

Modelling of Shallow Water Equations and the Weather Research and Forecasting Model (WRF)

by

Yash Bhatt

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Marek Stastna

Waterloo, Ontario, Canada, 2011

© Yash Bhatt 2011

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

The goal of this report is to analyze various computational techniques applied to model physical phenomena in the area of oceanography and atmospheric sciences. Cases are discussed, wherein, mathematical models are developed using the programming language MATLAB followed by implementing NCAR's (National Center for Atmospheric Research) WRF (Weather Research and Forecasting) model consisting of built in code and physics modeling capabilities. Before numerical modelling is carried out, various problems of interest, issues and generalized models regarding dynamics of coastal basins are put forth. Shallow water equations (SWE) form a reasonable starting point as far as modelling of physical, biological and chemical processes in lakes and coastal basins are concerned. Finite difference and Fourier methods along with various numerical schemes implemented to model SWE's are compared and analyzed. A small discussion on the implementation of an advantageous MATLAB function **profile** is provided. Producing results of idealized cases within the WRF model is carried out to test the modelling capabilities, especially, in light of capturing the physics. Studying the boundary layer parametrizations of the WRF model along with a brief description of the Large-Eddy Simulation (LES) configuration is discussed.

Acknowledgements

The author would like to take this opportunity to express deep gratitude to his supervisor, Dr. Marek Stastna, for his immense support and guidance. Without his enthusiasm and encouragement, this research would not have realized its full potential. Classmates and colleagues at the Environmental and Geophysical Fluid Dynamics laboratory have been extremely helpful and instrumental in sharing their knowledge and experiences. In particular, the author would like to thank Nancy Soontiens, for introducing him to important features of the Weather Research and Forecasting (WRF) model and Michael Dunphy for his insightful advice on the Matlab profiler and L^AT_EX. The section on dispersive shallow water equations would not have realized fruition without the help of Derek Steinmoeller. Friends and family have always been the back bone of this short journey and the author expresses his gratefulness to all of them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction & Background

The motivation of carrying out this research and constructing this report is to explore mathematical models, created by, using the Mathworks language MATLAB and government/industry codes, that provide insight into the understanding of dynamics of coastal basins and environmental flows. Today, along with experiment and theory, numerical modelling is the best tool available to humankind, to understand the dynamics of environmental and geophysical flows. Once the governing equations have been chosen, the next decision to be made is which numerical method must be used to best represent the equations. Progression or extension of a single model, discussion of alternative models, making the right choice of a model corresponding to the physics and trade-offs or disadvantages of the model under scrutiny, is addressed. Various numerical methods/schemes applied separately to each model, along with a brief discussion on the distribution of computing jobs to various processors provides information about the computing time and cost. Understanding the capability of government/industry codes that claim to capture various physical phenomena has been the motive behind studying the Weather Research and Forecasting (WRF) model.

To begin, Chapter 1, gives an insight into the various mathematical models of the coastal ocean as discussed in section ???. Idealized models as well as problems of interest are presented in this section. A summary of the hydrodynamics models, modelling tides in coastal basins are discussed in this section. This is followed by a discussion on the shallow water equations (SWE) and its dispersive extensions mentioned in section ???. Finally, a brief introduction to the Weather Research and Forecasting model (WRF) in its large eddy simulation (LES) configuration along with a brief background on the physics schemes available in WRF is provided in section ???.

1.1 Models of the Coastal Ocean

The term coastal basins represent the myriad of different water bodies, that are found in the region between the ocean and the continental shelf. They carry names such as estuary, bay, etc. A coastal basin is a shallow system i.e. typically less than 10m (although possibly some tens of meters deep) forced mainly by wind, tide and river flow. This section provides some background in the understanding and modelling of astronomical tides in coastal basins [?]. The astronomical tides of the ocean are primarily due to changes in the gravitational potential exerted on the large ocean basins by the Moon and Sun as the Earth rotates in their gravitational fields. When waves arrive in the coastal zone, i.e. the shallow continental shelf and coastal basins, they produce shallow water tides, which are quite different in magnitude from the deep ocean tides. Shallow water tides occur on spatial scales that are too small to be directly affected by the gravitational forcing of the Sun and Moon. The tides in the coastal basins are a direct consequence of the change in the water level at their boundary with the deep ocean. Most places in the ocean usually experience two high tides and two low tides each day (semidiurnal tide), but some locations experience only one high and one low tide each day (diurnal tide) [?].

There is a separation between the deep ocean and the coastal region created by change of depth. Tidal waves in the ocean and coastal regions have wavelengths much greater than the water depth and hence, propagate as long waves. Hence, the long gravity wave speed is given by \sqrt{gh} , where g is the acceleration due to gravity and h is the depth of the water column. The notation M is used for tides predominantly due to the motion of the moon and S is used for those due to the sun. The number after the letter is used to denote either the approximate frequency per day i.e. diurnal 1 or semi diurnal 2. For example, the dominant tidal component is denoted by M2. Mathematically, the frequencies of astronomical gravitational forcing are denoted by a set of angular frequencies ω . These are called tidal harmonics [?]. Hence, the elevation η at any point in the ocean will have the form,

$$\eta(t) = \sum_n A_n \cos(\omega_n t + \phi_n) \quad (1.1)$$

where A_n is the amplitude and ϕ_n is the phase angle. Contours of constant phase, ϕ_n are called co-tidal lines for the particular harmonic and represent points at which high and low water occur at the same time. Of special interest is the location of amphidromic points, defined as the point in which the particular harmonic has zero amplitude.

Let, the u velocity be defined along the X-axis, and the v velocity be defined along the Y-axis. If we denote the pressure gradient by F1, stress by F2 and the Coriolis force by

F3, then using the law of conservation of momentum in two dimensions (here the constant density has been absorbed in the F_i terms),

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial v}{\partial y} = F1 + F2 + F3. \quad (1.2)$$

The hydrostatic approximation is a very basic assumption made in almost all models of coastal basin. Its essence is to neglect vertical acceleration for large scale motions. Thus according to this approximation apart from the pressure gradient, the most important force that acts on a particle in the ocean is its weight. The hydrostatic equilibrium assuming the pressure above or atmospheric pressure is zero, is given by,

$$\frac{\partial p}{\partial z} = -g\rho \quad (1.3)$$

where, $p(z)$ is the pressure, g is the acceleration due to gravity and ρ is the density of water, i.e. the gravitational force, $-g\rho dz$, on an element of fluid from z to $z+dz$. This is balanced by the pressure difference $p(z+dz)-p(z)$. The pressure profile that satisfies this is called the hydrostatic pressure. The weight of water per unit area above a height z produces a hydrostatic pressure. For the case of constant density,

$$p = g(\eta - z). \quad (1.4)$$

The non-linear terms, the stress and the Coriolis force are neglected in this research. Using the hydrostatic approximation and the conservation of momentum equation,

$$\frac{\partial u}{\partial t} = -g \frac{\partial \eta}{\partial x}. \quad (1.5)$$

Mass conservation is one of the most important rules used in the development of models of coastal models. The continuity equation in two dimensions is given by,

$$\frac{\partial \eta}{\partial t} = -\frac{\partial hu}{\partial x} - \frac{\partial hv}{\partial y} \quad (1.6)$$

where, $h=H+\eta$ is the sum of the depth and the elevation. Only the one-dimensional form of equation ?? is considered in this study. Adding the frictional term that accounts for drag in the coastal basin, alters the equation for the continuity of momentum. Hence, from [?], the complete set of equations of motion, that are solved in tandem, i.e. continuity of momentum and continuity of volume are given by,

$$\frac{\partial u}{\partial t} = -g \frac{\partial \eta}{\partial x} - C_d |u|u, \quad (1.7)$$

$$\frac{\partial \eta}{\partial t} = -\frac{\partial hu}{\partial x}. \quad (1.8)$$

After discretizing, the above system of equations, can be represented in a matrix vector multiplication format given by, $Ax=B$. For a uniform grid and corresponding set of data points, the matrices A and B can be represented in terms of a differentiation matrix. Differentiation matrices can be represented as tridiagonal or pentadiagonal matrices having an order of accuracy of two and four respectively. An alternate representation of the matrix vector product $Ax=B$, is in the form $w_j = p_j(x_j)$, where, p_j represents a unique polynomial of degree less than or equal to two and x_j represents the grid point. The larger the number of points incorporated in the stencil, higher the accuracy of the solution. Hence, the larger the bandwidth of the matrices, the more accurate the solution is. These convergence rates can be predicted using Taylor series and verified numerically. Consideration of sixth, eighth and higher order schemes will lead to circulant matrices of increasing bandwidth [?]. This method discussed above, is known as the finite difference method. Spectral methods involve, at least in principle, working with a differentiation formula of full order and infinite bandwidth. Trefethen, [?], compares the output between finite difference and spectral methods. The errors in spectral methods decrease very rapidly until high precision is achieved and rounding errors on the computer prevent any further improvement. The governing equations of motion are transformed into their algebraic counterparts in order to carry out computation. The Chebyshev differentiation matrix is used as an operator to carry out the derivative on the operand. Using the leapfrog time scheme,

$$U^{n+1} = U^{n-1} - (2\Delta t)[gD(\eta) + C_d|u|u]^n. \quad (1.9)$$

Similarly, the continuity of volume equation is given by,

$$\eta^{n+1} = \eta^{n-1} - (2\Delta t)D(hu)^n. \quad (1.10)$$

As previously stated, the non-linear terms have been dropped. D is the Chebyshev differentiation matrix and $h = H$ i.e. the height is assumed to be fixed to ignore the non-linear effects and C_d is the drag coefficient. From a generalized perspective, the two physical boundaries of a coastal basin. One is the mouth of the basin, where the water enters the basin from the ocean and the other is the head i.e. where the water in the basin touches the coastline or beach or land. Boundary conditions were imposed on the two parameters viz., the velocity and the elevation. Two MATLAB scripts, that incorporate the velocity and the elevation boundary condition, introduce a periodic forcing on the entire basin. The boundary condition at the mouth is given by, $u = \sin(2\pi t/T)$ m/s. At the dry end/head of the coastal basin a wall boundary condition, with no normal flow, ($u = 0$ m/s) is imposed. Similarly, for the elevation, a periodic boundary condition was imposed at the mouth of the basin. However, at the head, there is no condition that can be imposed on the elevation. The velocity, u is solved for from the continuity of momentum equation.

Throughout this research, MATLAB's high-level commands have been used such as polynomial interpolation, matrix inversion, and FFT. The MATLAB function `cheb`, returns a vector x and a matrix D . This function, adopted from Trefethen [?] is called repeatedly whenever the computation requires the use of Chebyshev grids and differentiation matrices. The MATLAB script that incorporates the velocity boundary condition is provided on the following page. The script that incorporates the elevation boundary conditions is constructed in a similar manner to the one provided.


```

% cheb_tm.m - Tidal Model of a One Dimensional Basin with velocity BC's

% Setting the values of the constants
N=20; g=9.81; H=10;dt=0.01; const1=2*dt*g; const2=2*dt;
bigt=3600; onekm=1e3; L=10*onekm; uamp=0.5; Cd=2.5e-4;
% Implmenting the cheb.m code to calculate the first derivative
[D,x]=cheb(N);
x=x*L; D=(1/L)*D;
% Initializing the vectors
u0=zeros(size(x)); up=u0; un=u0; eta0=zeros(size(x));
t=0; numsteps=1000; numouts=500; etap=eta0; etan=eta0;
figure(1),clf
plot(x,un), axis([-1 1 -1 1])
% Starting the loop
for ii=1:numouts
    for jj=1:numsteps
        t=t+dt;
        uf=up-const1*D*etan-(2*dt*Cd.*abs(un).*(un));
        h=H+etan;
        flux=h.*un;
        etaf=etap-const2*D*flux;
% Boundary Conditions
        uf(1)=0;
        uf(end)=uamp*sin(2*pi*t/bigt);
% Leap frog time stepping + reassigning the vectors
        etaf(end)=etap(end)-const2*D(end,:)*H*un;
        up=un; un=uf;
        etap=etan; etan=etaf;
    end
% Plotting the solution
    subplot(2,1,1),plot(x,un), axis([-L L -2 2]),ylabel('u')
    subplot(2,1,2),plot(x,etan), axis([-L L -2 2]),ylabel('eta')
    drawnow
end

```

The operating conditions specified do not emulate or represent a real basin. However,

Operating parameters	
Acceleration due to gravity (g)	9.81 m/s^2
Number of Points (N)	20
Height (H)	10 m
Length of Basin (L)	10 Km
Co-efficient of Drag (C_d)	$2.5e-4$
Time Period (T)	1 $hour$
Amplitude (A)	0.5

Table 1.1: Operating parameters of the 1D bathtub model

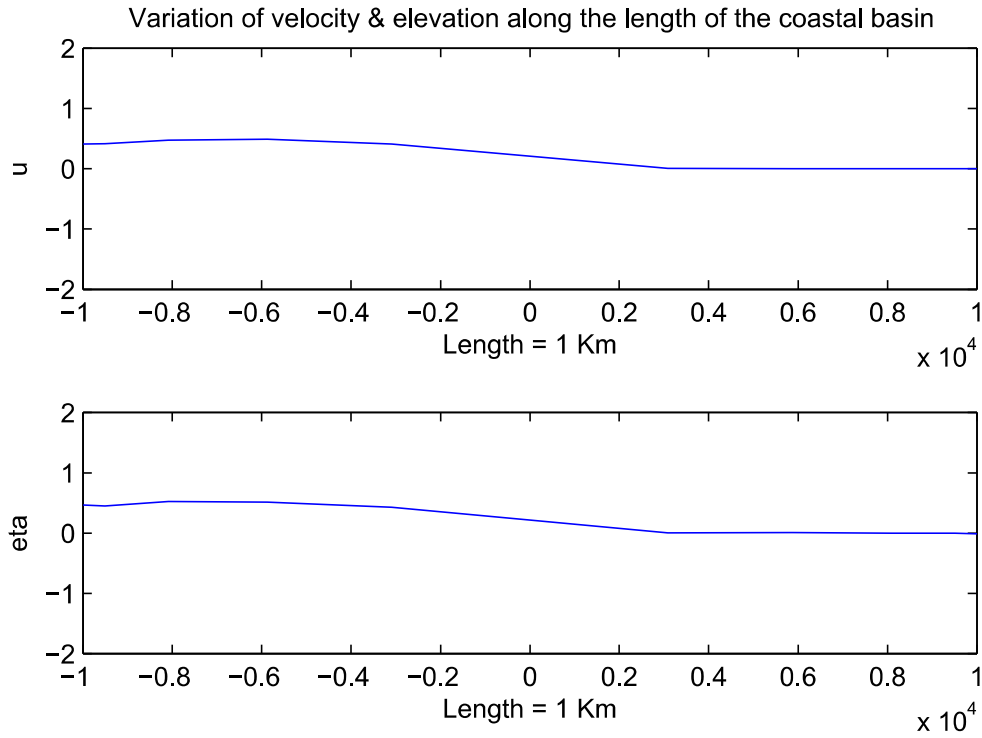


Figure 1.1: Plot showing the variation of velocity & elevation along the length of a basin.

the author has taken approximate values that act as surrogates since there is no exact data of a particular basin that was used in this research. This is because the focus was to test the methods and not to investigate a particular basin.

When using 2D models, one of the disadvantages of using rectangular grids that represent coastal basins is that they do not optimize the speed at which the model is run and also they compromise on the accuracy of the solution. The obvious answer in this case would be implementing a finer mesh or grid that captures the physics well. However, it would not be advisable to reduce the cell size throughout the computational domain. Adopting this approach would lead to an increase in computing time and performing accurate calculations in unimportant regions of the domain that leads to misuse of computing power. There are various ways in which the grid can be altered to serve the specific purpose of the computation. One of them is nesting in which we provide higher resolution in certain parts of the computational domain. For example, we take a continuous rectangular block of 810m cells and introduce nine of 270m cells inside each of the 810m cells in the rectangle. This is called nesting. The non-linear terms, when incorporated in the above specified idealized model, will take into account wave steepening and other effects. This will change the discretized governing equations. For future research, the non-linear terms when included, the continuity of momentum equation, given by, equation ?? is altered and can be written as,

$$LU^{n+1} = LU^{n-1} - 2\Delta t[UU_x + C_d|u|u + g\eta_x]^n \quad (1.11)$$

where,

$$L = I + \frac{H^2}{6}D^2 \quad (1.12)$$

I represents the identity matrix and D represents the Chebyshev differentiation matrix. The corresponding equation for continuity of volume, given by, ??, is further modified,

$$\eta^{n+1} = \eta^{n-1} - (2\Delta t)D(hu)^n \quad (1.13)$$

where, $h = H + \eta$ is not fixed and varies with the length of the basin.

1.2 Shallow Water Equations

The derivation of the shallow water equations has been adopted from the book on Fluid Mechanics by Kundu et al. [?]. Consider a layer of fluid over a flat horizontal bottom.

Let, z , be measured upward from the bottom surface, and η be the displacement from the top surface.

$$p = \rho g(H + \eta - z). \quad (1.14)$$

The horizontal pressure gradients are therefore,

$$\frac{\partial p}{\partial x} = \rho g \frac{\partial \eta}{\partial x} \quad \text{and} \quad \frac{\partial p}{\partial y} = \rho g \frac{\partial \eta}{\partial y}. \quad (1.15)$$

As these are independent of z , the resulting horizontal motion is also depth independent. Consider the continuity equation,

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \quad (1.16)$$

As $\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial y}$ are independent of z , the continuity equation requires that w varies linearly with z , from zero at the bottom to the maximum value at the free surface. Integrating vertically across the water column from $z = 0$ to $z = H + \eta$, and noting that u and v are independent of the depth,

$$(H + \eta) \frac{\partial u}{\partial x} + (H + \eta) \frac{\partial v}{\partial y} + w(\eta) - w(0) = 0, \quad (1.17)$$

where $w(\eta)$ is the vertical velocity at the surface and $w(0)$ is the vertical velocity at the bottom. The surface velocity is given by,

$$w(\eta) = \frac{D\eta}{Dt} = \frac{\partial \eta}{\partial t} + u \frac{\partial \eta}{\partial x} + v \frac{\partial \eta}{\partial y}. \quad (1.18)$$

The continuity equation (??), then becomes,

$$(H + \eta) \frac{\partial u}{\partial x} + (H + \eta) \frac{\partial v}{\partial y} + \frac{\partial \eta}{\partial t} + u \frac{\partial \eta}{\partial x} + v \frac{\partial \eta}{\partial y} = 0, \quad (1.19)$$

which can be written as,

$$\frac{\partial \eta}{\partial t} + \frac{\partial [u(H + \eta)]}{\partial x} + \frac{\partial [v(H + \eta)]}{\partial y} = 0. \quad (1.20)$$

For small amplitude waves, the quadratic nonlinear terms can be neglected in comparison to the linear terms, so that the divergence term in equation (??) simplifies to $H \vec{\nabla} \cdot \mathbf{u}$. The

linearized continuity and momentum equations are given by,

$$\begin{aligned} \frac{\partial \eta}{\partial t} + H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) &= 0, \\ \frac{\partial u}{\partial t} - fv &= -g\frac{\partial \eta}{\partial x}, \\ \frac{\partial v}{\partial t} + fu &= -g\frac{\partial \eta}{\partial y}. \end{aligned} \tag{1.21}$$

In the momentum equations of ??, the pressure gradient terms are written in the form ?? and the nonlinear advective terms have been neglected under the small amplitude assumption. Equations ??, are called the linear *shallow water equations* and they govern the motion of a layer of fluid in which the horizontal scale is much larger than the depth of the layer.

The standard shallow-water equations with the Boussinesq approximation and with the assumption of hydrostatic pressure can be derived for an n-layer stratification by vertically integrating the full equations of motion between $z = z_{i+1}$ to $z = z_i$, where z is the vertical coordinate, and z_{i+1} and z_i correspond to the bottom and upper interfaces that define the i th layer, respectively [?]. The three governing equations for each layer can be written as,

$$\frac{\partial U_i}{\partial t} + \frac{\partial F_i^x}{\partial x} + \frac{\partial F_i^y}{\partial y} = H_i, \tag{1.22}$$

where,

$$U_i = [h_i, u_i h_i, v_i h_i]^T, \tag{1.23}$$

$$F_i^x = \begin{pmatrix} u_i h_i \\ u_i u_i h_i \\ u_i v_i h_i \end{pmatrix} \quad F_i^y = \begin{pmatrix} v_i h_i \\ v_i u_i h_i \\ v_i v_i h_i \end{pmatrix}$$

and

$$H_i = \begin{pmatrix} 0 \\ -h_i g \sum_{j=1}^i \varepsilon_j \frac{\partial z_j}{\partial x} + \frac{\tau_i^x}{\rho_0} - \frac{\tau_{i+1}^x}{\rho_0} + f v_i h_i \\ -h_i g \sum_{j=1}^i \varepsilon_j \frac{\partial z_j}{\partial y} + \frac{\tau_i^y}{\rho_0} - \frac{\tau_{i+1}^y}{\rho_0} + f u_i h_i \end{pmatrix}$$

where ρ_i , u_i and v_i are the layer averaged density and velocities in x and y directions, respectively, τ_i^x and τ_{i+1}^x are the upper and bottom interfacial shear stresses in the x direction, and $\rho_0 \varepsilon_i = (\rho_i - \rho_{i-1})$ for i greater than 1 and $\rho_0 \varepsilon_1 = \rho_1$ for $i = 1$. The

equations assume an inviscid fluid, i.e. the shear stresses are only considered at the water surface, and at the side walls [?]. The above system was first proposed by Brandt *et al.* [?] in their study of internal waves in the Strait of Messina. The weak nonhydrostatic effects were proposed by Brandt *et al.* [?] who kept the terms of the order $\mu^2 = h^2/L^2$, but neglected the terms of the order $\varepsilon = a/h$, where a is the wave amplitude scale and h and L are the layer thickness and basin scale, respectively. With this simplification, the nonhydrostatic terms in the momentum equations for each layer can be included as follows,

$$\frac{\partial \vec{u}_i}{\partial h_i} = B_i \vec{\nabla} \left[\frac{\partial (\vec{\nabla} \cdot \vec{u}_i h_i)}{\partial t} \right] + \vec{A} \quad (1.24)$$

where,

$$B_i = \begin{cases} \frac{h_1^2}{6} & i = 1 \\ \frac{h_1^2}{3} + \frac{h_1 h_2}{3} & i = 2 \end{cases}$$

and \vec{A} is the vector containing all the F_i and H_i terms. The above approximation includes the assumption that h_1 and h_2 are constant in the conservation of volume calculations ???. The governing equations used by de la Fuente *et al.* [?] in their study of internal waves in a circular basin for a single layer fluid are,

$$\begin{aligned} \frac{\partial h}{\partial t} + \vec{\nabla} \cdot (h\mathbf{u}) &= 0, \\ \frac{\partial uh}{\partial t} + \vec{\nabla} \cdot ((uh)\mathbf{u}) &= -gh \frac{\partial \eta}{\partial x} + fvh + \frac{H^2}{6} \frac{\partial (\vec{\nabla} \cdot \frac{\partial (\mathbf{u}h)}{\partial t})}{\partial x}, \\ \frac{\partial vh}{\partial t} + \vec{\nabla} \cdot ((uh)\mathbf{u}) &= -gh \frac{\partial \eta}{\partial y} - fuh + \frac{H^2}{6} \frac{\partial (\vec{\nabla} \cdot \frac{\partial (\mathbf{u}h)}{\partial t})}{\partial y}. \end{aligned} \quad (1.25)$$

where $\mathbf{u} = (u(x,y,t), v(x,y,t))$ is the velocity field, $h(x,y,t) = H(x,y) + \eta(x,y,t)$ is the total depth with H representing the undisturbed depth, and η is the free surface displacement. The difference between the set of equations ??? and ??? is the introduction of the of the dispersive terms $\frac{H^2}{6} \vec{\nabla} \cdot (\vec{\nabla} \cdot (uh)_t)$ found in the momentum equations ???. The extra terms are an approximation of the nonhydrostatic pressure gradients and are derived from perturbation theory in a rather lengthy calculation.

Many different dispersively modified sets of equations are available in the literature, with well known examples being those due to Peregrine and Nwogu. The dispersive shallow water equations represented by equations, ???, may not be able to capture dispersive properties in situations involving variable depth, H . Peregrine's equations are a 2D dispersive shallow water model for variable depth derived using Boussinesq's idea of expanding the

velocity potential in a Taylor series about the bottom, resulting in a dispersive evolution equation for the depth averaged velocity. Nwogu's Boussinesq system attempts to improve the Boussinesq equations by evaluating the velocity at an arbitrary depth-level. This arbitrary depth-level provides a free parameter that is then optimized for accuracy. To give an example of the different dispersively modified sets of equations, consider Peregrine's 2D modified Boussinesq equations [?] where the depth-averaged velocity \mathbf{u} satisfies,

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -g\nabla\eta + \frac{H}{2}\nabla(\nabla \cdot (H\frac{\partial u}{\partial t})) - \frac{H^2}{6}\nabla(\nabla \cdot (\frac{\partial u}{\partial t})), \quad (1.26)$$

and the fluid column height h satisfies ???. In the case of flat bottom, the dispersive terms can be grouped together as $\frac{H^2}{3}\nabla\nabla \cdot \mathbf{u}_t$. However, when the dispersion is compared to the linear theory, it is observed that there are phase and group velocity errors for depths over 2/10 to 3/10 the wavelength, respectively. For details, the reader can refer to the analysis carried out by Walkley [?]. A more accurate set of equations are Nwogu's extended Boussinesq system [?] where the velocity is evaluated at an arbitrary depth $z = \theta h$:

$$\frac{\partial h}{\partial t} + \nabla \cdot (hu) + \nabla \cdot (B_1 H^3 \nabla(\nabla \cdot u) + B_2 H^2 \nabla(\nabla \cdot (Hu))) = 0, \quad (1.27)$$

$$\frac{\partial u}{\partial t} + g\nabla\eta + (u \cdot \nabla)u + A_1 H^2 \nabla(\nabla \cdot \frac{\partial u}{\partial t}) + A_2 H \nabla(\nabla \cdot (H\frac{\partial u}{\partial t})) = 0, \quad (1.28)$$

where, $A_1 = \frac{\theta^2}{2}$, $A_2 = \theta$, $B_1 = \frac{\theta^2}{2} - 1/6$ and $B_2 = \theta + 1/2$. Given an optimal choice of θ determined by Nwogu [?], Walkley [?] demonstrates that the errors in the group and phase velocity are reduced considerably. The models developed by Nwogu and Peregrine capture the physics more accurately, however, it makes the numerical solution highly complex. The shallow water equations, given by, ???, are relatively simple to model numerically and can be considered as a stepping stone to include some dispersion. The goal of this study is not to give the most accurate description of dispersive waves in shallow water, but to have a simple and perhaps more accurate representation of short wave dispersion.

Consider the 1D shallow water equations,

$$u_t = -g\eta_x, \quad \eta_t = -Hu_x, \quad (1.29)$$

where,

$$\eta = \eta_0 e^{i(kx - \sigma t)}, \quad u = u_0 e^{i(kx - \sigma t)}. \quad (1.30)$$

Substituting the corresponding expressions of u and η from equation ?? into equation, ??, and after some algebraic simplification, we get,

$$\frac{\sigma}{k} = \sqrt{gH}. \quad (1.31)$$

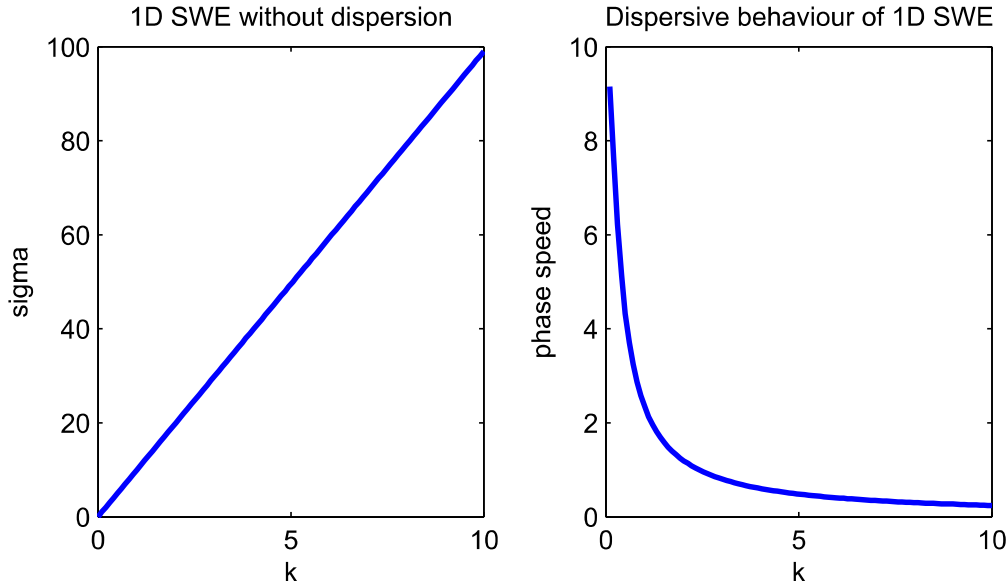


Figure 1.2: Dispersion relation in 1D: Plot of phase speed (σ/k) against the wavenumber (k). Plot to the left is without dispersion while plot to the right displays the dispersive behaviour.

Now, consider the 1D shallow water equations along with dispersive extensions,

$$u_t = -g\eta_x + \frac{H^2}{6}u_{xxt}, \quad \eta_t = -Hu_x. \quad (1.32)$$

Once again, substituting the values from, ??, into, ??, and after simplifying,

$$\frac{\sigma}{k} = \sqrt{\frac{gH}{1 + \frac{H^2}{6}k^2}}. \quad (1.33)$$

Figure ??, shows the 1D plot of phase speed σ as a function of k i.e. the wavenumber. As observed from the plot on the left, the phase speed has a linear relationship as the wavenumber increases for the 1D shallow water equations. However, for the plot on the right, for the dispersively modified shallow water equations, the phase speed decreases as k increases. However, as k reduces and approaches zero, the phase speed for both, the linear and dispersively modified equations should be the same. This is not the case in the figures, since different fields were plotted.

1.3 Weather Research and Forecasting (WRF) Model

The Weather Research and Forecasting (WRF) model is a numerical weather prediction (NWP) and atmospheric simulation system designed for both research and operational applications. The WRF.v.3.1 model is a fully compressible, non-hydrostatic model with an Eulerian mass dynamical core. Two dynamical cores are available: the Advanced Research WRF (ARW) core, developed and supported by NCAR (National Center for Atmospheric Research), and the Non-hydrostatic Mesoscale Model (NMM) core, whose development is centered at NCEPs (National Center for Environmental Prediction) Environmental Modeling Center (EMC) and support is provided by NCARs Development Testbed Center (DTC). The WRF-NMM is designed to be a real - time forecast model so physics schemes are preset. The WRF-ARW was chosen because it includes multiple physics options for turbulence/diffusion, radiation (long and shortwave), land surface, surface layer, planetary boundary layer, cumulus, and microphysics [?].

Time integration is performed with a 3rd-order Runge-Kutta scheme. WRF has split time integration, which uses smaller time steps used for fast processes like sound waves or gravity waves. WRF was designed to conserve mass, momentum, entropy, and scalars using flux form prognostic equations [?]. The grid format is Arakawa- C with all variables in the center of the grid except for wind velocity which is defined on the edges of the grid, and shared between adjacent grids. The WRF modeling system is divided into two main components: the WRF Preprocessing System (WPS) and the WRF modeling core. WPS helps define the WRF modeling domain, generate map, elevation/terrain, and land-use data, and generate horizontally interpolated input meteorological fields to the WRF grid. The WRF modeling core interpolates the meteorological fields processed by WPS to the WRF vertical levels and generates initial and lateral boundary conditions. The model solver integrates the atmospheric equations and interfaces with the physics parametrizations to generate forecasts of meteorological variables. WRF uses physics sub-models to simulate land surface, surface layer, and boundary layer dynamics, along with cumulus convection, microphysics, and radiation. To make these choices, the modeler must know not only the individual merits of each parameterization but also how it interacts with the other physics sub-model options [?].

Since idealized cases have been the subject of study in this research, the need to use the WPS did not arise. The WRF-ARW along with a MATLAB script that acts as a post-processing system was used to analyse the output. The study examines WRF in its Large Eddy Simulation (LES) configuration by running an idealized case provided with the WRF model. In order to study the bottom boundary layer (BBL) parametrizations in WRF an idealized case with a topography containing land and water was analysed along with the

full physics options. One of the key points of focus was to include a mean wind across the topography and to investigate how turbulence is represented in numerical models of environmental flows.

Chapter 2

Methods

2.1 Time Stepping Schemes for the Shallow Water Equations and Filtering

The non-rotating, conservative shallow water equations derived in section ?? and given by equation ?? can be rewritten as,

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \vec{\nabla})\vec{u} - g\frac{\partial \eta}{\partial x}, \quad (2.1)$$

$$\frac{\partial \vec{v}}{\partial t} = -(\vec{u} \cdot \vec{\nabla})\vec{v} - g\frac{\partial \eta}{\partial y}, \quad (2.2)$$

$$\frac{\partial \eta}{\partial t} = -\vec{\nabla} \cdot ((H + \eta)\vec{u}). \quad (2.3)$$

The time derivatives (LHS) of the equations ?? to ?? can be discretized by a time stepping scheme while the spatial derivatives (RHS) can be discretized by explicit numerical schemes. In this study, there are two kinds of time stepping schemes that have been implemented, the leapfrog and the Adams-Bashforth scheme. The first time stepping scheme discussed is the frequently adopted multi-step method known as the leapfrog scheme (also known as the midpoint method). It is an attractive method because it is simple to implement, second order accurate but most of all because it has very good stability for computing oscillatory solutions [?]. Consider U to be a grid variable representing a parameter such as velocity, pressure, temperature, etc. and Δt be the time step. The

leapfrog time stepping scheme can then be expressed as,

$$U^{n+1} = U^{n-1} + 2\Delta t f(U^n) \quad (2.4)$$

The grid variable U , is computed at the time level $n+1$ by extracting information about the variable from the previous two time levels n and $n-1$. The leapfrog time stepping scheme has a spurious mode, that needs to be taken care of. The leapfrog time stepping scheme when implemented on the time derivatives in equation ?? to ?? can then be written as,

$$u^{n+1} = u^{n-1} + 2\Delta t \left(-\vec{u} \frac{\partial \vec{u}}{\partial x} - \vec{v} \frac{\partial \vec{u}}{\partial y} - g \frac{\partial \eta}{\partial x} \right), \quad (2.5)$$

$$v^{n+1} = v^{n-1} + 2\Delta t \left(-\vec{u} \frac{\partial \vec{v}}{\partial x} - \vec{v} \frac{\partial \vec{v}}{\partial y} - g \frac{\partial \eta}{\partial y} \right), \quad (2.6)$$

$$\eta^{n+1} = \eta^{n-1} + 2\Delta t \left(-\frac{\partial [\vec{u}(H + \eta)]}{\partial x} - \frac{\partial [\vec{v}(H + \eta)]}{\partial y} \right). \quad (2.7)$$

A MATLAB function, named `rhseval.m` was created to compute the spatial derivatives (RHS) using the in built MATLAB function `fft2`. The function `fft2`, returns the two dimensional discrete Fourier transform, computed using a fast Fourier transform (FFT) algorithm. The input provided to the script `rhseval.m` includes the parameters u , v and η , that are matrices, along with k and l , that represent wave numbers along the X and Y co-ordinate axis respectively.

The aforementioned, second time stepping scheme, Adams-Bashforth method, is discussed hereafter. Both the time stepping methods discussed in this section are members of a class of methods called linear multi-step methods (LMM) [?]. In general an r -step LMM has the form given by,

$$\sum_{j=0}^r \alpha_j U^{n+j} = \Delta t \sum_{j=0}^r \beta_j f(U^{n+j}, t_{n+j}) \quad (2.8)$$

The value U^{n+j} is calculated from the previous values $U^{n+j-1}, U^{n+j-2}, \dots, U^n$ and f values at these points. In equation, ?? if the value of $\beta_j = 0$, then the method is explicit else it is implicit. The *Adams* methods have the form,

$$U^{n+r} = U^{n+r-1} + \Delta t \sum_{j=0}^r \beta_j f(U^{n+j}) \quad (2.9)$$

The β_j coefficients are chosen to maximize the order of accuracy. If we require $\beta_r = 0$ so the method is explicit then the ‘r’ coefficients $\beta_0, \beta_1, \dots, \beta_{r-1}$ can be chosen so that the method has order r . This gives the r -step *Adams-Bashforth* method [?]. The 3-step Adams-Bashforth method is given by,

$$U^{n+3} = U^{n+2} + \frac{\Delta t}{12}(5f(U^n) - 16f(U^{n+1}) + 23f(U^{n+2})) \quad (2.10)$$

The forward Euler method, denoted by equation ??, is used to calculate the previous conditions at time levels $n+1$ and $n+2$ in the Adams-Bashforth method.

$$U^{n+1} = U^n + \Delta t f(U^n) \quad (2.11)$$

The 3-step Adams-Bashforth method when implemented in a similar fashion on the shallow water equations, ?? to ??, can be written as,

$$\begin{aligned} u^{n+3} = & u^{n+2} + \frac{\Delta t}{12} \left(5 \left(-\vec{u} \frac{\partial \vec{u}}{\partial x} - \vec{v} \frac{\partial \vec{u}}{\partial y} - g \frac{\partial \eta}{\partial x} \right)^n \right. \\ & - 16 \left(-\vec{u} \frac{\partial \vec{u}}{\partial x} - \vec{v} \frac{\partial \vec{u}}{\partial y} - g \frac{\partial \eta}{\partial x} \right)^{n+1} \\ & \left. + 23 \left(-\vec{u} \frac{\partial \vec{u}}{\partial x} - \vec{v} \frac{\partial \vec{u}}{\partial y} - g \frac{\partial \eta}{\partial x} \right)^{n+2} \right), \end{aligned} \quad (2.12)$$

$$\begin{aligned} v^{n+3} = & v^{n+2} + \frac{\Delta t}{12} \left(5 \left(-\vec{u} \frac{\partial \vec{v}}{\partial x} - \vec{v} \frac{\partial \vec{v}}{\partial y} - g \frac{\partial \eta}{\partial y} \right)^n \right. \\ & - 16 \left(-\vec{u} \frac{\partial \vec{v}}{\partial x} - \vec{v} \frac{\partial \vec{v}}{\partial y} - g \frac{\partial \eta}{\partial y} \right)^{n+1} \\ & \left. + 23 \left(-\vec{u} \frac{\partial \vec{v}}{\partial x} - \vec{v} \frac{\partial \vec{v}}{\partial y} - g \frac{\partial \eta}{\partial y} \right)^{n+2} \right), \end{aligned} \quad (2.13)$$

$$\begin{aligned} \eta^{n+3} = & \eta^{n+2} + \frac{\Delta t}{12} \left(5 \left(-\frac{\partial [\vec{u}(H + \eta)]}{\partial x} - \frac{\partial [\vec{v}(H + \eta)]}{\partial y} \right)^n \right. \\ & - 16 \left(-\frac{\partial [\vec{u}(H + \eta)]}{\partial x} - \frac{\partial [\vec{v}(H + \eta)]}{\partial y} \right)^{n+1} \\ & \left. + 23 \left(-\frac{\partial [\vec{u}(H + \eta)]}{\partial x} - \frac{\partial [\vec{v}(H + \eta)]}{\partial y} \right)^{n+2} \right). \end{aligned} \quad (2.14)$$

A fundamental model equation in fluid mechanics, known as the Burgers equation, that accounts for the viscosity ν , and is given by,

$$\vec{u}_t = -u\vec{u}_x + \nu u_{xx} \quad (2.15)$$

The above equation gained popularity when Hopf (1950) and Cole (1951) proved that the general solution could be obtained explicitly [?]. Cole and Hopf noted that the equation ?? can be reduced to the linear heat equation by the nonlinear transformation,

$$u = -2\nu \frac{\varphi_x}{\varphi}. \quad (2.16)$$

The transformation can be carried out in two steps. First introduce,

$$u = \psi_x$$

so that ?? may be integrated to,

$$\psi_t + \frac{1}{2}\psi_x^2 = \nu\psi_{xx}.$$

Then introduce,

$$\psi = -2\nu \log \varphi$$

to obtain,

$$\psi_t = \nu\psi_{xx} \quad (2.17)$$

The nonlinear transformation eliminates the nonlinear term. The general solution of the heat equation ?? is well known and can be handled by a variety of methods [?]. This allows one to solve a initial value problem given by,

$$c = F(x) \quad \text{at} \quad t = 0.$$

This transforms through ?? into the initial value problem,

$$\varphi = \Phi(x) = \exp\left(-\frac{1}{2\nu} \int_0^x F(\eta) d\eta\right), \quad t = 0, \quad (2.18)$$

for the heat equation. The solution for φ is,

$$\varphi = \frac{1}{\sqrt{4\pi\nu t}} \int_{-\infty}^{\infty} \Phi(\eta) \exp\left(-\frac{(x-\eta)^2}{4\nu t}\right) d\eta. \quad (2.19)$$

Therefore, from ??, the solution for u can be written as,

$$u(x, t) = \frac{\int_{-\infty}^{\infty} \frac{x-\eta}{t} e^{-G/2\nu} d\eta}{\int_{-\infty}^{\infty} e^{-G/2\nu} d\eta} \quad (2.20)$$

where,

$$G(\eta, x, t) = \int_0^\eta F(\eta) d\eta + \frac{(x - \eta)^2}{2t}. \quad (2.21)$$

The behaviour of the solution, as $\nu \rightarrow 0$ leads to discontinuities and shocks. Fluid flows spontaneously develop narrow regions of very large gradients. These regions are known as ‘fronts’ in geophysics and ‘shocks’ in aerospace and plasma physics [?]. These discontinuities or shocks are unresolvable since their scales are smaller/narrower than the distance between two adjacent grid points, more commonly referred to as Δx . Therefore, it is necessary to apply either smoothing or filtering to avoid computational problems such as instability. In Fourier analysis,

$$\hat{u} = F^{-1}(A(k)F(u^n)). \quad (2.22)$$

In practice, there are many filters that can be applied. Low pass filters passes leave low frequency signals unchanged but attenuate signals with frequencies higher than the cutoff frequency. High pass filters leave high frequencies well unchanged but attenuate signals with frequencies lower than the cutoff frequency. The best choice of filter would be the one that keeps the largest part of the solution unchanged. In two dimensional analysis, isotropic filters, i.e. $A(\sqrt{k^2 + l^2})$ are used as opposed to the more general $A(k, l)$.

When discretizing the Burgers equation ??, care is taken to discretize the non-linear term explicitly and the diffusion term implicitly. This is because implicit diffusion acts as a filter. Taking the Fourier transform of equation ??,

$$(\vec{u}_t)^{n+1} = -ik\left(\frac{\vec{u}^2}{2}\right)^n - k^2\nu(\vec{u})^{n+1}, \quad (2.23)$$

$$(1 + k^2\nu)\vec{u}^{n+1} = -ik\left(\frac{\vec{u}^2}{2}\right)^n, \quad (2.24)$$

$$\vec{u}^{n+1} = \left(\frac{1}{1 + k^2\nu}\right)[-ik\left(\frac{\vec{u}^2}{2}\right)^n]. \quad (2.25)$$

When comparing equation ?? to equation ??, the term $A(k)$, is given by,

$$A(k) = \frac{ik}{1 + k^2\nu}. \quad (2.26)$$

The Burgers equation also provides insight on how the nonlinear terms affect the solution in the Fourier transformed domain. Consider the inviscid Burgers equation, with

initial conditions given by,

$$\begin{aligned}\vec{u}_t + u\vec{u}_x &= 0, \\ u(0, x) &= e^{ikx}\end{aligned}\tag{2.27}$$

Using the Euler time stepping scheme to discretize the above equation,

$$u^{n+1} = u^n + \Delta t(-uu_x)^n,\tag{2.28}$$

$$u^1 = u^0 + \Delta t(-e^{ikx}ike^{ikx}),\tag{2.29}$$

$$u^1 = e^{ikx} - \Delta tike^{i2kx},\tag{2.30}$$

$$u^2 = u^1 + \Delta t[-(e^{ikx} - \Delta te^{i2kx})(ike^{ikx} - 2\Delta tike^{i2kx})]\tag{2.31}$$

Thus, u^2 will have terms in e^{ikx} , e^{i2kx} , e^{i3kx} and so on and so forth. Hence, as noticed, as time increases higher and higher wavenumbers become part of the solution. For any grid size, there are a finite number of wavenumbers that can be represented. Therefore, for larger time, the part of the solution consisting of higher values of wavenumbers has to be filtered. This acts as a low pass filter, filtering the part of the solution with higher wavenumbers and allowing only those wavenumber that can be represented on the grid.

2.2 Time Stepping Schemes for Dispersively Modified Shallow Water Equations using FFT

It is well known that dispersion can balance, and hence tame, the non linearity in a similar manner as observed for diffusion. For example consider the model equation (sometimes called the regularized longwave equation, or RLW),

$$A_t + AA_x = \beta A_{xxt}\tag{2.32}$$

Taking the Fourier transform of the above equation,

$$A_t^{n+1} + ik\left(\frac{A^2}{2}\right) = -\beta k^2 A_t,\tag{2.33}$$

$$(1 + k^2\beta)A_t = -ik\frac{A^2}{2},\tag{2.34}$$

$$A_t = \frac{-ik}{1 + \beta k^2} \frac{A^2}{2}\tag{2.35}$$

where any explicit time stepper in Fourier space can be used to transform back into real number space. Notice that as k increases, the right hand side of equation ?? reduces and eventually tends to zero at a very high wave number. Hence, dispersion in that it removes energy at the of highest wavenumbers. Consider, the dispersively modified shallow water equations in equation ?. Neglecting the Coriolis force, the modified shallow water equations can be written as,

$$u_t - \frac{H^2}{6}(\vec{\nabla} \cdot u_t)_x = -uu_x - vv_y - g\eta_x, \quad (2.36)$$

$$v_t - \frac{H^2}{6}(\vec{\nabla} \cdot u_t)_y = -vv_y - vu_x - g\eta_y, \quad (2.37)$$

$$\eta_t = -[(H + \eta)u]_x - [(H + \eta)v]_y. \quad (2.38)$$

This can be rewritten schematically as,

$$u_t - \frac{H^2}{6}(\vec{\nabla} \cdot u_t)_x = F_x, \quad (2.39)$$

$$v_t - \frac{H^2}{6}(\vec{\nabla} \cdot u_t)_y = F_y, \quad (2.40)$$

$$\eta_t = F_\eta. \quad (2.41)$$

Carrying out a double Fourier transform (transformed variables are denoted by an overbar) it is observed that the conservation of mass (or η equation) can be time stepped independently of the momentum equations. The momentum equations read,

$$(1 + k^2 H^2/6)\bar{u}_t + klH^2/6\bar{v}_t = \bar{F}_x, \quad (2.42)$$

$$(klH^2/6)\bar{u}_t + (1 + l^2 H^2/6)\bar{v}_t = \bar{F}_y. \quad (2.43)$$

Stepping forward in time is thus not as simple as the purely explicit time stepping schemes of the shallow water equations. However, through the use Fourier methods, it is possible to step forward provided a 2×2 system is solved for each wave number pair (k, l) . In MATLAB this can be implemented by using component-wise operators like `.*` to construct and store the inverse for all wave numbers.

2.3 Implementation

In this section, a few sample MATLAB codes developed to solve the dispersively modified shallow water equations shall be presented and discussed in detail. An explicit ODE

solver such as the 3-step Adams-Bashforth along with the MATLAB function `fft2` has been implemented to solve the spatial derivatives. The MATLAB function `rhseval.m`, evaluates the right hand side of the dispersively modified shallow water equations, given by, ?? to ??, using the fast Fourier transform. The double Fourier transform MATLAB operator is applied to evaluate the right hand side. The output produced by the function `rhseval.m`, in Fourier space includes the terms, \bar{F}_x, \bar{F}_y and \bar{F}_η .

The 3-step Adams-Bashforth method solver, applied to solve the complete set of dispersively modified shallow water equations given by equation, ?? to ??, is given by the program, `absolver.m`. The program calls the function `rhseval.m` within its subroutine multiple number of times.

```
% rhseval.m - Evaluating the right hand side of equations

function [rhs1,rhs2,rhs3] = rhseval(k,l,u,v,eta)
H=10; g=9.81;
% derivative operators (in Fourier space)
Dx=sqrt(-1)*k; Dy=sqrt(-1)*l;
feta=fft2(eta);
% rhs1 evaluated
rhs1 = -0.5*sqrt(-1)*k.*fft2(u.*u)...
-fft2(v.*real(ifft(sqrt(-1)*l.*fft(u,[],1),[],1))) -g*sqrt(-1)*k.*feta;

% rhs2 evaluated
rhs2 = -fft2(u.*real(ifft(sqrt(-1)*k.*fft(v,[],2),[],2)))...
-0.5*sqrt(-1)*l.*fft2(v.*v) -g*sqrt(-1)*l.*feta;

% rhs3 evaluated
rhs3 = -sqrt(-1)*k.*fft2((H+eta).*u) -sqrt(-1)*l.*fft2((H+eta).*v);
```

RHS_1 , RHS_2 and RHS_3 are calculated by calling the function `rhseval.m`. The equation given by, ??, represents the 3-step Adams Bashforth method. The role that the function, `rhseval.m`, plays within `absolver.m`, and how they work in tandem, is clarified from the following description,

$$\begin{pmatrix} \bar{u} \\ \bar{v} \\ \bar{\eta} \end{pmatrix}_t + \frac{H^2}{6} \begin{bmatrix} -k^2\bar{u} - lk\bar{v} \\ -kl\bar{u} - l^2\bar{v} \\ 0 \end{bmatrix}_t = \begin{pmatrix} RHS_1 \\ RHS_2 \\ RHS_3 \end{pmatrix}$$

where, k, l represent the wavenumber in the X and Y direction, respectively. Representing the above expression in terms of matrices,

$$\begin{bmatrix} 1 - \frac{H^2}{6}k^2 & -lk\frac{H^2}{6} & 0 \\ -kl\frac{H^2}{6} & 1 - \frac{H^2}{6}l^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{u} \\ \bar{v} \\ \bar{\eta} \end{bmatrix} = \begin{pmatrix} RHS_1 \\ RHS_2 \\ RHS_3 \end{pmatrix}$$

and denoting the matrix (comprising wavenumbers) on the left as matrix A , the entire system can be written as,

$$\begin{bmatrix} \bar{u} \\ \bar{v} \\ \bar{\eta} \end{bmatrix} = A^{-1} \begin{pmatrix} RHS_1 \\ RHS_2 \\ RHS_3 \end{pmatrix}$$

```
% absolver.m - 3-step Adams-Bashforth solver

% Declaring the constants
Nx=128; Ny=128; Nt=40; t=0; numouts=10;
Lx=1e3; Ly=1e3; H=10; g=9.81;

% Grid
x1=linspace(-1,1,Nx+1)*Lx; x=x1(1:end-1);
y1=linspace(-1,1,Ny+1)*Ly; y=y1(1:end-1);
[xx,yy]=meshgrid(x,y);
dx=abs(xx(2,2)-xx(1,1)); dy=abs(yy(2,2)-yy(1,1));
cfl=min(dx,dy)/sqrt(g*H); dt=0.2*cfl;

% Wavenumbers in Matlab's order
dk=pi/Lx; dl=pi/Ly;
ksvec(1)=0; ksvec(Nx/2+1)=0;
lsvec(1)=0; lsvec(Ny/2+1)=0;
for ii=2:(Nx/2)
    ksvec(ii)=ii-1;
    ksvec(Nx/2+ii)=-Nx/2 + ii -1;
end
for ii=2:(Ny/2)
    lsvec(ii)=ii-1;
    lsvec(Ny/2+ii)=-Ny/2 + ii -1;
```

```

end
ksvec=ksvec*dk; lsvec=lsvec*dl;
[k,l]=meshgrid(ksvec,lsvec);
k2=k.*k; l2=l.*l; kl=k.*l;

% Building the 'radial' filter
filtorder=8;
cutoff=0.65;
alpha1=0.1;
alpha2=1.1;
kmax=max(k(:));
lmax=max(l(:));
kcrit=kmax*cutoff;
lcrit=lmax*cutoff;
kmag=sqrt(k.*k+l.*l);
myfilter=exp(-alpha1*(kmag/(alpha2*kcrit)).^filtorder);

% Matrix inverse by hand
dump1=(H*H)/6;
a11=1+(dump1*k2);
a12=(dump1*kl);
a21=(dump1*kl);
a22=1+(dump1*l2);
mydet=a11.*a22-a12.*a21;

% Calculating the inverse matrix
b11=a22./mydet;
b22=a11./mydet;
b12=-a12./mydet;
b21=-a21./mydet;

% Set the initial conditions
r=sqrt(xx.*xx+yy.*yy);
eta0=0.4*H*exp(-r.*r/(0.1*Lx*0.1*Lx));
u0 = zeros(size(xx)); v0 = zeros(size(yy));
upp=u0; vpp=v0; etapp=eta0;

% Forward Euler step for calculating the previous-previous conditions

```

```

[rhs1 rhs2 rhs3]=rhseval(k,l,u0,v0,eta0);
up = fft2(upp) + dt*(b11.*rhs1 + b12.*rhs2);
vp = fft2(vpp) + dt*(b21.*rhs1 + b22.*rhs2);
etap = fft2(etapp) + dt*(rhs3);
up=real(ifft2(up.*myfilter));
vp=real(ifft2(vp.*myfilter));
etap=real(ifft2(etap.*myfilter));

% Forward Euler step for calculating the previous conditions
[rhs1 rhs2 rhs3]=rhseval(k,l,up,vp,etap);
un = fft2(up) + dt*(b11.*rhs1 + b12.*rhs2);
vn = fft2(vp) + dt*(b21.*rhs1 + b22.*rhs2);
etan = fft2(etap) + dt*(rhs3);

% Filtering after the time step
un=real(ifft2(un.*myfilter));
vn=real(ifft2(vn.*myfilter));
etan=real(ifft2(etan.*myfilter));

% 3 - step Adams Bashforth
for jj=1:numouts
    for ii = 1:Nt
        t = t+dt;
        [rhs1n rhs2n rhs3n]=rhseval(k,l,un,vn,etan);
        [rhs1p rhs2p rhs3p]=rhseval(k,l,up,vp,etap);
        [rhs1pp rhs2pp rhs3pp]=rhseval(k,l,upp,vpp,etapp);

        uf = fft2(un)+ dt/12*(23*(b11.*rhs1n + b12.*rhs2n)...
            -16*(b11.*rhs1p + b12.*rhs2p)...
            +5*(b11.*rhs1pp + b12.*rhs2pp));

        vf = fft2(vn)+ dt/12*(23*(b21.*rhs1n + b22.*rhs2n)...
            -16*(b21.*rhs1p + b22.*rhs2p)...
            +5*(b21.*rhs1pp + b22.*rhs2pp));

        etaf = fft2(etan) + dt/12*(23*rhs3n-16*rhs3p+5*rhs3pp);
% Filtering after the 3 - step Adams-Bashforth calculations
uf=real(ifft2(uf.*myfilter));

```

```

        vf=real(iff2(vf.*myfilter));
        etaf=real(iff2(etaf.*myfilter));
% Reassigning the new values
        upp=up; up=un; un=uf;
        vpp=vp; vp=vn; vn=vf;
        etapp=etap; etap=etan; etan=etaf;
    end
end

```

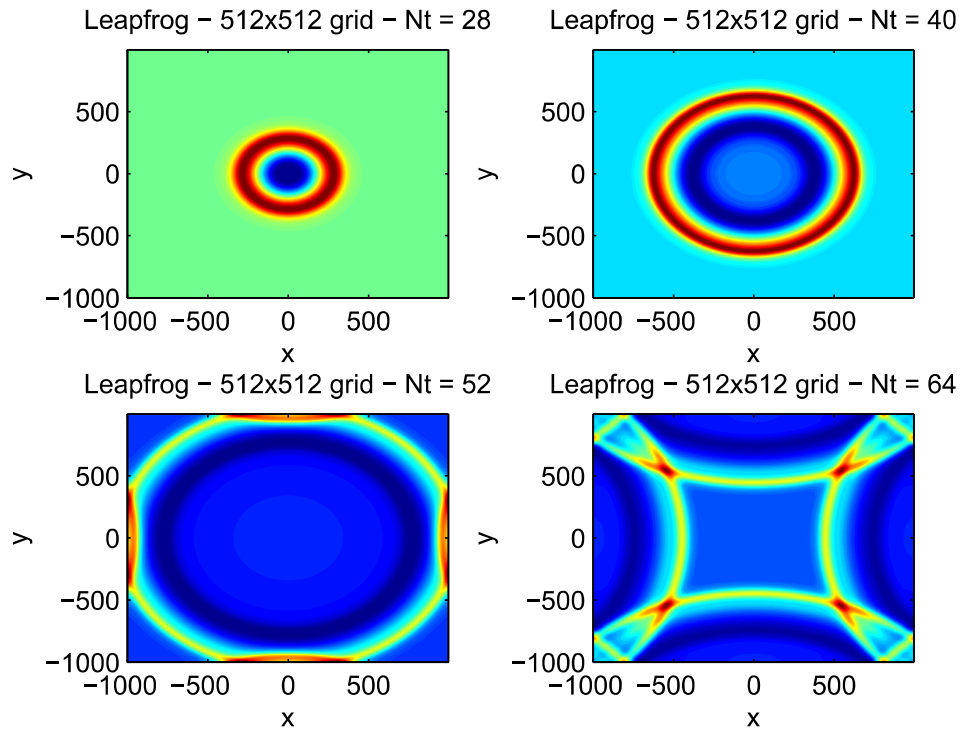


Figure 2.1: Variation of elevation using the leapfrog time stepping scheme on a 512 x 512 grid

An initial surface elevation/disturbance $\eta_0 = 0.4He^{-\frac{r^2}{0.01L_x^2}}$, is allowed to generate radially propagating waves. r is the radius of the domain and L_x is the length of the Fourier domain in the X direction. Due to the high order of the scheme, symmetry is preserved

even after wave-wave interactions take place. Figure ?? shows the variation of the elevation on a uniform Cartesian grid with 512 points in the X and Y directions. As observed (from the top), the initial disturbance, propagates radially outwards, colliding with the boundaries and returning back radially towards the center of the disturbance. The figure shows the variation in the elevation for different time intervals. The colours indicate the change in the value of the elevation, with the colour red indicating a higher value and blue the lower value, as the wave propagates radially outwards.

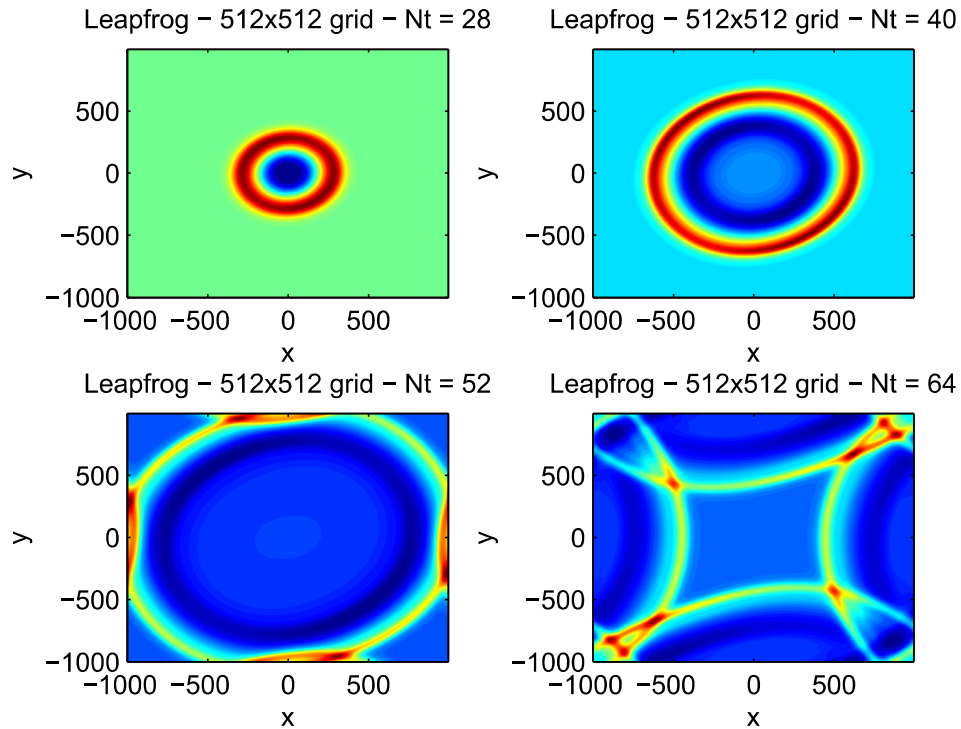


Figure 2.2: Variation of ‘ η ’, by altering the initial conditions for the ‘ v ’ velocity and introducing an initial current using the leapfrog time stepping scheme on a 512 x 512 grid

Similarly, using the same initial η , an initial velocity $v = \sin(\frac{\pi x}{L_x})$, is set up to generate radially propagating waves along with an initial current. Figure ?? shows the variation of the elevation on a uniform Cartesian grid with 512 points in the X and Y directions. The behaviour is not as symmetrical as the plot in figure ??. The figure shows the variation in elevation for different time intervals. Due to the initial current, from the ‘ v ’ velocity, there is no symmetry observed as in figure ??.

2.4 Timing and Matlab Profiler





The `profile` function in MATLAB helps debug and optimize the code files by tracking their execution time. MATLAB profiler records information about execution time, number of calls, parent functions, child functions, code line hit count, and code line execution time of functions and sub-functions within the code. There are two MATLAB code files, on which the `profile` function was implemented to analyse the amount of time taken individually by both scripts. The first code file, i.e. the M-file `lfrg_timing.m`, uses the ‘Leapfrog’ time stepping scheme while the second code file, `ab_solver.m` uses the three step ‘Adams-Bashforth’ scheme to model the dispersively modified shallow water equations using FFT methods. It is recommended to use `profile` without implementing any of the post processing functions (such as `plot`, etc.) in the script. This is because, when analysing the functions or sub-functions within the code, it helps to prioritize the analysis of sub routines. Sub routines that play a significant role in the architecture of the code and consume a big chunk of CPU time are given higher priority. One is curious to know about the CPU time taken by core task performing functions rather than extracting information from functions that perform trivial operations like plotting or displaying the output.

The figure ?? displays the output after implementing the `profile` function on the first script (`lfrg_timing.m`) that uses the leapfrog time stepping scheme. The dark band indicates the self time spend by a function. Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes the overhead resulting from the process of profiling. `rhseval.m` (that consists of a lot of fft routines) is a subroutine that is called 601 times in the main code `lfrg_timing.m`. The maximum amount of CPU time is consumed by the MATLAB function `fft2` is called by the main code `lfrg_timing.m` and the subroutine `rhseval.m` to a total of 6010 times.

The grid size was varied from a 64×64 sized grid, doubling the number of points all the way to 2048×2048 . Both the programs, i.e. the `lfrg_timing.m` and `absolver.m` were run on two workstations labelled ‘Belize’ and ‘Zambezi’ consisting of 16 and 4 processors respectively. The most important parameter under observation is the percentage of CPU memory used or the number of processors involved in running the job. Considering the number of processors, it is obvious, that it took less amount of time to run the job on ‘Belize’ as opposed to running it on ‘Zambezi’. On both workstations, for both the programs, the MATLAB function `fft`, consumes a larger percentage of CPU memory. Essentially, the `fft` function, i.e. the dominant command requiring maximum amount of time (refer to figure ??) in both the programs, also distributes its job among different processors in order to achieve minimum computation time/processor. For example, for a grid size of 2048×2048 , on the workstation ‘Belize’, the `fft` function uses 3 to 8 processors for the pro-

Profile Summary

Generated 07-Jul-2011 13:51:16 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
lfrg_timing	1	7.083 s	0.464 s	
rhseval	601	4.889 s	3.926 s	
ifft2	1803	1.650 s	1.650 s	
fft2	6010	1.043 s	1.043 s	
close	1	0 s	0.000 s	
close>safegetchildren	1	0 s	0.000 s	
close>checkfigs	1	0 s	0.000 s	
close>request_close	1	0 s	0.000 s	
linspace	2	0 s	0.000 s	
meshgrid	2	0 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

Figure 2.3: MATLAB profiler output, grid size of 1024 x 1024 points using the leapfrog time stepping scheme

gram `lfrg_timing.m` while it requires up to 11 processors for the program `absolver.m`. Hence, when compared, the `absolver.m` required more processors because of the 3-step multi-stage numerical method. As the grid size reduces, consequently, the amount of processors required reduce. As opposed to ‘Belize’, running jobs on the workstation ‘Zambezi’, required more computational time/processor.

Chapter 3

The Weather Research and Forecasting Model (WRF)

3.1 WRF in its LES configuration

The governing equations of motion used to model turbulence are the widely used Navier-Stokes equations. Modelling turbulence, is still considered as one of the challenges involved in capturing complex physics. Various numerical methods have been used to study turbulence, ranging from simple linear solvers to direct numerical simulations. Linear models being easy to implement have limited accuracy. Complex models being more difficult to compute, produce much more precise solutions. Different numerical methods used to study turbulence models are: 1) Linearized flow models 2) Reynolds average modelling (RANS) 3) Direct numerical simulation (DNS) and 4) Large Eddy Simulation (LES).

Linearized models are best suited for uncomplicated geometries and contain simple models for turbulence. The RANS are time averaged equations for fluid flow whereby an instantaneous quantity is split up into its time-averaged and fluctuating quantities. DNS is a simulation in which the Navier-Stokes equations are numerically solved, without any turbulence model. As a result, the entire range of the spatial and temporal scales of the turbulence must be resolved. The smallest of the scales, ranging from the Kolmogorov scale (further discussed) to the integral scale, associated with the motions containing most of the kinetic energy, is resolved on the computational domain. Although it is a very useful tool for turbulence research, DNS is highly computationally expensive.

This section examines the Large Eddy Simulation (LES) case provided within the Weather Research and Forecasting (WRF) model. A brief discussion on the representation

of turbulence in numerical models of environmental flows is provided. Many environmental flows contain interesting behaviour on a variety of scales (particularly high Reynolds number turbulent flows). Even with the computational power available today, it is impossible to represent all of these scales in a numerical simulation. For example, the convective boundary layer simulations described in this report are in a domain 2km high, while the Kolmogorov microscale for this type of atmospheric flow is about 10^{-3}m [?]. Thus, in order to represent the dissipative eddies characterized by the Kolmogorov microscale, over 2 million grid cells would be needed in the vertical. Extending this to three dimensions would be prohibitively expensive and computationally infeasible.

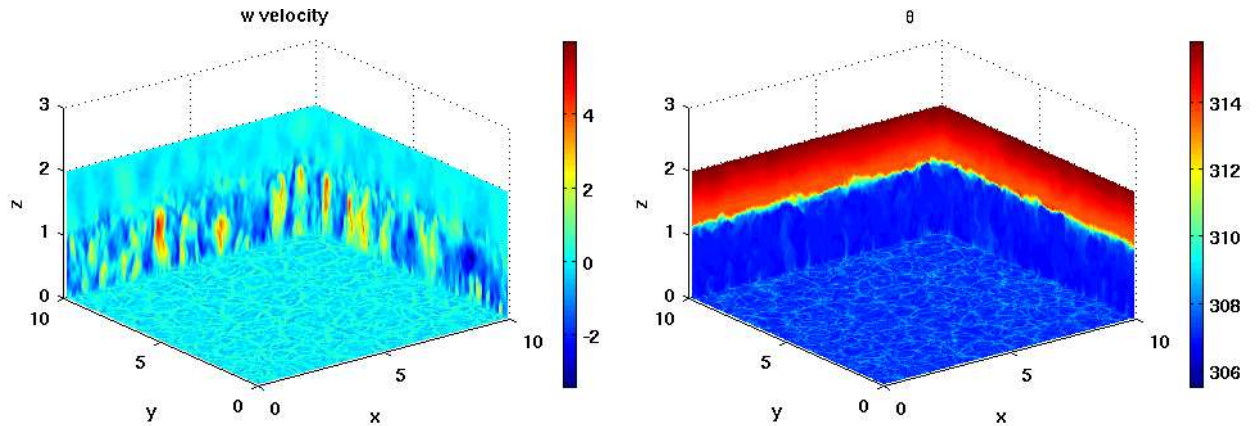


Figure 3.1: Plot showing the variation of w velocity and the potential temperature for a horizontal wind of 0 m/s (case 1)

In an LES model, variables are split into resolved and unresolved (or sub-filter scale) parts. Large scales contain most of the energy and fluxes that are significantly affected by the flow configuration. The turbulent sub-filter scale fluxes are parametrized by an eddy viscosity model. An outline on the three ways of defining the eddy viscosity is provided: 1) Constant eddy viscosity model, 2) Smagorinsky model, and 3) Turbulent Kinetic Energy model. There are many other options for representing turbulence with the WRF model. For instance, there are a variety of planetary boundary layer schemes and surface layer parametrizations.

Like other LES models, WRF uses an eddy viscosity treatment of the sub-filter scale stress tensor so that,

$$\tau_{ij} = -\mu_d K_{(h,v)} D_{ij},$$

where, τ_{ij} is the stress tensor, D_{ij} is the deformation tensor, μ_d is the mass of dry air in a column, and $K_{(h,v)}$ represents that horizontal or vertical eddy viscosity [?]. WRF treats the

horizontal (K_h) and vertical (K_v) eddy viscosities differently. In particular, K_h is allowed to vary along coordinate surfaces but K_v is not.

Obviously, in the constant eddy viscosity model, the eddy viscosities are constant. These values are set in the namelist.input file before the simulations begin. In this study, $K_h = 1$ and $K_v = 1$ is chosen for each of the resolution runs. In hindsight, it may have been more appropriate to choose a lower eddy viscosity for the better resolved cases. Essentially, the eddy viscosity adds diffusion and helps keep the method stable, so a lower eddy viscosity can be tolerated for better resolution.

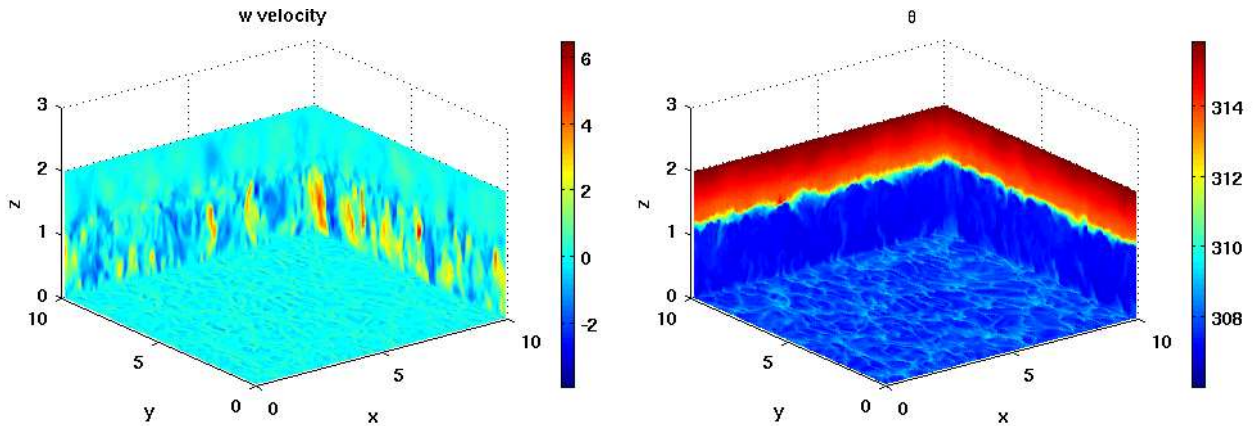


Figure 3.2: Plot showing the variation of w velocity and the potential temperature for a horizontal wind of 10 m/s (case 2)

The three dimensional Smagorinsky scheme calculates the eddy viscosity using the following formula in Skamarock et al. [?] given by,

$$K_{(h,v)} = C_s^2 l_{(h,v)}^2 \max(0, (D^2 - Pr^{-1}N^2)^{1/2}), \quad (3.1)$$

where, N^2 is the squared squared buoyancy frequency, Pr is the Prandtl number, $l_{(h,v)}$ is the horizontal or vertical mixing length and D^2 is a function of the deformation tensor $D_{(i,j)}$. C_s is a constant, which is typically set to a value of 0.25, however, in the simulations carried out, $C_s = 0.18$. The mixing length is related to the grid size by two different options i.e., isotropic mixing and anisotropic mixing. $\Delta x, \Delta y \simeq \Delta z$, the mixing length is set to $l_{(h,v)} = (\Delta x \Delta y \Delta z)^{1/3}$. In this case, the horizontal and vertical eddy viscosities are equal. For anisotropic mixing, $\Delta z \ll \Delta x, \Delta y$, the horizontal and vertical mixing lengths are $l_h = (\Delta x \Delta y)^{1/2}$ and $l_v = \Delta z$. In the simulations carried out, the isotropic mixing option was implemented since Δz is about the same order of magnitude as Δx and Δy . WRF offers

another 2D Smagorinsky model, where the horizontal eddy viscosity is set by a Smagorinsky equation similar to ?? above. The vertical eddy viscosity can then either be calculated using a planetary boundary layer scheme, or is set to constant in the namelist file. This option would be useful in simulations experimenting with different planetary boundary layer schemes.

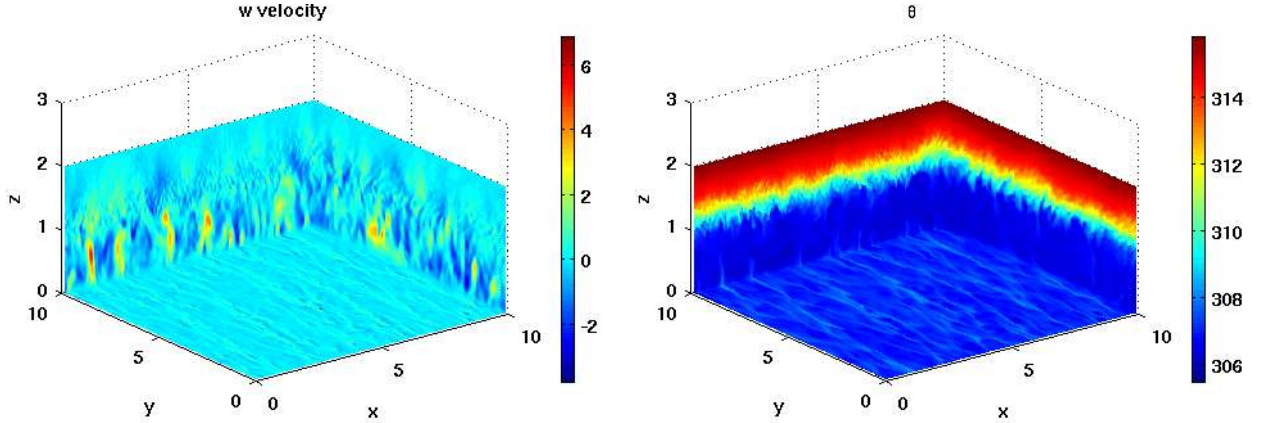


Figure 3.3: Plot showing the variation of w velocity and the potential temperature for a horizontal wind of 20 m/s (case 3)

The turbulent kinetic energy scheme calculates the eddy viscosities as a function of the turbulent kinetic energy. This involves solving an evolution equation for the turbulent kinetic energy, and so could be considered more expensive than the other schemes. The formula for the eddy viscosities according to Skamarock et al. [?] is given by,

$$K_{(h,v)} = C_k l_{(h,v)} \sqrt{e}, \tag{3.2}$$

where e is the turbulent kinetic energy, C_k is a constant and $l_{(h,v)}$ is the horizontal or vertical mixing length. In the simulations, $C_k = 0.10$. The evolution equation for the turbulent kinetic energy is outlined by Skamarock et al. [?]. The mixing length is again set with two options; isotropic and anisotropic (for the TKE runs, the isotropic setting is used). For the isotropic case, the mixing length is

$$l_{(h,v)} = \begin{cases} \min((\Delta x \Delta y \Delta z)^{(1/3)}, 0.76 \sqrt{e}/N) & \text{if } N^2 > 0 \\ (\Delta x \Delta y \Delta z)^{(1/3)} & \text{if } N^2 \leq 0 \end{cases}$$

In the anisotropic case, the vertical mixing length is,

$$l_{(h,v)} = \begin{cases} \min((\Delta z)^{(1/3)}, 0.76 \sqrt{e}/N) & \text{if } N^2 > 0 \\ \Delta z & \text{if } N^2 \leq 0 \end{cases}$$

and the horizontal mixing length is $l_h = (\Delta x \Delta y)^{(1/2)}$.

The LES configuration/ideal case in WRF was set up to simulate a convective boundary layer in three dimensions. The convective boundary layer simulations were performed on a computational domain of $10km \times 10km \times 2km$ ($X \times Y \times Z$) with periodic boundary conditions in the horizontal directions. The grid resolution was made finer by changing to a typical LES resolution having, $dx=dy=50m$ and $dz=20m$, as opposed to $dx=dy=100m$ and $dz=50m$ in the default case. A finer grid provides an accurate LES flow field since it resolves more turbulent scales. A temperature inversion is placed at a height of $1km$ and the convection is driven by specifying a surface heat flux. Three cases were constructed to study the effect of background wind on the convective boundary layer, viz., Case 1: Basic case, no mean wind. Case 2: A mean wind of $10 m/s$. Case 3: A mean wind of $20 m/s$. Each simulation was run for a computational time (time it takes to compute) of two hours with a history interval that saves the results file every 15 minutes.

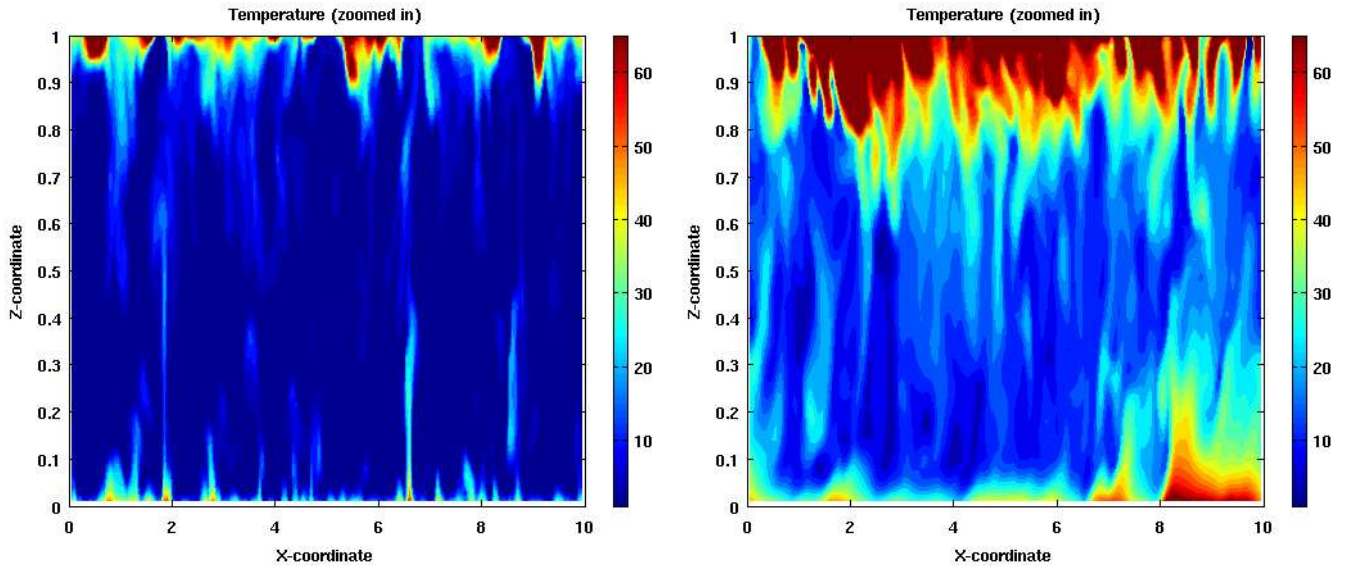


Figure 3.4: A comparison plot of the temperature (inversion), zoomed in on the bottom boundary by cutting a slice at $Y = 45$ for case1 - mean wind $0m/s$ (left) and case3 - mean wind $20m/s$ (right)

Figure ?? displays the variation in the w velocity and the potential temperature for Case 1, i.e. without any mean wind. The figure displays three slices of the computational domain, cut along the x-y, x-z and y-z planes. A drawback being, the interior of the domain is not visible which poses as a problem when one wants to analyze the physics inside the

domain. The planetary boundary layer (PBL) is the lowest layer of the atmosphere and extends from 100 to 3000 meters from the surface of the Earth. The boundary layer is directly influenced by the presence of the Earth's surface, responding to such forcings as frictional drag, solar heating, mean wind, etc. Each of these forcings generates turbulence of various-sized eddies, which can be as deep as the boundary layer itself, lying on top of each other. Consider the case of solar heating: solar heating of the ground on a sunny day creates thermals of warmer air that rise over colder air causing vertical mixing and turbulence. As observed, there are patches of negative velocity, the lower limit being $-2m/s$ and positive velocity, the upper limit being $4m/s$ along the x-z and y-z planes. In spite of there being no mean wind, this gives an indication of the mixing and turbulence, where the air with a higher and lower velocity, transfers vertically along the domain. Vertical mixing involves vertical distribution of three basic elements in the atmosphere, i.e., momentum, mass and heat. From figure ??, beyond a height of 1km, the velocity as well as the temperature remain approximately constant.

Thermally induced mixing is referred to as convection. The earth's surface heats the layer of air nearest the ground. A warm layer of air is now pinned beneath a cooler layer, resulting in an unstable atmosphere. In order to stabilize, the atmosphere must mix in the vertical and it does so by way of convection. Figures ?? and ?? depict a better understanding of the physics in terms of turbulence and mixing in the PBL. As observed from the figures, the turbulence and vertical mixing extend beyond 1000 meters from the surface of the Earth. Thermally induced mixing in the convective boundary layer is more evident in figure ?. When comparing Case 1 and Case 3, as the velocity increases, there is an occurrence of temperature inversion. A temperature inversion is a thin layer of the atmosphere where the normal decrease in temperature with height switches to the temperature increasing with height. As observed from the figure on the right, the temperature at the bottom of the computational domain is about 60 Kelvin, however, as the height increases, the temperature reduces and reverts back to approximately 60 Kelvin or higher. This can cause several weather-related effects. One is the trapping of pollutants below the inversion, allowing them to build up. Overall, the WRF-LES model handles cases related to the modelling of turbulence quite well and using options such as planetary boundary layer schemes can further improve the quality of the results. The features incorporated in WRF suggest that the WRF-LES model could serve as a powerful tool for turbulence modelling.

3.2 Sea Breeze Model in WRF

This section investigates the physics capabilities of WRF by studying the idealized (full physics) case known as the seabreeze model. A description of the physics models incorporated in WRF is provided as well. Modelling sea breezes remains a challenging task. Additional complexities such as the effects of terrain, land use and synoptic flow still ensure that sea breeze forecasting remains a challenging task. Presently, the majority of studies focus on the landward advance as it may have a significant impact on air quality, aviation and coastal city flow regimes.

A sea breeze is an example of a mesoscale front. It may extend across an entire coastline but may move inland only a few miles. A sea breeze is a local effect that occurs due to the temperature difference between land and water. The density of cooler air is greater than that of warmer air. Hence, the air pressure will be lower at the surface in the warm region due to the rising of this less dense air. During the day, the land is warmer than the ocean which is when a sea breeze occurs. Denser air over the ocean flows toward the less dense air over the land. The strength of the sea breeze is directly proportional to the temperature difference between the land and the sea. If the air temperature over the land sufficiently exceeds the air temperature over the sea, then a direct circulation results within the boundary layer of the atmosphere.

The idealized case for the seabreeze model in WRF has a computational domain of $404km \times 6km \times 20km$ ($X \times Y \times Z$) with periodic boundary conditions in the horizontal directions. Without altering the boundary conditions, the computational domain was extended in the Y direction and adding more grid points in the Z direction along with modifying the time step and introducing a mean wind. By adding more grid points along the Y co-ordinate and increasing the resolution, this makes the idealized case in WRF, originally a 2D case, a newly constructed 3D case. Land occupies 50 grid points in the middle of the domain. For a brief overview of the case, the reader is referred to the README.seabreeze file in the appendix. Two different cases are constructed by examining three different domain sizes. This was done by increasing the number of points in the y and z coordinates, i.e.: Case 1) Small domain size - 202pts \times 202pts \times 35pts, Case 2) Medium domain size - 202pts \times 202pts \times 50pts. Each case was run for a computational time of twelve hours. The different cases are compared using visualizations of the velocities.

There are various physics option available in WRF, each containing different choices depending on the case to be modelled. The physic categories in WRF are: 1) microphysics, 2) cumulus parameterization, 3) planetary boundary layer, 4) land-surface model, and 5) radiation. A short description of the schemes that were implemented in the sea breeze simulation case, have been presented in the following paragraphs.

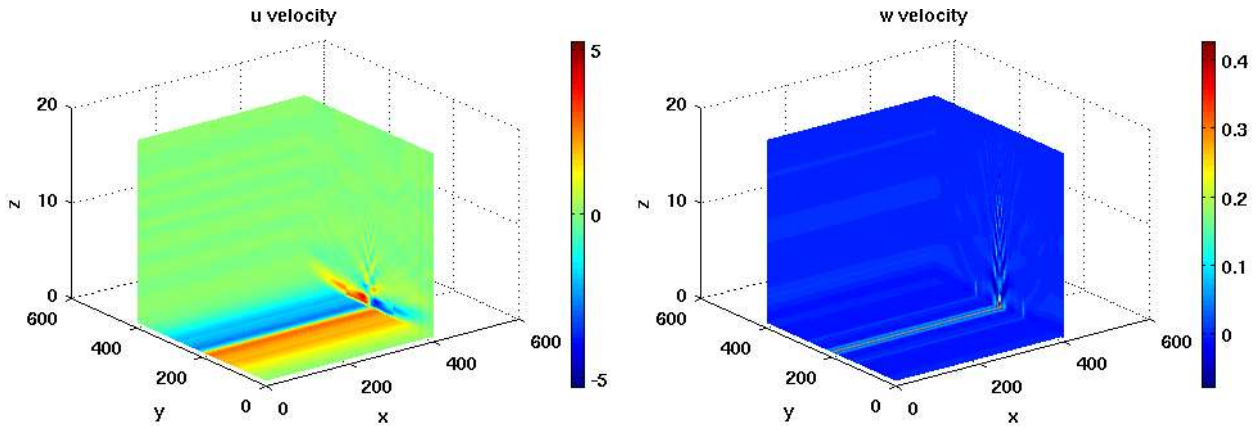


Figure 3.5: Plot showing the variation of u velocity and the w velocity for a horizontal wind of 0 m/s (case 1)

Microphysics includes explicitly resolved water vapour, cloud and precipitation processes. The model also accommodates any number of mass mixing ratio variables, and other quantities such as number concentrations [?]. Within the microphysics option, there are various options available indicating whether mixed-phase physics are included. Mixed-phase processes are those that result from the interaction of ice and water particles, such as riming (frost formed on cold objects by the rapid freezing of water vapour in cloud or fog) that produces graupel (soft hail or snow pellets) or hail. As per the documentation in WRF, for grid sizes less than 10km, mixed-phase schemes should be used, particularly in convective or icing situations. It is not worth it to implement these schemes for coarser grids, since riming is not likely to be well resolved. As per the idealized seabreeze case, the default scheme used to model the microphysics (`mp_physics = 2`) is the Purdue-Lin scheme that includes six classes of hydrometeors (an atmospheric phenomenon or entity involving water or water vapor, such as rain or a cloud). They are: water vapour, cloud water, rain, cloud ice, snow, and graupel. This is a relatively sophisticated microphysics scheme in WRF and is most suitable for use in research studies [?].

The radiation schemes provide atmospheric heating due to radiative flux divergence and surface downward longwave and shortwave radiation for the ground heat budget. Longwave radiation includes infrared or thermal radiation absorbed and emitted by gases and surfaces. Upward longwave radiative flux from the ground is determined by the surface emissivity that in turn depends upon land-use type, as well as the ground (skin) temperature. Shortwave radiation includes visible and surrounding wavelengths that make up the solar spectrum. Hence, the only source is the Sun, but processes include absorption, reflec-

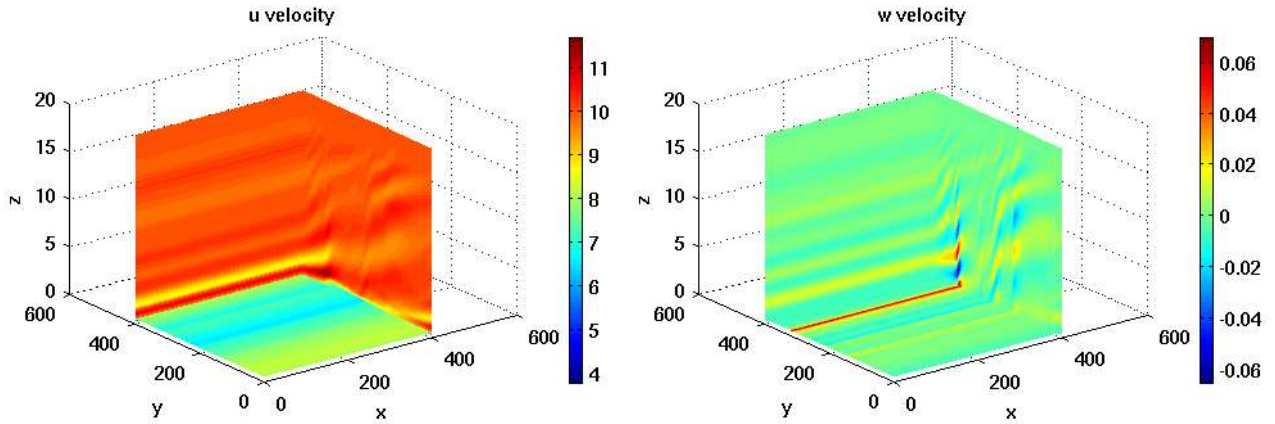


Figure 3.6: Plot showing the variation of u velocity and the w velocity for a horizontal wind of 10 m/s (case 2)

tion, and scattering in the atmosphere and at surfaces. All the radiation schemes in WRF currently are column (one-dimensional) schemes, so each column is treated independently, and the fluxes correspond to those in infinite horizontally uniform planes, which is a good approximation if the vertical thickness of the model layers is much less than the horizontal grid length [?].

The Rapid Radiative Transfer Model (RRTM) implemented to capture the longwave radiation (`ra_lw_physics = 1`) uses pre-set tables to accurately represent longwave processes due to water vapor, ozone, CO₂, and trace gases (if present), as well as accounting for cloud optical depth [?]. The molecular absorbers included are water vapor, carbon dioxide, ozone, methane, nitrous oxide, oxygen, nitrogen, and the halocarbons. It includes extinction from aerosols, clouds, and Rayleigh scattering [?].

The Dudhia shortwave radiation scheme (`ra_sw_physics = 1`) has a simple downward integration of solar flux, accounting for clear-air scattering, water vapor absorption, and cloud albedo and absorption [?]. It uses look-up tables for clouds. The WRF Version 3 scheme has an option to account for terrain slope and shadowing effects on the surface solar flux. Sloping and shadowing effects are turned on for the 2007 runs. For more information on this shortwave radiation scheme, see Dudhia (1989) [?].

From figure ?? and ??, one can observe the strip of land in the middle of the domain. Figure ??, shows the variation of the u and w velocity, without any mean wind, along the computational domain with slices cut along the x - y , y - z and z - x planes. Perceiving the w velocity (on the right) the velocity is maximum along the strip of land and minimum on

water. Along the z-x plane, the velocity does not show much variation and has a constant value of approximately $0m/s$. However, the x-y and y-z plane produce interesting results. Along the horizontal x-y plane, there is negative velocity, approximately $-2m/s$ on the left (225-400 points on the y axis) of the strip and positive velocity approximately, $2m/s$, on the right (175-0 points on the y axis) of the strip. Along the vertical y-z plane, however, there is a contradiction. There is positive velocity on the left (225-400 points on the y axis) of the land and negative velocity on the right (175-0 points on the y axis) of the land. This gives an indication of a circulation along the land boundary or along the coast, inside the computational domain. Noticing the x-y plane, the v velocity from the ocean is approaching one side of the land, while there is a negative velocity on the other side of the land.

Figure ?? shows the variation of the u and w velocity, with a mean wind at $10m/s$, along the computational domain with slices cut along the x-y, y-z and z-x planes. There is the distinctive strip of land along the x co-ordinate observed in the plots of the u and w velocity. One cannot predict any sort of a circulation with the effect of a mean wind.

Chapter 4

Conclusions

In this report, a basic understanding to the modelling of coastal basins is provided by discussing some essential coastal models that take into account tidal forces. Various physical factors, for example, Coriolis forces, could be included to make the model capture the physics accurately. However, a fundamental starting point, given in terms of the bathtub model, conveys some essential information and insight into various models that can be developed to account for different environmental phenomena. One would require accurate experimental and geographical data to model a real life case, which is hard to attain. There are many physical properties that are not represented by the system of shallow water equations represented by equations ???. These phenomena include: surface stresses, bottom drag, wave-breaking and viscosity. This is highly important for practical applications. Adding various extensions to the shallow water equations by keeping the numerical method efficient is one of the many challenges involved in developing a model.

The finite difference method is applied to model the governing shallow water equations using the leapfrog and Adams-Bashforth time stepping schemes. Filtering, in terms of dispersion, discards higher wavenumbers and allows part of the solution comprising of lower wavenumbers to be represented on the finite grid. The drawback using the standard set of shallow water equations is that they completely neglect dispersion which is a real world phenomenon observed in waves. The dispersion modified shallow water equations are a first step towards a more physically correct model. Including terms like surface stress and wave breaking would certainly contribute towards achieving a realistic solution. Alternative numerical methods, such as the finite volume method, that can handle discontinuities relatively well, can be implemented to resolve the physical features of the flow accurately. Of course, advanced numerical schemes could be implemented to increase the accuracy of the solution, however, there is a threshold beyond which every method would end up

producing a similar solution. A brief discussion about the MATLAB function `profiler`, provides the reader with some essential information about timing codes in MATLAB. This is a handy tool when one is interested in detecting sections in scripts, that end up consuming maximum time and acts as a guide to increase the efficiency of the program. When implemented on the 3-step Adams-Bashforth and leapfrog scheme it was observed that the 3-step Adams-Bashforth takes more amount of computational time compared to the leapfrog method. Similar results were obtained using both methods, however, there was no filtering applied to the leapfrog scheme. Due to a larger stencil size, the 3-step Adams Bashforth is a more accurate method as opposed to the leapfrog.

The results obtained from chapter ?? provide valuable information about the modelling capabilities of WRF. The unique WRF large eddy simulation model focuses on investigating turbulent structures inside the planetary boundary layer. The simulations indicate that large turbulent eddies are very efficient in transporting momentum, heat, and moisture within the boundary layer and between the planetary boundary layer and the layer above. Different models to capture sub-grid or sub-filter scale eddies are presented. Plots of energy spectra would be an effective technique to analyze the performance of different turbulence models. There are various planetary boundary layer schemes available in WRF that can be implemented and would be worth exploring. As far as the sea breeze model in WRF is concerned, there were a few challenges involved. Various physics options, numerical schemes implemented in the sea breeze case were discussed. There can be a few modifications that can be made to the ideal case. The width of the land can be extended by modifying the variables in the source code file, however, the entire case would have to be re-compiled. Nevertheless, having a larger surface of land next to the ocean would help analyze the physics. The temperature and densities of the land and ocean can be changed to study the case well. Overall, WRF is a valuable tool for the study of macro and micro scale weather related phenomena. Easily accessible data for real cases would be extremely helpful in understanding and forecasting day to day weather related developments.

APPENDICES

Appendix A

WRF Input Files

A.1 LES case - namelist file

The README.les file on winisk, about the idealized LES simulation reads as follows:

This test case produces a large-eddy simulation (LES) of a free convective boundary layer (CBL). The environmental wind (or the initial wind profile) is set to zero in this default case. The turbulence of the free CBL is driven/maintained by the surface heat flux, which is specified in the namelist file as ‘tke heat flux’=0.24 (in MKS units).

A random perturbation is imposed initially on the mean temperature field at the lowest four grid levels to kick off the turbulent motion. Double periodic boundary condition is used in X and Y. The default version uses a grid resolution of dx=dy=100m and dz=50m, which is considered to be rather coarse for an LES of the CBL. A typical grid mesh for an LES of the CBL is dx=dy=50m and dz=20m. An LES flow field is more accurate when the grid resolution is finer because it resolves more turbulent scales.

This LES version uses the Deardorff’s TKE scheme to compute the SGS eddy viscosity and eddy diffusivity for turbulent mixing, that is ‘diff opt’=2 and ‘km opt’=2 in the namelist. The Coriolis parameter is set to $f=10^{-4}/s$. It takes at least 30 minutes of simulation time to spin up the turbulent flow field; only after the spin-up, the turbulence inside the CBL is considered well established. A sign of well-established turbulence is that the total (i.e., the resolved-scale plus the subgrid-scale) heat flux profile should decrease linearly with height within the CBL.

To simulate a CBL with a mean wind, change the initial wind profile in the input sounding. When ‘pert coriolis’= true is set in the namelist, there is no need to include the

geostrophic wind terms in the right-hand sides of the u and v equations for LES's with non zero geostrophic wind. Note, parameterization constants, c_s and c_k in this namelist are different from the defaults and are the ones recommended to use with LES.

A sample of the namelist.input file in WRF for the LES configuration reads as follows:

```

&time_control
run_days           = 0,
run_hours          = 2,
run_minutes        = 00,
run_seconds        = 00,
start_year         = 0001, 0001, 0001,
start_month        = 01, 01, 01,
start_day          = 01, 01, 01,
start_hour         = 00, 01, 00,
start_minute       = 00, 30, 00,
start_second       = 00, 00, 00,
end_year           = 0001, 0001, 0001,
end_month          = 01, 01, 01,
end_day            = 01, 01, 01,
end_hour           = 01, 02, 00,
end_minute         = 00, 30, 00,
end_second         = 00, 00, 00,
history_interval_m = 15, 10, 1,
history_interval_s = 00, 00, 1,
frames_per_outfile = 1000, 1000, 1000,
restart            = .false.,
restart_interval_m = 60,
io_form_history    = 2
io_form_restart    = 2
io_form_input      = 2
io_form_boundary   = 2
debug_level        = 0
/

&domains
time_step          = 0,
time_step_fract_num = 3,

```



```

time_step_fract_den      = 10,
max_dom                  = 1,
s_we                     = 1,      1,      1,
e_we                     = 200,    100,    151,
s_sn                     = 1,      1,      1,
e_sn                     = 200,    100,    151,
s_vert                  = 1,      1,      1,
e_vert                  = 100,    100,    41,
dx                       = 50,     50,     16.6667,
dy                       = 50,     50,     16.6667,
ztop                    = 2000,    2000,    2000,
grid_id                  = 1,      2,      3,
parent_id                = 0,      1,      2,
i_parent_start           = 0,      10,     15,
j_parent_start           = 0,      10,     15,
parent_grid_ratio        = 1,      3,      3,
parent_time_step_ratio   = 1,      3,      3,
feedback                 = 0,
smooth_option            = 0
/

&physics
mp_physics               = 0,      0,      0,
ra_lw_physics            = 0,      0,      0,
ra_sw_physics            = 0,      0,      0,
radt                     = 0,      0,      0,
sf_sfclay_physics       = 1,      1,      1,
sf_surface_physics      = 0,      0,      0,
bl_pbl_physics           = 0,      0,      0,
bldt                     = 0,      0,      0,
cu_physics               = 0,      0,      0,
cudt                     = 0,      0,      0,
isfflx                   = 2,
ifsnow                   = 0,
icloud                   = 0,
num_soil_layers          = 5,
mp_zero_out              = 0,
/

```

```

&fdda
/

&dynamics
rk_ord           = 3,
diff_opt         = 2,
km_opt          = 2,
damp_opt        = 0,
zdamp           = 15000., 5000., 5000.,
dampcoef        = 0.1, 0.2, 0.2
khdif           = 1., 1., .05,
kvdif           = 1., 1., .05,
c_s             = 0.18
c_k             = 0.10
mix_isotropic   = 1
smdiv           = 0.1, 0.1, 0.1,
emdiv           = 0.01, 0.01, 0.01,
epssm          = 0.1, 0.1, 0.1
tke_heat_flux  = 0.24, 0.24, 0.24,
time_step_sound = 6, 6, 6,
h_mom_adv_order = 5, 5, 5,
v_mom_adv_order = 3, 3, 3,
h_sca_adv_order = 5, 5, 5,
v_sca_adv_order = 3, 3, 3,
mix_full_fields = .true., .true., .true.,
non_hydrostatic = .true., .true., .true.,
pert_coriolis   = .true., .true., .true.,
/

&bdy_control
periodic_x      = .true., .false., .false.,
symmetric_xs    = .false., .false., .false.,
symmetric_xe    = .false., .false., .false.,
open_xs         = .false., .false., .false.,
open_xe         = .false., .false., .false.,
periodic_y      = .true., .false., .false.,
symmetric_ys    = .false., .false., .false.,

```

```

symmetric_ys = .false.,.false.,.false.,
open_ys      = .false.,.false.,.false.,
open_ys      = .false.,.false.,.false.,
nested       = .false., .true., .true.,
/

&grib2
/
&namelist_quilt
nio_tasks_per_group = 0,
nio_groups = 1,
/

```

A.2 Sea breeze case - namelist file

The README.seabreeze file on winisk, about the idealized seabreeze simulation seabreeze_2d_x reads as follows:

The purpose of this case is to demonstrate how one can set up all land variables in order to use a full-physics set-up in an idealized case. This test case is an attempt to produce a two dimensional sea breeze simulation. Configuration needs tuning to produce desirable results, as the current settings give a very shallow sea breeze.

The input sounding has no wind. There is no Coriolis [grid f(i,j)=0]. There is a diurnal cycle and the latitude and longitude are set for radiation to work. The routine initializing this case is dyn_em/module_initialize_seabreeze2d_x.F. Note that since the longitude is set to zero, start_hour in the namelist is the local time as well as the UTC time (5Z in the namelist). For other longitudes the start_hour refers to UTC time. The land-surface fields are filled so that the slab, Noah or RUC LSMs can be used.

This setup is for a 2D case with 202 grid points in x. The land occupies 50 grid points in the middle of the domain. The width of the land can be changed by modifying variable lm (half width for land points) in dyn_em/module_initialize_seabreeze2d_x.F.

A sample of the namelist.input file in WRF for the seabreeze model reads as follows:

```
&time_control
run_days           = 0,
run_hours          = 12,
run_minutes        = 0,
run_seconds        = 0,
start_year         = 2007,
start_month        = 06,
start_day          = 1,
start_hour         = 5,
start_minute       = 00,
start_second       = 00,
end_year           = 2007,
end_month          = 06,
end_day            = 2,
end_hour           = 5,
end_minute         = 00,
end_second         = 00,
history_interval   = 15,
frames_per_outfile = 1000,
restart            = .false.,
restart_interval   = 360,
io_form_history    = 2
io_form_restart    = 2
io_form_input      = 2
io_form_boundary   = 2
debug_level        = 0
/

&domains
time_step          = 10,
time_step_fract_num = 0,
time_step_fract_den = 1,
max_dom            = 1,
s_we               = 1,
e_we               = 202,
s_sn               = 1,
```

```

e_sn                = 202,
s_vert              = 1,
e_vert              = 50,
dx                  = 2000,
dy                  = 2000,
ztop                = 20000.,
/

&physics
mp_physics          = 2,
ra_lw_physics       = 1,
ra_sw_physics       = 1,
radt                = 5,
sf_sfclay_physics  = 1,
sf_surface_physics = 1,
bl_pbl_physics      = 1,
bldt                = 0,
cu_physics          = 0,
cudt                = 0,
isfflx              = 1,
ifsnow              = 0,
icloud              = 0,
num_soil_layers     = 5,
mp_zero_out         = 0,
/

&fdda
/

&dynamics
rk_ord              = 3,
diff_opt            = 1,
km_opt              = 4,
damp_opt            = 2,
dampcoef            = .003,
zdamp               = 5000.,
khdif               = 300,
kvdif               = 1,

```

```

smdiv                = 0.1,
emdiv                = 0.01,
epssm               = 0.1,
time_step_sound     = 6,
h_mom_adv_order     = 5,
v_mom_adv_order     = 3,
h_sca_adv_order     = 5,
v_sca_adv_order     = 3,
mix_full_fields     = .true.,
non_hydrostatic     = .true.,
/

&bdy_control
periodic_x          = .true.,
symmetric_xs       = .false.,
symmetric_xe       = .false.,
open_xs            = .false.,
open_xe            = .false.,
periodic_y          = .true.,
symmetric_ys       = .false.,
symmetric_ye       = .false.,
open_ys            = .false.,
open_ye            = .false.,
/

&grib2
/

&namelist_quilt
nio_tasks_per_group = 0,
nio_groups = 1,
/

```