Two-layer ocean flow over a bump topography

Jostein Brændshøi

Thesis submitted for the degree of
Master of science in Meteorology and Oceanography
60 Credits

Department of Geosciences
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

June 1, 2018

Abstract

The flow response to a bump topography in a two-layer non-linear quasi-geostrophic fluid on the β-plane, has been studied using numerical simulations looking at both linear and turbulent runs. The system was initialized with a baroclinically unstable surface trapped wave above a resting lower layer and topographic effects with regards to deep energy transfer and consequently lower layer spin-up, was examined. In particular, the response was diagnosed for parameters such as topographic height, wavenumber, i.e. both the vertical and horizontal scale of the bumps, and friction. In order to study a vast range of parameters, a simulation framework was developed permitting automated systematic large scale parameter variation through parallel execution of independent simulations on an improvised computational cluster consisting of a set of desktop workstations. To determine the degree of deep energy transfer in all these simulations, the fraction of total energy at the end of the simulations to the initial total energy, was used as a measure of dissipation and thus deep energy transfer (friction only in lower layer). Dissipation dependence on a large range of parameters was determined. Once interesting parameter trends were identified, closer inspection of the associated flow fields was conducted. Among the results were that the deep transfer depends non-monotonically on bump height for small friction in both the linear and non-linear system, i.e. moderate dissipation at small heights, a blocking effect at large heights and a dissipation maximum for intermediate values strongly dependent on the closing of q_s contours facilitating bump flow as seen in earlier studies. Thus bump height exhibit similar behaviour to friction as described in earlier studies (and this one). The main contributor of linear deep transfer was when the horizontal topography scale resembles the wave x-scale combined with closed q_s contours. Large and many bumps were found to decouple the layers and (in the non-linear system) stabilise the baroclinically unstable surface wave for a significant amount of time, but the flow still eventually went turbulent giving deep transfer, only later and weaker.

Acknowledgements

First and foremost I would like to greatly thank my main supervisor Joseph H. LaCasce. Thank you for the opportunity that was this interesting and challenging project. I really appreciate your great guidance as well as patience and positive attitude in answering my many questions. I have learned a lot from you and even when we were separated by the Atlantic ocean and the entire US during the final semester, we had numerous email and Skype exchanges which thought me new things each time.

I would also like to express gratitude to my co-supervisor Pål Erik Isachsen for always being up for discussing any questions I had. From that I learned a lot.

Additionally, thanks a lot to thank Anne Claire Fouilloux for assisting me with everything Fortran and to Kjell Andresen for helping me with Bash scripting. Both of your expertise has provided valuable input for my work!

I appreciate the time together with my lovely fellow students and friends at MetOs. We have shared some great conversations, both socially and academically, and it has been interesting learning about your projects as well. Thank you!

Finally, and at the core, I am massively grateful for the support of my family. It means a lot!

Contents

1 Introduction **1**
- 1.1 Background 1
- 1.2 Description of this study 2

2 The model **4**
- 2.1 Deriving the equations of motion 5
  - 2.1.1 Layer equations 6
  - 2.1.2 Barotropic and baroclinic equations 8
  - 2.1.3 The non-dimensional equations 9
- 2.2 Updating the model 10
- 2.3 Implementing topography 11
  - 2.3.1 Theory 11
  - 2.3.2 Numerics 12
- 2.4 Model configuration and initial condition 14
  - 2.4.1 Parameters 14
  - 2.4.2 Initial condition 15
- 2.5 Grid resolution analysis 17

3 Simulation framework **21**
- 3.1 Parallel computation 21
  - 3.1.1 Local parallelism 22
  - 3.1.2 Cluster parallelism 23
- 3.2 Implementation 26

4 Results & discussion **30**
- 4.1 Introduction 30
- 4.2 Frictional effects 31
- 4.3 Topography: Linear simulations 33
  - 4.3.1 Response to topographic amplitude 33
  - 4.3.2 Response to topographic wavenumber 37
- 4.4 Topography: Non-linear simulations 38
- 4.5 Large topography runs 40
  - 4.5.1 Flow response 43
  - 4.5.2 Topographic suppression of deep turbulent transfer 46

5 Summary & Conclusions **50**
- 5.1 Future work 51

A Additional mathematics **54**
A.1 Layer equations to barotropic equation 54
A.2 Layer equations to baroclinic equation 55

Chapter 1

Introduction

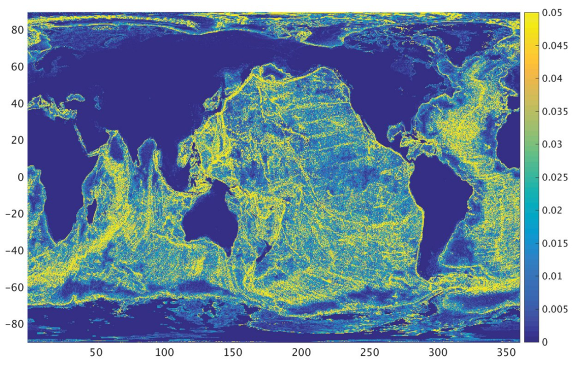
1.1 Background

Geophysical fluid dynamics applied to the oceans has been studied for a long time period, however, relatively speaking, not all that much attention has been given to the impact of bottom topography on the flow. And while a flat-bottom approximation may be analytically useful, the ocean floors are most definitely not flat. The bottom is scattered with topographic features such as continental slopes, seamounts, ridges and trenches, bumps and depressions as well as other irregularities much like the topography on land. Known examples include the Mariana Trench, the Mid-Atlantic Ridge and the Lofoten Basin by the coast of Norway. Global topographic slopes in figure 1.1(a) calculated by de La Lama et al. [2016] from ETOPO1 data, give a realistic view of the bathymetry of the oceans.[1] Rarely are the slopes flat and the ocean floors varies essentially everywhere. Topographic effects caused by uneven bottoms such as those described above, are of great importance to many oceanic flows and therefore worth studying. See figure 1.1(b) for an example. Note here the strong flow along the continental margin as well as the clockwise flow in the center of the basin. In both these cases the fluid appears to move nearly parallel to the isobaths (contours of constant depth) and it is evident that the overall motion is significantly affected by bathymetry.

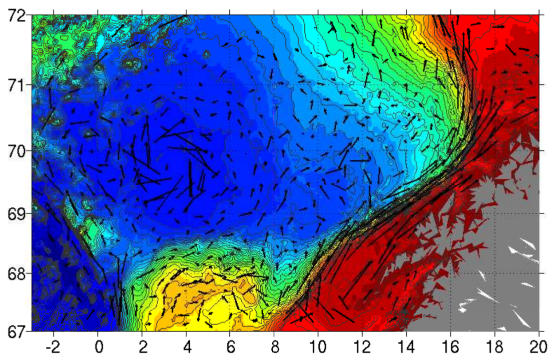
Much ocean theory in the past, though, has ignored bathymetry, treating the bottom as a smooth, flat surface. The general thinking was that deep motion is weak, so the topographic influence is probably also weak. For example, the models of Stommel [1948], Munk [1950], Stommel and Arons [1960], Anderson and Gill [1975] and Fofonoff [1954] all assumed a flat bottom. However, recent work (e.g. de La Lama et al. [2016] and LaCasce [2017]) suggests topography influences the ocean response throughout the water column further and significantly the vertical structure of the flow. The more recent findings have motivated further study of topographic effects in order to help uncover the physics at work. The present study attempts to contribute to this area.

A study of particular of interest for this work is LaCasce and Pedlosky [2004]. The ocean responds to changes in atmospheric forcing via the planetary (Rossby) waves, which propagate westward across the basin [Anderson and Gill, 1975]. These have scales of

[1]The terminology "topography" and "bathymetry" both refer to the ocean floor in the contex of this study. The former will predominantly be used going forward.



(a) Topographic slope from ETOPO1 data, calculated at the product resolution 0.0167°. The axis are latitude and longitude.



(b) Mean velocities estimated from surface drifters in the Lofoten Basin west of Norway. The color contours indicate the water depth and the axis are latitude and longitude.

Figure 1.1: (a) from de La Lama et al. [2016] and (b) from Koszalka et al. [2011] with caption from LaCasce [2018, sec. 4.2]

hundreds to thousands of kilometers and are clearly observed in satellite measurements of sea surface height (SSH) [Chelton and Schlax, 1996]. LaCasce and Pedlosky [2004] suggested these waves are unstable, breaking into smaller, deeper eddies. This would have an enormous impact on oceanic adjustment, making it much more turbulent. But the authors employed a flat bottom in their study. This leads to the study of this thesis.

1.2 Description of this study

We have, in part, revisited the study by LaCasce and Pedlosky [2004], using the same numerical model, solving the two-layer quasi-geostrophic equations, but implementing a non-uniform bottom. We have examined to what extent the introduced topography alters the stability, and how the wave propagation is affected as well as the general vertical structure of the flow. Our hypothesis regarding this is that surface-trapped waves will be stabilized, and successfully cross the basin.

However, our study is not purely focused on the above, but also concerns generally adding to the knowledge of topographically affected flows and comparing with results obtained by others. This requires elaboration: We have employed a two-dimensional bump-like topography in our studies. In particular, we focused on the topographies

$$h_B(x,y) = h_0 \cos(k_t x)\cos(l_t y) \tag{1.1}$$

$$h_B(x,y) = h_0 \sin(k_t x)\sin(l_t y) \tag{1.2}$$

where $h_B(x,y)$ denotes the topographic height above the resting depth of the fluid, h_0 the maximum/minimum amplitude and k_t and l_t the topographic wavenumbers in the x- and y-direction. We were interested in the flow response as a function of h_0 and k_t, l_t (we always used $k_t = l_t$), i.e. how the flow reacts to both the horizontal and vertical scale of the bumps. We examine the transfer of energy to the lower layer and subsequent spin-up as a function of the mentioned parameters (we also examined friction). This is our main topic of investigation.

We conducted our investigation using numerical simulations to examine the flow in a two-layer quasi-geostrophic system using a model written by J. H. LaCasce (used in LaCasce and Pedlosky [2004]). Especially important to us is the two-layer structure of the system; this is the simplest configuration that yields a baroclinic structure. As one of our main aspirations is to study the vertical structure of the system's time-evolution, this is an essential feature of the simplified system. We initialized the upper layer flow with a baroclinically unstable wave similar to that by LaCasce and Pedlosky and the lower layer from rest. Even though the two-layer quasi-geostrophic system is quite idealized compared to the real oceans, it has been popular in many earlier studies and is believed to still provide useful insights into the dynamics at work.

A somewhat unique (at least as far as the author is aware) aspect of study is the approach taken to diagnose the system. In order to get a broad view of the two-layer response, we wanted to test a wide range of values for the relevant parameters. In particular, simulate the flow for a vast set of h_0 and k_t, l_t values and combinations of these, preferably several hundreds or even thousands. This means performing a massive number of model runs which is both tedious and in fact physically impossible manually (the model itself is not parallel meaning large runtimes overall). This motivated the necessary development of a simulation framework for both automating parameter variation and enabling executing independent model runs in parallel on a set desktop computers available at the author's university. This greatly enhanced our abilities to examine the topographic response and had a fundamental impact on the results we were able to produce.

The thesis is divided into four further chapters. Chapter 2 concerns the numerical model we have updated, extended and used. Therein, we derive the equations of motion, discuss the performed update and extension, present our model setup (parameters and initial conditions) and conduct an analysis for choice of grid resolution. Chapter 3 discusses the ideas behind, and development of, the simulation framework that we used in conjunction with the Fortran model to execute our numerical computations. Chapter 4 presents the results generated from application of said model and framework. Here we show and discuss both the linear and non-linear simulations. Finally, chapter 5 summarizes the work done and the results produced while offering some further discussion and conclusions as well as suggestions for what could be done in the future if one were to extend upon the work in this thesis. At the very end is an appendix containing some mathematics done for some derivations in Chapter 2.

Chapter 2

The model

We utilised numerical simulations to study the effect of the bottom topography. In particular, a model solving the non-dimensional two-layer quasi-geostrophic potential vorticity (QGPV) equations was employed. This model was written in Fortran 77 by J. H. LaCasce and has been used in some of his earlier work, e.g. LaCasce and Pedlosky [2004] were they examined the instability of Rossby waves as they propagate across basins.

The model solves the full non-linear equations in a square $\pi \times \pi$ (non-dimensional) basin domain (see figure 2.1) for the upper and lower layer streamfunctions ψ_1 and ψ_2 using no-normal-flow boundary conditions, i.e.

$$u_1(0, y, t) = u_1(\pi, y, t) = u_2(0, y, t) = u_2(\pi, y, t) = 0 \tag{2.1}$$

$$v_1(x, 0, t) = v_1(x, \pi, t) = v_2(x, 0, t) = v_2(x, \pi, t) = 0 \tag{2.2}$$

where $\mathbf{u}_1(x, y, t) = (u_1, v_1)$ and $\mathbf{u}_2(x, y, t) = (u_2, v_2)$ are the 2D fluid flow fields in the upper and lower layer, respectively. The model solves the QGPV equations in terms of the barotropic ψ_B and baroclinic ψ_T modes defined by

$$\psi_B = \delta_1\psi_1 + \delta_2\psi_2 \tag{2.3}$$

$$\psi_T = \psi_1 - \psi_2 \tag{2.4}$$

where

$$\delta_1 = \frac{H_1}{H_0}, \qquad \delta_2 = \frac{H_2}{H_0} \tag{2.5}$$

are the layer thickness ratios with H_1 and H_2 being the upper and lower layer resting depth, respectively, and $H_0 = H_1 + H_2$. By rearrangement and substitution, (2.3) and (2.4) quivalently states

$$\psi_1 = \psi_B + \delta_2\psi_T \tag{2.6}$$

$$\psi_2 = \psi_B - \delta_1\psi_T \tag{2.7}$$

which is useful to have in mind for later.

The boundary models boundary condition is invoked on the surface elevation by requiring $\psi_B = 0$ along all four boundaries on the interface through the integral condition

$$\frac{\partial}{\partial t}\int_0^L\int_0^L \psi_T \; dxdy = 0 \tag{2.8}$$

which ensures mass conservation by allowing the interface to move up and down as a function of time, but have the same value along all boundaries at any given time (see e.g. LaCasce [2000, sec. 4] for details).

LaCasce [2002] used a single-layer version of this model and as noted there, the model uses finite differences for the spatial derivatives and Fourier transformations to obtain the streamfunctions from the time-stepped vorticity. A third-order Adams-Bashfort scheme is used for time-stepping. For advection, the model employs the Quadratic Upstream Interpolation for Convective Kinematics (QUICK) scheme. In particular, a two-dimensional version of that in Leonard [1979] (also discussed in Ferziger and Peric [1999, sec. 4.4]). This third-order scheme has fairly good accuracy, but some dissipation at small scales causing energy loss. The effects of this is discussed in more detail in section 2.5.

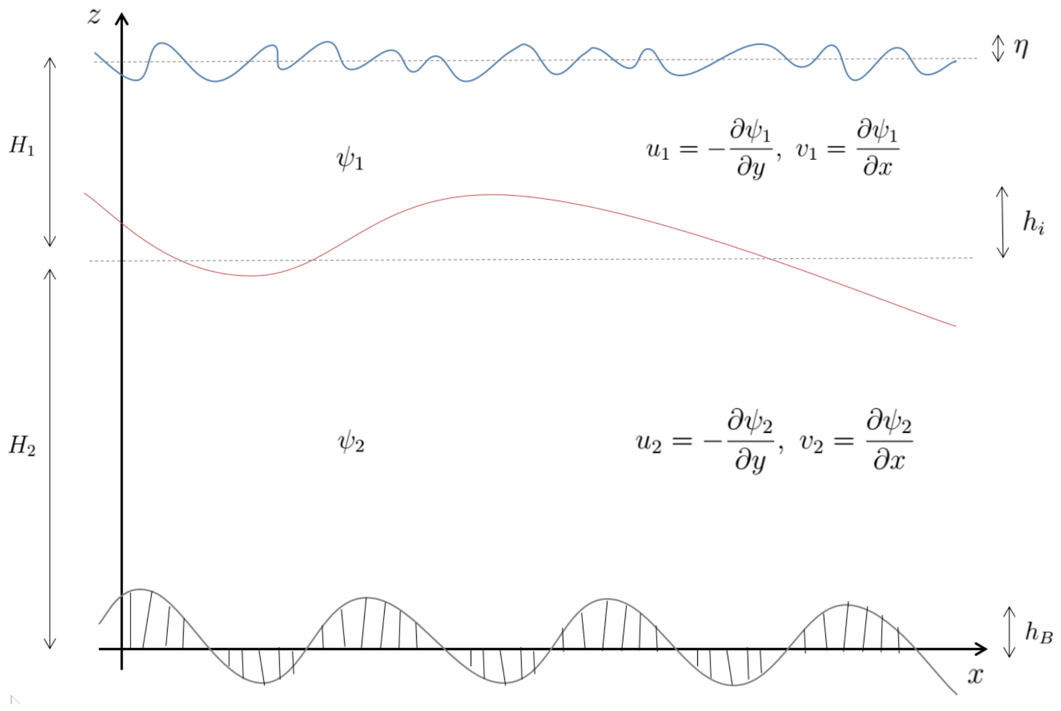

z
η
H_1
ψ_1
$u_1 = -\frac{\partial \psi_1}{\partial y},\ v_1 = \frac{\partial \psi_1}{\partial x}$
h_i
H_2
ψ_2
$u_2 = -\frac{\partial \psi_2}{\partial y},\ v_2 = \frac{\partial \psi_2}{\partial x}$
h_B
x

Figure 2.1: Sketch of a cross-section in the (x, z)-plane in the two-layer domain. The y-axis points inwards.

Furthermore, the model assumes a flat ocean floor and thus there is no support for bottom topography. Therefore, modifying the model to include topography became an important part of the work in this project and section 2.3 is dedicated to describing this process. However, before we could implement topography, we needed to see how the equations of motion were altered by the inclusion of topography We start by showing this. We end the chapter by discussing our parameter setup (including initial condition) and various considerations made related to the models spatial grid resolution.

2.1 Deriving the equations of motion

The model does not solve the the two-layer QGPV equations directly, but for mathematical and numerical reasons, formulated in terms of the barotropic and baroclinic modes instead. We now derive the two-layer equations with bottom topography, then the corresponding barotropic and baroclinic equations together with their non-dimensional versions which are the ones solved by the model.

2.1.1 Layer equations

To obtain the two-layer quasi-geostrophic equations, we start at the shallow water potential vorticity equation [Vallis, 2006]. We have

$$\frac{Dq}{Dt} = \left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right) q = \mathcal{F} \tag{2.9}$$

where $\mathcal{F}(x, y, t)$ is some forcing (from wind stress and/or friction) and the potential vorticity q is defined as

$$q = \frac{\zeta + f}{H} = \frac{\zeta + f_0 + \beta y}{H_0 - h} \tag{2.10}$$

with $f = f_0 + \beta y$ as the coriolis parameter using the standard β-plane approximation, H is the total depth of the fluid, h is the height deviation from the total depth (e.g. due to surface elevation, bottom topography and/or layer interface as we shall see below) and $\zeta = \partial v/\partial x - \partial u/\partial y$ is the relative vorticity. As is common and appropriate for the ocean interior, we assume the bottom topography (or interface height) to be very small compared to the total depth of the fluid, i.e. $h \ll H_0$. Given this we note that the following approximation

$$\frac{1}{1 - h/H_0} \approx 1 + \frac{h}{H_0}$$

can be made using the geometric Taylor series expansion. Knowing this, we may write (2.10) as

$$\begin{aligned} q &= \frac{f_0}{H_0}\frac{1 + \zeta/f_0 + \beta y/f_0}{1 - h/H_0} \approx \frac{f_0}{H_0}\left(1 + \frac{\zeta}{f_0} + \frac{\beta y}{f_0}\right)\left(1 + \frac{h}{H_0}\right) \\ &= \frac{f_0}{H_0} + \frac{f_0 h}{H_0^2} + \frac{\zeta}{H_0} + \frac{\zeta h}{H_0^2} + \frac{\beta y}{H_0} + \frac{\beta h y}{H_0^2} \\ &\approx \frac{f_0}{H_0} + \frac{f_0 h}{H_0^2} + \frac{\zeta}{H_0} + \frac{\beta y}{H_0} \end{aligned}$$

where, in the final step, the two terms that were product of two small terms have been dropped. This is possible due to the standard quasi-geostrophic assumption of assuming a small Rossby number and that $\beta y \ll f_0$. Substituting this expression for q back into (2.9) as well as multiplying through by H_0 gives

$$\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right)\left(\zeta + \beta y + \frac{f_0}{H_0}h\right) = \mathcal{F} \tag{2.11}$$

which is the QGPV equation we will use onwards. The following derivation is a simplified two-layer specific version of the general multi-layer consideration in Pedlosky [1987, sec. 6.16]. We now move to apply (2.11) separately for both layers in figure 2.1. Expressing the potential vorticity content of right parenthesis in (2.11) in each layer for each layer yields the PV's (really PV multiplied by H_1, H_2 depending on the layer)

$$q_1 = \zeta_1 + \beta y - \frac{f_0}{H_1}(\eta - h_i) \tag{2.12}$$

$$q_2 = \zeta_2 + \beta y - \frac{f_0}{H_2}(h_i - h_B) \tag{2.13}$$

where $\eta(x, y, t)$ is the surface elevation, $h_i(x, y, t)$ the interface height, $h_B(x, y)$ the bottom topography height and H_1, H_2 are the two layer equilibrium thicknesses. Considering the horizontal velocities are approximately non-divergent as shown formally in [Pedlosky, 1987, sec. 3.12], we may introduce streamfunctions ψ_1 and ψ_2 for the flow in each layer as

$$\mathbf{u}_1 = \mathbf{k} \times \frac{g}{f_0}\nabla\eta = \mathbf{k} \times \nabla\psi_1 \tag{2.14}$$

$$\mathbf{u}_2 = \mathbf{k} \times \nabla\left(\frac{g}{f_0}\eta - \frac{g'}{f_0}h_i\right) = \mathbf{k} \times \nabla\psi_2 \tag{2.15}$$

with

$$\psi_1 = \frac{g}{f_0}\eta \qquad \text{and} \qquad \psi_2 = \frac{g}{f_0}\eta + \frac{g'}{f_0}h_i \tag{2.16}$$

while $\mathbf{u}_1(x, y, t)$ and $\mathbf{u}_2(x, y, t)$ is the 2D velocity field in the upper and lower layer, respectively, and g' is the reduced gravity. We may rewrite (2.16)

$$\eta = \frac{f_0}{g}\psi_1 \qquad \text{and} \qquad h_i = \frac{f_0}{g'}(\psi_2 - \psi_1)$$

and further substitution into (2.12) and (2.13) yields

$$q_1 = \nabla^2\psi_1 + \beta y + F_1(\psi_2 - \psi_1) - G\psi_1 \tag{2.17}$$

$$q_2 = \nabla^2\psi_2 + \beta y + F_2(\psi_1 - \psi_2) + \frac{f_0}{H_2}h_B \tag{2.18}$$

where we have used the connection to relative vorticity

$$\zeta_i = \frac{\partial v_i}{\partial x} - \frac{\partial u_i}{\partial y} = \frac{\partial^2\psi_i}{\partial x^2} + \frac{\partial^2\psi_i}{\partial y^2} = \nabla^2\psi_i \tag{2.19}$$

and that

$$u_i = -\frac{\partial\psi_i}{\partial y}\ , \quad v_i = \frac{\partial\psi_i}{\partial x} \tag{2.20}$$

for $i \in \{1, 2\}$ and also defined the constants

$$F_1 = \frac{f_0^2}{g'H_1}\ , \qquad F_2 = \frac{f_0^2}{g'H_2}\ , \qquad G = \frac{f_0^2}{gH_1}. \tag{2.21}$$

Next we apply the rigid lid assumption; in (2.17), we observe that the term $F_1(\psi_2 - \psi_1) \gg G\psi_1$ due to $g \gg g'$ and the layer interface height h_i being much larger than the surface elevation η, and thus drop the $G\psi_1$ term. Inserting the PV (2.17) and (2.18) for each layer in turn into (2.11) gives the two equations

$$\left(\frac{\partial}{\partial t}+\mathbf{u}_1\cdot\nabla\right)\left[\nabla^2\psi_1+\beta y+F_1(\psi_2-\psi_1)\right]=\mathcal{F}_1 \tag{2.22}$$

$$\left(\frac{\partial}{\partial t}+\mathbf{u}_2\cdot\nabla\right)\left[\nabla^2\psi_2+\beta y+F_2(\psi_1-\psi_2)+\frac{f_0}{H_2}h_B\right]=-r\nabla^2\psi_2 \tag{2.23}$$

where we have specified the forcing function in the lower layer as a Rayleigh friction on the lower layer vorticity, i.e. $\mathcal{F}_2(x,y,t)-r\nabla^2\psi_2$, and left the upper layer forcing as some general wind forcing function $\mathcal{F}_1(x,y,t)$. We may rewrite (2.22) and (2.23) equivalently in terms of Jacobian operators as

$$\frac{\partial}{\partial t}\left[\nabla^2\psi_1+F_1(\psi_2-\psi_1)\right]+\mathcal{J}(\psi_1,\nabla^2\psi_1)+\mathcal{J}(\psi_1,F_1\psi_2)+\beta\frac{\partial\psi_1}{\partial x}=\mathcal{F}_1 \tag{2.24}$$

$$\begin{aligned}&\frac{\partial}{\partial t}\left[\nabla^2\psi_2+F_1(\psi_1-\psi_2)\right]+\mathcal{J}(\psi_2,\nabla^2\psi_2)+\mathcal{J}(\psi_2,F_2\psi_1)\\&\qquad+\beta\frac{\partial\psi_2}{\partial x}+\mathcal{J}\left(\psi_2,\frac{f_0}{H_2}h_B\right)=-r\nabla^2\psi_2\end{aligned} \tag{2.25}$$

where we have isolated the β-term while the self-advection vanishes in each layer and the Jacobian operator is defined as

$$\mathcal{J}(f,g)=\frac{\partial f}{\partial x}\frac{\partial g}{\partial y}-\frac{\partial g}{\partial x}\frac{\partial f}{\partial y}. \tag{2.26}$$

Equations (2.24) and (2.25) (or (2.22) and (2.23)) constitute the layer equations governing the two-layer flow and are the dimensional versions of (6.16.34a,b) in [Pedlosky, 1987, p. 423] . These are prognostic non-linear third-order partial differential equations for the upper and lower layer streamfunctions ψ_1 and ψ_2, respectively. We observe that the time evolution of the streamfunctions is impacted by interfacial motion, self advection of relative vorticity, advection by the other layer flow as well as the planetary vorticity gradient. Additionally, the lower layer flow feels the topography and bottom friction while the upper layer may be subjected to a wind forcing of some kind.

2.1.2 Barotropic and baroclinic equations

Now that we have the layer equations, we are ready to construct the barotropic and baroclinic equations that are solved for in the model. Recall the baroptropic and baroclinic streamfunctions in (2.3) and (2.4). In order to get a prognostic equation for ψ_B we multiply (2.24) by δ_1 and (2.25) by δ_2 and then add them together. This process requires quite a bit of calculations and was done, but the details are left in appendix A.1. These calculations result in the barotropic equation

$$\begin{aligned}&\frac{\partial q_B}{\partial t}+\mathcal{J}(\psi_B,q_B)+\delta_1\delta_2\mathcal{J}(\psi_T,q_T)+\beta\frac{\partial\psi_B}{\partial x}\\&\qquad+\delta_2\mathcal{J}\left(\psi_B-\delta_1\psi_T,\frac{f_0}{H_2}h_B\right)=\delta_1\mathcal{F}_1-\delta_2 r\nabla^2(\psi_B-\delta_1\psi_T)\end{aligned} \tag{2.27}$$

where

$$q_B = \nabla^2 \psi_B \tag{2.28}$$

$$q_T = \nabla^2 \psi_T - F\psi_T \tag{2.29}$$

is the barotropic and baroclinic potential vorticity, respectively, and

$$F = F_1 + F_2 = \frac{f_0^2}{g'H_1} + \frac{f_0^2}{g'H_2} = \frac{f_0^2(H_2 + H_1)}{g'H_1H_2} \tag{2.30}$$

is the combined interface constant (Burger number).

We constructed the baroclinic equation analogously; we subtract (2.25) from (2.25) and do a similar term-by-term procedure with the details again left in appendix A.2. Going though these calculations yields the baroclinic equation

$$\begin{aligned}\frac{\partial q_T}{\partial t} &+ \mathcal{J}(\psi_B, q_T) + \mathcal{J}(\psi_T, q_B) + (\delta_2 - \delta_1)\mathcal{J}(\psi_T, q_T) \\ &+ \beta\frac{\partial \psi_T}{\partial x} - \mathcal{J}\left(\psi_B - \delta_1\psi_T, \frac{f_0}{H_2}h_B\right) = \mathcal{F}_1 + r\nabla^2(\psi_B - \delta_1\psi_T)\end{aligned} \tag{2.31}$$

2.1.3 The non-dimensional equations

Having found how topography modifies the layer and barotropic/baroclinic equations, we now use the results from above to formulate the non-dimensional versions of the equations needed for the actual implementation in the model. We scale the variables by

$$x' = \frac{x}{L} \qquad \Rightarrow \qquad x = Lx' \tag{2.32}$$

$$y' = \frac{y}{L} \qquad \Rightarrow \qquad y = Ly' \tag{2.33}$$

$$t' = t\frac{U}{L} \qquad \Rightarrow \qquad t = \frac{L}{U}t' \tag{2.34}$$

$$\psi_B' = \frac{\psi_B}{UL} \qquad \Rightarrow \qquad \psi_B = UL\psi_B' \tag{2.35}$$

$$\psi_T' = \frac{\psi_T}{UL} \qquad \Rightarrow \qquad \psi_T = UL\psi_T' \tag{2.36}$$

$$h_B' = h_B\frac{f_0L}{UH_2} \qquad \Rightarrow \qquad h_B = \frac{UH_2}{f_0L}h_B' \tag{2.37}$$

where the prime denote non-dimensional variables. The constant L is the width/length of the square basin and U is a typical flow velocity scale (their particular values are mentioned in section 2.4). We insert the scaled variables into the dimensional equations (2.27) and (2.31) and perform necessary algebra (and dividing through by U^2/L^2 from the time-derivative term) to get the non-dimensional barotropic and baroclinic equations:

$$\frac{\partial q_B'}{\partial t'} + \mathcal{J}'(\psi_B', q_B') + \delta_1\delta_2\mathcal{J}'(\psi_T', q_T') + \beta'\frac{\partial \psi_B'}{\partial x'}$$

$$+ \delta_2 \mathcal{J}' \left(\psi'_B - \delta_1 \psi'_T, h'_B\right) = \delta_1 \mathcal{F}'_1 - \delta_2 r' \nabla'^2 (\psi'_B - \delta_1 \psi'_T) \tag{2.38}$$

$$\frac{\partial q'_T}{\partial t'} + \mathcal{J}'(\psi'_B, q'_T) + \mathcal{J}'(\psi'_T, q'_B) + (\delta_2 - \delta_1)\mathcal{J}'(\psi'_T, q'_T)$$

$$+ \beta' \frac{\partial \psi'_T}{\partial x'} - \mathcal{J}' \left(\psi'_B - \delta_1 \psi'_T, h'_B\right) = \mathcal{F}'_1 + r' \nabla'^2 (\psi'_B - \delta_1 \psi'_T). \tag{2.39}$$

where $\mathcal{F}_1$ is the non-dimensional upper layer forcing and the non-dimensional barotropic and baroclinic potential vorticities are defined by

$$q'_B = \nabla'^2 \psi_B \tag{2.40}$$

$$q'_T = \nabla'^2 \psi_T - F' \psi_T \tag{2.41}$$

and the non-dimensional altered physical constants through

$$\beta' = \beta \frac{L^2}{U}\ , \qquad r' = r \frac{L}{U}\ , \qquad F' = F L^2, \tag{2.42}$$

as well as the non-dimensional Laplace operator ∇'^2 and Jacobian operator $\mathcal{J}'$ as

$$\nabla'^2 = \frac{\partial^2}{\partial x'^2} + \frac{\partial^2}{\partial y'^2} \qquad \text{and} \qquad \mathcal{J}'(f, g) = \frac{\partial f}{\partial x'} \frac{\partial g}{\partial y'} - \frac{\partial g}{\partial x'} \frac{\partial f}{\partial y'}.$$

Equations (2.38) and (2.39) (together with (2.40) and (2.41)) are the ones we solved using the numerical model and are therefore essential in this project. Finally, note that they correspond exactly with the equations of motion in LaCasce and Pedlosky [2004] (as noted earlier they also used this model) with the exception of the forcing and topography terms being absent there. As seen, they describe a time-evolving system, and as such, the equations in question require initial conditions as well. We discuss these in section 2.4.2. From here onwards we will work with the non-dimensional equations and variables, but drop the primes for simplicity.

2.2 Updating the model

Since we were in need of extending the model to include the additional physics, we preferred to have a flexible baseline for the code such that the extensions in question did not become cumbersome and clutter the codebase. We wanted to be able to add to the code without compromising the models original intentions, in particular let usage of the model with and without topography be easily interchangeable by "turning a switch". Additionally, we aimed at making the model more user accessible in terms of specifying parameters. The combination of theses desires prompted a restructure of the model codebase before we went about implementing the topography. Below are stages of this update process elaborated upon in chronological order of implementation.

Fortran 77 to Fortran 2003: In order to achieve the above mention flexibility, we found it helpful and necessary to rewrite the model in a modern version of Fortran. The main reasoning behind this was founded in modern Fortran's support of a modular object oriented programming style (see next paragraph). The process of extending the code from the base Fortran 77 version, while definitely possible, would have been far more cumbersome and yielded less flexible as well as harder-to-read code. We chose Fortran 2003 for the updated version and the first step was to update all syntax to be consistent with the modern style. At first we attempted to use various existing tools for automated conversion of Fortran code, but these were found to be insufficient for our purposes and was abandoned. We then turned to the manual approach. In practice this meant rewriting all variable declarations, `if`, `do`, `while`, `goto`, etc. control structures as well as other syntactic details, to the new standard. After this was completed, the code was compliant with a modern `gfortran` compiler Gfortran. Version 4.8 was used for all subsequent compilation of the code.

Object orientation: With the modern syntax, we were able to utilise the modular programming style supported in the newer Fortran versions. The code was restructured in terms of module files, types/classes and member subroutines all together constituting an object oriented version of the code. This step was very helpful when later adding new features to the model.

Dynamic memory allocation: As with almost all numerical models, there is a need for using array structures to store data in random access memory (RAM) during simulations. This model was originally written using static memory allocation for such arrays. In our attempt to add flexibility, we instead implemented dynamic memory allocation and introduced a new parameter file; a simple namelist that allowed users to specify all model input parameters in a text file which was then read by the model at runtime and further allocating the necessary memory.

After having completed the above described stages as well several other minor fixes, we had a functioning modern version of the model producing the exact same results as the original Fortran 77 version. Completing the update without introducing bugs that would tamper with the functionality of the model, was a major concern. During the update, a lot of time was spent on testing and verification in order to ensure the model had not been compromised in any way. This testing was done through designing various test cases (simulations) and quantitatively checking output values as well as frequent visual inspection of the time evolution of the energy and streamfunctions. We also used version control software (git) as another safety measure for avoiding bugs.

2.3 Implementing topography

With the updated model in place, we moved forward with implementation of topography. Specifically, we extended the model to support the topography terms in (2.38) and (2.39). We now show how we applied the work from section 2.1 to achieve this.

2.3.1 Theory

In section 2.1 we found the topography enters as advection by the lower layer flow $\mathbf{u}_2 = (u_2, v_2)$. Recall the topography term enter in both the non-dimensional barotropic and baroclinic equation as

$$\mathcal{J}(\psi_B - \delta_1\psi_T, h_B) = \mathcal{J}(\psi_2, h_B) \tag{2.43}$$

where (2.7) has been applied. This term can further be written as

$$\mathcal{J}(\psi_2, h_B) = \frac{\partial \psi_2}{\partial x}\frac{\partial h_B}{\partial y} - \frac{\partial h_B}{\partial x}\frac{\partial \psi_2}{\partial y} = \mathbf{u}_2 \cdot \nabla h_B = u_2\frac{\partial h_B}{\partial x} + v_2\frac{\partial h_B}{\partial y} \tag{2.44}$$

using the definition (2.20) of the streamfunction. The latter part of (2.44) is the term we implemented in the model. We chose to formulate the term via the layer flow instead of the barotropic/baroclinic flow as the layer flows are required for the energy computations anyway, and so it made the code more readable. Although, since the model is built around the barotropic and baroclinic equations, u_2 and v_2 were unavailable at this point in the code. Therefore we needed a way to compute these.

The barotropic and baroclinic streamfunctions ψ_B and ψ_T were readily available in the code. Taking the spatial partial derivatives of (2.7) yields

$$\frac{\partial \psi_2}{\partial x} = \frac{\partial \psi_B}{\partial x} - \delta_1\frac{\partial \psi_T}{\partial x} \qquad \Rightarrow \qquad v_2 = v_B - \delta_1 v_T \tag{2.45}$$

$$\frac{\partial \psi_2}{\partial y} = \frac{\partial \psi_B}{\partial y} - \delta_1\frac{\partial \psi_T}{\partial y} \qquad \Rightarrow \qquad u_2 = u_B - \delta_1 u_T \tag{2.46}$$

where

$$v_B = \frac{\partial \psi_B}{\partial x}\ , \quad v_T = \frac{\partial \psi_T}{\partial x}\ , \quad u_B = -\frac{\partial \psi_B}{\partial y} \quad \text{and} \quad u_T = -\frac{\partial \psi_T}{\partial y} \tag{2.47}$$

This tells us that we have a way of computing u_2 and v_2 through computing the spatial derivatives of ψ_B and ψ_T and then invoking the latter parts of (2.45) and (2.46).

2.3.2 Numerics

Before we look at the numerics used for the implementation, it is useful to briefly mention what part of the code we extended. A section of the code deals with collecting all linear terms of the equation and this is where we added the topography term. Our job was then to evaluate (2.44) at every spatial grid point each time step. Additionally, we had to keep in mind the models use of a staggered grid with potential vorticity defined at the cell centers while the streamfunctions are defined at the southwest corner of the grid cells.

The finite difference approximations (and their related truncation errors) we mention in this section are as described in Røed [2016, sec. 2.7]. The numerical schemes that make up the model dictate a use of forward differences when computing the derivatives of the streamfunctions (it is actually more complicated than this since the model employs a numerically staggered grid). In line with this, we used

$$v^B_{i,j} = \left.\frac{\partial \psi_B}{\partial x}\right|_{i,j} \simeq \frac{\psi^B_{i+1,j} - \psi^B_{i,j}}{\Delta x}\ , \qquad i = 1, 2, \ldots, N-1,\ j = 1, 2, \ldots, N \tag{2.48}$$

$$v^T_{i,j} = \left.\frac{\partial \psi_T}{\partial x}\right|_{i,j} \simeq \frac{\psi^T_{i+1,j} - \psi^T_{i,j}}{\Delta x}\ , \qquad i = 1, 2, \ldots, N-1,\ j = 1, 2, \ldots, N \tag{2.49}$$

$$u^B_{i.j} = -\left.\frac{\partial \psi_B}{\partial y}\right|_{i,j} \simeq \frac{\psi^B_{i,j+1} - \psi^B_{i,j}}{\Delta y}\ , \qquad i = 1, 2, \ldots, N,\ j = 1, 2, \ldots, N-1 \tag{2.50}$$

$$u^T_{i,j} = -\frac{\partial \psi_T}{\partial y}\bigg|_{i,j} \simeq \frac{\psi^T_{i,j+1} - \psi^T_{i,j}}{\Delta y}\ , \qquad i = 1, 2, \ldots, N,\ j = 1, 2, \ldots, N-1. \tag{2.51}$$

where $i, j \in [1, N] \in \mathbb{N}$ represent the indices of the grid points in the x- and y-direction, respectively, while N is the number of grid points in each direction. These forward differences are first order accurate in space meaning they have an associated truncation error of $\mathcal{O}(\Delta x)$ or $\mathcal{O}(\Delta y)$ depending on the term. Equations (2.48)-(2.51) gives the BT/BC flow. However, a feature of the staggered grid in this model is that $\psi^B_{i,j}$ and $\psi^T_{i,j}$ are defined at the southwest corner of each grid cell and causes $u^B_{i,j}$ and $u^T_{i,j}$ to be defined at the center of the western edge of grid cell (i, j) while $v^B_{i,j}$ and $v^T_{i,j}$ are defined at the center of the southern edge. As required in the model, each term in the equation must be centered in the grid cells when collecting all terms. This also applies to the topography term and thus we found the centered BT/BC flows by an ordinary two-point interpolation on neighbouring cell edge values, that is

$$u^{B*}_{i,j} = \frac{u^B_{i+1,j} + u^B_{i,j}}{2}\ , \qquad u^{T*}_{i,j} = \frac{u^T_{i+1,j} + u^T_{i,j}}{2} \tag{2.52}$$

$$v^{B*}_{i,j} = \frac{v^B_{i,j+1} + v^B_{i,j}}{2}\ , \qquad v^{T*}_{i,j} = \frac{v^T_{i,j} + v^T_{i,j}}{2} \tag{2.53}$$

where the stars indicate the cell centered versions of the velocities. From (2.45) and (2.46) we then computed the (cell centered) lower layer flow $\mathbf{u}_2$.

We also needed the partial derivatives of the topography h_B as seen in (2.44). We employed a centered-in-space scheme in the interior of the domain, i.e.

$$\frac{\partial h_B}{\partial x}\bigg|_{i,j} \simeq \frac{h^B_{i+1,j} - h^B_{i-1,j}}{2\Delta x}\ , \qquad i = 2, 3, \ldots, N-1,\ j = 1, 2, \ldots, N \tag{2.54}$$

$$\frac{\partial h_B}{\partial y}\bigg|_{i,j} \simeq \frac{h^B_{i,j+1} - h^B_{i,j-1}}{2\Delta y}\ , \qquad i = 1, 2, \ldots, N,\ j = 2, 3, \ldots, N-1. \tag{2.55}$$

The centered differences have truncation errors $\mathcal{O}(\Delta x^2)$ and $\mathcal{O}(\Delta y^2)$, respectively, and are thus good quite good for accuracy. Along the boundaries we had to resort to other means as the centered differences would require points outside the domain. We instead used one-sided forward and backward differences. In particular, forward differences along the two boundaries represented by $i = 1$ and $j = 1$ and similarly backward differences on the two boundaries $i = N$ and $j = N$:

$$\frac{\partial h_B}{\partial x}\bigg|_{i,j} \simeq \frac{h^B_{i+1,j} - h^B_{i,j}}{\Delta x}\ , \qquad i = 1,\ j = 1, 2, \ldots, N \tag{2.56}$$

$$\frac{\partial h_B}{\partial x}\bigg|_{i,j} \simeq \frac{h^B_{i,j} - h^B_{i-1,j}}{\Delta x}\ , \qquad i = N,\ j = 1, 2, \ldots, N \tag{2.57}$$

$$\frac{\partial h_B}{\partial y}\bigg|_{i,j} \simeq \frac{h^B_{i,j+1} - h^B_{i,j}}{\Delta y}\ , \qquad i = 1, 2, \ldots, N,\ j = 1 \tag{2.58}$$

$$\frac{\partial h_B}{\partial y}\bigg|_{i,j} \simeq \frac{h^B_{i,j} - h^B_{i,j-1}}{\Delta y}\ , \qquad i = 1, 2, \ldots, N,\ j = N. \tag{2.59}$$

Since the topography is stationary, i.e. $\partial h_B/\partial t = 0$, the spatial partial derivatives remain constant throughout and thus we only computed them once before simulating the system. We have now gone through the necessary steps for evaluating (2.44) in the model. In short, we used (2.48)-(2.53) and then (2.45)-(2.46) to get $\mathbf{u}_2$ and combined this with the topography partial derivatives (2.54)-(2.59). Algorithm 2.1 summarizes the method.

Algorithm 2.1 Topography implementation

```
procedure ADD_TOPOGRAPHY_TERMS($\psi_B$, $\psi_T$, $h_B$)
    - Compute topography partial derivatives $\partial h_B/\partial x, \partial h_B/\partial y$

    for every time step do
        for every grid point in $x$-direction do
            for every grid point in $y$-direction do
                - Compute $u_B, u_T, v_B, v_T$ from $\psi_B, \psi_T$
                - Center $u_B, u_T, v_B, v_T$ in the grid cells
                - Compute $u_2, v_2$ from $u_B, u_T, v_B, v_T$
                - Compute topography terms using $u_2, v_2$ and $\partial h_B/\partial x, \partial h_B/\partial y$
                - Add topography terms to both BT and BC equation
```

2.4 Model configuration and initial condition

Our intentions with this section are to specify the physical and numerical constants as well as the initial condition(s) we used for the two-layer system. We do this both to give a clearer picture of the simulations we have done for chapter 4, but also to add to the reproducibility of the study.

2.4.1 Parameters

As the model deals with the non-dimensional equations (2.38) and (2.39), the dimensional physical constants β, r and F that enter equations (2.27) and (2.31) were modified to their non-dimensional versions β', r' and F' given by (2.42). The dimensional values are also relevant if one wishes to scale the model output back to dimensional space. Table 2.1 lists the physical constants we used. Key features from the table include a basin of synoptic scale, an upper layer that has a resting depth 1/4 of the thickness to that of the lower layer, coriolis parameter based on a latitude of 45° and farily weak bottom friction. However, friction and topographic height valuea are merely examples as we varied them substantially between runs. Of the physical constants in table 2.1, L, U, β, r and F (and thus implicitly H_1 and H_2) enter the non-dimensional parameters in (2.42). In addition physical (non-dimensional) constants, the model also takes in a set of numerical constants. The latter may either be parameters like the number of grid points or for enabling/disabling certain features. In table 2.2 we list all model input parameters (except parameters related to wind forcing as this was never used) that are taken as input and that we used in our runs. Some of them, however, were varied between simulations and this is mentioned in chapter 4 when relevant. Otherwise, the values in the table were used.

Constant	**Value**	**Description**
U	0.1 m/s	typical zonal and meridional flow
L	10^6 m	basin length/width
H_1	1000 m	depth of upper layer
H_2	4000 m	depth of lower layer
F	10^{-9} m^{-2}	related to layer density difference (Burger number)
f_0	10^{-4} s^{-1}	constant part of Coriolis parameter
β	10^{-13} s^{-1}	meridional Coriolis gradient
r	10^{-8} s^{-1}	Rayleigh friction coefficient
h_0	10 m	typical topographic height

Table 2.1: List of physical constants used for the two-layer system.

2.4.2 Initial condition

We use initial conditions comparable to those of LaCasce and Pedlosky [2004] and Dukowicz and Greatbatch [1999] with a baroclinically unstable upper layer wave. Ideally, we specify the initial conditions in terms of the upper and lower layer streamfunctions ψ_1 and ψ_2. However, since the model is initialized with barotropic and baroclinic potential vorticity q_B and q_T, we needed to derive these from our chosen layer streamfunctions. Assuming known $\psi_1(x, y, 0) = \psi_1^0$ and $\psi_2(x, y, 0) = \psi_2^0$ we have through (2.3) and (2.4) that

$$\psi_B^0 = \delta_1\psi_1^0 + \delta_2\psi_2^0 \tag{2.60}$$

$$\psi_T^0 = \psi_1^0 - \psi_2^0 \tag{2.61}$$

which combined with (2.40) and (2.41) gives

$$q_B^0 = \nabla^2\psi_B^0 = \nabla^2(\delta_1\psi_1^0 + \delta_2\psi_2^0) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)(\delta_1\psi_1^0 + \delta_2\psi_2^0) \tag{2.62}$$

$$q_T^0 = \nabla^2\psi_T - F\psi_T = \nabla^2(\psi_1^0 - \psi_2^0) - F(\psi_1^0 - \psi_2^0) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} - F\right)(\psi_1^0 - \psi_2^0) \tag{2.63}$$

where q_B^0 and q_T^0 are the corresponding initial conditions for the barotropic and baroclinic potential vorticity. We used (2.62) and (2.63) to initiate the model based on our chosen initial conditions ψ_1^0 and ψ_2^0.

As note earlier, since we were interested in studying the lower layer spin-up as a function of topography, we used an initial upper layer flow and while a lower layer starting from rest. An initial surface trapped wave in the east-west direction was used. This means a wave-like upper layer streamfunction with a lower layer at rest. In particular,

$$\psi_1(x, y, 0) = A\sin(kx)\sin(ly) \qquad \text{and} \qquad \psi_2(x, y, 0) = 0. \tag{2.64}$$

Parameter	**Symbol**	**Value**	**Description**
`fortype`		0	forcing: =0 (off), =1 (wavelike), =2 (random)
`nonlin`		1	non-linearity: =0 (off), =1 (on)
`rvd`		1	relative vorticity damping: =0 (off), =1 (on)
`topo`		1	topography: =0 (off), =1 (on)
`topofile`		1	topo from file: =0 (off), = 1 (on)
`ginit`		0	init. con. from file: =0 (off), =1 (on)
`mbc`		2	num. ghost cells outside domain
`N`	N	256	num. grid points in each direction
`ntot`	N_t	10^6	total number of time steps
`Nener`	N_{en}	10^3	energy output interval
`Nwrite`	N_{write}	10^4	streamfunction output interval
`Navg`	N_{avg}	10^{10}	mean streamfunction ouptut interval
`L`	L'	π	non-dim length of domain
`delt`	Δt	10^{-4}	non-dim time step
`h1oh2`	δ_1/δ_2	0.25	layer depth ratio H_1/H_2
`rayl`	r'	0.04	non-dim friction coefficient
`beta`	β'	1	non-dim coriolis gradient
`F0`	F'	10^3	non-dim "interface" constant

Table 2.2: List of model parameters present in the configuration file together with some typical values. The non-dimensional β', r' and F' were calculated using (2.42) *and the physical values in table 2.1.*

Here A is some constant amplitude and k is the wavenumber in the x-direction and from scaling implies a reasonable value of $A = 0.25$ which we used throughout. The $\sin(ly)$ factor was included only to satisfy the north-south no-normal-flow boundary conditions and $l = 1$ throughout. We let $k = n$ with $n \in \mathbb{N}$ (recall domain $x, y \in [0, \pi]$) to satisfy the western and eastern boundary conditions, while simultaneously allowing for zero-crossings between $x = 0$ and $x = \pi$ to get the wave structure. Through (2.20) there is an upper layer initial velocity field

$$u_1 = -\frac{\partial \psi_1}{\partial y} = -lA\sin(kx)\cos(ly) \tag{2.65}$$

$$v_1 = \frac{\partial \psi_1}{\partial x} = kA\cos(kx)\sin(ly) \tag{2.66}$$

associated with this initial condition while the lower layer flow $\mathbf{u}_2 = \mathbf{0}$. Using (2.64) in

(2.60) and (2.61) and applying (2.62) then (2.63) yields the barotropic/baroclinic potential vorticity initial conditions

$$q_B^0 = -(k^2 + l^2)\delta_1 A \sin(kx)\sin(ly) \tag{2.67}$$

$$q_T^0 = -(k^2 + l^2 + F)A \sin(kx)\sin(ly) \tag{2.68}$$

while further inserting with $k = 4$ and $l = 1$, gives

$$q_B^0 = -17\delta_1 A \sin(4x)\sin(y) \tag{2.69}$$

$$q_T^0 = -(F + 17)A \sin(4x)\sin(y) \tag{2.70}$$

which were the initial conditions we used for all simulations in chapter 4 unless otherwise specified.

2.5 Grid resolution analysis

When done updating and extending the model we were almost ready to run it for our purposes. However, some more preparation serves well in the long run and, in particular, this section involves a discussion on the impact of the models spatial grid resolution on the accuracy of the numerical solutions. We found it useful to get a grasp on this such that more informed choices could be made when setting up the model for producing results in chapter 4. Additionally, it provided some insight into the potential shortcomings of the model output.

As noted earlier, the QUICK scheme used for advection is inherently dissipative at small scales and hence some loss of energy is expected and especially coarser grid resolutions where this effect is magnified. Such energy loss is not ideal and could potentially increase the difficulty in interpretation of the physical features present in the simulations. Therefore we sought to minimize this effect. At first thought, one might suggest simply increasing the resolution of the model, that is to use more grid points $N \times N$ to limit the small scale dissipation to even smaller scales. This cannot, however, be done without concern. For example, doubling the number of grid points N in each direction, quadruples the total number of grid points meaning that, for each time step, each internal loop iterating over both the x- and y direction (which are most of them) uses four times the amount of time. This ultimately results in the runtime to increase by a factor of about four. Additionally, since the numerical stability condition depends directly on the spacing dx and dy between grid points, increasing N puts a stricter constraint on the time step for the solution to remain numerically stable and not grow without bounds. With this in mind, increasing the resolution should be done with care and with consideration that a finite amount of time is available for running the model.

We now attempt to solidify this discussion with some actual data from the model. Figure 2.2 shows the time evolution of energy in the flat-bottom two-layer system for various grid resolutions $N \times N$ (N is chosen as power of 2 for numerical optimizations related to Fourier transformations) in the linear case. A typical surface layer wave $\psi_1 = A\sin(4x)\sin(y)$ and $\psi_2 = 0$ was used as initial conditions. There is no wind forcing or bottom friction enabled in any of these runs, and thus there is no physical source or sink of energy

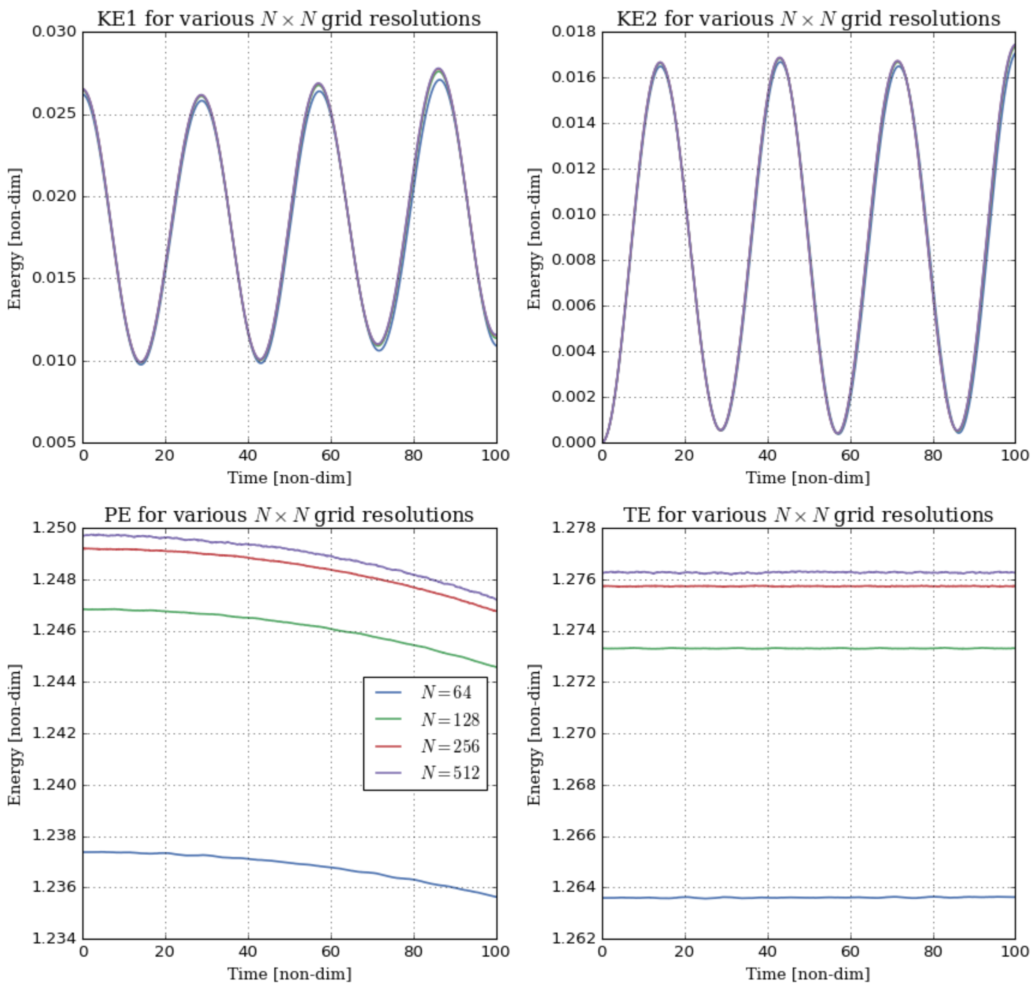


Figure 2.2: From upper left panel to lower right: Time evolution of upper layer kinetic, lower layer kinetic, potential and total energy, respectively, for four different spatial resolutions N. Linear runs with flat bottom and no friction. Initial wave in the upper layer while lower layer starts at rest.

meaning the total energy should be conserved throughout. It is visible that this is the case in the linear simulations. The total energy stays constant for all runs, however there is a slight difference in the starting value for the potential, and thus also the total energy, between the four grid resolutions. This likely comes from the model using the standard rectangle/column summation as an approximation to the spatial integral (see e.g. 1D version in Mørken [2017, p. 300]) that give the potential energy. This standard method underestimates functions that are concave down (and overestimate function that are concave up) and more so for coarser resolutions. Since our initial condition (2.64) for ψ_1 is mostly concave down (and $\psi_T^0 = \psi_1^0 - \psi_2^0, \psi_2^0 = 0$), this seems to be the cause. The difference, though, is quite small and stays constant throughout these typical runs.

A similar plot for the non-linear system can be seen in figure 2.3. A key difference with these runs compared to the linear ones, is the instabilities that occur and cause the flow to turn turbulent. This transition can be seen at roughly time $t = 5$ manifested as the sudden decrease in potential energy and corresponding increase in the kinetic energies. It

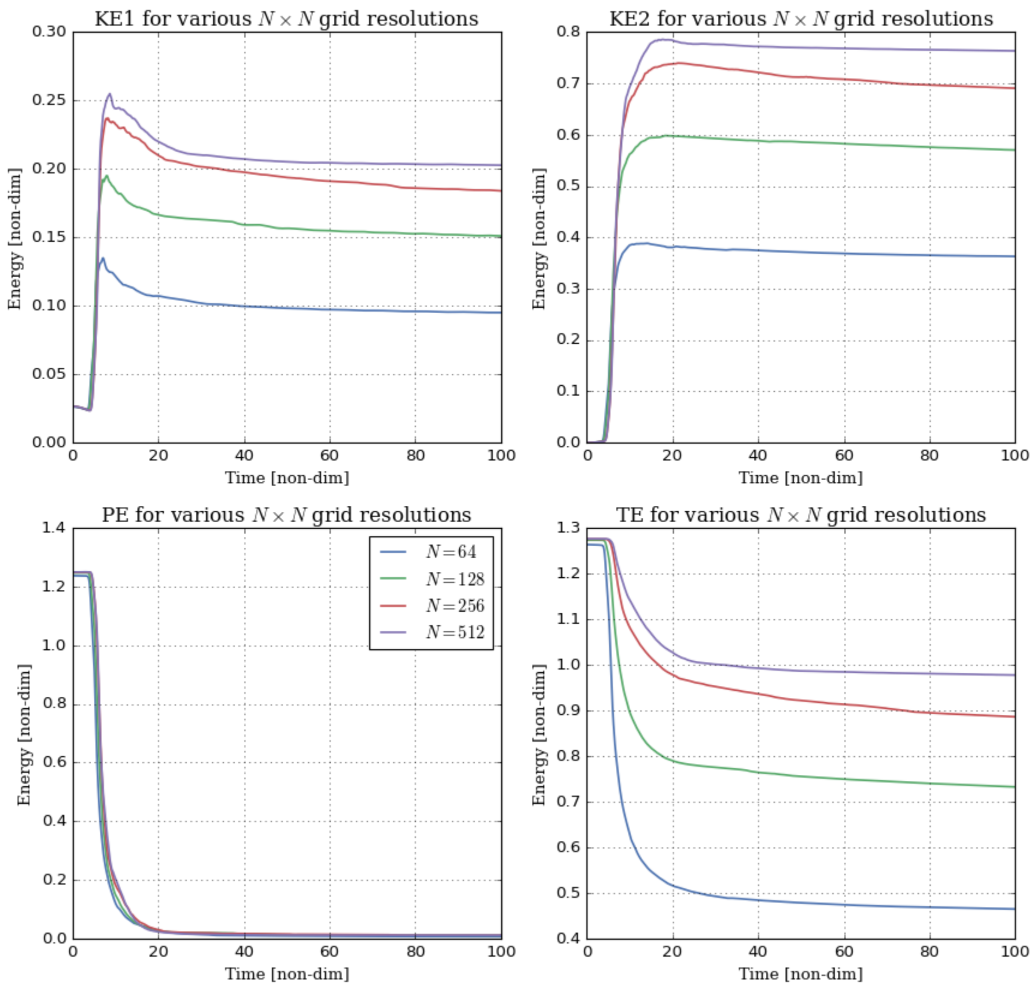


Figure 2.3: From upper left panel to lower right: Time evolution of upper layer kinetic, lower layer kinetic, potential and total energy, respectively, for four different spatial resolutions N. Non-linear runs with flat bottom and no friction using an initial wave in the upper layer while lower layer starts at rest.

is at this time the differences due to grid resolution really become apparent; even though there is a clear loss of total energy present for all grid resolutions, the loss is greater for the smaller resolutions and decreases with increasing N as expected. This indicates that, regardless of N, the loss of energy is predominantly in the form of kinetic energy (in both layers) considering the evolution of the potential energy is very similar for all N. In light of the above, this is something we expected from the small scale dissipation present in the QUICK scheme. When the field goes turbulent, motion generated at smaller scales facilitates the leak of energy. Finally, note that the loss of energy even for $N = 512$ is quite significant and that the $N = 256$ runs are largely similar making the latter not much worse of a choice especially bearing in mind the vastly larger computational time needed when using $N = 512$.

In conclusion, the usage of $N = 128$ was deemed fully acceptable in the linear system (with the exception of cases discussed in the next paragraph), but not desirable in the non-linear case. That being said, using $N = 512$ would significantly increase the overall runtime to a point where completing the necessary runs would become an issue. Consequently, we found $N = 256$ to be a decent and necessary compromise between accuracy and time usage. The latter is very important also considering the sheer number of simulations we ran in this project.

There is one more impact of grid resolution that is worth considering. One of the parameters, whose effect we examine in chapter 4, is the wavenumber of the bottom topography. If we define a rapidly varying topography, care should be taken to ensure the topography is well resolved on the grid. Otherwise the observed response of the system could be misleading in the sense that the flow reacted to an unresolved topography, which in practice simply corresponds to a different topography. In order to visualize this more clearly, consider the example in figure 2.4 from Røed [2016]. The black dots indicate what values of the topography h_B that is stored on a grid of distance Δx between grid points (although we are only considering one dimension here, the argument is the same in the y-direction as well). As Røed notes, we see that the more rapidly varying topography is indistinguishable from the longer wavelength topography. Both these topographies would be represented the exact way on this grid and the system would respond equally to both. The shorter wavelength topography is mapped onto the longer, i.e. to the shortest wavelength that can be represented on the grid, and is thus not present in the simulation as intended. With this pitfall in mind, we chose the topography wavenumber and the grid resolution carefully when producing the results for chapter 4.

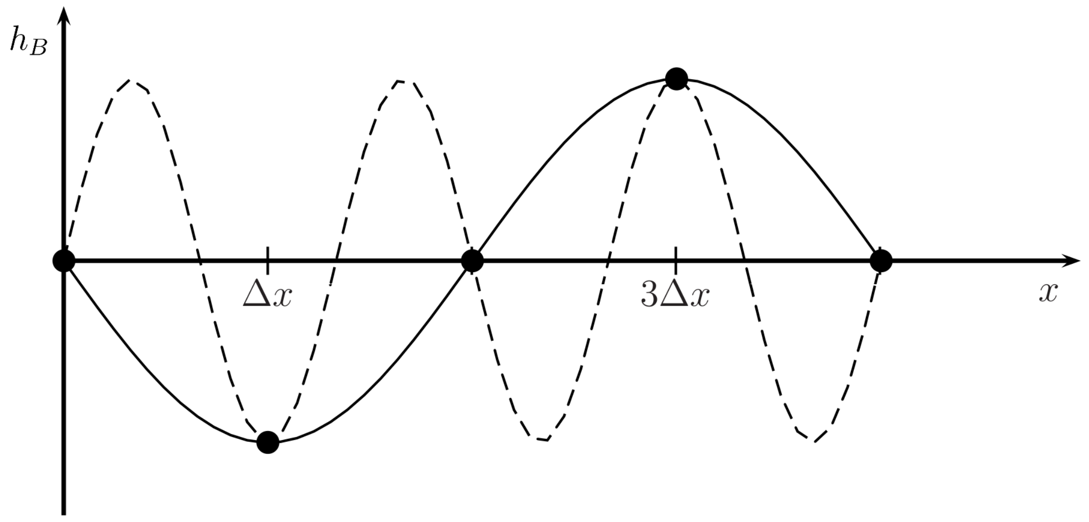


Figure 2.4: Figure from Røed [2016, p. 169]. Illustration of two sinusoidal topographies (in the x direction) with wavelength $4\Delta x$ (solid curve) and $4\Delta x/3$ (dashed curve) on a grid with distance Δx between grid points.

Chapter 3

Simulation framework

We developed a method that significantly changed how we produced results in the project. As noted, one aim was to explore dependences in the model over a wide range of parameters, for example, of the topographic height or the bottom friction coefficient. This would normally entail re-running the model manually many times, a tedious process (and in fact impossible for the several thousands runs we did). This chapter focuses on how we automated this process both which enhanced the models "ease of use" and permitted systematic parameter variation through parallel execution of independent simulations. The parallelization was made possible by utilizing a set of desktop computers as a computational cluster. We start by discussing parallelism and derive the parallel algorithm used to oversee the model runs for every result produced in chapter 5. Then we discuss the actual implementation of the framework.

3.1 Parallel computation

The Fortran model employed in this project is in itself not parallelized and thus its instructions are executed sequentially on a single central processing unit (CPU) core by default. We considered the task of parallelizing the model, and while definitely possible through the use of a distributed memory message passing library (e.g. MPI) or a shared memory approach (e.g. OpenMP), this was abandoned as it was deemed too time consuming and hence potentially having a negative impact on other parts of the project. Another reason is that the nature of the project did not imply a strong requirement for one single simulation to execute very fast, but rather have a set of simulations execute in a short amount time. This leads to the parallel approach that was implemented for this project.

In general, when running the same numerical model several times while changing the value of an input parameter between each run, it follows that each single run is totally independent of every other run. In other words, there is no need for any exchange of information between the runs and they execute without knowledge of eachother. It then emerges that these independent runs can be performed simultaneously. This was indeed the case for the model used here and we exploited this fact by designing a method for running separate simulations in parallel. The method applies generally, but was also customized for the particular model in this project. Given a set of N simulations (each simulation characterized by an altered value of some input parameter) to be performed, we explored two approaches for running them in parallel. The following two sections elaborate on these approaches and are inspired by parallel theory described in [Grama et al., 2003, chap. 3].

3.1.1 Local parallelism

After starting to explore the possibilities of reducing the total computational time for all simulations, we initially intended to use one single computer to run all N simulations. Assume this computer had p available CPU cores for processing. This meant that, at any time, p simulations could be executed in parallel if we utilised all p CPUs. This gave rise to an idea about how we could distribute the work in a sequence of "parallel sessions". [1] A general discussion of this follows.

In general, N is not an integer multiple of p, or in other words, N/p is not an integer. This means that the N simulations cannot be distributed perfectly in an integer number of parallel sessions where each session performs p simulations simultaneously. The work can instead be distributed as evenly as possible by structuring the workload into $\lfloor N/p \rfloor$ parallel sessions, where $\lfloor \cdot \rfloor$ denotes the floor operator, with each session doing p simulations concurrently using all p CPUs. This amounts to a total of $\lfloor N/p \rfloor p$ executed simulations. The number of remaining simulations that are not yet run is given by the remainder $N \bmod p$. A work distribution can then be formulated in terms of rewriting N as

$$N = \left\lfloor \frac{N}{p} \right\rfloor p + N \bmod p. \tag{3.1}$$

This equation holds for any $a, b \in \mathbb{N}$ and can be seen as a decomposition of the integer N. In this context, equation (3.1) provides and algorithm for executing N simulations as efficiently as possible: Perform $\lfloor N/p \rfloor$ parallel sessions with each session executing p simulations in parallel and after the final session is done, the remaining $N \bmod p$ simulations are executed using $N \bmod p$ of the total p CPUs (recall that $N \bmod p < p$, so the remainder amounts to one parallel session, but without using all p CPUs. This leads to a total of $\lfloor N/p \rfloor + 1$ parallel sessions if $N \bmod p \neq 0$). It is worth noting that in the special case of N/p actually being an integer, the the work distribution becomes optimal and we utilise all CPUs throughout all N/p parallel sessions. Observe also that (3.1) holds for the case in which $N < p$; then the first term is zero and get one parallel session using $N \bmod p$ CPUs.

In terms of performance gains, the obtained speed-up S is defined similarly to [Grama et al., 2003, sec. 5.3.2] as the ratio of the total serial execution time T_s to the total parallel execution time T_p, that is

$$S = \frac{T_s}{T_p}. \tag{3.2}$$

Assume now N is not a multiple of p such that $N \bmod p \neq 0$. The total execution time is directly proportional to the number of simulations that are run, so

$$T_s = kN \qquad \text{and} \qquad T_p = k\left(\left\lfloor \frac{N}{p} \right\rfloor \frac{p}{p} + \frac{N \bmod p}{N \bmod p} \right) = k\left(\left\lfloor \frac{N}{p} \right\rfloor + 1 \right)$$

[1]From here onwards, let "parallel session" be defined as the event where one or more CPUs execute separate simulations simultaneously.

where $k \in \mathbb{R}$ is some proportionality constant (which is the same for both times) and the workload in the parallel case has been divided p and $N \bmod p$ for the two terms, respectively, to account for the work being done in parallel. Using (3.2), the speed-up becomes

$$S = \frac{N}{\lfloor N/p \rfloor + 1} \approx \frac{N}{\lfloor N/p \rfloor} \approx p \tag{3.3}$$

where the approximations comes from assuming $N \gg p$ (as is often the case). The local parallelism described in this section provides a speed-up of roughly factor p compared to simply running all N simulations sequentially one after another on a single CPU. It is intuitively reasonable that, for completely independent simulations, using p times as many CPUs leads to completing all the simulations in a factor $1/p$ of the time, but here it is shown that this is approximately the case also when one doesn't obtain the optimal work distribution, i.e. when $N \bmod p \neq 0$. The perfect speedup of $S = p$ is achieved in the aforementioned special case where $N \bmod p = 0$. While the parallelism described in this section is useful in itself and was employed when doing some of the runs for this project, its main purpose is to serve as a vital building block for the next section.

3.1.2 Cluster parallelism

The local parallelism described in section 3.1.1 significantly sped up the process of running a set of several simulations, but was still found too slow to be able to fully explore the desired range of parameters. As discussed more detailed in section 3.2, we had a small machine cluster available for use by this project and so our attention turned to deriving a method for optimally distributing the work amongst the multiple machines to fully utilise the cluster.

Suppose still a total number of N simulations are to be executed and that the cluster consists of M machines, each with p CPUs (assume same number of CPUs per machine for simplicity, although algorithmically, allowing different CPU counts would not make a big difference). Using similar logic to that of the previous section we may divide the N simulations among the M machines as evenly as possible by giving every machine $\lfloor N/M \rfloor$ simulations and then distributing the remaining simulations $N \bmod M$ among a subset $N \bmod M$ of the M machines in the cluster (meaning these particular machines is assigned one additional simulation each). Similar to that of equation (3.1), this work division can be expressed mathematically as

$$N = \left\lfloor \frac{N}{M} \right\rfloor M + N \bmod N. \tag{3.4}$$

However, since the number of simulations assigned to each machine, that is either $\lfloor N/M \rfloor$ or $\lfloor N/M \rfloor + 1$, generally may exceed the number of CPUs p on each machine, additional work distribution must be done locally on each machine. Here the work is divided exactly as described in section 3.1.1 with the only difference the input of total simulations for each machine being $\lfloor N/M \rfloor$ or $\lfloor N/M \rfloor + 1$ instead of N, meaning (3.1) then gives

$$\left\lfloor \frac{N}{M} \right\rfloor = \left\lfloor \frac{\lfloor N/M \rfloor}{p} \right\rfloor p + \left\lfloor \frac{N}{M} \right\rfloor \bmod p \tag{3.5}$$

or

$$\left\lfloor \frac{N}{M} \right\rfloor + 1 = \left\lfloor \frac{\lfloor N/M \rfloor + 1}{p} \right\rfloor p + \left(\left\lfloor \frac{N}{M} \right\rfloor + 1 \right) \bmod p \tag{3.6}$$

depending respectively on whether the machine in question was one that was given only the default amount of simulations or one that was also assigned one extra simulation from the remainder pool.

See algorithm 3.1 which summarizes the method described above and outlines the application of equations (3.4), (3.5) and (3.6). Note however that the algorithm is a set of instructions for generating a work distribution and does not directly include the actual running of the simulations (although this is quite accessible once the work distribution has been achieved). Also worth noting is that the special cases of $M > N$ and/or $p > \lfloor N/M \rfloor$ are covered by the algorithm through then only handing out work to machines and CPUs via the remainder if-statements. The special case of zero remainder terms is also taken care of by the if-statements. Details regarding implementation of algorithm 3.1 are discussed in section 3.2.

Algorithm 3.1 Cluster work distribution

```
procedure CLUSTER_DIST($N$, $M$, $p$)
    // N [int]: Total number of tasks
    // m [int]: Number of machines in cluster
    // p  [int]: Number of CPUs per machine in cluster

    - Assign $\lfloor N/M \rfloor$ tasks to each of the $M$ machines in the cluster

    if $N \bmod M \neq 0$ then
        - Assign 1 additional task to the "first" $N \bmod M$ machines in the cluster

    for each machine in cluster do
        - Define $n$ to be the number of tasks assigned to current machine
          (could be either $\lfloor N/M \rfloor$ or $\lfloor N/M \rfloor + 1$ depending on the machine)
        - Set up $\lfloor n/p \rfloor$ parallel sessions
        - Have each parallel session assign 1 task to each of the $p$ CPUs

        if $n \bmod p \neq 0$ then
            - Set up a final parallel session executing the remaining $n \bmod p$ tasks
              using $n \bmod p$ of the CPUs
```

As with the purely local parallelism, it is useful to briefly look quantitatively at the gains in performance using the cluster approach. The total time taken for the cluster to finish all simulations is directly proportional to the number of parallel sessions on the machine(s) that performs the largest number of parallel sessions. Thus equation (3.6) is used for the general (both remainders non-zero) performance analysis. By dividing the terms by the number of simulations executed in parallel in each of the sessions, we get the parallel computation time T_p given by

$$T_p = k\left\{\left\lfloor \frac{\lfloor N/M \rfloor + 1}{p} \right\rfloor \frac{p}{p} + \frac{\left(\lfloor \frac{N}{M} \rfloor + 1\right) \bmod p}{\left(\lfloor \frac{N}{M} \rfloor + 1\right) \bmod p}\right\} = k\left(\left\lfloor \frac{\lfloor N/M \rfloor + 1}{p} \right\rfloor + 1\right) \tag{3.7}$$

where $k \in \mathbb{R}$ again is some proportionality constant. The speed-up is given by (3.2) as

$$S = \frac{T_s}{T_p} = \frac{N}{\lfloor (\lfloor N/M \rfloor + 1)/p \rfloor + 1}. \tag{3.8}$$

This provides the speed-up factor in the general case, but sometimes $\lfloor N/M \rfloor \gg p$ such that $S \approx pM$. More utilised in this project though, was the special case $N \bmod M = 0$ and $(N/M) \bmod p = 0$. Using (3.5), this gives

$$S = \frac{N}{(N/M)/p} = pM \tag{3.9}$$

which is the ideal performance gain for this approach.

In summary, the work distribution in this section generally leads to executing a total of N simulations by using M computers each executing either $\lfloor N/M \rfloor$ or $\lfloor N/M \rfloor + 1$ simulations in the space of $\lfloor \lfloor N/M \rfloor / p \rfloor + 1$ or $\lfloor (\lfloor N/M \rfloor + 1)/p \rfloor + 1$ parallel sessions, respectively, assuming non-zero remainders (if the remainders are zero, the "+1's" disappear). One consequence of this is that, depending on the sizes of N, M and p, some machines in the cluster may end up having to do one additional parallel session compared to the others. From the perspective of the individual using algorithm 3.1, it may then be wise to choose N such that this is not the case and that all machines perform an equal number of parallel sessions. This is especially important if a single run of the model requires a large amount of time as can often be the case if a small time step increment is used.

To illustrate the above with an example, consider having $N = 61$, $M = 15$ and $p = 4$. Algorithm 3.1 yields 14 machines to get 4 simulations while 1 machine gets 5. When this is distributed locally on each machine, 14 of the machines perform 1 parallel session (using all 4 CPUs) while 1 of the machines must perform 2 parallel sessions (the first session using all 4 CPUs and the second using only 1 of the CPUs). Equation (3.8) confirms this by giving a speed-up of $S = 30$, and hence the total time taken for all simulations to finish is the double of that if the user had chosen to do only one fewer simulation ($N = 60$). The latter choice would eliminate the second parallel session on one of the machines and (3.9) would give a speed-up of $S = 60$. It is then evident that when using a parallel approach like the one described in this section, choosing N with care may save a lot of time without necessarily making a significant compromise on the data amount of collected.

We finalize this section by generalizing some of the pitfalls alluded to in the example above. One solution to the problem of avoiding close to redundant parallel sessions is to choose $N < pM$ such that all simulations are executed simultaneously in one single parallel session on all machines. However, there is often a need to run more simulations than this, and a general rule of thumb is then to try to make sure that the total number of simulations is close to (at least not slightly larger than) a multiple of the total number of CPUs in the cluster. This means we let $N \lesssim npM$ where $n = 1, 2, 3, \ldots$ which implies optimal usage of the cluster. This is knowledge that proved very useful when we ran simulations for chapter 4. In fact, similar to that shown in the above example, large amounts of time were saved by avoiding values of N that would have resulted in time being spent on a few "leftover" simulations on one or two machines while all other machines/CPUs were idle. Consequently, as a result of detailed investigation of the method in this section, we did nearly all of the simulations presented in chapter 4 with the ideal speed-up $S = pM$ given by (3.9).

3.2 Implementation

A large part of the work when developing software is setting a clear plan for the purpose and general workings of the program. An immediate extension of this is possessing ideas and algorithms that enables one to accomplish the aforementioned purpose. In our case, some of these ideas have been described in the previous sections, and while such ideas can be interesting in itself, they are often of little use in a project like this if they are not applied in the code. We devote this final section of the chapter to the implementation of the simulation framework. In line with the introduction to the chapter, the developed code performs a variety of tasks related to the process of running the two-layer quasi geostrophic Fortran model from chapter 2, with the parallel computation in section 3.1 being one of the main features. The framework was implemented as a class hierarchy using the Python programming language and was run in a Linux Red Hat and Ubuntu environment throughout. For now we still refer to the machine cluster in general, but will expand expand upon the actual cluster that was used at the end of the chapter.

As has been mentioned before, the main purpose of the code is to automate the process of running a set of simulations where one parameter is varied between each simulation. The idea is to give the user ease of access by only needing to specify the parameter to vary and the corresponding range of values for that parameter to vary over. Additionally, the user should perform a couple of function calls to the codebase and then the rest is taken care of behind the scenes. While these function calls could be hidden in a wrapper, required user interaction was chosen due to the functions arguments adding further flexibility for the user. The main operations we handle behind the scenes are elaborated on below in order of execution.

Directory generation: When the user has chosen a root directory in which to store all output data and supplied the parameter name as well as the range of values for the parameter, the code applies this input to generate a set of directories (within the chosen root) each with a unique name corresponding to the values of the parameter. Suppose a is the parameter in question; the generated directory naming is such that simulation i with input parameter value a_i is scheduled for outputting data to the directory with name that includes a_i.

File logistics: This section of the code deals with compiling the user specified model and copying user requested files to all the generated output directories. This may be files such as the executable generated by the compilation or other needed input files (e.g. parameter file) for the model.

Cluster network query: When the cluster is used, a few logistical matters need taken care of. In particular, the clusters connectivity to the network is checked. This means using the network protocol Secure Shell (SSH) in order to remotely log in to every computer in the cluster. This process filters out the computers disconnected from the network and returns a set of connected machine names. Additionally, a similar remote login process is used to retrieve the number of CPUs on each of the connected machines. This CPU informations is later used when computing the work distribution.

Work distribution: The code "thinks" of work (tasks/simulations) in terms of directories. There is support for three main approaches to distributing these directories: All directories are set to run sequentially on a single computer, all directories run in parallel locally on one computer utilising all CPUs or all directories run in parallel on all computers, and their respective CPUs, in the available cluster. Algorithm 3.1 is implemented for the latter case while simplified versions apply for the two former. In terms of data structures, the work distribution is stored as a Python dictionary such that one elements key is equal to the machine name and value equal to a two dimensional list with the directory names (tasks) that are to be executed by that machine. The inner sequence of lists corresponds to the earlier mentioned parallel sessions.

Run model: After all output directories are generated, the necessary files are moved to the intended directories and the work distribution has been computed, running the simulations may initiate. The work distribution determines the architecture (serial, local parallel or cluster) on which the runs will occur. Using the cluster approach, every machine is remotely logged into and each CPU on each machine will run the executable in each of the designated directories for each machine. Once every simulation is finished, some clean-up is performed and a report of the successfulness of all runs is generated through the use of logging.

We implemented the above operations in terms of various functions working as methods for a main class `Simulation` and is what is referred to as the "simulation framework". In general, a software framework is collection of code providing general functionality which can be extended upon by user written code. In the spirit of this, we wrote the `Simulation` class as an abstract class to act as a foundation to extend upon for a particular model. It was written as general as possible in an attempt to add to the usability of the code outside this particular project. In fact, the code should work, perhaps only with minor modifications or extensions, on any other numerical model that is run by executing an executable and with an arbitrary number of input files located in the same directory as the executable. With the above functionality in place, the attention moved to creating an instantiation of the framework. In practice, this meant writing subclasses of the abstract framework class utilising and extending its functionality as well as customising its specific behaviour to the required needs. For this project, there were mainly written two such subclasses in order to interface with the Fortran model. Their functionality is outlined below.

Parameter set-up: A subclass `ParamSim` which inherits from `Simulation` acts by generating new model input parameter files based on the values this parameter is requested to vary over. Stores the generated parameter files in their respective output directory to be used as input for the model in each of those directories.

Topography set-up: A subclass `TopoSim` similar to `ParamSim`, but which instead of generating parameter files, generates a set of topography data files as requested by the user through wanting to vary a parameter related to the topography. Then outputs the generated topographies to the output directories ready to be used as model input.

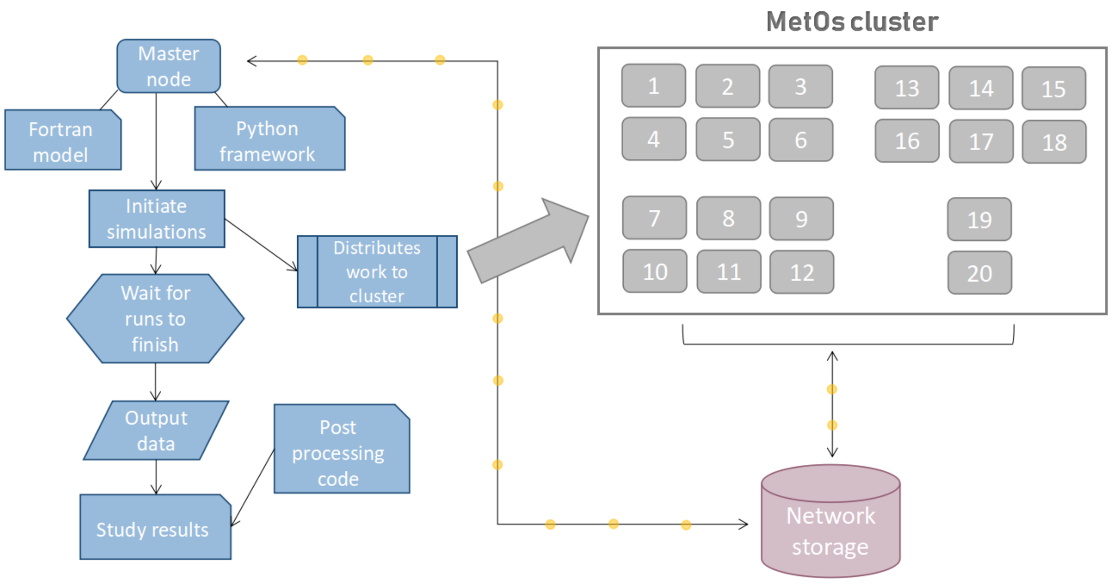


Figure 3.1: Flowchart illustrating the workings of the Python framework for running simulations in parallel on the MetOs cluster at the University of Oslo. It is important to note that the master node and all cluster nodes are connected to the same network storage where all source code and data is stored.

Almost all results we present in chapter 4 were produced using the Fortran model together with the above described Python framework and its two extension subclasses. Additionally, we used the cluster approach in nearly all cases. The referenced cluster was in the form of a set of 20 desktop computers each with 4 CPUs and connected to the same network storage, all located in the masters office of the author. These computers are intended for ordinary work by all master students, but often resources were available to run simulations for this project. Thus the networking query described above involved, in practice, testing the connectivity to the machines in this cluster as well as remotely logging in to all machines and running bash commands initiating every simulation, all automated by the Python code of course. See figure 3.1 for an overview of the process. We did consider making the framework work on the proper supercomputing cluster [Abel cluster, UiO] at the University of Oslo in order to access even more CPUs than available in the ”MetOs” cluster. This was, however, abandoned due to a combination of logistical issues as well as judging the ”MetOs” clusters CPU count to be sufficient for our purposes. After every simulation was done, depending on the particular case, a large set of data was generated in terms of several directories with output files. The data includes time evolution of the upper and lower layer streamfunctions and kinetic energies as well as time evolution of the potential energy of the system. Additional code was then written to gather, combine, process and visualize the data in a meaningful way, leading to the figures in chapter 4.

Chapter 4

Results & discussion

4.1 Introduction

Now we present results we produced by applying the methods described in chapter 2 and 3. We studied the effects of three main (non-dimensional) parameters on the two-layer flow; the bottom friction coefficient r, the topographic amplitude h_0 and the topographic wavenumber(s) k_t, l_t. In other words, the strength of friction as well as both the vertical and horizontal scale of the topography. The dimensional bottom topography is a 2D time-independent function specifying the height $h_B(x, y)$ above the resting depth H_2 in the lower layer. In line with the model, our topography is required to be non-dimensional and it is scaled through (2.37). We reiterate from the introduction the topographies

$$h_B(x, y) = h_0 \cos(k_t x) \cos(l_t y) \tag{4.1}$$

$$h_B(x, y) = h_0 \sin(k_t x) \sin(l_t y). \tag{4.2}$$

Initially we used (4.1), but later also ran simulations with (4.2), the reasons for which shall become apparent below. As the non-dimensional domain is defined on $x, y \in [0, \pi]$, these topographies consist of k_t and l_t (we always used $k_t = l_t$) bumps in the x- and y-direction, respectively, and with a maximum/minimum amplitude of $\pm h_0$ (we always let $h_0 > 0$). Throughout this chapter we have used the initial condition described in section 2.4.2, i.e.

$$\psi_1(x, y, 0) = \frac{1}{4} \sin(k_w x) \sin(y) \tag{4.3}$$

and $\psi_2 = 0$ to initialize the system with a surface layer wave field flow above deep layer at rest. In section 4.3.2 we examined variants of (4.3), but everywhere else $k_w = 4$ was used.

We used an upper non-dimensional simulation time of $t = 100$. Experimentation showed this to be a decent compromise between giving the lower layer enough time properly spin-up (to avoid diagnosing the system with a lower layer still in a spin-up phase) and computational runtime. Furthermore, we defined the value of the total energy TE at $t = 100$ as a fraction of the initial ($t = 0$) energy, i.e.

$$W = \frac{TE(t = 100)}{TE(t = 0)} \tag{4.4}$$

as a metric for examining the system. One could define similar metrics to (4.4) with e.g. potential energy PE, upper kinetic energy KE_1 or lower kinetic energy KE_2. Since we employed the initial condition (4.3) in nearly all our simulations, the initial energy was always $TE(t = 0) \simeq 1.27$ [non-dim]. Depending on the particular set of simulations, W is a function of the three varied parameters, that is $W = W(r, h_0, k_t = l_t)$.

The idea with the above metric is that there is a link between the amount of energy left in the system at $t = 100$ and the lower layer flow during the time evolution of the system. As the system's only energy sink (friction) is present purely in the lower layer, the loss of energy throughout the simulation acts as a measure of the energy transfer to the lower layer (friction cannot act to remove energy without some non-zero flow in the lower layer), and through W we hope to gain knowledge of how the lower layer is spun up as a function of the mentioned parameters. This statement is, in the non-linear case, neglecting the inevitable numerical loss of energy discussed in section 2.5 (figure 2.3). This loss is a potential source of error as the non-linear runs dissipate more energy than dictated by the dynamics. Additionally, amongst the non-linear runs; the ones with the most small scale flow will experience the largest numerical energy loss. However, as the effect occurs on all non-linear runs, we believe our simulations still provide a useful view onto physics at work.

Below we have looked $W(h_0, r)$, $W(h_0, k_t = l_t)$ as well as time evolutions of the energetics in addition the actual flow fields for several simulations varying topographic amplitude, wavenumber and friction. We applied (4.4) to explore the systems response to a wide parameter range both to learn about the parameters effects itself, but also as a tool to "zoom" in on more specific parameter values to which we examined the flow and energetics in more detail. We will first discuss some frictional effects, then look at the linear response before discussing the non-linear system. The chapter ends with a discussion in the large topography regime on the bumps ability to limit transfer to the lower layer and stabilize the surface flow.

4.2 Frictional effects

It is useful to first briefly look at some frictional effects both to compare and distinguish between the topographic effects we observe later. Inspired by the appendix from LaCasce and Brink [2000], we ran simulations for six different bottom friction coefficients r all with a flat bottom. The time evolution of KE_2 and TE is seen in figure 4.1. Without any friction ($r = 0$) there is no energy dissipation (except for the numerical loss in the non-linear case). Very small friction dissipates some energy while increasing r shows a suppression effect on the lower layer both in the linear and non-linear system. In particular, for very large damping ($r = 50$) the TE remains constant throughout while the KE_2 is trapped at zero. Very large friction acts to almost completely block the lower layer from spinning up, and in the non-linear system the upper layer is also stabilised just as in the referenced article. In particular, PE and KE_1 remain roughly constant throughout (although not shown here) indicating the upper layer baroclinically unstable wave field never turns turbulent. For such large damping the two-layer system (both linear an non-linear) appears to approach the 1.5 layer system, i.e. a non-zero upper layer flow above a lower layer at rest.

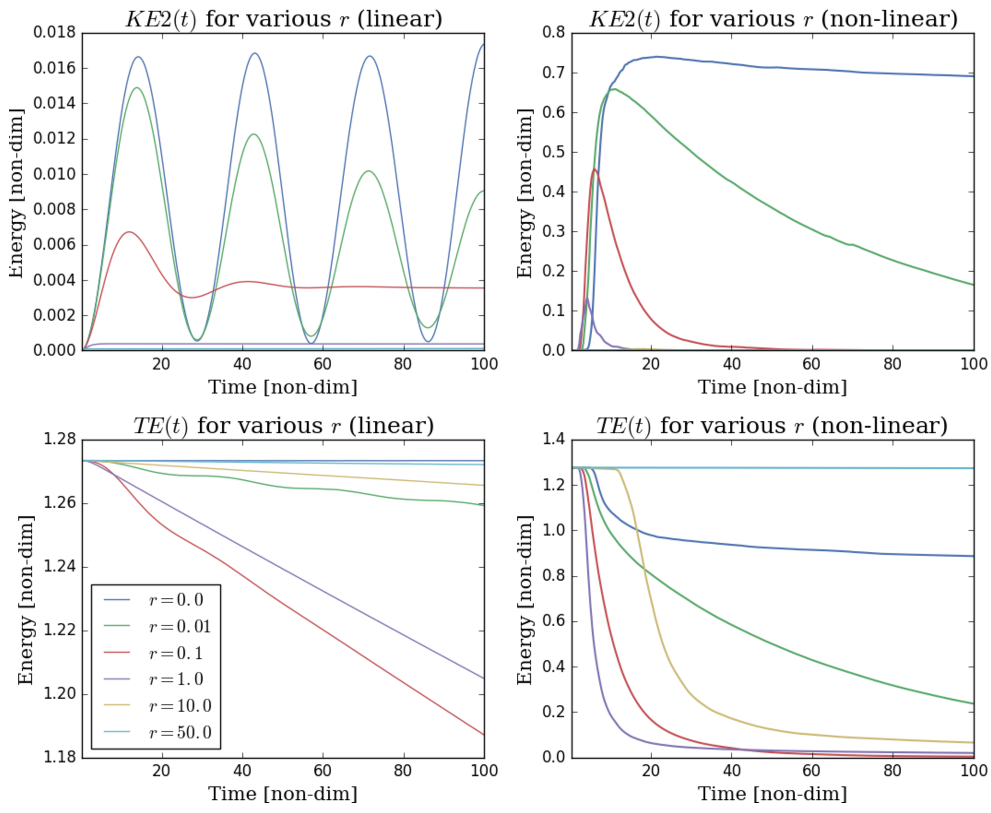


Figure 4.1: Time evolution of total energy and lower layer kinetic energy for 6 different values of bottom friction coefficient r. Linear and non-linear runs done with 128×128 *and* 256×256 *grid resolution, respectively, and corresponding time steps* $\Delta t = 5 \cdot 10^{-4}$ *and* $\Delta t = 2.5 \cdot 10^{-4}$.

The intermediate values of r strikes a balance between the above both in the linear and non-linear case. The red lines show that $r = 0.1$ is optimal for energy dissipation in the linear system by removing energy in the lower layer at just the right rate through balancing allowance of deep spin-up and subsequent dissipation. In the non-linear system, the violent turbulent flow results in $r = 1$ being the optimal. For weaker or stronger friction in either systems, the dissipation is less.

Observing the linear plots, we note the tendency for all (except the friction-less run) KE_2 evolutions to approach a steady state (most visible for $r = 0.1$). The motion of the interface gives some rise in KE_2, then it oscillates before remaining at a constant value. TE is decreasing over the same time and so is PE (not shown here) indicating a balance between conversion of PE to KE_2 and removal of KE_2 by friction. The steady state value depend on r, and with larger r this value gets smaller before, at massive friction ($r = 50$), just remaining at zero from the very start.

Overall, the above discussion seem to be related to a similar dual role played by friction mentioned by LaCasce and Brink; very small friction, while allowing lower layer spin-up, is not large enough to dissipate that much energy. Conversely, very large friction blocks the lower layer spin-up and there is never any energy there to dissipate.

4.3 Topography: Linear simulations

We start by looking at numerical solutions to the linear versions of the layer equations (2.24) and (2.25) (remember the model actually solves (2.38) and (2.39)), i.e. neglecting the non-linear terms. Since $\psi_2 = 0$ initially, the linear layer equations tell that the lower layer flow is forced and spun-up by the motion of the interface h_i. In this section we look at how spin-up of the lower layer is affected by topographic amplitude and wavenumber as well as friction. These parameters modify the latter two terms in the lower layer equation 2.25 and might thus have significant impacts the time evolution of ψ_2. All model runs in this chapter were performed with a time step $\Delta t = 5 \cdot 10^{-4}$ and a resolution of 128×128.

4.3.1 Response to topographic amplitude

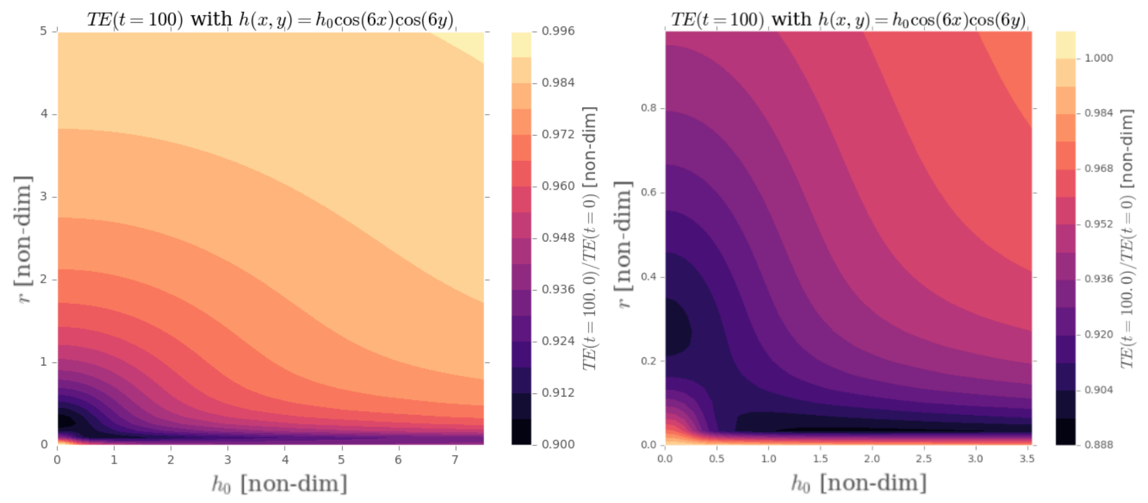


Figure 4.2: Left panel displays total energy in the linear system at ($t = 100$) as a fraction of the initial ($t = 0$) total energy plotted for a set of 60 different friction coefficients $r \in [0.005, 5]$ and topography 60 amplitudes $h_0 \in [0, 7.5]$. Right panel shows a similar set of simulations, but zoomed in on the lower left region of the left panel obtaining a higher resolution for this area.

We looked at the linear response with $k_t = l_t = 6$ cosine topography (4.1) for several bump amplitudes h_0 and friction coefficients r. Using our simulation framework we found numerical solutions for 60 values of $h_0 \in [0, 7.5]$ and for each h_0 value, 60 values of $r \in [0.005]$, a total of 3600 model runs. We extracted the total energy at $t = 100$ from each of those runs and the left panel in figure 4.2 shows the metric $W(h_0, r)$, that is how much energy is left in the system ("inverse" of the dissipation) as a function of h_0 and r. The range for h_0 was chosen to examine the region of "smaller" topographies and the friction range was motivated by the previous section. The right panel shows another 3600 simulations for on a large (h_0, r) resolution in the lower left corner of the left panel.

As expected based on the previous section, very small friction and a flat bottom (lower left corner of the figure 4.2) result in very little dissipation seen by $W \approx 1$ here. Then we note the trend along the r-axis shows the same as the r-dependence seen in figure 4.1, i.e. small dissipation at small r, maximum dissipation with $W \approx 0.9$ at intermediate values (however dependent on h_0) and a blocking effect for large r. Interestingly, it appears topography h_0 exhibits similar behaviour to friction. For larger friction ($r > 0.1$) there is generally more energy is left in the system with increasing h_0. There is here a monotonically increasing dependence on h_0. In this regime, the suppression effect of friction coexists with the apparent suppression effect of the bumps and the upper right corner of the figure shows almost no dissipation at all.

For $r < 0.1$ the dependence on h_0 is more complex. For very small r, W is monotonically decreasing with h_0, but moving up towards $r = 0.1$, non-monotonic dependence on h_0 is seen; energy left is large to about $h_0 \approx 0.25$ before rapidly decreasing reaching a minimum then allowing significant energy loss across values of h_0, before slightly increasing (hard to see) for larger h_0.

The simulations in figure 4.2 seem to have covered most of the variation with the two parameters except for the large h_0-dependence at small r. So a question we are left with is whether the bumps can show a proper lower layer suppression effect for small friction as well, i.e. without the need for strong friction. To investigate this, we perform large h_0 simulations at small friction in section 4.5.

Motivated by a desire to further study the topographic response, we took a more detailed look at some select simulations in figure the small friction regime. Using small r means we will be able to study the impact of the bumps without the blocking effect of friction. We used $r = 0.01$ and re-ran the model for $h_0 \in \{0, 0.01, 0.05, 0.25, 1, 5\}$ with output of streamfunction and energy time evolution (these runs corresponds to points along a horizontal line at $r = 0.01$ in figure 4.2).

The time evolution of energy can be seen in figure 4.3. The kinetic energies cycle sinusoidally due to the wave propagation on the surface and interface. Looking at all energies it is evident that $h_0 \in \{0, 0.01, 0.05\}$ have little to no impact on the response while the three larger values $h_0 \in \{0.25, 1, 5\}$ show quite modified behaviour. For the former three topographies, $h_0 = 0.01$ and $h_0 = 0.05$ are seen to be approximately equal to the flat bottom solution and KE_2 oscillates and decreases in amplitude slowly due to friction (similar to $r = 0.01$ in 4.1) for all three topographies. Whereas for the three larger amplitudes there are significant energy transfer to the lower layer facilitating the larger energy loss seen in TE.

There is a clear jump in response between $h_0 = 0.05$ and $h_0 = 0.25$ and this appear related to an important event between these values; the closing of stationary potential vorticity q_s contours. The stationary PV in the lower layer is given by the time-independent part of the lower layer PV given in (2.18), i.e. (in non-dimensional variables (dropping primes) through use of the scalings in section 2.1.3)

$$q_s(x, y) = \beta y + h_B(x, y) \tag{4.5}$$

with h_B given by (4.1) and non-dimensional $\beta = 1$ (used in the model) through (2.42). The contours of (4.5) closes if there is a local maximum (or minimum) in q_s. Insert for h_B and look for maximum when the y-derivative vanishes through

$$\frac{\partial q_s}{\partial y} = \beta + \frac{\partial h_B}{\partial y} = \beta - l_t h_0 \cos(k_t x) \sin(l_t x) = 0. \tag{4.6}$$

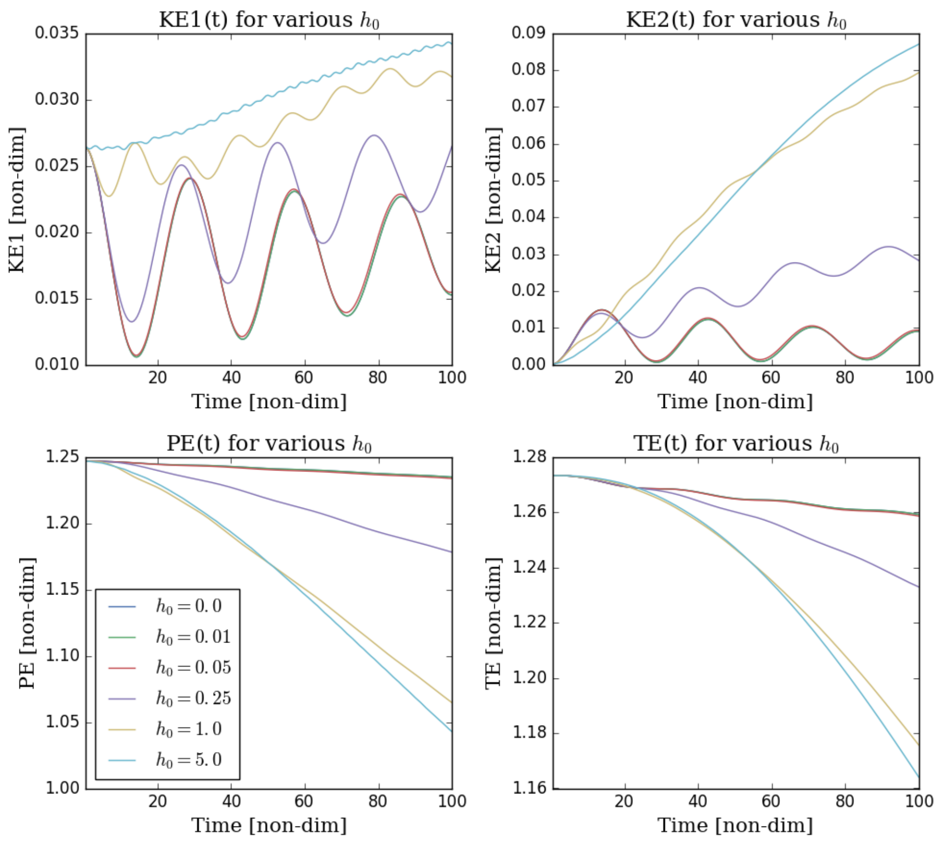


Figure 4.3: Time evolution of kinetic, potential and total energy of the linear system for flat bottom and 5 other values of h_0. A small friction coefficient of $r = 0.01$ was used.

The maximum value of $\cos(k_t x)\sin(l_t x)$ is 1 and the critical value for closed contours becomes

$$h_0 = \beta / l_t \tag{4.7}$$

meaning that if $h_0 < \beta/l_t$ there is no maximum and no closed contours. Inserting for $l_t = 6$ gives $h_0 > 0.167$ to achieve closed contours. Figure 4.4 shows the q_s contours and gives a visual perspective by considering $h_0 = 0.1$ and $h_0 = 0.3$. Notice that they close for the latter. Also note that $h_0 = 0.05 < 0.167 < 0.25$ implying the contours are closed for the three larger h_0 runs (in figure 4.3), but not for the three smaller. In Welander [1968] (also e.g. Nøst and Isachsen [2003] or LaCasce et al. [2008]) it is discussed that closing q_s contours significantly modify the lower layer flow around the bumps making the spin-up more prominent and producing "topographic gyres" as part of the solution. It appears the same is happening in our case and is looked more at below.

To gain further knowledge of the dynamics at work, we extracted the lower layer streamfunction ψ_2 from all six simulations in 4.3 to examine the flow field in each of these cases. To look for prevalent topographic effects in ψ_2, we computed the time mean over the latter half of the simulation, i.e. from $t = 50$ to $t = 100$, through

$$\overline{\psi}_2(x_i, y_j) = \frac{1}{q-p+1}\sum_{n=p}^{q} \psi_2(x_i, y_j, t_n) \tag{4.8}$$

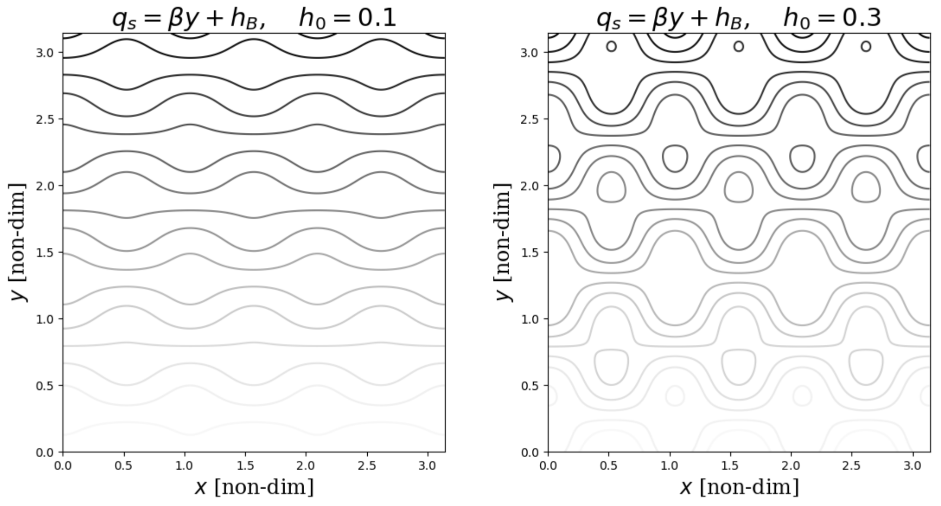


Figure 4.4: Stationary PV contours $q_s = \beta y + h_B$ for $h_B = h_0 \cos(6x)\cos(6y)$ with $h_0 = 0.1$ (left) and $h_0 = 0.3$ (right). Darker grey corresponds with larger q_s values.

where i, j are indices for the grid cells and n for the time step while p is a time step index such that $t_p = 50$ and q such that $t_q = 100$. In words, (4.8) implies computing the average field by, for each grid cell, summing up all ψ_2 values between $t = 50$ and $t = 100$ and then dividing by the number of time steps in said interval. The reasoning for using the time mean field was to see if there are consistent flow features associated with the topography. Also, the latter half mean was used to diagnose the field after the main spin-up phase was completed.

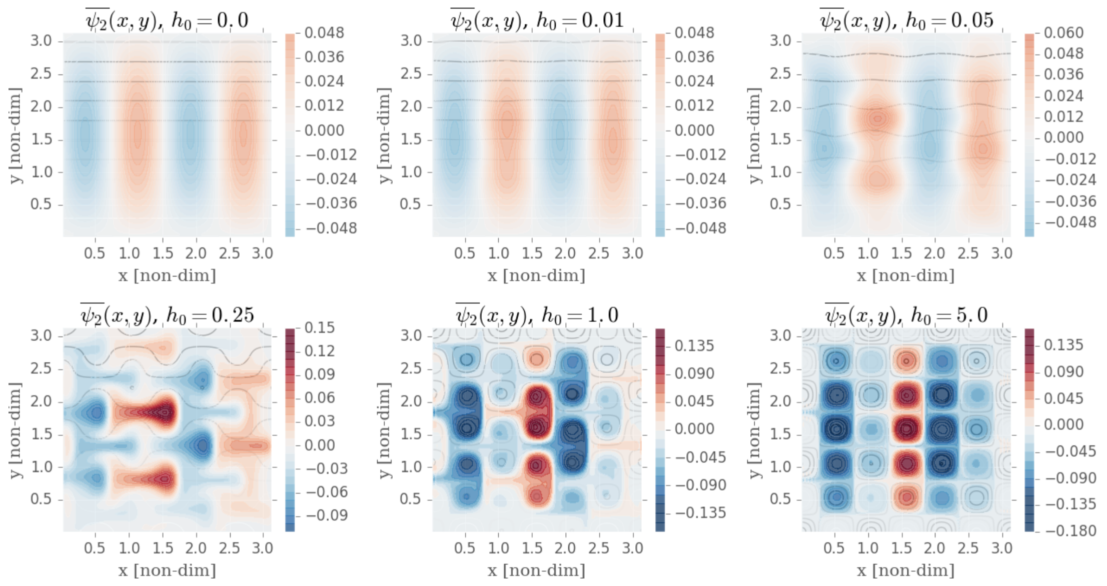


Figure 4.5: Time mean of ψ_2 over the latter half (from $t = 50$ to $t = 100$) of the linear simulation for 6 different values of h_0. A small friction coefficient of $r = 0.01$ was used and the color range is the exact same for all panels. All 6 fields are overlaid their respective topography altered q_s contours.

Figure 4.5 shows $\overline{\psi}_2(x, y)$ for the six values of h_0 overlaid their respective topography altered q_s contours. All panels are with the same color range making not just the colorbar values, but also the colors itself, directly comparable. In all simulations we have westward propagating Rossby waves. In correspondence with the energy evolutions in 4.3, we observe the mean flow for $h_0 = 0.01$ and $h_0 = 0.05$ to be largely similar, both in form and magnitude, to the flat bottom ($h_0 = 0$) mean flow. However, the three larger bump cases show significantly modified mean flow both in terms of magnitude (being about three times stronger in amplitude) and appearance. The altered appearance is in the form flow spinning around the bumps super-positioned on the westward propagating Rossby waves with the bump trapped flow most visible in the $h_0 = 5$ case. This corresponds well with the above discussed closing of the q_s contours between the $h_0 = 0.05$ and $h_0 = 0.25$ run. This is a significant factor in the flow response around the bumps. While smaller bumps cause distortions to the wave field, proper bumps flow is not observed until they close, i.e. for $h_0 > 0.25$ amongst these runs. Based on this, it appears the larger dissipation for larger h_0 seen in figure 4.3 and figure 4.2 (for small r), is a result of the larger bumps supporting more lower layer spin-up through this bump flow.

4.3.2 Response to topographic wavenumber

We also investigated the linear response to topographic amplitude h_0 together with topographic wavenumber $k_t = l_t$ for a similar small friction coefficient $r = 0.01$. In particular, we looked at $W(h_0, k_t = l_t)$ for two different initial upper layer flows while $\psi_2(x, y, 0) = 0$ as usual, and performed 600 model runs in $(h_0, k_t = l_t)$ space using the simulation framework. The metric (4.4) is plotted in figure 4.6 for initial condition (4.3) with $k_w = 4$ (left) and $k_w = 6$ (right).

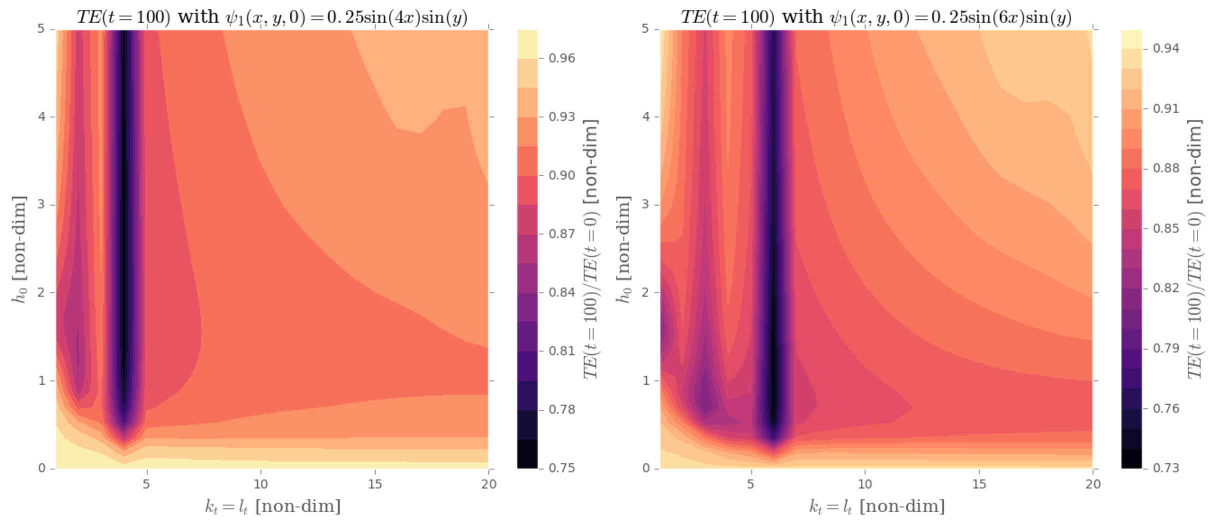


Figure 4.6: Total energy in the linear system at ($t = 100$) as a fraction of the initial ($t = 0$) total energy plotted for a set of 20 topographic wavenumbers $k_t = l_t \in [1, 20] \in \mathbb{N}$ and 30 amplitudes $h_0 \in [0, 5]$. Done for two different x-wavenumbers in the initial wave and with $r = 0.01$ for all runs.

Again, we see the non-monotonic dependence on h_0 (from figure 4.2 with small r) for nearly all k_t, l_t; small dissipation at small h_0, the rapidly increasing at intermediate values (in line with q_s closing), but too large h_0 actually suppresses the energy loss. Also note that that the dependence on h_0 is stronger for larger k_t, l_t.

The by far most prominent feature is the massive dissipation maximum for $k_t, l_t = 4$ and $k_t, l_t = 6$ together with $k_w = 4$ and $k_w = 6$, respectively. In other words, there is large energy loss when the topographic wavenumber is equal to the x-wavenumber of the initial (and sustaining because of linearity) wave, i.e. when $k_t, l_t = k_w$. In these valleys, there are around 75% energy left in the system indicating large frictional damping further suggestion a large energy transfer to the lower layer. In figure 4.2, the smallest amount of energy left was 90%, a significantly larger number, indicating that the effects seen here are the main source of energy transfer to the lower layer.

Dynamically, the dissipation is preferential at a topographic scale resembling the wave scale as this is seen to be the optimal way to force the flow over the closed q_s contours. In this case, the layer interface is displaced over the entirety of each bump rather than only parts of it, enhancing the bump trapped flow. This will also, to some extent, be the case when k_t generally is close to k_w as reflected by the visibly enhanced loss for such configurations. In particular, there is a secondary dissipation maximum when the topographic wavenumbers are half of that to the initial condition, i.e. at $k_t, l_t = 2$ for $k_w = 4$ and $k_t, l_t = 3$ for $k_w = 6$.

4.4 Topography: Non-linear simulations

Having discussed the linear system, we now move onto looking at numerical solutions to the full non-linear equations (2.38) and (2.39). We performed similar simulations to those in the linear case, that is we looked at topographic effects both in terms of amplitude h_0 and wavenumber $k_t = l_t$ for comparable ranges as before. The non-linear runs are inherently different due to the initial instability causing subsequent turbulent flow and it is interesting to see if we spot similar features (and/or different) to that of the linear simulations. As before, we used the initial condition (4.3) with $k_w = 4$ and the cosine bump topography (4.1) throughout this section. A resolution of 256×256 and a time step of $\Delta t = 5 \cdot 10^{-4}$ was used for all runs in this section except for figure 4.8 and 4.9 where $\Delta t = 2.5 \cdot 10^{-4}$ was used.

Figure 4.7 is the non-linear analogue to 4.7 and shows a section of the full data set $W(h_0, r)$ generated from running the model for 45 values of $r \in [0.005, 0.4]$ and 45 values of $h_0 \in [0, 3]$. The reason for not including the full friction range is simply that the black section seen in the plot extends all the way to the upper value $r = 0.4$. About 20% of the initial total energy is left at the smallest frictions while larger r dissipates away almost all energy. Had we also here ran for very large friction, up to about e.g. $r = 50$, we would have observed the lower layer blocking effect seen in figure 4.1 with nearly 100% total energy left (or $T(t = 100) \approx 1.27$ [non-dim]). In contrast to figure 4.2, there is almost no dependence on h_0 in this range suggesting that the topography term in (2.38) and (2.39) is too small to compete with the larger non-linear terms.

As in the linear case, we examined the flow response for $h_0 \in \{0, 0.01, 0.05, 0.25, 1, 5\}$ further and the energetics are shown in figure 4.8. This confirms the observations in the previous paragraph; there are only minor differences between the different values of h_0

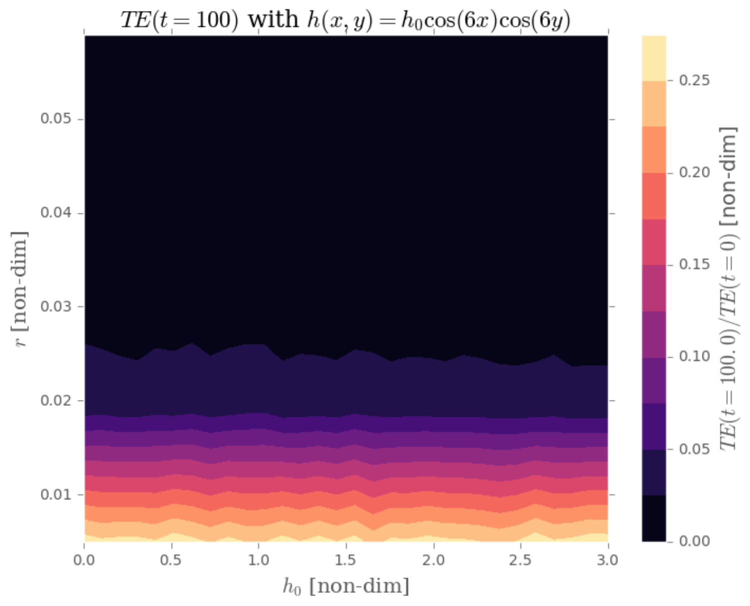


Figure 4.7: Total energy in the non-linear system at ($t = 100$) as a fraction of the initial ($t = 0$) total energy for a set of 5 different friction coefficients $r \in [0.005, 0.06]$ and 45 topographic amplitudes $h_0 \in [0, 3]$.

with $h_0 = 5$ providing the largest energy loss, but they are all are close to the flat bottom evolution. The fact that $h_0 = 5$ is the most dissipative also in the non-linear system is interesting and suggesting that the lower layer spin-up effect we saw in the linear system also has some impact here, albeit much smaller relatively speaking due to the turbulent transfer also happening.

In figure 4.9 are the corresponding latter half time averaged ψ_2 fields. The different topographies give quite similar response in $\overline{\psi}_2$, all quite close to the flat bottom simulation even though the q_s contours have for $h_0 > 0.25$. That being said, $h_0 = 5$ provides slight distortions indicative of the above mentioned topographic spin-up having an impact on the mean field. The other values for h_0 does not show any visible evidence of the bumps. We shall see in section 4.5.1, that larger bumps have much more drastic impacts on the non-linear flow field than seen here.

The flow we observe in all panels are Fofonoff-like gyres (originally described in Fofonoff [1954]), emerging by a non-linear transfer to larger scales though an inverse energy cascade. They are similar to the mean field gyres from the single-layer quasigeostrophic turbulence simulations in Dukowicz and Greatbatch [1999] and LaCasce [2002], the former to which we, as noted earlier, used a similar initial condition.

We also examined the non-linear analogue to figure 4.6 to see if the non-linear system also gave more dissipation for $k_t = l_t \approx k_w$. Figure 4.10 shows $W(h_0, k_t = l_t)$ for 600 simulations with h_0 reaching about ten times as large as in previous sections. The larger h_0 values was motivated by the lack of topographic effects as seen so far for the non-linear response in this section (the reason why we did not simply use larger topography for all the earlier presented results is a numerical issue discussed in section 4.5). Based on the figure, the there is no visible larger dissipation for $k_t, l_t = 4$ (corresponding with the initial

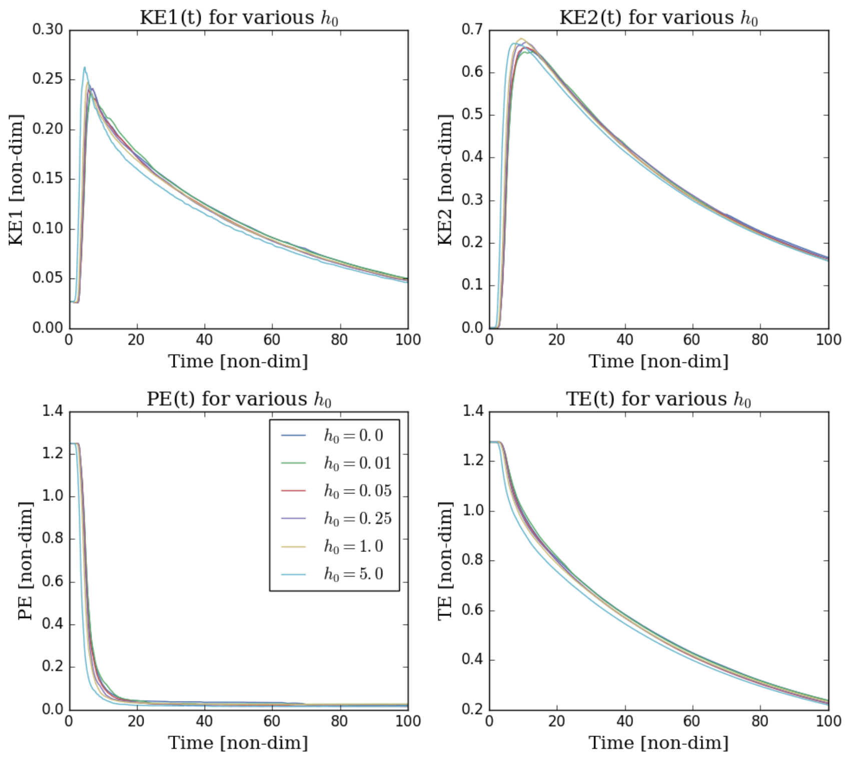


Figure 4.8: Time evolution of kinetic, potential and total energy of the non-linear system for 6 different values of h_0. A small friction coefficient of $r = 0.01$ was used.

condition wave scale) and, as opposed to the linear response, this effect does not appear to be important here compared to the turbulent transfer to the lower layer. Note that the energy left in the system is very small compared to the minimum of 75%+ in the linear system indicating the instabilities in the non-linear system is a far more violent and effective source of deep energy transfer than any effect we saw in the linear system.

Finally, we note that there generally more energy left in the system (although still only about $5\% - 7\%$) as we move towards the larger values of h_0, especially at the smaller k_t, l_t. This smaller dissipation seem to suggest the topography somehow limit the energy transfer to the lower layer. This sparked an interest to investigate even larger h_0 and closer study the associated flow fields which is the focus of the next section. On the way to larger h_0 in figure 4.10, there are signs of a non-monotonic dependence on h_0 for all k_t, l_t; an energy minimum at intermediate values before increasing for larger h_0. This effect is reminiscent of similar behaviour by friction in sections 4.2 and 4.3.1 as well as linear response to topography in figure 4.2 and is something we will see more of.

4.5 Large topography runs

Motivated by the end of the previous section, we sought after numerical solutions in the regime of large values of the topographic amplitude h_0 for some different topographic wavenumbers $k_t = l_t$, still using the standard initial condition (4.3) with $k_w = 4$.

The large h_0 values we employ in this section correspond roughly to a dimensional values

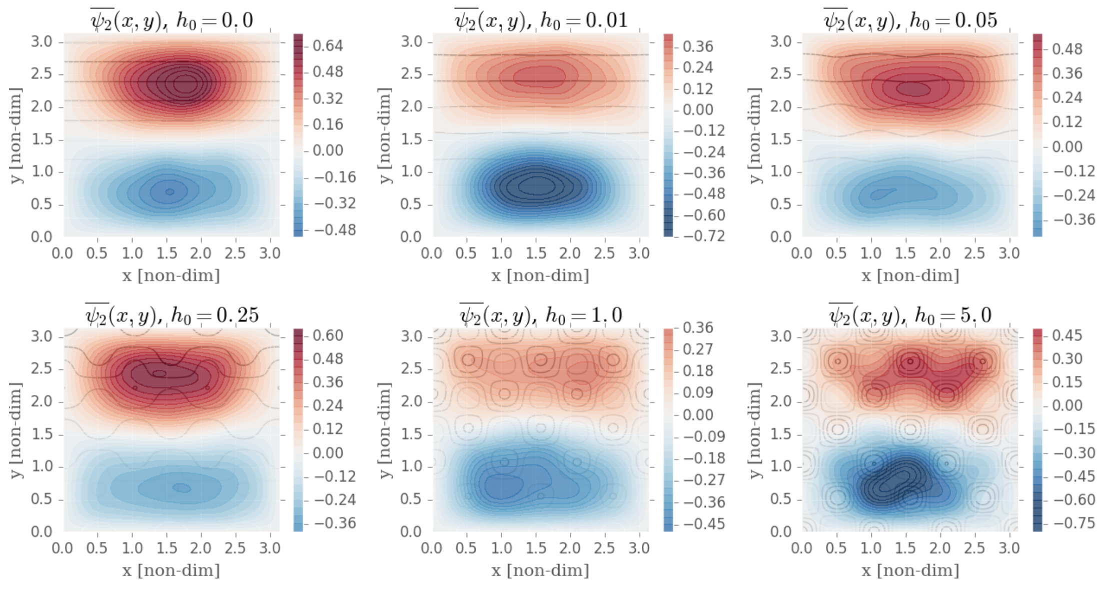


Figure 4.9: Time mean of ψ_2 (non-dim) over the latter half (from $t = 50$ to $t = 100$) of the non-linear simulation for 6 different values of h_0. A small friction coefficient of $r = 0.01$ was used. The color range is the exact same for all panels.

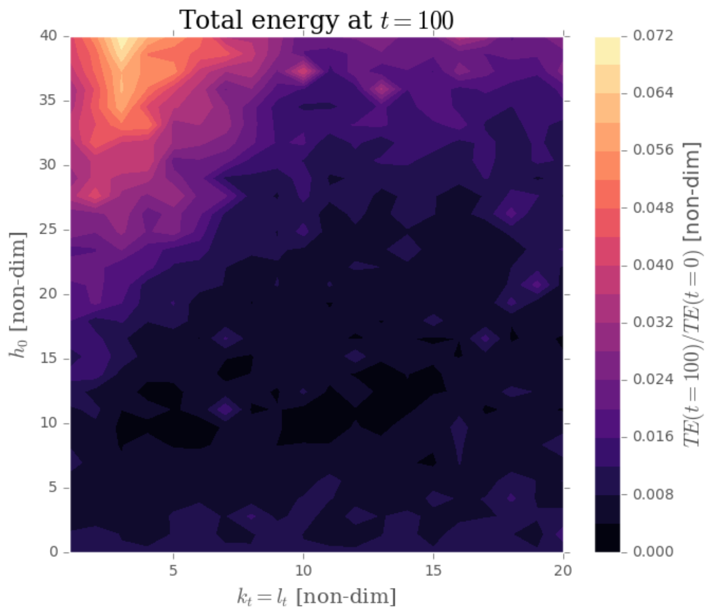


Figure 4.10: Total energy in the non-linear system at $t = 100$ as a fraction of the initial ($t = 0$) total energy plotted for a set of 20 topographic wavenumbers $k_t, l_t \in [1, 20] \in \mathbb{N}$ and 30 amplitudes $h_0 \in [0, 40]$. Small friction $r = 0.04$ was used.

of up to 500 meters, so still within quasi-geostrophic assumptions considering $H_2 = 4000$ meters.

As before, we used a small friction coefficient, now $r = 0.04$. The reason for investigating these larger bumps separately from the smaller ones, was mainly due to numerical stability in the model. In particular, h_0 and $k_t = l_t$ modifies the numerical stability condition for the schemes used to solve (2.38) and (2.39) resulting in a much stricter requirement on the time step. This was discovered through experimentation, and likely due to topography supporting topographic waves propagating at large speeds violating the CFL criterion that would otherwise be satisfied without the presence of topography. This turned out to be a problem especially in the linear case; we were unable to run the model for large h_0 and k_t, l_t as even decreasing the time step dramatically still resulted in numerically unstable solutions (these are mentioned below). We were surprised the instability issue first and foremost occurred in the linear case, contrary to the more typical cases where stable non-linear solutions are a lot tougher to achieve. In our runs, it would seem the non-linear triad interaction transferred energy to other wavelengths causing these fast topographic waves to be lost making the linear runs more exposed to violation of the stability criterion.

Through experimentation we decreased the time step by a factor of 50 to $\Delta t = 10^{-5}$ to achieve numerical stability (for most runs). This further implied the need for 50 times as many time steps in order to reach $t = 100$ meaning 50 times as large runtime. Therefore, some of the results below contain fewer simulations than that of sections 4.3 and 4.4. Additionally, a resolution of 256×256 was used for all runs, even the linear ones we did for this section. The reason for the latter is found in the final paragraph of section 2.5; amongst other, we used $k_t, l_t = 30$ meaning 30 bumps in either direction of the domain. This is barely resolved on a 128×128 grid and hence a 256×256 grid is preferential for such rapidly varying bumps. For section we have used the sine bump topography (4.2) (except in figure 4.11) as the cosine bumps are $+h_0$ in magnitude in the domain corners which gave some strong corner flow making the streamfunction elsewhere hard to interpret. However, all sine bump runs were also done for the cosine bumps and the response was very similar meaning either topography would yield the same discussion.

Figure 4.11 shows the metric (4.4) applied for upper kinetic, lower kinetic, potential and total energy in the non-linear system for 45 simulations ranging from flat bottom to $h_0 = 100$ performed for 3 topographic wavenumbers. We also wanted to produce a similar figure in the linear system, but numerical stability was too hard to achieve for $k_t, l_t \in \{18, 30\}$. The small uneven oscillations reflect run-to-run variations, typical of turbulent systems. Also, as all these energy values are normalized with their initial values, disregard the massive KE_2 values; initially the lower layer is at rest so the values emerge from dividing by a value very close to zero. That being said, the KE_2 graphs are still valid in terms of shape.

An interesting aspect is that TE decreases slightly to a minimum at around $h_0 \approx 15$, for all wavenumbers, before increasing significantly up to $h_0 = 100$ reaching e.g. 30%+ energy left in the system for $k_t, l_t = 18$. Thus we see a dissipation maximum for intermediate values h_0 (while quite similar for small h_0 values) and some blocking/stabilising effect for large values of h_0. This non-monotonic dependence is the same as in figure 4.10 and similar to the linear observation in figure 4.2. In addition it reminds us of similar behaviour by friction in figure 4.1 and 4.2.

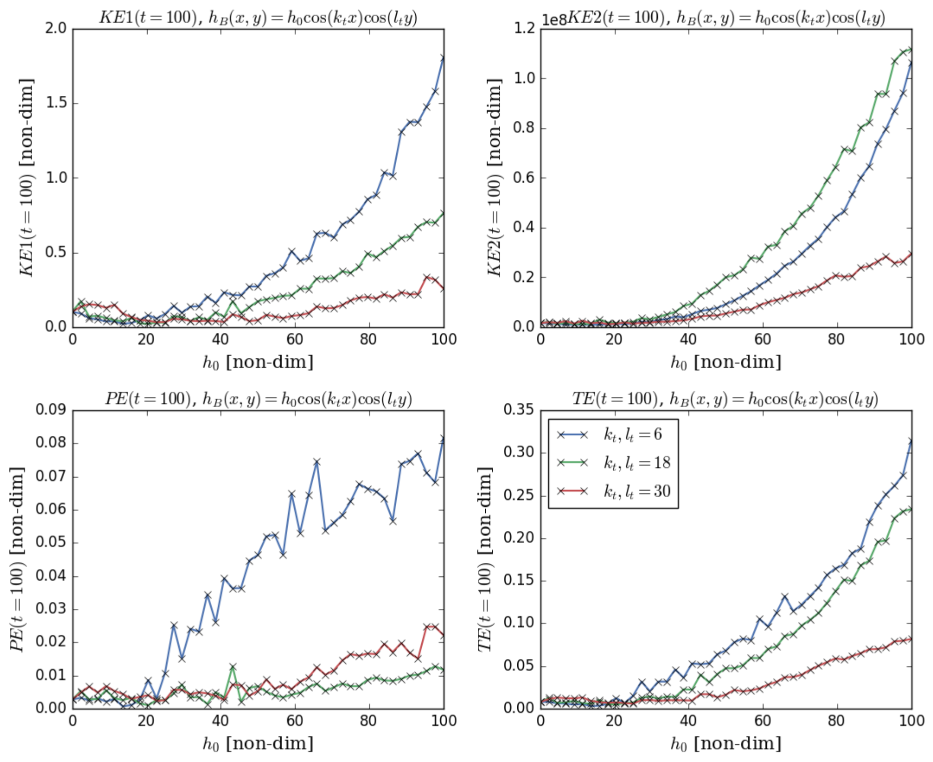


Figure 4.11: Upper kinetic, lower kinetic, potential and total energy in the non-linear system at $t = 100$ as a fraction of their respective initial ($t = 0$) values. plotted for a set of 3 topographic wavenumbers $k_t = l_t \in \{6, 18, 30\}$ and 45 amplitudes $h_0 \in [0, 100]$. Each run used $r = 0.04$.

4.5.1 Flow response

Motivated by the previous paragraph, we further examined the flow response for select parameters. We focused on $k_t, l_t = 6$ and $k_t, l_t = 30$ and, for each of those, ran the model with $h_0 \in \{0.1, 5, 60, 100, 150\}$ to compare both small/large $k_t = l_t$ and small/large h_0. We ran both linear and non-linear simulations for these parameters and the latter half time mean deep flow overlaid q_s contours corresponding to their respective topographies can be seen in figure 4.12 (we left out the $h_0 = 150$ runs for readability, but their response were similar to the $h_0 = 100$ cases and are discussed in section 4.5.2). The linear $k_t, l_t = 30$, $h_0 \geq 60$ runs developed numerical instabilities and are therefore excluded in the figure. To prepare for the below discussion we also mention that figure 4.13 shows the corresponding non-linear upper layer flow at $t = 35$ (using the time mean in the upper layer made less sense as the prevalent bump flows are predominantly features of the lower layer and averaging captures a lot of bypassing flow cluttering the mean field (although we shall below see that there exists quite barotropic flows for which the upper layer mean flow would be informative)).

The linear $\overline{\psi}_2$ fields for $k_t, l_t = 6$ are similar to what we saw in figure 4.5 in $h_0 = 0.1$ and $h_0 = 5$, i.e. is some bump trapped flow super-positioned on the westward moving wave field. The two parameters differ in response as before though; distorted wave-like flow for open q_s contours ($h_0 = 0.1$) and bump flows for closed q_s contours ($h_0 = 5$). For $h_0 \geq 60$ the response is very similar to $h_0 = 5$, but the amplitude is slightly smaller. The

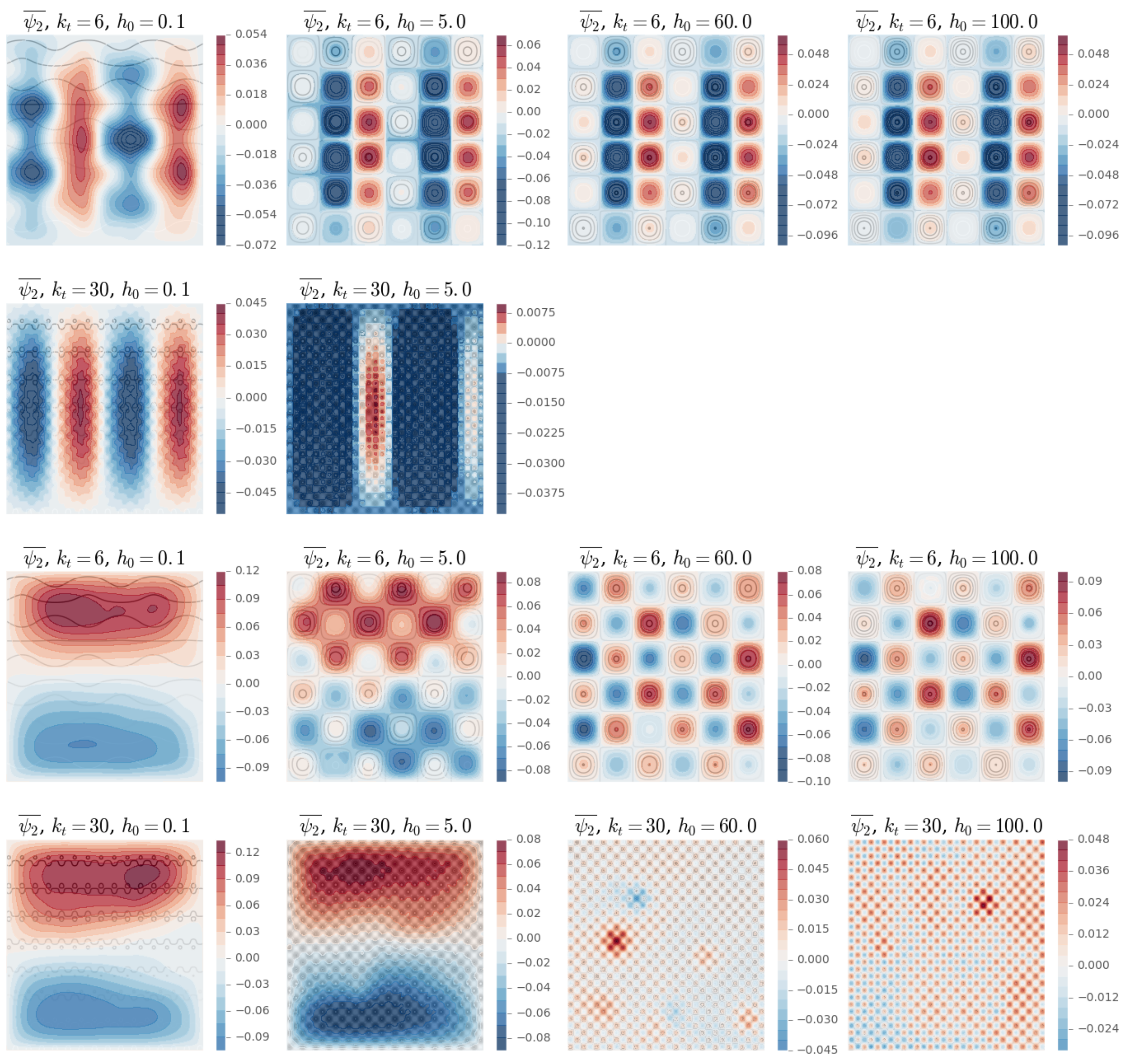


Figure 4.12: Latter half time mean $\overline{\psi_2}$ for $k_t = l_t \in \{6, 30\}$ and $h_0 \in \{0.1, 5, 60, 100, 150\}$. The top 6 panels are linear solutions while the bottom 8 are non-linear. All runs done with $r = 0.04$. All panels have x- and y-axis ranging from 0 to π. The linear $k_t, l_t = 30$, $h_0 \geq 60$ runs are excluded as they went numerically unstable.

$k_t, l_t = 30$ (recall that topographic wavenumber also affects the closing of the q_s contours, in partiuclar making them close at smaller h_0) case show smaller impact on the overall flow field, albeit with visible topographic distortions to the wave. The magnitudes of the streamfunctions are even smaller for this many-bump case and especially for $h_0 = 5$. In all these runs, the spin-up of the lower layer happens through interfacial motion and spin-up around topography, but to different degrees as seen in e.g. figure 4.2. To keep the plot count reasonable, linear ψ_1 plots have been omitted, but seen in those are an upper layer of largely similar wave structure as the initial condition, but slightly distorted and more so for $k_t, l_t = 6$ than $k_t, l_t = 30$, that is the bump flow has less penetration to the upper layer when there are more bumps though the q_s contours are more closed). Like topographic waves, the vertical scale of the flow trapped over the bumps increases with the bump horizontal scale as in (e.g. LaCasce [1998]).

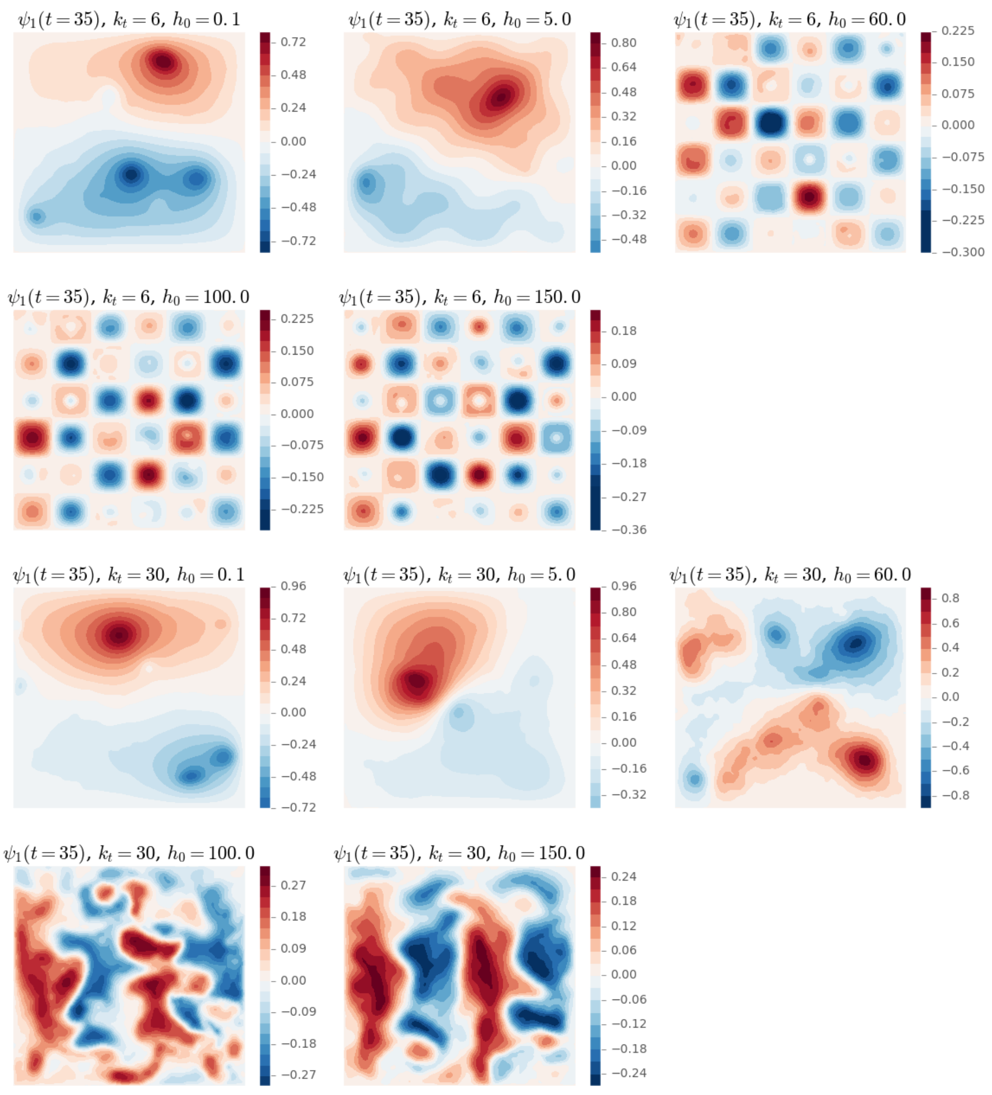


Figure 4.13: Snapshot of non-linear $\psi_1(t = 35)$ for $k_t, l_t \in \{6, 30\}$ and $h_0 \in \{0.1, 5, 60, 100, 150\}$. All panels have x- and y-axis ranging from 0 to π.

We move onto the non-linear solutions. With $h_0 = 0.1$, the evolution is nearly the same as over a flat bottom for both $k_t, l_t = 6$ and $k_t, l_t = 30$, i.e. the inverse cascade produces Fofonoff gyres in $\overline{\psi}_2$ (as also seen in figure 4.9), in the north and south, without visible bump features. The $\psi_1(t = 35)$ field show the gyres with vortices superimposed. Note the anti-cyclonic (red) vortices are in the north and the cyclonic (blue) in the south.

For $h_0 = 5$ the q_s contours are closed and the situation it is more complex; with $k_t, l_t = 6$, the gyres are still evident in $\overline{\psi}_2$, but are concentrated on the bumps. So we observe a hybrid flow between bump features and the Fofonoff gyres. The bump flows are of the same sign as the gyre in which they are contained, i.e. anti-cyclonic bump flows in anti-cyclonic northern gyre while cyclonic bump flows are in the cyclonic southern gyre, indicating the bump flows are driven by the surface flow. With $k_t, l_t = 30$ though), the gyres still dominate and bump flow is barely visible. This is counter-intuitive as the q_s contours are even more closed for $k_t, l_t = 30$, but evidently it is harder to drive the bump flows when there is a large discrepancy between the scales of the bottom topography and the surface flow. This is something we also saw in the linear runs (see linear $\overline{\psi}_2$ or figure 4.6) and it appears the effect is of significance in the non-linear system as well. The corresponding $\psi_1(t = 35)$ fields are like those with $h_0 = 0.1$, just with slightly more evidence of the bumps, most notably for $k_t, l_t = 6$.

With $h_0 \geq 60$ and $k_t, l_t = 6$, there are no longer Fofonoff gyres visible in $\overline{\psi}_2$. Rather the mean flow is in fact dominated by the bumps and with anti-cyclonic flow around seamounts and cyclonic flow around depressions (grey q_s contours represent seamounts and white depressions) as predicted by LaCasce and Nycander [2004] and Bretherton and Haidvogel [2004]. In the upper layer, $\psi_1(t = 35)$ also shows evidence of the bumps and no gyres, suggesting the energy has transferred into bump circulations. Although, not shown here, we did examine $\psi_2(t = 35)$ as well and the bump dominated flow is nearly barotropic (because the associated topographic waves with $k_t, l_t = 6$ are nearly barotropic), that is the $\psi_1(t = 35)$ anti-cyclones and cyclones are aligned with those in $\psi_2(t = 35)$.

Finally, for $k_t, l_t = 30$ ($h_0 \geq 60$ still), the response in $\overline{\psi}_2$ is mostly bump dominated as well. Also here there is anti-cyclonic flow over the seamounts and cyclonic flow over depressions. In the corresponding $\psi_1(t = 35)$, especially for $h_0 = 100$ and $h_0 = 150$, we observe something very interesting (why we chose $t = 35$); an upper layer flow that shows evidence of the wave-like structure that are about to break up. Note the zonally generated "wiggles" on the wave resembling those in the breaking waves in LaCasce and Pedlosky [2004]. It is evident the waves remain intact for longer as h_0 and k_t, l_t increases. This suggests a slower (but still non-zero) energy transfer to the lower layer in these cases. A similar result was found in LaCasce and Brink [2000] for a sloping bottom. During the phase up to about $t = 35$, the $h_0 = 150$ response has a lower layer decoupled from the upper layer. The upper layer is wave-like while the lower layer flow is weak. In this phase, the system acts more like Rossby waves in a 1.5 layer system. We examine this more in the next section.

4.5.2 Topographic suppression of deep turbulent transfer

The $\psi_1(t = 35)$, $k_t, l_t = 30$, $h_0 \geq 60$ fields suggest the upper layer wave is somewhat stabilised and survive longer. The energy time evolutions in figure 4.14 (corresponding to the runs in figures 4.12 and 4.13) gives an additional perspective on this effect. For these parameters, the potential energy remain roughly constant quite some time before suddenly decreasing when the instability grows and the upper layer becomes turbulent. A sudden increase in KE_1 and KE_2 can be seen at the same times. KE_2 increases roughly at the same time as a direct consequence of the the turbulent upper layer; turbulent transfer of energy occurs through the non-linear terms of equations (2.24) and (2.25). As a further consequence, the total energy is roughly conserved until the instability happens before dropping due to removal of energy by friction in the recently activated lower layer. The observed instability delay seems to be related to modifying the growth rate for baroclinic instability studied in LaCasce and Pedlosky [2004]. They employed a flat bottom, but it appears reasonable that their Rossby waves would survive longer and consequently have an easier time crossing the basin if using large and many bumps in their simulations.

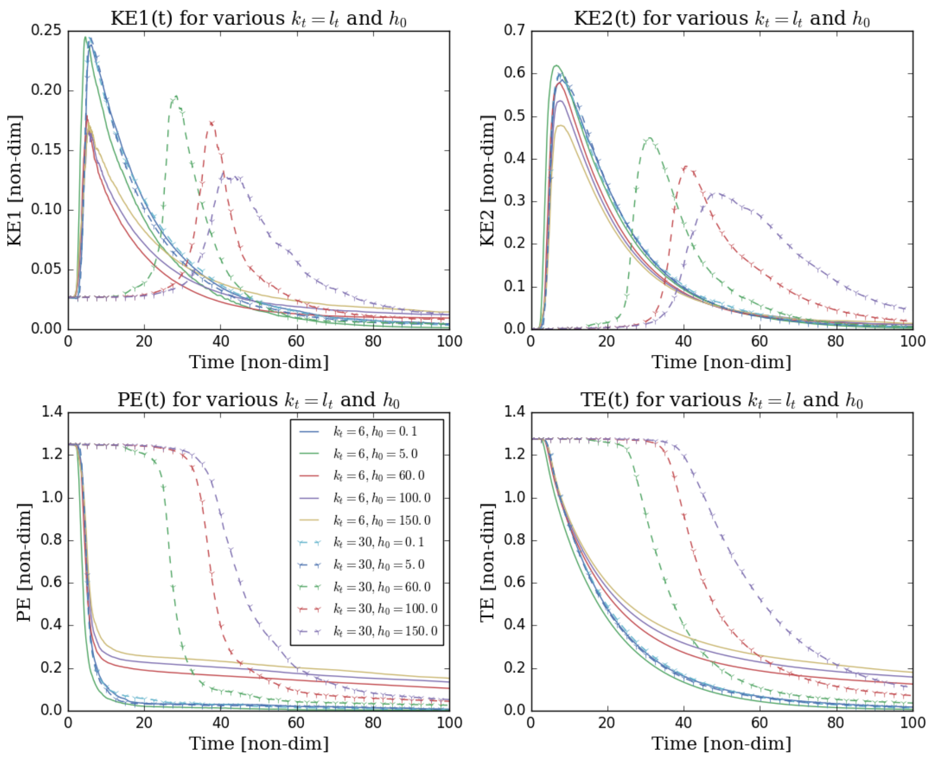


Figure 4.14: Time evolution of kinetic, potential and total energy of the non-linear system for $k_t, l_t \in \{6, 30\}$ and $h_0 \in \{0.1, 5, 60, 100\}$. The topography $h_B = h_0 \sin(k_t x) \sin(l_t y)$ and a small friction coefficient of $r = 0.04$ was used for all runs.

Note that larger h_0 (and especially $h_0 = 150$) not only delays the instability even more, but also weakens it, i.e. the extrema of KE_1 and KE_2 are smaller resulting in less overall dissipation as seen in TE. In terms of the weakening of the instability, the large and many bumps appear to exhibit a similar effect to that of friction in section 4.2; bumps and friction may both dampen the instability and limit subsequent energy loss. Very large friction, however, completely shut off the lower layer, something which we have not seen with the bumps; there is still lower layer transfer, only later and weaker. This still heavily modifies the flow field evolution though.

The $k_t, l_t = 6$ with $h_0 \geq 60$ runs also show signs of weakening (but not as much as above) the instability by observing the earlier flattening-out in PE and the corresponding smaller peak KE_1 and KE_2 compared to the small h_0 runs. However, the longer stable period before the turbulent break-up seen for $k_t, l_t = 30$, is not present here suggesting implying the flow is significantly more barotropic throughout and that the stabilising effect depends on there being enough (and tall) to decouple the lower from the upper layer.

For both $k_t, l_t = 6$ and $k_t, l_t = 30$, the smaller $h_0 = 0.1$ and $h_0 = 5$ behave quite similarly in terms of energy. This is not all that surprising considering the corresponding streamfunctions in figures 4.12 and 4.13 were quite similar to the flat bottom flow (especially for $k_t, l_t = 30$) and can also be seen in figure 4.11 along the initial fairly flat section in all panels. That being said though, we note that $k_t, l_t = 6$ and $h_0 = 5$ provides the largest peak KE_2 and the largest dissipation indicating that this is the run giving the largest transfer of energy to the lower layer. This is in line with what we have seen earlier this chapter in terms of intermediate values of h_0 enabling enhanced lower layer spin-up and also seem related to the nearly barotropic flow for these parameters.

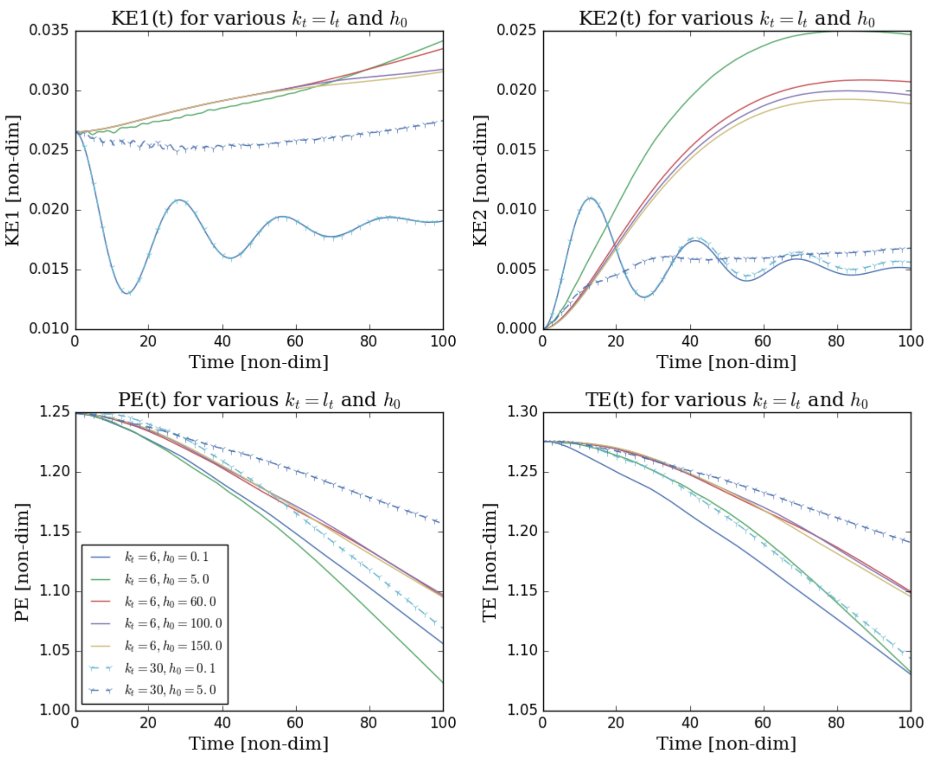


Figure 4.15: Time evolution of kinetic, potential and total energy of the linear system for $k_t, l_t \in \{6, 30\}$ and $h_0 \in \{0.1, 5, 60, 100\}$. The topography $h_B = h_0 \sin(k_t x)\sin(l_t y)$ and a small friction coefficient of $r = 0.04$ was used for all runs.

Finally, for comparison, we look at the linear analogue in figure 4.15. These energy plots are from the same runs as the upper 8 streamfunctions in figure 4.12. Immediately evident is what finished the non-linear discussion; the $k_t, l_t = 6$, $h_0 = 5$ run has the largest loss of PE and largest increase in KE_2 and thus having the most efficient transfer to the lower layer which is consistent with it being easier to force the near barotropic flow. Else, all $h_0 \geq 60$ runs all follow almost the exact same lines indicating very weak dependence on h_0. This fits well with the $\overline{\psi}_2$ fields in the upper right two panels in figure 4.12 having the same structure and magnitude. In this context it is also worth mentioning the observed lower horizontal band in figure 4.2 where we saw weak dependence on h_0 (above the smallest h_0) for small friction (we only ran for $h_0 \leq 5$ there though).

For $k_t, l_t = 30$ we only have the small h_0 runs as mentioned earlier. Recall topographic wave phase speed $c_x, c_y \propto \partial h_B/\partial x, \partial h_B/\partial y$ (see e.g. LaCasce [2018, sec. 5.7]) implying $k_t, l_t = 30$ vs. $k_t, l_t = 6$ yields a five-multiple increase in phase speeds which is more prone to violate the numerical stability condition (and so it did even for tiny time steps Δt). As already seen for the streamfunction, the $h_0 = 0.1$ behaves very similarly to the flat bottom case and the $h_0 = 5$ run has the least dissipation consistent with small magnitude in figure 4.12 as well as the upper right region of the left panel in figure 4.6 (although we stopped at $k_t, l_t = 20$ there).

Chapter 5

Summary & Conclusions

We have studied fluid flow in a two-layer ocean above a bump topography and examined the response as a function of both the horizontal and vertical scale of the bumps as well as friction. We conducted this study using numerical simulations with a two-layer quasi-geostrophic model. First, we derived the equations used by LaCasce and Pedlosky [2004]. Then we wrote an update of the model porting it over to a modern version of Fortran, but altered to include topography to facilitate use by the author for this project, but also for others who may employ the model at later times. Then we implemented the topography terms into the model and showed our model setup. Before running simulations, we developed a Python framework for model run automation and parallel execution of independent model runs, enabling the wide parameter seen in chapter 4 and saving large amounts of time when exploring parameter space. Finally, we presented and discussed the results from our numerical simulations. The following summarizes some of results that were found:

- In our method regarding the metric (4.4) for examining deep spin-up we need friction. Otherwise we would see no dissipation, comma making it hard to determine how much the lower layer was spun up over several 1000 simulations.
- Generally, the linear mean field evolves as bump flows superimposed on westward moving Rossby waves. The non-linear field is more complex, but often exhibit bump-trapped flows and Fofonoff gyres.
- With weak friction, the lower layer spins up quickly. At large friction, there is a blocking effect preventing the lower layer from spinning up. Maximum dissipation consequently occurs at intermediate values, both linearly and non-linearly, as friction continually rain the lower layer energy, facilitating even greater transfer to that layer.
- Both the linear and non-linear system exhibit (for small friction) similar dependence on h_0 as on r, i.e. moderate dissipation for small h_0, maximum dissipation for intermediate h_0 (due to closed q_s contours) and a blocking effect for large h_0.
- At larger r in the linear system, the combined effect of friction and h_0 seem to make the blocking happen more prominently and the dependence on both h_0 and r is more pronounced.

- The main contributor of linear deep transfer is when the horizontal topography scale resembles the wave x-scale, and the q_s contours are closed. This allows for the interface to be displaced coherently above the entirety of the bumps enabling optimal spin-up around the closed q_s contours. For such cases, the linear vertical structure is heavily modified with strong topographic influence.
- The non-linear fields are not significantly altered with $h_0 \leq 5$ (even with closed q_s contours) and largely approximate the flat-bottom case indicating the instability overpowers the topographic effects. The emerging mean flow is Fofonoff gyres, as is found in turbulent barotropic flows.
- If the bumps are tall enough (closed q_s contours for linear case and $h_0 \geq 60$), the flow response is nearly barotropic and thus easier to drive with surface flows, both linearly and non-linearly. For bumps smaller than the deformation radius, the bump flow is more bottom-trapped and the layers more decoupled leaving less evidence of the bumps in the upper layer.
- If the topography (both horizontally and vertically) gets large enough, it exhibits a stabilising effect on the upper layer wave field and prohibits wave instability, and thereby severely limits energy transfer to the lower layer. However, the stabilising effect only delay (albeit with quite much) the field from going turbulent. The field eventually breaks up anyway, but the peak kinetic energies are also smaller in these cases, so the overall dissipation is less.
- The topographic stabilization of the upper layer means the lower layer remains at rest for longer and the solution for these times are more like Rossby waves in the upper layer over a stationary lower layer, that is a 1.5-layer-like situation.
- Both the topographic height h_0 and wavenumber k_t, l_t are important in the resulting flow response.

The main conclusion is: The 2-D bumps show (for small friction) the ability both to enhance deep spin-up at for intermediate values of topographic height and the ability to suppress deep transfer at large topographic heights and small bump wavelength. In the non-linear system, the surface wave survives significantly longer for large h_0 and k_t, l_t; the bumps both delay and weaken the instability. But instability occurs eventually indicating the deep transfer is still present, only later and weaker.

5.1 Future work

Due to time constraints there will always be interesting aspects of a project that one does not have the time to explore, and this project is no different in that regard. In this section we offer some thoughts and discussion on areas that could be explored in more detail or extended in the future to gain further insights.

With more time, it would have been interesting to investigate the stabilising effect of the surface wave in more detail, in terms of altering growth rates for baroclinic instability, seen in LaCasce and Pedlosky [2004]. One may then be in a better position to suggest modifications to those results by the findings here.

An immediate extension is to investigate even larger topographic amplitudes (while still keeping within quasi-geostrophic assumptions) and wavenumbers. The observed non-linear upper layer stabilization from section 4.5.2 should be even stronger for such cases, maybe even throughout the entire simulation time we used. This would, however, require a finer grid resolution and even smaller time steps, both due to the stricter numerical stability condition, but also to resolve the proposed large bump count, further resulting in massive runtimes. With the numerical schemes used in this model, it is doubtful the linear runs would even be feasible given the trouble we already had with the amplitudes and wavenumbers we used.

Related to the above, it would be interesting to also include completely different topographies, i.e. other $h_B(x, y)$ than those of (4.1) and (4.2). Examples of such topographies include a sloping bottom as used in LaCasce and Brink [2000], a random field (inherently containing a wide range of wavenumbers), a Gaussian ridge in the x-direction (same as initial wave) or a single Gaussian in both x- and y-direction similar to those used in LaCasce and Nycander [2004] as a seamountain to study vortex dynamics. All these topographies have associated parameters whose impact on the two-layer system can be explored. Particularly interesting would be to look at their ability to stabilize the upper layer non-linear flow to see if this effect is a general one and not specific to the bump topography used in this work. If true, one would expect the mentioned alternative topographies to achieve a stabilising effect and a lower layer suppression (which was seen in LaCasce and Brink [2000] for the sloping bottom, but with $\beta = 0$) at large topographic heights in line with what found here.

The parameter F is also relevant for transfer to the lower layer. It is related to the density difference between the layers and determines to what degree the interface is allowed to move up and down. For instance, a smaller F implies a larger reduced gravity g' which corresponds to a larger density difference between the layers. This makes it more difficult for the interface to move up and down, and to have independent flow in the layers. The parameter thus has an impact on the lower layer spin-up, and it would be useful to study this impact by investigating different values of `F0 = 1000` (which we used throughout) in the model.

In addition to looking at different topographies, the effect of using a different upper layer initial flow could be investigated. Although we did use a different a initial condition in section 4.3.2 (and speculated on more), it was quite similar still and so appeared the response. One might ask if the lower layer spin-up would be significantly altered by very different initial conditions. Worth noting is that we performed a few sample runs with more zero-crossings in y as well (as used in Dukowicz and Greatbatch [1999]), and the response appeared similar.

Another interesting extension is the inclusion of wind forcing $\mathcal{F}(x, y, t)$ on the upper layer. All our simulations were done without any external forcing besides lower layer friction, but as seen in section 2.1.1 in equation (2.24), the upper layer can be exposed to a spatial- and time-dependent forcing from the e.g. the winds. The altered response would be of interest partly because this is closer the real oceans. Depending on the specified forcing, its strength might overpower the topographic effects seen here making the bumps less important than in our runs. As seen in the model-implemented equations (2.38) and

(2.39), the model supports this feature making study of its impact readily available.

Even though $t = 100$ as an upper simulation time appeared reasonable and captured the system after the initial lower layer spin-up phase (see e.g. linear $KE_2(t)$ flattening out in figure 4.15 as well as non-linear $KE_2(t)$ flattening out in figure 4.14 for large t), one might suggest using an even larger upper time limit to allow for more of a steady-state. Additionally, one might suggest a resolution of 512×512 for the turbulent runs to achieve higher accuracy, but wary of the significant increase in computational runtime increase.

For the sake of quantifying the upper layer stabilization seen in figure 4.14 to a larger degree, one might consider performing a set of simulations for a wide range of topographic amplitudes h_0 and wavenumbers k_t, l_t to look for the dependence of the time T_t when flow goes turbulent. To achieve this, one could for instance develop a method for approximating when upper layer wave-like flow transitions to turbulent flow, e.g. using the extrema of $KE_1(t)$, and then apply this to each simulation to construct the $T_t(h_0, k_t = l_t)$ function.

Finally, we note that the two-layer quasi-geostrophic approach used in this study, while definitely very useful for gaining insights on the dynamics, has its limitations in terms of accurately portraying the real oceans. Therefore, another possible future endeavour includes employing a primitive equation model for a similar study, that is one with continuous stratification and other features neglected by the our approach. Such a model could be used to conduct similar numerical simulations with a focus on flow response to topography and its defining parameters.

Appendix A

Additional mathematics

A.1 Layer equations to barotropic equation

In order to get a prognostic equation for ψ_B we multiply (2.24) by δ_1 and (2.25) by δ_2 and then add them together. We look at each of the terms in (2.24) and (2.25) and how they combine in turn. Starting with the time-derivative term, we have

$$\delta_1 \frac{\partial}{\partial t}\left[\nabla^2\psi_1 + F_1(\psi_2 - \psi_1)\right] + \delta_2 \frac{\partial}{\partial t}\left[\nabla^2\psi_2 + F_2(\psi_1 - \psi_2)\right] = \frac{\partial}{\partial t}\nabla^2\psi_B \tag{A.1}$$

where we have used (2.3), 2.4 and that

$$\delta_1 F_1 = \frac{H_1}{H_0}\frac{f_0^2}{g' H_1} = \frac{f_0^2}{g' H_0} = \frac{H_2}{H_0}\frac{f_0^2}{g' H_2} = \delta_2 F_2 \tag{A.2}$$

such that the second term inside the bracket-parenthesis vanish. Moving on to the first of the two non-linear jacobian terms. Using (2.6) and (2.7) these combine as

$$\begin{aligned}
&\delta_1\mathcal{J}(\psi_1, \nabla^2\psi_1) + \delta_2\mathcal{J}(\psi_1, \nabla^2\psi_1)\\
&= \delta_1\mathcal{J}(\psi_B + \delta_2\psi_T, \nabla^2(\psi_B + \delta_2\psi_T)) + \delta_2\mathcal{J}(\psi_B - \delta_1\psi_T, \nabla^2(\psi_B - \delta_1\psi_T))\\
&= \delta_1\left[\mathcal{J}(\psi_B, \nabla^2\psi_B) + \delta_2\mathcal{J}(\psi_B, \nabla^2\psi_T) + \delta_2\mathcal{J}(\psi_T, \nabla^2\psi_B) + \delta_2^2\mathcal{J}(\psi_T, \nabla^2\psi_T)\right]\\
&\quad + \delta_2\left[\mathcal{J}(\psi_B, \nabla^2\psi_B) - \delta_1\mathcal{J}(\psi_B, \nabla^2\psi_T) - \delta_1\mathcal{J}(\psi_T, \nabla^2\psi_B) + \delta_1^2\mathcal{J}(\psi_T, \nabla^2\psi_T)\right]\\
&= \mathcal{J}(\psi_B, \nabla^2\psi_B) + \delta_1\delta_2\mathcal{J}(\psi_T, \nabla^2\psi_T)
\end{aligned} \tag{A.3}$$

where we have used that $\delta_1 + \delta_2 = 1$ (see (2.5)). Next up is the second jacobian term. Here the combination vanishes as follows.

$$\begin{aligned}
\delta_1\mathcal{J}(\psi_1, F_1\psi_2) + \delta_2\mathcal{J}(\psi_2, F_2\psi_1) &= \delta_1 F_1\mathcal{J}(\psi_1, \psi_2) - \delta_2 F_2\mathcal{J}(\psi_1, \psi_2)\\
&= (\delta_1 F_1 - \delta_2 F_2)\mathcal{J}(\psi_1, \psi_2) = 0
\end{aligned} \tag{A.4}$$

where we again used (A.2) in the final equality. Moving on to the β-term, we have

$$\delta_1\beta\frac{\partial\psi_1}{\partial x} + \delta_2\beta\frac{\partial\psi_2}{\partial x} = \beta\frac{\partial}{\partial x}(\delta_1\psi_1 + \delta_2\psi_2) = \beta\frac{\partial\psi_B}{\partial x} \tag{A.5}$$

through invoking (2.3) and 2.4 again. the last term on the left hand side is the topographic term (only present in the lower layer equation). This becomes:

$$\delta_2 \mathcal{J}\left(\psi_2, \frac{f_0}{H_2} h_B\right) = \delta_2 \mathcal{J}\left(\psi_B - \delta_1 \psi_T, \frac{f_0}{H_2} h_B\right). \tag{A.6}$$

Finally we have the forcing/dissipation terms on the right hand side:

$$\delta_1 \mathcal{F} - \delta_2 r \nabla^2 \psi_2 = \delta_1 \mathcal{F} - \delta_2 r \nabla^2 (\psi_B - \delta_1 \psi_T) \tag{A.7}$$

Now that all terms are taken care of, we collect all terms (A.1), (A.3), (A.4), (A.5), (A.6) and (A.7) to get the barotropic equation

$$\begin{aligned} &\frac{\partial q_B}{\partial t} + \mathcal{J}(\psi_B, q_B) + \delta_1\delta_2 \mathcal{J}(\psi_T, q_T) + \beta \frac{\partial \psi_B}{\partial x} \\ &\quad + \delta_2 \mathcal{J}\left(\psi_B - \delta_1 \psi_T, \frac{f_0}{H_2} h_B\right) = \delta_1 \mathcal{F} - \delta_2 r \nabla^2 (\psi_B - \delta_1 \psi_T). \end{aligned} \tag{A.8}$$

where

$$q_B = \nabla^2 \psi_B \tag{A.9}$$

$$q_T = \nabla^2 \psi_T - F\psi_T \tag{A.10}$$

is the barotropic and baroclinic potential vorticity, respectively.

A.2 Layer equations to baroclinic equation

Finding a prognostic equation for ψ_T can be done by subtracting (2.25) from (2.24). We do a term-by-term procedure starting with the time-derivative term:

$$\begin{aligned} &\frac{\partial}{\partial t}\left[\nabla^2 \psi_1 + F_1(\psi_2 - \psi_1)\right] - \frac{\partial}{\partial t}\left[\nabla^2 \psi_2 + F_2(\psi_1 - \psi_2)\right] \\ &\quad = \frac{\partial}{\partial t}\left[\nabla^2(\psi_1 - \psi_2) - (F_1 + F_2)(\psi_1 - \psi_2)\right] = \frac{\partial}{\partial t}\left[\nabla^2 \psi_T - F\psi_T\right] \end{aligned} \tag{A.11}$$

where we have defined

$$F = F_1 + F_2 = \frac{f_0^2}{g' H_1} + \frac{f_0^2}{g' H_2} = \frac{f_0^2 (H_2 + H_1)}{g' H_1 H_2}. \tag{A.12}$$

The first jacobian terms combine as

$$\begin{aligned} &\mathcal{J}(\psi_1, \nabla^2 \psi_1) - \mathcal{J}(\psi_1, \nabla^2 \psi_1) \\ &= \mathcal{J}(\psi_B + \delta_2 \psi_T, \nabla^2(\psi_B + \delta_2 \psi_T)) - \mathcal{J}(\psi_B - \delta_1 \psi_T, \nabla^2(\psi_B - \delta_1 \psi_T)) \\ &= \mathcal{J}(\psi_B, \nabla^2 \psi_B) + \delta_2 \mathcal{J}(\psi_B, \nabla^2 \psi_T) + \delta_2 \mathcal{J}(\psi_T, \nabla^2 \psi_B) + \delta_2^2 \mathcal{J}(\psi_T, \nabla^2 \psi_T) \\ &\quad - \mathcal{J}(\psi_B, \nabla^2 \psi_B) + \delta_1 \mathcal{J}(\psi_B, \nabla^2 \psi_T) + \delta_1 \mathcal{J}(\psi_T, \nabla^2 \psi_B) - \delta_1^2 \mathcal{J}(\psi_T, \nabla^2 \psi_T) \\ &= \mathcal{J}(\psi_B, \nabla^2 \psi_T) + \mathcal{J}(\psi_T, \nabla^2 \psi_B) + (\delta_2 - \delta_1)\mathcal{J}(\psi_T, \nabla^2 \psi_T) \end{aligned} \tag{A.13}$$

where we have used that

$$\delta_2^2 - \delta_1^2 = (\delta_2 - \delta_1)(\delta_2 + \delta_1) = \delta_2 - \delta_1$$

and again that $\delta_1 + \delta_2 = 1$. Next up is the second jacobian term. Here the combination vanishes as follows

$$\begin{aligned} &\mathcal{J}(\psi_1, F_1\psi_2) - \mathcal{J}(\psi_2, F_2\psi_1) = F_1\mathcal{J}(\psi_1, \psi_2) + F_2\mathcal{J}(\psi_1, \psi_2) = F\mathcal{J}(\psi_1, \psi_2) \\ &\quad = F\mathcal{J}(\psi_B + \delta_2\psi_T, \psi_B - \delta_1\psi_T) = -\delta_1 F\mathcal{J}(\psi_B, \psi_T) + \delta_2 F\mathcal{J}(\psi_T, \psi_B) = \mathcal{J}(\psi_B, -F\psi_T) \end{aligned} \tag{A.14}$$

and the β-term becomes

$$\beta\frac{\partial\psi_1}{\partial x} - \beta\frac{\partial\psi_2}{\partial x} = \beta\frac{\partial}{\partial x}(\psi_1 - \psi_2) = \beta\frac{\partial\psi_T}{\partial x} \tag{A.15}$$

while the topographic term remains largely unchanged

$$-\mathcal{J}\left(\psi_2, \frac{f_0}{H_2}h_B\right) = -\mathcal{J}\left(\psi_B - \delta_1\psi_T, \frac{f_0}{H_2}h_B\right) \tag{A.16}$$

and finally the forcing/dissipation terms on the right hand side combine to

$$\mathcal{F} + r\nabla^2\psi_2 = \mathcal{F} + r\nabla^2(\psi_B - \delta_1\psi_T). \tag{A.17}$$

At last we can combine all terms (A.11), (A.13), (A.14), (A.15), (A.16) and (A.17) to get the baroclinic equation.

$$\begin{aligned} &\frac{\partial q_T}{\partial t} + \mathcal{J}(\psi_B, q_T) + \mathcal{J}(\psi_T, q_B) + (\delta_2 - \delta_1)\mathcal{J}(\psi_T, q_T) \\ &\qquad + \beta\frac{\partial\psi_T}{\partial x} - \mathcal{J}\left(\psi_B - \delta_1\psi_T, \frac{f_0}{H_2}h_B\right) = \mathcal{F} + r\nabla^2(\psi_B - \delta_1\psi_T). \end{aligned} \tag{A.18}$$

Bibliography

Abel cluster, UiO. University of Oslo. `http://www.uio.no/english/services/it/research/hpc/abel/`. Accessed: 05.05.2018.

D. L. T. Anderson and A. E. Gill. Estimating Subsurface Velocities from Surface Fields with Idealized Stratification. *Deep Sea Research and Oceanographic Abstracts*, 22(9): 583 – 596, 1975.

F. Bretherton and D. Haidvogel. Two-dimensional turbulence above topography. *Journal of Fluid Mechanics*, 78:129–154, 2004.

D. B. Chelton and M. G. Schlax. Global Observations of Oceanic Rossby Waves. *Science*, 272(5259), 1996.

M. S. de La Lama, J. H. LaCasce, and H. K. Fuhr. The vertical structure of ocean eddies. *Dynamics and Statistics of the Climate System*, 1(1):dzw001, 2016. doi: 10.1093/climsys/dzw001. URL `http://dx.doi.org/10.1093/climsys/dzw001`.

J. K. Dukowicz and R. J. Greatbatch. Evolution of mean-flow Fofonoff gyres in barotropic quasigeostrophic turbulence. *Journal of Physical Oceanography*, 29:1832–1852, 1999.

J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer-Verlag, 2nd edition, 1999.

N. P. Fofonoff. Steady flow in a frictionless homogeneous ocean. *Journal of Marine Research*, 13:254–262, 1954.

Gfortran. The GNU Fortran compiler, part of GCC. `https://gcc.gnu.org/wiki/GFortran`. Accessed: 07.05.2018.

A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introduction to Parallel Computing*. Addison-Wesley, 2nd edition, 2003.

I. Koszalka, J. LaCasce, M. Andersson, K. Orvik, and C. Mauritzen. Surface circulation in the Nordic Seas from clustered drifters. *Deep Sea Research Part I: Oceanographic Research Papers*, 58(4):468 – 485, 2011. ISSN 0967-0637. doi: https://doi.org/10.1016/j.dsr.2011.01.007. URL `http://www.sciencedirect.com/science/article/pii/S0967063711000306`.

J. H. LaCasce. A Geostrophic Vortex over a Slope. *Journal of Physical Oceanography*, 28:2362–2381, 1998.

J. H. LaCasce. Baroclinic Rossby Waves in a Square Basin. *Journal of Physical Oceanography*, 30:3161–3178, 2000.

J. H. LaCasce. On turbulence and normal modes in a basin. *Journal of Marine Research*, 60:431–460, 2002.

J. H. LaCasce. Geophysical Fluid Dynamics, 2018. *Unpublished lecture notes.* `http://folk.uio.no/josepl/papers/dynbook7.pdf`. Accessed: 31.05.2018.

J. H. LaCasce and K. H. Brink. Geostrophic Trubulence over a Slope. *Journal of Physical Oceanography*, 30:1305–1324, 2000.

J. H. LaCasce and J. Nycander. Stable and unstable vortices attached to seamounts. *Journal of Fluid Mechanics*, 507:71–94, 2004.

J. H. LaCasce and J. Pedlosky. The Instability of Rossby Basin Modes and the Oceanic Eddy Field. *Journal of Physical Oceanography*, 34:2027–2041, 2004.

J. H. LaCasce, P. E. Isachsen, and O. A. Nøst. Asymmetry of Free Circulations in Closed Ocean Gyres. *Journal of Physical Oceanography*, 38:517–526, 2008.

B. P. Leonard. A stable and accurate convection modelling procedure based on quadratic upstream interpolation. *Computer Methods in Applied Mechanics and Engineering*, 19: 59–98, 1979.

K. Mørken. Numerical Algorithms and Digital Representation, 2017. *Unpublished lecture notes.* `http://www.uio.no/studier/emner/matnat/math/MAT-INF1100/h17/kompendiet/matinf1100.pdf`. Accessed: 30.05.2018.

W. H. Munk. On the wind-driven ocean circulation. *Journal of Meteorology*, 7(2):80–93, 1950.

O. A. Nøst and P. E. Isachsen. The large-scale time-mean ocean circulation in the Nordic Seas and Arctic Ocean estimated from simplified dynamics. *Journal of Marine Research*, 61:175–210, 2003.

J. Pedlosky. *Geophysical Fluid Dynamics*. Springer-Verlag, 2nd edition, 1987.

L. P. Røed. Atmospheres and Oceans on Computers: Fundamentals, 2016. *Unpublished lecture notes.*

Stommel. The westward intensification of wind-driven ocean currents. *Eos, Transactions American Geophysical Union*, 29(2):202–206, 1948. doi: 10.1029/TR029i002p00202. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/TR029i002p00202`.

H. Stommel and A. Arons. Gulf stream transport variability in the period of decades. *Deep-sea Research*, 6(2), 1960.

G. K. Vallis. *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-scale Circulation*. Cambridge University Press, Nov. 2006. ISBN 978-1-139-45996-9. Google-Books-ID: cC0Kye7nHEEC.

P. Welander. Wind-driven circulation in one- and two-layer oceans of variable depth. *Tellus*, 20(1):1–16, 1968. doi: 10.1111/j.2153-3490.1968.tb00347.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2153-3490.1968.tb00347.x`.