

SEMIBLIND CALIBRATION OF GAIN AND DRIFT IN  
A SENSOR NETWORK USING GAUSSIAN PROCESSES

by  
Zachary Bastiani

A Senior Thesis Submitted to the Faculty of  
The University of Utah  
In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science

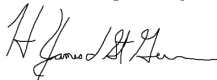
In  
Computer Science

Approved:



Ross Whitaker  
Thesis Faculty Supervisor

DocuSigned by:



72F18627B2034A8...  
Jim de St. Germain  
Director of Undergraduate Studies

DocuSigned by:



72F18627B2034A8...  
Mary Hall  
Director, School of Computing

December 2021

Copyright © 2021

All Rights Reserved

# 1 Acknowledgements

I would like to express my gratitude towards my advisor, Professor Ross Whitaker, for his teachings, knowledge, time, and trust he gave me. Furthermore, I would like to thank him for his continuous effort to instill good coding practices, consistent vocabulary, and attention to detail in me.

Thanks to the National Science Foundation for funding the sensor network used in this work. This work is supported by the National Science Foundation under grant no. CNS-1646408.

Besides my advisor, I want to thank Wenzheng Tao and Professor Wei Xing for their assistance and time spent helping me.

Lastly, I want to thank my significant other, Ailish Harris, for her support during the long nights and my parents, Michael Bastiani and Maria Denise-Dearing, for their continual support in my education and mental well-being.

## 2 Abstract

This thesis develops a method for incorporating drift and gain estimates into a Gaussian process model PM2.5 with respect to space and time within the Salt Lake Valley. Currently, there is a low-cost, large-scale sensor network collecting PM2.5 data in the Salt Lake Valley. Each sensor within the network is susceptible to gain and drift. Furthermore, two high-accuracy, high-cost national weather stations are collecting PM2.5 readings, but they are reliable, i.e., not susceptible to noise, gain, or drift. Our goal is to use the sensor network in tandem with the national weather stations to build an accurate model of the PM2.5 in the Salt Lake City Valley.

In this thesis, we assume that gain is constant throughout the sensor network and that each sensor has its own drift that is changing in time. Furthermore, we used a second Gaussian process to model the drift for our sensor network. We evaluated the effectiveness of three different optimization strategies' ability to retrieve the underlying data from a synthetic data set. The first optimization strategy calculates gain and drift using an alternating fixed-point algorithm. The second optimization strategy uses an optimizer to find the gain while drift is changed to be dependent upon gain. Lastly, we progressed the second optimization strategy by also having the optimizer find the bandwidths of our kernel function, the third optimization strategy.

The tests on synthetic data showed the proposed optimization strategies can drastically outperform a naive Gaussian process with no corrections for gain and drift. Furthermore, we tested the proposed optimization strategies to determine if the number of sensors in the sensor network is more impactful than the number of ground truth points in reducing the average error in the whole synthetic data set. We then evaluated the effects of noise and drift bandwidth on the quality of the model produced by the proposed optimization strategies. Lastly, we tested the model on the Salt Lake Valley sensor network. In conclusion, the proposed optimization strategies were able to, under supportive conditions, accurately calibrate the sensor network and were able to more accurately model the synthetic data than a naive Gaussian process under any reasonable condition.

Contents

1 Acknowledgements 1

2 Abstract 2

3 Introduction 4

4 Background 5

5 Related Work 7

6 Methods 9

6.1 Gaussian Process Models . . . . . 9

6.2 Priors on Gain and Drift . . . . . 12

6.3 Fixed-Point for Drift . . . . . 12

6.4 Fixed-Point for Gain . . . . . 15

6.5 Proposed Optimization Strategies . . . . . 17

7 Results 19

7.1 Overview of the Synthetic Data . . . . . 19

7.2 Ground Truth Sensors Versus Network Sensors . . . . . 22

7.3 Noise Levels . . . . . 24

7.4 Drift Bandwidth . . . . . 26

8 Conclusion 29

### 3 Introduction

Air pollution has become a common occurrence in the modern world. An important constituent of air pollution is fine particulate matter (PM2.5), which is all particulate matter less than or equal to 2.5 micrometers in aerodynamic diameter. PM2.5 was ranked as the sixth highest health risk factor, contributing to 4.1 million deaths worldwide in 2016 [1]. Measuring and modeling PM2.5 is paramount to more accurately understand the effects of PM2.5 on humans, and to helping people regulate their PM2.5 intake. Ideally, information about PM2.5 would become similar to a weather forecast with high accuracy in space and time.

To measure PM2.5 in the Salt Lake Valley, the NSF-funded, AirQ (NSF CNS-1646408) deployed a low-cost, large-scale sensor network. The sensor network is made up of  $N_s$  sensors located at a position  $x_i = [latitude, longitude, altitude]$  and collects  $N_t$  PM2.5 readings at times  $t_j$  that are uniformly distributed throughout a set interval. Thus, the sensor network will provide us with PM2.5 data at virtually every point  $(x_i, t_j)$ . However, as time progresses, sensor readings will drift from the ground truth if the sensors are not regularly calibrated. In addition, two national weather stations within Salt Lake City measure PM2.5 at the station locations  $x'_1$  and  $x'_2$ . These sensors are much higher quality than the sensors deployed in the network and are regularly calibrated. We assume that a smooth underlying process,  $f(x, t)$ , governs the PM2.5 concentration.

The goal of this thesis is to accurately estimate PM2.5 concentrations at arbitrary spatial and temporal points within Salt Lake City using the uncalibrated sensor network and the regularly calibrated national weather stations. The ability to efficiently calibrate a sensor network in Salt Lake City is key because of the fast-changing episodic pollution events that occur. During the winter, inversions can cloud the Salt Lake Valley sky for weeks on end, but can dissipate within a few hours with a snowstorm. The calibrations for the sensors during and after the inversion need to be different to accurately measure the PM2.5 in the valley. By building a method to automatically calibrate the sensor network, we will be able to more accurately monitor/model PM2.5 in the Salt Lake Valley.

## 4 Background

The main method used in this thesis is the Gaussian process. “The Gaussian process is a generalization of the Gaussian probabilistic distribution”[2]. It allows us to take an input of points,  $X = \{x_1, \dots, x_n\}$ , the observed data at those points,  $\mathbf{Y}$ , and model the underlying function,  $f(x)$ , as a generalized from Gaussian distribution. In order to use the Gaussian process, we first need to zero (center) the observed data, which will simplify the equations and is often a good first step. The model built by the Gaussian process can have the mean added back into the estimated data set to accurately model the initial observed data.

$$f(X) \sim \mathcal{GP}(0, K(X, X) + \sigma^2 I) \quad (1)$$

$$y_i = f(x_i) + \epsilon_i \quad (2)$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2) \quad (3)$$

The kernel function,  $K(X, X)$ , which determines the co-variance between the index set,  $X$ , in the Gaussian process. The kernel function,  $K(x_i, x_j)$ , can be thought of as the correlation between any two points,  $x_i, x_j$ , signifying how correlated their  $\mathbf{Y}$  values are. The Gaussian process also is able to handle random noise,  $\epsilon$ , in the system by adding the variance of the noise to the kernel matrix. The noise is modeled by a normal distribution, equation 3. By altering the kernel matrix, the noise adds uncertainty to the  $\mathbf{Y}$  values, which causes the Gaussian process to prefer building a smoother model, rather than passing directly through the observed data point. Kernel functions should be selected based on the properties of the underlying function that is being modeled, because this function controls the properties of the final model, such as smoothness. For instance, using the Gaussian process to model a parabolic function requires a kernel that builds a parabolic model. The matrix that the kernel function builds must be invertible. We use the squared exponential kernel function, equation 4, because it builds a smooth model and it estimates the mean at points that have low correlation to the sampled points,  $X$ . By estimating the mean at points with low correlation to the measurements, the Gaussian process will be making a safe guess if data is normally distributed. The squared exponential kernel is also a standard function to use in machine learning. Lastly, this kernel function also gives us control over the bandwidth,  $\theta$ , of the model. The bandwidth controls the rate at which the correlation diminishes in space and time. For example, if the bandwidth is large, there will be very little change in the data.

$$K(x_i, x_j) = \exp\left(-\frac{(x_i - x_j)^2}{2\theta^2}\right) \quad (4)$$

The Gaussian process can be used to estimate values at unknown points. This method is called Gaussian process regression or Kriging. Kriging is based on building a joint probability distribution between the known Gaussian process and the unknown point  $F^*$ . The joint probability distribution corresponds to conditioning the Gaussian process on the data  $\mathbf{Y}$ , which gives the equations for the mean and variance at  $F^*$ .

$$\begin{bmatrix} \mathbf{Y} \\ F^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(X, X) + \sigma^2 I & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right) \quad (5)$$

$$\text{Mean at } F^* = K(X^*, X)(K(X, X) + \sigma^2 I)^{-1} \mathbf{Y} \quad (6)$$

$$\text{Variance at } F^* = K(X^*, X^*) - K(X^*, X)(K(X, X) + \sigma^2 I)^{-1} K(X, X^*) \quad (7)$$

In order to find the best model for a given data set, we need to be able to quantify the quality of a Gaussian process. The quality is calculated by the likelihood of the Gaussian process, equation 6. The likelihood of the Gaussian process comes from marginalizing out  $f(X)$  and from observing  $\mathbf{Y} \sim \mathcal{N}(0, K(X, X) + \sigma^2 I)$ , which is true only if the observed data has been zeroed. The likelihood equations returns a value quantifying the quality of the Gaussian process. The quality of a Gaussian process is representative of two features: how well it fits the kernel properties, how well it passes through  $\mathbf{Y}$ .

$$\ln p(\mathbf{Y}|X) = -\frac{1}{2} \ln |K(X, X) + \sigma^2 I| - \frac{1}{2} \mathbf{Y}^\top (K(X, X) + \sigma^2 I)^{-1} \mathbf{Y} - \frac{N}{2} \ln(2\pi) \quad (8)$$

Researchers have used the Gaussian process with noise to model PM2.5 levels. At least two papers that used the Gaussian process were published in recent years; *Community-based measurements reveal unseen differences during air pollution episodes*[3] and *Patterns of distributive environmental inequity under different PM2.5 air pollution scenarios for Salt Lake County public schools*[4]. The Gaussian process used in paper 2 assumed that the kernel function of the Gaussian process was separable in space and time to reduce the run time. The proposed methods also make a similar assumption based on paper 2 to reduce the run time of several steps. While the primary focus of both papers was on the disparity in PM2.5 levels at specific locations in the Salt Lake Valley, they showed the need for accurate PM2.5 models and the accuracy of current models.

## 5 Related Work

Several methods have been proposed to calculate drift and/or gain in a sensor network. Maen, Subhash, and Rabjib used interpolation and Kalman filtering to solve for sensor drift [5]. They built two functions to calculate drift: one assumes that the drift is smooth through time, and the other assumes that drift is nonsmooth through time. Both functions are dependent upon being able to estimate the ground truth at a given sensor point.

$$f(x_i, t_j) = f(\text{neighboring data}) \quad (9)$$

For both functions,  $f(\text{neighboring data})$  is set to be the average of the neighboring sensors' reported values. The data are input to the smooth drift correction algorithm using a Kalman filter to create the function for calculating drift when the drift is smooth through time. When the drift is nonsmooth through time, the data are input into the second function, which is an upgraded version of the first by utilizing the interacting multiple model algorithm.

Kumar, Rajasegarar, and Palaniswami used spatial Kriging and Kalman filtering to estimate drift [6]. In the Kriging method, drift is assumed to be smooth throughout time. The Kriging method is similar to the interpolation method, with the difference that  $f(\text{neighboring data})$  is a Kriging algorithm otherwise known as a Gaussian process regression. Furthermore, the authors proved that the Kriging method is able to decrease the error in the system when compared to the interpolation method. Both the Kriging method and the interpolation method are blind calibration. In this thesis, we use Gaussian process regression without Kalman filtering, and with ground truth sensors, which the aforementioned methods did not use. Furthermore, the methods proposed in this thesis estimate a gain for the sensor network and a smooth drift for each sensor.

Becnel estimated sensor drift when measuring PM2.5 in the Salt Lake Valley using a weighted neighborhood calibration algorithm that incorporated recursive least squares (RLS) [7]. However, the RLS algorithm calculates a constant drift and gain for a sensor throughout the time interval selected. The RLS algorithm works by generating a graph of the sensors and then cycling through the sensor nodes using a breadth-first search starting at a ground truth sensor node and calibrating each node with a combination of RLS, sample correlation, and distance weight. For the RLS algorithm to work, the graph it generates must be constant in time, causing the algorithm to require every sensor to be spatially static. The RLS algorithm dealt with a similar sensor network located in Salt Lake City and measured PM2.5. While Becnel's method can estimate a constant gain and a constant drift, it does not have a statistical base. Meanwhile, the proposed methods



are based on statistical principles, which provide a high likelihood of accurately estimating the gain and drift.

The current methods have two key differences. Whereas both the Kriging method [6] and interpolation method [5] estimate drift as a function of time, the RLS method [7] estimates a drift that is constant in time. On the other hand, the RLS method also estimates a sensor gain, but the Kriging method and interpolation method disregard sensor gain. The Kriging method is a direct alteration of the interpolation method and therefore should be a more accurate method. All the methods have run times less than or equal to  $\mathcal{O}(N^3)$  to allow for scalability of the sensor network and are lacking in some aspect of the overall model in terms of estimating the gain and the drift of sensors. In comparison, the proposed methods also have a run time of  $\mathcal{O}(N^3)$ , but progress the modeling assumptions by incorporating a constant gain and a smooth drift.

## 6 Methods

### 6.1 Gaussian Process Models

The optimization strategies proposed here all use the Gaussian process as a backbone to build the model. Starting with equation 10, we assume the observation,  $y(x_i, t_j)$ , is mapped from the underlying signal,  $f(x_i, t_j)$ . From here, we build the Gaussian process that models the observed data. This Gaussian process incorporates only the observed data from the sensor network and has two undefined terms in it: the gain,  $\alpha$ , and the drift,  $\mathbf{B}$ .

$$y(x_i, t_j) = \alpha f(x_i, t_j) + b(x_i, t_j) + \epsilon \quad (10)$$

$$y(x_i, t_j) \sim \mathcal{GP}(\mathbf{B}, \alpha^2 K(X, X) + \sigma^2 \mathbf{I}). \quad (11)$$

$\mathcal{L}_1$  is the basic loss function for our Gaussian process with a couple of priors placed on it for the gain and drift.  $\mathcal{L}_1$  is built by calculating the log probability of the observed data,  $\mathbf{Y}$ , the gain,  $\alpha$ , and the drift,  $\mathbf{B}$ . Equation 12 is altered by using the assumptions that the observed data is dependent on the priors and that the priors are independent of each other to generate equation 13.

$$\mathcal{L}_1 = \ln(p(\mathbf{Y}, \alpha, \mathbf{B})) \quad (12)$$

$$= \ln(p(\mathbf{Y}|\alpha, \mathbf{B})p(\alpha)p(\mathbf{B})) \quad (13)$$

$$= \ln(p(\mathbf{Y}|\alpha, \mathbf{B})) + \ln(p(\alpha)) + \ln(p(\mathbf{B})) \quad (14)$$

$$= -\frac{1}{2} \ln |\hat{\Sigma}| - \frac{1}{2} (\mathbf{Y} - \mathbf{B})^T \hat{\Sigma}^{-1} (\mathbf{Y} - \mathbf{B}) - \frac{N}{2} \ln(2\pi) + \ln(p(\mathbf{B})) + \ln(p(\alpha)) \quad (15)$$

where,

$$\hat{\Sigma} = \alpha^2 K(X, X) + \sigma^2 \mathbf{I}. \quad (16)$$

In order to incorporate the ground truth sensors in the Gaussian process, we build a second likelihood equation,  $\mathcal{L}_2$ .  $\mathcal{L}_2$  is built by calculating the probability that the ground truth data points fit into the Gaussian process. Because the Gaussian process is built by assuming every point is normally distributed, we know that the probability of the ground truth data fitting into the Gaussian process is the same as it fitting into a normal distribution at that point. By using the equations for mean and variance at a point, we can find the normal distribution for the ground truth point. However, because the gain is altering the kernel of the sensor network and drift is altering the data, we need to redo the joint distribution to estimate the ground

truth points.

$$\begin{bmatrix} \mathbf{Y} \\ F^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix}, \begin{bmatrix} \alpha^2 K(X, X) + \sigma^2 I & \alpha K(X, X^*) \\ \alpha K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right) \quad (17)$$

$$\text{Mean at } F^* = \alpha K(X^*, X)(\alpha^2 K(X, X) + \sigma^2 I)^{-1}(\mathbf{Y} - \mathbf{B}) \quad (18)$$

$$\text{Variance at } F^* = K(X^*, X^*) - \alpha K(X^*, X)(\alpha^2 K(X, X) + \sigma^2 I)^{-1} \alpha K(X, X^*). \quad (19)$$

Using the new joint distribution, we show how  $\mathcal{L}_2$  is formed starting with equation 20:

$$p(y(x_i, t_j)) \sim \mathcal{N}(\mu(x_i, t_j), v(x_i, t_j)) \quad (20)$$

$$\mu(x_i, t_j) = \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1}(\mathbf{Y} - \mathbf{B}) \quad (21)$$

$$v(x_i, t_j) = \mathbf{k}_{**} - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \alpha \mathbf{k}_* \quad (22)$$

$$\mathcal{L}_2 = \sum_{i=1}^{N'_s} \sum_{j=1}^{N'_t} \ln p(y(x_i, t_j)) \quad (23)$$

$$\mathcal{L}_2 = \sum_{i=1}^{N'_s} \sum_{j=1}^{N'_t} \frac{1}{2} (-\ln(v(x'_1, t'_j)) - (y(x'_1, t'_j) - \mu(x'_1, t'_j))^2 / v(x'_1, t'_j) - \ln(2\pi)) \quad (24)$$

$$\mathcal{L}_2 = \sum_{i=1}^{N'_s} \sum_{j=1}^{N'_t} \frac{1}{2} (-\ln(\mathbf{k}_{**} - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \alpha \mathbf{k}_*) - \frac{(y(x'_i, t'_j) - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1}(\mathbf{Y} - \mathbf{B}))^2}{\mathbf{k}_{**} - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \alpha \mathbf{k}_*} - \ln(2\pi)). \quad (25)$$

By combining the likelihood of our Gaussian process,  $\mathcal{L}_1$ , with the likelihood of the ground truth sensors,  $\mathcal{L}_2$ , we build the maximum a posteriori (MAP) estimate for the Gaussian process as follows:

$$\hat{\mathcal{L}} = \mathcal{L}_1 + \mathcal{L}_2 \quad (26)$$

$$\begin{aligned} \mathcal{L}_1 = & -\frac{1}{2} \ln |\hat{\Sigma}| - \frac{1}{2} (\mathbf{Y} - \mathbf{B})^T \hat{\Sigma}^{-1} (\mathbf{Y} - \mathbf{B}) - \frac{N}{2} \ln(2\pi) \\ & + \ln(p(\mathbf{B})) + \ln(p(\alpha)) \end{aligned} \quad (27)$$

$$\mathcal{L}_2 = \sum_{i=1}^{N'_s} \sum_{j=1}^{N'_t} \frac{1}{2} (-\ln(\mathbf{k}_{**} - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \alpha \mathbf{k}_*) - \frac{(y(x'_i, t'_j) - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1}(\mathbf{Y} - \mathbf{B}))^2}{\mathbf{k}_{**} - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \alpha \mathbf{k}_*} - \ln(2\pi)) \quad (28)$$

where,

$$\hat{\Sigma} = \alpha^2 K(X, X) + \sigma^2 \mathbf{I} \quad (29)$$

$$\mathbf{k}_{**} = k([x'_i, t'_j], [x'_i, t'_j]) \quad (30)$$

$$\mathbf{k}_* = k([x'_i, t'_j], X). \quad (31)$$

Lastly, we determine the kernel function that the Gaussian process uses. As stated in the background, we use a squared exponential kernel function with a few changes. First, our squared exponential kernel is broken down into two separate kernels: time and space. In order to decompose the kernel, we made use of one of the initial assumptions made about the sensor network: that every sensor in the sensor network is taking a reading at the same time. Using this assumption means that we can break the points,  $X$ , into our sensors' location in space,  $x$ , and our sensors' readings in time,  $t$ . We then use the Kronecker product on the space and time kernel matrices to calculate the complete kernel matrix [4]. Because we use the Kronecker product, we can decompose the space kernel matrix and time kernel matrix down into their eigenvectors and eigenvalues using singular value decomposition,  $K(X, X) = USU^T$ , which gives the following inverse matrix statement from inside of the MAP estimate:

$$(\alpha^2(K(X, X)) + \sigma^2 I)^{-1} = (\alpha^2(K_x(X, X) \otimes K_t(X, X)) + \sigma^2 I)^{-1} \quad (32)$$

$$= (\alpha^2(U_x S_x U_x^T) \otimes (U_t S_t U_t^T) + \sigma^2 I)^{-1} \quad (33)$$

$$= (\alpha^2((U_x \otimes U_t)(S_x \otimes S_t)(U_x^T \otimes U_t^T) + \sigma^2 I)^{-1} \quad (34)$$

$$= (U_x \otimes U_t)(\alpha^2(S_x \otimes S_t) + \sigma^2 I)^{-1}(U_x^T \otimes U_t^T). \quad (35)$$

Because  $S$  is a diagonal matrix, as it holds the eigenvalues, it can be quickly inverted. By using the properties of the Kronecker product and eigendecomposition, we are able to reduce the run time of this step from  $\mathcal{O}(N^3)$  to  $\mathcal{O}(N_s^3 + N_t^3)$  because we change the rate limiting step from inverting the complete matrix to eigendecomposition of the space kernel matrix and the time kernel matrix [8]. Secondly, we have multiple bandwidths for the kernel functions. The space kernel uses two bandwidths and time kernel uses one. The two bandwidths for the space kernel control the altitude,  $\theta_a$ , and the horizontal,  $\theta_l$ , correlations. The separation of these bandwidths is key because the rate of change for pollution can be drastically different between the rate of change in longitude and latitude versus the rate of change in altitude. There also exists a  $\theta_0$  term multiplying the space kernel function.  $\theta_0$  is able to control the variance of the data in Gaussian

process model as given below:

$$K(X, X) = K_x(X, X) \otimes K_t(X, X) \quad (36)$$

$$K_x(x_i, x_j) = \theta_0 \exp\left(-\frac{(x_{i,\text{long}} - x_{j,\text{long}})^2}{2\theta_l^2} - \frac{(x_{i,\text{lat}} - x_{j,\text{lat}})^2}{2\theta_l^2} - \frac{(x_{i,\text{alt}} - x_{j,\text{alt}})^2}{2\theta_a^2}\right) \quad (37)$$

$$K_t(t_i, t_j) = \exp\left(-\frac{(t_i - t_j)^2}{2\theta_t^2}\right). \quad (38)$$

## 6.2 Priors on Gain and Drift

The MAP estimate at this point is partially incomplete, as we have yet to define the distributions for the priors. The  $\alpha$  represents the gain for the whole sensor network and is modeled by a scaled inverse chi-squared distribution. Scaled inverse chi-squared distributions have two variables that control the distribution:  $v, \tau^2$ .  $v$  is the degrees of freedom for the distribution and  $\tau^2$  controls the mean for the distribution, which is:

$$\alpha \sim \text{Scale-inv-}\chi^2(v, \tau^2) \quad (39)$$

$$p(\alpha) = \frac{(\tau^2 v/2)^{v/2} \exp(-\frac{v\tau^2}{2\alpha})}{\Gamma(v/2) \alpha^{1+v/2}}. \quad (40)$$

$\mathbf{B}$  is modeled by its own Gaussian process. The drift,  $\mathbf{B}$ , is a matrix representing the drift for each sensor at each point in time. Furthermore, we assume that the drift is unique for each sensor and correlated in time, which allows us to model the drift as its own Gaussian process by using a kernel that is correlated in time, but completely uncorrelated in space. This kernel function for the drift Gaussian process also has its own bandwidth,  $\theta_b$ , that controls the rate of change of the drift. Lastly, the drift kernel was given a  $\theta_v$  term that controls the variance of the Gaussian process, as below:

$$b(x_i, t_j) \sim \mathcal{GP}(0, K_b(X, X)) \quad (41)$$

$$K_b(X, X) = k_{bx}(x, x) \otimes k_{bt}(t, t) \quad (42)$$

$$k_{bx}(x, x) = \mathbf{I} \quad (43)$$

$$k_{bt}(t_i, t_j) = \theta_v \exp\left(-\frac{(t_i - t_j)^2}{\theta_b}\right) \quad (44)$$

$$\ln p(\mathbf{B}) = -\frac{1}{2} \ln |K_b(X, X)| - \frac{1}{2} (\mathbf{B})^T (K_b(X, X))^{-1} (\mathbf{B}) - \frac{N}{2} \ln(2\pi). \quad (45)$$

## 6.3 Fixed-Point for Drift

Here we formulate a fixed-point algorithm to calculate the drift and gain of the sensor network. The most probable drift is when the MAP estimate is the largest, which means we are looking for the system

maximum with respect to drift. In order to find the system maximum, we have to take the derivative of the MAP estimate with respect to the drift, given as:

$$\frac{d\hat{\mathcal{L}}}{d\mathbf{B}} = \frac{d}{d\mathbf{B}}(\mathcal{L}_1) + \frac{d}{d\mathbf{B}}(\mathcal{L}_2) \quad (46)$$

We start by evaluating the derivative of  $\mathcal{L}_1$ ,

$$\frac{d}{d\mathbf{B}}(\mathcal{L}_1) = \frac{d}{d\mathbf{B}}\left(-\frac{1}{2}\ln|\hat{\Sigma}| - \frac{1}{2}(\mathbf{Y} - \mathbf{B})^T \hat{\Sigma}^{-1}(\mathbf{Y} - \mathbf{B}) - \frac{N}{2}\ln(2\pi) + \ln(p(B)) + \ln(p(\alpha))\right) \quad (47)$$

$$\frac{d}{d\mathbf{B}}(\mathcal{L}_1) = -\frac{d}{d\mathbf{B}}\left(\frac{1}{2}(\mathbf{Y} - \mathbf{B})^T \hat{\Sigma}^{-1}(\mathbf{Y} - \mathbf{B})\right) + \frac{d}{d\mathbf{B}}(\ln(p(B))). \quad (48)$$

Before simplifying equation 48, we use the distributive property of matrix multiplication within the derivative to get a form that is easier to simplify:

$$\frac{1}{2}(\mathbf{Y} - \mathbf{B})^T \hat{\Sigma}^{-1}(\mathbf{Y} - \mathbf{B}) = \frac{1}{2}(\mathbf{Y}^T - \mathbf{B}^T)(\hat{\Sigma}^{-1}\mathbf{Y} - \hat{\Sigma}^{-1}\mathbf{B}) \quad (49)$$

$$= \frac{1}{2}(\mathbf{Y}^T \hat{\Sigma}^{-1}\mathbf{Y} - \mathbf{B}^T \hat{\Sigma}^{-1}\mathbf{Y} - \mathbf{Y}^T \hat{\Sigma}^{-1}\mathbf{B} + \mathbf{B}^T \hat{\Sigma}^{-1}\mathbf{B}) \quad (50)$$

$$= \frac{1}{2}(\mathbf{Y}^T \hat{\Sigma}^{-1}\mathbf{Y} - 2\mathbf{Y}^T \hat{\Sigma}^{-1}\mathbf{B} + \mathbf{B}^T \hat{\Sigma}^{-1}\mathbf{B}). \quad (51)$$

Next, we calculate the derivative of each part of equation 48. We use the matrix derivative  $\frac{d}{dx}(x^T A x) = x^T(A + A^T)$  on equation 52. In this case, we know that  $\Sigma = \Sigma^T$  because the kernel matrix is always symmetric, which allows for a simpler solution, as in equation 53.

$$\frac{d}{d\mathbf{B}}\left(\frac{1}{2}(\mathbf{Y}^T \hat{\Sigma}^{-1}\mathbf{Y} - 2\mathbf{Y}^T \hat{\Sigma}^{-1}\mathbf{B} + \mathbf{B}^T \hat{\Sigma}^{-1}\mathbf{B})\right) = \frac{1}{2}(-2\mathbf{Y}^T \hat{\Sigma}^{-1} + 2\mathbf{B}^T \hat{\Sigma}^{-1}) \quad (52)$$

$$= \mathbf{B}^T \hat{\Sigma}^{-1} - \mathbf{Y}^T \hat{\Sigma}^{-1}. \quad (53)$$

We also get,

$$\frac{d}{d\mathbf{B}}(\ln(p(\mathbf{B}))) = \frac{d}{d\mathbf{B}}\left(-\frac{1}{2}\ln|(K_b(X, X))| - \frac{1}{2}(\mathbf{B})^T(K_b(X, X))^{-1}(\mathbf{B}) - \frac{N}{2}\ln(2\pi)\right) \quad (54)$$

$$= -\mathbf{B}^T(K_b(X, X))^{-1}. \quad (55)$$

Substituting equations 53 and 55 back into 48 gives the following derivative for  $\mathcal{L}_1$ :

$$\frac{d}{d\mathbf{B}}(\mathcal{L}_1) = \mathbf{Y}^T \hat{\Sigma}^{-1} - \mathbf{B}^T \hat{\Sigma}^{-1} - \mathbf{B}^T(K_b(X, X))^{-1}. \quad (56)$$

Next, we evaluate the derivative of  $\mathcal{L}_2$  with respect to  $d\mathbf{B}$ :

$$\frac{d}{d\mathbf{B}}\mathcal{L}_2 = \frac{d}{d\mathbf{B}}\left(\sum_{k=1}^2\left(\sum_{j=1}^{N'_t}\frac{1}{2}(-\ln(v(x'_k, t'_j)) - (y(x'_k, t'_j) - \mu(x'_k, t'_j))^2/v(x'_k, t'_j) - \ln(2\pi)))\right)\right). \quad (57)$$

Because  $\mu(x'_k, t'_j)$  is the only term that contains  $\mathbf{B}$ , it is the only non-zero term after taking the derivative, as shown in the following equations:

$$\frac{d}{d\mathbf{B}}\mathcal{L}_2 = \frac{d}{d\mathbf{B}}\left(\sum_{k=1}^2\left(\sum_{j=1}^{N'_t}\frac{1}{2}(-(y(x'_k, t'_j) - \mu(x'_k, t'_j))^2/v(x'_k, t'_j))\right)\right) \quad (58)$$

$$= -\frac{1}{2}\sum_{k=1}^2\left(\sum_{j=1}^{N'_t}\frac{d}{d\mathbf{B}}\left(\frac{(y(x'_k, t'_j) - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}(\mathbf{Y} - \mathbf{B}))^2}{\mathbf{k}_{**} - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\alpha\mathbf{k}_*}\right)\right) \quad (59)$$

$$= -\frac{1}{2}\sum_{k=1}^2\sum_{j=1}^{N'_t}\frac{2(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1})(y(x'_k, t'_j) - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{Y} + \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{B})}{\mathbf{k}_{**} - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\alpha\mathbf{k}_*}. \quad (60)$$

Now we substitute  $\frac{d}{d\mathbf{B}}(\mathcal{L}_1)$  and  $\frac{d}{d\mathbf{B}}(\mathcal{L}_2)$  back into the starting equation, and set  $\frac{d}{d\mathbf{B}}(\hat{\mathcal{L}}) = 0$ .

$$\begin{aligned} 0 &= \frac{d}{d\mathbf{B}}(\hat{\mathcal{L}}) = \mathbf{Y}^T\hat{\Sigma}^{-1} - \mathbf{B}^T\hat{\Sigma}^{-1} - \mathbf{B}^T(K_b(X, X))^{-1} \\ &\quad - \frac{1}{2}\sum_{k=1}^2\sum_{j=1}^{N'_t}\frac{2(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1})(y(x'_k, t'_j) - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{Y} + \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{B})}{\mathbf{k}_{**} - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\alpha\mathbf{k}_*} \end{aligned} \quad (61)$$

$$\begin{aligned} 0 &= \mathbf{Y}^T\hat{\Sigma}^{-1} - \mathbf{B}^T\hat{\Sigma}^{-1} - \mathbf{B}^T(K_b(X, X))^{-1} \\ &\quad - \sum_{k=1}^2\sum_{j=1}^{N'_t}\frac{(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1})(y(x'_k, t'_j) - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{Y}) + (\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1})(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{B})}{\mathbf{k}_{**} - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\alpha\mathbf{k}_*} \end{aligned} \quad (62)$$

$$\begin{aligned} &\mathbf{B}^T\hat{\Sigma}^{-1} + \sum_{k=1}^2\sum_{j=1}^{N'_t}\frac{(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1})(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{B})}{\mathbf{k}_{**} - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\alpha\mathbf{k}_*} + \mathbf{B}^T(K_b(X, X))^{-1} = \mathbf{Y}^T\hat{\Sigma}^{-1} \\ &+ \sum_{k=1}^2\sum_{j=1}^{N'_t}\frac{(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1})(\alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\mathbf{Y} - y(x'_k, t'_j))}{\mathbf{k}_{**} - \alpha\mathbf{k}_*^T\hat{\Sigma}^{-1}\alpha\mathbf{k}_*}. \end{aligned} \quad (63)$$

At this point, we manipulate this equation into the form  $\mathbf{A}\mathbf{B} = \mathbf{C}$  so that we can invert  $\mathbf{A}$  and solve for  $\mathbf{B}$ .

Inverting  $\mathbf{A}$  is the rate limiting step in the proposed methods. Because  $\mathbf{A}$  is an  $N$  by  $N$  matrix, the run time

of the proposed methods is  $\mathcal{O}(N^3)$ . Thus, we find a fixed-point algorithm for drift when gain ( $\alpha$ ) is known,

$$A = (\hat{\Sigma}^{-1})^T + \sum_{k=1}^2 \sum_{j=1}^{N'_t} \frac{(\hat{\Sigma}^{-1} \mathbf{k}_* \alpha)(\alpha \mathbf{k}_*^T \hat{\Sigma}^{-1})}{\mathbf{k}_{**} - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \alpha \mathbf{k}_*} + (K_b(X, X)^{-1})^T \quad (64)$$

$$C = \mathbf{Y}^T \hat{\Sigma}^{-1} + \sum_{k=1}^2 \sum_{j=1}^{N'_t} \frac{(\alpha \mathbf{k}_*^T \hat{\Sigma}^{-1})(\alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \mathbf{Y} - y(x'_k, t'_j))}{\mathbf{k}_{**} - \alpha \mathbf{k}_*^T \hat{\Sigma}^{-1} \alpha \mathbf{k}_*} \quad (65)$$

$$B = (A)^{-1} C. \quad (66)$$

## 6.4 Fixed-Point for Gain

We also want to find a fixed-point algorithm to calculate gain, however, we can not move  $\alpha$  completely outside of the inverse function, as shown in equation 67, so no fixed-point algorithm exists to calculate the gain.

$$(\alpha^2 K(X, X) + \sigma^2 \mathbf{I})^{-1}. \quad (67)$$

Instead, we add some noise lag so that we can pull  $\alpha$  outside of the inverse of  $\hat{\Sigma}$ :

$$\sigma_{lag}^2 = \frac{\sigma^2}{\alpha^2} \quad (68)$$

$$\hat{\Sigma} = \alpha^2 K(X, X) + \alpha^2 \sigma_{lag}^2 \mathbf{I} \quad (69)$$

$$\hat{\Sigma} = \alpha^2 (K(X, X) + \sigma_{lag}^2 \mathbf{I}) \quad (70)$$

$$\Sigma = K(X, X) + \sigma_{lag}^2 \mathbf{I} \quad (71)$$

Although altering the noise of the Gaussian process in anyway is highly irregular, altering the noise allows for a closed form solution to approximate the gain. By using a fixed-point algorithm for gain, the optimization strategy can avoid the run time increase caused by having to use an optimizer. Using equation 70 instead of our initial definition of  $\hat{\Sigma}$ , we take the derivative of  $\hat{\mathcal{L}}$  with respect to  $\alpha$  and generate a fixed-point algorithm for  $\alpha$ , given as:

$$\frac{d\hat{\mathcal{L}}}{d\alpha} = \frac{d}{d\alpha}(\mathcal{L}_1) + \frac{d}{d\alpha}(\mathcal{L}_2). \quad (72)$$



Starting with,  $\mathcal{L}_1$ :

$$\frac{d}{d\alpha}(\mathcal{L}_1) = \frac{d}{d\alpha} \left( -\frac{1}{2} \ln |\alpha^2 \Sigma| - \frac{1}{2} (\mathbf{Y} - \mathbf{B})^T (\alpha^2 \Sigma)^{-1} (\mathbf{Y} - \mathbf{B}) - \frac{N}{2} \ln(2\pi) + \ln(p(b)) + \ln(p(\alpha)) \right) \quad (73)$$

$$= \frac{d}{d\alpha} \left( -\frac{1}{2} \ln |\alpha^2 \Sigma| - \frac{1}{2\alpha^2} (\mathbf{Y} - \mathbf{B})^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) \right) + \frac{d}{d\alpha} (\ln(p(\alpha))). \quad (74)$$

Starting from left to right, we take the derivative of each part one step at a time as follows:

$$\frac{d}{d\alpha} \left( -\frac{1}{2} \ln |\alpha^2 \Sigma| \right) = \frac{d}{d\alpha} \left( -\frac{1}{2} \ln(\alpha^{2N} |\Sigma|) \right) = \frac{d}{d\alpha} (-N \ln(\alpha) - \frac{1}{2} \ln |\Sigma|) = -\frac{N}{\alpha}. \quad (75)$$

In equation 75, we move  $\alpha$  outside the determinate through the property  $|c\mathbf{A}| = c^N |\mathbf{A}|$ , where  $N$  equals the number of elements in matrix  $\mathbf{A}$  [9].

$$\frac{d}{d\alpha} \left( -\frac{1}{2\alpha^2} (\mathbf{Y} - \mathbf{B})^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) \right) = \frac{1}{\alpha^3} (\mathbf{Y} - \mathbf{B})^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) \quad (76)$$

Next, we simplify the  $\ln(p(\alpha))$  with  $\alpha$  being modeled by a scaled inverse chi-squared distribution,

$$p(\alpha) = \frac{(\tau^2 \frac{v}{2})^{v/2}}{\Gamma(v/2)} \frac{\exp(-\frac{v\tau^2}{2\alpha})}{\alpha^{1+v/2}} \quad (77)$$

$$\ln(p(\alpha)) = \ln \left( \frac{(\tau^2 \frac{v}{2})^{v/2}}{\Gamma(v/2)} \frac{\exp(-\frac{v\tau^2}{2\alpha})}{\alpha^{1+v/2}} \right) \quad (78)$$

$$= \ln((\tau^2 \frac{v}{2})^{v/2}) - \ln(\Gamma(v/2)) + \ln(\exp(-\frac{v\tau^2}{2\alpha})) - \ln(\alpha^{1+v/2}) \quad (79)$$

$$= \ln((\tau^2 \frac{v}{2})^{v/2}) - \ln(\Gamma(v/2)) + \frac{-v\tau^2}{2\alpha} - (1 + v/2) \ln(\alpha). \quad (80)$$

and then take the derivative of the equation with respect to  $\alpha$ .

$$\frac{d(\ln(p(\alpha)))}{d\alpha} = \frac{d}{d\alpha} \left( \ln((\tau^2 \frac{v}{2})^{v/2}) - \ln(\Gamma(v/2)) + \frac{-v\tau^2}{2\alpha} - (1 + v/2) \ln(\alpha) \right) \quad (81)$$

$$= \frac{v\tau^2}{2\alpha^2} - \frac{(1 + v/2)}{\alpha}. \quad (82)$$

We substitute the calculated derivatives into  $\frac{d}{d\alpha}(\mathcal{L}_1)$ , which gives us,

$$\frac{d}{d\alpha}(\mathcal{L}_1) = -\frac{N}{\alpha} + \frac{1}{\alpha^3} (\mathbf{Y} - \mathbf{B})^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) + \frac{v\tau^2}{2\alpha^2} - \frac{(1 + v/2)}{\alpha}. \quad (83)$$

Next, we calculate  $\frac{d}{d\alpha}(\mathcal{L}_2)$  to obtain,

$$\frac{d}{d\alpha}(\mathcal{L}_2) = \frac{d}{d\alpha} \left( \sum_{k=1}^2 \left( \sum_{j=1}^{N'_t} \frac{1}{2} (-\ln(v(x'_k, t'_j)) - (y(x'_k, t'_j) - \mu(x'_k, t'_j))^2 / v(x'_k, t'_j) - \ln(2\pi))) \right) \right) \quad (84)$$

$$= \sum_{k=1}^2 \left( \sum_{j=1}^{N'_t} \frac{1}{2} \left( -\frac{d}{d\alpha} \left( \frac{(y(x'_k, t'_j) - \alpha \mathbf{k}_*^T (\alpha^2 \Sigma)^{-1} (\mathbf{Y} - \mathbf{B}))^2}{\mathbf{k}_{**} - \alpha \mathbf{k}_*^T (\alpha^2 \Sigma)^{-1} \alpha \mathbf{k}_*} \right) \right) \right) \quad (85)$$

$$= \sum_{k=1}^2 \left( \sum_{j=1}^{N'_t} \frac{1}{2} \left( -\frac{d}{d\alpha} \left( \frac{(y(x'_k, t'_j) - \frac{1}{\alpha} \mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}))^2}{\mathbf{k}_{**} - \mathbf{k}_*^T \Sigma^{-1} \mathbf{k}_*} \right) \right) \right) \quad (86)$$

$$= \sum_{k=1}^2 \sum_{j=1}^{N'_t} \left( -\frac{(\frac{1}{\alpha^2} \mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}))(y(x'_k, t'_j) - \frac{1}{\alpha} \mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}))}{\mathbf{k}_{**} - \mathbf{k}_*^T \Sigma^{-1} \mathbf{k}_*} \right) \quad (87)$$

$$= \sum_{k=1}^2 \sum_{j=1}^{N'_t} \frac{-\frac{1}{\alpha^2} \mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) y(x'_k, t'_j) + \frac{1}{\alpha^3} (\mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}))^2}{\mathbf{k}_{**} - \mathbf{k}_*^T \Sigma^{-1} \mathbf{k}_*}. \quad (88)$$

Now we combine the derivative and set the equation equal to zero, as in the following equations:

$$0 = \frac{d}{d\alpha}(\hat{\mathcal{L}}) = -\frac{N}{\alpha} + \frac{1}{\alpha^3} (\mathbf{Y} - \mathbf{B})^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) + \frac{v\tau^2}{2\alpha^2} - \frac{(1+v/2)}{\alpha} + \sum_{k=1}^2 \sum_{j=1}^{N'_t} \frac{\frac{1}{\alpha^3} (\mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}))^2 - \frac{1}{\alpha^2} \mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) y(x'_k, t'_j)}{\mathbf{k}_{**} - \mathbf{k}_*^T \Sigma^{-1} \mathbf{k}_*} \quad (89)$$

$$0 = -N\alpha^2 + (\mathbf{Y} - \mathbf{B})^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) + \alpha \frac{v\tau^2}{2} - \alpha^2 (1+v/2) + \sum_{k=1}^2 \sum_{j=1}^{N'_t} \frac{(\mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}))^2 - \alpha \mathbf{k}_*^T \Sigma^{-1} (\mathbf{Y} - \mathbf{B}) y(x'_k, t'_j)}{\mathbf{k}_{**} - \mathbf{k}_*^T \Sigma^{-1} \mathbf{k}_*}. \quad (90)$$

Equation 90 is a second-degree polynomial that can be solved by the quadratic equation. Although equation 90 gives a fixed-point algorithm for gain, it exists only due to the initial approximation of the noise. Because of the approximation, we expect some numerical error in any gain we find by using this fixed-point algorithm. The numerical error is most likely caused by allowing the gain to scale the noise. One of our initial assumptions is that the gain only affects the underlying signal, but this assumption is not necessarily correct. In the real world, the gain could affect the underlying signal and the noise, which would cause the fixed-point algorithm for gain, without the noise lag term, to find a more accurate estimate for the gain.

## 6.5 Proposed Optimization Strategies

At this point, we have two fixed-point algorithms that we can alternate between to calculate the  $\alpha$  and  $\mathbf{B}$  for our model. However, the use of the noise lag could cause problems, including numerical errors. The chance of numerical error in the alternating fixed-point optimization strategy causes us to build a second

optimization strategy using an Adam optimizer from PyTorch to optimize the MAP estimate for  $\alpha$ , while using the fixed-point algorithm for drift [10]. We still use the fixed-point algorithm for the drift because it is calculated directly from the MAP estimate and, as such, the fixed-point algorithm always finds the drift that maximized the MAP estimate for a given gain( $\alpha$ ). Lastly, we build a third optimization strategy that optimizes the bandwidths as well as the gain with respect to the MAP estimate, and it uses the fixed-point algorithm for drift. The third optimization strategy is proposed because we want to understand the effect that optimization strategy 2, optimizing gain, has on the difficulty of estimating the bandwidths,  $\theta$ . Thus, we propose the following three optimization strategies:

- Optimization strategy 1 uses the fixed-point algorithm for gain ( $\alpha$ ) and the fixed-point algorithm for drift, alternating back and forth between the two fixed-point algorithms until the values for the gain and the drift converge.
- Optimization strategy 2 uses an optimizer, Adam from Pytorch, to control the gain ( $\alpha$ ). When the optimizer alters the gain, use the drift's fixed-point algorithm to calculate the new drift.
- Optimization strategy 3 uses an optimizer, Adam from Pytorch, to control the gain ( $\alpha$ ) and the kernel's bandwidths. When the optimizer alters the gain or bandwidth, use the drift's fixed-point algorithm to calculate the new drift.

## 7 Results

### 7.1 Overview of the Synthetic Data

To test the proposed optimization strategies, we build synthetic data that matches our initial assumptions about the problem. First, we use a uniform random distribution to place the sensors in the sensor network and the ground truth sensors. Next, we build a multi-variant Gaussian Distribution, using PyTorch, with a given space and time bandwidth, and we sample the distribution at every sensor location at each time reading [10]. Note that the synthetic data is built with only a two-dimensional space, rather than the three-dimensional space of the real world. We also sample a fine-grain grid from a multi-variant Gaussian distribution to validate the accuracy of the models. We construct a second multi-variant Gaussian distribution to sample the drift for each sensor. Lastly, the gain for the sensor network is sampled from a scaled inverse chi-squared distribution. The data, gain, and drift are then combined using the assumed data equation 91.

$$y(x_i, t_j) = \alpha f(x_i, t_j) + b(x_i, t_j) + \epsilon \quad (91)$$

We then test a variety of variables defining synthetic data to understand their impact on the proposed optimization strategies. As a baseline for the synthetic data, we are setting the variables to average values, Table 1, so that none of the variables are initially responsible for the success or failure of the optimization strategies. The variables for the distribution of  $\alpha$  are chosen arbitrarily to give a distribution between about 0 and 2. Using these variables, we ran 30 trials with our three optimization strategies and a naive Gaussian process.

Table 1: Baseline variables

Network Sensors	50	Ground Truth Sensors	2
Space Bandwidth $\theta_s$	2.0	Time Bandwidth $\theta_t$	1.0
Noise Standard Deviation	0.01	Drift Standard Deviation	1.0
Drift Mean	0.0	Drift Variance	1.0
$v$	50	$\tau^2$	1.0

Table 2: Baseline error

Model	L1 Gain( $\alpha$ ) Error	L2 Drift Error	L2 GT Error	L2 Model Error	MAP Estimate
Naive GP	0.1646	0.9824	10.46	11.80	-208800
Fixed-point	0.1208	0.02056	0.0108	0.07037	3491
Optimizing $\alpha$	0.07554	0.01093	0.008667	0.05149	3496
Optimizing $\alpha$ and $\theta$	0.07949	0.01124	0.009371	0.05526	3495

From Table 2 we can see that we are obtaining an L2 model error, in the best case, of 0.05149. This L2 model error is found by taking 15625 points in space and time, having each model estimate the data value at

those points, and then averaging the squared error from the actual data at those points. Table 2 also shows that the  $\mathcal{L}_2$  is working well, as the L2 ground truth error is low in all three optimization strategies. This L2 ground truth error is just the L2 error at every reading from the ground truth sensors. In comparison, the naive Gaussian process has an extremely high L2 ground truth error because it does not use any of the observed data from the ground truth sensors. The only category in which the naive Gaussian process is comparable to our models is in the L1 gain error because the naive Gaussian process is assuming no gain affecting the system, and thus uses a gain value of one. Using the mean means that the naive Gaussian process's gain error is comparable to the standard deviation of our  $\text{gain}(\alpha)$  distribution. However, knowing that the standard deviation is comparable to the gain error of our models is problematic. This comparability has two potential causes: there is an error in the math, or estimating the gain is difficult. For the alternating fixed-point optimization strategy, we know that there is an error in the math because an approximation was made. However, we also found that accurately estimating the gain with our optimization strategies requires more data than accurately estimating the drift.

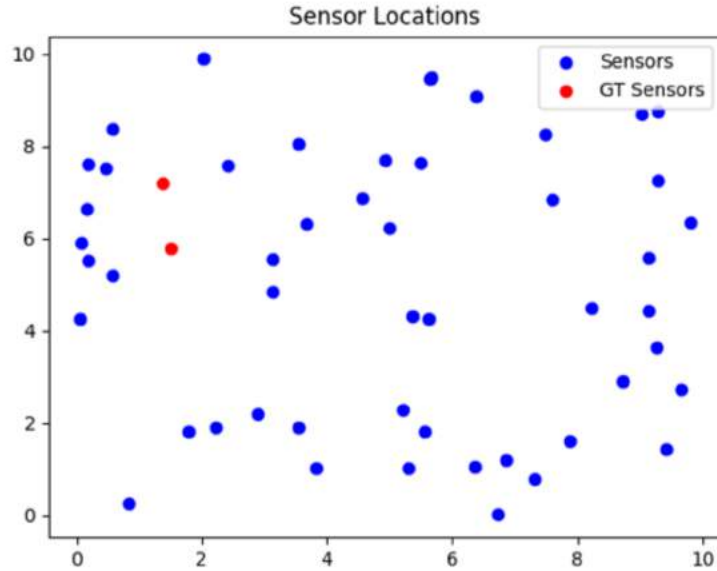


Figure 1: Sensor locations from one trial

We start by examining why the naive Gaussian process is performing so poorly in Table 2. To understand why the naive Gaussian process is struggling, we first look at the sensor distribution. Figure 1 shows a variety of distances between the sensors. These different distances lead to some confounding effects on the Gaussian process. First, if there is only one sensor to cover a large area of space, the Gaussian process has an error in that area proportional to the sensor's drift. Second, if two sensors are close together, each with its unique

drift, the Gaussian process tries to appease both points within the noise. Between these two situations, the L2 model error of the Gaussian process can skyrocket. However, looking back at equation 91, we can see that the drift term ( $\mathbf{B}$ ) can be combined with our noise term ( $\epsilon$ ) to build a more accurate naive Gaussian process. This alteration is done by simply adding the standard deviation of the drift to the standard deviation of the noise. In fact, after testing this theory we can see in Table 3 that this change has drastically improved the L2 model error and the MAP estimate of the naive Gaussian process. Although the high noise variant of the naive Gaussian process is able to perform much better, it still has a significantly higher L2 model error than the proposed optimization strategies.

Table 3: Naive Gaussian process with high noise

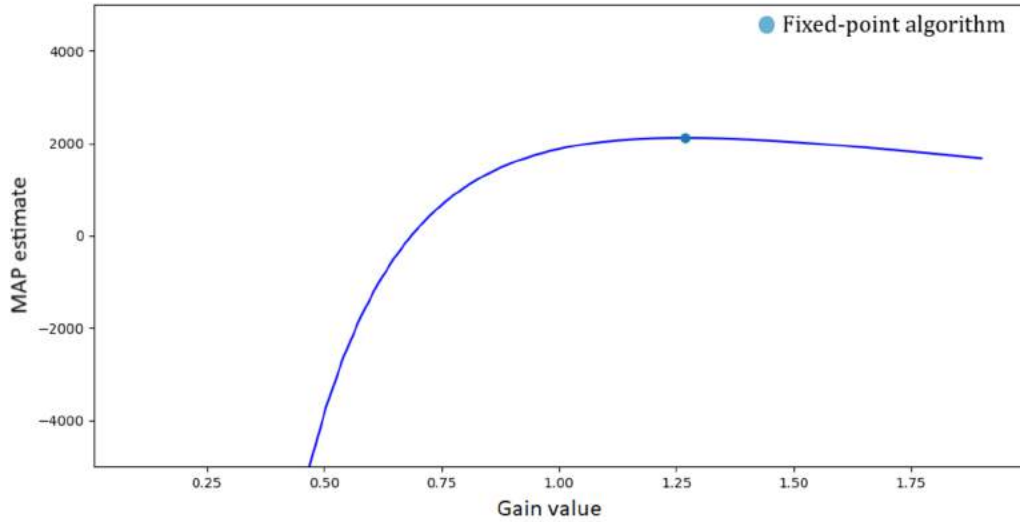
Model	L1 Gain( $\alpha$ ) Error	L2 Drift Error	L2 GT Error	L2 Model Error	MAP Estimate
High Noise GP	0.1823	0.9711	0.2929	0.3418	2363

As shown in Table 2, the fixed-point model performs worse, overall, than the optimize gain( $\alpha$ ) model, most likely due to the inaccuracy caused by using a lagging noise variable. To verify the cause of the numerical error, we ran some ablation studies focusing on the gain and drift. Figure 2 shows that gain’s fixed-point algorithm is finding the maximum MAP estimate when the drift is given, we can see from the last column of Table 2 that this is not the case when the alternating fixed-point optimization strategy is used. To confirm that the noise lag is causing the problem, the drift’s fixed-point algorithm is also checked for its quality. Using an Adam optimizer, we check if the drift’s fixed-point algorithm is finding the best drift for the MAP estimate. We find that the optimizer is unable to find a better drift than the drift’s fixed-point algorithm when optimizing the MAP estimate. Upon validating the drift’s fixed-point algorithm, the noise lag is the only place where an error can be occurring. Furthermore, from Table 2 we can see that even though the alternating fixed-point optimization strategy has a much higher error in gain( $\alpha$ ) estimations, its L2 drift error is similar to the other proposed optimization strategies.

To better understand how the optimization strategies function, we ran several tests with varying variable values. These tests vary the number of ground truth sensors, the number of network sensors, the noise standard deviation, and the drift bandwidth. By varying the number of network sensors and the number of ground truth sensors, we hope to understand the effect they have upon the optimization strategies. In addition, the tests enable us to quantify the value of the addition of a network sensor and a ground truth sensor. The next test is to vary the standard deviation of the noise. The goal of this test is to find the breaking point of the proposed optimization strategies, which is when the proposed optimization strategies became comparable to the naive Gaussian process, at which point there is no longer any benefit in using the proposed optimization strategies. For the last test, we decrease the drift bandwidth to find how well the



Figure 2: Gain's effect on the MAP estimate



optimization strategies deal with drift that changes at the same rate as the underlying Gaussian distribution.

## 7.2 Ground Truth Sensors Versus Network Sensors

The first test with the synthetic data is comparing the number of ground truth sensors versus the number of network sensors. For this test, the system holds the baseline variables in Table 1 constant while changing only the number of ground truth sensors or the number of network sensors. As stated earlier, the baseline variables have been selected to help us understand how well the optimization strategies are performing and, therefore, are set to provide both minimal support and harm to the optimization strategies. Understanding the baseline variables, we select a range from zero to four ground truth sensors to see the rate of improvement. We use a range of 25 to 100 network sensors in 25 sensor increments.

Looking at Figure 3 (a), we can see the results of testing the number of ground truth sensors versus the number of network sensors with respect to the L2 model error. First, examining the optimization strategies with respect to the number of ground truth sensors, we find a negligible difference in the L2 model error once the number of ground truth sensors exceeds two. Meanwhile, the difference between one and two ground truth sensors is much more significant. Looking exclusively at the optimizing gain optimization strategy, we obtain a decrease of 0.03 L2 error when increasing from one to two ground truth sensors and less than a 0.01 L2 error decrease when increasing the number of ground truth sensors past two. This insignificant change is most likely due to two ground truth sensors being enough to effectively cover the space. Furthermore, the variance of the L2 model error did not change significantly between two and four ground truth sensors, which

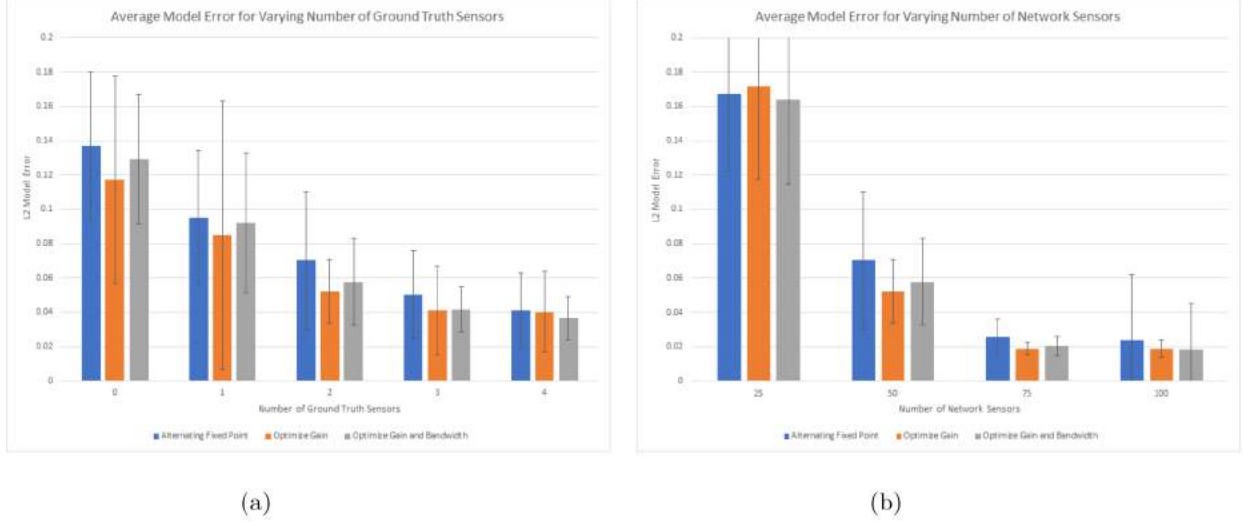


Figure 3: 30 trials measuring L2 model error with varying number of ground truth sensors or a varying number of network sensors

supports the idea that 2 ground truth sensors are enough to cover the space. Lastly, when the system has zero ground truth sensors, the proposed models are still able to reduce the L2 model error significantly in comparison to the naive Gaussian process with high noise, Table 3. This result is expected as past research into blind calibration has been done using the Gaussian process.

Next, looking exclusively at the effect of the number of network sensors on the L2 model error, Figure 3 (b), the number of network sensors has a significant effect on the L2 model error. All three optimization strategies are able to get below an L2 model error of 0.03, 3% error, with 100 network sensors, and the optimizing gain optimization strategy is able to achieve an L2 model error of 0.01, 1% error. On the other hand, when there are only 25 network sensors, the L2 model error is 0.16. The dichotomy of the L2 model error between 25 network sensors and 100 network sensors supports the idea that the number of networks sensors has a direct effect on the L2 model error. In comparison to the number of ground truth sensors, we see that increasing the number of network sensors has a much greater impact on reducing the L2 ground truth error, most likely because of the importance of the number of network sensors in the L2 model error. As we increase the number of network sensors, more of the space is covered by sensors and, in turn, is more accurately modeled by the Gaussian process. In comparison, as we increase the number of ground truth sensors, we are not directly increasing how accurately the space is modeled, but instead increasing the calibration accuracy.

Looking now at Figure 4, the number of ground truth sensors has a much more consistent impact on reducing the error of the gain estimations throughout all optimization strategies than the number of network sensors. However, this finding should be taken with a grain of salt, because the optimization strategy



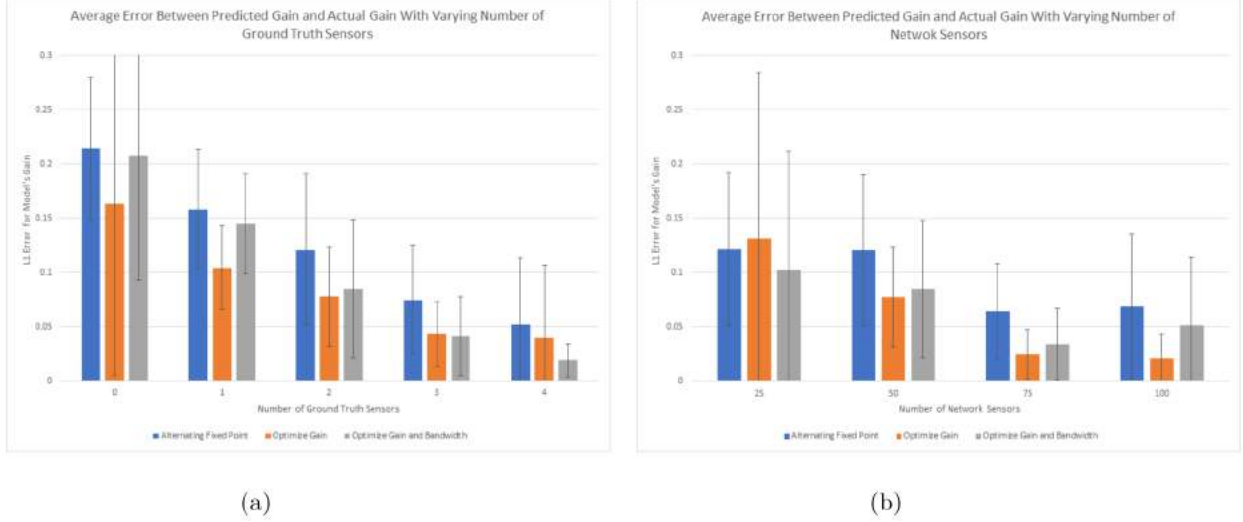


Figure 4: 30 trials measuring gain error with varying number of ground truth sensors or a varying number of network sensors

causing the large gain error with respect to the number of ground truth sensors is the alternating fixed-point optimization strategy. If this optimization strategy's data are ignored in the gain error comparison, the effect that the number of ground truth sensors and the number of network sensors has on gain error is similar. In the end, the number of network sensors has a greater impact on the performance of the optimization strategies than the number of ground truth sensors. This comparison might not seem fair because increasing the number of ground truth sensors by a higher amount could lead to results that are comparable to increasing the number of network sensors by 25, but in the real world, the cost of a ground truth sensor is much more comparable to increasing the number of network sensors by at least 25. For example, for the sensor network in the Salt Lake Valley, a single network sensor costs about 150 dollars whereas a ground truth sensor costs thousands. Therefore, we are more interested in increasing the number of network sensors before increasing the number of ground truth sensors.

### 7.3 Noise Levels

For the noise level test, we increase the noise until the breaking point is found. Looking at Figure 5, the point at which the optimization strategies becomes comparable to the naive Gaussian process is when the standard deviation of the noise is  $\sigma = 1$ . The optimization strategies are able to handle relatively high levels of noise when compared to the standard deviation of the signal. When the noise standard deviation is 0.5, 50% that of the signal standard deviation, the optimization strategies has an L2 model error of about 0.2, 20% error, whereas the naive Gaussian process is still getting an L2 model error of 0.5. Unfortunately, once

the standard deviation of the noise equals the signal strength, the optimization strategies become unable to outperform the naive Gaussian process, because of how the noise standard deviation affects the Gaussian process. Noise standard deviation affects how much the Gaussian process is influenced by the observed data. Once the noise standard deviation gets large enough such that the Gaussian process cannot tell the difference between the mean and the observed data, it will estimate the mean. Because both the proposed optimization strategies and the naive Gaussian process handle the noise similarly, they both start having similar levels of L2 model error.

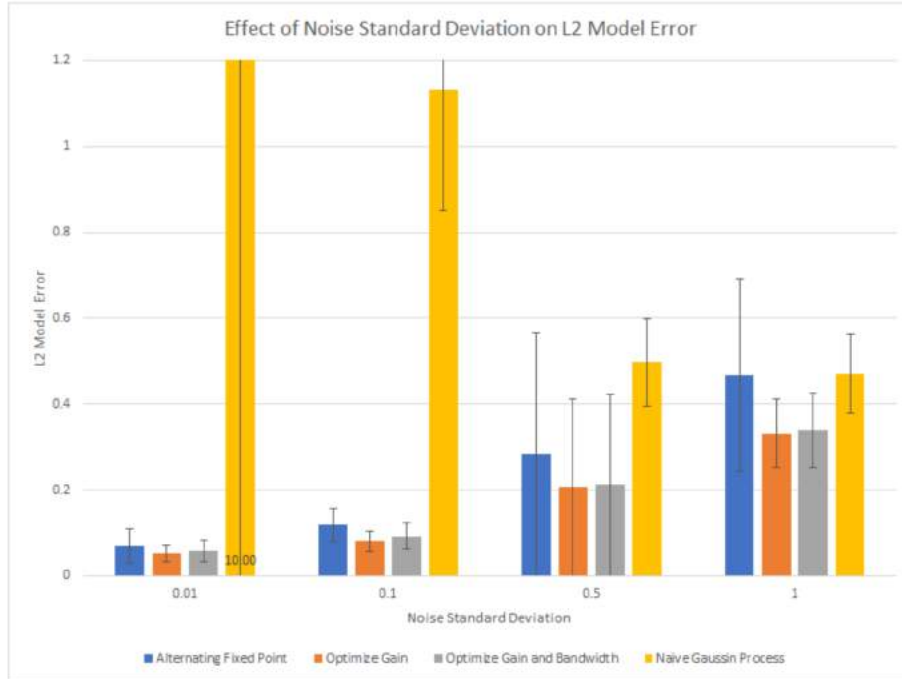


Figure 5: 30 trials measuring L2 model error with varying noise levels

The noise level affects the accuracy of the optimization strategies' drift estimations. As the noise standard deviation gets higher, the L2 error for drift also increases, as seen in Figure 6, possibly due to two effects. First, as the noise level increase, each signal becomes more sporadic and, in turn, less correlated. By decreasing the correlation in the system, the optimization strategies has a harder time telling the difference between the underlying Gaussian distribution and the drift. Second, the optimization strategies deprioritize obtaining high accuracy. The noise standard deviation is already known by the optimization strategies and is already being added to the kernel matrix. By adding a high noise to the kernel matrix, the Gaussian process deprioritizes going directly through the observed data points, even after being altered by drift and gain. Because the Gaussian process goes through a larger range centered at the observed data points, the optimization strategies, in turn, deprioritize the accuracy of the drift and gain estimations. Instead, the proposed optimization strategies prioritizes moving the range around the altered observed data points such

that the Gaussian process passes through the ground truth data points, which causes the decrease in accuracy of the drift and gain calibrations.

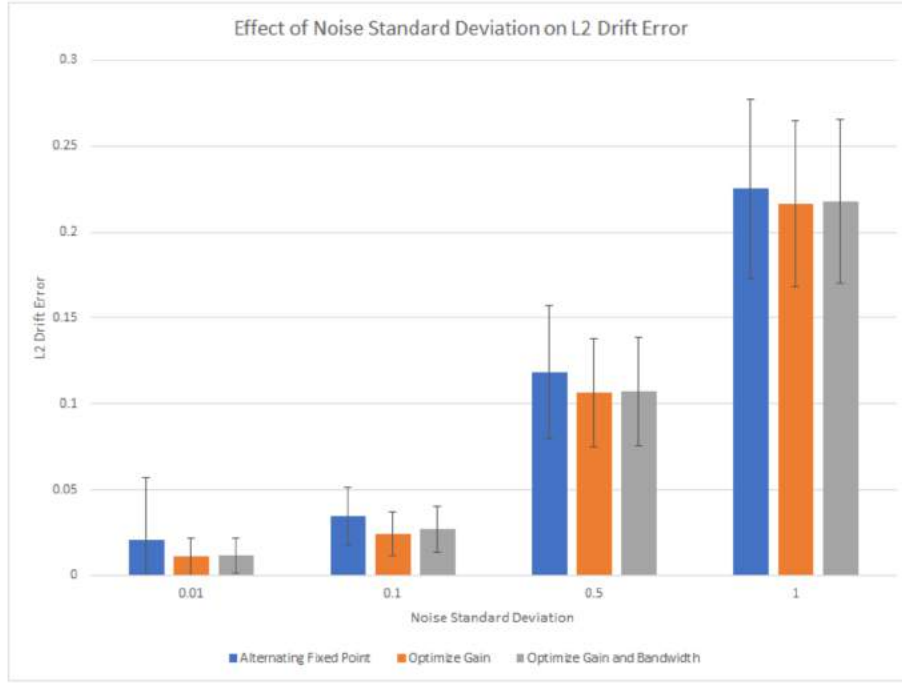


Figure 6: 30 trials measuring L2 drift error with varying noise levels

## 7.4 Drift Bandwidth

For the last test on the synthetic data, we test the effect of the drift bandwidth on the performance of the optimization strategies. Looking at Figure 7, as the drift bandwidth approaches the time bandwidth, the L2 error of the optimization strategies gets higher. In the test, we vary the drift bandwidth from 2 to 8, whereas our time bandwidth is set to 1. Thus, this test shows how dependent all of the optimization strategies are on the assumption that the sensors are changing much slower than the signal in time. Although the L2 model error helps us understand how well a optimization strategy is performing, it has difficulty providing any reason for the optimization strategies failing. To more accurately understand why the optimization strategies are failing, we look at the effect that the drift bandwidth has on the optimization strategies' gain and drift estimations.

Looking at Figure 9 (b), the drift bandwidth has little effect on the L2 drift error. Although there is an increase in the L2 drift error of 0.06 for the optimizing gain optimization strategy from a drift bandwidth of 8 to a bandwidth of 2, the change in error does not account for the increase of the 0.2 L2 model error that occurs. Because we do not understand what is causing the increase in L2 model error, we also check the effect

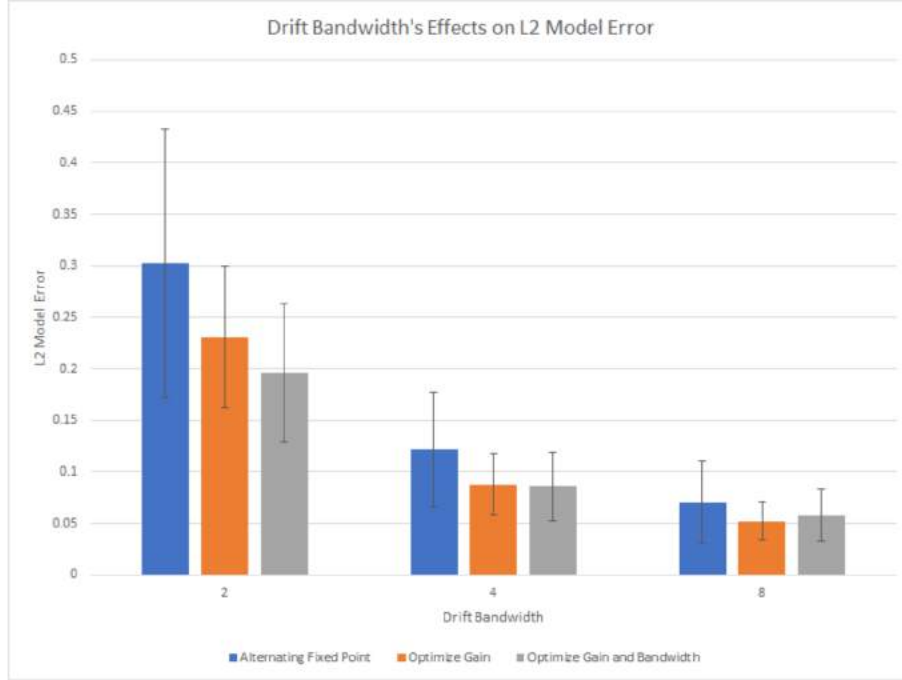


Figure 7: 30 trials measuring L2 model error with varying drift bandwidths

that drift bandwidth has on gain error. Looking at Figure 9 (a), at a drift bandwidth of 2, the optimization strategies are unable to accurately estimate gain at all. Furthermore, these optimization strategies obtain a drastically higher gain error than the naive Gaussian process, which always guesses the mean value of the gain. The cause of this correlation is unclear, but may be the result of the difference in estimation difficulty between the gain and drift. In other words, the required amount of information to calculate the drift is much lower than the required amount of information to calculate gain. I hypothesize that we can even visualize this difference by graphing the relationship between a one-dimensional drift and a one-dimensional gain. For this graph, we use the equation  $z = x^{-2}y^2$ , which is a simplified version of  $\mathcal{L}_1$ .

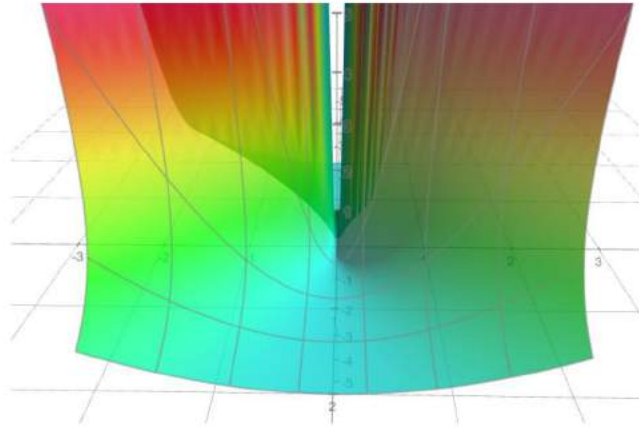


Figure 8: Graphing a simplified MAP estimate



We can see that the gain, the red x axis in Figure 8, has an extremely flat slope. To find a good estimate for the gain, the proposed optimization strategies need strong enough data to support the gain value. However, as the drift bandwidth gets closer to the time bandwidth, the strength of the data gets weaker. As Figure 8 shows, weakening the data has relatively little effect on the difficulty of finding the drift because of the steep slope, the green y axis, but this change has a large impact on the optimization strategies' ability to estimate the true gain. One potential way to overcome weak data would be to set the gain equal to the mean once the drift bandwidth approaches the time bandwidth. Although this solution will not fix the problem, it could potentially reduce the L2 model error for the proposed optimization strategies when the drift bandwidth approaches the time bandwidth.

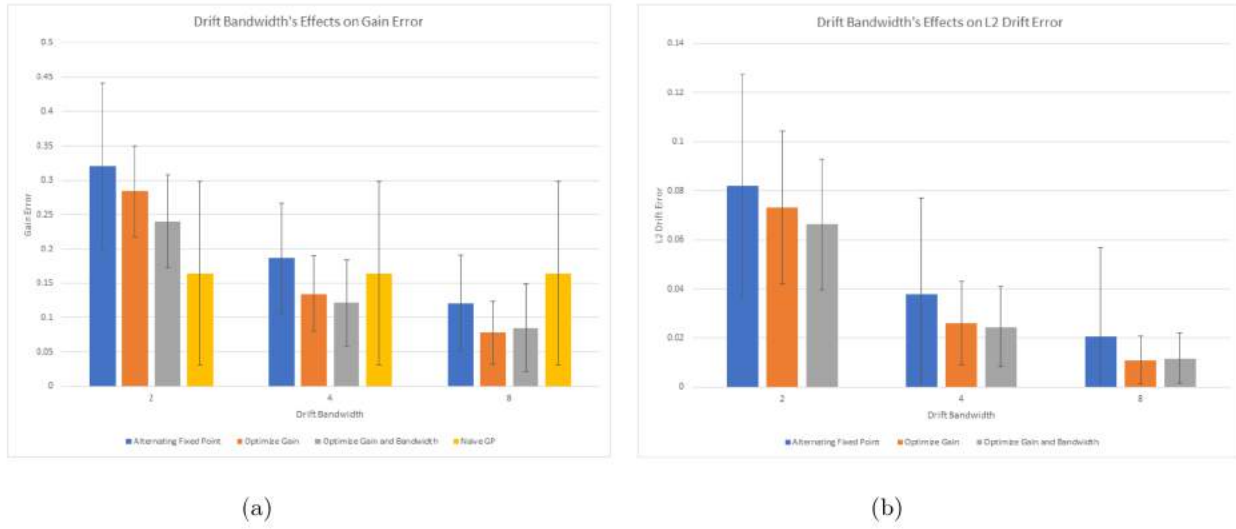


Figure 9: 30 trials measuring gain error and L2 drift error with varying drift bandwidths

## 8 Conclusion

The proposed optimization strategies can make accurate calibration estimations under decent conditions: enough sensors to cover the space, enough ground truth sensors to manage the space, the drift bandwidth significantly larger than the time bandwidth, noise smaller than the signal. When the conditions are poor, the proposed optimization strategies are unable to consistently make accurate gain estimations. However, using the proposed optimization strategies to build the Gaussian process model has a lower L2 model error than a completely naive Gaussian process. Furthermore, the poor conditions have a manageable effect on the L2 drift error, because the drift requires lower levels of data to support accurate estimations. There could be potential in setting the gain value to the mean when conditions worsen because poor conditions cause inaccurate gain estimations.

Optimization strategy 1, the alternating fixed-point optimization strategy, has higher error than optimization strategy 2, optimizing gain with the drift fixed-point, in every situation because of the numerical error caused by the noise lag. This outcome, although predictable, is unfortunate because it means that the best solution requires the use of an optimizer, which increases the run time. On the plus side, we can use optimization strategy 1 to generate a starting point for an optimizer, which reduces the run time because we know that the alternating fixed-point optimization strategy's estimation is a more accurate starting point than the mean.

Finally, optimization strategy 3 shows that estimating bandwidths does not affect the proposed method's accuracy in estimating calibrations as optimization strategy 3's accuracy is comparable to optimization strategy 2. Furthermore, optimization strategy 3 is always able to estimate within 1 percent accuracy of the bandwidths, including when the noise has a standard deviation of 1 and when the drift bandwidth is 2. Thus, optimization strategy 3 shows that we can use semiblind calibration to reduce the problem to a modeling problem by removing the gain and drift from the observed data. Overall, the proposed optimization strategies are successfully able to calibrate synthetic data. The next step will be to apply the proposed optimization strategies to real world data, which will be sampled from the sensor network deployed in the Salt Lake Valley.

## References

- [1] Health Effects Institute, “State of global air 2018,” *Boston, MA:Health Effects Institute*, vol. Special Report, p. 22, 2018. [Online]. Available: <https://www.stateofglobalair.org/sites/default/files/soga-2018-report.pdf>.
- [2] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, ser. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006, 248 pp., OCLC: ocm61285753, ISBN: 978-0-262-18253-9.
- [3] K. E. Kelly, W. W. Xing, T. Sayahi, *et al.*, “Community-based measurements reveal unseen differences during air pollution episodes,” *Environmental Science & Technology*, vol. 55, no. 1, pp. 120–128, 2021, PMID: 33325230. DOI: 10.1021/acs.est.0c02341. eprint: <https://doi.org/10.1021/acs.est.0c02341>. [Online]. Available: <https://doi.org/10.1021/acs.est.0c02341>.
- [4] C. Mullen, S. Grineski, T. Collins, *et al.*, “Patterns of distributive environmental inequity under different pm2.5 air pollution scenarios for salt lake county public schools,” *Environmental Research*, vol. 186, p. 109543, 2020, ISSN: 0013-9351. DOI: <https://doi.org/10.1016/j.envres.2020.109543>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0013935120304369>.
- [5] M. Takruri, S. Challa, and R. Chakravorty, “Recursive bayesian approaches for auto calibration in drift aware wireless sensor networks,” *Journal of Networks*, vol. 5, no. 7, pp. 823–832, Jul. 1, 2010, ISSN: 1796-2056. DOI: 10.4304/jnw.5.7.823-832. [Online]. Available: <http://www.academypublisher.com/ojs/index.php/jnw/article/view/2997> (visited on 07/20/2021).
- [6] D. Kumar, S. Rajasegarar, and M. Palaniswami, “Automatic sensor drift detection and correction using spatial kriging and kalman filtering,” in *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, ISSN: 2325-2944, May 2013, pp. 183–190. DOI: 10.1109/DCOSS.2013.52.
- [7] T. Becnel, T. Sayahi, K. Kelly, and P.-E. Gaillardon, “A recursive approach to partially blind calibration of a pollution sensor network,” in *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, Las Vegas, NV, USA: IEEE, Jun. 2019, pp. 1–8, ISBN: 978-1-72812-437-7. DOI: 10.1109/ICESS.2019.8782523. [Online]. Available: <https://ieeexplore.ieee.org/document/8782523/> (visited on 07/18/2021).
- [8] S. Zhe, W. Xing, and R. M. Kirby, “Scalable high-order gaussian process regression,” in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, K. Chaudhuri and M. Sugiyama, Eds., ser. Proceedings of Machine Learning Research, vol. 89, PMLR, 16–18 Apr 2019, pp. 2611–2620. [Online]. Available: <https://proceedings.mlr.press/v89/zhe19a.html>.

- [9] K. B. Petersen and M. S. Pedersen. “The matrix cookbook.” (Nov. 15, 2012), [Online]. Available: <https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>.
- [10] A. Paszke, S. Gross, S. Chintala, *et al.*, “Automatic differentiation in pytorch,” 2017.



Name of Candidate: Zachary Bastiani  
Birth date: October 10, 1997  
Birth place: Salt Lake City, Utah  
Address: 1016 Kensington Ave S,  
Salt Lake City, Utah, 84105