UC San Diego
UC San Diego Electronic Theses and Dissertations

Title
Methods for data assimilation in chaotic systems : examples from simple geophysical models

Permalink
https://escholarship.org/uc/item/92q0b4g5

Author
Whartenby, William G.

Publication Date
2012

Peer reviewed|Thesis/dissertation
Methods for data assimilation in chaotic systems - examples from simple geophysical models

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy

in

Physics

by

William G. Whartenby

Committee in charge:

Henry D. I. Abarbanel, Chair
Bruce Cornuelle
Philip Gill
Kenneth Intrilligator
Thomas O’Neil

2012
The dissertation of William G. Whartenby is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2012
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>x</td>
</tr>
<tr>
<td>Vita</td>
<td>xi</td>
</tr>
<tr>
<td>Abstract of the Dissertation</td>
<td>xii</td>
</tr>
<tr>
<td>Chapter 1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2. Data Assimilation</td>
<td>7</td>
</tr>
<tr>
<td>2.1. Theory</td>
<td>7</td>
</tr>
<tr>
<td>2.2. SNOPT</td>
<td>9</td>
</tr>
<tr>
<td>2.3. Path Integral Method</td>
<td>11</td>
</tr>
<tr>
<td>2.4. Algorithms</td>
<td>12</td>
</tr>
<tr>
<td>2.4.1. Steepest Descent Method</td>
<td>12</td>
</tr>
<tr>
<td>2.4.2. Numerical Computation Using the Metropolis Hastings algorithm</td>
<td>14</td>
</tr>
<tr>
<td>Chapter 3. Discretization Used for Partial Differential Equations</td>
<td>18</td>
</tr>
<tr>
<td>3.1. Discretization of the Spacial Derivatives, Including Laplacian</td>
<td>19</td>
</tr>
<tr>
<td>3.2. Algorithm for Generating the Matrices for Nonuniform Spacings</td>
<td>20</td>
</tr>
<tr>
<td>3.3. Matrices for square grid</td>
<td>22</td>
</tr>
<tr>
<td>Chapter 4. Geophysical Models Used</td>
<td>28</td>
</tr>
<tr>
<td>4.1. Barotropic Vorticity Equations</td>
<td>28</td>
</tr>
<tr>
<td>4.1.1. Derivation</td>
<td>28</td>
</tr>
<tr>
<td>4.1.2. Arakawa Jacobian</td>
<td>29</td>
</tr>
<tr>
<td>4.1.3. Convert to equation in just $\zeta$</td>
<td>31</td>
</tr>
<tr>
<td>4.2. Shallow Water Equations</td>
<td>32</td>
</tr>
<tr>
<td>4.2.1. derivation</td>
<td>32</td>
</tr>
<tr>
<td>4.2.2. Ekman pumping</td>
<td>35</td>
</tr>
<tr>
<td>4.2.3. Discretization</td>
<td>37</td>
</tr>
<tr>
<td>4.2.4. 2 layer Shallow water equations</td>
<td>40</td>
</tr>
<tr>
<td>Chapter 5. Study</td>
<td>42</td>
</tr>
<tr>
<td>5.1. Objectives</td>
<td>42</td>
</tr>
<tr>
<td>5.2. Prediction</td>
<td>68</td>
</tr>
<tr>
<td>5.3. Analysis of Synchronization from an Information Theory Standpoint</td>
<td>78</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Chapter 6. Conclusion</td>
<td>80</td>
</tr>
<tr>
<td>Appendix A. Methods for Inverting Singular Matrices</td>
<td>83</td>
</tr>
<tr>
<td>References</td>
<td>89</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 4.1. One layer shallow water system ............................................. 33
Figure 4.2. Location of various quantities - velocities, heights and vorticities - on a staggered C-grid ................................................................. 37
Figure 4.3. Two layer shallow water system .............................................. 40

Figure 5.1. Velocity profile for the East - West component of velocity at location 6,8 .............................................................................................................. 44
Figure 5.2. Velocity profile for the North - South component of velocity at location 6,8 .............................................................................................................. 45
Figure 5.3. Location of measurements provided for 4 columns of data ........... 46
Figure 5.4. Location of measurements provided for 4 rows of data ............... 47
Figure 5.5. Location of measurements provided for 4 rows and 4 columns of data .............................................................................................................. 47
Figure 5.6. Value of ”presented” $A_0$ with columns of data added ................ 48
Figure 5.7. Value of ”presented” $A_0$ with rows of data added ................... 49
Figure 5.8. Value of ”presented” $A_0$ with both columns and rows of data added .................................................................................................................. 49
Figure 5.9. Value of ”full” $A_0$ with columns of data added .......................... 50
Figure 5.10. Value of ”full” $A_0$ with rows of data added ............................. 50
Figure 5.11. Value of ”full” $A_0$ with both columns and rows of data added .... 51
Figure 5.12. Estimate of east-west velocity profile for location 6,8 given 1 column of velocity data ...................................................................................... 52
Figure 5.13. Estimate of north-south velocity profile for location 6,8 given 1 column of velocity data ...................................................................................... 53
Figure 5.14. Estimate of east-west velocity profile for location 6,8 given 2 columns of velocity data ...................................................................................... 53
Figure 5.15. Estimate of north-south velocity profile for location 6,8 given 2 columns of velocity data ...................................................................................... 54
Figure 5.16. Estimate of east-west velocity profile for location 6,8 given 3 columns of velocity data ...................................................................................... 54
Figure 5.17. Estimate of north-south velocity profile for location 6,8 given 3 columns of velocity data ...................................................................................... 55
Figure 5.18. Estimate of east-west velocity profile for location 6,8 given 4 columns of velocity data ...................................................................................... 55
Figure 5.19. Estimate of north-south velocity profile for location 6,8 given 4 columns of velocity data ...................................................................................... 56
Figure 5.20. Estimate of east-west velocity profile for location 6,8 given 1 row of velocity data ...................................................................................... 56
Figure 5.21. Estimate of north-south velocity profile for location 6,8 given 1 row of velocity data ...................................................................................... 57
Figure 5.22. Estimate of east-west velocity profile for location 6,8 given 2 rows of velocity data ...................................................................................... 57
Figure 5.23. Estimate of north-south velocity profile for location 6,8 given 2 rows of velocity data ................................................................. 58
Figure 5.24. Estimate of east-west velocity profile for location 6,8 given 3 rows of velocity data ................................................................. 58
Figure 5.25. Estimate of north-south velocity profile for location 6,8 given 3 rows of velocity data ................................................................. 59
Figure 5.26. Estimate of east-west velocity profile for location 6,8 given 4 rows of velocity data ................................................................. 60
Figure 5.27. Estimate of north-south velocity profile for location 6,8 given 4 rows of velocity data ................................................................. 60
Figure 5.28. Estimate of east-west velocity profile for location 6,8 given 1 row and 1 column of velocity data ................................................. 62
Figure 5.29. Estimate of north-south velocity profile for location 6,8 given 1 row and 1 column of velocity data ................................................. 62
Figure 5.30. Estimate of east-west velocity profile for location 6,8 given 2 rows and 2 columns of velocity data ............................................. 63
Figure 5.31. Estimate of north-south velocity profile for location 6,8 given 2 rows and 2 columns of velocity data ............................................. 63
Figure 5.32. Estimate of east-west velocity profile for location 6,8 given 3 rows and 3 columns of velocity data .............................................. 64
Figure 5.33. Estimate of north-south velocity profile for location 6,8 given 3 rows and 3 columns of velocity data .............................................. 64
Figure 5.34. Estimate of east-west velocity profile for location 6,8 given 4 rows and 4 columns of velocity data .............................................. 65
Figure 5.35. Estimate of north-south velocity profile for location 6,8 given 4 rows and 4 columns of velocity data .............................................. 65
Figure 5.36. Estimate of east-west velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 of velocity data ......................... 66
Figure 5.37. Estimate of north-south velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 of velocity data ......................... 66
Figure 5.38. Estimate of east-west velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 and 2,1 of velocity data ................ 67
Figure 5.39. Estimate of north-south velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 and 2,1 of velocity data ................ 67
Figure 5.40. Prediction of east-west velocity profile for location 6,8 given 3 columns of velocity data ............................................................... 69
Figure 5.41. Prediction of north-south velocity profile for location 6,8 given 3 columns of velocity data ............................................................... 70
Figure 5.42. Prediction of east-west velocity profile for location 6,8 given 3 rows of velocity data ............................................................... 70
Figure 5.43. Prediction of north-south velocity profile for location 6,8 given 3 rows of velocity data ............................................................... 71
LIST OF TABLES

Table 5.1. Inputs to twin integration model ........................................ 43
Table 5.2. Inputs for MHMC ............................................................... 46
ACKNOWLEDGEMENTS

I would like to start by thanking my advisor, Professor Henry Abarbanel for his guidance throughout the process of getting my Ph.D. I benefitted from the advice I received, conferences I attended, and talks I presented, all of which were invaluable in my finishing this dissertation.

I also appreciate my research group - Dan Creveling, Reza Farsian, Sallee Klein, Chris Knowlton, Mark Kostuk and Bryan Toth, who provided invaluable insights into our research. Additionally, I’d like to acknowledge Paul Bryant for his expertise and assistance.

I also appreciate the emotional support I received from family and friends in this endeavor. I especially appreciate Renate Ward. Her support and kindness was invaluable during the dissertation process. Finally, I especially acknowledge Mia.
VITA

Ph.D. University of California, San Diego La Jolla, CA 92093

M.A. Mathematics Johns Hopkins University Baltimore MD 1982

B.A. Mathematics Johns Hopkins University Baltimore MD 1982

PUBLICATIONS


Data assimilation has wide ranging applications, including neuroscience, oceanography and climate science. In this dissertation we will examine data assimilation as a tool for systems of partial differential equations on a discretized spatial grid, using simple geophysical models as a twin for our study. We will use the 1 layer shallow water equations (SWE), and describe how to extend the method to a 2 layer SWE. Although we only used the SWE for this dissertation, we examine how we would use the barotropic vorticity equations (BVE) as the twin in the same study.

We will examine two different methods for performing data assimilation on chaotic systems. The first method relies on the measurements to smooth the synchronization manifold, allowing a nonlinear optimizer to correctly determine the most likely path, or the path which minimizes the cost function.

The second method we call Metropolis-Hastings Monte Carlo (MHMC) integration scheme. MHMC also allows retention of a group of path samples whose statistics
reflect the probability of each path, allowing histograms of state vector values for analysis or inputs to particle filter methods for prediction.

The study uses MHMC with the SWE as twin. in this chapter we will examine a data set used for the study. We then describe the various numbers of state vectors needed as data, and the increase in the quality of the fit. We determine the number of state vectors needed as measurements to accurately predict the unmeasured ones.
Chapter 1

Introduction

Data assimilation is the use of experimental and observational data to improve a models’ description and prediction of a system, including estimating unmeasured state vectors and parameters Evensen (2008); Kalnay (2003), or what is frequently referred to as the ”inverse problem”. We may define the problem as follows: Given a system under study, a model (which we suspect is the accurate model for the system), as well as measurements of some, but not all, parameters and state vectors of this system, we wish to determine

- the most likely state of the system
- the skill level of the assumed model for this system
- the statistics of this system for example, the moments of the system
- a histogram of possible system states, as well as give relative probabilities of outcomes, including ”extreme” low probability outcomes

These questions arise in many fields including oceanography Cornuelle et al. (2000); Derber and Rosati (1989); Lermusiaux et al. (2006); Lorenzo et al. (2007); Dobricic and Pinardi (2008); Hoteit et al. (2005) , climatology Chýlek and J. A. Coakley (1979); Carton and Geise (2008); Kondrashov et al. (2008), estuarial and costal studies Yang and Hamrick (2005) and neuroscience Huys et al. (2006) for description of full
current states. The estimates of both measured and unmeasured state vector elements are
frequently used directly for analysis of the state as well as inputs to predictive models. Various
data assimilation schemes include the adjoint method, Kalman filter Evensen (2008); Hunt et al.
(2007), nudging Auroux and Blum (2008), mean field approximations Eyink et al. (2004) and
particle filter methods Eberhart and Kennedy (1995); Kennedy and Eberhart (1995); van Leeuwen
(2009).

However, these methods have difficulty with chaotic systems. Using the adjoint
method on chaotic systems leads to an inability to predict the sensitivity of time aver-
gaged properties to changes in the parameters in the Lorenz63 model Lea et al. (2000)
or the forced-dissipative shallow water potential vorticity equation Lea et al. (2002).
The ensemble Kalman filter can have difficulty with either nonlinear systems or non-
Gaussian error Kivman (2003). All of the particle filter methods have difficulty with
chaotic systems, due to "collapse", or the reduction of the number of particles with non-
negligible weights to a number insufficient to give correct estimates and statistics, Judd
and Stemmler (2009) as well as other problems van Leeuwen (2009). Although there
has been some work in the efficiency in particle filters van Leeuwen (2010), the essential
problems listed above remain.

Additionally, these methods frequently can not completely answer the assimila-
tion problem. 4DVAR provides only the most likely path. The ensemble Kalman filter
assumes Gaussianity, as well as being a linearized approximation. Without complete
information on state vectors and parameters (in which case there is no need for data
assimilation) the above methods frequently (and generally) cannot determine necessary
information on the unmeasured ones. In addition, the methods above can not tell which
state vectors or parameters must be measured in order to determine the remaining ones
[REFERENCES HERE]. As a result, any results on the unmeasured state vectors or
parameters must be verified by other means [REFERENCES HERE].

While an exact theoretical solution to this problem exists Jazwinski (1970), a
full numerical implementation has proven difficult to implement practically Hoteit et al.
(2005), due to the "curse of dimensionality". Essentially, as the number of dimensions
increase, the number of independent iterations needed to sufficiently model the numerical solution to an integral increases geometrically (i.e. by a multiplicative factor that is the number of points per dimension necessary to cover that dimension).

Recently, a path integral monte carlo (PIMC) method has been described Restrepo (2008) which observed its computationally expensive nature. However, parallel processing has allowed us to consider many previously computational infeasible calculations. One aspect of parallel computing our lab has explored is the use of graphical processing units (GPUs) using the Common Unified Device Architecture (CUDA) API to parallelize computing tasks. GPUs contain a relatively large number of processors, each one designed to work independently with limited memory available, and very slow communication between the processors. PIMC’s computationally expensive nature is parallelizeable; the computational cost derive from considering an extremely large number of potential paths independently. By parallelizing these computations, a single CUDA card can get a speed up of 100x or more (even considering the expense incurred with data transfer from the GPU to CPU). As a result, GPU computing provides a path to reduce the time needed for completing computationally intensive (but independent) calculations.

Our lab has constucted a PIMC parallelized for GPU computation Quinn and Abarbanel (2010) which allows one to determine moments (mean, standard deviation, skewness, kurtosis, and in general $\langle x^n \rangle$), using a cost function which includes both model and measurement error. Applying this using the Metropolis Hastings (M-H) method Metropolis et al. (1953); Hastings (1970) allows for computation of histograms, or samples with the same statistics as the measured data. The path integral method has been implemented for GPU computation using CUDA in several, computationally smaller, problems, such as the Lorenz96 model and the Hodgekin-Huxley model Quinn and Abarbanel (2010).

This dissertation will focus on applying this method to partial differential equations, using the shallow water equation as our example. We illustrate how the method’s user can determine a sufficient number of measurements to accurately determine param-
eters and state vectors, and show that this number of state vectors may produce accurate estimation of the unmeasured ones, even in chaotic regimes. In this dissertation we will examine data assimilation as a tool for oceanography, using simple geophysical models as a twin for our study. We will use the 1 layer shallow water equations (SWE), and describe how to extend the method to a 2 layer SWE.

To motivate our choice for twin, we observe that the changes in the Pacific ocean due to air sea interactions and internal ocean mixing have an important effect on global climate. In addition, the unmeasured parameters and state vectors of the Pacific ocean strongly influence our forecasting and prediction Cornuelle et al. (2000). However, our measurement of the state vectors (which will show both space and time variations) and parameters (which are more slowly varying quantities approximated as constants over the space and time of the simulation) will be very sparse for the foreseeable future. As a result, using data assimilation to estimate both parameters and unmeasured state vectors is an important issue.

In Chapter 2 of the dissertation we will examine two different methods for performing data assimilation on chaotic systems. The first method relies on the measurements to smooth the synchronization manifold, allowing a nonlinear optimizer to correctly determine the most likely path, or the path which minimizes the cost function Abarbanel (2009). However, this method can not produce moments, histograms, or probabilities of different paths. As a result, we decided to use another method for the dissertation.

The second method, and the one used for this dissertation, does not rely on smoothing of the manifold, although sufficient measurements are still necessary for accurate estimates of unmeasured state vectors. This method involves computing the probability of a given path, and then integrating over the state and parameter space. The disadvantage of this method is that the integral is not, in general, soluble. As a solution to this, we will turn to a numerical integration method: the Metropolis-Hastings method for computing this integral. This numerical integration method uses a monte-carlo random ”guessing” scheme for evaluation - this method and evaluation we call
Metropolis-Hastings Monte Carlo (MHMC) integration scheme. MHMC also allows retention of a group of path samples whose statistics reflect the probability of each path, allowing histograms of state vector values for analysis or inputs to particle filter methods for prediction.

In Chapter 3 we describe the spacial discretization used in converting the partial differential equations in the previous chapter. We will start with discretizing space and defining the discrete differences we will use as finite derivatives. We then refer to results from ?? for inverting the Laplacian operator, which we will need to compute velocities from vorticities in the BVE. This chapter covers discretization concepts which we would use for any discretization of partial differential equations. Discretization issues specific to the models are covered in their respective sections.

Chapter 4 describes the geophysical models used as twins in this study. We use the barotropic vorticity equation (BVE) and the shallow water equation (SWE) as the twins for our study. In this chapter, we derive the equations from first principles and put them in the form used in the study. We finally cover model specific discretization descriptions. For the BVE we use the Arakawa JacobianArakawa (1966) as the discretization for the advective part of the equation. For the SWE we use an enstrophy conserving staggered C-grid Sadourny (1975). In this chapter we discuss the conserved quantities in each, as well as the specific discretization schemes used.

Chapter 5 is the study using MHMC with the SWE as twin. in this chapter we will examine a data set used for the study. We then describe he various numbers of state vectors needed as data, and the increase in the quality of the fit. We provide the model with all the pressure (height) data. We provide velocity (both east-west and north-south components). We provide from 1 to 4 columns (every other column) of data, 1 to 4 rows, and 1 to 4 row-column combinations. We comment on the number and location of state vectors needed as measurements to accurately predict the unmeasured ones.

Chapter 6 provides the conclusion. We discuss the results of chapter 5, and make observations on determining the quality of the fit. We then discuss the implications of increasing the resolution, which involves increasing the number of state vectors in the
system. Using a previously published study, we compare the 8 by 8 to 16 by 16 for speed vs. size tradeoffs. We then determine the aims and successes of the study, and we describe future extensions of this method.

Appendix A examines the singular value decomposition (SVD) as a tool in our study. We need to inverse the discrete matrix representation of the Laplacian in the barotropic vorticity equations. As this matrix is singular, it may not be naively inverted. However, for a matrix $A$ we may describe a pseudo inverse $B$, which may be computed using a SVD. We examine the information that is lost in the SVD, as well as the physical implications of this information loss.
Chapter 2

Data Assimilation

2.1 Theory

As a theoretical basis, we view data assimilation as information transferred from measurements to the dynamical equations of a model. In this chapter, given a set of measurements and a model, we attempt to determine

- state vectors most likely values
- state vectors mean values (may be different from most likely)
- moments of various quantities
- histograms of possible stat vector values

Given a model and data, we may specify a model error as the difference between the state vector values and the discrete map (a discretized version of the ODEs or PDEs describing the physics), as well as a measurement error (relating the difference between the model and observations).

Take a discretized Markov process, with dimension D, and \( M < D \) of these dimensions have measurements. Let the state vector be represented by a D dimensional vector \( \mathbf{x} \), and let the data be represented by a M dimensional vector \( \mathbf{y} \). Also, let the
time series of \(\vec{y}\) be defined as \(\vec{y}_m\) is the value of the state vector at time m, while \(\vec{Y}_m\) is the state vector \(\vec{y}\) for time step 1 to m.

We may write the probability of a state \(\vec{x}_m\) given a set of measurements \(\vec{Y}_m = \vec{y}_m \cup \vec{Y}_{m-1}\) as \(P(\vec{x}_m \mid \vec{y}_m) = P(\vec{x}_m \mid \vec{y}_m, \vec{Y}_{m-1})\) using Bayes’ law for the last equality. We may therefore write this probability as Abarbanel (2009)

\[
P(\vec{x}_m \mid \vec{Y}_{m-1}) = \frac{P(\vec{x}_m, \vec{y}_m \mid \vec{Y}_{m-1})}{P(\vec{y}_m \mid \vec{Y}_{m-1})} P(\vec{y}_m \mid \vec{Y}_{m-1})
\]

The term in brackets is the exponential of the conditional mutual information.

The Chapman-Kolmogorov equation states

\[
P(\vec{x}_m \mid \vec{Y}_{m-1}) = \int P(\vec{x}_m \mid \vec{x}_{m-1}) P(\vec{x}_{m-1} \mid \vec{Y}_{m-1}) d\vec{x}_{m-1}
\]

We then make 2 observations. First, we observe that \(P(\vec{x}_m \mid \vec{x}_{m-1})\) is the probability of measuring the new state, given the previous one. This is the model error, or the imprecision of the single step integration used in the model (vis a vis the actual state). as a result we may write \(P(\vec{x}_m \mid \vec{x}_{m-1})\) as \(P(\vec{x}_m \mid f(\vec{x}_{m-1}, \vec{p}))\) where \(f\) describes the model dynamics.

The second is that, since we have written \(P(\vec{x}_m \mid \vec{Y}_{m-1})\) in terms of \(P(\vec{x}_{m-1} \mid \vec{Y}_{m-1})\) we may iterate backwards to compute the probability of a given path to be \(e^{-A_0(\vec{y}, \vec{x}, \vec{p})}\), where

\[
A_0(\vec{x}, \vec{y}, \vec{p}) = \sum_{n=0}^{T} \log \frac{P(\vec{x}_n, \vec{y}_n \mid \vec{Y}_{n-1})}{P(\vec{y}_n \mid \vec{Y}_{n-1})} - \sum_{n=1}^{T} \log \{P(\vec{x}_n \mid f(\vec{x}_{n-1}, \vec{p}))\}
\]

The first term is the total mutual information. This term represents the information given the model by the measurements Abarbanel (2009). The second term represents the error resulting from computing the state vectors using our chosen model, or model error.
For example, consider the case that the model and measurement errors each has an independent Gaussian distribution, with the standard deviations for model and data accuracy \((2R_f)^{-\frac{1}{2}}\) and \((2R_m)^{-\frac{1}{2}}\), respectively. As we assume the errors are independent, we may drop all conditioning on \(Z_{n-1}\), or the previous measurements. We also observe that for the Gaussian distribution \(P(\vec{x}_m, \vec{y}_m) = e^{-R_m|\vec{x}_m - \vec{y}_m|^2}\) and a similar result applies for the model error - \(P(y_n|y_{n-1}) = e^{-R_f|\vec{x}_n - \vec{f}(\vec{x}_{n-1})|^2}\)

as a result, may give the action as

\[
A_0(\vec{y}, \vec{x}, \vec{p}) = R_m \sum_{n=0}^{T} \sum_{l=0}^{M} (y_n^{(l)} - z_n^{(l)})^2 + R_f \sum_{n=0}^{T-1} \sum_{l=0}^{D} (y_n^{(l)} - f^{(l)}(\vec{x}_n, \vec{p}))^2
\]

Phrased in this form, we may see that

- The path integral method derives the result that minimization of a least squares cost function (4D-VAR) gives the most probable path, given measurements and model errors which satisfy an independent Gaussian distribution. This allows us to show that 4D-VAR is the most probable result with independent Gaussian errors on all quantities contributing to the cost function.

- Given that we may express the probability of the state as \(e^{-A_0}\), we may then determine moments by calculating high dimensional integrals:

\[
< y^n > = \int \ldots \int d\vec{x} y^n e^{-A_0(\vec{x}, \vec{y}, \vec{p})}
\]

### 2.2 SNOPT

One method we pursued involved using nonlinear optimization Gill et al. (1998). Our lab had previously used this method for other problems. Nonlinear (and in general) optimizers minimize a cost function subject to constraints. Recall that optimization has
difficulty with nonlinear (and especially chaotic) systems, due to the irregular nature of
the cost function, even in the neighborhood of the "true" solution (using the idea of
"true" for our twin experiments). However, we argued Abarbanel (2009) that adding
additional measurements had the effect of smoothing the synchronization manifold, or
cost function (at least in the neighborhood of the true solution), allowing the optimizers
to find the "correct" solution.

We defined the function to be minimized to be the cost function: for a D dimen-
sional system of equations with \( M < D \) observations, where \( \dot{x}^{(l)} = f^{(l)}(\vec{x}) \), \( \vec{x}_n \) is the
value at the \( n \)th of N time steps then the standard 4DVAR cost function can be written
as

\[
\sum_{i=0}^{N-1} \sum_{j=0}^{D-1} \left( x_{i+1}^{(j)} - x_i^{(j)} - \Delta T * f^{(j)}(\vec{x}) \right)^2 + \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left( x_i^{(j)} - y_i^{(j)} \right)^2
\]

Our lab applied IPOPT Wächter and Biegler (2006) mostly for this work, although we also used SNOPT Gill et al. (1998) for nonlinear optimization. Both SNOPT
and IPOPT require significant coding of calling programs to use. We were able to ac-
cquire software to automate the procedure for SNOPT Gill et al. (2002) as well as IPOPT
Toth (2010). The IPOPT preprocessor is written in Python.

I was successful in using SNOPT on the 4 by 4 BVEs for parameter estimation. However, after that we ran into several stumbling blocks, and moved to IPOPT. We
made several runs using the BVEs. However, we ran into problems as we increased the
grid size.

One of the metrics we used is scalability. Since oceanographic and climate mod-
els frequently employ large dimensional systems, we naturally want to ask the question
of how this solution scales. Our answer is ”not well with the current tools” for the BVEs.
For example, the IPOPT preprocessor will run in a fairly short time (less than an hour)
for the 4 by 4 BVEs, yet over a week for the 8 by 8 case. This increase is due to the fact
that IPOPT requires not only the vector field, but also the Jacobian and Hessian fields
as well. We could improve our IPOPT by allowing the script to require code for the
Jacobian and Hessian calculation be provided, as this is (except for a very little bit) all
of the time used by the preprocessor. This was one of the problems that led us to use SWE instead of BVE for this study.

However, once we complete the preprocessor and have the code to run IPOPT, we still have scalability issues. The IPOPT run shows a similar time increase given the increase of dimensionality as SNOPT. My understanding of the similarity is qualitative; I have not attempted to quantify the time scaling, only noting that the time needed increases far too quickly for these approaches to be currently applicable. I have identified the following tasks needed to allow SNOPT or IPOPT to be used

- Preprocessor - Modify IPOPT Python script so that the preprocessor does not attempt to produce code for the vector field, Jacobian or Hessian. Produce a speed optimized code for these by hand

- Parallel processing - Ensure that both optimizers are using fully parallelized libraries and are optimized for parallel processing

- Review code produced by preprocessors in order to ensure maximum efficiency

Our lab has been successful in applying optimization methods to data assimilation in neuroscience [REFERENCES HERE], however, I was not able to effectively use the tools for this study. We were able to adapt a path integral approach. This will be the topic of the next section.

2.3 Path Integral Method

As a result of the difficulties using nonlinear optimizers, I investigated using a path integral method (PIM) for data assimilation. PIM has advantages

- PIM does not require Jacobians or Hessians computed

- PIM itself (not just the linear algebra section) may easily be parallelized
• We have access to a coded version of PIM in our lab, making the programming for it no more extensive than coding for the nonliner optimizers

We first derive the algorithm of the path integral method. Secondly we discuss the implementation of the theory. We use the Metropolis Hastings algorithm in our evaluation of the path integral. In addition to the most likely state vector, the Metropolis Hastings algorithm can additionally produce histograms, or representative samples of state vectors for ensemble methods.

2.4 Algorithms

Given that the probability of a path given measurements may be written as $e^{-A_0(\vec{X},\vec{Y})}$, then the expectation of a value may be written as

$$E(B | \vec{Y}) = \frac{\int B e^{-A_0(\vec{X}|\vec{Y})}d\vec{X}}{\int e^{-A_0(\vec{X}|\vec{Y})}d\vec{X}},$$

where $V$ represents the state vector space. This integral is, in general, intractable. We may employ 2 possible methods for solving this integral.

2.4.1 Steepest Descent Method

Use the steepest descent approximation. We need to first find a minimum of $A_0$, which may be determined by 4DVAR. We also need the second derivative of $A_0$ with respect to each dimension of the state vector space at the minimum. This can be computed numerically. We first compute the normalization factor $G$ in this case: $G = \int_V e^{-A_0(\vec{X}_0|\vec{Y})} + A''_0(\vec{X}_0|\vec{Y}) \left| \vec{X} - \vec{X}_0 \right|^2 d\vec{X}$ so

$$E(B | \vec{Y}) \approx \frac{1}{G} e^{-A_0(\vec{X}_0|\vec{Y})} \int B e^{-A''_0(\vec{X}_0|\vec{Y})} \left| \vec{X} - \vec{X}_0 \right|^2 d\vec{X},$$

which is frequently much more tractable.

As an example of this method, let us compute the value of $<x>$. In this case the
B is equal to x. We first compute the value of $G$, the normalization constant.

$$G = \int_V e^{-A_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} + A''_0(\vec{\bar{X}}_0|\vec{\bar{Y}}) |\vec{\bar{X}} - \vec{\bar{X}}_0|^2 d\vec{\bar{X}}$$

$$= e^{-A_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} \int_V e^{A''_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} |\vec{\bar{X}} - \vec{\bar{X}}_0|^2 d\vec{\bar{X}}$$

$$= e^{-A_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} \frac{\pi}{\sqrt{A''_0(\vec{\bar{X}}_0|\vec{\bar{Y}})}}$$

Next, let’s compute $<\vec{\bar{X}}> = \frac{1}{G} \int_V \vec{\bar{X}} e^{-A_0(\vec{\bar{X}}|\vec{\bar{Y}})} d\vec{\bar{X}}$

$$<\vec{\bar{X}}> = \frac{1}{G} \int_V \vec{\bar{X}} e^{-A_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} + A''_0(\vec{\bar{X}}_0|\vec{\bar{Y}}) |\vec{\bar{X}} - \vec{\bar{X}}_0|^2 d\vec{\bar{X}}$$

$$= e^{-A_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} \int_V e^{A''_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} |\vec{\bar{X}} - \vec{\bar{X}}_0|^2 d\vec{\bar{X}}$$

$$= \frac{1}{G} e^{-A_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} \left( \int_U e^{A''_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} |\vec{U}|^2 d\vec{U} + \int_V \vec{\bar{X}}_0 e^{A''_0(\vec{\bar{X}}_0|\vec{\bar{Y}})} |\vec{U}|^2 d\vec{U} \right)$$

$$<\vec{\bar{X}}> = \vec{\bar{X}}_0$$

where in the last 2 steps we made the substitution $\vec{\bar{U}} = \vec{\bar{X}} - \vec{\bar{X}}_0$ and noted that the integral over all space $\int_V e^{-A''_0|\vec{U}|^2} d\vec{U} = 0$.

We can see from this, (and from the structure of the steepest descent), that the steepest descent method translates all problems into Gaussian statistics. It can not, therefore, correctly identify cases where the most likely is different from the mean, or cases where the statistics are not inherently Gaussian. Additionally, this method places restrictions on the structure of $A_0$ in order to get accurate answers. Due to these and other problems, we chose a numerical evaluation of the integral. This method is described in the next section.
2.4.2 Numerical Computation Using the Metropolis Hastings algorithm

The M-H algorithm is an algorithm designed to efficiently evaluate high dimensional integrals by excluding paths with small probabilities. We may use M-H to produce moments of the state vectors given a model, data, the distribution (Gaussian, Lorentzian, etc.), and estimates of the accuracies of each (for example, using a Gaussian distribution, the metric would be the standard deviation). We use this algorithm to get predicted moments, and compare with the known distribution presented to the model.

An additional benefit to the M-H algorithm is that it may also produce a histogram, or collection of points which satisfy the distribution defined by $A_0$ also describe a distribution of states for this system, by weighting a uniform random group of paths by each one’s probability, or $e^{-A_0}$. This distribution may be used as would a distribution acquired from ensemble Kalman filter or particle swarm methods.

Procedure

We define a path as a value (guessed or otherwise) for each state vector element and parameter. The M-H algorithm consists of the following steps:

- Initialize the path. This may be done by random guessing, although very poor or nonphysical initial guesses will dramatically slow convergence rates.
- Compute the action ($A_0$) of the path. This is the first ”stored” $A_0$
- Propose a new path and compute its $A_0$
- Compare the new $A_0$ with the stored value
  - If the new value is lower than the stored value, accept the new value
  - else
    * compute the difference between the new value and the stored value.
      Call this (positive) difference $\Delta$
• select a random number between 0 and 1.
• if the value $e^{-\Delta}$ is greater than the random number accept the new path
• else retain the stored path

After a warm up period, we then retain each path that comes out. This accretion of paths may be used to determine statistics or contribute ensembles as inputs for particle swarm or ensemble Kalman filter methods.

While this method will always generate ensembles with the same statistics as that of the underlying ”real” system, this method can take a long time to converge. Additionally, if this method finds a solution that is in a local minimum, the solution reflecting this local minimum can incorrectly dominate the reported paths and statistics for a long time. In this case, we use a simulated annealing approach to increase the ability of the M-H procedure to converge on the global solution.

**Simulated Annealing**

Simulated annealing is based on the metallurgical concept of annealing, or the repeated heating and slow cooling of a metal in order to reduce defects. A simulation of this process (simulated annealing) has been incorporated in a large class of optimization and fitting algorithms van Laarhoven and Aarts (1987). For our method, simulated annealing involves multiplying the model error’s contribution to the cost function by a factor that is between 0 and 1 (lower value representing a higher temperature, as this is analagous to the term $E/kT$ in statistical mechanics), and increasing this factor to 1 over the course of the number of ”warm-up” iterations. We compute this factor (denoted as $\beta$, in analogy with the term $1/kT$ in statistical mechanics) in the following way for our study:

• Choose an initial $\beta$; define this as $\beta_0$

• Choose a number of ”simulated annealing” iterations. This number should be less than the number of initialization iterations, and the simulated annealing will
occur during initialization

- set $\beta$ to $\beta_0$

- Determine the multiplicative factor for $\beta$: if the number of simulated annealing iterations is $N$, defining $\beta_N = \frac{1}{N}$ and then define the factor $F$ as $\left(\frac{1}{\beta_0}\right)^{\beta_N}$. Note that $\beta_0 F^N = 1$

- for each iteration in the warmup sequence
  - compute the model error’s contribution to the action
  - multiply this contribution by $\beta$ ($\beta < 1$)
  - Compute the data’s contribution to the action, if there is data for this location
  - Sum the 2 contributions to get the action. Use this action as described above to accept/reject the proposed path
  - multiply $\beta$ by $F$

- After $\beta$ becomes 1, complete the rest of the warm up iterations

- After the warm up, complete the iterations, collect samples, statistics, etc.

In this way, the contribution to the cost function due to the mismatch of the model with the presented path is lowered - much as in annealing, where the high temperatures allow the atoms to become mobile, even when in an (energetic) deep local minimum.

**Number of Iterations Needed**

The number of statistics iterations need to be as many as possible. This is where the actual guesses representing the accepted values are accreted. With problems with well-defined and deep local minima, a large number of iterations may be needed to fully represent the space.
The meta-data for the runs used in the study are specified in the study section in 5.1. We found that we needed to make 4.8 million iterations for our data. While there is some indication that fewer iterations are needed, we also have indications that the integral is still improving accuracy. However, we note that our measurement of the full error show that all of the cases have "effectively" converged (except the 4 row and 4 column case is still converging, but its estimates are already extremely accurate).
Chapter 3

Discretization Used for Partial Differential Equations

In this chapter we convert a system of partial differential equations into ordinary differential equations by discretizing the spatial derivatives, keeping only the time derivative in the vector field calculations. In addition, we discuss nonuniform grids (for example, at a given time, choosing a much finer grid near shock fronts and other areas with rapidly changing quantities). Derivatives become finite differences and, as such, may be represented by matrices operating on the state vectors. We look at the first derivatives, and extend this to the $n$th derivatives. We consider a special case of spatial derivatives - the Laplacian function. In the BVE there are 2 quantities of interest $\psi$ and $\zeta$, which are related by the Laplacian: $\zeta = \nabla^2 \psi$. We will need to compute $\psi$ from $\zeta = \frac{1}{\nabla^2} \psi$, or invert the Laplacian. In representing the Laplacian as a matrix, we may construct the inverse Laplacian by inverting a matrix. As the Laplacian matrix is singular, it may not be directly inverted. We investigate using the singular value decomposition (SVD) to invert the matrix, and the implications of its use. Finally, we investigate using a Green’s function approach which allows us to compute the inverse of the Laplacian without explicitly inverting the matrix.
3.1 Discretization of the Special Derivatives, Including Laplacian

We use the standard discretizations of the derivative. Using the basic definition
\[
\frac{\partial f(x,y)}{\partial x} = \frac{f(x+\Delta x,y) - f(x,y)}{\Delta x},
\]
applied to a spatial grid. \(x_i, 0 \leq i \leq N - 1\), whose grid spacing is \(\Delta\), whose discrete location is \((m,n)\) we may define 2 types of derivative
\[
\Delta_+ = \frac{f(m+1,n) - f(m,n)}{\Delta x} \quad \text{'forward derivative'}
\]
\[
\Delta_- = \frac{f(m,n) - f(m-1,n)}{\Delta x} \quad \text{'backward derivative'}
\]

In this scheme, we can define the derivative we will use (the "centered" derivative) as being \(\frac{\Delta_+ + \Delta_-}{2}\) and the 2\textsuperscript{nd} derivative (the x component Laplacian operator) as being \(\frac{\Delta_+ - \Delta_-}{\Delta x}\).

Finally, we note that the first derivative is a linear average of 2 different discrete representations of the derivative for each point, there is only 1 definition of the second derivative (as the change in first derivatives, or the difference between the two derivatives, divided by grid spacing (the change of the change)).

We will use the previous paragraph to derive an expression for the higher order derivatives (although we will not use any in the study, dissipation effects like \(\nabla^2 n, n/gt\) are commonly used to represent gentler diffusion processes).

The first will be to look at the 3\textsuperscript{rd} order derivative. We may define this discretization similarly to the 1\textsuperscript{st} derivative. Since the second derivative is defined uniquely for each point (unlike the first derivative, which has a "forward" or backward " (or linear combinations of these 2) interpretation), we may define the change in the second derivative similarly to the 1\textsuperscript{st} derivative - as a "forward" and "backward" interpretation,
\[
\nabla^3 = \frac{\left(\nabla^2_+ - \nabla^2_\cdot\right)}{\Delta x}
\]

We can also see (or determine by a quick calculation) a few properties which will be useful

1. the 1\textsuperscript{st} derivative is not uniquely defined, but has 2 separate representations: we use the 50-50 linear combination of the 2
2. the 2\textsuperscript{nd} derivative is uniquely defined as a separate linear combination of the two
1\textsuperscript{st} derivative.

3. the 1\textsuperscript{st} and 2\textsuperscript{nd} derivatives commute with each other

4. Using the interpretation of the 3\textsuperscript{rd} derivative as the "forward" 3\textsuperscript{rd} derivative minus the "backward" 3\textsuperscript{rd} derivative, the 3\textsuperscript{rd} derivative is the 1\textsuperscript{st} derivative times the 2\textsuperscript{nd} derivative (or the other way around, as they commute)

5. we may therefore write any power of the derivative $\nabla^m$
   
   - for an even power of $\nabla$, we may simply compute $\frac{m}{2}$ powers of the Laplace matrix or $\nabla^2$
   
   - for an odd power of $\nabla$ we may simply compute the even power 1 less than this power, then multiply the result by the 1\textsuperscript{st} derivative matrix.

3.2 Algorithm for Generating the Matrices for Nonuniform Spacings

In this section we describe the methods for generating the grids for the first and second derivatives, as well as the Laplacian. We list the computer code needed to write these matrices, as well as the matrices themselves, in Appendix...

We recall from the previous section that the discrete first derivative is not uniquely defined, but has a forward and backward derivative (as well as linear combinations of the two). Here we describe the algorithm for computing them.

Algorithm for backward derivative - we remember that for some state vector $\vec{w}$ represented by the backwards derivative may be written as matrix multiplying the column vector $\vec{w}$ where the matrix is constructed as follows

- Loop over each state vector dimension i - each i is a row

- within this row, loop over all state vector dimension j
• if this j is "the previous in x to i" (we use periodic boundary conditions), then the vector element is $-\frac{1}{\Delta x_{j}}$.

• if this j is i (we use periodic boundary conditions), then the vector element is $\frac{1}{\Delta x_{i}}$.

• else this matrix element is 0

We may then write a similar algorithm for the forward derivative

• Loop over each state vector dimension i - each i is a row

• within this row, loop over all state vector dimension j

• if this j is "the next in x to i" (we use periodic boundary conditions), then the vector element is $\frac{1}{\Delta x_{j}}$.

• if this j is i, then the vector element is $-\frac{1}{\Delta x_{i}}$.

• else this matrix element is 0

And the "centered derivative" we use is a linear combination of the 2 derivatives above, $\frac{1}{2}$ of each.

Next we consider the second derivative, or $\frac{\partial^{2}}{\partial x^{2}}$. This is again the linear combination of the 2 basic first derivatives above. By definition, we interpret the second derivative as the difference between the two above, or forward - backward, divided by the spacing between them. Once we have generated the forward and backward derivative matrices [f] and [b] respectively, we may compute the second derivative. For example, the second x derivative will be $\left([f] \Delta_{j}^{T} - [b] \Delta_{i}^{T}\right)$. The 1 by N row vector $[\Delta]$ is a row vector whose elements are formed by the linear combination of spacings: for example, the first entry in $[\Delta]$
3.3 Matrices for square grid

In this section, we use the standard discrete approximations where each grid point’s separation from the next (in either the x or y directions) is $\Delta$. Consider an $N \times N$ grid. My mapping of x-y coordinate pair to ODE number is $n = i + N*(j-1)$. In addition, using doubly periodic boundary conditions (ip1 = i + 1, if ip1 > N then ip1 = 1, im1 = i -1 if im1 = 0 then im1 = N) the matrix representing the "centered" $\frac{\partial}{\partial x}$ is

$$\begin{bmatrix}
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2\Delta x} & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 \\
\end{bmatrix}$$
and the matrix representing \( \frac{\partial}{\partial y} \) is

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
\end{bmatrix}
\]
We now calculate the second derivatives separately. The matrix representing \( \frac{\partial^2}{\partial x^2} \) is

\[
\frac{1}{\Delta^2_x} \begin{bmatrix}
-2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2
\end{bmatrix}
\]
and the matrix representing $\frac{\partial^2}{\partial y^2}$ is

$$\begin{pmatrix}
-2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 \\
\end{pmatrix}$$
As a result we may write the Laplace matrix as the sum of the 2 second derivatives, or

\[
\begin{bmatrix}
-2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
-2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
In the case $\Delta_x = \Delta_y \equiv \Delta$ the Laplace matrix is

$$
\begin{bmatrix}
-4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & -4 \\
\end{bmatrix}
$$

As the rows and columns add up to 0, we can see that this matrix is singular. This implies that the solution will require a pseudo-inverse of the Laplacian. We will examine the different ways of doing that and discuss the advantages of the different methods in Appendix A. For now, we will take the result from Appendix A, that we can under certain assumptions, validly use the pseudo-inverse as the matrix inverse.
Chapter 4

Geophysical Models Used

4.1 Barotropic Vorticity Equations

We consider the BVE as the model for examining the application of the path integral method to PDEs. We begin by deriving the BVE from the Navier Stokes equation.

4.1.1 Derivation

We first assume the fluid is incompressible, and we start with the 3 dimensional Navier Stokes momentum balance equation for incompressible fluids ($\nabla \cdot \mathbf{v} = 0$)

$$
\rho \left( \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right) = -\nabla P + \mathbf{F}
$$

Where $\mathbf{F}$ includes all body forces. These include the coriolis force ($-\mathbf{f} \times \mathbf{v}$), gravity ($\rho g$) viscosity ($\nu \nabla^2 \mathbf{v}$) bottom drag ($\epsilon \mathbf{v}$) and any external body forces. we may then remove gravitational force and the $\hat{z}$ component of $\nabla P$, as these components cancel out. Then, if there are no more explicit forces in the vertical direction, we may take all motion to be in the horizontal plane.

Next, we apply $\nabla \times$ to both sides of the equation, and define the relative vorticity $\zeta$ as $\left\{ \left( \nabla \times \mathbf{v} \right) \cdot \hat{z} \right\} \hat{z}$. We may then use the vector identity

$$
\nabla (\mathbf{A} \cdot \mathbf{B}) = (\mathbf{A} \cdot \nabla) \mathbf{B} + (\mathbf{B} \cdot \nabla) \mathbf{A} + \mathbf{A} \times (\nabla \times \mathbf{B}) + \mathbf{B} \times (\nabla \times \mathbf{A}) \quad (4.1)
$$
letting $\overrightarrow{A} = \overrightarrow{B} = \nabla$

\[
\nabla (\nabla \cdot \nabla) = (\nabla \cdot \nabla) \nabla + (\nabla \cdot \nabla) \nabla + \nabla \times (\nabla \times \nabla) + \nabla \times (\nabla \times \nabla)
\]

\[
(\nabla \cdot \nabla) \nabla = \frac{1}{2} \nabla (\nabla \cdot \nabla) - \nabla \times (\nabla \times \nabla)
\]

\[
(\nabla \cdot \nabla) \cdot \nabla = \frac{1}{2} \nabla (\nabla \cdot \nabla) - \nabla \times \zeta
\]

\[
\Rightarrow \nabla \times (\nabla \cdot \nabla) \nabla = \nabla \times \left( \nabla \times \zeta \right)
\]

using the vector relation

\[
\nabla \times (\overrightarrow{A} \times \overrightarrow{B}) = \overrightarrow{A} \left( \nabla \cdot \overrightarrow{B} \right) - \overrightarrow{B} \left( \nabla \cdot \overrightarrow{A} \right) + \left( \overrightarrow{B} \cdot \nabla \right) \overrightarrow{A} - \left( \overrightarrow{A} \cdot \nabla \right) \overrightarrow{B}
\]

and recalling that $\zeta$ is in the $\hat{z}$ direction, we arrive at the relation

\[
\nabla \times \left( \nabla \cdot \nabla \right) \nabla = \left( \nabla \cdot \nabla \right) \zeta. \quad \text{Therefore, the vorticity } \zeta = |\zeta| \hat{z} \text{ is materially conserved - that is, that } \frac{\partial \zeta}{\partial t} = 0. \quad \text{Using the relationships between } x \text{ and } y \text{ velocities and derivatives of the stream function allow us to also write this equation as } \frac{\partial \zeta}{\partial t} + \frac{\partial \psi}{\partial x} \frac{\partial \zeta}{\partial y} - \frac{\partial \psi}{\partial y} \frac{\partial \zeta}{\partial x} = 0.
\]

Finally, if we explicitly include the coriolis force, we see that the term in the original momentum conservation is $f \hat{z} \times \nabla$. Taking the curl of this results in the term $\nabla \cdot \nabla f$. The potential vorticity $\eta = \zeta + f$ also satisfies the conservation of material derivative $\frac{\partial \eta}{\partial t} = 0$.

### 4.1.2 Arakawa Jacobian

Equation for the vorticity can be discretized using a standard discretization scheme. However, this leads to numerical instability for long term integration. This phenomenon is due to the "cascading" of kinetic energy to shorter and shorter length scales. As the length scales eventually reach the discretization length, the vorticity "noodles", or collects into long stringlike structures. Even though some of this is physical, the unphysical conclusion is due to the result that enstrophy (the integral, or spacial sum of the square of the vorticity) is not conserved with the naive discretization. This problem was rec-
ognized and solved by Arakawa. The solution involves writing the discretization of the spacial derivatives as combinations of possible discrete interpretations: for example, the derivative could be written as a linear combination of forward derivative and backward derivative. Phrasing the discretization with coefficients (describing the linear combination) and setting the coefficients such that the mean vorticity, mean enstrophy and mean kinetic energy are conserved. The resulting representation for the Jacobian is typically referred to as the Arakawa Jacobian. The next section states the method to compute the Jacobian.

We will be expressing the values for the vorticity $\zeta$, which is really the value of vorticity at each point on a two dimensional grid, as a single index vector. As a result, I will use the notation $\zeta_{n(i,j)}$ to represent an element in a column vector, similar to the original notation in the Arakawa paper. The function $n(i,j)$ converts the i,j location on a grid to a single number; $n(i,j) = \text{dimensionX}\ast j + i$. In addition, we have doubly periodic boundary conditions, so boundary terms are $i = \text{dimensionX}$; $i+1 = 1$; $i = 1$; $i-1 = \text{dimensionX}$, etc.

In this context, the Arakawa Jacobian may (after a bit of algebra) be written as

$$\dot{\zeta}_{n(i,j)} = \frac{1}{12 * \Delta^2} \cdot$$

$$\left( \psi_{n(i+1,j+1)} + \psi_{n(i,j+1)} - \psi_{n(i,j-1)} - \psi_{n(i+1,j-1)} \right) \ast \zeta_{n(i+1,j)} +$$

$$\left( \psi_{n(i,j+1)} - \psi_{n(i+1,j)} \right) \ast \zeta_{n(i+1,j+1)} +$$

$$\left( \psi_{n(i-1,j+1)} + \psi_{n(i-1,j)} - \psi_{n(i,j+1)} - \psi_{n(i,j+1)} \right) \ast \zeta_{n(i,j+1)} +$$

$$\left( \psi_{n(i-1,j+1)} - \psi_{n(i,j+1)} \right) \ast \zeta_{n(i-1,j+1)} +$$

$$\left( \psi_{n(i-1,j-1)} + \psi_{n(i,j-1)} - \psi_{n(i,j-1)} - \psi_{n(i-1,j-1)} \right) \ast \zeta_{n(i-1,j)} +$$

$$\left( \psi_{n(i,j-1)} - \psi_{n(i-1,j)} \right) \ast \zeta_{n(i-1,j-1)} +$$

$$\left( \psi_{n(i+1,j-1)} + \psi_{n(i+1,j-1)} - \psi_{n(i-1,j)} - \psi_{n(i-1,j-1)} \right) \ast \zeta_{n(i+1,j-1)} +$$

$$\left( \psi_{n(i,j-1)} - \psi_{n(i-1,j)} \right) \ast \zeta_{n(i+1,j-1)}$$

as a result there is a matrix for each point on the grid; each matrix $M(n(i,j), n'(i',j'))$ has a 0 value for each entry except for the entries corresponding to a $\psi \zeta$ combination. To numerically speed up this computation the user may create this as a sparse matrix, where the first index would be the ODE number, while the second would be the state vector.
4.1.3 Convert to equation in just $\zeta$

In order to use the formalism developed for the path integral method we need to convert the equations to ODEs. We here use the inverse of the Laplacian $\nabla^2$. Note that $\zeta$ and $\psi$ are related by an algebraic transformation, $\zeta = \nabla^2 \psi$. We may then define $\psi = \frac{1}{\nabla^2} \zeta$, where $\frac{1}{\nabla^2}$ is the inverse operator of the Laplacian. However, from the discussion of the SVD, we must first work in fields which have 0 means.

The mean of $\psi$ is easy to remove by definition. Since the equations do not depend on $\psi$, but only on its derivatives, and the addition of an arbitrary constant to $\psi$ still satisfies the boundary conditions - we may enforce $\psi = 0$ with no information loss. This may mean that $\psi$ has a constant (but non zero) value at the boundary, in conflict with usual practice, but such problems are easily transferred by simply adding a constant to these cases.

However, the removal of $\bar{\zeta}$ is harder to justify from simply the equations. In fact, in general $\zeta$ will have a nonzero mean. This will imply that relevant information on the vorticity will be lost. Therefore, in order to justify our using of the SVD, we need to examine any interpretation that allows us to remove the mean of the vorticity from our equations. We may generate such an interpretation if we can express the equations in 2 parts, 1 for mean and one for variations.

First, we write the vorticity in mean vorticity ($\bar{\zeta}$) and the variation $\zeta - \bar{\zeta} \equiv \zeta_v$

The equation for $\bar{\zeta}$ then becomes

$$\frac{d\zeta (x,y,t)}{dt} = \bar{J}(\psi (x,y,t), \zeta (x,y,t)) - \epsilon \zeta (x,y,t) + \nu \nabla^2 \zeta (x,y,t) + \bar{\zeta}$$

$$\frac{d\zeta_v (x,y,t)}{dt} = J(\psi (x,y,t), \zeta_v (x,y,t)) - \epsilon \zeta_v (x,y,t) + \nu \nabla^2 \zeta_v (x,y,t) + (\bar{\zeta} - \bar{\zeta})$$

However, since $\bar{J}(\psi (x,y,t), \zeta (x,y,t)) = 0$ (recalling that the Jacobian conserves mean vorticity), and that $\nabla^2 \bar{\zeta} = 0$, then the equation for the mean vorticity becomes easy to solve using elementary calculus:
\[
\frac{d\zeta(t)}{dt} = -\epsilon \zeta(t) + \bar{\zeta}
\]

Or, in another words

\[
\bar{\zeta}(t) = \begin{cases} 
\frac{1}{-\epsilon} \left( \epsilon \zeta(t_0) - \bar{\zeta} \right) e^{-\epsilon t} + \bar{\zeta} & \text{for } \epsilon \neq 0 \\
\bar{\zeta} \cdot t & \text{for } \epsilon = 0
\end{cases}
\] 

(4.0)

and the equation for \(\zeta_V\) is the same as the original equation. Note explicitly that if we start with \(\zeta = \bar{\zeta} = 0\) then \(\zeta = 0\) for all time. Therefore, we may compute mean separately (using analytical formulation), and consider only the equation for the vorticity with the mean removed. We can add in the mean after we have determined \(\zeta_V\), however, including the mean is not germane to our project, and we do not compute it.

### 4.2 Shallow Water Equations

#### 4.2.1 derivation

The shallow water equations may be derived from the original momentum equations with the following assumptions \cite{Vallis2006}

- The water is shallow - specifically that there is no vertical variation in the horizontal velocities
- The waves are shallower - specifically, there is a mean water depth, and the variations on that depth are some small percentage of that mean depth
- The density is constant.
- The pressure is determined by hydrostatic balance \(\frac{\partial P}{\partial z} = -\rho g\)
With these assumptions we may start with the (3 dimensional) Navier Stokes momentum equation

$$\rho \frac{D \vec{U}}{Dt} = -\nabla \cdot \vec{P} + \vec{F}'$$

where $\vec{F}'$ includes all the body forces.

For geophysical flows the body force vector includes both gravity and the Coriolis force. Under the last assumption the deviatoric stress tensor reduces to the gradient of the pressure and a kinematic viscosity term. As a result we may express the Navier Stokes equations as

$$\rho \frac{D \vec{U}}{Dt} + \vec{f} \times \rho \vec{u} = -\nabla \vec{P} - \rho g \hat{z} + \vec{F}$$

where the gradient of pressure is the 2 dimensional gradient, $\vec{F}$ includes all the body forces excluding Coriolis ($\vec{f}$) and gravity.

Using the last 2 assumptions, we may may integrate the pressure in the $\hat{z}$ direction, noting that the "constant of integration" can be a function of $x$ and $y$. If we define the bottom topography is $\eta_B(x,y)$, the top of the fluid column is $\eta(x,y,t)$ and the column heigh $h(x,y,t) = \eta(x,y,t) - \eta_B(x,y)$, we may write the pressure as $P(x,y,z) = \rho g (\eta(x,y) - z)$. We may also note that the $z$ derivative of the pressure $= -\rho g$ exactly cancels out the gravitational forces $- \frac{\partial P}{\partial z} - \rho g = 0$

As a result, we have no vertical forces (excluding the vertical component of the Coriolis force, which we ignore). In addition, the horizontal gradient of pressure is independent of height ($= -\rho g \nabla_2 \eta(x,y)$). This means that the horizontal velocities
will also be independent of \( z \). We now write the momentum equation for the horizontal velocities as

\[
\frac{D}{Dt} \mathbf{U} + \mathbf{f} \times \rho \mathbf{U} = -\rho g \nabla \eta (x,y)
\]

where this is in the horizontal (2 dimensional) space.

For the final equation we use mass conservation. This equation may be generally written as

\[
\frac{\partial}{\partial t} \int \int_V \rho \, dV = -\int_S \rho \mathbf{u} \cdot d\mathbf{S} = -\int_S \mathbf{u} \cdot d\mathbf{S} \quad (4.0)
\]

As a surface, consider a column from the bottom to the surface, then integrate over the height. The right hand integral is strictly a horizontal integral (as \( d\mathbf{S} \) is a horizontal vector. The flux through the \( \mathbf{z} \) direction is accounted for by the height, which is varying in both space and time. As a result, with the "volume" now a 2 dimensional square in \( x,y \), and our "surface being the square’s perimeter, we write (4.2.1) as

\[
\frac{\partial}{\partial t} \int \int_V (\eta(x,y,t) - \eta_b(x,y)) \, dV = -\int_S (\eta(x,y,t) - \eta_b(x,y)) \mathbf{u} \cdot d\mathbf{S}
\]

Next we use Gauss’ theorem to express \( \int_S \eta \mathbf{u} \cdot d\mathbf{S} = \int \int_V \nabla \cdot (\eta \mathbf{u}) \). As a result, we may write (4.2.1) as

\[
0 = \int \int_V \left( \frac{\partial h(x,y,t)}{\partial t} + \nabla \cdot (h \mathbf{u}) \right)
\]

As the volume (or surface in our example) may be arbitrary, we require that the integrand be 0. This completes the shallow water equation system (recalling that the \( \mathbf{u} \) equations are the 2 dimensional horizontal equations

\[
\frac{D}{Dt} \mathbf{u} + \mathbf{f} \times \mathbf{u} = -g \nabla \eta \]

\[
\frac{\partial h}{\partial t} + \nabla \cdot (h \mathbf{u}) = 0
\]
4.2.2 Ekman pumping

There are two possible ways of implementing an external force on the fluid. The first is to include it as a body force added directly to the right hand side of the horizontal equations. In this case, the water (although obeying the shallow water equations for a fluid) is, in some sense, being "pushed as a rigid slab", or, in another words, the fluid is transmitting the external force throughout its body in a manner similar to rigid body dynamics. This is generally used for short time scales. In this case, if a body of water is well mixed, we may assume that the boundary layers (necessary for satisfying the boundary conditions) are being immediately mixed into the body of fluid and thus transmitting the momentum directly into the fluid. This interpretation has been applied to estuarial and coastal modelling Kärnä et al. (2011), as well as describing the western boundary current Munk (1950). This would be an analogy to a well-mixed layer where the momentum is being transmitted directly to the interior by mixing. This implementation treats the entire body of water as the boundary layer.

This method has applications listed above, but also conceptual counterarguments. The main one would be that this solution will not, in general, satisfy the boundary conditions, as the horizontal vector velocity is height independent.

We frequently model the boundary conditions at the bottom layer parametrically with a Rayleigh friction. As the external force acts directly on the body, the forcing term (if the external stress is $\tau$) is $\frac{DU}{Dt} = ... + \tau - k U$, where $k$ makes the equation dimensionally correct. By then adding the term $- k U$ to the already existing Rayleigh bottom parameterization, we get the form of the equation where the external stress is added directly into the horizontal momentum equations.

Again we note that this solution does not satisfy the boundary conditions. The bottom boundary condition and top boundary conditions are parameterized by Rayleigh friction and the top by the forcing (minus the current velocity). However, as the horizontal velocity is height independent, this implies that only the trivial solution can satisfy both.
When we model the upper boundary layer, the wind stress, or surface stress on
the fluid must be handled differently. In this case, the horizontal wind stress \((\vec{\tau},\text{which}
\) is actually the momentum transport), results in a downward (or upward) velocity into the
liquid, known as Ekman pumping.McWilliams (2006); Pedlosky (1979); Vallis (2006)
This velocity may be written as \(\vec{z} \cdot \left(\nabla \times \frac{\vec{\tau}}{f}\right)\). This velocity will be an addition to
the time derivative of the height.

Then the total height of the fluid column is \(h = \eta - \eta_B(x,y)\). In this case we
may write the shallow water system as 2 coupled equations:

\[
\frac{D\vec{u}}{Dt} + f\hat{k} \times \vec{u} = -g \nabla \eta \\
\frac{\partial h}{\partial t} + \nabla \cdot (h \vec{u}) = h \vec{z} \cdot \left(\nabla \times \frac{\vec{\tau}}{\rho f}\right)
\]

where we have removed the coriolis force from the generic body force term, and have
put it on the LHS of the equation. We may then use the vector identity

\[
\nabla \times \left(\nabla \times F\right) = \nabla_F \left(\nabla \cdot F\right) - \left(\nabla \cdot \nabla\right) F
\]

where \(\nabla_F\) means that the gradient operator only acts on the "F" vector. Replacing F and
\(v\) with \(u\) (the velocity gives us the following representation

\[
\frac{\partial \vec{u}}{\partial t} = - \left\{ (f + \zeta) \hat{k} \times \vec{u} + \nabla \left( \frac{P}{\rho_0} + \frac{1}{2} \left| \vec{u} \right|^2 \right) \right\}
\]

where

\[
\begin{align*}
    u_x &= \vec{u} \cdot \hat{x} \\
    u_y &= \vec{u} \cdot \hat{y} \\
    \zeta &= \frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}
\end{align*}
\]
4.2.3 Discretization

The discrete implementation of the SWE we use is an enstrophy conserving shallow water system on a staggered C grid Sadourny (1975). The figure above shows the locations of each of the quantities in question. Both the figure and the discussion are from Sadourny (1975). First we will need to define how we will identify the staggered elements. The staggered C-grid defines:

- pressure (per unit mass) and height information on gridpoints. The pressure per unit mass is symbolized by $P$; the height is symbolized by $h$. For our case, $P$ is related to $h$ by a constant: the acceleration due to earth’s gravity near the surface of the earth.

- the $x$ component of velocities is defined on lines connecting adjacent horizontal gridpoints. The $x$ component (”east”) velocity is symbolized by $u$.

- the $y$ component of velocities is defined on lines connecting adjacent vertical
gridpoints. The y component ("north") velocity is symbolized by \( v \).

For example, a given horizontal velocity \( \mathbf{u} \), there will be 2 \( P \) values associated with it - one at the left end of the line segment containing \( \mathbf{u} \), and one at the right. In the formulas below, the average of these 2 \( P \)s would be written as \( \overline{P} \). We will use this notation when defining horizontal mass fluxes and other intermediate quantities below.

The counterpart of this is the finite derivative, as opposed to the finite average. We use the symbol \( \delta \) for finite derivatives, with the subscript indicating the direction the derivative is taken. For example, when computing the vorticity \( \zeta \) we need the finite derivatives of velocity, or the curl. We may again notice that the position of the vorticity has an "above" and "below" horizontal velocity \( u \). The finite derivative corresponding to \( \frac{\partial}{\partial y} \) is \( \delta_y = (\text{above } u - \text{below } u) / \text{grid spacing} \).

Also, the finite x derivative of the velocity in the \( \hat{x} \) direction is located at a grid point. As the similar y derivative of the velocity in the \( \hat{y} \) direction is also at a grid point, we may define the horizontal divergence of velocity at the grid points.

We may then define mass fluxes located at the same places as the velocity components, as well as an energy term \( H \) at each grid point and a vorticity at the center of each square formed by 4 grid points:

\[
U = \overline{P} \hat{x} u \tag{4.-6}
\]
\[
V = \overline{P} \hat{y} v \tag{4.-5}
\]
\[
H = P + \frac{1}{2} \left( \overline{u'^2} + \overline{v'^2} \right) \tag{4.-4}
\]
\[
\zeta = \frac{\overline{\delta_y u - \delta_x v}}{\overline{P} \delta y} \tag{4.-3}
\]

where the left hand values are computed at their points, and any overbars are computed using the proper spacial averages. Note, by dimensional analysis, \( H \) and \( P \) are in dimensions of pressure per unit mass, or meters\(^2\)/seconds\(^2\). Finally, notice the double bar on \( P \) in the \( \zeta \) definition. The represents an average over the 4 corners of the square surrounding the location of \( \zeta \).
For example, consider computing a particular $U$. The $U$ (horizontal mass transport per unit mass) is located at the same location as the horizontal velocity $u$ (on the lines connecting 2 horizontally adjacent points). In order to compute the appropriate $P$ (pressure, related to height), we would need the pressure defined at the location of the $U$. As the $U$ vector spans horizontally adjacent $P$ values, we will use the horizontal average of these 2 for the computation. This is written as $\overline{P}$.

Finally, Sadourny (1975) presents 2 separate discretization schemes - an energy conserving and an enstrophy conserving scheme. We use the latter. The vector field we use is the following

$$
\begin{align*}
\frac{\partial u}{\partial t} &= \bar{\zeta} y V_{xy} - \delta_x H \\
\frac{\partial v}{\partial t} &= -\bar{\zeta} x U_{yx} - \delta_y H \\
\frac{\partial P}{\partial t} &= -\delta_x U - \delta_y V
\end{align*}
$$

Where $u$ and $v$ are evaluated along the vector lines, and $P$ is defined at each intersection, or grid point.

For example, in order to compute the ODE representing $u$ (or the $x$ component of velocity, you would compute the $y$ average of $\zeta$ (the average of the rotation values above and below the vector in question), the average of the 4 adjoining $V$ values (computed from $y$ velocity above), as well as the difference between the adjacent $H$ values, divided by the horizontal ($x$) grid spacing. An analogous procedure is employed to translate the remainder of the equations into computer form.

The scheme presented up to now is enstrophy-conserving. For our study we add bottom friction, lateral friction and forcing. We model bottom friction as a Rayleigh drag term; friction directly proportional to velocity. In which case, the bottom friction (to be added to the velocity terms) is co-located with each velocity. In addition we use a lateral drag $\nabla^2$. The Laplacian of a velocity is also located at the same place as the velocity, defining the second derivative as the difference between the forward derivatives of the current grid point and the preceding grid point (preceding in $x$ or $y$, depending
on whether we are taking the derivatives with respect to x or y).

The wind forcing will be a velocity, defined so that its curl will be located at the grid points (the location of the height data). This implies that the As a result, the final equations, including pressure and forcing are:

\[ \frac{\partial u}{\partial t} = \zeta^y \nabla^2 y - \delta_x H - R u + \nabla^2 u \]
\[ \frac{\partial v}{\partial t} = -\zeta^x \nabla^2 x - \delta_y H - R v + \nabla^2 v \]
\[ \frac{\partial P}{\partial t} = -\delta_x U - \delta_y V + g \nabla \times \frac{\tau}{\rho_f} \]

4.2.4 2 layer Shallow water equations

The picture and most of this discussion are taken from Vallis Vallis (2006). We start with a 2 layer system. The bottom topography is \( \eta_B \), and we have 2 layers, density \( \rho_i \), horizontal velocity \( \vec{u}_i \) and pressure \( p_i \), for i=1,2. In addition, the height of the top of layer 1 is \( \eta_0 \) and the height of the top of layer 2 is \( \eta_1 \). We will also use the reduced gravity \( g' = \frac{\rho_2 - \rho_1}{\rho_1} \). We may then write both height equations in the same form

\[ \frac{\partial h_i}{\partial t} + \nabla \cdot (h_i \vec{u}_i) = 0 \]
Using the definition of reduced gravity above, we may write pressures such that the momentum equations may be written as

\[
\frac{\partial \vec{u}_1}{\partial t} + (\vec{u}_1 \cdot \vec{\nabla}) \vec{u}_1 + \vec{f} \times \vec{u}_1 = -g \vec{\nabla} \eta_0
\]

\[
\frac{\partial \vec{u}_2}{\partial t} + (\vec{u}_2 \cdot \vec{\nabla}) \vec{u}_2 + \vec{f} \times \vec{u}_2 = -\frac{\rho_1}{\rho_2} \left( g \vec{\nabla} \eta_0 + g' \vec{\nabla} \eta_1 \right)
\]

We will make the simplifying assumption of no bottom topography. In this case we may write \(\eta_0 = h_1 + h_2\) and \(\eta_1 = h_2\). We now have 6 state vectors, \(\eta\) and \(\vec{u}\) (x and y components) each of which have a value at each grid point.

We may then rewrite the 2 height equations to be expressed in the variable \(\eta\). By adding the 2 height equations to obtain \(\eta_0\)

\[
\frac{\partial \eta_0}{\partial t} + \vec{\nabla} \cdot (h_1 \vec{u}_1 + h_2 \vec{u}_2) = 0 \text{ (or)}
\]

\[
\frac{\partial \eta_0}{\partial t} + \vec{\nabla} \cdot ((h - h_2) \vec{u}_1 + h_2 \vec{u}_2) = 0
\]

\[
\frac{\partial h_2}{\partial t} + \vec{\nabla} \cdot (h_2 \vec{u}_2) = 0
\]

\[
\frac{\partial \vec{u}_1}{\partial t} + (\vec{u}_1 \cdot \vec{\nabla}) \vec{u}_1 + \vec{f} \times \vec{u}_1 = -g \vec{\nabla} \eta_0
\]

\[
\frac{\partial \vec{u}_2}{\partial t} + (\vec{u}_2 \cdot \vec{\nabla}) \vec{u}_2 + \vec{f} \times \vec{u}_2 = -\frac{\rho_1}{\rho_2} \left( g \vec{\nabla} \eta_0 + g' \vec{\nabla} h_2 \right)
\]

We may discretize these in the normal way. Finally, we are going to consider measuring only the top layer, then we will have data for \(\eta_0\) and \(\vec{u}_1\) only.
Chapter 5

Study

5.1 Objectives

In this section we use the methods, SWE model and discretization to provide a study of the following questions

• Can we quantify the quality of a state estimation?

• Can we define a ”minimum” number of measurements for accurate prediction?

• Can we determine the valid prediction window given the number of measurements?

• If we have fewer than the ”minimum” number of measurements, do we have options to increase the prediction window?

A quantification of item 1 would be ”can we define a number of measurements below which MHMC poorly predicts unmeasured states, and above which MHMC accurately predicts unmeasured states? We first need to quantify the idea of ”quality of state prediction”. For a twin experiment, we may compute A_0 using all data, both data presented and hidden from MHMC. We compute 2 separate metrics - A_0 for the entire data set, and the A_0 for the final values. These would be the initial conditions in a prediction model. Note that MHMC allows collection of systems of state vector estimates...
Table 5.1: Inputs to twin integration model

<table>
<thead>
<tr>
<th>symbol</th>
<th>value</th>
<th>units</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>8</td>
<td>N/A</td>
<td>x (and y) dimension of grid; we use a square grid</td>
</tr>
<tr>
<td>δ</td>
<td>10^5</td>
<td>meters</td>
<td>grid spacing</td>
</tr>
<tr>
<td>ϵ</td>
<td>10^{-7}</td>
<td>1/sec</td>
<td>Rayleigh (bottom) friction</td>
</tr>
<tr>
<td>ν</td>
<td>2 × 10^{-5}</td>
<td>m^2/sec</td>
<td>lateral (viscous) friction</td>
</tr>
<tr>
<td>F</td>
<td>0.2</td>
<td>m^2/sec^3</td>
<td>Ekman pumping maximum velocity (times gravitational constant.)</td>
</tr>
</tbody>
</table>

which, taken aggregately, will have the statistics (mean, standard deviation, etc.) approximately equal to the ”truth” in a twin experiment, or the statistics expected from the actual system in a physical experiment. These values will be used to compute the $A_0$. We graph $A_0$ as a function of number of reporting sites for various configurations and observe that the value decreases precipitously 136 measurements for the 8 by 8 (192 total state vectors). This is the number of vectors needed for accurate measurements.

For item 2, we recognize that prediction methods given incomplete state information is a current research topic, and will not attempt to evaluate these. We instead use a simple integration scheme on the available samples, and use this for our predictions at a given time. We will consider this simple prediction scheme as our metric. We may observe that we now establish a lower bound for prediction-prediction quality available to a numerical integrator.

For item 3, we will examine the length of time the prediction is valid as a function of $A_0$. We will not provide a full answer for item 4, however, we will discuss strategies for predicting given that there are insufficient measurements.

We use the following parameters in the 8 by 8 SWE: First, we confirm that this is a chaotic system. We do this by numerically integrating two ”nearby” trajectories. Using a time step of $\tau$, after a ”large” number of time steps $n$, then the approximate formula a maximum Laypunov exponent $\lambda = \frac{1}{n\tau} \sum_{i=1}^{n} \ln \left| \frac{\delta X_i}{\delta X_0} \right|$. Computing this using the values above and a time step of $3.6 \times 10^{-3}$ seconds yields a maximum Laypunov
exponent bounded by about $2.7 \times 10^{-5}$ per second, or approximately 0.1 per hour.

We generate a data set by integrating the shallow water equations forward. In order to ensure that the system has been integrated past any transients we start from initial conditions and integrate for 50 hours. We then begin our data set there. 5.1 is a graph of state vector 62. We are discussing an 8 by 8 system - with the columns numbered (from west to east) as 1 through 8, and rows numbered 1-8, south to north. So state vector 62 is the U (east-west velocity) between grid points 6,8 and 7,8 (recalling that velocities are defined between adjacent grid points, with the last velocity of each row being an entry which gives the periodic boundary conditions).5.2 is a graph of the north-south velocity for the same location. We will use these two data streams to examine the quality of fit for the unmeasured state vectors. These data will not be provided to the model in any of our examples.

Figure 5.1: Velocity profile for the East - West component of velocity at location 6,8

For the study we present 3 cases. For each of these cases we provide the model with all the height data, and a given number of rows and columns of velocity data (both in the U and V directions at each point).
In each case we run the simulation 8 times, using the final results of the previous run as input to the next. The monte carlo run parameters are

For our study we provided the model with height (or pressure per unit mass) data at each location. We provided varying amount of velocity data, from 1 to 4 columns, 1 to 4 rows and 1 to 4 rows and columns. We provide charts which show the locations of provided velocity (both east west and north south components). For the cases where we provide fewer than 4 columns (or 4 rows) simply remove the eastmost column(s) (or northmost row(s)).

I use the twin data to compare with the expected/predicted data using the following formula: (using $\vec{x}$ for the expected/predicted data and $\vec{y}$ for the twin data (truth), $x_n^l$ is the $l^{th}$ the value of the $l^{th}$ dimension of the state vector at time $n$) $\sum_{i=1}^{N} \sum_{i=1}^{D} (x_n^l - y_n^l)^2$. Note that now all the data is included, comparing both measured and unmeasured estimates equally. We first compare the values of this metric vs. number of repeated runs for each of the cases. We then examine this metric using prediction as well as fitted data, and look at the length of time we may predict. We present the data below - the first two

Figure 5.2: Velocity profile for the North - South component of velocity at location 6,8
Table 5.2: Inputs for MHMC

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of time steps in time series</td>
<td>4200</td>
</tr>
<tr>
<td>Dimension of system</td>
<td>192</td>
</tr>
<tr>
<td>(is 3 times grid dimension - 2 velocities and height at each grid point)</td>
<td></td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>600000</td>
</tr>
<tr>
<td>Number warm up iterations</td>
<td>320000</td>
</tr>
<tr>
<td>Number Simulated Annealing steps</td>
<td>240000</td>
</tr>
<tr>
<td>Initial $\beta$ for simulated annealing</td>
<td>0.00002</td>
</tr>
<tr>
<td>Report frequency</td>
<td>100</td>
</tr>
<tr>
<td>(when data copied from GPU to CPU)</td>
<td></td>
</tr>
<tr>
<td>Initial step size for velocity guesses</td>
<td>1.0</td>
</tr>
<tr>
<td>Initial step size for pressure guesses</td>
<td>10.0</td>
</tr>
<tr>
<td>Optimum acceptance frequency</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Figure 5.3: Location of measurements provided for 4 columns of data
Figure 5.4: Location of measurements provided for 4 rows of data

Figure 5.5: Location of measurements provided for 4 rows and 4 columns of data
columns are number of measured columns and run number (results of previous run used as initial conditions to next run).

We note the following:

- We recall that the full $A_0$ is calculated using the model and data errors, effectively setting $R_f = R_m = 1.0$. Therefore, even though there are more error terms, the “full” $A_0$ is lower than the ”presented” $A_0$.

- The “full” $A_0$ approaches final value sooner than the ”presented” $A_0$. Therefore, when the ”presented” $A_0$ approaches its asymptotic limit, we may be sure that the actual fit (including unmeasured, which is our concern).

- We continue to see improvement through at least 8 repetitions of the run (600,000 iterations per repetition). We may conclude that these runs will need at least 5 - 10 million iterations in order to come to final answer. We may use this information, along with timing information, to calculate the number of iterations needed.

Figure 5.6: Value of ”presented” $A_0$ with columns of data added
Figure 5.7: Value of "presented" $A_0$ with rows of data added

Figure 5.8: Value of "presented" $A_0$ with both columns and rows of data added
Figure 5.9: Value of "full" $A_0$ with columns of data added

Figure 5.10: Value of "full" $A_0$ with rows of data added
We see a parallel between the rows only and the columns only graphs. However, note the scale of the rows and columns. We see that the case of 4 rows and 4 columns provides very accurate data.

We next examine how graphically how accuracy of fit is affected as velocity measurements added by columns are added for our test location 6,8. We compare both north-south and east-west components of the velocity at location 6,8. We observe that our estimate of the unmeasured north-south component of velocity remains poor even as additional columns of data are provided, while we estimate the east-west component of velocity well with 3 or more columns of data.

- Measure column 1 (figures 5.12 and 5.13)
- Measure columns 1 and 3 (figures 5.14 and 5.15)
- Measure columns 1, 3 and 5 (figures 5.16 and 5.17)
Figure 5.12: Estimate of east-west velocity profile for location 6,8 given 1 column of velocity data

- Measure columns 1, 3, 5 and 7 (figures 5.18 and 5.19)

Next, we examine the fit given 2 rows and 2 columns of data provided. We can see that the quality of the fit has not improved significantly. Finally, we examine the fit providing the model with all P, and 4 columns of velocity data. We get very good fit to the expected data for the east-west component, however, it is little better than the 3 column data. The north-south component of the velocity is better than for the 3 column data, but still not very accurate.

- Measure row 1 (figures 5.20 and 5.21)
- Measure rows 1 and 3 (figures 5.22 and 5.23)
- Measure rows 1, 3 and 5 (figures 5.24 and 5.25)
- Measure rows 1, 3, 5 and 7 (figures 5.26 and 5.27)
Figure 5.13: Estimate of north-south velocity profile for location 6,8 given 1 column of velocity data

Figure 5.14: Estimate of east-west velocity profile for location 6,8 given 2 columns of velocity data
Figure 5.15: Estimate of north-south velocity profile for location 6,8 given 2 columns of velocity data

Figure 5.16: Estimate of east-west velocity profile for location 6,8 given 3 columns of velocity data
Figure 5.17: Estimate of north-south velocity profile for location 6,8 given 3 columns of velocity data

Figure 5.18: Estimate of east-west velocity profile for location 6,8 given 4 columns of velocity data
Figure 5.19: Estimate of north-south velocity profile for location 6,8 given 4 columns of velocity data

Figure 5.20: Estimate of east-west velocity profile for location 6,8 given 1 row of velocity data
Figure 5.21: Estimate of north-south velocity profile for location 6,8 given 1 row of velocity data

Figure 5.22: Estimate of east-west velocity profile for location 6,8 given 2 rows of velocity data
Figure 5.23: Estimate of north-south velocity profile for location 6,8 given 2 rows of velocity data

Figure 5.24: Estimate of east-west velocity profile for location 6,8 given 3 rows of velocity data
Figure 5.25: Estimate of north-south velocity profile for location 6,8 given 3 rows of velocity data

Next, we examine the fit given 2 rows of data provided. We can see that the quality of the fit has not improved significantly. Finally, we examine the fit providing the model with all P, and 4 columns of velocity data. We get very good fit to the expected data for the north-south component, however, it is little better than the 3 column data. The east west component of the velocity is poorly estimated. In this study, we observe that rows of (x velocity), or columns of V (y velocity) do not provide good estimates. We interpret this by observing that providing U already gives some information on row values (as U is the velocity in the x direction), while V provides column information. Therefore, providing rows of U values provides "duplicate" row information, while providing columns of U provides both row (U values) and column (the column of U) information.

As a specific observation, using the stream function $\psi$, we may write $V$ as $\frac{\partial \psi}{\partial x}$, and as a result, information on all Vs give information on numerical derivatives in both the x and y directions, or information on both $\frac{\partial^2 \psi}{\partial x^2}$ and $\frac{\partial^2 \psi}{\partial x \partial y}$. However, (noting that U is $\frac{\partial \psi}{\partial y}$) providing a row of U only provides additional information on $\frac{\partial^2 \psi}{\partial x \partial y}$.
Figure 5.26: Estimate of east-west velocity profile for location 6,8 given 4 rows of velocity data

Figure 5.27: Estimate of north-south velocity profile for location 6,8 given 4 rows of velocity data
duplicating information already available to the model. It is only when a column of U is presented does the missing information \( \frac{\partial^2 \psi}{\partial y^2} \) become available to the model. The same argument (using the stream function argument) may be used to explain the need for a row of V when all U are presented (The missing information is \( \frac{\partial^2 \psi}{\partial x^2} \)).

We then provide the model with row-column combinations. As we observed, supplying only rows (columns) to the model results in poor estimates in north-south (east-west) velocity. This suggests that row-column combinations will give full synchronization. We next try the following cases:

- Measure row 1 and column 1
- Measure rows 1 and 3 and columns 1 and 3
- Measure rows 1,3 and 5 and columns 1, 3 and 5
- Measure rows 1,3,5 and 7 and columns 1, 3, 5 and 7
- Measure rows 1,3,5 and 7 and columns 1, 3, 5 and 7, excluding 1,1
- Measure rows 1,3,5 and 7 and columns 1, 3, 5 and 7 excluding 1,1 and 2,1

We determine quality of fit by comparing differences between our results and the "truth" of the twin over a fitting and then predicting. Prediction we may also qualitatively determine the length of time that our data and fit allow accurate prediction. First, we examine the fit given only 1 row and 1 column of data provided. We can see that the quality of the fit is poor. Next, we examine the fit given 2 rows and 2 columns of data provided. We can see that the quality of the fit has not improved significantly. Next, we examine the fit providing the model with all P, and 3 rows and 3 columns of velocity data. We notice that the fit has improved to excellent. Finally, we examine the fit providing the model with all P, and 4 columns of velocity data. We get very good fit to the expected data, however, it is little better than the 3 column data. As the 3 column data gives excellent estimation of unmeasured states, the addition of an additional row and column of data is unnecessary. We may then use the output of these runs for
Figure 5.28: Estimate of east-west velocity profile for location 6,8 given 1 row and 1 column of velocity data

Figure 5.29: Estimate of north-south velocity profile for location 6,8 given 1 row and 1 column of velocity data
Figure 5.30: Estimate of east-west velocity profile for location 6,8 given 2 rows and 2 columns of velocity data

Figure 5.31: Estimate of north-south velocity profile for location 6,8 given 2 row and 2 columns of velocity data
Figure 5.32: Estimate of east-west velocity profile for location 6,8 given 3 rows and 3 columns of velocity data

Figure 5.33: Estimate of north-south velocity profile for location 6,8 given 3 rows and 3 columns of velocity data
Figure 5.34: Estimate of east-west velocity profile for location 6,8 given 4 rows and 4 columns of velocity data

Figure 5.35: Estimate of north-south velocity profile for location 6,8 given 4 rows and 4 columns of velocity data
Figure 5.36: Estimate of east-west velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 of velocity data

Figure 5.37: Estimate of north-south velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 of velocity data
Figure 5.38: Estimate of east-west velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 and 2,1 of velocity data

Figure 5.39: Estimate of north-south velocity profile for location 6,8 given 4 rows and 4 columns excluding 1,1 and 2,1 of velocity data
prediction. We observe that increased fidelity in the quality of assimilation leads to an increase in the quality of fit. We first evaluate the quality of fit qualitatively, by determining how long we get a fit whose error bars include the values provided by the twin data. In this case we examine the fits for 58 hours after the initial fitting. We present the fits from the cases above. Notice that the time range is 100 hours. The first 42 hours are the fit. This is information duplicated from above. The next 58 hours of the graph show the prediction. We use this prediction information to quantify the quality of the prediction and compare to the quality of fit.

5.2 Prediction

As an additional measure of the quality of the state and parameter estimation, we include a section on prediction. As mentioned in the section on the M-H algorithm, we may retain a group of accepted paths. This group will be a group which reflect the distribution which reflects the relative likelihood of each member of the group given the cost function. Each of these available paths is integrated forward using a standard “particle swarm” (or “particle filter”) method. We use a standard 4th order Runge-Kutta algorithm to integrate forward, and we integrate forward for at least 80 hours.

Our metric for the quality of the state and parameter estimation in this study is the quality of prediction. We illustrate in graphs the mean and standard deviation (as error bars). We examine the graphs of predictions obtained for the following cases

- 3 rows of velocity data
- 3 columns of velocity data
- 3 columns and 3 rows of velocity data
- 4 rows of velocity data
- 4 columns of velocity data
Figure 5.40: Prediction of east-west velocity profile for location 6,8 given 3 columns of velocity data

- 4 columns and 4 rows of velocity data
- 4 columns and 4 rows of velocity data excluding 1,1
- 4 columns and 4 rows of velocity data excluding 1,1 and 2,1

In each case we again use site 6,8 as our validation site. We can see from our graphs that the 4 column and 4 row case provides good estimation and long prediction times. In this case we see that the estimates of the states at the end of the assimilation period are excellent, and we are able to get high quality predictions for about 40 hours.

The case where we have provided 4 columns and 4 rows excluding point 1,1 and excluding 1,1 and 2,1 both also provide good estimation, but the prediction times are lower. Visually inspecting the graphs for excluding 1,1 5.52 and 5.2 yields a prediction window of approximately 20 hours. Inspecting the graphs for excluding 1,1 and 2,1 5.54 and 5.2 yields a prediction window of approximately 20 hours again, although it is visible that the graphs with the additional point (only excluding point 1,1) provide slightly better prediction.
Figure 5.41: Prediction of north-south velocity profile for location 6,8 given 3 columns of velocity data

Figure 5.42: Prediction of east-west velocity profile for location 6,8 given 3 rows of velocity data
Figure 5.43: Prediction of north-south velocity profile for location 6,8 given 3 rows of velocity data.

Figure 5.44: Prediction of east-west velocity profile for location 6,8 given 3 rows and 3 columns of velocity data.
Figure 5.45: Prediction of north-south velocity profile for location 6,8 given 3 rows and 3 columns of velocity data

Figure 5.46: Prediction of east-west velocity profile for location 6,8 given 4 columns of velocity data
Figure 5.47: Prediction of north-south velocity profile for location 6,8 given 4 columns of velocity data

Figure 5.48: Prediction of east-west velocity profile for location 6,8 given 4 rows of velocity data
Figure 5.49: Prediction of north-south velocity profile for location 6,8 given 4 rows of velocity data

Figure 5.50: Prediction of east-west velocity profile for location 6,8 given 4 rows and 4 columns of velocity data
Figure 5.51: Prediction of north-south velocity profile for location 6,8 given 4 rows and 4 columns of velocity data

Figure 5.52: Prediction of east-west velocity profile for location 6,8 given 4 rows and 4 columns of velocity data excluding 1,1
Figure 5.53: Prediction of north-south velocity profile for location 6,8 given 4 rows and 4 columns of velocity data excluding 1,1

Figure 5.54: Prediction of east-west velocity profile for location 6,8 given 4 rows and 4 columns of velocity data excluding 1,1 and 2,1
The other cases have lower quality estimates and predictions. The cases where 3 rows or 3 columns are measured are poor. The prediction immediately diverges from the twin data. If we only measure 4 columns or 4 rows then we get concordance with twin data for several hours. We observe a few qualitative features of the predictions in this case. Although neither the east-west nor north-south velocity estimates do not capture the height of the peak directly after the beginning of prediction, the prediction correctly identifies the time of the peak velocity (approximately 41-42 hours) as well as correctly modelling the next 2 or so hours. However, both components of velocity are poorly modeled after less than 10 hours.

We finally consider the question of the number of measurements needed for an accurate estimate of unmeasured states and parameters. We will not completely solve the problem, but make a few observations based on 1 case - the case where 4 rows and 4 columns are presented.

To consider this case, we run (although do not graph and present here) the case
where 4 rows and 4 columns of velocity data are presented, along with the velocity data from the “evaluation site”, site 6,8. The data for site 6,8 is well modelled when presented with the other data. We then re-examine the graphs 5.50 5.2 and observe that this site is well modelled. We then provide the algorithm for evaluating sufficient data presented.

- generate estimates and predictions providing model with all data
- remove 1 or more data item(s) from the stream
- re-run the estimates and predictions, using the removed data stream(s) now as verification.
- if the removed data streams are similar to the data, then we may assume that the model has synchronized to the data

Alternatively we may examine the action ?? and observe that the case which gives excellent predictions shows the action decreasing for at least 3-4 orders of magnitude over the non-synchronized cases. While we may not be absolutely certain that either of these observations are sufficient to determine synchronization (and quality prediction) we may certainly use them both as indication of lack of synchronization.

5.3 Analysis of Synchronization from an Information Theory Standpoint

We examine the following cases from a previous article Whartenby et al.. This article uses the 16 by 16 (not 8 by 8) SWE. All cases include measurements for all P. In addition we measure the following

1. No U, all V. This case results in bad fits
2. One column of U, all V. This case results in excellent fits
3. One row of U, all V. This case results in bad fits.
4. All U, one row of V. This case results in excellent fits.

5. All U, one column of V. This case results in bad fits.

We interpret this data as follows. First, we make the following observations.

- case 1 fails due to insufficient measurements.
- However, we cannot explain the failure of case 3 relative to the success of case 2 in terms of number of sites measured, since each case provides the same number of sites.
- We also notice a symmetry between cases 2 and 3 and cases 4 and 5.

In this study, we observe that rows of (x velocity), or columns of V (y velocity) do not provide good estimates. We interpret this by observing that providing U already gives some information on row values (as U is the velocity in the x direction), while V provides column information. Therefore, providing rows of Us provides "duplicate” row information, while providing columns of U provides both row (U values) and column (the column of U) information. We note that this interpretation also explains the results we obtain from our study.

As a specific observation, using the stream function \( \psi \), we may write V as \( \frac{\partial \psi}{\partial x} \), and as a result, information on all Vs give information on numerical derivatives in both the x and y directions, or information on both \( \frac{\partial^2 \psi}{\partial x^2} \) and \( \frac{\partial^2 \psi}{\partial x \partial y} \). However, (noting that U is \( \frac{\partial \psi}{\partial y} \)) providing a row of U only provides additional information on \( \frac{\partial^2 \psi}{\partial x \partial y} \), duplicating information already available to the model. It is only when a column of U is presented does the missing information \( \frac{\partial^2 \psi}{\partial y^2} \) become available to the model. The same argument (using the stream function argument) may be used to explain the need for a row of V when all U are presented (The missing information is \( \frac{\partial^2 \psi}{\partial x^2} \)).
Chapter 6

Conclusion

Data Assimilation for chaotic systems is an ongoing research topic. While much progress has been made on this subject, there is still much work to be done. In this dissertation we have examined the one layer shallow water equations in a chaotic regime. We used a path integral formulation to determine unmeasured states given incomplete measurements, and described the different numbers and positions of measurements needed in order to accurately estimate the undetermined ones. We described a method by which we can determine if a number of measurements is sufficient to accurately determine the unmeasured ones. Finally, we described how to use this method on other geophysical models, including the BVEs and 2 layer SWEs.

We have used the 1 layer SWE as a twin for our study. Based on the very low resolution 8 by 8 discretization and assuming all height measurements may be made, we have determined that, if we provide all the height data, we need approximately 50% of the remaining state vectors measured in order to accurately determine the unmeasured ones. We can see when we have 1 more state vector measured than we need - by removing 1 measurement, and observing that the estimation of this measurement gives an accurate approximation of this available (now ”twin”) data.

While the question of determining a priori how many measurements are needed for accurate estimation of the unmeasured state vectors, we may determine if a given
number and location is sufficient by comparing a known state vector element with its estimate when not presenting it to the PIM. We have not addressed the issue of time independent parameters, nor have we significantly addressed the effects of various levels of noise.

In this dissertation we accomplished the following

- We used the 8 by 8 SWE as a twin for the study
- We used MHMC to determine the effect of the number of measurements needed for accurate determination of unmeasured states
- We described the relationship between the number of measurements, the action and the accuracy of estimation and length of time where we may make valid predictions
- We made observations on the amount of information transferred to the model by measurements.

Further research along this line would include the following

- the effects of noisy data on the data needed
- the effects of scaling - how will this problem scale?
  - Will we retain the same ratio of data needed to total data as the size increases?
  - How will the GPU code scale as the size of the problem scales?
- Can we further relate the idea of information transferred from measurements to model to the number of measurements needed?
- Can we relate the previous investigations into the ”smoothness of the synchronization manifold”
The items listed as further research opportunities would be amenable to the methodology in this study. A researcher could repeat this study with a twin data set with differing levels of noise, or re-code the MHMC to take advantage of a massively parallel implementation. Additionally, a researcher could do theoretical study to devise a method to calculate information transferred to the model using 2.1. Finally, a researcher could study the idea of a synchronization manifold from forward integration or synchronization error using nonlinear optimizers. Each of these studies could be based on techniques described in this dissertation.
Appendix A

Methods for Inverting Singular Matrices

In many applications, including this one, we need to compute the discrete analogue to the function $\frac{1}{\nabla^2}$. However, the discrete Laplacian is a singular matrix, preventing direct inversion. In this section we will explore two different methods for inverting the discrete Laplacian. We used the first presented method for our codes.

Description of Singular Value Decomposition

The singular value decomposition is a method used to allow computation of a solution to ill-conditioning; that is, when the matrix may not be inverted (there are not the same number of independent constraints as unknowns).

The SVD computes the decomposition of a $n$ by $p$ matrix $A$ as $A = U\Sigma V^T$, where $U$ is an $n$ by $n$ matrix, $\Sigma$ is a $n$ by $p$ matrix, and $V$ is a $p$ by $p$ matrix. Additionally, $U^TU = V^TV = I$ and $\Sigma$ is a diagonal matrix of the eigenvalues, that is $\Sigma_{i,j} = \delta_{i,j}\lambda_i$.

In this expression, the inverse $A^{-1} = V\Sigma^{-1}U^T$.

This is a (computationally extremely expensive) way to compute an inverse of a non-singular matrix. However for a singular matrix, one or more of the eigenvalues are zero, which prevents the method from providing an inverse.
The problem is solved by removing the zero eigenvalues from the inverse. That is, for each \( \lambda_i = 0 \), set \( \lambda_i^{-1} = 0 \) in \( \Sigma^{-1} \).

Numerically we can replace \( \frac{1}{\epsilon} = \inf \) with 0 for the following reason: The \( \epsilon \) eigenvalue will be associated with an eigenvector. For the system we will use \( (\nabla^2 \psi = \zeta) \) the small eigenvalue means that \( \vec{\zeta} \) is only extremely weakly dependent on changes of this eigenvector of \( \psi \). As a result, for many applications it is a good assumption that the inverse should be true as well. The SVD effectively decouples a weakly interacting eigenvector from the system. However, this means that the dimensionality of the system is reduced by the number of 0 eigenvalues, or, that there will be an eigenvector removed from \( \psi \) and \( \zeta \) (\( \psi \) and \( \zeta \) may be seen as simply different representations of the solution space, like a different coordinate system in physical space). The removal of the eigenvector in \( \zeta \) is the result of the replacement of \( \frac{1}{0} \) by 0. Therefore we need a complete understanding of which eigenvector(s) are being excluded as having 0 eigenvalues.

We may independently confirm that the missing eigenvector is actually \( \vec{\zeta} \) by looking at the pseudo inverse - the eigenvector corresponding to the removed eigenvalue is 0.

\[ \text{we can do this by using the SVD to compute the eigenvector associated with the removed eigenvalue show this later and show that the removed eigenvector on the other side is } \vec{\zeta}. \text{ You can do this by adding the equation for } \vec{\psi} \text{ as an additional equation, and notice that this will linearly relate } \vec{\zeta} \text{ to } \vec{\psi}. \text{ Since } \vec{\psi} \text{ is the removed eigenvector, we wish to set } \vec{\psi} = 0 \text{ and maintain it there. We will use the SVD to show that this means that } \vec{\zeta} \text{ information will be lost.} \]

As a result, the SVD should be used in systems where the mean of \( \vec{\psi} \) and \( \vec{\zeta} \) may be removed.
Green’s Function

In general, we may use a Green’s function to solve Laplace’s equations, as the Green’s function for this equation is known. This would require boundary conditions, and the conditions on the vorticity are not easily determined. In addition, this method does not bring about a major reduction in computational effort. However, we present it here for completeness.

Using the result $\zeta = \nabla^2 \psi$ then we can write $\psi$ in terms of $\zeta$ in the following way

$$\psi (\vec{x}) = \int G (\vec{x}, \vec{x}') \zeta (\vec{x}') \, d\vec{x}'$$

in two dimensions this becomes

$$\psi (x, y) = \int G (x, x', y, y') \zeta (x', y') \, dx' \, dy'$$

and the Green’s function (or Kernel) for the Laplacian operator is well known. Its specific form is

$$G (x, x', y, y') = \frac{1}{4\pi} \ln \left( \frac{(x-x')^2 + (y-y')^2}{r'^2} \right) \quad (A.-1)$$

(notice that the sign is the opposite of convention. This is because the conventional equation has the opposite sign for the $\nabla^2$ term)

One problem we must address is the problem is with the numerical integration of equation (A.-1). The apparent singularity at $x = x': y = y'$ means that we can not explicitly numerically integrate this Green’s function without special consideration of this point. This is not an analytic singularity for the following reason. For an integral around the area near the singularity of the Green’s function, change to polar coordinates

$$\psi (x, y) = \int_0^\epsilon \int_0^{2\pi} \zeta (r', \theta') \frac{1}{4\pi} \ln (r'^2) r' \, dr' \, d\theta$$

$$= \int_0^\epsilon \int_0^{2\pi} \zeta (r', \theta') \frac{1}{2\pi} \ln (r') r' \, dr' \, d\theta \quad (A.-1)$$
and use the fact that $\lim_{r \to 0} r \ln (r) \to 0$ to conclude that there is no analytical singularity at $r = 0$.

The issue is with a numerical solution to this problem. The way to avoid the inf near the origin is to simply set to 0 any terms which are sufficiently close to the singularity. However, this method may still result in overestimating $\psi$, as this method will cut off only negative contributions. As a result, I recommend for some very small $\epsilon$ define the numerical contribution of the Green’s function to be

$$
\epsilon < r \quad \text{use} \quad \frac{1}{4\pi} \ln (r^2) = \frac{1}{2\pi} \ln (r)
$$

$$
r < \epsilon \quad \text{use} \quad 0.0
$$

We can then use a numerical integration to determine $\psi$ from $\zeta$. Discretizing equation A.-1 gives a numerical expression for $\psi$ given $\zeta$

$$
\psi (x, y) = \sum_{n=0}^{N_y} \sum_{m=0}^{N_x} \zeta (x_m, y_n) G (x, x_m, y, y_n)
$$

Using $r_{m,n}^2 = (x - x_m)^2 + (y - y_n)^2$

$$
G (x, x_m, y, y_n) = \begin{cases} 
\frac{1}{2\pi} \ln (r_{m,n}) & \text{if } 1 < r_{m,n} \\
\frac{r_{m,n}}{2\pi} \ln (r_{m,n}) & \text{if } \epsilon < r_{m,n} < 1 \\
0 & \text{if } r_{m,n} < \epsilon
\end{cases}
$$

or, defining matrices $G_{m,n} (x,y) = G (x, x_m, y, y_n)$ and $\zeta_{m,n} = \zeta (x_n, y_m)$ (notice the reversal of $m$ and $n$ to allow for matrix multiplication notation) $\psi (x,y) = \text{Tr} ( G (x,y) \times \zeta )$.

As this is a linear equation in $\zeta$ it may be inverted, and the inversion gives $\zeta$ in terms of $\psi$

First, consider the case where we have a square grid: $N_x = N_y$ with a small but finite step size. In this case, there are no points too close to the singularity (except for the point itself) and so we may write the Green’s function as

$$
\Sigma' \ln \left( \text{dist}\{x_n - x'_n\} \right) \zeta_n \quad \text{(A.-4)}
$$

$$
\psi (x,y) = \text{Tr} ( G (x,y) \times \zeta )
$$
Since we may express $\psi$ in terms of $\zeta$ and vice versa we are left with one equation (put everything in terms of $\zeta$). Viewing each point on the grid as an individual equation, we now have $N^2$ “dimensions” for our problem. As a result, the equation ?? becomes

$$\frac{\partial \zeta}{\partial t} + \vec{v} \cdot \nabla \zeta = 0,$$

where

$$\vec{v} = \vec{k} \times \nabla \psi = \vec{k} \times \nabla \text{Tr} (G (x,y) \zeta) \quad (A.-5)$$

which in 2 dimensions gives us the equation

$$\frac{d\zeta}{dt} + \vec{k} \times \nabla \text{Tr} (G (x,y) \zeta) \cdot \nabla \zeta = 0$$

Using our definition for the Green’s function along with the definition of $G (x,y)$ $\zeta$ gives us the result that, if we define

$$\vec{T}_{m,n} \equiv \nabla G_{m,n}(x,y) \zeta_{n,m} \quad rm,n \equiv \sqrt{(x-x_m)^2 + (y-y_n)^2}$$

then $\vec{T}_{m,n} =$

$$\frac{1}{2\pi} \zeta (x_m,y_n) \left(\frac{1}{r_{m,n}^2} \right) \left((x-x_m) \hat{x} + (y-y_n) \hat{y}\right) \quad 1 < r_{m,n}$$

$$\frac{1}{\pi} \zeta (x_m,y_n) \left(1 + \log (r_{m,n})\right) \left((x-x_m) \hat{x} + (y-y_n) \hat{y}\right) \quad \epsilon < r_{m,n} < 1$$

$$0 \quad r_{m,n} < \epsilon$$

or, letting $\vec{T}_{m,n} = M_{m,n} \vec{d}_{m,n},$ where

$$\vec{d}_{m,n} = \left((x-x_m) \hat{x} + (y-y_n) \hat{y}\right)$$

and

$$M_{m,n} = \frac{1}{2\pi} \zeta (x_m,y_n) \left(\frac{1}{r_{m,n}} \right) \times \left\{ \begin{array}{ll}
\frac{1}{r_{m,n}} & 1 < r_{m,n} \\
2 (1 + \log (r_{m,n})) & \epsilon < r_{m,n} < 1 \\
0 & r_{m,n} < \epsilon
\end{array} \right. \quad (A.-14)$$
as a result the equation
\[
\vec{k} \times \vec{\nabla} \text{Tr} \left( \mathbf{G}(x,y) \zeta \right) = \hat{z} \times \vec{\nabla} \text{Tr} \left( \mathbf{G}(x,y) \zeta \right)
\]
\[
= \sum_{m=1}^{N_x} \sum_{n=1}^{N_y} M_{m,n} \left( \hat{z} \times \vec{d}_{m,n} \right)
\]
\[
= \sum_{m=1}^{N_x} \sum_{n=1}^{N_y} M_{m,n} \left( - (y - y_n) \hat{x} + (x - x_m) \hat{y} \right)
\]
(A.-14)

and so
\[
\vec{k} \times \vec{\nabla} \text{Tr} \left( \mathbf{G}(x,y) \zeta \right) \cdot \vec{\nabla} \zeta(x,y,t) =
\]
\[
\sum_{m=1}^{N_x} \sum_{n=1}^{N_y} M_{m,n} \left( - (y - y_n) \frac{\partial \zeta(x,y,t)}{\partial x} + (x - x_m) \frac{\partial \zeta(x,y,t)}{\partial y} \right)
\]

Finally, we can give an algorithm for our equation in the form required. Write the equation as
\[
\frac{d\zeta(x,y,t)}{dt} + \sum_{m=1}^{N_x} \sum_{n=1}^{N_y} M_{m,n} \left( - (y - y_n) \frac{\partial \zeta(x,y,t)}{\partial x} + (x - x_m) \frac{\partial \zeta(x,y,t)}{\partial y} \right) = 0 \quad (A.-14)
\]

So, at each gridpoint \(x_m, y_n, t_0\) let
\[
\frac{\partial \zeta(x_m, y_n, t_0)}{\partial x} = \frac{\zeta(x_{m+1}, y_n, t_0) - \zeta(x_{m-1}, y_n, t_0)}{x_{m+1} - x_{m-1}} \quad (A.-14)
\]
and
\[
\frac{\partial \zeta(x_m, y_n, t_0)}{\partial y} = \frac{\zeta(x_m, y_{n+1}, t_0) - \zeta(x_m, y_{n-1}, t_0)}{y_{n+1} - y_{n-1}} \quad (A.-14)
\]

So, for each point \(k, l\) on the grid
\[
0 = \frac{d\zeta(x_k, y_l, t_0)}{dt} + \sum_{m=1}^{N_x} \sum_{n=1}^{N_y} M_{m,n} \left( (x_k - x_m) \frac{\zeta(x_{k+1}, y_l, t_0) - \zeta(x_{k-1}, y_l, t_0)}{y_{l+1} - y_{l-1}} - (y_l - y_n) \frac{\zeta(x_k, y_{l+1}, t_0) - \zeta(x_k, y_{l-1}, t_0)}{x_{k+1} - x_{k-1}} \right)
\]

would be the time derivative and advective part of the equation. If we added all additional forces, including coriolis, frictional and external forcing, we would use this numerical formulation for computing the time derivative for each discrete point.
References


