Reconstructing 3D ocean temperature fields from real-time satellite and buoy surface measurements

by

Bianca Champenois

B.S., University of California, Berkeley (2020)

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

© Massachusetts Institute of Technology 2022. All rights reserved.
Reconstructing 3D ocean temperature fields from real-time satellite and buoy surface measurements

by

Bianca Champenois

Submitted to the Department of Mechanical Engineering on May 6, 2022, in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Abstract

Despite advancements in computational science, nonlinear geophysical processes still present important modeling challenges. Physical sensors (such as satellites, AUVs, or buoys) can collect data at specific points or regions, but are often scarce or inaccurate. Here, we present a framework to build improved spatiotemporal models that combine dynamics inferred from high-fidelity numerical models with measurements from sensors. Specifically, we are interested in ocean temperature which can serve as a useful indicator for ocean acidification, and we are motivated by a data set of sensor measurements only available at the surface of the ocean. We first apply standard principal component analysis (PCA) at every ocean surface coordinate to a numerical simulation of a 3D temperature field (reanalysis data) over time. For each horizontal location, the vertical structure of the field can be represented with just two PCA modes and their corresponding time coefficients, significantly reducing the dimensionality of the data. Next, a conditionally Gaussian model implemented through a temporal convolutional neural network (TCN) is built to predict the time coefficients of the PCA modes, as well as their variance, as a function of the surface temperature. The full 2D surface temperature field is estimated by a multi-fidelity Gaussian process regression scheme, for which the buoys have the highest fidelity and the satellite measurements have lower fidelity. The surface temperature is then inputted into the neural network to obtain probabilistic predictions for the PCA coefficients, which are used to stochastically reconstruct the full 3D temperature field. The techniques described provide a framework for building less expensive and more accurate models of conditionally Gaussian estimates for full 3D fields, and they can be applied to geophysical systems where data from both sensors and numerical simulations are available. We implement these techniques to estimate the full 3D temperature field of the Massachusetts and Cape Cod Bays, an area with a significant ocean economy. We compare the predictions with in-situ measurements at all depths. Finally, we discuss how the developed ideas can be leveraged to make more informed decisions about optimal in-situ sampling and path planning.
Thesis Supervisor: Themistoklis Sapsis
Title: Associate Professor
Acknowledgments

I started this project in the fall of 2020, in the thick of the pandemic. I did not meet my advisor or lab mates in person during the entire first year. I spent spring semester in Santa Barbara where I had nothing but my best friend and roommate, Katia Gibson, and the ocean. There was nothing for us to do other than work on problem sets, keep a six foot distance from people, and learn how to surf. I was taking courses in hydrodynamics, numerical fluids, and machine learning, and I was trying to figure out what exactly it was I wanted to get out of my research project and time in graduate school.

During the many repetitive days of pre-vaccine 2021, I read William Finnegan’s Barbarian Days: A Surfing Life, and I discovered a quote that motivated using machine learning to solve ocean problems and gave meaning and excitement to what I was working on:

The close, painstaking study of a tiny patch of coast, every eddy and angle, even down to individual rocks, and in every combination of tide and wind and swell—a longitudinal study, through season after season—is the basic occupation of surfers at their local break. Getting a spot wired—truly understanding it—can take years. At very complex breaks, it’s a lifetime’s work, never completed. This is probably not what most people see, glancing seaward, noting surfers in the water, but it’s the first-order problem that we’re out there trying to solve: What are these waves doing, exactly, and what are they likely to do next? Before we can ride them, we have to read them, or at least make a credible start on the job. Nearly all of what happens in the water is ineffable—language is no help. Wave judgment is fundamental, but how to unpack it? You’re sitting in a trough between waves, and you can’t see past the approaching swell, which will not become a wave you can catch. You start paddling upcoast and seaward. Why? If the moment were frozen, you could explain that, by your reckoning, there’s a fifty-fifty chance that the next wave will have a good takeoff spot about ten yards over and a little farther out from where you are now. This calculation is based on: your last two or three glimpses of the swells outside, each glimpse caught from the crest of a previous swell; the hundred-plus waves you have seen break in the past hour and a half; your cumulative experience of three or four hundred sessions at this spot, including fifteen or twenty days that were much like this one in terms of swell size, swell direction, wind speed, wind direction, tide, season,
and sandbar configuration; the way the water seems to be moving across the bottom; the surface texture and the water color; and, beneath these elements, innumerable subcortical perceptions too subtle and fleeting to express. These last factors are like the ones that the ancient Polynesian navigators relied upon when, on the open seas, they used to lower themselves into the water between the outriggers on their canoes and let their testicles tell them where in the great ocean they were.

In the following pages are my attempts at capturing the subcortical perceptions it takes to understand and model the intricacies of the ocean.

I would first like to thank my advisor, Professor Themis Sapsis, for supporting me these last two years. I am grateful for the complete patience and trust that he showed me throughout the entire process of writing this thesis. Without these, I would not have had the confidence to undertake such a project on my own.

I would also like to thank all of the other mentors and advisors who have supported me throughout my life: my high school teachers, Carol Evans, Adam Randall, and Greg Stoehr, and my undergraduate advisors, James Casey, Oliver O’Reilly, and Evan Variano.

Next, I thank my wonderful lab mates, current and past. I will list them here in order of having met them (on Zoom): Andreas Mentzelopoulos, Alexis Charalamposopoulos, Stephen Guth, Antoine Blanchard, Debbie Eelktink, Samuel Rudy, Ethan Pickering, Alireza Mojahed, Anthony Kriezis, Sualeh Khurshid, Brady Hammond, Antonio Naveira, and Ben Barthel. These people have only reaffirmed my beliefs that people, not work, make a workplace.

Finally, I dedicate this thesis to my family: my parents Vannina and Ludovic who have always encouraged me to pursue everything with complete dedication, and who instilled in me the curiosity and work ethic I needed to write this thesis; my sister Flora who is always the first to pick up the phone when I need something and the last to proofread my work for last-minute typos before an important deadline; and my brothers Elio and Lucas. And as always, Go Bears!
## Contents

1 Introduction ............................................. 13
   1.1 Nonlinear Geophysical Systems ......................... 13
   1.2 Massachusetts and Cape Cod Bays ....................... 13
   1.3 Contributions ......................................... 14

2 System and Data ........................................ 17
   2.1 Reanalysis Data ........................................ 17
   2.2 Sensor Measurements ................................... 18
      2.2.1 Satellite Measurements ........................... 18
      2.2.2 In-Situ Measurements .............................. 20

3 Framework ................................................. 23
   3.1 Representing the Vertical Structure with Principal Component Analysis 23
   3.2 Predicting PCA Coefficients from Surface Temperature with Machine Learning ............................................. 25
      3.2.1 Training ............................................ 27
      3.2.2 Parameter Selection: Number and Location of Input Points 28
      3.2.3 Parameter Selection: Memory Length Scale ............ 29
      3.2.4 Surface Temperature Constraint ...................... 30
      3.2.5 Results ............................................. 31
   3.3 Filling Gaps in the Sensor Data ........................ 34
      3.3.1 Gaussian Process Regression ........................ 34
      3.3.2 Hyperparameter Selection ........................... 36
List of Figures

1-1 Framework flow chart. .................................................. 15

2-1 Sigma layers of system at a latitude of 41.73°. .................. 19

2-2 3D temperature field from FVCOM reanalysis data on September 13th, 2012 at sigma level -0.5. ................................. 19

2-3 Satellite data. ............................................................. 20

2-4 In-situ sensors. ........................................................... 21

3-1 Characteristic segments of PCA decomposition for one (x,y) pair. ................................. 25

3-2 Results of 3D temperature field reconstruction with two PCA modes at three sigma layers on March 8th, 2012. ................. 26

3-3 Architecture of TCN: the residual blocks consist of a sequence of two causal convolutional layers with ReLU activation and a dropout layer. The dilation factor of each residual block is doubled at each depth. ................................. 27

3-4 Examples of different neighborhoods of points that are tested as input of neural network. .................................................. 29

3-5 TCN prediction of first and second PCA coefficients and mean temperature with the reanalysis data as input. The black lines delineate the training, validation, and test sets, respectively. ................................. 32

3-6 Full 3D temperature field reconstruction and standard deviation estimate from TCN predictions at three sigma levels for March 8th, 2012 with reanalysis data as input. .................................................. 33

3-7 Spatial mean over the whole spatial domain for each day. ............ 35
3-8 Results of extrapolating the available satellite and buoy data to obtain an estimate for surface temperature over the full domain. 38

3-9 First 9 eigenvectors from the covariance of surface temperature transformed into 2D matrices. 41

4-1 TCN prediction of first and second PCA coefficients and mean temperature with the real-time sensor surface temperature measurements as input. 44

4-2 3D temperature field reconstruction from real-time satellite measurement at three sigma levels for March 8th, 2016 and September 13th, 2016. 45

4-3 Comparison between buoy measurements and predictions from neural network at different depths for days with different amounts of satellite coverage. The red shading corresponds to the standard deviation of the absolute error. 46

4-4 SEAGLASS online data visualization platform. 47

6-1 Results of simulation in one dimension 53

6-2 Discretization error. 54

6-3 May 16th, 2016: example of a day for which conformal mapping would improve results. 55

B-1 Full 3D temperature field reconstruction and standard deviation estimate from TCN predictions at three sigma levels for September 13th, 2012 with reanalysis data as input. 64
List of Tables

3.1 Neural Network Model Evaluation ........................................... 31

6.1 Matrix Coefficients .................................................................... 53

A.1 MAE of Prediction of PCA Coefficients ...................................... 57
A.2 MNACC of Prediction of Standard Deviation of PCA Coefficients ... 57
A.3 MAE of Prediction of Standard Deviation of PCA Coefficients ..... 58
A.4 Effect of Memory Length Scale on First PCA Coefficient .......... 58
A.5 Effect of Memory Length Scale on Second PCA Coefficient ....... 59
A.6 Effect of Memory Length Scale on First PCA Coefficient Standard
  Deviation ................................................................................... 60
A.7 Effect of Memory Length Scale on Second PCA Coefficient Standard
  Deviation ................................................................................... 61
A.8 Effect of Surface Temperature Constraint ................................. 61
A.9 Model Evaluation (Real-Time Measurements) ............................. 62
Chapter 1

Introduction

1.1 Nonlinear Geophysical Systems

Environmental and geophysical systems can be modeled with nonlinear equations that typically require complex and computationally expensive numerical solvers. Even with highly accurate numerical methods, model errors still exist, and there can be losses of predictability due to intrinsic instabilities in the system. Such challenges can be mitigated by incorporating additional information about the system. Physical sensors can be used to collect the necessary supplemental information on quantities of interest. However, sensors only provide information about the system locally in space (e.g. buoys or drifters) or with a high degree of sparsity (e.g. satellite data). There is a need for improved data assimilation techniques that can estimate the state and uncertainty of a system in real time.

1.2 Massachusetts and Cape Cod Bays

To provide a specific example, we consider the temperature of the Massachusetts and Cape Cod Bays, an area with great biodiversity (fish, shellfish, whales, etc.) and significant fishing and tourism industries. The ability to predict temperature is helpful in tracking ocean health and ocean acidification properties [20]. Coastal waters are more susceptible to temperature rise and acidification because the influx of fresh-
water changes the composition of the ocean and reduces its buffering capacity, and ocean acidification has greater implications in coastal waters where most fisheries are located [18, 39, 12, 15]. Ocean temperature is governed by a set of high dimensional nonlinear equations. These nonlinear equations can, in principle, be solved to evaluate the temperature field using a numerical scheme, such as finite volumes, but such approaches are typically computationally demanding and need to be repeated for each new set of boundary, initial, and excitation conditions [23]. The temperature can also be determined using in-situ buoys and satellites. The buoys provide reliable measurements, but they are scarce. A satellite can cover the whole domain, but there are gaps in the data due to cloud coverage, and the measurements are less accurate. Most importantly, sensor measurements are only available at or near the surface of the ocean, leaving the bottom depths of the ocean unaccounted for [21, 25].

The goal of this work is to leverage data science techniques to characterize the vertical structure of the ocean temperature field from physics-based numerical simulations, and subsequently combine machine-learning methods with real-time sensor measurements of surface temperature to reconstruct and hindcast the full 3D temperature field and its uncertainties. We consider temperature data because it is readily available, and it is a useful indicator for ocean acidification, but the techniques discussed can be applied to other quantities of interest such as salinity, dissolved inorganic carbon (DIC), aragonite, and pH. We develop a framework that can build a full 3D model of ocean temperature from real-time satellite and buoy measurements of surface temperature, and estimate the uncertainty associated with both the system and the model.

1.3 Contributions

In this thesis, we introduce a framework to build improved models of nonlinear geophysical systems, such as 3D coastal ocean temperature fields, by taking advantage of data from multiple sources. We use a combination of data science techniques including principal component analysis (PCA), temporal convolutional neural networks
(TCN), Gaussian process regression (GPR), and optimal sampling. As a result, we develop a computationally inexpensive model of the temperature of the Massachusetts and Cape Cod Bays that leverages data from physics-based numerical solvers, buoys, and satellites to predict the temperature, and uncertainty of the model, at all points in the domain of interest. The model is also useful to make decisions about where and how to sample future data [37, 43] and to evaluate the quality of new sensors.

The framework is organized into multiple steps (Figure 1-1). The first two steps are independent. First, in step 1, we use multi-fidelity Gaussian process regression (GPR) to improve the sensor measurements of surface temperature by merging information from satellites and in-situ buoys. Next, in step 2, we use the reanalysis data to build a data-driven reduced order model and derive a functional relationship between 3D...
temperature and surface temperature; this connection is possible given the reduced-
order vertical structure of the problem that we obtain from principal component
analysis (PCA). Finally, in step 3, we input the real-time 2D surface temperature
measurements into the reduced-order model to obtain a real-time estimate for the 3D
temperature field. The framework can be modified or rearranged based on the type
and location of new data that become available.
Chapter 2

System and Data

Broadly, data assimilation is the process of combining information from different sources to approximate or analyze the state of a system of interest. Here, we combine information about temperature from physics-based numerical simulations and sensors to estimate the full 3D field in real time. Details about each of the data sets are described in the following sections.

2.1 Reanalysis Data

We start with reanalysis data consisting of a 3D temperature field over time for the Northeast Coastal Ocean from the FVCOM (Finite Volume Community Ocean Model) simulation from Chen et al. [9] which we use to learn the structure and dynamics of the system. Reanalysis data refer to data from a numerical simulation that integrates real world measurements in the computation. The model uses a fractional step method to solve the spatially and temporally evolving momentum equations for velocity, density, temperature, and salinity, among other variables. The simplified primitive equation for temperature in Cartesian coordinates is as follows

\[
\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} + w \frac{\partial T}{\partial z} = \frac{\partial}{\partial z} (K_h \frac{\partial T}{\partial z}) + F_T \tag{2.1}
\]
$K_h$ is the thermal vertical eddy diffusion coefficient, and $F_T$ captures the horizontal thermal diffusion. In general, mixing and diffusion are more dominant in the vertical direction. Estimates from this model were found to agree well with in-situ measurements with a root mean squared error of 2.28 °C [25]. We consider a truncated portion of the domain in the Massachusetts and Cape Cod Bays that consists of 45 sigma levels from January 2005 to December 2013 (nine years total). Here, a sigma level refers to a layer of the sigma coordinate system. In the sigma coordinate system, horizontal layers follow the model terrain, so for a given $(x,y)$ point, each horizontal layer has the same thickness [28]. The $z$ coordinate is defined by the following transformation

$$z(x, y, k, t) = \eta(x, y, t) + s(x, y, k, t)$$  \hspace{1cm} (2.2)$$

$$s(x, y, k, t) = \sigma(k)(H(x, y) + \eta(x, y, t))$$  \hspace{1cm} (2.3)$$

$H(x, y)$ is the depth of the bottom surface, and $\eta(x, y, t)$ is the height of the free surface, both relative to $z = 0$. $\sigma(k)$ varies from 0 to -1. The primitive equations are then rewritten using the new transformed coordinate system. This coordinate system is a convenient way to discretize the domain because it results in a continuous temperature field. As an example, a snapshot of the data from September 13th, 2012 at sigma level -0.5 is plotted in Figure 2-2. In this project, we use the data from the numerical simulation to build a neural network that is based on the vertical physics of the system.

2.2 Sensor Measurements

In addition to the data from the finite volume scheme, we have real-time surface temperature data from physical sensors which include satellites and in-situ stations.

2.2.1 Satellite Measurements

Satellites measure sea surface temperature by quantifying the energy of wavelengths coming from the ocean. Different satellites operate at varying resolutions and levels
Figure 2-1: Sigma layers of system at a latitude of 41.73°.

Figure 2-2: 3D temperature field from FVCOM reanalysis data on September 13th, 2012 at sigma level -0.5.

of accuracy [8], but the main challenge associated with using satellite data is that there are gaps due to cloud coverage. In this project, we have access to daily satellite
imagery from the MODerate-resolution Imaging Spectroradiometer (MODIS) Terra, but we see in Figure 2-3 that each day has a different amount of cloud coverage. Specifically, many days during winter months have no available measurements.

### 2.2.2 In-Situ Measurements

In contrast to satellites, in-situ stations are not affected by cloud coverage. Measurements are available from the Massachusetts Water Resources Authority (MWRA) (Figure 2-4b), but they are only collected on a monthly basis, and there are only 14 locations. The MWRA stations gather data by collecting samples of water at multiple depths and directly measuring the temperature. While this method is accurate, it is also very costly. We also have National Oceanic and Atmospheric Administration (NOAA) measurements from environmental monitors on lobster traps and large trawlers (eMOLT) (Figure 2-4c) and the Northeast Fisheries Science Center (NEFSC) (Figure 2-4d). Unlike for the MWRA measurements, there is less information about the accuracy and maintenance of the NOAA sensors. Consequently, we only use data from the MWRA stations to build our model, but the model can be augmented with any number of available sources of data. The in-situ measurements are also useful to
validate the quality of our model.
Chapter 3

Framework

3.1 Representing the Vertical Structure with Principal Component Analysis

We first apply principal component analysis (PCA) to the reanalysis data set to reduce the dimensionality of the data while retaining its patterns and information. Principal component analysis (PCA), also known as proper orthogonal decomposition (POD) or Karhunen–Loève decomposition, among other names, has long been used in many fields. In the context of fluid mechanics, weather prediction [26, 19], and oceanography, PCA extracts features or trends from large empirical data sets to accurately reconstruct the dynamics of the system using a small number of empirical orthogonal functions (EOF) and corresponding coefficients. Significant work has been done on the use of empirical orthogonal functions to reconstruct spatio-temporal sea surface temperature (SST) for which empirical measurements from sensors are available [13, 7, 17, 38]. In some cases, the basis is used to fill gappy data [14]. Here, we use PCA to extract the vertical structure, or 3D features, of existing reanalysis data. This process allows us to represent the vertical structure of the temperature field with just a few modes at each location of the ocean surface, and it can be proven that PCA results in an optimal orthogonal transformation that captures maximum variance. We are interested in the vertical structure of the temperature field because
most of the energy of the system is coming from solar radiative flux which is normal to
the surface of the ocean, and the vertical modes capture vertical mixing and diffusion.
At each \((x_i, y_i)\) pair, the temperature field is discretized into \(n\) depths and \(m\) time
steps.

\[
T_i = \begin{bmatrix}
T(z_1, t_1) & \ldots & T(z_1, t_m) \\
T(z_2, t_1) & \ldots & T(z_2, t_m) \\
\vdots & \ddots & \vdots \\
T(z_n, t_1) & \ldots & T(z_n, t_m)
\end{bmatrix}
\] (3.1)

\[
Z_i = T_i - \bar{T}_i
\] (3.2)

\[
C_i = \frac{1}{m}Z_iZ_i^T
\]

We can reconstruct \(Z_i\) with the eigenvectors \(\phi_{ij}\) of \(C_i\) and corresponding time
coefficients \(q_{ij}(t)\)

\[
Z_i(t) = \sum_{j=1}^{n} q_{ij}(t)\phi_{ij}
\] (3.3)

The time coefficients are calculated with the following inner product

\[
q_{ij}(t) = \langle Z_i(t), \phi_{ij} \rangle
\] (3.4)

and the full temperature field can be reconstructed with the original mean \(T_i\)

\[
T_{i,proj}(t) = \sum_{j=1}^{2} q_{ij}(t)\phi_{ij} + \bar{T}_i(t)
\] (3.5)

The eigenvalues obtained from the decomposition confirm that we have a low rank
problem as the first two modes capture more than 85% of the data’s energy and are
sufficient for reconstructing the temperature field (Figure 3-1). The spatial modes \(\phi_{ij}\)
represent the vertical structure of the field and are constant through time. The first
mode roughly corresponds to the thermocline. The time coefficients \(q_{ij}(t)\) and mean
temperature \(\bar{T}_i(t)\) vary with time and need to be determined for new time steps.
3.2 Predicting PCA Coefficients from Surface Temperature with Machine Learning

Next, we construct a neural network that predicts temperature over depth at each \((x_i, y_i)\) point as a function of surface temperature. We choose surface temperature as the input of the neural network because it is available from sensor measurements.
We also build a second neural network to predict the associated standard deviation and estimate the uncertainty of our system. Recent developments in machine learning have increased the popularity of using neural networks to model geophysical processes. A previous study used a neural network to predict subsurface temperature as a function of surface variables [2]. However, they did not perform data preprocessing to reduce the dimensionality of their outputs, nor did they predict the variance associated with the processes they were modeling. We specifically build a neural network that predicts the mean and standard deviation of the PCA coefficients $q_{ij}(t)$ and mean temperature $\bar{T}_i(t)$ obtained in the previous section. Many studies have been done on the use of neural networks to predict such time-varying PCA coefficients [30, 27, 29, 34, 42]. In this project, we build a temporal convolutional network (TCN), a type of convolutional network (CNN) that performs convolutions on one dimensional time series data. Unlike a traditional CNN, a TCN is causal which is useful for modeling dynamic systems [40]. TCNs have also been shown to outperform recurrent neural networks for sequence modeling [5, 1, 22]. As such, they are
creasingly being used in geophysical applications [41, 3, 16]. We adapt the Stochastic Machine Learning (SMaL) [40] code and retain the same residual block architecture (Figure 3-3). Each sequence spans 20 time steps with a stride of 1. The data are standardized before training for improved results. The batch size is 5, the filter width is 2, the dropout layer has a probability of 0.5, and the network has a depth of 6. The dilation factor is doubled at each depth.

Figure 3-3: Architecture of TCN: the residual blocks consist of a sequence of two causal convolutional layers with ReLU activation and a dropout layer. The dilation factor of each residual block is doubled at each depth.

3.2.1 Training

Typically, the weights of a neural network are obtained by minimizing a loss function that quantifies the error between the true data and the model predictions.

\[
J(\theta) = \frac{1}{T} \sum_{t=1}^{T} L(\hat{y}(\theta) - y)
\] (3.6)

Here, we build two neural networks at each \((x, y)\) point, one for the mean and one for standard deviation. We train these sequentially because we require the mean prediction to train the second neural network. Furthermore, we optimize different loss functions for each network. For the mean of the PCA coefficients, we minimize
the mean absolute error (MAE), a standard loss function for neural networks.

\[ J_{MAE} = \frac{1}{m} \sum_{i=1}^{m} |\hat{y} - y| \]  

(3.7)

For the standard deviation of the PCA coefficients, we minimize the mean negative anomaly correlation coefficient (MNACC). It is a correlation-based loss function, so it does not scale with magnitude, and it more strongly penalizes anomalies.

\[ J_{MNACC} = \frac{1}{m} \sum \frac{\sum (\hat{z} - [\hat{z}]) (z - [z])}{\sqrt{\sum (\hat{z} - [\hat{z}])^2} \sqrt{\sum (z - [z])^2}} \]  

(3.8)

\[ z = y - y^{ref} \]  

(3.9)

The reference \( y^{ref} \) is the cyclic mean, and for ocean temperature it corresponds to the annual variation due to seasons. Without a reference, this loss reduces to the Pearson correlation coefficient, another standard loss function in many machine learning applications.

### 3.2.2 Parameter Selection: Number and Location of Input Points

While the weights and biases can be found by optimizing a loss function, other parameters of the neural network need to be fine-tuned through discrete numerical experiments. For example, the choice of input points affects the output of the neural network. We know from the physics of the system (Equation 2.1) that the temperature gradients in the x and y direction also contribute to the vertical temperature profile. As such, we include neighboring points in the input of the neural network to produce a non-local parametrization. We perform numerical experiments to find the number and location of input points that result in the lowest testing error and are best suited for generalizability. We perform these tests on three \((x, y)\) pairs in the domain, denoted A, B, and C in Figure 3-4, and we adopt the same parameters for the models of all other \((x, y)\) pairs. We first test the neural network with one, two, three, and
five input points. Then, we experiment with the distance between the input points and the point of interest. From the results of these numerical experiments (Tables A.1, A.2, A.3), we build the inputs of the TCN with the surface temperature of four additional nearby points for which the distance is between nine and ten kilometers. This distance, obtained from numerical experiments, makes physical sense because it corresponds to submesoscale processes of the ocean [6].

### 3.2.3 Parameter Selection: Memory Length Scale

The temporal convolutional network also has parameters associated with the dynamics in time. Starting with timeseries arrays of surface temperature, $T_S$, PCA coefficients, $q_1$ and $q_1$, and mean temperature, $\bar{T}$,

\[
x = \begin{bmatrix} T_S(t_0) & T_S(t_1) & T_S(t_2) & \ldots & T_S(t_n) \end{bmatrix}
\]

\[
y = \begin{bmatrix} q_1(t_0) & q_1(t_1) & q_1(t_2) & \ldots & q_1(t_n) \\
q_2(t_0) & q_2(t_1) & q_2(t_2) & \ldots & q_2(t_n) \\
\bar{T}(t_0) & \bar{T}(t_1) & q_2\bar{T} & \ldots & \bar{T}(t_n) \end{bmatrix}
\]

Figure 3-4: Examples of different neighborhoods of points that are tested as input of neural network.
we build matrices of smaller sequences on which we apply the convolutional filter.

\[
x_{TCN} = \begin{bmatrix} T_S(t_0) & T_S(t_1) & \cdots & T_S(t_m) \\ T_S(t_s) & T_S(t_{s+1}) & \cdots & T_S(t_{m+1}) \\ \vdots & \vdots & \ddots & \vdots \\ T_S(t_{n-m+1}) & \cdots & \cdots & T_S(t_n) \end{bmatrix}
\]

(3.12)

\[
y_{TCN} = \begin{bmatrix} q_1(t_1) & q_1(t_2) & \cdots & q_1(t_m) \\ q_1(t_s) & q_2(t_{s+1}) & \cdots & q_1(t_{m+1}) \\ \vdots & \vdots & \ddots & \vdots \\ q_1(t_{n-m+1}) & \cdots & \cdots & q_1(t_n) \end{bmatrix}
\]

(3.13)

When building these smaller sequences, we have the ability to choose how much data to use which affects the performance of the neural network. The sampling rate determines how many time steps to skip within an input time series, the stride, \( s \), determines how many time steps to skip between each successive time series, and the memory length scale, \( m \), determines how many points back in time to consider in one time series. Again, we perform numerical experiments to find the values for these parameters that result in the lowest testing error (Tables A.4, A.5, A.6, A.7). The memory length scale is set to be twenty days, and the sampling rate and stride are both set to one day. In our final model, each PCA coefficient is predicted using the surface temperature from all of the data from the twenty previous days, which shows agreements with ocean time scales for upwelling and eddies [6].

### 3.2.4 Surface Temperature Constraint

The output of the neural network is used to reconstruct the full 3D temperature field, but we want to ensure that the prediction at the surface of the ocean matches the input surface temperature which is assumed to be exact.

\[
q_1 \phi_1(z = 0) + q_2 \phi_2(z = 0) + \bar{T} = T(z = 0)
\]

(3.14)
This requirement can be written as a constraint function

\[ f(\hat{y}(\theta)) = q_1\phi_1(z = 0) + q_2\phi_2(z = 0) + \bar{T} - T(z = 0) \quad (3.15) \]

Finally, we embed the soft constraint \( \lambda |f(\hat{y}(\theta))| \) into the loss function

\[ J(\theta) = \frac{1}{T} \sum L(\hat{y}(\theta) - y) + \lambda |f(\hat{y}(\theta))| \quad (3.16) \]

From numerical experiments (Table A.8), we find that the neural network is able to match the surface temperature without the soft constraint. The constraint does not significantly improve the results.

### 3.2.5 Results

By using additional nearby points and previous time steps, we create a non-local parametrization in both space and time. The neural network is built using four years of data for training, two years for validation, and three years for testing. The error associated with the neural network prediction are calculated relative to both the original reanalysis data and the PCA reconstruction (Table 3.1). The raw outputs of the neural network are simply the PCA coefficients and mean temperature. However, these raw outputs can be combined with the PCA modes to reconstruct the full 3D temperature field (Figure 3-6). For each \((x, y)\) pair, it takes one minute to train a neural network on a standard CPU. Once the neural network is fully optimized, it only takes a few seconds to make a prediction.
Figure 3-5: TCN prediction of first and second PCA coefficients and mean temperature with the reanalysis data as input. The black lines delineate the training, validation, and test sets, respectively.
Figure 3-6: Full 3D temperature field reconstruction and standard deviation estimate from TCN predictions at three sigma levels for March 8th, 2012 with reanalysis data as input.
3.3 Filling Gaps in the Sensor Data

As described in Section 2.2.1, satellites can provide useful information about surface temperature, but they are significantly affected by cloud coverage. Work has been done to improve measurements from satellites and to blend data from multiple satellites [10, 45]. In many projects, in-situ buoy measurements are used to either validate or improve the accuracy of models [44, 11, 24, 35, 36, 46, 8]. One recent approach that has been shown to obtain quick, accurate, and useful results is Gaussian process regression (GPR) [4, 33]. GPR is not ideal because the matrix inversion becomes slow for large numbers of input points. However, GPR is convenient for problems with a low number of input points because it produces an estimate for uncertainty which can be used for active sampling. Furthermore, unlike with other machine learning techniques, the hyperparameters of the model, specifically those of the kernel, have an intuitive physical meaning and can be set according to properties of the system. Here, we use GPR to extrapolate the available data. The features of the model are the longitude, latitude, and time, and the value that is being predicted is the surface temperature. For points at which sensor data are available, we keep the original data, but for points at which there are no measurements, we predict the temperature using nearby points both in time and space.

3.3.1 Gaussian Process Regression

The kernel $K$ relates all of the data points, and the mean and variance are predicted from the following equations. The mean prediction is

$$\bar{f}_* = m(X_*) + K(X_*, X)[K(X, X) + \sigma^2_n I]^{-1}(y - m(X)) \quad (3.17)$$

and the variance is:

$$cov(f_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma^2_n I]^{-1}K(X, X_*) \quad (3.18)$$
For our application, the mean function \( m(X_s) \) is explicitly set to be the spatial mean (Figure 3-7) of the available satellite data. To avoid inverting large matrices, we keep the size of the kernel small by building a new GPR model for each time step. The input data consists of the available data on the day of interest, data from one day before and one day after. In other words, we only use data from three days to predict the surface temperature for one day, and we repeat this process for all time steps. The features for each time step \( k \) are

\[
\begin{bmatrix}
  x_i & y_i & t_{k-1} \\
  x_i & y_i & t_k \\
  x_i & y_i & t_{k+1}
\end{bmatrix}
= \begin{bmatrix}
  x_i & y_i & -1 \\
  x_i & y_i & 0 \\
  x_i & y_i & 1
\end{bmatrix}
\]

(3.19)

where \((x_i, y_i)\) are all of the available spatial points at each time step \( k \).
3.3.2 Hyperparameter Selection

For the kernel, we use the radial basis function (RBF) with automatic relevance determination (ARD) as the covariance function.

\[
cov(f(x_p), f(x_q)) = k(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2}(x_q - x_p)^T \Theta (x_q - x_p))
\] (3.20)

The automatic relevance determination chooses different characteristic lengthscales for each input to determine the relevant inputs. The signal variance \(\sigma_f\) and characteristic lengthscales \(\Theta\) are hyperparameters of the model. The characteristic lengthscale represents how far apart two points need to be for their function values to become uncorrelated. The inverse of the lengthscale represents how relevant a given feature is. There are three characteristic lengthscales: one for the input longitude, one for the input latitude, and one for the input time. The noise variance, \(\sigma_n\), is not a parameter of the kernel, but it can also be considered one of the hyperparameters of the whole system. This parameter assumes that we know the uncertainty of the sensors.

Typically, the hyperparameters are found by optimizing the following loss function

\[
\log p(y|X) = -\frac{1}{2} y^T (K + \sigma_n^2 I)^{-1} y - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi
\] (3.21)

However, to avoid overfitting and to generalize the models, we manually set the same hyperparameters for all days, changing only the training data for each day. For days with no available training data, we take the average over 10 days (5 previous and 5 following days). For the spatial lengthscales, we choose values of 0.25 degrees or 25 kilometers, which is equivalent to six “gridpoints” or “pixels,” where one gridpoint is the spatial granularity. This choice assigns more weight to spatial points that are within 25 kilometers of the point of interest; it corresponds to the mesoscales of the ocean [6]. For the time lengthscale, we set the hyperparameter to one day. Finally, we choose to set the noise variance to \(\sigma_n = 0.1\), and we set the signal variance to \(\sigma_f = 0.3\) by taking the average of minimizing the objective function over all models.
3.3.3 Multi-Fidelity Gaussian Process Regression

We improve the model by incorporating the buoy data, which has a lower uncertainty than the satellite data, through a multi-fidelity Gaussian process regression scheme. The satellite and buoy measurements are shown to have good agreement [4]. Given $s$ levels of fidelity, the model with the lowest fidelity is denoted with $x_1, y_1, \bar{f}_1$, and the model with the highest fidelity is denoted with $x_s, y_s, \bar{f}_s$ [32]. The prediction for the model with the lowest fidelity follows the Gaussian process regression steps from equations (3.17, 3.18)

$$\bar{f}_1(x_*) = K(X_*, X_1)[K(X_1, X_1) + \sigma_n I]^{-1}y_1$$  \hspace{1cm} (3.22)

with covariance

$$\text{cov}(\bar{f}_1) = K(X_*, X_*) - K(X_*, X_1)[K(X_1, X_1) + \sigma_n I]^{-1}K(X_1, X_*)$$  \hspace{1cm} (3.23)

Each following model will be of the form

$$\bar{f}_t(x_*) = \rho_{t-1}\bar{f}_{t-1} + \delta_t \hspace{1cm} t = 2, \ldots, s$$  \hspace{1cm} (3.24)

In this project, there are only two levels of fidelity, so the prediction for the highest level of fidelity, $s = 2$, can be computed with the following equation

$$\bar{f}_2(x_*) = \rho\bar{f}_1(x_*) + \mu_d + K(X_*, X_2)[K(X_2, X_2) + \sigma_n I]^{-1}(y - \rho\bar{f}_1(x_2) - \mu_d)$$  \hspace{1cm} (3.25)

Its corresponding covariance is

$$\text{cov}(\bar{f}_2) = \rho^2\text{cov}(\bar{f}_1) + K(X_*, X_*) - K(X_*, X_2)[K(X_2, X_2) + \sigma_n I]^{-1}K(X_2, X_*)$$  \hspace{1cm} (3.26)

$\rho$ and $\mu_d$ are hyperparameters that are different for each level of fidelity. We could, however, augment the model with additional levels of fidelity if new sensors become available. Like $\sigma_f$ and $\Theta$ of the covariance function, $\rho$ and $\mu_d$ can be chosen through
maximum likelihood estimation or other optimization techniques. We use the Emukit [31] Python package, which builds on the GPy Python package, to build the multi-fidelity model. Such techniques have already been used to predict surface temperature, but our model differs with respect to the choice of input points [4]. Babaee et al. used all of the available data to build a model while we only use spatial points from three time steps. Because we use less data at each time step, our model is faster at making predictions, and therefore more practical for real-time modeling. For consistency, we set $\rho$ and $\mu_d$ to be the same as those from the optimized model in Babaee et. al.

### 3.3.4 Results

(a) High cloud coverage: March 8th, 2016.

(b) Minimal cloud coverage: September 13th, 2016.

Figure 3-8: Results of extrapolating the available satellite and buoy data to obtain an estimate for surface temperature over the full domain.

The results of the extrapolation are plotted both for a day with high cloud coverage (Figure 3-8a) and for a day with minimal cloud coverage (Figure 3-8b). The uncertainty of the extrapolation is higher in regions with significant cloud coverage. Overall, the results from using just three days show good agreement with those from
Babae et al., and the new model is much faster.

### 3.3.5 Generating Random Realizations

As a final step, we use the estimate for surface temperature to generate many randomly perturbed realizations, and these are used to test the sensitivity of the neural network to different inputs. For each day, the GPR model generates a prediction for the temperature at each grid point

\[
\mathbf{T}_{\text{matrix}} = \begin{bmatrix}
T(x_1, y_1) & \cdots & T(x_1, y_n) \\
T(x_2, y_1) & \cdots & T(x_2, y_n) \\
\vdots & \ddots & \vdots \\
T(x_m, y_1) & \cdots & T(x_m, y_n)
\end{bmatrix}
\]  

(3.27)

We first turn the matrix into a vector of size \( p = m \times n \) where each row corresponds to a coordinate pair \( \mathbf{x} = \{x, y\} \):

\[
\mathbf{T} = \begin{bmatrix}
T(x_1, y_1) \\
T(x_1, y_2) \\
\vdots \\
T(x_k, y_l) \\
\vdots \\
T(x_n, y_m)
\end{bmatrix} = \begin{bmatrix}
T(x_1) \\
T(x_2) \\
\vdots \\
T(x_j) \\
\vdots \\
T(x_p)
\end{bmatrix}
\]  

(3.28)

We calculate the covariance between all points in the domain.

\[
C(\mathbf{x}_1, \mathbf{x}_2) = K(\mathbf{x}_1, \mathbf{x}_2) - K(\mathbf{x}_1, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1}K(\mathbf{X}, \mathbf{x}_2)
\]  

(3.29)

For days with no available satellite data, the covariance simply becomes the kernel of the model

\[
C(\mathbf{x}_1, \mathbf{x}_2) = K(\mathbf{x}_1, \mathbf{x}_2)
\]  

(3.30)
Then, we divide the covariance matrix by its trace

$$\tilde{C} = \frac{C}{tr(C)}$$  \hspace{1cm} (3.31)

We find the eigenvectors $v_i$ and eigenvalues $\lambda_i$ of $\tilde{C}$. Then, we turn the eigenvectors of size $p$ back to matrices with dimensions $m \times n$.

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_j \\ \vdots \\ v_p \end{bmatrix} = \begin{bmatrix} v_{1,1} \\ v_{1,2} \\ \vdots \\ v_{k,l} \\ \vdots \\ v_{m,n} \end{bmatrix}$$  \hspace{1cm} (3.32)

$$V_{matrix} = \begin{bmatrix} v_{1,1} & \cdots & v_{1,n} \\ v_{2,1} & \cdots & v_{2,1} \\ \vdots \\ v_{m,1} & \cdots & v_{m,n} \end{bmatrix}$$  \hspace{1cm} (3.33)

For the first 5 eigenvectors, we compute a random number from a normal distribution with variance $\lambda_i^2$

$$\theta_i \sim \mathcal{N}(0, \lambda_i^2)$$  \hspace{1cm} (3.34)

which we use to generate random realizations of the temperature field

$$\tilde{T} = T + \sum_{n=1}^{5} \theta_i V_i$$  \hspace{1cm} (3.35)

We repeat this process of computing random numbers 20 times to generate 20 random realizations for each day, and these are used as inputs of the TCN to make predictions for the full 3D temperature field. By looking at the standard deviation of all of the predictions for the 20 inputs, we can verify that our model generalizes well, and we take the mean of the 20 predictions as the final prediction.
Figure 3-9: First 9 eigenvectors from the covariance of surface temperature transformed into 2D matrices.
Chapter 4

Results

4.1 3D Temperature Field Reconstruction

We input the real-time estimate for surface temperature into the neural network to obtain the PCA coefficients and mean temperature for the day of interest. When estimating surface temperature, we left out measurements from three stations (N04, F13, F29 Figure 2-4b) which we saved for validation. These stations also collect measurements for temperature over multiple depths, which we divide into shallow (0-25m), medium (25-45m), and deep (>45m). We also have NOAA measurements from environmental monitors on lobster traps and large trawlers (eMOLT Figure 2-4c) and the Northeast Fisheries Science Center (NEFSC Figure 2-4d). We validate the results of our full model by comparing the predictions from the neural network to the in-situ measurements.

4.2 Validation

For the MWRA measurements, which are the most reliable in-situ measurements, we find that our model performs similarly to other state of the art methods. The mean absolute error of our predictions is $1.37^\circ$C, and 79% of predictions fall within two degrees of the truth. We also observe that the model performs best for days with the most amount of available satellite data (80-100%). We found more significant
Figure 4-1: TCN prediction of first and second PCA coefficients and mean temperature with the real-time sensor surface temperature measurements as input.
Figure 4-2: 3D temperature field reconstruction from real-time satellite measurement at three sigma levels for March 8th, 2016 and September 13th, 2016.
errors when we applied our analysis to the eMOLT and NEFSC data sets (Figure 4-3); some of the errors seem to be associated with system trends which require further investigation (e.g. condition of sensors, calibration, etc.). We find that there

Figure 4-3: Comparison between buoy measurements and predictions from neural network at different depths for days with different amounts of satellite coverage. The red shading corresponds to the standard deviation of the absolute error.

are no significant improvements from including the buoy measurements. However, the framework allows us to seamlessly incorporate data from multiple sources which could be useful in applications where data are more sparse. Furthermore, the framework provides an estimate for uncertainty given the level of accuracy of each sensor.
4.3 SEAGLASS Integration

We integrated our model into the MIT Sea Grant SEAGLASS online geospatial visualization platform. The web application takes daily surface temperature measurements from NASA satellites, and inputs them into our trained neural networks. It then reconstructs and maps the full 3D temperature field. The data can be downloaded for analysis, and the platform is available to researchers, stakeholders, and other members of the community.

Figure 4-4: SEAGLASS online data visualization platform.
Chapter 5

Conclusions and Future Work

5.1 Conclusion

We introduced a practical method, based on new machine learning techniques and previously run traditional numerical simulations, to quickly and inexpensively reconstruct 3D ocean temperature fields from real-time sensor measurements of surface temperature. Such a model has many uses, ranging from monitoring general ocean health for fisheries to understanding changes in ocean acidification. Our framework also estimates the variance associated with the system, which can be leveraged for active sampling. We compared the results from our framework to in-situ measurements, and we found that the error associated with our predictions is comparable to that of other state of the art models. We also demonstrated how our model can evaluate the quality of sensor measurements from different sources. Finally, we incorporated our model into a web-based application for anyone to access.

5.2 Future Work

In the future, we plan to use our model to make decisions about where to place additional sensors, a process often referred to as active sampling or optimal sampling. Because our model produces an estimate for variance, we can define an acquisition function based on uncertainty, and we can optimize this acquisition function to choose
which points to sample from next. A common choice for optimal sampling is the uncertainty sampling acquisition function which chooses points in the domain that are associated with the highest amount of uncertainty. However, other acquisition functions can be defined to identify points that are more likely to occur or points that result in extreme events, and acquisition functions can be weighted to prioritize sensors that are less expensive to employ.
Chapter 6

Miscellaneous

6.1 FVCOM Error Analysis

To better understand the numerical model that we are using (FVCOM) and its associated errors, we repeated the simulation for a simplified case. FVCOM is in 3D and solves equations for velocity, temperature, salinity, and density among other variables, while our simulation is in 1D and only considers temperature. A more complete understanding of numerical error is useful in estimating the overall uncertainty of the model.

6.1.1 Fractional Step Method

Like FVCOM, we solve the equation for temperature (Equation 2.1) using a similar fractional step method. We use a constant value for the thermal diffusivity, and we set the boundary condition to be a simple sine function. First, we explicitly solve for the “transition” temperature $T^*$ using only the advection term. Then, we implicitly find the true temperature $T^{m+1}$ by balancing the change in the transition temperature with the diffusion term.

$$\frac{\partial T}{\partial t} + \frac{\partial w T}{\partial z} = \frac{\partial}{\partial z}(K \frac{\partial T}{\partial z})$$  (6.1)

$$\frac{\partial T}{\partial t} + \text{Advection} = \text{Diffusion}$$  (6.2)
\[(T^n)^* = T^n - (\text{Advection}^n)\Delta t \tag{6.3}\]

\[T^{n+1} = (T^n)^* + (\text{Diffusion}^{n+1})\Delta t \tag{6.4}\]

We experiment with different schemes, and we evaluate the model at different resolutions to estimate the discretization error.

### 6.1.2 Advection: Finite Volume QUICK Scheme

For the advection term, we use the finite volume QUICK (quadratic upstream interpolation for convective kinematics) scheme.

\[\phi = wT \tag{6.5}\]

\[
\phi_e = \begin{cases} 
\frac{6}{8}\phi_P + \frac{3}{8}\phi_E - \frac{1}{8}\phi_W & \text{if } \vec{v} \cdot \hat{n} > 0 \\
\frac{6}{8}\phi_W + \frac{3}{8}\phi_P - \frac{1}{8}\phi_W W & \text{if } \vec{v} \cdot \hat{n} < 0 
\end{cases} \tag{6.6}\]

\[
\phi_w = \begin{cases} 
\frac{6}{8}\phi_E + \frac{3}{8}\phi_P - \frac{1}{8}\phi_E E & \text{if } \vec{v} \cdot \hat{n} > 0 \\
\frac{6}{8}\phi_P + \frac{3}{8}\phi_W - \frac{1}{8}\phi_E & \text{if } \vec{v} \cdot \hat{n} < 0 
\end{cases} \tag{6.7}\]

\[
\Delta z \frac{\partial T^*}{\partial t} + \phi_e - \phi_w = 0 \tag{6.8}\]

\[(T^n)^* = T^n \frac{\Delta t}{\Delta z} ((wT)_e - (wT)_w) \tag{6.9}\]

### 6.1.3 Diffusion: Implicit Finite Volume Central Difference Scheme

For the diffusion term, we use an implicit finite volume central difference scheme.

\[
\Delta z \frac{\partial T^{**}}{\partial t} = K \left( \frac{\partial T}{\partial z} \right)_e - K \left( \frac{\partial T}{\partial z} \right)_w \tag{6.10}\]

\[
\Delta z \frac{T_k^{n+1} - (T_k^n)^*}{\Delta t} = K \frac{T_{k+1} - T_k}{\Delta z} - K \frac{T_k - T_{k-1}}{\Delta z} \tag{6.11}\]

\[
(-K \frac{\Delta t}{\Delta z^2})T_{k-1}^{n+1} + (1 + 2K \frac{\Delta t}{\Delta z^2})T_k^{n+1} + (-K \frac{\Delta t}{\Delta z^2})T_{k+1}^{n+1} = (T_k^n)^* \tag{6.12}\]
Table 6.1: Matrix Coefficients

<table>
<thead>
<tr>
<th></th>
<th>(\alpha)</th>
<th>(\beta)</th>
<th>(\gamma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((-K \frac{\Delta t}{\Delta z^2}))</td>
<td>((1 + 2K \frac{\Delta t}{\Delta z^2}))</td>
<td>((-K \frac{\Delta t}{\Delta z^2}))</td>
<td></td>
</tr>
</tbody>
</table>

\[
A = \begin{bmatrix}
\frac{7}{4} & -1 & \frac{1}{4} & 0 & \ldots & 0 \\
\alpha & \beta & \gamma & 0 & \ldots & 0 \\
0 & \alpha & \beta & \gamma & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \ddots \\
0 & \ldots & 0 & \alpha & \beta & \gamma \\
0 & \ldots & 0 & \frac{1}{4} & -1 & \frac{7}{4}
\end{bmatrix}
\]

(6.13)

\[
AT_{k+1} = (T^n_k)^*
\]

(6.14)

The boundary conditions are handled with a one-sided fourth order finite difference scheme. We use a constant grid size for both terms.

Figure 6-1: Results of simulation in one dimension

(a) FVCOM  (b) Simple 1D Model

6.1.4 Error Analysis

Given that the finite volume schemes used are second order, the discretization error from the finite volume scheme can be estimated by comparing the solution at different resolutions.

\[
\varepsilon_{\Delta x} \approx \frac{T_{\Delta x} - T_{2\Delta x}}{2^p - 1}
\]

(6.15)
6.2 Choice of Kernel (Gaussian Process Regression)

In this thesis, we used the standard radial basis function (RBF) for the kernel. This choice is convenient for optimal sampling because the gradients are simple to compute. However, other functions could have been chosen to improve other aspects of the model. These alternatives are described in the following sections.

6.2.1 Periodicity

For models with time as an input, a periodic kernel can be used to capture seasonal changes.

\[
\text{cov}(f(x_p), f(x_q)) = k(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{2\sin^2(\pi|x_p - x_q|/p)}{\theta^2}\right) \quad (6.16)
\]

Such a periodic kernel is convenient for models with data that spans multiple years. The kernel does not treat summer measurements (high temperature) differently from winter measurements (low temperature). Furthermore, the periodic kernel can use information from previous years to forecast temperature in future years.
6.2.2 Conformal Mapping

The standard radial basis function (Equation 6.16) assumes that points in the domain are distributed according to a standard Cartesian coordinate system. It does not take the boundaries into account which means that the distance between points in the water that are separated by land is the traditional Euclidean distance. A new kernel function can be defined that uses conformal mapping to assign distances between points according to physical boundaries. For the purpose of this project, we ignore this inconsistency and use the standard kernel function. Figure 6-3 demonstrates an example of a day for which conformal mapping would improve the accuracy of the model around the Cape Cod peninsula.

Figure 6-3: May 16th, 2016: example of a day for which conformal mapping would improve results.
## Appendix A

## Tables

### Table A.1: MAE of Prediction of PCA Coefficients

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCA 1</td>
<td>PCA 2</td>
</tr>
<tr>
<td>One Input</td>
<td>0.3288</td>
<td>0.6946</td>
</tr>
<tr>
<td>Five Inputs</td>
<td>0.3349</td>
<td>0.6759</td>
</tr>
<tr>
<td>Five Inputs (farther)</td>
<td>0.2576</td>
<td>0.6947</td>
</tr>
<tr>
<td>Five Inputs (even farther)</td>
<td>0.3417</td>
<td>0.6346</td>
</tr>
</tbody>
</table>

### Table A.2: MNACC of Prediction of Standard Deviation of PCA Coefficients

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STD 1</td>
<td>STD 2</td>
</tr>
<tr>
<td>One Input</td>
<td>0.9875</td>
<td>1.1337</td>
</tr>
<tr>
<td>Five Inputs</td>
<td>0.5292</td>
<td>1.0173</td>
</tr>
<tr>
<td>Five Inputs (farther)</td>
<td>0.6312</td>
<td>0.9509</td>
</tr>
<tr>
<td>Five Inputs (even farther)</td>
<td>1.1461</td>
<td>0.8744</td>
</tr>
</tbody>
</table>
Table A.3: MAE of Prediction of Standard Deviation of PCA Coefficients

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STD 1</td>
<td>STD 2</td>
<td>STD 1</td>
</tr>
<tr>
<td>One Input</td>
<td>0.1399</td>
<td>0.7840</td>
<td>0.1594</td>
</tr>
<tr>
<td>Five Inputs</td>
<td>0.2558</td>
<td>0.6168</td>
<td>0.3760</td>
</tr>
<tr>
<td>Five Inputs (farther)</td>
<td>0.2292</td>
<td>0.2249</td>
<td>0.3050</td>
</tr>
<tr>
<td>Five Inputs (even farther)</td>
<td>0.1071</td>
<td>0.3293</td>
<td>0.2845</td>
</tr>
</tbody>
</table>

Table A.4: Effect of Memory Length Scale on First PCA Coefficient

<table>
<thead>
<tr>
<th></th>
<th>Point A</th>
<th>Point B</th>
<th>Point C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory (days)</td>
<td>Stride</td>
<td>MNACC</td>
<td>MSE</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.9082</td>
<td>0.3042</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.9359</td>
<td>0.3201</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.9204</td>
<td>0.3600</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.9246</td>
<td>0.3623</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.9563</td>
<td>0.2333</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.9460</td>
<td>0.3134</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.9558</td>
<td>0.2250</td>
</tr>
<tr>
<td>Memory (days)</td>
<td>Stride</td>
<td>MNACC</td>
<td>MSE</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.8975</td>
<td>0.3156</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.9293</td>
<td>0.1809</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.9061</td>
<td>0.2800</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.9273</td>
<td>0.2166</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.9471</td>
<td>0.1941</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.9367</td>
<td>0.3094</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.9501</td>
<td>0.1952</td>
</tr>
<tr>
<td>Memory (days)</td>
<td>Stride</td>
<td>MNACC</td>
<td>MSE</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.8848</td>
<td>0.4047</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.9026</td>
<td>0.3235</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.8950</td>
<td>0.3362</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.9090</td>
<td>0.3684</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.9421</td>
<td>0.1593</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.9281</td>
<td>0.2360</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.9448</td>
<td>0.1779</td>
</tr>
</tbody>
</table>
Table A.5: Effect of Memory Length Scale on Second PCA Coefficient

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.4245</td>
<td>0.8236</td>
<td>0.6429</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.4595</td>
<td>0.8002</td>
<td>0.6196</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.1850</td>
<td>0.9724</td>
<td>0.6989</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>-0.0740</td>
<td>1.0125</td>
<td>0.7100</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.4566</td>
<td>0.7854</td>
<td>0.6441</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.3747</td>
<td>0.9471</td>
<td>0.6991</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.5764</td>
<td>0.6808</td>
<td>0.5928</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.1812</td>
<td>0.7778</td>
<td>0.5944</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.0504</td>
<td>0.7956</td>
<td>0.5959</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.1769</td>
<td>0.7714</td>
<td>0.5810</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.1951</td>
<td>0.7725</td>
<td>0.5917</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.3595</td>
<td>0.7923</td>
<td>0.6144</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.2663</td>
<td>0.7673</td>
<td>0.5971</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.4509</td>
<td>0.6761</td>
<td>0.5656</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.2313</td>
<td>0.7066</td>
<td>0.6090</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.4155</td>
<td>0.6843</td>
<td>0.6137</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.2484</td>
<td>0.6954</td>
<td>0.6106</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.4093</td>
<td>0.8342</td>
<td>0.6819</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.4550</td>
<td>0.6686</td>
<td>0.6102</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.4212</td>
<td>0.7193</td>
<td>0.6288</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.4508</td>
<td>0.6935</td>
<td>0.6105</td>
</tr>
</tbody>
</table>
Table A.6: Effect of Memory Length Scale on First PCA Coefficient Standard Deviation

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.7670</td>
<td>18.7508</td>
<td>4.1266</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.7734</td>
<td>3.8974</td>
<td>1.8979</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.7736</td>
<td>1.8286</td>
<td>1.2282</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.7697</td>
<td>0.7193</td>
<td>0.8109</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.7745</td>
<td>0.2969</td>
<td>0.4438</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.7840</td>
<td>0.0424</td>
<td>0.1803</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>-0.3470</td>
<td>2.4578</td>
<td>1.3820</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>-0.3548</td>
<td>0.2260</td>
<td>0.3813</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>-0.4559</td>
<td>0.1513</td>
<td>0.3154</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>-0.4181</td>
<td>0.2473</td>
<td>0.4406</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>-0.4346</td>
<td>0.0958</td>
<td>0.2720</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.5783</td>
<td>0.0571</td>
<td>0.1890</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.6318</td>
<td>6.8042</td>
<td>1.9560</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.6317</td>
<td>2.1668</td>
<td>1.0168</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.5761</td>
<td>2.4006</td>
<td>1.1407</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.6122</td>
<td>0.7943</td>
<td>0.8551</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.5470</td>
<td>0.3126</td>
<td>0.5192</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.6218</td>
<td>0.0579</td>
<td>0.1981</td>
</tr>
</tbody>
</table>
Table A.7: Effect of Memory Length Scale on Second PCA Coefficient Standard Deviation

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.8637</td>
<td>179.7003</td>
<td>12.7936</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.8754</td>
<td>38.6846</td>
<td>5.9577</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.8243</td>
<td>31.5875</td>
<td>5.3498</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.8703</td>
<td>8.7933</td>
<td>2.8038</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.8631</td>
<td>3.3528</td>
<td>1.7346</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.8789</td>
<td>0.0990</td>
<td>0.2727</td>
</tr>
</tbody>
</table>

Point B

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.8664</td>
<td>238.5451</td>
<td>14.5463</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.8583</td>
<td>66.8761</td>
<td>7.7286</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.8667</td>
<td>26.0307</td>
<td>4.6910</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.8029</td>
<td>9.6708</td>
<td>2.9657</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.8750</td>
<td>4.2984</td>
<td>1.8322</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.9069</td>
<td>0.5606</td>
<td>0.5758</td>
</tr>
</tbody>
</table>

Point C

<table>
<thead>
<tr>
<th>Memory (days)</th>
<th>Stride</th>
<th>MNACC</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>0.7803</td>
<td>212.4202</td>
<td>13.6823</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.7839</td>
<td>50.2144</td>
<td>6.6258</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0.7651</td>
<td>24.5355</td>
<td>4.7300</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0.7895</td>
<td>8.5675</td>
<td>2.7450</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>0.7889</td>
<td>3.1728</td>
<td>1.6905</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.8094</td>
<td>0.4012</td>
<td>0.5071</td>
</tr>
</tbody>
</table>

Table A.8: Effect of Surface Temperature Constraint

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>MAE</th>
<th>MSE</th>
<th>MAE</th>
<th>$T$</th>
<th>MSE</th>
<th>MAE</th>
<th>$T_s$</th>
<th>MSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>yes</td>
<td>0.256</td>
<td>0.166</td>
<td>0.941</td>
<td>0.972</td>
<td>0.144</td>
<td>0.065</td>
<td>0.035</td>
<td>0.002</td>
</tr>
<tr>
<td>0.1</td>
<td>yes</td>
<td>0.300</td>
<td>0.180</td>
<td>0.930</td>
<td>0.970</td>
<td>0.196</td>
<td>0.071</td>
<td>0.153</td>
<td>0.065</td>
</tr>
<tr>
<td>0.5</td>
<td>yes</td>
<td>0.366</td>
<td>0.241</td>
<td>0.898</td>
<td>0.955</td>
<td>0.248</td>
<td>0.110</td>
<td>0.0177</td>
<td>0.0015</td>
</tr>
<tr>
<td>1</td>
<td>no</td>
<td>0.799</td>
<td>0.589</td>
<td>0.851</td>
<td>0.929</td>
<td>0.876</td>
<td>0.462</td>
<td>0.003</td>
<td>0.00007</td>
</tr>
</tbody>
</table>
Table A.9: Model Evaluation (Real-Time Measurements)

<table>
<thead>
<tr>
<th>data</th>
<th>target</th>
<th>output</th>
<th>MAE (°C)</th>
<th>RMSE (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>full field</td>
<td>all data</td>
<td>single fid. TCN</td>
<td>2.03</td>
<td>2.58</td>
</tr>
<tr>
<td>full field</td>
<td>all data</td>
<td>multi-fid. TCN</td>
<td>2.07</td>
<td>2.61</td>
</tr>
<tr>
<td>shallow (0-24m)</td>
<td>all data</td>
<td>single fid. TCN</td>
<td>2.13</td>
<td>2.78</td>
</tr>
<tr>
<td>shallow (0-24m)</td>
<td>all data</td>
<td>multi-fid. TCN</td>
<td>2.19</td>
<td>2.82</td>
</tr>
<tr>
<td>medium (25-44m)</td>
<td>all data</td>
<td>single fid. TCN</td>
<td>2.17</td>
<td>2.58</td>
</tr>
<tr>
<td>medium (25-44m)</td>
<td>all data</td>
<td>multi-fid. TCN</td>
<td>2.17</td>
<td>2.57</td>
</tr>
<tr>
<td>deep (&gt;45m)</td>
<td>all data</td>
<td>single fid. TCN</td>
<td>1.54</td>
<td>1.88</td>
</tr>
<tr>
<td>deep (&gt;45m)</td>
<td>all data</td>
<td>multi-fid. TCN</td>
<td>1.61</td>
<td>1.94</td>
</tr>
<tr>
<td>low coverage (0-33%)</td>
<td>all data</td>
<td>single fid. TCN</td>
<td>1.80</td>
<td>2.30</td>
</tr>
<tr>
<td>med. cov. (33-66%)</td>
<td>all data</td>
<td>single fid. TCN</td>
<td>2.48</td>
<td>3.09</td>
</tr>
<tr>
<td>high cov. (66-100%)</td>
<td>all data</td>
<td>single fid. TCN</td>
<td>2.40</td>
<td>2.89</td>
</tr>
<tr>
<td>low coverage (0-33%)</td>
<td>MWRA</td>
<td>single fid. TCN</td>
<td>1.37</td>
<td>1.93</td>
</tr>
<tr>
<td>med. cov. (33-66%)</td>
<td>MWRA</td>
<td>single fid. TCN</td>
<td>1.32</td>
<td>1.86</td>
</tr>
<tr>
<td>high cov. (66-100%)</td>
<td>MWRA</td>
<td>single fid. TCN</td>
<td>1.52</td>
<td>2.18</td>
</tr>
<tr>
<td>full field</td>
<td>MWRA (val.)</td>
<td>single fid. TCN</td>
<td>1.13</td>
<td>1.46</td>
</tr>
<tr>
<td>shallow (0-24m)</td>
<td>MWRA (val.)</td>
<td>single fid. TCN</td>
<td>1.43</td>
<td>1.97</td>
</tr>
<tr>
<td>medium (25-44m)</td>
<td>MWRA (val.)</td>
<td>single fid. TCN</td>
<td>1.39</td>
<td>1.95</td>
</tr>
<tr>
<td>deep (&gt;45m)</td>
<td>MWRA (val.)</td>
<td>single fid. TCN</td>
<td>1.44</td>
<td>1.89</td>
</tr>
<tr>
<td>low coverage (0-33%)</td>
<td>MWRA (val.)</td>
<td>single fid. TCN</td>
<td>1.17</td>
<td>1.68</td>
</tr>
<tr>
<td>med. cov. (33-66%)</td>
<td>MWRA (val.)</td>
<td>single fid. TCN</td>
<td>2.23</td>
<td>2.71</td>
</tr>
<tr>
<td>high cov. (66-100%)</td>
<td>MWRA (val.)</td>
<td>single fid. TCN</td>
<td>1.15</td>
<td>1.51</td>
</tr>
<tr>
<td>full field</td>
<td>NEFSC</td>
<td>single fid. TCN</td>
<td>2.03</td>
<td>2.33</td>
</tr>
<tr>
<td>shallow (0-24m)</td>
<td>NEFSC</td>
<td>single fid. TCN</td>
<td>1.77</td>
<td>2.17</td>
</tr>
<tr>
<td>medium (25-44m)</td>
<td>NEFSC</td>
<td>single fid. TCN</td>
<td>2.51</td>
<td>2.69</td>
</tr>
<tr>
<td>deep (&gt;45m)</td>
<td>NEFSC</td>
<td>single fid. TCN</td>
<td>1.92</td>
<td>2.19</td>
</tr>
<tr>
<td>low coverage (0-33%)</td>
<td>NEFSC</td>
<td>single fid. TCN</td>
<td>1.65</td>
<td>1.86</td>
</tr>
<tr>
<td>med. cov. (33-66%)</td>
<td>NEFSC</td>
<td>single fid. TCN</td>
<td>2.89</td>
<td>3.23</td>
</tr>
<tr>
<td>high cov. (66-100%)</td>
<td>NEFSC</td>
<td>single fid. TCN</td>
<td>2.65</td>
<td>2.82</td>
</tr>
<tr>
<td>full field</td>
<td>EMOLT</td>
<td>single fid. TCN</td>
<td>2.32</td>
<td>2.95</td>
</tr>
<tr>
<td>shallow (0-24m)</td>
<td>EMOLT</td>
<td>single fid. TCN</td>
<td>2.68</td>
<td>3.31</td>
</tr>
<tr>
<td>medium (25-44m)</td>
<td>EMOLT</td>
<td>single fid. TCN</td>
<td>2.17</td>
<td>2.63</td>
</tr>
<tr>
<td>deep (&gt;45m)</td>
<td>EMOLT</td>
<td>single fid. TCN</td>
<td>0.97</td>
<td>1.20</td>
</tr>
<tr>
<td>low coverage (0-33%)</td>
<td>EMOLT</td>
<td>single fid. TCN</td>
<td>2.21</td>
<td>2.81</td>
</tr>
<tr>
<td>med. cov. (33-66%)</td>
<td>EMOLT</td>
<td>single fid. TCN</td>
<td>2.39</td>
<td>3.13</td>
</tr>
<tr>
<td>high cov. (66-100%)</td>
<td>EMOLT</td>
<td>single fid. TCN</td>
<td>2.62</td>
<td>3.18</td>
</tr>
</tbody>
</table>
Appendix B

Figures
Figure B-1: Full 3D temperature field reconstruction and standard deviation estimate from TCN predictions at three sigma levels for September 13th, 2012 with reanalysis data as input.
Bibliography


