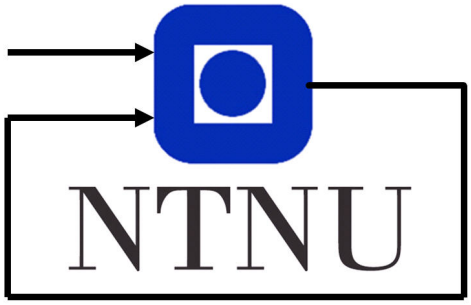
Data-driven Current Model as Part of a Digital Twin for Submarine Drones

Carsten Antonio Bing

June 2022



Department of Engineering Cybernetics

Acknowledgment

I would like to thank iDROP AS and NTNU for starting the project and allowing me to contribute. I would also like to thank Morten Omholt Alver, Daniel Ørnes Halvorsen from NTNU, and Finn Are Michelsen from SINTEF for their contributions and guidance. This help has been much appreciated, and this report would not have been the same without you.

Secondly, I would like to thank my friends in Trondheim, who have supported me academically and socially throughout my studies here. Through a time when newspaper headlines mainly were regarding a virus, you kept me sane.

Lastly, a thank you to my family who have always been there to support me no matter my decision.

C.A.B.

Abstract

Underwater drones have historically not been able to use full ocean simulations due to requiring extensive processing power. This project develops a cheap data-driven ocean model simulating data from SINTEF's ocean model SINMOD using Radial Basis Functions (RBFs) and a Gaussian Kernel. The RBFs in this project combines sines and cosines to model tidal dynamics and Laguerre polynomials to model non-tidal dynamics. Initial parameterization and reparameterization are completed in Python using the MOSEK Optimization Suite software package. The ocean model should be able to take measurements at points in the grid and adjust its parameters through a Kalman filter. The objective of the model, according to SINTEF, is a 2.5m horizontal landing accuracy at 1000m depth.

Testing showed that the RBFs chosen could follow the flow velocity at the RBF points accurately and for the whole field with a second optimization. A Kalman filter was implemented to adjust for incoming measurements. Three case studies were explored; one static measurement, measurements exceeding RBF points, and a realistic dynamic drone scenario. These showed that distance from measurement to RBF point was proportional to the change. The results from these highlighted the importance of measurement position. There was an issue of convergence for the cases, speculated to be caused by available processing power or setup of the reparameterization problem. The dynamic case showed how points closer to the measurement drones were more prone to adjustment than those further away.

The analysis completed in this paper lays the groundwork for current prediction in underwater drones. However, for the method to be used efficiently, issues of forecasting and convergence will need to be resolved.

Sammendrag

Undervannsdroner har historisk sett ikke vært i stand til å bruke fulle havsimuleringer på grunn av at de krever omfattende prosessorkraft. Dette prosjektet utvikler en enkel datadrevet havmodell som simulerer data fra SINTEFs havmodell SINMOD ved hjelp av Radial Basis Functions (RBFs) og en Gaussian Kernel. RBF-ene i dette prosjektet kombinerer sinus og cosinus for å modellere tidevannsdynamikk og Laguerre-polynomer for å modellere ikke-tidevannsdynamikk. Initiell parameterisering og optimalisering gjennomføres i Python ved å bruke programvarepakken MOSEK Optimization Suite. Havmodellen skal kunne ta målinger på punkter i rutenettet og justere sine parametere gjennom et Kalman-filter. Målet med modellen, ifølge SINTEF, er en 2,5m horisontal landingsnøyaktighet på 1000m dybde.

Testing viste at de valgte RBF-ene kunne følge strømningshastigheten ved RBF-punktene nøyaktig og for hele feltet med en andre optimalisering. Et Kalman-filter ble implementert for å justere for innkommende målinger. Tre casestudier ble utforsket; én statisk måling, når antall målinger overskrider RBF-punkter og et realistisk dynamisk dronescenario . Disse viste at avstanden fra måling til RBF-punkt var proporsjonal med endringen. Resultatene fra disse fremhevet viktigheten av målepunkt posisjon. Det var et problem med konvergens for casene, antatt å være forårsaket av tilgjengelig prosessorkraft eller oppsett av reparameteriseringsproblemet. Den dynamiske casen viste hvordan punkter nærmere måledronene var mer utsatt for justering enn de som var lenger unna.

Analysen fullført i denne artikkelen legger grunnlaget for havstrømningsprediksjon i undervannsdroner. For at metoden skal kunne brukes effektivt, må problemer med prognoser og konvergens imidlertid løses.

Contents

CHAPTER 1

Introduction

Exploring and analyzing large areas of the ocean floor has been a challenge for many years. Ocean Bottom Nodes (OBNs) has recently emerged as a practical approach. OBNs are underwater drones that perform seismic analysis of the seabed through external sensors. iDrop A/S is a company that works with the Oceanid™ as an R&D initiative funded by the European Commission through their Horizon 2020 program [1]. Their drone works through gravity-assisted navigation, dropping it in the place of interest and steered down to the intended location. Upon completing the sensor readings, the drone returns safely to the surface using a patented neutral salt-slurry release mechanism.

This thesis aims to create and test a simplified current ocean model to aid navigation to and from the seabed. Furthermore, the model should be able to take in sensor data to adjust the simplified model through a Kalman filter. Previously, Oceanid drone trajectory used SINTEF's SINMOD ocean model to generate currents. The simplified model in this project uses these currents as initialization and should be able to predict future currents with the help of new measurements and a Kalman filter. SINTEF states that the goal for the OBNs is to have a horizontal location accuracy of 2.5m at a depth of 1000m. The specific operational area is an arbitrary 50×50 km^2 grid.

1.1 Background

Why this project?

As mentioned in the previous section, the Oceanid™ underwater drones can measure seismic reading through a gravity-assisted system. However, when not having actuators in an underwater drone, the accuracy of positioning and overview of surrounding currents is critical. These are attributes that require expensive and heavy equipment to obtain. Mounting this type of equipment to the OCEANID™ drone could lead to higher costs for the drone and increased complexity due to additional weight. A solution was proposed with SINTEF Ocean and NTNU to implement a more time-efficient, data-driven model to predict future currents. The model uses Gaussian RFB to estimate flow velocities at specific grid points. It would also implement traditional cybernetic methods such as the Kalman filter to take in sensor readings and update the model parameters. Since the model estimates flow

velocity only within specific grid points and interpolate the points in between, it uses much less computing power.

The scope of the OBN project can be seen in figure 1. The OCEANID drone is divided into three sections: Navigation and control, acoustic measurement, and guidance. Navigation and control describe the physical systems on the drone, such as the actuators and control systems. Secondly, acoustic measurement creates an acoustic positioning system using sensor data and a previously developed hydro acoustic model. Lastly, the guidance system takes in the acoustic measurements, feeds them into a current model, which is developed in this thesis, and forwards it to an OCEANID model, which gives the OBN position references.

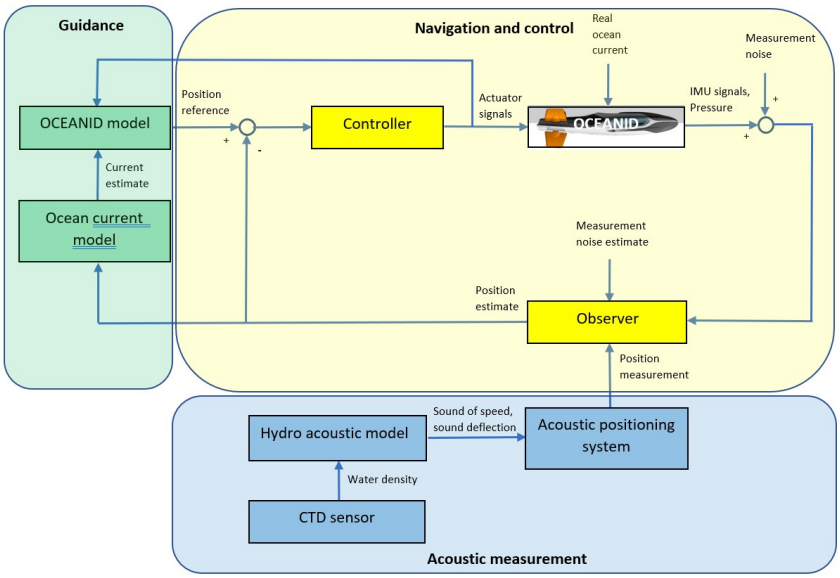


Figure 1: OCEANID Drone Three Main Components: Navigation and Control, Aucustic Measurement and Guidance

Related work

A great help in this thesis has been a research paper from 2014 written by researchers at the Georgia Institute of Technology. In it, ocean currents were estimated using RFB and a Gaussian kernel. Its purpose was to represent high-resolution ocean current model estimations at a low computational cost. The method was further analyzed to implement a Kalman filter and change the grid analysis center. This problem is hence similar to this paper.

The data set used in this project is taken from a SINTEF-made ocean modeling system called SINMOD, briefly introduced in section 2.1.5. However, the model is extensive and requires further reading for complete understanding. The data was used to initialize the model and to create testing data sets.

A 2D time-varying model was created and analyzed during the fall semester of 2021 in a specialization project leading to this thesis using sinusoidal and constant functions as Gaussian RFBs. These analyses provide the groundwork for this thesis and contain valuable hyper-parameter analyses, which have proven helpful when developing the method. Furthermore, it helped the author understand Gaussian radial basis functions and the scaling of the Gaussian kernel.

What remains to be done?

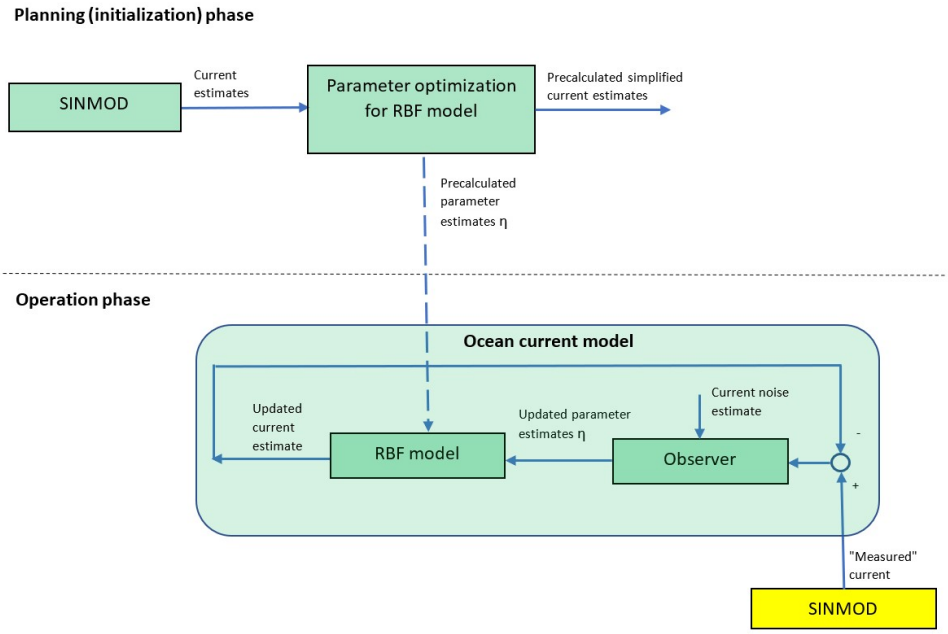


Figure 2: Detailed Explaination of Ocean Current Model from Figure 1

The ocean current model block seen in figure 1 is expanded in figure 2. It contains a planning- and operation phase. The planning phase is an initialization of a model which outputs simplified current estimates and precalculated parameters for future estimations based on SINMOD current data. The operational phase takes measurements, runs through a Kalman filter, and creates updated parameters η at the specific grid points. These updated parameters can then be fed into the RBF model, which updates

the current estimates. The main purpose of this project is being able to estimate future flow estimations in order for the OCEANID drone to adjust its trajectory. Analyses must be completed on the accuracy of the initial model, reparameterized model, and the Kalman filter adjustments.

1.2 Problem Formulation

A computationally cheap data-driven ocean model shall be developed as part of a digital twin for navigation of submarine drones from the sea surface to target positions at the sea floor. Full ocean models (such as the SINMOD model developed at SINTEF) are too computationally heavy to be used efficiently in real-time high resolution current estimation, so a combined approach using simplified models is needed. In a specialization project in the fall of 2021, a 2D time-varying model has been set up based on a parameterization of the flow field using Gaussian radial basis functions (RBFs) and periodic functions, and initialized using data computed by the full ocean model SINMOD. In this master project, the model will be further developed and tested. The concrete tasks of the thesis are:

1. Establish simplified 3D time-varying model with current components u and v, and including a combination of tidal and non-tidal variable terms.

2. Implement model correction based on measurements using a Kalman filter

3. Test model and model correction using a test data set from SINMOD, including comparison of the accuracy of a static vs. a dynamic model

1.3 Outline

The report structure following this section is explained in table 1:

Section	Title	Content
2	Underlying Theory	Explains the general idea and theory of current dynamics, flow simulations, relevant underwater drones, RBF functions and Kalman filtering. This prepares the reader for further analysis in the results and discussion part of the paper.
3	Previous Work	Goes through work done in the previous semester leading to the research completed in this thesis.
4	Method	Starting with an overview of the data set and variables used. Afterward shows how the problems were analyzed and which resources were used to achieve this goal. It ends with cases used in simulations.
5	Results	Results from section 4 is presented.
6	Discussion	From the results, different points of interest are discussed in detail.
7	Conclusion	Based on experimentation, a conclusion is made upon the valuable parts of the paper and achieved goals.
8	Recommendations for Further Work	Further elaboration from the discussion on what specific points require more work. Creates suggestions for further analyses and points of interest.

Table 1: Section Overview

CHAPTER 2

Theory

2.1 Current Dynamics and SINMOD

In order to get a proper understanding of what the simplified model will simulate, we first have to understand the underlying theory. This subsection goes through introductory hydrodynamics and current theory to get a foundation to understanding the SINMOD model.

2.1.1 Tides and Tidal Constituents

Tides are defined as the rise and fall of sea surface levels caused by the combined gravitational forces of the moon and the sun [2]. This motion can decompose into a series of sinusoidal signals at different frequencies called *tidal constituents*, each representing the effects of the sun, the moon, or an interaction between the two. Tidal constituents can be identified by astronomy, and data collection [3]. Although there are hundreds of predictable periodic motions between the earth, moon, and sun, 37 of these typically have the most significant effect on the tidal movement. The five most common tidal constituents at the data set location are used for the sinusoidal functions in this paper and are taken from the National Oceanic and Atmospheric Administration (NOAA) [4]. Their values and description can be shown in table 2.

Name	Frequency	Description
M2	28.984104	Principal lunar semidiurnal constituent
S2	30.0	Principal solar semidiurnal constituent
N2	28.43973	Larger lunar elliptic semidiurnal constituent
K1	15.041069	Lunar diurnal constituent
M4	57.96821	Shallow water overtides of principal lunar constituent

Table 2: NOAA Frequency Table

2.1.2 Internal dynamics

When looking at an infinitesimal element of water (Figure 3), we can model three primary forces: Gravitational (F_g), pressure (F_p), and shear (F_s.)

Gravity acts on the mass of the water, which generates a gravitational force. The pressure force acts on all four sides of the 2-dimensional element, and its net effect depends on the pressure gradient. Shear will occur when one side of the element moves faster. This shear generates shear stress multiplied by the occurring area to find the shear force. In addition to this, the element will experience electromagnetic forces due to particle interaction, centrifugal forces due to the rotation of the element, and Coriolis forces due to the earth's rotation. These, along with the gravitational force, are called body forces, while the pressure and shear forces are called surface forces.

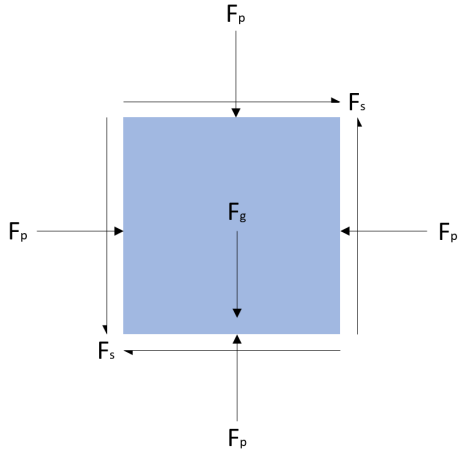


Figure 3: Force Diagram for Internal Dynamics

We assume an incompressible fluid and constant viscosity to obtain the incompressible Navier-Stokes equation with constant viscosity. An incompressible fluid means that the density does not change with pressure. Viscosity is defined as the resistance of a fluid to change in shape [5]. It comes from Newton's second law of motion $\vec{F} = m\vec{a}$ and ensures conservation of momentum in its solutions. The forces in figure 3 combined with the density give the acceleration of the fluid. The acceleration due to the pressure forces is represented using a pressure gradient. The vertical pressure gradient balances the gravitational force, and the shear force is represented using a diffusion term.

$$\underbrace{\frac{D\vec{V}}{Dt}}_{\text{Total Derivative}} - \underbrace{\mu\nabla^2\vec{V}}_{\text{Diffusion Term}} = \underbrace{-\frac{1}{\rho}\nabla p}_{\text{Pressure Gradient}} + \underbrace{\vec{g}}_{\text{Body Force}} \tag{1}$$

The total derivative is a composition of the change of velocity of the particle with respect to time and the rate of change of the position of a particle within a fluid flow, also called the convection term. The first of

these two is usually described as a Lagrangian view, while the latter is called an Eulerian view. The transition to an Eulerian view causes the expression to be nonlinear. If considering Coriolis forces, one must include a Coriolis term in the expression. The Navier-Stokes equation is an essential building block for the SINMOD simulation.

2.1.3 Density distribution

Density is described as mass over volume. Seawater density will not be the same everywhere we look. Factors such as temperature, salinity, and pressure create an uneven density distribution and can be challenging to model accurately. Freshwater run-offs, external temperature variations, geographical location, and depth are just a tiny portion of all factors. Whenever density is modeled incorrectly, the internal dynamics calculations will not represent the actual dynamics, making our model inaccurate. In the Navier Stokes equation (equation 1), inaccuracies in density distributions will affect the pressure gradient, which will consequently lead to an inaccurate reading of the particle acceleration and current dynamics.

2.1.4 Atmospheric forcing

The environment surrounding the ocean is also essential since it affects the boundary conditions. The sun and its accessibility to the ocean will heat up and cool down water which will cause evaporation and condensation. Wind will create shear stresses at the surface, which will affect current gradients deeper into the ocean. Rain primarily affects the salinity of the water at the surface. Differences in weather could lead to a varying freshwater run-off due to increased or decreased melting of ice and water.

2.1.5 SINMOD

SINMOD is an ocean simulation system continuously developed and maintained by SINTEF since the early 80s [6]. It models multiple different variations in ocean dynamics as well as biological processes. Its hydrodynamic simulation is based on solving simplified Navier-Stokes equations using the finite difference method on an Arakawa C-grid. The model is commonly used to model flow fields off the coast of Norway and can model down to a 32m spatial resolution. The SINMOD model creates open boundary conditions through a hydrodynamic model together with a nested model [7]. The boundary conditions are implemented using a flow relaxation scheme between these open- and nested boundaries. Atmospheric forcing, freshwater run-off, density, tidal forcing, and currents are all inputs to the model as atmospheric input.

SINMOD is used in this project to initialize our modeling functions and generate noisy data used as measurements. Ideally, one would like to use actual measurements, but these were not available.

2.2 OCEANID™ - Underwater Drone

Based on an underwater drone patent from 2012, iDrop A/S participated in the EU Horizon 2020 program with the project – Oceanid™. The system under development implements installing thousands of unique sensors via submarine drones to predefined locations on the seafloor for seismic monitoring. When approaching the bottom, the node uses gravity-assisted navigation with the help of its attached fins. Once readings finish, a ballast is released to create uplift [8]. The nodes can then be retrieved at the surface again. The project includes multiple disciplines such as engineering cybernetics, underwater acoustics, communications, and physical oceanography.

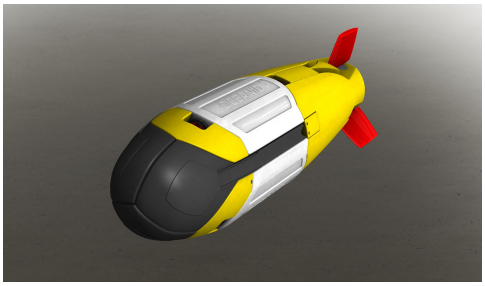


Figure 4: OCEANID™Underwater Drone Illustration [9]

The drones have multiple measurement systems, including depth, IMU, and compass measurements. There is also an acoustic measurement device in a planned 1 out of 9 drones that can measure current velocity during travel. The drones will communicate via acoustic distance sensors between themselves and vessels at the sea surface. These measurements will aid the drone in accurate positioning. The release of the drones will happen via a surface vessel, and nine drones will be released every 15 minutes. Typical distances between drones are 200-1500 m at an operational area of $50 \times 50 \text{km}^2$. Descent speeds are 5 m/s, while ascend speeds are 3.5 m/s. As mentioned in section 1, the goal of each OCEANID underwater drone is to have a horizontal accuracy of 2.5 m at 1000 m depth. This accuracy primarily comes into play at the seabed, where measurements are made. If the drones are positioned incorrectly, the readings taken will also be inaccurate. As seen in figure 1, the ocean current model created in this project is one of many components meaning there can be many pitfalls. The ocean current model aims to look at the correlation between prediction and measurements and try to adjust the underwater drone trajectory based on this.

2.3 Radial Basis Functions

An efficient way of approximating functions with many variables is by using radial basis functions (RBF). The method has emerged as a valuable tool within machine learning as far back as 1988 [10] for functional interpolation and acts as a simplified neural network. The essence of the method is that the current point we would like to approximate is influenced more by nearby points than by faraway points. The standard form is [11]:

$$h(x_m) = \sum_{n=1}^{N} \xi_n \underbrace{\phi\|x_m - x_n\|}_{\text{Radial Basis Function}} \tag{2}$$

Where x_m is defined as the current point of analysis and ξ_n is the value at specified RBF points. A visual explanation of this can be seen in figure 5, where each RBF point has an estimated value ξ_n, and the drone creates a new value h by multiplying the radial basis function ϕ with the estimated value. The values of ϕ are displayed as thicker arrows for closer RBF points.

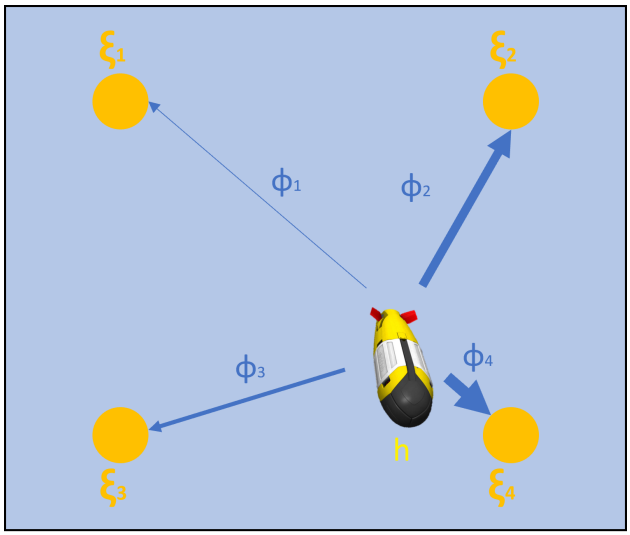


Figure 5: Values at RBF Points (Yellow, ξ) and Contributions From the RBF (Blue Arrows, ϕ), Thicker Arrows Indicating Higher Values

The RBF points do not necessarily need to be equidistant but are both in this project and example. RBF gets its name due to its two components: Radial, which refers to the Euclidean distance, and the basis function, which refers to the function used in estimation. In most cases the basis function contains a Gaussian kernel ($\exp(-\gamma\|x_m - x_n\|^2)$, where γ is a tuned hyper-parameter) due to its trade-off between accuracy and smoothness in comparison to other basis functions [12]. Expanding on equation 2, with h_n being

the RBF and ξ_n the value at the RBF point, we can portray it in matrix form:

$$\underbrace{\begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix}}_{\mathbf{H}} = \underbrace{\begin{bmatrix} \phi_1(\|x_1 - x_1\|) & \dots & \phi_n(\|x_1 - x_n\|) \\ \vdots & \ddots & \vdots \\ \phi_1(\|x_m - x_n\|) & \dots & \phi_n(\|x_m - x_n\|) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{bmatrix}}_{\Xi} \tag{3}$$

If Φ is invertible, the RBF constants Ξ are obtained by taking $\Phi^{-1}\mathrm{H}$. Alternatively, an optimization technique could be utilized. The algorithm at work can be seen in figure 6, where a Gaussian RBF reconstructs an image which sustains data loss. The data is reconstructed based on nearby points.

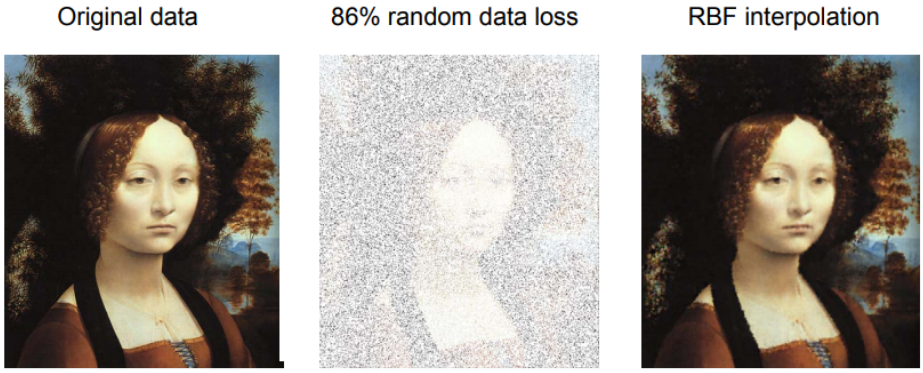


Figure 6: Image reconstruction using RBF [13]

2.4 Temporal and Spatial Basis Functions

As seen in the previous section, the distance from which we take measurements gives different actuation factors. This actuation factor also need an accurate estimation at the RBF point. A complex system can be modeled more accurately by implementing multiple different, less complex subfunctions to describe the system's dynamics further. The functions used in this case are temporal and spatial basis functions, meaning that they vary depending on time and position. After generation, these multiply with the RBF functions. There are plenty of ways to represent physical systems; however, we will try to generate functions that are relevant to the dynamics mentioned in 2.1.

2.4.1 Sinusoidal Functions

Perhaps the most famous of all basic functions, sinusoidal functions are commonly used to model physical phenomena. This project will take into consideration sine- and cosine functions:

$$K\sin(\omega t+\phi), \qquad K\cos(\omega t+\phi) \tag{4}$$

Where K, ϕ, and ω are scalar values that decide the amplitude, phase shift, and frequency. These are extremely useful when modeling tidal dynamics since the motion will be sinusoidal. The hyper-parameters, in this case, will be the amplitude, frequency, and phase shift. Tidal behavior can be modeled by summing up sinusoidal functions using the tidal constituents mentioned in 2.1.1 with the correct amplitude and phase shift.

2.4.2 Laguerre Polynomials

This set of functions is a solution to the Laguerre (differential) equation. In this project, a sum of different degrees of Laguerre polynomials are scaled to model non-tidal dynamics. The standard form is:

$$L_n^{\alpha}(x) = \frac{x^{-\alpha}e^{x}}{n!}\frac{d^n}{dx^n}(x^{\alpha+n}e^{-x}), \quad n=0,1,2,\dots \tag{5}$$

For these simulations, a function called `genlaguerre` from the Python library Scipy will be used to generate these functions.

2.5 Kalman Filter and Underlying Theory

Rudolf Emil Kalmán invented Kalman filtering to reduce noisy sensor data during the Apollo missions using the power of Bayesian probability [14]. Today, Kalman filters are standard practice in many cybernetic applications, from robots to chemical plants. The following section introduces the α-β filter (also called the g-h filter), the discrete Bayes filter, and the concept of Gaussians before finishing with how a Kalman filter is set up. The intention is to create a clear understanding of the underlying concepts leading up to the Kalman filter. The theory is based on the book *Kalman and Bayesian Filters in Python*.

2.5.1 The α - β Filter

The α - β filter derives from trust in either measurement or estimation. We have to have a physical model and sensor data to implement the filter.The model could for example be physical laws of how the system should behave, such as the speed of a car based on engine output. The problem is that the car might not have this ideal value due to external factors such as friction or drag. Sensor data could prove helpful in these estimations; however, sensor data is historically noisy and could provide less accurate results.

To illustrate how the algorithm works, an example of weight gain is provided. We see in figure 8 that we have a prediction based on 1 lb gain per

day, which will be our physical model, and a measurement presumably from a scale showing a much higher number, which will be our sensor value. We can then compute an estimate based on an α value between the measurement and prediction using equation 6.

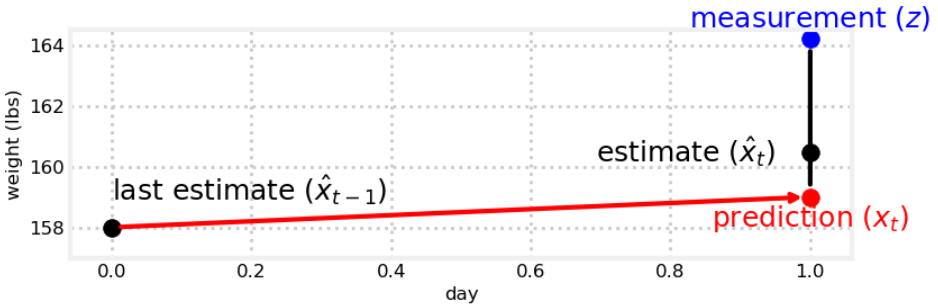


Figure 7: α - β Filter Prediction, Measurement and Estimate Based on Previous Estimate[15]

$$\hat{x}_t = x_t + \alpha\,(z - x_t) \tag{6}$$

Where α is a tunable parameter. However, this is only part of the solution. For the filter to represent the system's dynamics more accurately, it needs to be adaptable to change. Using equation 7, we can compute an updated value of α, which contains how far off our prediction was the measurement. This offset is also scaled depending on the distance between measurements.

$$\alpha_{t+1} = \alpha_t + \beta\,\frac{z - x_t}{\Delta t} \tag{7}$$

These two steps are often referred to as the predict, or *a-priori* step, and update, or *a-posteriori* step. A prediction step is used to create a prediction x_t which uses the last estimate $\hat{x}_{t-1}$. The block diagram in 8 shows how the process uses previous results to generate new results continuously. By generating the new estimate $\hat{x}_t$, one can use it to create a new estimate, and the loop continues. The algorithm can be seen in 1.

2.5.2 The Discrete Bayes Filter

This filter contains Bayesian statistics, which considers prior information when calculating uncertainty. Bayesian statistics provide a result called posterior probability distribution, or posterior for short, which multiplies the likelihood from frequentist statistics with our prior information and normalizes it because all probabilities have to sum up to one. Bayesian statistics contrast with frequentist statistics, which contains the probability based on infinite events.

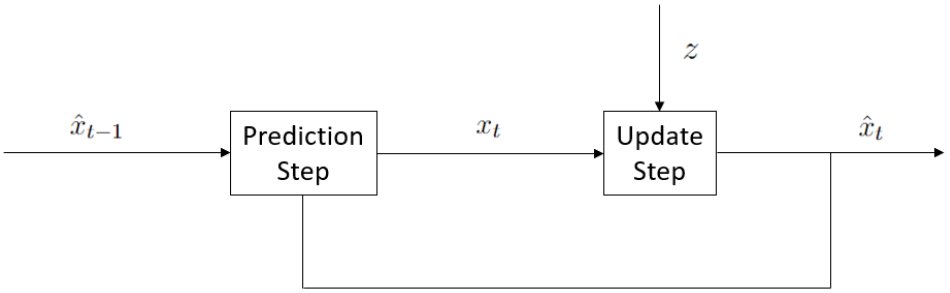


Figure 8: Prediction and Update Block Diagram

Algorithm 1 α - β filter

procedure AlphaBetaFilter(α, β)
Input x_t, z, α, β, Δt
  for t = 1 ... n **do**
    $R = z - x_t$
    $\hat{x}_t \leftarrow x_t + \alpha_t \times (R)$ ▷ Update step
    $\alpha_{t+1} = \alpha_t + \beta \times (R/\Delta t)$
  end for
end procedure

$$posterior = \frac{likelihood \times prior}{normalization} \tag{8}$$

This information is useful when we have noisy sensor data because it can model just how accurate our sensors have been in the past and how reliable our future readings will become. The prediction-update format will become:

$$\begin{aligned} \bar{\mathbf{x}} &= \mathbf{x} \star \mathbf{f}_{\mathbf{x}}(\bullet) \qquad \textbf{Prediction Step} \\ \mathbf{x} &= \|\mathcal{L} \cdot \bar{\mathbf{x}}\| \qquad \textbf{Update Step} \end{aligned} \tag{9}$$

Where $f_x(\bullet)$ is called a kernel, which Gaussian RBFs use. This term usually contains the prior information system behavior, which allows for a better prediction. The update step contains a likelihood function $\mathcal{L}$ which contains the associated belief regarding the accuracy of the sensors. Combining these two statistical predictions makes the discrete Bayes filter which accounts for noisy data values.

After the prediction step, the states are called **a priori**, while after the update step, they are called **a posteriori**. This convention is used in the Kalman filter and hence throughout the paper.

Convolution

The $\star$ icon indicates that one uses convolution to solve the term. Convolution means that the product is a third function describing the relationship and interaction between the two functions [16].

$$f(t) \star g(t) = \int_0^t f(\tau)g(t-\tau)d\tau \tag{10}$$

Function 10 is the official convolution formula, and in the Bayes filter, this means that the prediction estimates how our states interact with our chosen kernel. Let us say that we have a kernel for the probability of accuracy within a measurement. Performing convolution for a new measurement means that we go over each element in the array, multiply by the kernel and sum the results.

2.5.3 Gaussian Distribution

A problem with the Bayes filter is that we do not have a concrete answer to where our states are located, just an overview of the probability of each occurrence. A Gaussian distribution provides a solution to transfer this probabilistic data into a model of real-life applications.

Mean, Variance and Standard Deviation

Before explaining the concept of Gaussians, we will go through the underlying principles, which are the mean, variance, and standard deviation of the dataset. The **mean** value of a data set is defined as the sum of numbers divided by the amount of numbers.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{11}$$

The **variance** of a data set is defined as the sum of the difference between all data points to the mean squared, divided by the amount of numbers.

$$VAR(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{12}$$

Finally, the **standard deviation** is simply defined as the square root of the variance.

$$\sigma = \sqrt{VAR(X)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{13}$$

1-D Gaussian Distribution

The central limit theorem tells us that the *given a sufficiently large sample size from a population with a finite level of variance, the mean of all sampled variables from the same population will be approximately equal to the mean of the whole population* [17]. As the amount of measurements or data points goes towards infinity, plotting the result will look like a Gaussian. The essentials of this theorem were first discovered by Laplace in 1810 [18] and have had a significant impact on how we model probability and the world around us. The terms introduced previously are vital because they allow us to build a Gaussian distribution using function 14.

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{14}$$

One can see from figure 9 that a lower variance means a higher peak. In terms of probability, lower variance indicates more accurate readings. If the Gaussian is not normalized, it is called a *Gaussian function* instead of a *Gaussian distribution.*

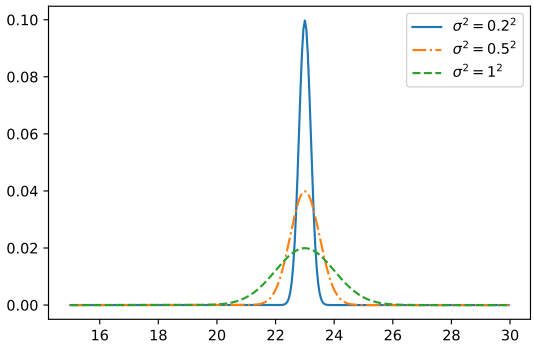


Figure 9: Continuous Gaussian Distribution Representation with Changing Variance

An interesting observation is that the product of two Gaussian distributions will be a Gaussian distribution. However, adding two Gaussian distributions creates a Gaussian function. By adding two Gaussians, the new mean and standard deviation becomes:

$$\begin{aligned} \mu &= \mu_1 + \mu_2 \\ \sigma^2 &= \sigma_1^2 + \sigma_2^2 \end{aligned} \tag{15}$$

And for a product, the mean and standard deviation becomes:

$$\begin{aligned} \mu &= \frac{\sigma_1^2\mu_2 + \sigma_2^2\mu_1}{\sigma_1^2 + \sigma_2^2} \\ \sigma^2 &= \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \end{aligned} \tag{16}$$

Correlation and Covariance

Covariance describes how two variables correlate. If the change is similar, then the variables are positively correlated. One example of this would be grain- and bread prices. If one variable tends to increase when the other decreases, they are negatively correlated. The equation for covariance is:

$$COV(X,Y) = \sigma_{xy} = \mathbb{E}\left[(X-\mu_x)(Y-\mu_y)\right] \tag{17}$$

This equation is similar to the equation for variance. One can say that variance describes the differences within a variable while covariance describes the difference between all the variables. $\mathbb{E}$ is called the expected value and is defined as the weighted average of the values in a range [19]. An example of this would be the expected value of dice. Since six sides range from 1 to 6, all equally likely, the expected value will be $\sum_1^6 (1/6)x_i$, adding up to 3.5. If all data points are weighted equally and the problem is discrete, the equation is:

$$\mathbb{E} = \frac{1}{N}\sum_{i=1}^{n} x_i \tag{18}$$

In multivariate systems, to keep track of all covariance values we use a covariance matrix Σ.

$$\Sigma = \begin{bmatrix} \mathbb{E}\left[(X-\mu_1)(Y-\mu_1)\right] & \mathbb{E}\left[(X-\mu_1)(Y-\mu_2)\right] & \dots & \mathbb{E}\left[(X-\mu_1)(Y-\mu_n)\right] \\ \mathbb{E}\left[(X-\mu_2)(Y-\mu_1)\right] & \mathbb{E}\left[(X-\mu_2)(Y-\mu_2)\right] & \dots & \mathbb{E}\left[(X-\mu_2)(Y-\mu_n)\right] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}\left[(X-\mu_n)(Y-\mu_1)\right] & \mathbb{E}\left[(X-\mu_n)(Y-\mu_2)\right] & \dots & \mathbb{E}\left[(X-\mu_n)(Y-\mu_n)\right] \end{bmatrix}$$

The diagonal contains the variance of the different variables, while the off-diagonal elements contain the covariance between the elements. The matrix is symmetric meaning that $\mathbb{E}\left[(X-\mu_x)(Y-\mu_y)\right] = \mathbb{E}\left[(X-\mu_y)(Y-\mu_x)\right]$.

Multivariate Gaussian Distribution

Using our covariance matrix, we are able to create a new Gaussian distribution function for a multivariate case:

$$f(x, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left[-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right] \tag{19}$$

Which is very similar to the 1-D case. Another similarity is the adding and multiplying of Gaussians. Adding multivariate Gaussians give yields the following mean and error covariance matrix:

$$\begin{aligned} \mu &= \Sigma_2(\Sigma_1 + \Sigma_2)^{-1}\mu_1 + \Sigma_1(\Sigma_1 + \Sigma_2)^{-1}\mu_2 \\ \Sigma &= \Sigma_1(\Sigma_1 + \Sigma_2)^{-1}\Sigma_2 \end{aligned} \tag{20}$$

While multiplying two multivariate Gaussians gives:

$$\begin{aligned} \mu &= (\Sigma_1 + \Sigma_2)^{-1}(\Sigma_1\mu_2 + \Sigma_2\mu_1) \\ \Sigma &= (\Sigma_1 + \Sigma_2)^{-1}\Sigma_1\Sigma_2 \end{aligned} \tag{21}$$

2.5.4 The Kalman Filter

This section will go through a Kalman filter's essentials based on the theory introduced in previous sections. First, going through a 1-D case before extending into multiple dimensions.

The Kalman filter is a version of the discrete Bayes filter which uses Gaussians instead of discrete values for the distributions and measurements. A pair of mean and standard deviation values can replace a whole set of discrete values, making the algorithm more space-efficient. The resulting prediction will also be a continuous function rather than multiple discrete values. The updated steps from 9 will be:

$$\begin{aligned} \bar{\mathbf{x}}_{\mathcal{N}} &= \mathbf{x}_{\mathcal{N}} + \mathbf{f}_{\mathbf{x}_{\mathcal{N}}}(\bullet) \qquad && \textbf{Prediction Step} \\ \mathbf{x}_{\mathcal{N}} &= \mathcal{L} \cdot \bar{\mathbf{x}}_{\mathcal{N}} && \textbf{Update Step} \end{aligned} \tag{22}$$

Where the subscript $\mathcal{N}$ describes that the item is Gaussian. The convolution step in 9 was initially put in place to sum the interaction between two functions. However, when using Gaussians, this convolution can simply be replaced by the adding of the two functions, since the result is also a Gaussian.

Kalman Gain

A term which is used to simplify calculations in the Kalman filter is the Kalman gain. Equation 16 described the resulting mean and standard deviance from multiplying two Gaussians. This equation is simplified using the Kalman gain K:

$$\begin{aligned}\mu &= \left(\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)\mu_1 + \left(\frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)\mu_2 \\ &= K\mu_1 + (1-K)\mu_2 \\ &= \mu_2 + K(\mu_1-\mu_2)\end{aligned} \tag{23}$$

$$\begin{aligned}\sigma^2 &= \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2} \\ &= K\sigma_2^2 \\ &= (1-K)\sigma_1^2\end{aligned} \tag{24}$$

From the simplification of mean and standard deviation, the scaling factor K is defined as:

$$K = \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2} \tag{25}$$

Which is the equation for the Kalman gain.

1-D Kalman Filter

Using all the tools we have derived up until now, we can set up an alorithm for the prediction and update steps in the Kalman filter. The book *Kalman and Bayesian Filters in Python* sets up the equations as:

Table 3: 1-D Kalman Filter Prediction Step

Equation	Implementation	Kalman Form
$\bar{x} = x + f_x$	$\bar{\mu} = \mu + \mu_f$	$\bar{x} = ax + bu_k$
	$\bar{\sigma^2} = \sigma^2 + \sigma_{f_x}^2$	$\bar{P} = P + Q$

Table 4: 1-D Kalman Filter Update Step

Equation	Implementation	Kalman Form
$x = \|\mathcal{L}\bar{x}\|$	$y = z - \bar{\mu}$	$y = z - \bar{x}$
	$K = \frac{\bar{\sigma}^2}{\bar{\sigma}^2+\sigma_z^2}$	$K = \frac{\bar{P}}{\bar{P}+R}$
	$\mu = \bar{\mu} + Ky$	$x = \bar{x} + Ky$
	$\sigma^2 = \frac{\bar{\sigma}^2\sigma_z^2}{\bar{\sigma}^2+\sigma_z^2}$	$P = (1-K)\bar{P}$

The *Implementation* column describes how the problem is set up based on principles described in previous sections, while the *Kalman Form* column

is the formal notation. The latter is more commonly used in cybernetic settings and will be the default for the remainder of the paper.

Multivariate Kalman Filter

Extending the Kalman filter to multiple dimensions is the last part of this subchapter and contains certain differences from a 1-D Kalman filter. The equations will be:

Table 5: Multivariate Kalman Filter Prediction Step

Equation	Multivariate Kalman Form
$\bar{x}_{\mathcal{N}} = x_{\mathcal{N}} + f_{x_{\mathcal{N}}}(\bullet)$	$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{u}$
	$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^{\top} + \mathbf{Q}$

Table 6: Multivariate Kalman Filter Update Step

Equation	Multivariate Kalman Form
$x_{\mathcal{N}} = \mathcal{L} \cdot \bar{x}_{\mathcal{N}}$	$\mathbf{y} = \mathbf{z} - \mathbf{H}\bar{\mathbf{x}}$
	$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^{\top}(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^{\top} + \mathbf{R})^{-\mathbf{1}}$
	$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}$
	$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}$

One can see that there are a lot of new variables introduced, however the process is the same as the 1-D filter. Use Gaussians to represent the model, errors and measurements which are used to create the next state and estimate a value between the measured and estimated state. An overview over the newly introduced variables are found in table 7.

The multivariate Kalman filter is what is used to adjust the model in this paper.

Table 7: Kalman Filter Variable Overview

Variable	Name	Description
$\mathbf{x}$	State Mean	
$\mathbf{P}$	Covariance	
$\mathbf{F}$	State Transition Function	
$\mathbf{Q}$	Process Covariance	
$\mathbf{B}$	Input Matrix	
$\mathbf{u}$	Input Vector	
$\mathbf{H}$	Measurement Function	
$\mathbf{z}$	Measurement Mean	
$\mathbf{R}$	Measurement Noise Covariance	
$\mathbf{y}$	Residual	
$\mathbf{K}$	Kalman Gain	

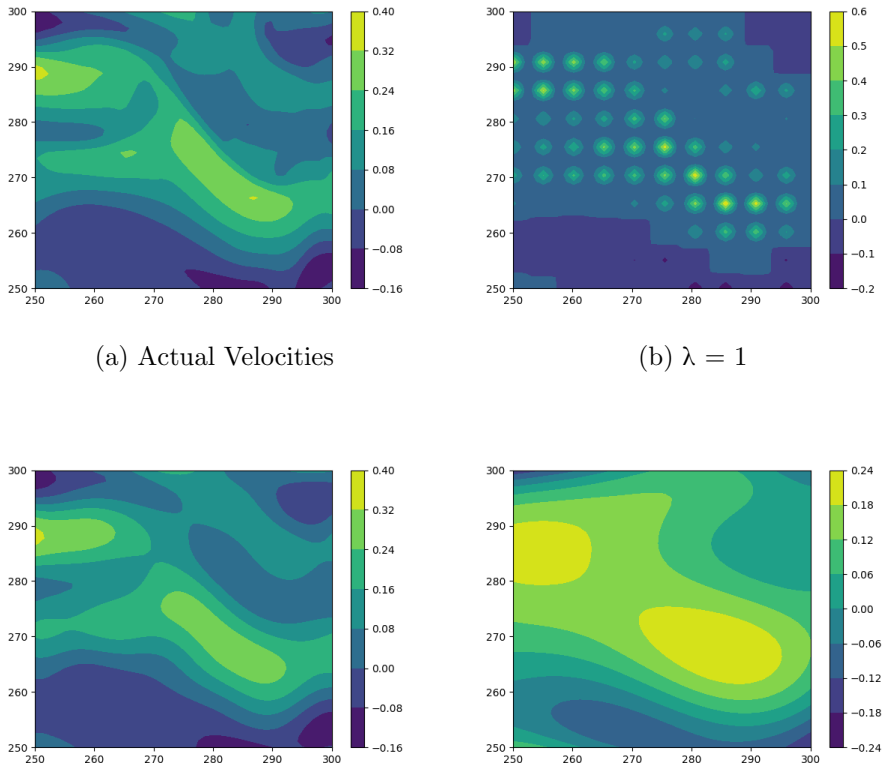
CHAPTER 3

Previous Work

The research completed during the fall semester of 2021 focused on the behavior of RBFs initialized on the SINMOD dataset [20]. The RBFs used constant and sinusoidal functions, which differ from this thesis's equations. In addition to this, initialization was completed using the scipy command `scipy.optimize.minimize`, which was a lot more time-consuming than the `mosek.fusion` package introduced in this paper. The minimization problem was set up to consider phase shifts in contrast to this paper with sine and cosine functions. There are however relevant results that will be helpful in this thesis. This section will go through those.

3.1 Scaling Factor λ

In the previous paper, the scaling factor used the letter σ, however, this has been changed to λ in this paper to avoid confusing it with Kalman filter variable names. The scaling factor refers to a hyper-parameter within the Gaussian kernel: $\exp(-\gamma\|x_m - x_n\|^2)$, where $\gamma = 1/(2\lambda)$. Essentially, this value is how we scale the kernel to get contributions from the RBF points. Over-scaling means too much information is gained at a point, and under-scaling means there is not enough information. The resolution for each RBF point was set to be 10 points between each. λ values were changed over a 50×50 point grid to observe variations. A contour plot was used to show flow velocities at different points in the grid. The results of the changing scaling parameter can be seen in figure 10. A λ value of 1000 displayed over-scaling, and 1 displayed under-scaling. The most accurate representation of the actual flow field can be found in figure 10c, at $\lambda = 100$. This show an optimal λ value depending on the flow field and resolution, which do not cause over-or under scaling.



(a) Actual Velocities

(b) λ = 1

(c) λ = 100

(d) λ = 1000

Figure 10: Varying Lambda Values in Gaussian Kernel in Simulations and Resulting Velocities

More tests were completed on the same 50×50 data grid, but this time over a broader range of λ values, taken at logarithmic intervals. The Frobenius norm was taken at each λ value. Frobenius norm of an $m \times n$ matrix A is defined as the square root of a sum of the absolute squares of its elements [21], as displayed in equation 26.

$$\|A_F\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i,j}|^2} \tag{26}$$

The results can be seen in 11, where there exists a minimum at around $\lambda \in [10, 10\,000]$. Furthermore, the effects of under scaling reduce exponentially before the optimal λ values, while afterward, over scaling occurs at a more linear rate. As mentioned before, σ was used instead of λ in this paper, which can be seen in the x-axis label.

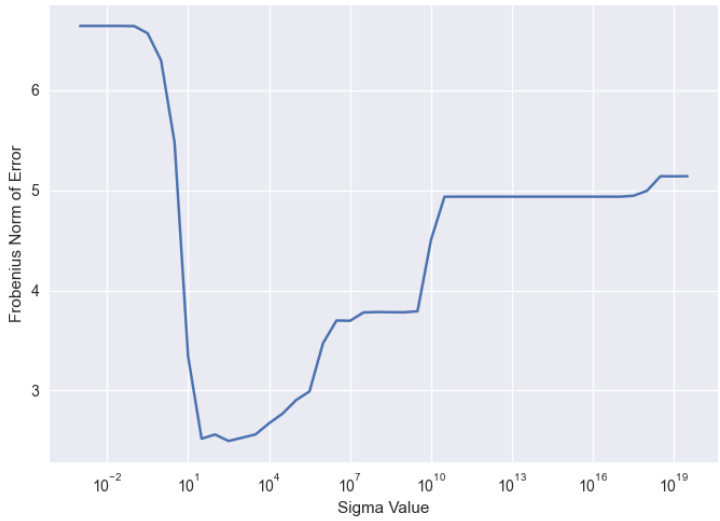


Figure 11: Frobenius Norm of Error Between Estimated and SINMOD Velocities on Dataset Versus Logarithmic λ

3.2 Resolution

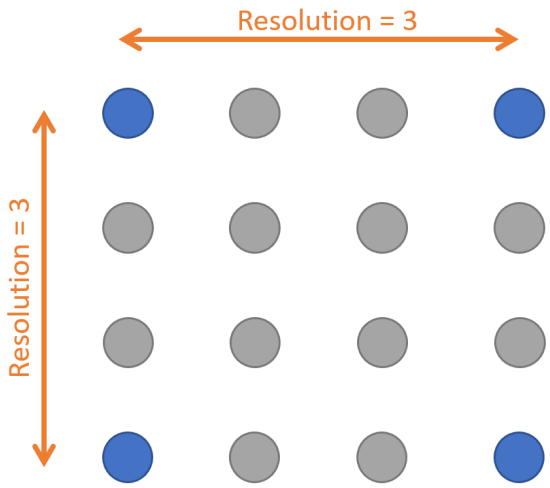
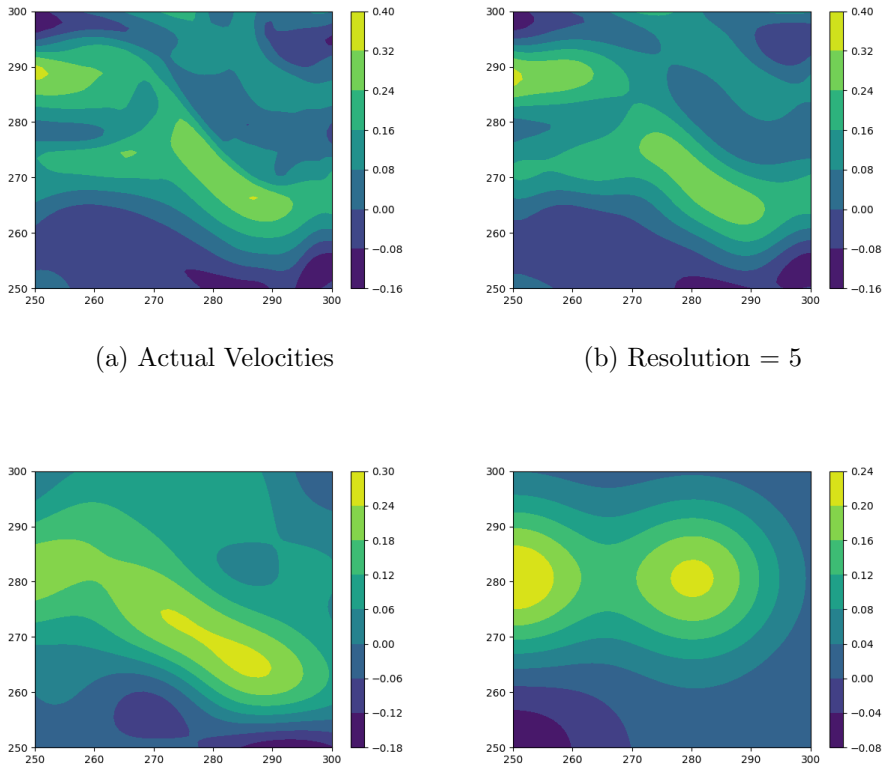


Figure 12: Resolution (Orange) Defined in This Project as the Amount of Data Points (Grey) Including Current RBF Point till the Next RBF Point Horizontally and Vertically

The resolution in the project was defined as the distance between RBF points horizontally and vertically, including the current RBF point. An illustration of this is seen in figure 12. Note that resolution is not the

distance between any arbitrary RBF point but the ones horizontally and vertically next to each other. Since the RBF points are placed equidistant, these distances will be equal throughout the grid. Tests were completed for the resolution as were completed for λ over a 50×50 data grid with a set λ value of 150. The results are seen in figure 13.



(a) Actual Velocities

(b) Resolution = 5

(c) Resolution = 10

(d) Resolution = 30

Figure 13: Varying Resolution in Simulation and Values Resulting Velocities

Lower resolutions, meaning lower distances between RBF points, led to a better representation of the flow field.

Next, the Frobenius error norm between the simulation and the actual flow was computed at different resolutions with a set λ value of 120. The results can be seen in figure 14.

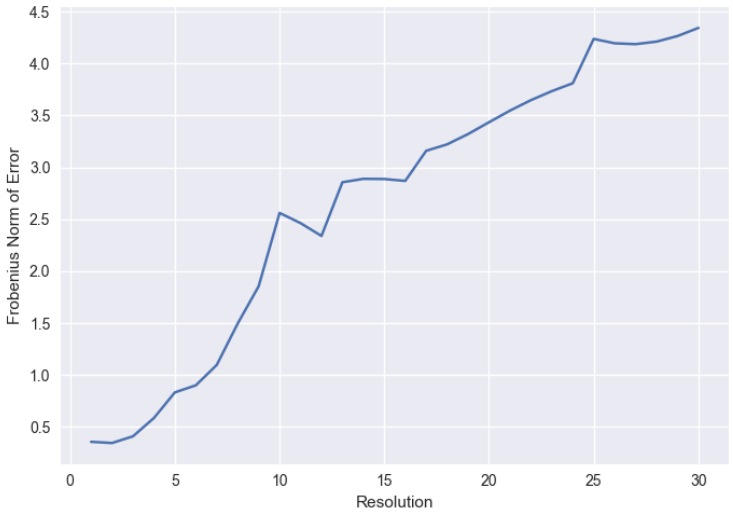


Figure 14: Frobenius Norm of Error Between Estimated and SINMOD Velocities on Dataset Versus Resolution

Figure 14 shows a linear trend. This means that a low resolution (lower distance between RBF points) yields a better result across the field.

3.3 Correlation Scaling Factor and Resolution

The results from the previous two sections prompted the question of the correlation between scaling factors and resolution and how they affect each other. Optimal λ values ranging from 1-1000 with steps of 1 were found at resolutions ranging from 10-30 in steps of 1. The optimal λ value was defined as the value that yielded the lowest Frobenius error norm for that resolution.

After running these tests, there was no clear indication of a trend between resolution and λ value. There is a slight linear trend after a resolution of 18, but no conclusions were drawn in the project paper.

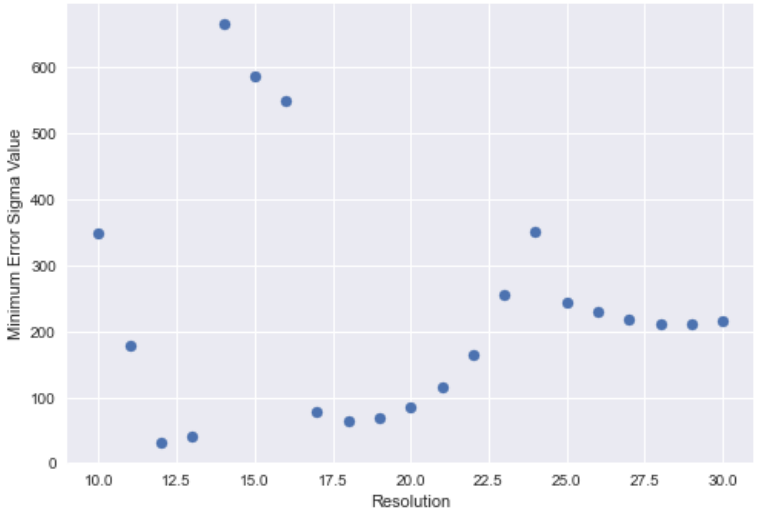


Figure 15: Extraction of Best λ Value at Different Resolutions Based on Lowest Frobenius Norm of Error Between Estimated and SINMOD Velocities

3.4 Conclusion

The valuable items derived from this research were that we would like an optimal λ and resolution value that yields accurate flow estimates. One could also see that a lower distance between RBF points significantly decreased error. This decrease means that one should choose a sufficiently low resolution and find out the optimal λ value for that resolution.

There was no clear trend for optimal λ value at different resolutions, but one can theorize that the size and behavior of the dataset significantly impact the optimal λ value.

CHAPTER 4

Method

This chapter describes the development process and methods used to use RBFs to model flow in real-time. This includes the implementation of a Kalman filter, different optimization routines, analyses of model pitfalls and case studies.

4.1 Data Set Location

To test and develop our method, a trial data set was extracted using the SINMOD simulation right outside of Trondheim. Its dimensions are $x, y, z, t \in \{400, 350, 36, 217\}$ with a resolution of 800 m between each point horizontally and one hour between each estimation. Vertically, the resolution varies with lower steps down to 3 m closer to the surface and steps up to 500 m at depths beyond 1000 m. This was considered a sufficient data set to complete testing in since the underwater drones have a typical operational area of 50×50 km^2. If field testing was necessary, then the data set is of close proximity to the student's university.

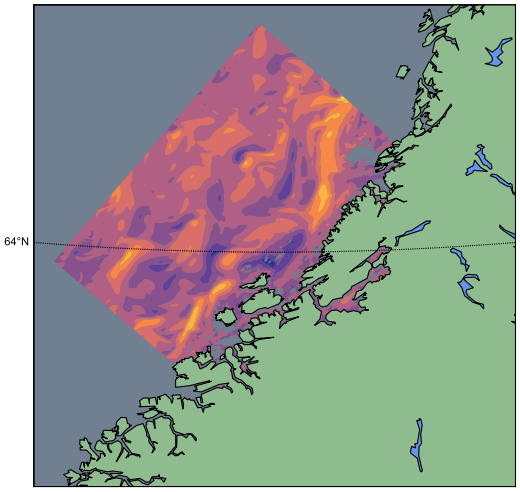


Figure 16: Contour Plot Showing u-Velocity Simulation Located Outside Trondheim, Norway. Contours Range from Dark (High Velocity) to Light (Low Velocity)

4.2 Initial Model Extensions

The initial model establishes an accurate representation of the flow field at the specified RBF points before using the Gaussian kernel. The RBF model established during the semester project contained constant- and sinusoidal functions combined with a Gaussian kernel to scale the contribution of each RBF point. This project introduces Laguerre polynomials from section 2.4.2 to model non-sinusoidal flow contributions. The previous paper used scaled sinusoidal functions with phase shift as a variable.This paper uses scaled sines and cosines instead. The reasoning for this is further explored in section 4.2.1. Further extensions include using a MOSEK library for optimization of variables and doing a second optimization of the whole grid to improve grid point flow optimization.

4.2.1 MOSEK Library Implementation and Equation Change

As mentioned in section 3, the function `scipy.optimize.minimize` was changed to the MOSEK Optimization Suite software package. The MOSEK package can not be implemented simply when using a phase shift. The equations were therefore altered to only include scaled sines and cosines. Tests were completed using the `scipy.optimize.minimize` and compared scaled sines with a scaled phase shift to scaled sine- and cosine functions at the point $(x, y, z) = (250, 250, 5)$ at times $t \in [15, 150]$ to see the performance of both and if scaled sines and cosines would be a reliable alternative.

4.2.2 Function Setup

Building on theory from section 2.3, the basis functions at each RBF point will need to be decided before optimization. As mentioned previously, the choice fell on Laguerre polynomials as well as sines and cosines. The estimates at each RBF point will be denoted as ξ_i and have the function:

$$\xi_i = \sum_{n=1}^{W} \eta_{1,i,n} \sin(\omega_n t) + \eta_{2,i,n} \cos(\omega_n t) + \sum_{m=1}^{L} \eta_{3,i,m} \mathcal{L}(m, t) \tag{27}$$

Where i denotes the specific grid point, W is the amount of constitutional frequencies and L is the order of Laguerre polynomials. The parameters η in equation 27 was found using least-squares regression in combination with the MOSEK library optimization. The minimization problem is set up as:

$$\min_{w \in \mathbb{R}^d} \; ||y - Xw||_2 \tag{28}$$

Where X is the matrix of the optimization functions, w contains parameter values and y has the optimization values. In the case of our RBF point

estimation function ξ_i, the setup will be equal to the one seen in 29, where T is the amount of optimization points and M contains the SINMOD estimation values. There will be $2W + L$ parameters to be optimized because there are W sines and cosines and L Laguerre functions.

$$X = \begin{bmatrix} \sin(\omega_1 t_1) & \cos(\omega_1 t_1) & \dots & \cos(\omega_W t_1) & \mathcal{L}(1, t_1) & \dots & \mathcal{L}(L, t_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \sin(\omega_1 t_T) & \cos(\omega_1 t_T) & \dots & \cos(\omega_W t_T) & \mathcal{L}(1, t_T) & \dots & \mathcal{L}(L, t_T) \end{bmatrix}$$
$$w = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_{2W+L} \end{bmatrix}, \qquad y = \begin{bmatrix} M_1 \\ \vdots \\ M_T \end{bmatrix} \tag{29}$$

The primary change in the functions from the project thesis were the implementation of Laguerre polynomials. Testing was completed on varying degrees of these to see their accuracy at different degrees. These tests were computed at the RBF point $(x, y, z) = (250, 250, 5)$ at timepoints $t \in [0, 100]$.

4.2.3 Second Optimization

As mentioned in 2.3, the RBF also uses a Gaussian kernel which was set up using a λ value as a hyper parameter. Equation 30 shows the functions where i indicates the point at which the calculations are taking place and m is the RBF point.

$$\phi_i = \exp\left(-\frac{(x_i - x_m)^2 + (y_i - y_m)^2 + (z_i - z_m)^2}{2\lambda^2}\right) \tag{30}$$

For N RBF points, this equation is summed over N iterations. We get an estimate V of the flow at an arbitrary point in the grid by combining equations 27 and 30.

$$V = \sum_{i=1}^{N} \phi_i \xi_i \tag{31}$$

A second optimization was completed to scale the Gaussian Kernel and the contribution of each RBF point to the flow field. There were two proposed solutions: To have a constant scaling for the whole field, or to have a scaling for each RBF. The two options would mathematically look equations 32 and 33, where the constant adjustment (32) has one variable whilst the RBF adjustment (33) has as many as there are RBFs.

$$K_4 \sum_{i=1}^{N} \phi_i \xi_i \tag{32}$$

$$\sum_{i=1}^{N} K_{4,i}\, \phi_i \xi_i \tag{33}$$

The methods were tested over the data set $x \in [200, 240]$, $y \in [200, 240]$, $z \in [5, 10]$ and $t \in [0, 50]$. Mean and standard deviation for the whole grid was computed over the whole grid. To graphically see the effects, values at the point $(x, y, z) = (202, 202, 5)$ were shown over 50 timesteps.

4.3 Gaussian Analysis

An experiment was completed on an arbitrary 40×40 2D grid to showcase the potential overlapping of Gaussian RBF values and its influence by the λ value. RBF values were set to be 5 at a resolution of 5. This experiment is to show over-and under-scaling much like in the project thesis. The field was analyzed at $\lambda \in [1, 10, 100, 1000000]$. Function 30 was used to generate results in between RBF points.

4.4 Kalman Filter Implementation

The next step after simulating RBF points was to implement a Kalman filter to adjust between measurements and model. This draws from theory introduced in 2.5.4 where the first step is to generate a state-space model of our system to be used in the filter. States in this case are the flow estimates ξ at the RBF points. The state transition function and control input matrix will both be identity matrices which is a linear approach. We choose this because we know that the values are already time varying and hence a rough estimate is sufficient. w_k is defined as the process noise. The reason for choosing identity matrices is because the state at timestep k, x_k, will need to be retrieved using the re-optimization introduced in section 4.5. It should be noted that there is no input u_k. The function can be seen in 34, where k indicates the timestep and m the amount of RBF points.

$$x_{k+1} = \underbrace{I_m}_{F} x_k + \underbrace{I_m}_{B} u_k + w_k \tag{34}$$

The model we have initialized will be realized in y, so H will be a function of our Gaussian functions at the points of measurements. Hence, when multiplied with the states it creates an estimated value at that point. v_k is defined as the measurement noise. The function 35 shows this process, with n and m being the amount of measurements and RBF points respectively.

$$y = \underbrace{\begin{bmatrix} \phi_1(x_1) & \dots & \phi_m(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \dots & \phi_m(x_n) \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{bmatrix}}_{x} + v_k \tag{35}$$

To test the validity of the filter, a data set $x \in [250, 265]$, $y \in [250, 265]$, $z = 5$ and $t \in [0, 50]$ was used. Resolution was set to be 5, meaning there were 16 RBF points and λ was set to be 10. There were 27 measurements taken in between each RBF point for all 50 timepoints. Noise parameters were set to be $Q = 0.001I_m$, $R = 0.001I_n$ and $P = 0.001I_m$. The library `filterpy.kalman` was used to create a `KalmanFilter` object to emulate the filter. After estimating the new RBF point values they were multiplied with the Gaussian kernel 30 to create estimates at the measurement point. The measurement used constant offset noise in the form introduced in 4.6.

4.5 Reparameterization

After implemnting the Kalman filter, the ocean current model's next step is generating new η variables. The Kalman filter gives updated states which represent the RBF point values. For each a posteriori state, one value is outputted at each RBF point. Previous a posteriori states from the Kalman filter at the RBF point are used in conjunction with the current to create more analysis points for the optimization algorithm since one analysis point is insufficient for complete optimization.

The optimization problem is similar to the one in 29 and 28, however, in this case, the y matrix will be the accumulated RBF values from the Kalman filter. This will generate new η variables which can be used in the ocean current model. The function setup will then be:

$$X = \begin{bmatrix} \sin(\omega_1 t_1) & \cos(\omega_1 t_1) & \dots & \cos(\omega_W t_1) & \mathcal{L}(1, t_1) & \dots & \mathcal{L}(L, t_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \sin(\omega_1 t_{KT}) & \cos(\omega_1 t_{KT}) & \dots & \cos(\omega_W t_{KT}) & \mathcal{L}(1, t_{KT}) & \dots & \mathcal{L}(L, t_{KT}) \end{bmatrix}$$

$$w = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_{2W+L} \end{bmatrix}, \qquad y = \begin{bmatrix} S_1 \\ \vdots \\ S_{KT} \end{bmatrix} \tag{36}$$

Where KT is the time the simulation has been running for and S are the Kalman state estimates at that point. As mentioned previously, the S values

are cumulative throughout simulation and are optimized at their respective time points.

4.6 Synthetic Measurement Data

We use our knowledge of Gaussian distributions from section 2.5.3 to create Gaussian, or white, noise for testing. This means that the values added or to our original data is taken randomly from a Gaussian distribution. Mathematically, as seen in 37, we use equation 14 and add it to our original data. In this thesis, x is the SINMOD simulation.

$$x_r = x + f(x, \mu, \sigma) \tag{37}$$

Next, we would like our artificial measurements to have systematic differences from the original data, this means either having a scaling value or a constant offset. Scaling is represented in equation 38 as multiplying x_r with a scaling variable K_S. To add constant offset, a K_O parameter is added to the function. A visual representation of both of these effects can be seen in figure 17.

$$x_{aug} = K_S x_r + K_O \tag{38}$$

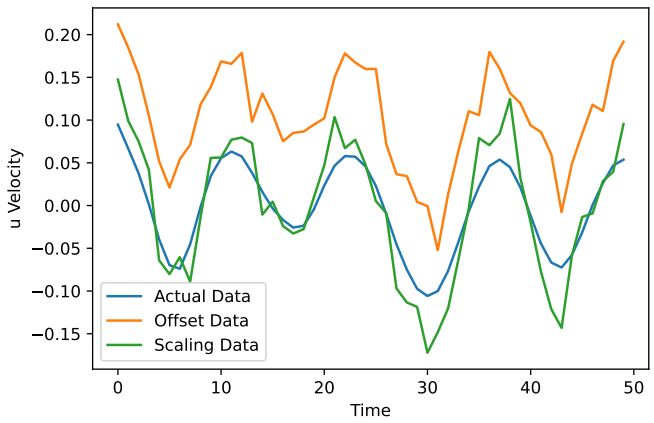


Figure 17: Effects of Offset (Orange) and Scaling Factor (Green) with Gaussian Noise

Ultimately for testing the algorithms, the choice fell on the constant offset augmentation. This was to see clear differences between the model, the measurements and the actual data when comparing.

4.7 Case Study I - Static Measurement

The first case study will obtain one static measurement and see its effects on the rest of the grid. This was completed over a 2-dimensional 40×40 point grid in a timespan of $t \in [0, 50]$. The coordinates of the grid are $x \in [200, 240]$, $y \in [200, 240]$, $z = 5$ with the one static measurement being taken at $(x, y, z) = (223, 223, 5)$. An overview over the RBF points and measurements can be seen in figure 18.

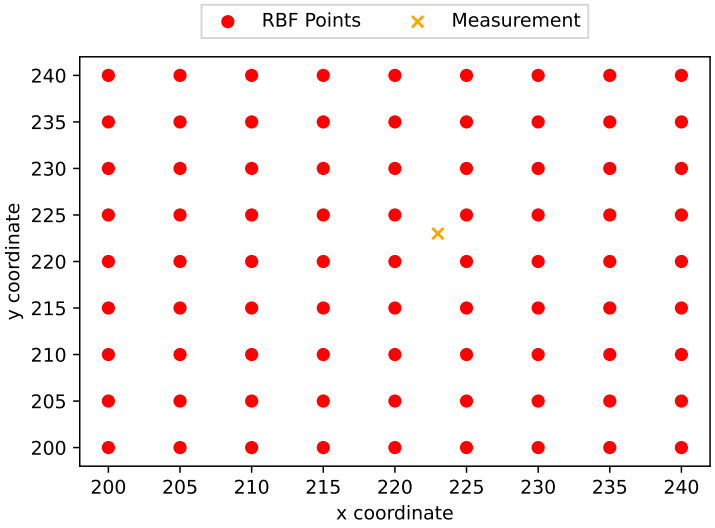


Figure 18: RBF and Measurement Locations for Case I

Analyses were completed for the whole grid at one time point and specific points for all times.

4.7.1 Timespan Analysis

As mentioned in section 4.5, the Kalman filter values used in reparameterization are cumulative. Hence, reparameterization will have different results and accuracies throughout the simulation. To see how effective the method would be in predicting future results, reparameterized parameters at $KT = 50$ and coordinates (x, y, z) = (223, 223, 1) in case I were tested for exceeding timepoints. The static model initialized on the SINMOD data was also tested.

Following this, parameters η at $KT = 5$ was extracted and used to generate flow estimates for times $t = 5$ and $t = 50$. This experimentation was to see how effective the reparameterization is at lower time points.

4.8 Case Study II - Measurements Exceeding RBF Points

The second case study tests the model's response to having more measurements than RBF points. The case area and timeframe will be the same as in case I, but case II includes two measurements in between each edge connecting the RBF points as seen in figure 19.

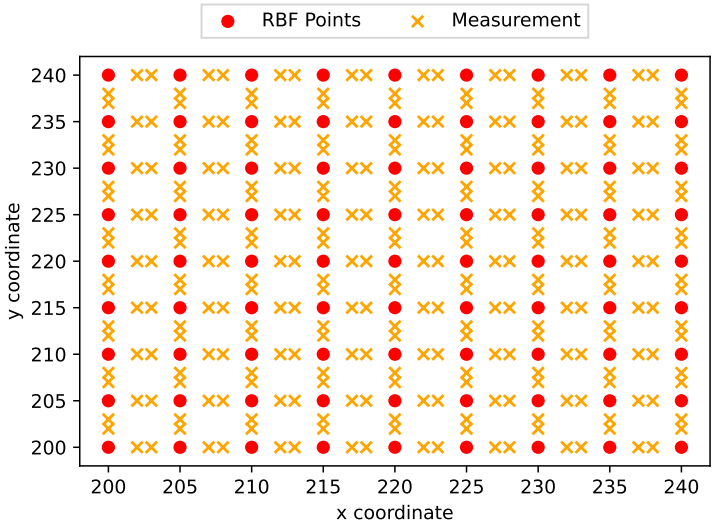


Figure 19: RBF and Measurement Locations for Case II

Analyses completed in this case were similar to the ones in case I, with whole-grid analyses at one time point and specific points for all time points.

4.9 Case Study III - Dyanmic Underwater Drones

A third case study was created to simulate real-life dynamic measurements. In section 2.2, the OCEANID drone, its measurements systems and intended deployment was introduced. Case III builds upon these requirements and uses them to create a dynamic example of measurements taken at different locations. This is to test the accuracy of the updated method for a realistic case and to see if the model correction is sufficient for different points within the grid.

A typical operation involving the drones have an operational area of $50\times50\text{km}^2$ and depth of 1000 m. To simplify the modelling process, we say that the resolutions for the data set are 250 m horizontally and 100 m vertically. According to specifications, 9 drones are dropped in between 4 RBF coordinates. The resolution specified means the drones are 250 m apart horizontally and vertically which is within typical operation. If the middle drone takes measurements every 100 meters, we can simulate a measurement

for each resolution layer, meaning 10 measurements for ascent and descent. From the specifications, we also know that 9 new drones are released every 15 minutes at a new location. A reasonable estimate would be that the drones takes 7.5 minutes to transverse each ascent and descent. This comes from the ascent and descent speeds and allows for an ascent and descent between each drop. A resolution of 4 was chosen to have three points between each RBF and the measurement drone centered between these. The deployment procedure with dropping drones each 15 minutes can be seen in figure 20.

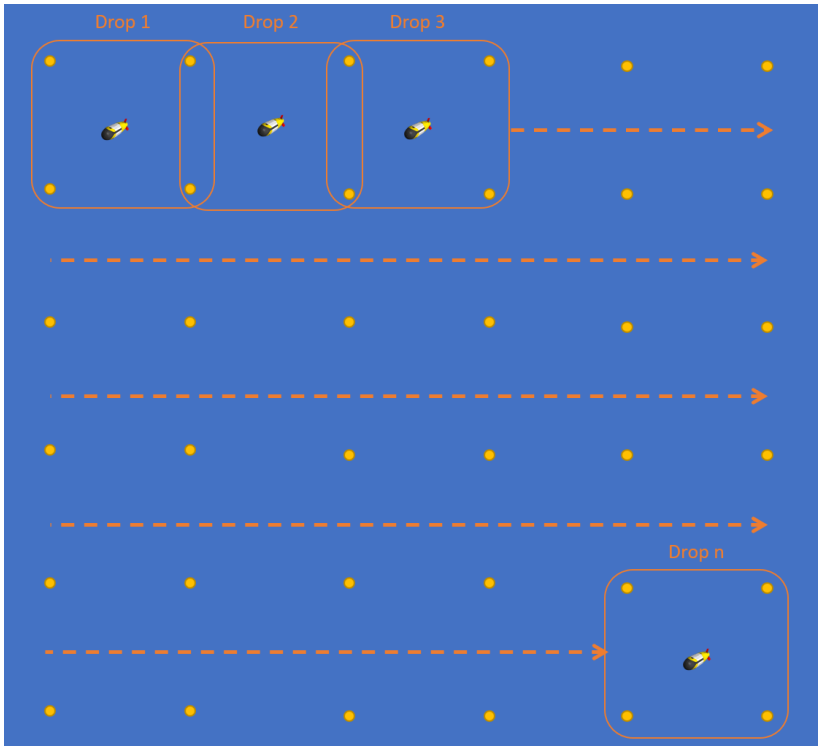


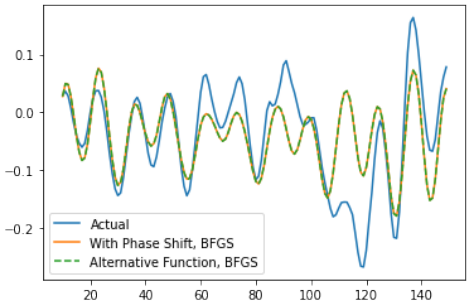
Figure 20: Node Drop Location Illustration for Case III Containing Nodes, Drop Zones (Orange Boxes) and RBF Points (Yellow Circles) up to n Drops

The data set used in this project has a time frame from 0 to 217 hours with 1 hour intervals. For this case, the interval was changed to 7.5 minutes, meaning one update per ascent and descent. This was done to include sufficient updates across the field and a realistic amount of node deployments. At the specified resolution, the SINMOD data set was not large enough to compute a $50\times50\text{km}^2$ grid, so a $5\times5\text{km}^2$ grid was chosen instead. All in all, this gives 25 drops over a 187.5 minute long period.

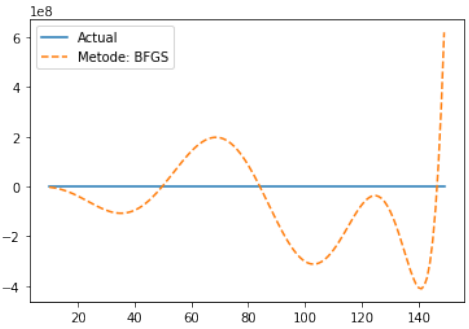
CHAPTER 5

Results

5.1 MOSEK and Function Testing



(a) SINMOD Data (Blue) Comparing Sine With Amplitude and Phase Shift Scaling (Orange) and Sines and Cosines With Amplitude Scaling (Green)

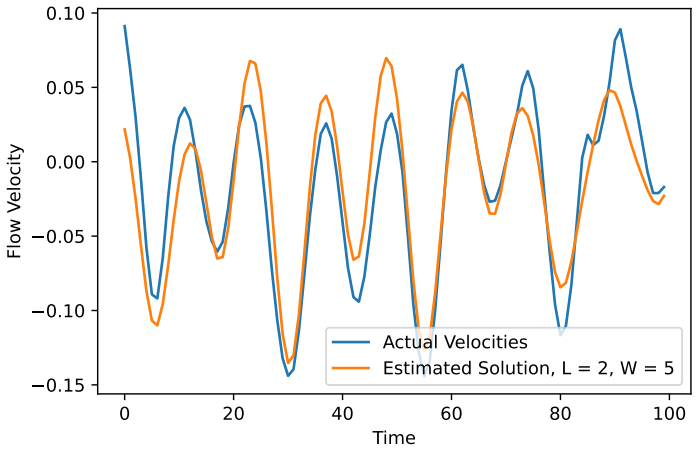


(b) Failed Convergence When Summing Up To 10th Degree Laguerre Polynomial

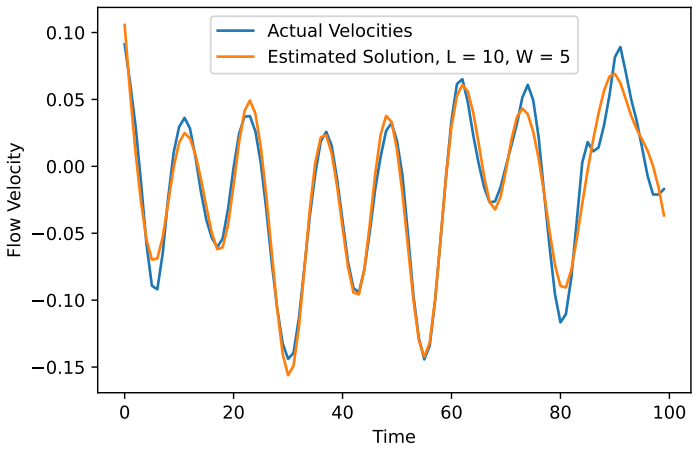
Figure 21a represents the difference between optimizing a sine with phase shift and an alternative function consisting of sines and cosines. One can see both function outputs were similar. This means that the MOSEK Optimization Suite is a viable option and that the function setup in 27 was changed to sines and cosines.

Figure 21b shows the Scipy minimize function not converging with an 8th degree Laguerre polynomial. Meanwhile, the MOSEK optimization converged up to the 12th degree. The MOSEK package showed significantly faster results overall and convergence at higher Laguerre polynomials. The package was therefore used for the remainder of this thesis.

5.2 RBF Point Estimation



(a) Laguerre 2nd Degree



(b) Laguerre 12th Degree

Figure 22: Comparing SINMOD Velocities to Estimates Using Sums of Laguerre Polynomials and Scaled Sines and Cosines With 5 Tidal Frequencies

Figure 22 shows the result of summing up to a 2nd- and 12th degree Laguerre polynomial at the RBF point estimation. One can see that the graph follows the actual velocities better at higher degrees. 10th degree polynomials were used for the remainder of this project because it was believed that Laguerre polynomials of too high degree could be problematic.

5.3 Second Optimization

As mentioned in 4.2.3, a second optimization was completed to scale the Gaussian kernel for the whole flow field. Two methods were suggested, one with a constant scaling of the whole field and one scaling each RBF. Table 8 shows the mean error and standard deviation for each of the suggested methods. Both adjustments improved the standard deviation and mean error of the result however the RBF adjustment had a slightly lower value.

Table 8: Second Optimization Results

Method	Mean Error	Standard Deviation
Original	4.721×10^{-2}	2.990×10^{-1}
Constant	-6.502×10^{-3}	1.020×10^{-1}
RBF Adjustment	3.970×10^{-3}	6.104×10^{-2}

The datapoint at $(x, y, z) = (202, 202, 5)$ during times $t \in [0, 50]$ are shown in figure 23 and shows both methods and their outputs. The RBF point adjusted method is closer to the SINMOD data in blue.

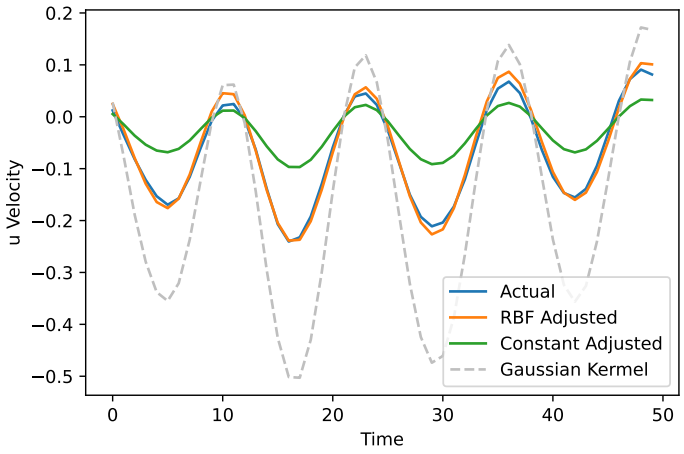
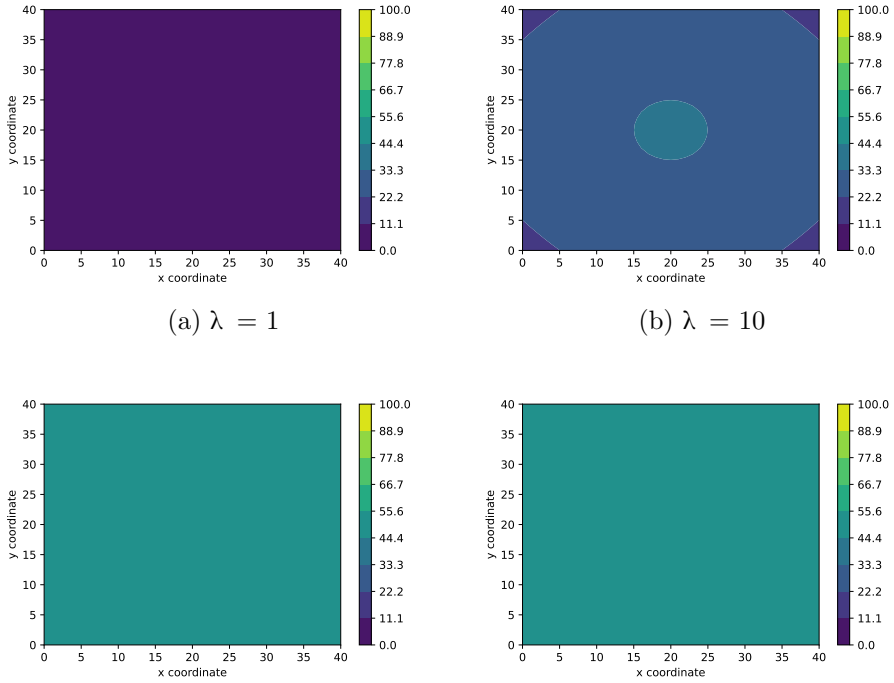


Figure 23: Comparing SINMOD Data (Blue) With Optimized RBF Point Multiplied With Gaussian Kernel, No Adjustment (Grey), Constant Adjustment (Blue) and RBF Point Adjustment (Orange)

The conclusion from both of these results is that the RBF point estimation performs better than the constant adjustment. This was therefore the preferred method when continuing experimentation.

5.4 Gaussian Analysis



(a) $\lambda = 1$ (b) $\lambda = 10$

(c) $\lambda = 100$ (d) $\lambda = 1\ 000\ 000$

Figure 24: Gaussian Analysis at Constant RBF Values

Figure 24 showcases four λ values for a constant RBF point field calculated using a Gaussian kernel. These results aim to see the pitfalls if a Gaussian kernel is over or under-tuned. The RBF values are not from the dataset but instead set to be constant values of 5. The RBF points do not contribute to calculating the grid values when lambda is low, creating a lower value than at the RBF points. At higher λ values, the contributions overlap, creating a larger value than the initial 5. At a λ value of 10, the overlap is not as present compared to the higher values; however, one can see that a middle point obtains higher values than the surrounding ones. These higher values are an indication that overlapping effects are taking place. These results are useful when it comes to discussing the effects of using a Gaussian kernel.

5.5 Kalman Filter Implementation

Testing the Kalman filter with the parameters mentioned in 4.4 and multiplying the a posteriori states with the Gaussian kernel led to the plot seen in figure 25 at the point $(x, y, z) \in (262, 262, 5)$.

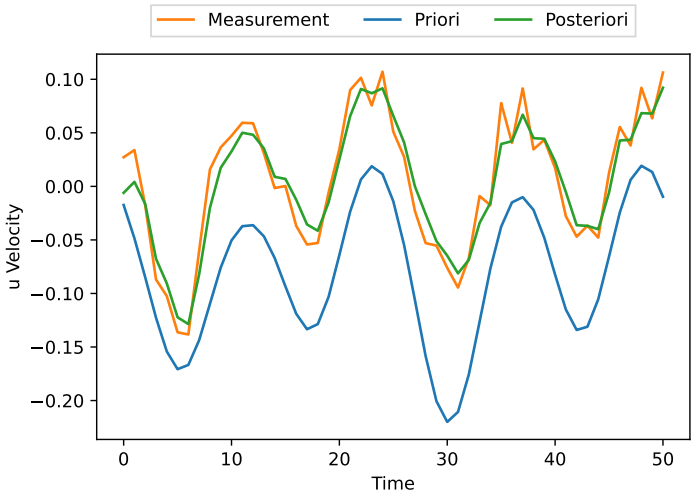


Figure 25: Kalman Filter Results

Here we can see that the filter follows the measurements closely. If the filter was tuned to trust the model more than the measurements, the results might follow the model estimate curve more closely. However, this indicated that the Kalman filter worked according to the expectations and no further experimentation was completed.

5.6 Re-Optimizing Parameters

Figure 26 displays the a priori and a posteriori states for the Kalman filter as well as the updated, or reparameterized, model created using the methods described in 4.5. After getting measurements from 50 timesteps, the updated model fits the a posteriori state well.

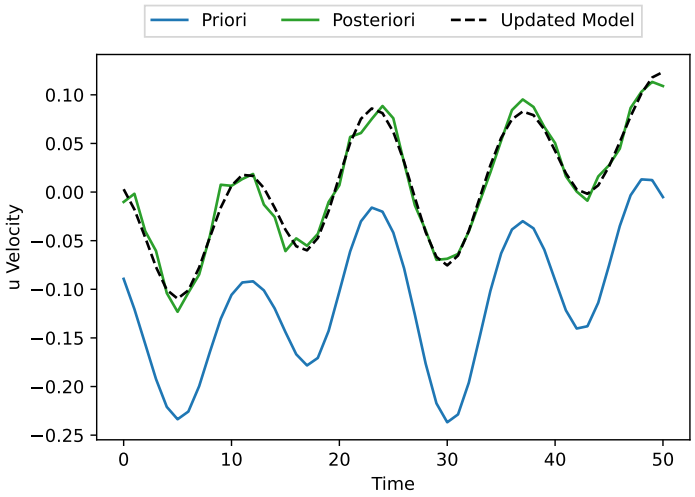


Figure 26: Updated Model at RBF

5.7 Case Study I - Static Measurement

Case I received one measurement creating an a priori, a posteriori, and reparametrized model per timepoint. The absolute error between the a posteriori state and the reparametrized model was also computed. One can see from figure 27 that there are primarily corrections close to the measurement point. As the analysis point moves further away from the measurement, there is a lower correction. These are expected results from a Kalman filter.

When running through the simulation, it was experienced that the MOSEK optimization did not converge at specific time points, and an error was received that the solution was not optimal. The code would then stop running, which prevented further experimentation. When this occurred, a fail-safe mechanism was implemented where the reparametrized model was set to be the a posteriori state. The reparameterized field was then set to zero. This effect can be seen in 28. The error, in this case, was representative of the whole field. The error messages were received at different time points when running the simulation multiple times, concluding that the effect is non-deterministic and could be caused by software or hardware in computation. The errors received happened in the $t \in [11, 17]$ range.

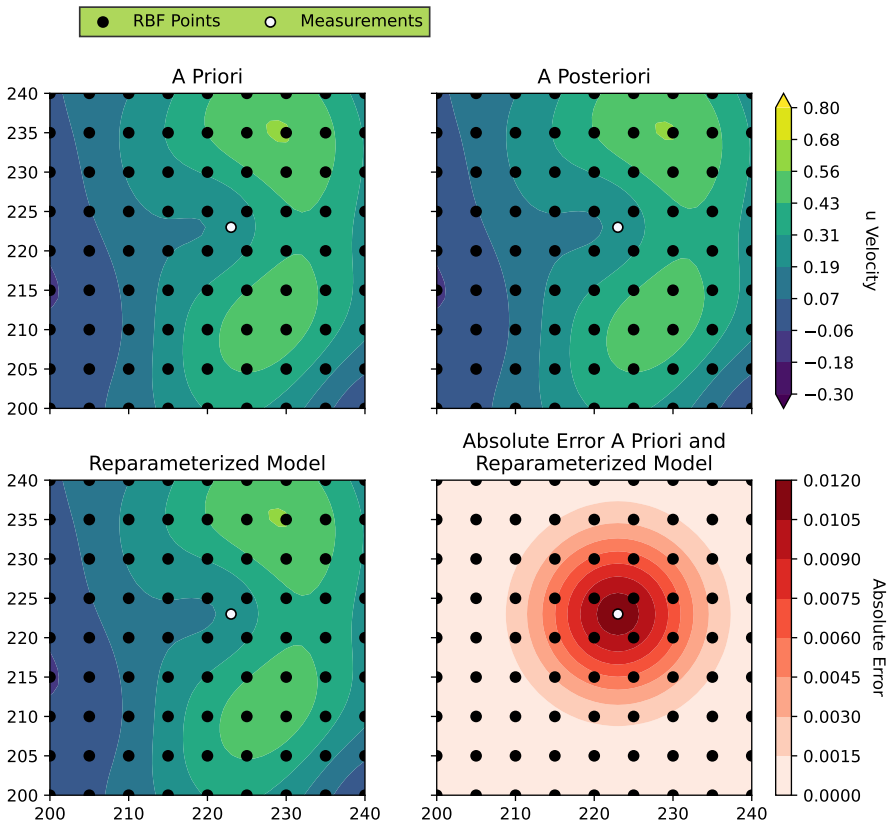


Figure 27: Kalman Filter A Priori, A Posteriori, Reparametirized Model Based on A Posteriori States and Absolute Error Between A Priori and Reparametirized Model for Case I at $t = 5$

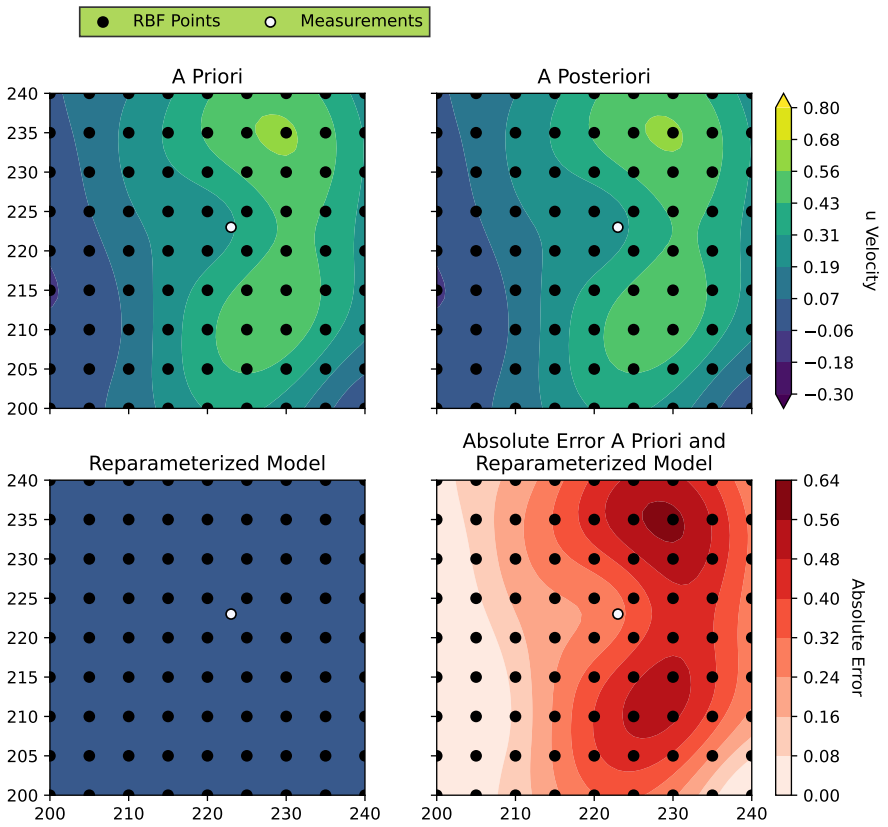
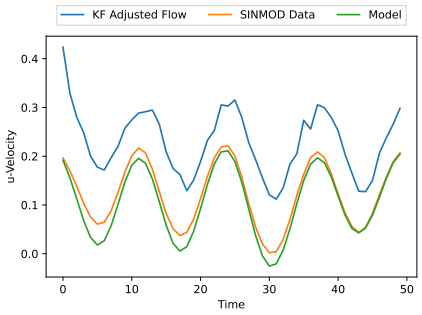
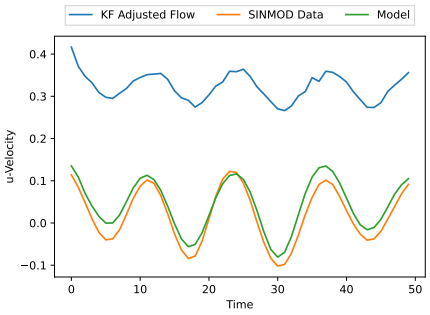


Figure 28: Kalman Filter A Priori, A Posteriori, Reparametirized Model Based on A Posteriori States and Absolute Error Between A Priori and Reparametirized Model for Case I During Error at $t = 14$

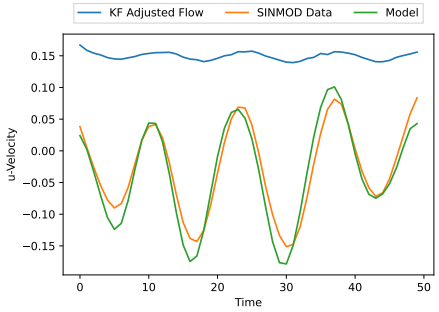
A final analysis was completed by plotting the Kalman filter output at three different points: The measurement point, an RBF point, and an arbitrary point which was neither. Getting these measurements means multiplying the a posteriori states with the Gaussian kernel. This was compared to the SINMOD data and initial model developed in 5.2 and 5.3. Figure 29 shows similar results to figure 27, where larger corrections to the states are completed closer to the point of measurement.



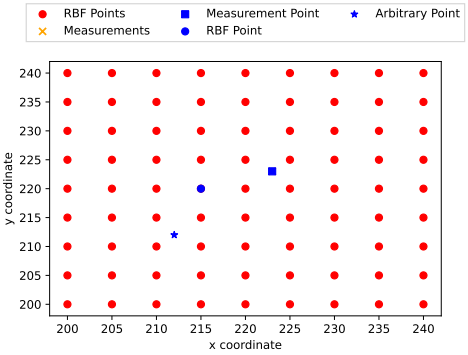
(a) Measurement Point, (x, y, z) = (223, 223, 1)



(b) RBF Point, (x, y, z) = (215, 220, 1)



(c) Arbitrary Point, (x, y, z) = (212, 212, 1)

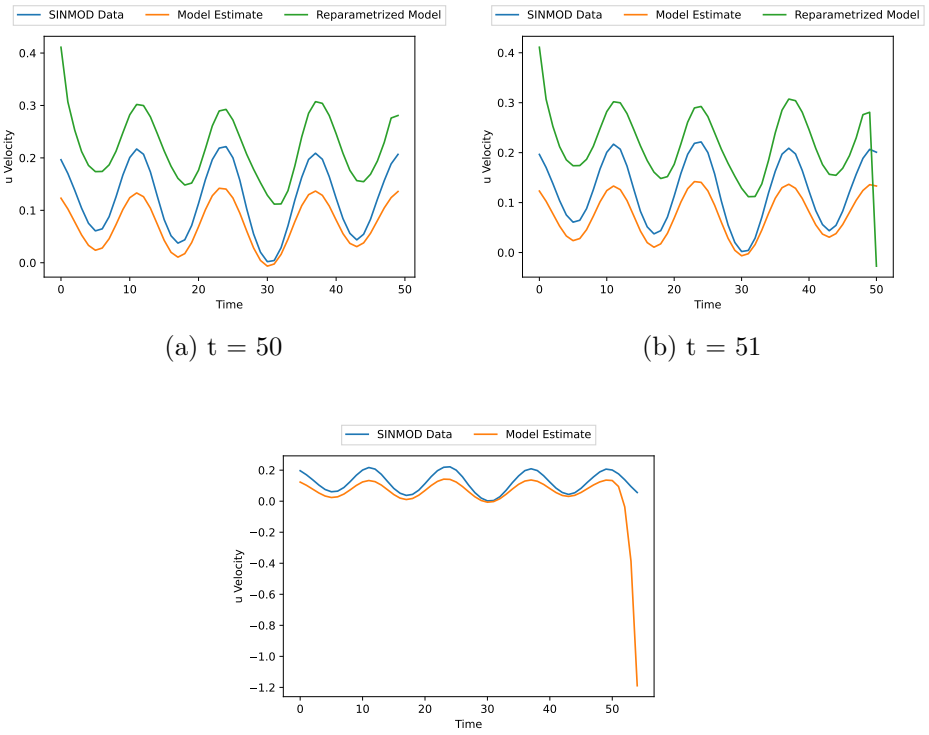


(d) Measurement Point, RBF Point and Arbitrary Point Locations

Figure 29: SINMOD Data and Kalman Filter (KF) Adjusted Point Values at Measurement Point, RBF Point and Arbitrary Point With Locations for Case I

One can see that the arbitrary point in 29c is the furthest away and hence has the least correction. The measurement point follows the graph more closely.

5.7.1 Case I Timespan Analysis



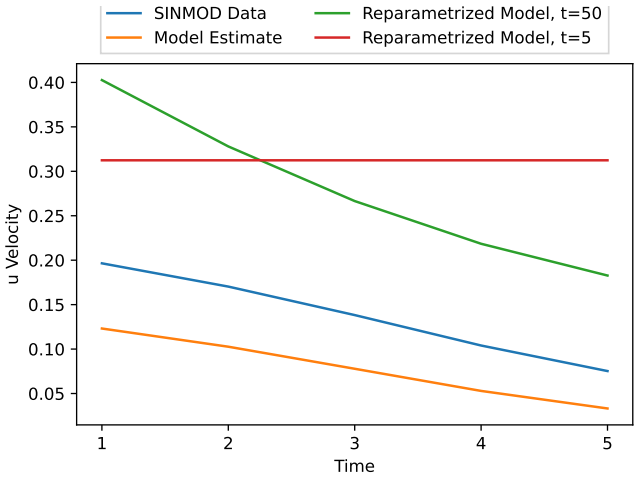
(a) t = 50

(b) t = 51

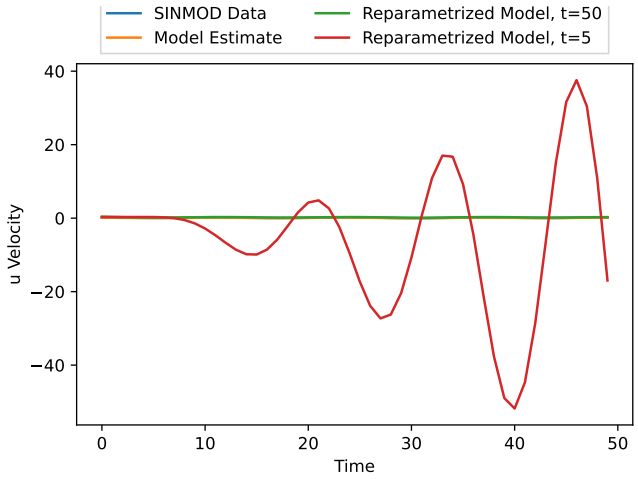
(c) t = 55

Figure 30: Timespan Analyses for (x, y, z) = (223, 223, 1)

Figure 30 shows how the Kalman filter and initial model react to forecasting. The initial, or static, model (orange) and reparameterized, or dynamic, model (green) are in figure 30a plotted and compared for timepoints up to 50. The reparameterized model is based on the a posteriori states from the previous 50 time points. Both graphs follow the SINMOD data well up to 50; however, at one timepoint beyond, the dynamic model diverges and does not fit the data anymore, which can be seen in 30b. The same thing happens to the static model in figure 30c at around a time point of 52. These divergences shows that the models are not a good fit for time points beyond the measurements.



(a) t = 5



(b) t = 50

Figure 31: Timespan Analyses for Reparametirized Model at $t = 5$

For the second analysis, the reparameterization model was optimized on a posteriori states for up to $KT = 5$. The static model, and the model in the previous experimentation at $KT = 50$ were tested for $t \in [1, 5]$ and $t \in [1, 50]$ and compared with the SINMOD data. Results can be seen in figure 31a. In 31a, the reparameterized model for $KT = 5$ looks to have an almost constant value; hence, the fit is not representative of the flow field. Figure 31b shows that the model diverges from the initial values at higher timepoints. This divergence means that the reparameterization for lower timepoints is under-scaled and not a good representation of the flow field.

5.8 Case Study II - Measurements Exceeding RBF Points

Using more measurements than RBF points yielded a more even update across the field. Figure 32 shows the a priori, a posteriori, and reparameterized model. Also shown is the absolute error between the a priori and reparameterized model. Compared with only one measurement, the a posteriori looks more similar to the a priori state. A higher correction across the field can be seen in the magnitude difference of the absolute error between figure 32 and figure 27.

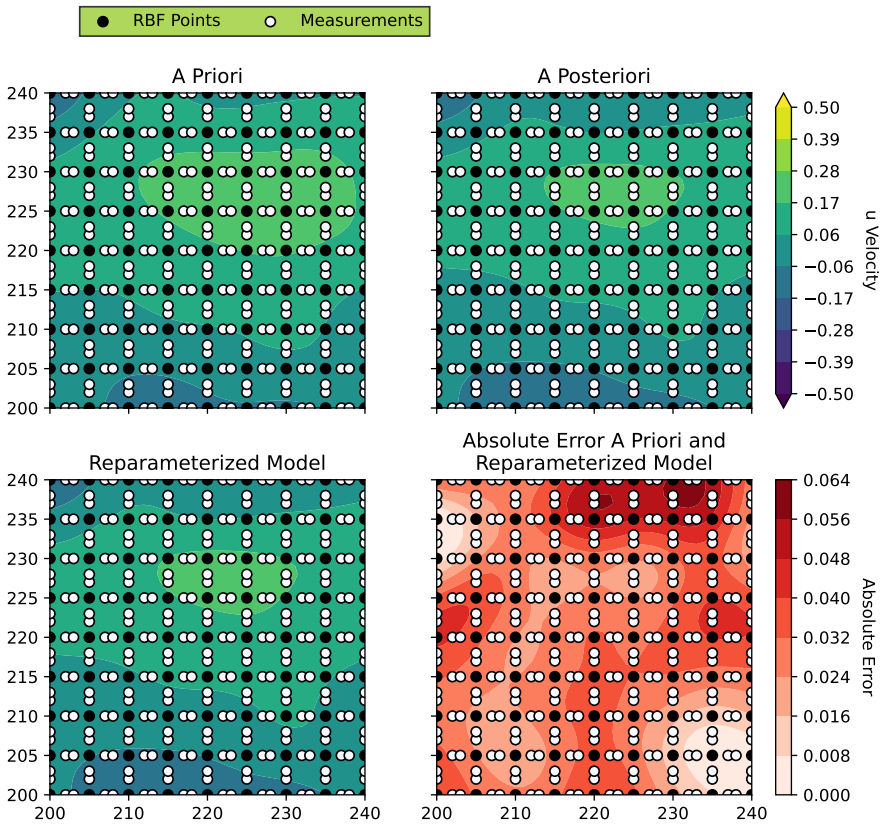
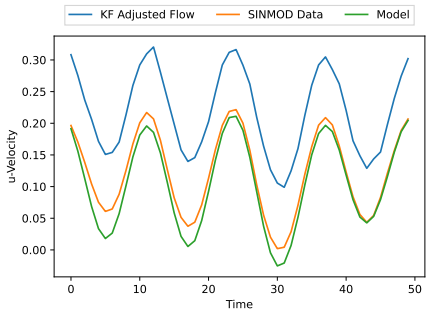


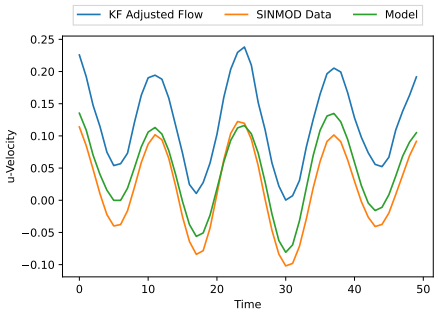
Figure 32: Kalman Filter A Priori, A Posteriori, Reparametirized Model Based on A Posteriori States and Absolute Error Between A Priori and Reparametirized Model for Case II at $t = 5$

For case II, the algorithm still had convergence issues, and the parameters did not reach an optimal solution at multiple time points. These errors were observed in the range $t \in [13, 32]$.

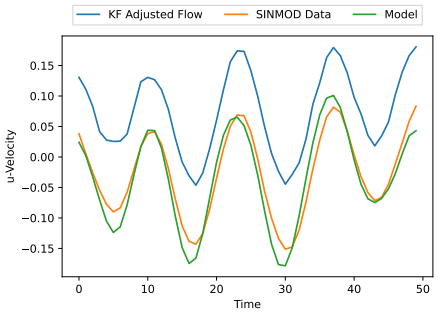
The Kalman filter output is compared with the SINMOD data and initialized model developed in 5.2 and 5.3 at three different points with locations seen in figure 33d. Using more measurements led to higher corrections of graphs at all points as opposed to only one measurement in figure 29 , which had the highest effect at the measurement point.



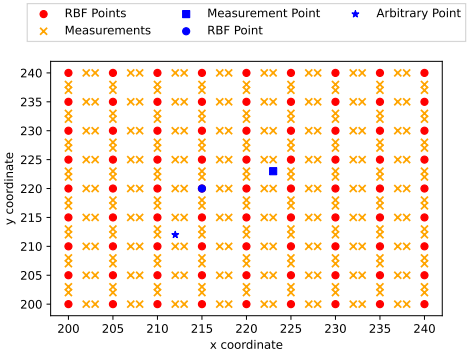
(a) Measurement Point, (x, y, z) = (223, 223, 1)



(b) RBF Point, (x, y, z) = (215, 220, 1)



(c) Arbitrary Point, (x, y, z) = (212, 212, 1)



(d) Measurement Point, RBF Point and Arbitrary Point Locations

Figure 33: SINMOD Data and Kalman Filter (KF) Adjusted Point Values at Measurement Point, RBF Point and Arbitrary Point With Locations for Case II

5.9 Case Study III - Dyanmic Underwater Drones

For the final case study, OBNs provided measurements dynamically throughout the grid. A priori, a posteriori, reparameterized model and the absolute error between the a priori state and reparameterized model can be seen in figure 34 at $t = 10$ and $z = 1$. Like case study I, which used one measurement, the absolute error plot shows the most correction happening at the measurement point. At $t = 10$, the measurement is at $(x, y) = (218, 218)$.

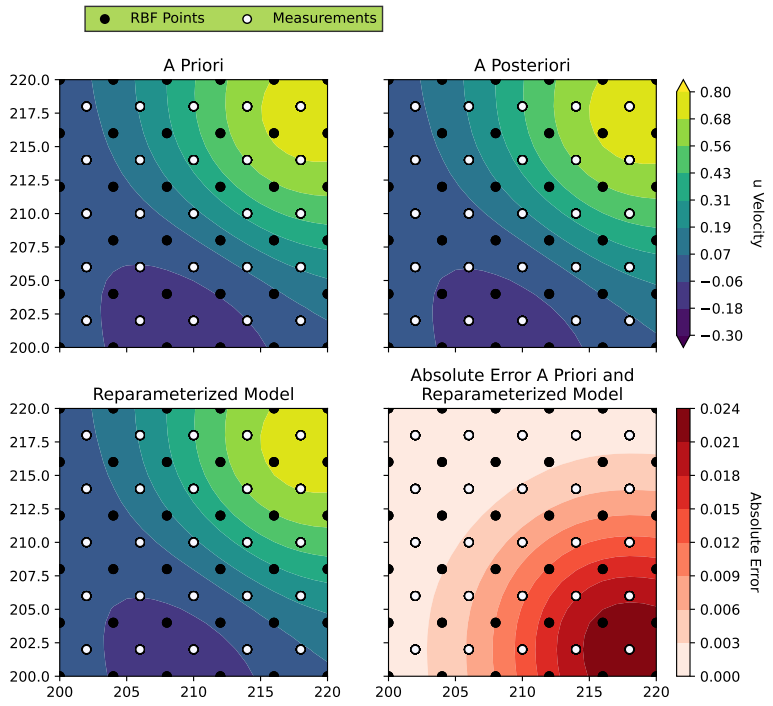
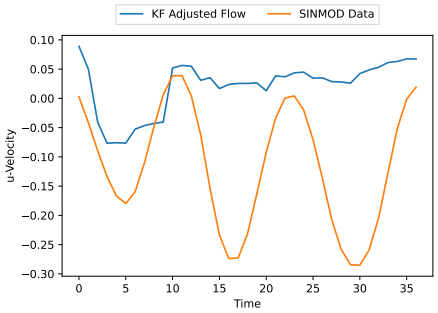


Figure 34: Kalman Filter A Priori, A Posteriori, Reparametirized Model Based on A Posteriori States and Absolute Error Between A Priori and Reparametirized Model for Case III at $t = 10$

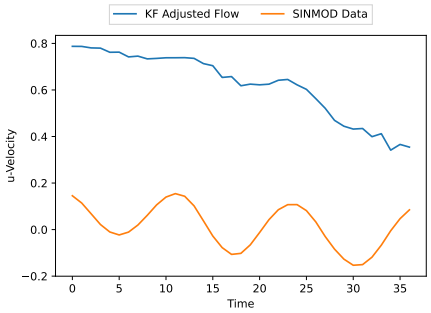
The reparameterization for this case did not converge for timepoints $t \in [13, 30]$. Since the simulation runs for 36 timesteps, this represents a large portion of the operational time and could hinder the effective use of the reparameterized model.

Plots were also created to showcase the development of the model at the first, the last, and a middle drone drop location. For the first drop location, the Kalman filter adjusted values follow the SINMOD data more closely at

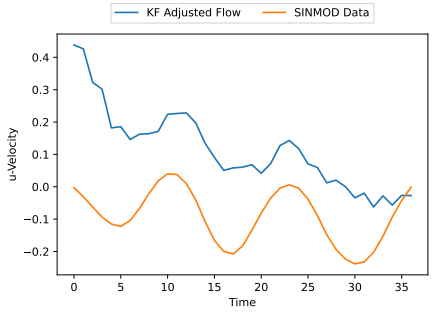
lower timesteps meaning measurements are closer to the point. The data becomes less accurate as time progresses and measurements are made further away. The middle drop location is evenly adjusted and seems to follow the trend of the SINMOD data more closely than the other two. The last drop location has little to no following of the SINMOD data throughout the time series.



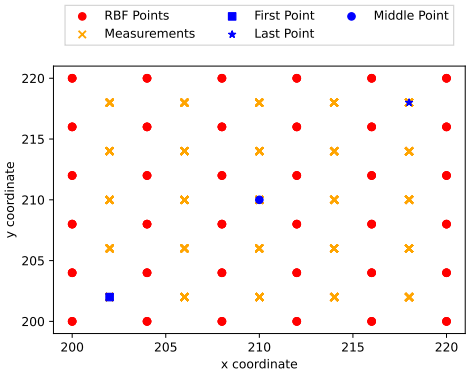
(a) First Drop Location, (x, y, z) = (203, 203, 1)



(b) Last Drop Location, (x, y, z) = (218, 218, 1)



(c) Middle Drop Location, (x, y, z) = (210, 210, 1)



(d) Measurement Points, RBF Points and Drone Drop Locations

Figure 35: SINMOD Data and Kalman Filter (KF) Adjusted Point Values at Drop Location 1 and 2 With Locations for Case III

Like the other cases, the model receives higher corrections the closer it is to the measurement. One can also see that there is little to no correction for the first and last drop locations. These are both far away for most of the measurement points. This inaccuracy could be harmful if the drones are dependent on far-away measurements. However, if the drones are released next to each other, they receive input from nearby measurements, and the model could be efficient in making local adjustments to the model. Further testing on this efficiency was not researched in this paper.

CHAPTER 6

Discussion

6.1 Model Fitting Functions

Throughout this project, a linear combination of sines and cosines as well as Laguerre polynomials of different degrees has been used to model the data from SINMOD and the a posteriori Kalman filter states. Higher-degree Laguerre polynomials were chosen due to them fitting the data accurately and their common use in modelling non-periodical natural data. At certain time points during testing, the methods did not converge. It can be theorized that this was due to the divergence of Laguerre polynomials at higher degrees. The choice of other functions could lead to higher accuracy and faster convergence. Examples of divergence are figures 30 and 31 which show how Laguerre polynomials could lead to exponential divergence at higher time steps. Constant functions were tested during the project thesis leading up to this master thesis, however these were not as efficient in modelling. Choosing other functions which do not converge as rapidly at higher time points could help decrease this effect.

6.2 Gaussian Kernel

The Gaussian kernel is an essential part of the updated model and makes us able to calculate values from the RBF points. As seen in 5.4, there is a potential risk of overlap if the scaling factor becomes too high, resulting in a higher value than the initial RBF points. This can also be seen in 3.1, where over- and underscaling was present at different λ values. These results are important because it says something about how scaling of the Gaussian kernel provides different results. To correct for a wrong scaling factor in the initial model, a second optimization was put into place taking in data from the whole field via the SINMOD simulation. However, when implementing a Kalman filter we are only given the a posteriori states of the RBF points and cannot do a second optimization over the whole field. This could hence lead to an over- or underscaling of the model depending on initial values of λ.

A way to correct for scaling in the reparametirized model was not explored in this paper. However, a proposed solution could be to find a scaling factor for each timepoint based off SINMOD data, computed via methods in 4.2.3, and multiply the a posteriori states with this. This would however require a great deal of processing power at larger data sets and could defeat the purpose of preserving computational power. Additionally, the efficiency of this method would need to be tested.

6.3 Kalman Filter

The Kalman filter builds upon the state-space model, and in this thesis, the state transition and control input matrices are identity matrices. These matrices are set up like this because states are extracted from the reparametrized model rather than system equations. The state transition matrix could be computed based on historical data of the ocean and its currents to try to compute a new value [12]. This might prevent issues regarding optimization and convergence of reparametrization because a priori states would not come from the reparametrized model.

The H matrix in the filter was composed of Gaussian functions at each point of measurement. As seen in 6.2, using a Gaussian kernel could be problematic due to its sensitivity and could provide inaccurate results if not tuned properly. Depending on the need, one could choose H to be another measurement kernel, either using displacement as a parameter or other parameters such as time, but this would most likely require tuning. The conclusion in 3 was that the optimal resolution and λ value is dependent on the available resolution and data set. Therefore, the Kalman filter could have received better results if the optimal λ value was found and used in the Gaussian functions. This option was nonetheless only an afterthought following experimentation. If the reparameterization of variables somehow accounted for the scaling factor, most of this uncertainty would be resolved. How this could be done was not researched further in this paper.

Q and R were in this project set to identity matrices of magnitude 0.001. However, these will need to be measured in real-life applications from sensor data and general model noise. The P matrix was also initialized as an identity matrix of magnitude 0.001. It could be improved if the values depended on the distance between each RBF point. This distance dependency was written about in a previous research paper by Berget et al. [22], where a Matern kernel was used containing the Euclidean distance as input and decaying covariance with the distance between points.

6.4 Convergence of Reparametrization

For specific time points during the Kalman filter iteration, the reparameterization for the model did not converge, and an error message was received from the MOSEK solver. The error messages informed that the solution was not optimal. There might be several reasons for this. Interestingly, this error did not occur for optimizing the initial model. One theory is that the model is not set up to assure convergence, and the values become too high for the solver. This divergence could be solved by putting a constraint on the parameters, making divergence less likely. Another possible pitfall could

be the processing power of the computer. Since the initial model did not have this issue, one could also theorize that the inclusion of the Kalman filter used more processing power, leading to non-convergence. When running the same simulation multiple times, it was observed that the error message occurred at different time points. A possible reason for this could be the random noise that affects convergence.

The programming language Python uses more computer memory than, for example, object-oriented programming. A possible solution to this memory issue could be to use functions that do not scale up as much as Laguerre polynomials, to use another programming language, or to use more effective hardware.

Convergence error messages were not received early or late in the optimization process but rather in the middle. Since the data is cumulative throughout the optimization, smaller amounts of data points are available during the early stages. It could be that the model requires a set amount of data points before the method assures convergence and under-fits before this point is reached. Figure 31a shows proof of underfitting, which supports the argument.

6.5 Placement of RBF Points

In this project, the RBF points were placed equidistant throughout the grid. However, different placement of the RBF points could lead to better results. For example, placing them along the trajectory of the underwater drone means that the RBF points will get Kalman filter adjustments which are closer and hence have a higher impact. However, this placement faces many of the same challenges faced by the equidistant setup. If an RBF point is predicted inaccurately, it could have dire consequences for the drone's trajectory.

Another placement suggestion would be to have a movable grid, presented in Chang 2014 [12]. In it, a model is initialized within a grid, and as the drone moves outside of it, the grid changes location. Reparametrization is therefore reset each time the grid is moved. It is also noted that the H matrix needs to be square, meaning there need to be as many measurements as there are RBF points. These measurements would require many devices, meaning more drones or a larger grid. More drones could lead to higher costs, and a larger grid could lead to an accuracy drop-off. There is, however, no justification for having H be a square matrix. It can be theorized that a lower amount of measurements could lead to equally fast convergence.

6.6 Future Predction

Most of the analyses completed were over a set timeframe; however, one of the purposes of this project is to predict the flow to be used in OCEANID underwater drones. Looking at figure 30, we see that neither the reparameterized model nor the initial model are reasonable estimates at higher timepoints. From 30b, we see that the reparameterized model falls off quickly and is not suitable for future predictions. This deviation could be due to Laguerre polynomials which increase exponentially at higher timesteps. This exponential divergence can be seen in both the initial and reparameterized models. A solution to this could be to use different functions which do not scale as highly with higher timesteps—alternatively, using lower degrees of Laguerre polynomials. Modification could also be implemented to use constraints on some of the functions such that they do not go way out of bounds, as discussed in 6.1.

From figure 31, the method does not converge to the solution with too few data inputs in a smaller timespan. When testing for larger time values, the model diverges from the static and dynamic models, which use parameters at timepoint 50. This divergence means that the reparameterized model at lower timesteps is a poor estimation of future results, and more data is needed to create an accurate fit. When the model is to be used in OCEANID, it might need to be initialized with a couple of measurements beforehand to get accurate reparameterization. Another solution could be to take measurements one at a time, running through the Kalman filter and getting more a posteriori states for reparameterization, not per timestep, but measurement. This paper did not explore this option but could provide a viable solution.

6.7 Measurements and Sampling Frequency

A large part of the updated model depends upon the measurements and sampling frequency. More accurate measurements lead to more accurate updates of the a posteriori state and a higher chance that the node will reach its desired destination. It has been mentioned that the reparameterized model requires a certain amount of inputs in order to portray the flow field accurately. A higher sampling frequency could shorten the time it takes to receive these. One of the goals for the algorithm in 1.1 was that the Kalman filter was supposed to run at an update frequency of 15 minutes. Obtaining measurements at a higher frequency means that the model would be able to update more accurately when running through the Kalman filter step. Still, it took the drone 6 minutes to transverse 1000m depth, so a higher update frequency might yield better results.

6.8 Case Results

The first two cases in this paper focused on one measurement received and more measurements than RBF points received. These analyses show that more measurements led to more model updates across the flow field. Employing more drones, or employing more drones with measurement devices, would lead to a more accurate result. However, this expansion could lead to higher costs due to more equipment and boats dropping drones being required or more acoustic measurement sensors required.

A case study inspired by the drone requirements from section 2.2 was created to showcase how updating the model works in a dynamic environment. This experimentation showed that adjustments to the model depend on the distance from the measurements. The result means that drones dropped after measurements are made at a neighboring point receive substantial adjustments to the model. These adjustments would help prediction in drones dropped directly afterward. How accurate this update would also depend on the factors discussed previously in this section, such as sampling frequency and RBF placement. Future work on this case would include realistic measurements and further testing of the updated model. This paper initialized the project; however, many aspects require analysis before the model can be put into use.

CHAPTER 7

Conclusion

A method for modeling ocean current was created using RBFs and a Gaussian kernel. Initialization was completed using SINMOD data at the RBF points and a second scaling optimization over the flow field. Updates to the model were made possible by implementing a Kalman filter. Afterward, the initial functions were reparameterized from available information.

This thesis aimed to develop an ocean current model for underwater drone positioning. Cases were completed for different scenarios which showcases the adjustment of grid to measurements using a Kalman filter. This highlighted that further work is needed in the development of the algorithm and potential restrictions. In addition there are a multitude of different alternatives when it comes to adjusting parameters and methods. No conclusion is drawn on the 2.5 m horizontal accuracy specification and further work is recommended including field testing to test the method.

The information and analysis found in this paper lay the groundwork for future experimentation and development of the algorithm.

CHAPTER 8

Recommendations for Further Work

8.1 Improving Minimization Problem

Convergence issues are closely related to the setup of the minimization problem. Section 6 goes through different pitfalls, including the use of Laguerre polynomials. At larger timesteps, these can become exponentially high and cause divergence. Tests should be completed using different types of estimation functions. Additionally, boundaries can be set on parameters to be optimized to avoid non-converging optimization. Other changes can be made to the minimization problem, which could improve the stability and performance of output. However, these were the ones discussed in this paper.

8.2 Point Estimation

A Gaussian kernel with equally spaced RBF points was used to get the point estimations. The Gaussian kernel can have scaling issues, so alternatives should be explored. The placement of the RBF points could also potentially affect final results, and most likely, the two are closely tied.

8.3 Kalman Filter

In 6.3, the idea of using historical data was presented. The matrices in the state-space system of the Kalman filter influence the results. The H matrix can also be changed depending on which kernel is used to see the effects.

8.4 Field Testing

A final note is made that all data collected in this research is synthetic or extracted from SINMOD simulations. For the project to be realized, real-life data is required to initialize and test the model. This data includes current data, Q and R matrices, and resulting trajectories.

References

[1] Research Council of Norway. idrop oceanid™ navigator (idrona). https://prosjektbanken.forskningsradet.no/en/project/FORISS/310157?Kilde=FORISS&distribution=Ar&chart=bar&calcType=funding&Sprak=no&sortBy=date&sortOrder=desc&resultCount=30&offset=60&Departement=Arbeids-+og+sosialdepartementet. Accessed: 2022-06-06.

[2] Morten Omholt Alver. Introduction to ocean dynamics measurement and modelling, 2021.

[3] Center for Operational Oceanographic Products and Services. About harmonic constituents. https://tidesandcurrents.noaa.gov/about_harmonic_constituents.html. Accessed: 2022-04-01.

[4] Center for Operational Oceanographic Products and Services. Harmonic constituents for 9410170, san diego, san diego bay ca. https://tidesandcurrents.noaa.gov/harcon.html?id=9410170. Accessed: 2021-01-23.

[5] The Editors of Encyclopaedia Britannica. viscosity. ://www.britannica.com/science/viscosity. Accessed: 2022-04-01.

[6] SINTEF. Sinmod. https://www.sintef.no/en/ocean/initiatives/sinmod/. Accessed: 2021-25-09.

[7] Dag Slagstad and Thomas A. McClimans. Modeling the ecosystem dynamics of the barents sea including the marginal ice zone: I. physical and chemical oceanography. *Journal of Marine Systems*, 58(1):1–18, 2005.

[8] iDrop. idrop projects. https://www.idrop.no/projects. Accessed: 2021-01-11.

[9] GCE Ocean Technology. Oceanid™ – the future of exploration seismic. https://www.gceocean.no/news/posts/2020/september/oceanid-the-future-of-exploration-seismic/. Accessed: 2021-01-11.

[10] David Lowe D.S. Broomhead. Radial basis functions, multi-variable functional interpolation and adaptive networks. 1988.

[11] M. D. Buhmann. *Acta Numerica, Radial basis functions.* Cambridge University Press, 2000.

[12] W. Wu F. Zhang. Real-time modeling of ocean currents for navigating underwater glider sensing networks. 2013.

[13] Ramani Duraiswami Nail A Gumerov. Fast radial basis function interpolation via preconditioned krylov iteration. page 1876–1899, 2007.

[14] Roger R. Labbe. *Kalman and Bayesian Filters in Python*. 2015.

[15] Roger R. Labbe. *Kalman and Bayesian Filters in Python*, chapter Chapter 1: The g-h Filter. 2015.

[16] Discretised. What is convolution? this is the easiest way to understand.

[17] AKHILESH GANTI. Central limit theorem (clt). `https://www.investopedia.com/terms/c/central_limit_theorem.asp`. Accessed: 2022-08-06.

[18] Hans Fischer. *A History of the Central Limit Theorem*. 2011.

[19] Hossein Pishro-Nik. Introduction to probability, statistics and random processes, 3.2.2 expectation.

[20] Carsten Antonio Bing. Data-driven Current Model as Part of a Digital Twin for Submarine Drones. 2021.

[21] Eric W. Weisstein. Frobenius norm. ://mathworld.wolfram.com/FrobeniusNorm.html. Accessed: 2022-08-06.

[22] Gunhild Elisabeth Berget, Trygve Olav Fossum, Tor Arne Johansen, Jo Eidsvik, and Kanna Rajan. Adaptive sampling of ocean processes using an auv with a gaussian proxy model. *IFAC-PapersOnLine*, 51(29):238–243, 2018. 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018.