - z0)*(y2 - z0) * h11

h_max = MAX(h11, h12, (h21+h11)/2, (h22+h12)/2)
h_min = MIN(h11, h12, (h21+h11)/2, (h22+h12)/2)

iF &
    ( ABS(AA) <= 0.0001) THEN
       delta_h_v(i,j)  =  -1*CC / BB
ELSE IF &
    ( MAX(0.,h_min) < (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
       h_max >= (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
       delta_h_v(i,j)  =  (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA
ELSE IF &
    ( MAX(0.,h_min) < (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
       h_max >= (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
       delta_h_v(i,j)  =  (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA

ELSE
       delta_h_v(i,j)  =  -999
ENDIF

delta_x_v(i,j) = A1 * delta_h_v(i,j)
delta_y_v(i,j) = A2 * delta_h_v(i,j)
dsv(i,j)    = SQRT(delta_h_v(i,j)**2 + delta_x_v(i,j)**2 + delta_y_v(i,j)**2)

h11 = (phb(i,kts+1,j-1) + ph(i,kts+1,j-1)) / g &
     -  z_at_v_surf(i,j)
h12 = (phb(i,kts+1,j) + ph(i,kts+1,j)) / g &
     -  z_at_v_surf(i,j)

IF ( A1 >= 0 ) THEN
       h21 = (phb(i+1,kts+1,j-1) + ph(i+1,kts+1,j-1)) / g &
      -  z_at_v_surf(i,j)
       h22 = (phb(i+1,kts+1,j) + ph(i+1,kts+1,j)) / g &
      -  z_at_v_surf(i,j)
ELSE
       h21 = (phb(i-1,kts+1,j-1) + ph(i-1,kts+1,j-1)) / g &
     -  z_at_v_surf(i,j)
      h22 = (phb(i-1,kts+1,j) + ph(i-1,kts+1,j)) / g &
     -  z_at_v_surf(i,j)
ENDIF


AA = A1*( h22 - h12 - h21 + h11)*A2
BB = ((z0 - y1)*A1 + (z0 - x1)*A2) * h22 + &
      ((y1 - z0)*A1 + (x2 - z0)*A2) * h12 + &
      ((y2 - z0)*A1 + (x1 - z0)*A2) * h21 + &
      ((z0 - y2)*A1 + (z0 - x2)*A2) * h11 - &
       (y2 - y1) * (x2 - x1)

CC = (z0 - x1)*(z0 - y1) * h22 + (x2 - z0)*(z0 - y1) * h12 + &
      (z0 - x1)*(y2 - z0) * h21 + (x2 - z0)*(y2 - z0) * h11

h_max  = MAX(h11, h12, (h21+h11)/2, (h22+h12)/2)
h_min  = MIN(h11, h12, (h21+h11)/2, (h22+h12)/2)

iF &
    ( ABS(AA) <= 0.0001) THEN
      delta_h_v_f(i,j)  =  -1*CC / BB
ELSE IF &
    ( MAX(0.,h_min) < (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
      h_max >= (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
      delta_h_v_f(i,j)  =  (-1*BB + SQRT(BB*BB - 4 * AA * CC))/2/AA
ELSE IF &
    ( MAX(0.,h_min) < (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA .AND. &
      h_max >= (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA) THEN
      delta_h_v_f(i,j)  =  (-1*BB - SQRT(BB*BB - 4 * AA * CC))/2/AA

ELSE
      delta_h_v_f(i,j)  =  -999
ENDIF

delta_x_v_f(i,j) = A1 * delta_h_v_f(i,j)
delta_y_v_f(i,j) = A2 * delta_h_v_f(i,j)

A1 = -A1
A2 = -A2

v_norm_v(i,j) =(ABS(delta_x_v(i,j)           * delta_y_v(i,j)            ) * v_2(i+i_dir,kts,j+j_dir) & !mapfactor
              + ABS((i_dir*dx_m-delta_x_v(i,j)) * delta_y_v(i,j)            ) * v_2(i      ,kts,j+j_dir) &
              + ABS(delta_x_v(i,j)           * (j_dir*dy_m-delta_y_v(i,j))) * v_2(i+i_dir,kts,j      ) &
              + ABS((i_dir*dx_m-delta_x_v(i,j)) * (j_dir*dy_m-delta_y_v(i,j))) * v_2(i      ,kts,j      ))&
              / dx_m / dy_m
u_norm_v(i,j) =((dx_m/2 + delta_x_v(i,j)) * (dy_m/2 + delta_y_v(i,j)) * u_2(i+1,kts,j)   &
              + (dx_m/2 - delta_x_v(i,j)) * (dy_m/2 + delta_y_v(i,j)) * u_2(i,kts,j) &
              + (dx_m/2 + delta_x_v(i,j)) * (dy_m/2 - delta_y_v(i,j)) * u_2(i+1,kts,j-1)     &
              + (dx_m/2 - delta_x_v(i,j)) * (dy_m/2 - delta_y_v(i,j)) * u_2(i,kts,j-1)  )&
              / dx_m / dy_m
w_norm_v(i,j) =(ABS((dy_m/2 + delta_y_v_f(i,j)) * delta_x_v_f(i,j))            * w_2(i+i_dir,kts+1,j)  &
              + ABS((dy_m/2 - delta_y_v_f(i,j)) * delta_x_v_f(i,j))            * w_2(i+i_dir,kts+1,j-1)&
              + ABS((dy_m/2 + delta_y_v_f(i,j)) * (i_dir*dx_m-delta_x_v_f(i,j))) * w_2(i,kts+1,j)         &
              + ABS((dy_m/2 - delta_y_v_f(i,j)) * (i_dir*dx_m-delta_x_v_f(i,j))) * w_2(i,kts+1,j-1)     )&
              / dx_m / dy_m
```

```
        u_drag = (u_2(i   ,kts,j   ) + u_2(i+1,kts,j   ) + u_2(i   ,kts,j-1) + u_2(i+1,kts,j-1))/4
        v_drag = v_2(i,kts,j)
        w_drag = u_drag * A1 + v_drag * A2

        n1_v(i,j) = - A1 / SQRT(A1**2 + A2**2 + 1)
        n2_v(i,j) = - A2 / SQRT(A1**2 + A2**2 + 1)
        n3_v(i,j) = 1. / SQRT(A1**2 + A2**2 + 1)

        V0_v       = SQRT(u_drag**2 + v_drag**2 + w_drag**2 - &
                          (u_drag * n1_v(i,j) + v_drag * n2_v(i,j) + w_drag * n3_v(i,j)) ** 2 ) + epsln
        D_y_v(i,j) =-(( ust(i,j-1) + ust(i,j) ) / 2 ) ** 2 * (rho(i,kts,j)+rho(i,kts,j-1)) / 2 * &
                     ( v_drag - w_drag * A2 ) / SQRT(A2**2 + 1) / V0_v
        D_x_v(i,j) =-(( ust(i,j-1) + ust(i,j) ) / 2 ) ** 2 * (rho(i,kts,j)+rho(i,kts,j-1)) / 2 * &
                     ( u_drag - w_drag * A1 ) / SQRT(A1**2 + 1) / V0_v

        IF (i > ids+1 .AND. i < ide-1 .AND. j > jds+1 .AND. j < jde -1) THEN
                d2hdx2 = 0.5/dx_m/dx_m/g * (phb(i+1,kts,j) + phb(i+1,kts,j-1) + phb(i-1,kts,j) + phb(i-1,kts,j-1) - &
                         2 * phb(i,kts,j) - 2 * phb(i,kts,j-1))
                d2hdy2 = 0.5/dy_m/dy_m/g * (phb(i,kts,j+1) + phb(i,kts,j-2) - phb(i,kts,j) - phb(i,kts,j-1))
                d2hdxy = 0.5/dx_m/dy_m/g * (phb(i+1,kts,j) - phb(i+1,kts,j-1) - phb(i-1,kts,j) + phb(i-1,kts,j-1))
        ELSE
                d2hdx2 = 0.
                d2hdy2 = 0.
                d2hdxy = 0.
        ENDIF

        aa1        = delta_h_v(i,j) / delta_h_v_f(i,j) * n1_v(i,j) * A1 - n3_v(i,j) + n3_v(i,j)**2 * d2hdx2 * dsv(i,j)
        aa2        = delta_h_v(i,j) / delta_h_v_f(i,j) * n2_v(i,j) * A1 + n3_v(i,j)**2 * d2hdxy * dsv(i,j)
        aa3        = delta_h_v(i,j) / delta_h_v_f(i,j) * n1_v(i,j) * A2 + n3_v(i,j)**2 * d2hdxy * dsv(i,j)
        aa4        = delta_h_v(i,j) / delta_h_v_f(i,j) * n2_v(i,j) * A2 - n3_v(i,j) + n3_v(i,j)**2 * d2hdy2 * dsv(i,j)

        aa5        = -D_x_v(i,j) * SQRT(n1_v(i,j)**2 + n3_v(i,j)**2) * dsv(i,j) / (xkmh(i,kts,j) + xkmh(i-1,kts,j)) * 2 - &
                      n3_v(i,j) * u_norm_v(i,j) + n1_v(i,j) * delta_h_v(i,j) / delta_h_v_f(i,j) * w_norm_v(i,j)
        aa6        = -D_y_v(i,j) * SQRT(n2_v(i,j)**2 + n3_v(i,j)**2) * dsu(i,j) / (xkmh(i,kts,j) + xkmh(i-1,kts,j)) * 2 - &
                      n3_v(i,j) * v_norm_v(i,j) + n2_v(i,j) * delta_h_v(i,j) / delta_h_v_f(i,j) * w_norm_v(i,j)


        u_sfc_v(i,j) = (aa4 * aa5 - aa3 * aa6) / (aa1 * aa4 - aa2 * aa3)
        v_sfc_v(i,j) = (aa1 * aa6 - aa2 * aa5) / (aa1 * aa4 - aa2 * aa3)
        w_sfc_v(i,j) = v_sfc_v(i,j) * A2

    END DO
    END DO

    !Again, we need to be carful of the values at the boundaries
    IF (its == ids) THEN
      DO j = jts, MIN( jte, jde-1)
        w_sfc_u(its-1,j) = w_sfc_u(its,j)
        w_sfc_v(its-1,j) = w_sfc_v(its,j)
        v_sfc_v(its-1,j) = v_sfc_v(its,j)
      ENDDO
        IF (jte == jde) THEN
          w_sfc_v(its-1,jte) = w_sfc_v(its,jte)
        ENDIF
    ENDIF
```

```
IF (ite == ide) THEN
  DO j = jts, MIN( jte, jde-1)
     w_sfc_u(ite+1,j) = w_sfc_u(ite,j)
     w_sfc_v(ite,j) = w_sfc_v(ite-1,j)
  ENDDO
     IF (jte == jde) THEN
        w_sfc_v(ite,jte) = w_sfc_v(ite-1,jte)
     ENDIF
ENDIF

IF (jts == jds) THEN
  DO i = its, MIN( ite, ide-1)
     w_sfc_v(i,jts-1) = w_sfc_v(i,jts)
     w_sfc_u(i,jts-1) = w_sfc_u(i,jts)
     u_sfc_u(i,jts-1) = u_sfc_u(i,jts)
  ENDDO
     IF (ite == ide) THEN
        w_sfc_u(ite,jts-1) = w_sfc_u(ite,jts)
     ENDIF
ENDIF

IF (jte == jde) THEN
  DO i = its, MIN( jte, jde-1)
     w_sfc_v(i,jte+1) = w_sfc_v(i,jte)
     w_sfc_u(i,jte) = w_sfc_u(i,jte-1)
  ENDDO
     IF (ite == ide) THEN
        w_sfc_u(ite,jte) = w_sfc_u(ite,jte-1)
     ENDIF
ENDIF

END SUBROUTINE get_norm_intersection
```

A.3 cal_deform_and_div

We modified the computation of deformations at the ground, so that we could make use the wind fields near the ground we calculated above.

```
! norm_stress is the parameter we created to apply norm-grad or full-stress condition
IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
        hatavg(i,1,j) = ( u_sfc_u(i,j) / msfuy(i,j) + u_sfc_u(i+1,j) / msfuy(i+1,j) ) /2
    END DO
    END DO
END IF

! define the new array: defor11_sfc
IF (config_flags%norm_stress == 1) THEN
  DO j = j_start, j_end
  DO i = i_start, i_end
     defor11_sfc(i,j) = 2. * mm(i,j) * ( rdx * (u_sfc_u(i+1,j)/msfuy(i+1,j) - u_sfc_u(i,j)/msfuy(i,j)) - &
                  ((hat(i,kts,j) + hat(i+1,kts,j)) / 2. - hatavg(i,1,j)) * &
                  (zx(i,kts   ,j) + zx(i+1,kts   ,j)) * rdzw(i,kts,j) ) !stretch
  END DO
  END DO
ENDIF

IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
        hatavg(i,1,j) = ( v_sfc_v(i,j) / msfvx(i,j) + v_sfc_v(i,j+1) / msfvx(i,j+1) ) / 2.
    END DO
    END DO
END IF

! define the new array: defor22_sfc
IF (config_flags%norm_stress == 1) THEN
  DO j = j_start, j_end
  DO i = i_start, i_end
     defor22_sfc(i,j) = 2. * mm(i,j) * ( rdy * (v_sfc_v(i,j+1)/msfvx(i,j+1) - v_sfc_v(i,j)/msfvx(i,j)) - &
                     ((hat(i,kts,j) + hat(i,kts,j+1) ) / 2. - hatavg(i,1,j)) * &
                     (zy(i,kts   ,j) + zy(i,kts   ,j+1)) * rdzw(i,kts,j) )
  END DO
  END DO
ENDIF

IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
        tmp1(i,kts,j) = ( w(i,kts+1,j) - (w_sfc_u(i,j)+w_sfc_u(i+1,j)+w_sfc_v(i,j)+w_sfc_v(i,j+1))/2. ) * rdzw(i,kts,j)
    END DO
    END DO
END IF

! define the new array: defor33_sfc
IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
        defor33_sfc(i,j) = defor33(i,kts,j)
    END DO
    END DO
END IF

IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
        hatavg(i,1,j) = (u_sfc_u(i,j) / msfux(i,j) + u_sfc_u(i,j-1) / msfux(i,j-1) ) / 2
    END DO
    END DO
END IF

! first term of defor12_sfc
IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
       defor12_sfc(i,j) = mm(i,j) * ( rdy * (u_sfc_u(i,j)/msfux(i,j) - u_sfc_u(i,j-1)/msfux(i,j-1)) - &
                     ((hat(i,kts,j) + hat(i,kts,j-1) ) / 2. - hatavg(i,1,j)) * &
                     (zy(i,kts   ,j) + zy(i-1,kts   ,j))  * 0.25 * &
                     (rdzw(i,kts,j) + rdzw(i-1,kts,j) + rdzw(i,kts,j-1) + rdzw(i-1,kts,j-1)) )
    END DO
    END DO
END IF

IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
        hatavg(i,1,j) = (v_sfc_v(i,j) / msfvy(i,j) + v_sfc_v(i-1,j) / msfvy(i-1,j)) / 2.
    END DO
    END DO
END IF

! second term of defor12_sfc
IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end
      defor12_sfc(i,j) = defor12_sfc(i,j) + &
                       mm(i,j) * ( rdx * (v_sfc_v(i,j)/msfvy(i,j) - v_sfc_v(i-1,j)/msfvy(i-1,j)) - &
                       ((hat(i,kts,j) + hat(i-1,kts,j) ) / 2. - hatavg(i,1,j)) * &
                       (zx(i,kts  ,j) + zx(i,kts  ,j-1))  * 0.25 * &
                       (rdzw(i,kts,j) + rdzw(i-1,kts,j) + rdzw(i,kts,j-1) + rdzw(i-1,kts,j-1)) )
    END DO
    END DO
END IF

! change the first layer defor13
IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end
    DO i = i_start, i_end+1 !important change by YiLi 11/2019
      defor13(i,kts,j  ) = mm(i,j) * ( rdx * (hat(i,kts,j) / msfty(i+1,j) - hat(i-1,kts,j) / msfuy(i-1,j)) -   &
                              (hatavg(i,kts,j) - (4*w_sfc_u(i,j)+w_sfc_v(i,j)+w_sfc_v(i,j+1)+w_sfc_v(i-1,j)+w_sfc_v(i-1,j+1)) / &
                               4.  / msfuy(i,j)) * zx(i,kts,j) * &
                              ( rdzw(i,kts,j) + rdzw(i-1,kts,j) ) ) + &
                              (u(i,kts,j) - u_sfc_u(i,j) ) * ( rdzw(i,kts,j) + rdzw(i-1,kts,j) )
      defor13(i,ktf+1,j) = 0.0
    END DO
    END DO
END IF

! change the first layer defor23
IF (config_flags%norm_stress == 1) THEN
    DO j = j_start, j_end+1 !important change by YiLi 11/2019
    DO i = i_start, i_end
      defor23(i,kts,j  ) = mm(i,j) * ( rdy * (hat(i,kts,j) / msftx(i,j) - hat(i,kts,j-1) / msftx(i,j-1)) -   &
                              (hatavg(i,kts,j) - (4*w_sfc_v(i,j)+w_sfc_u(i,j)+w_sfc_u(i+1,j)+w_sfc_u(i,j-1)+w_sfc_u(i+1,j-1)) / &
                               4.  / msfvx(i,j)) * zy(i,kts,j) * &
                              ( rdzw(i,kts,j) + rdzw(i,kts,j-1) ) ) + &
                              (v(i,kts,j) - v_sfc_v(i,j) ) * ( rdzw(i,kts,j) + rdzw(i,kts,j-1) )
      defor23(i,ktf+1,j) = 0.0
    END DO
    END DO
END IF
```