EFFICIENT TECHNIQUES FOR STATISTICAL MODELING OF CALIBRATION AND SPATIO-TEMPORAL SYSTEMS USING GAUSSIAN PROCESSES

By

BABAK FARMANESH

Bachelor of Science in Industrial Engineering
Sharif University of Technology
Tehran, Iran
2014

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
August, 2018

EFFICIENT TECHNIQUES FOR STATISTICAL MODELING OF CALIBRATION AND SPATIO-TEMPORAL SYSTEMS USING GAUSSIAN PROCESSES

Dissertation Approved:

Dr. Arash Pourhabib

Dissertation Advisor

Dr. Balabhaskar Balasundaram

Committee Chair

Dr. Farzad Yousefian

Dr. Ramesh Sharda

ACKNOWLEDGMENTS

I would like to express my deepest appreciation and gratitude toward my advisor, Dr. Arash Pourhabib, for his continuous support, superb guidance, constant encouragement, and patience during my PhD study. Of course, this dissertation would not have been possible without his excellent supervision.

I would also like to greatly appreciate Dr. Balabhaskar Balasundaram for his tremendous and priceless help during the past four years. I owe much to him for his efforts to support my PhD study. Furthermore, I am truly grateful for the time, considerations, and constructive comments I received from the rest of my committee members, Dr. Farzad Yousefian and Dr. Ramesh Sharda, over these years.

I am sincerely thankful to the head of the IE&M department, Dr. Sunderesh Heragu, for all his invaluable support. I also acknowledge Dr. Austin Buchannan, Dr. Nagaraj Kalyani, Matthias Katzfuss, and Jianhua Huang for their help on various parts of this dissertation. In addition, a very special thank to Laura Brown for her flawless administrative assistance during my Ph.D journey.

Furthermore, I am thankful to all my friends, especially Solmaz Bastani, Ronny Pacheco, Shahrouz Mohagheghian, and Akash Gupta, for their kindness, their support, the enjoyable memories we made together, and all the good times.

Finally, I would like to wholeheartedly express my love and gratitude toward my parents and my sister. I am so blessed to have such a wonderful and supportive family, and dedicate this dissertation to them.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University

Name: BABAK FARMANESH

Date of Degree: August, 2018

Title of Study: EFFICIENT TECHNIQUES FOR STATISTICAL MODELING OF CALIBRATION AND SPATIO-TEMPORAL SYSTEMS USING GAUSSIAN PROCESSES

Major Field: INDUSTRIAL ENGINEERING AND MANAGEMENT

Gaussian processes (GPs) are one of the most widely used tools in statistical modeling of various engineering systems. In this dissertation, we study three common types of problems in statistical modeling, i.e., prediction, calibration, and forecasting, using GPs and other related techniques.

First, we study the problem of prediction using Gaussian Process Regression (GPR) in large-scale spatial systems that contain exogenous variables. We propose a Sparse Pseudo-input Local Gaussian Process (SPLGP) that addresses the inefficiencies of GPR, i.e., computational complexity and covariance heterogeneity, in dealing with spatial systems in a unifying framework. We propose new theorems that form the basis of our decomposition policy and develop an optimization procedure to find the optimal policy. We also impose continuity constraints on the boundaries of the subdomains to alleviate the problem of discontinuity of the global predictor.

Next, we study the calibration problem for expensive computational models (ECM), i.e., computational models that cannot be evaluated a large number of times. We propose a Bayesian Non-isometric Matching Calibration (BNMC) approach that allows calibration of ECM. The proposed model uses GPs to embrace the restrictions of ECM and makes inferences on the calibration parameters through a Bayesian framework. We also present a geometric interpretation of calibration that enables us to take advantage of combinatorial optimization techniques to extract necessary information for constructing prior distributions of our Bayesian framework.

Finally, we study the problem of forecasting in complex spatio-temporal systems with the primary focus on short-term wind speed forecasting in wind farms. We propose a similarity-based forecasting model capable of taking any type of spatial and temporal information into account to improve spatio-temporal forecasting, in particular wind speed forecasting. The proposed model is inspired by the weighted averaging technique used in a class of regression models known as non-parametric linear smoothers which includes GPR. We also equip our model with a variable selection and a parameter training procedure, so that it can be easily applied to any spatio-temporal system.

We present a set of experimental results for each problem to demonstrate the efficiency of our proposed models comparing to other existing models.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

The extensive availability of data and computational power due to recent advancements in computer technologies has offered opportunities for researchers to develop statistical models to better understand the behavior of complex real world processes. In this study, we are interested in statistical modeling of various engineering systems by using Gaussian Processes (GPs) and their variations. The current chapter first introduces GPs and some of the related methods of interest, then overviews the research topics included in this dissertation.

1.1 An introduction to Gaussian stochastic processes and related statistical models

Gaussian Stochastic Processes or Gaussian Processes (GPs) in short provide a fully probabilistic foundation based on the concept of normal (Gaussian) distribution for various problems in *statistical learning* [39]. Specifically, many applications of GPs can be found in the domain of *supervised learning*, where one seeks to find the relationship between inputs and outputs of a system of interest. These applications include regression, classification [88], time series analysis [90], and analysis of computer experiment outputs [92]. Informally, GPs generalize the concept of multivariate normal distribution to functions; therefore, any statistical learning problem that can be reduced to a problem of recovering an underlying function governing the system under study can be inferred by GPs.

We can trace back the early use of GPs to 1940's for time series analysis [50].

Later on, GPs were introduced to geostatistics, statistics, and machine learning communities [57, 67, 114]. Although, the emergence of GPs is credited to statisticians, they are also a well known concept in the field of mathematics. In fact, GPs can be studied from the mathematical point of view through the lens of *functional analysis* [111]. Therefore, most of the models developed based on GPs can be derived from purely mathematical [75] as well as statistical perspectives [88]. GPs are also closely connected to another well known class of machine learning methods known as *kernel machines*, such as Support Vector Machines [15], and Neural Networks [65]. In some sense, GP can be thought of as an interdisciplinary concept that brings independent studies of learning theory in statistics, mathematics, and machine learning communities together.

We formally introduce GPs as follows: given a probability space $(\Omega, \mathcal{S}, \mathcal{P})$, and an index set $\mathbf{T}$, where Ω is the sample space, $\mathcal{S}$ is a sigma-algebra, and $\mathcal{P}$ is a probability measure, a Gaussian process (GP) is a stochastic process where for any $\mathbf{T}' = \{t_1, \ldots, t_N\}$ as a finite subset of $\mathbf{T}$, random vector $[f_1, \ldots, f_N]^T$ follows a multivariate normal distribution [88], where f_i is a realization of a measurable function $\mathcal{F} : \Omega \to \mathbb{R}$ for a given $t_i \in \mathbf{T}'$. Note that index set $\mathbf{T}$ can simply be time [112] in processes involving time evolution, or can be a subset of $\mathbb{R}^p$ which is the case in most of the statistical learning applications [39]. In the latter case, based on the GP definition, for a given set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^p$, random vector $[f_1, f_2, \ldots, f_N]^T$ follows a multivariate normal distribution, where $f_i = \mathcal{F}(\mathbf{x}_i)$ for any $i \in [N]$. In this dissertation we denote by $[n]$ the set of positive integers smaller than or equal to n, i.e., $[n] = \{1, \ldots, n\}$.

Analogous to multivariate normal distribution, a GP can be fully specified by its mean and covariance function. In other words, when function $\mathcal{F}$ follows a GP distribution with mean function $\mathcal{M}(\cdot)$ and covariance function $\mathcal{K}(\cdot, \cdot)$,

$$\mathcal{F} \sim \mathcal{GP}\big(\mathcal{M}(\cdot), \mathcal{K}(\cdot, \cdot)\big), \tag{1.1}$$

then the mean vector and the covariance matrix of random vector $\mathbf{f}$, i.e., $\boldsymbol{\mu} = \mathbb{E}(\mathbf{f})$ and $\mathbf{K} = \mathbb{E}\left((\mathbf{f}-\boldsymbol{\mu})(\mathbf{f}-\boldsymbol{\mu})^T\right)$, where $\mathbb{E}(\cdot)$ denotes the expectation operator, can be specified by functions $\mathcal{M}(\cdot)$ and $\mathcal{K}(\cdot,\cdot)$. This means that $\mu_i = \mathbb{E}(f_i) = \mathcal{M}(\mathbf{x}_i)$, $\mu_j = \mathbb{E}(f_j) = \mathcal{M}(\mathbf{x}_j)$, and $k_{ij} = \mathbb{E}\left((f_i-\mu_i)(f_j-\mu_j)\right) = \mathcal{K}(\mathbf{x}_i,\mathbf{x}_j)$.

We note that covariance function $\mathcal{K}(\cdot,\cdot)$ is in fact a bi-argumental and real-valued function which is denoted as a *kernel* function in mathematics. However, for $\mathcal{K}(\cdot,\cdot)$ to be a valid covariance function, it must be a symmetric and positive semidefinite kernel, i.e., $\mathcal{K}(\mathbf{x},\mathbf{x}') = \mathcal{K}(\mathbf{x}',\mathbf{x})$ and

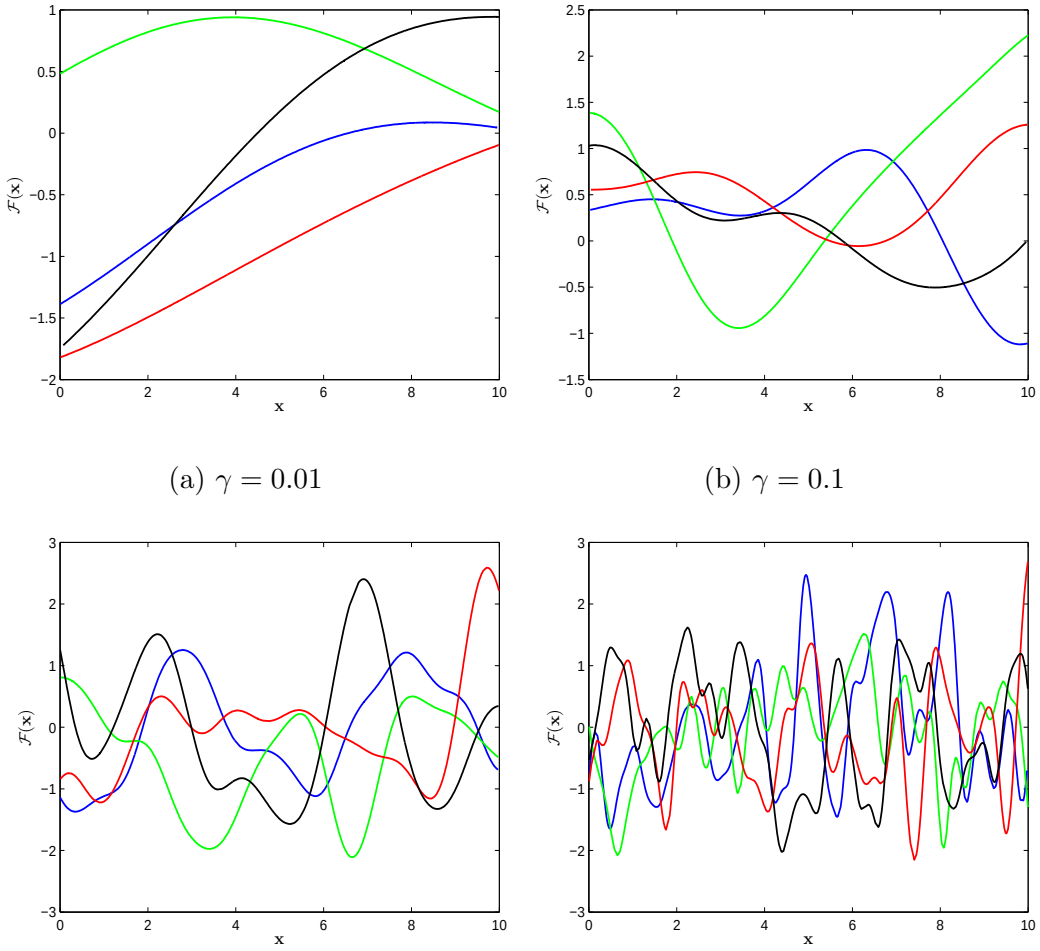
$$\int \mathcal{K}(\mathbf{x},\mathbf{x}')\mathcal{F}(\mathbf{x})\mathcal{F}(\mathbf{x}')\nu(d\mathbf{x})\nu(d\mathbf{x}') > 0, \tag{1.2}$$

for any $\mathcal{F} \in L^2(\mathbb{R}^p,\nu)$, where ν is a measure. Detailed explanation of the kernel functions and their properties can be found in [111].

The most widely used symmetric positive semidefinite kernel function as the choice of covariance function is the squared exponential kernel. This kernel in its simplest form, known as isotropic squared exponential kernel, can be written as

$$\mathcal{K}(\mathbf{x},\mathbf{x}') = \exp\left(-\gamma||\mathbf{x}-\mathbf{x}'||_2^2\right), \tag{1.3}$$

where $||\cdot||_2$ is the Euclidean norm operator, and γ is the length-scale parameter of $\mathcal{K}(\cdot,\cdot)$ that controls the smoothness of the GP. Figure 1.1 shows a few random draws from different GPs with mean function zero and covariance function (1.3) with different values of the length-scale parameter.



(a) $\gamma = 0.01$ (b) $\gamma = 0.1$

(c) $\gamma = 1$ (d) $\gamma = 10$

Figure 1.1: A few random draws from different GPs with $\mathcal{M}(\mathbf{x}) = 0$ and covariance function (1.3) with different values of γ

1.1.1 Gaussian process regression

In supervised learning, regression refers to finding a functional relationship between the sets of inputs and outputs collected from a system, where the outputs are real-valued continuous variables. Two main advantages make GPs popular in the domain of regression. First, the non-parametric nature of GPs allows the handling of complex and nonlinear functional structures. Second, GPs explain the relationship between

the sets of inputs and outputs probabilistically. The latter provides us with full predictive distributions as apposed to most of the regression models that only obtain point and confidence interval predictions.

We set up the regression problem by letting $\mathbf{D} = \{(\mathbf{x}_i, y_i) \mid i \in [N], \mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}\}$ be a training dataset containing N observations, where each observation consists of a p-dimensional input vector $\mathbf{x}_i$ and a corresponding output y_i. The main assumption in any regression modeling is the existence of an underlying function $\mathcal{F}$ that maps each $\mathbf{x}_i$ to y_i, and the goal is to infer $f_* = \mathcal{F}(\mathbf{x}_*)$ for a given $\mathbf{x}_* \in \mathbb{R}^p$. We call the problem of finding the predictive distribution, $f_*|\mathbf{y}$, i.e., the distribution of f_* given $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$, by assuming a GP on the regresion function, $\mathcal{F}$, as the Gaussian Process Regression (GPR).

In most practical settings, the training dataset consists of noise-contaminated outputs. Assuming the noise is normally distributed with zero mean and the variance σ^2, the GPR model is written as

$$y_i = \mathcal{F}(\mathbf{x}_i) + \epsilon_i \quad i \in [N], \tag{1.4}$$

where $\mathcal{F} \sim \mathcal{GP}\big(\mathcal{M}(\cdot), \mathcal{K}(\cdot,\cdot)\big)$ and each $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

There are equivalent ways to present the inference based on the GPR, for example through regularized optimization [75], optimal spatial prediction [19], or an extension of Bayesian linear regression. Here, we adopt a Bayesian treatment from [88], where we specify a prior distribution for $\mathbf{f} = [\mathcal{F}(\mathbf{x}_1), \ldots, \mathcal{F}(\mathbf{x}_N)]^T$ and then calculate the predictive distribution at $\mathbf{x}_*$. Assuming the prior distribution as $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{XX}})$ implies $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma^2\mathbf{I})$, where $\mathbf{K_{XX}}$ is the $N \times N$ covariance matrix of pairwise elements in $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{I}$ is a $N \times N$ identity matrix. To find the marginal distribution of $\mathbf{y}$ we integrate out $\mathbf{f}$, i.e.,

$$p(\mathbf{y}) = \int_\Omega p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f}, \tag{1.5}$$

which results in another normal distribution, namely $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{XX}} + \sigma^2\mathbf{I})$. There-

fore, based on the joint normality assumption,

$$[\mathbf{y}, f_*]^T \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K_{XX}} + \sigma^2\mathbf{I} & \mathbf{k_{Xx_*}} \\ \mathbf{k_{x_*X}} & k_{\mathbf{x_*x_*}} \end{bmatrix}\right), \tag{1.6}$$

where $\mathbf{k_{Xx_*}}$ is the $N \times 1$ vector of covariances between the input set $\mathbf{X}$ and $\mathbf{x}_*$, and $k_{\mathbf{x_*x_*}}$ is the variance at $\mathbf{x}_*$. Hence, the predictive distribution of f_* at $\mathbf{x}_*$ is obtained as the multivariate normal distribution,

$$f_*|\mathbf{y} \sim \mathcal{N}\left(\mathbf{k_{x_*X}}(\mathbf{K_{XX}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, k_{\mathbf{x_*x_*}} - \mathbf{k_{x_*X}}(\mathbf{K_{XX}} + \sigma^2\mathbf{I})^{-1}\mathbf{k_{Xx_*}}\right). \tag{1.7}$$

We note that prior to use the predictive distribution (1.7) for prediction, we should train the hyper-parameters of the GPR, which are variance of the error and the parameters of covariance function $\mathcal{K}$. Maximizing the logarithm of the marginal likelihood of the training data, $p(\mathbf{y})$, is the most popular approach for learning the hyper-parameters, that is

$$\max_{\boldsymbol{\theta}} -\frac{1}{2}\log|\mathbf{K_{XX}} + \sigma^2\mathbf{I}| - \frac{1}{2}\mathbf{y}^T(\mathbf{K_{XX}} + \sigma^2\mathbf{I})^{-1}\mathbf{y} - \frac{N}{2}\log 2\pi, \tag{1.8}$$

where $\boldsymbol{\theta}$ denotes as the set of the hyper-parameters in the GPR model. The optimization (1.8) can be done iteratively by numerical unconstrained optimization methods such as the methods based on gradient descent [5].

1.1.2 GPR and other non-parametric linear smoothers

GPR can also be viewed as a member of a class of regression models known as *non-parametric linear smoothers* [39]. This class contains the set of all flexible regression models that let the training data determine the shape of the regression function, $\mathcal{F}$, but restrict the point predictor for a given $\mathbf{x}_*$, i.e., $\hat{f}_*$, to satisfy

$$\hat{f}_* = \sum_{i\in[N]} w_i(\mathbf{x}_*)y_i = \mathbf{w}^T(\mathbf{x}_*)\mathbf{y}, \tag{1.9}$$

that is $\hat{f}_*$ should be a linear combination of outputs in the training dataset, where $w_i(\mathbf{x}_*)$ is a weight assigned to output y_i given $\mathbf{x}_*$ and $\mathbf{X}$. Considering the mean of

predictive distribution (1.7) as the point predictor of GPR, i.e., $\hat{f}_* = \mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$, places the GPR model in the class of non-parametric linear smoothers with the weight vector $\mathbf{w}(\mathbf{x}_*) = \mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I})^{-1}$.

The simplest regression model in this class is k-Nearest-Neighbors Regression [102]. In this model, $\hat{f}_*$ is inferred by taking the average of k nearest neighbors of $\mathbf{x}_*$. In other words, the weight $w_i(\mathbf{x}_*)$ is defined as

$$w_i(\mathbf{x}_*) = \begin{cases} \frac{1}{k} & \text{if } \mathbf{x}_i \text{ is among the } k \text{ nearest neighbors of } \mathbf{x}_* \\ 0 & \text{otherwise.} \end{cases} \tag{1.10}$$

Nadaraya-Watson Regression (NWR) [64, 110] is another non-parametric linear smoother that uses a symmetric positive semidefinite kernel $\mathcal{K}$ to weight the outputs by $w_i(\mathbf{x}_*) = \frac{\mathcal{K}(\mathbf{x}_*,\mathbf{x}_i)}{\sum_{i\in[N]}\mathcal{K}(\mathbf{x}_*,\mathbf{x}_i)}$. This weighting scheme results in the NWR point predictor,

$$\hat{f}_* = \frac{\sum_{1=1}^{N}\mathcal{K}(\mathbf{x}_*,\mathbf{x}_i)y_i}{\sum_{1=1}^{N}\mathcal{K}(\mathbf{x}_*,\mathbf{x}_i)} = \mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{k}_{\mathbf{x}_*\mathbf{X}}\mathbf{J}_N)^{-1}\mathbf{y}, \tag{1.11}$$

where $\mathbf{J}_N$ is a vector of ones with length N. NWR is closely related to (or actually a simplified version of) GPR. Superficially, we can see this relation where the two point predictors differ only by their middle terms, i.e., the inverted matrix $(\mathbf{K}_{\mathbf{XX}} + \sigma\mathbf{I})^{-1}$ in (1.7) and the fraction $(\mathbf{k}_{\mathbf{x}_*\mathbf{X}}\mathbf{J}_N)^{-1}$ in (1.11); however, authors of [47] show that the relation between GPR and NWR is more profound, and in fact, NWR converges to GPR if we continue applying NWR on the residuals of each step, i.e.,

$$\mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{k}_{\mathbf{x}_*\mathbf{X}}\mathbf{J}_N)^{-1}\mathbf{y} + \mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{k}_{\mathbf{x}_*\mathbf{X}}\mathbf{J}_N)^{-1}\mathbf{e}_1 + \ldots + \mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{k}_{\mathbf{x}_*\mathbf{X}}\mathbf{J}_N)^{-1}\mathbf{e}_{N-1} =$$

$$\mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{k}_{\mathbf{x}_*\mathbf{X}}\mathbf{J}_N)^{-1}(\mathbf{y} + \mathbf{e}_1 + \ldots + \mathbf{e}_{N-1}) \to \mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I})^{-1}\mathbf{y} \text{ as } N \to \infty, \tag{1.12}$$

where $\mathbf{e}_j$ is the vector of residuals from the j^{th} step.

Relaxing the restriction of NWR in using symmetric positive semidefinite kernels leads to another group of non-parametric linear smoothers known as similarity-based regression models. Similarity-based regression models have been developed independently and fully axiomatized in the field of economics [29]. These models use a broader

class of bi-argumental and real-valued functions, so called *similarity functions*, for weighting the outputs. Although, NWR and similarity-based regression look identical if one uses a symmetric positive semidefinite kernel for both, the system of axioms discussed in [29] conceptually distinguishes these two models.

More sophisticated versions of NWR can be found in the literature of linear smoother such as local linear and local polynomial NWR regression [20]. Also the family of spline-based models such as Regression splines, Natural splines, and Smoothing splines fit into the class of non-parametric linear smoothers [39] as well. However, spline-based models have been originally designed for univariate regression problems, and unlike kernel-based regression models cannot be easily extended to multivariate case due to theoretical and computational barriers. Therefore, in this study, we do not focus on this family.

More complex non-parametric linear smoothers use the method of *regularization* [94] to obtain $\mathcal{F}$. In the theory of statistical learning, regularization refers to the method of training the parameters of a statistical model by optimizing a loss function penalized by the norm of some parameters, which is known as *regularizer*. In other words, considering $\boldsymbol{\theta}$ as the set of the parameters of a model of interest, regularization is defined as

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathbf{y}) + \lambda \mathcal{R}(\boldsymbol{\theta}), \tag{1.13}$$

where $\mathcal{L}(\boldsymbol{\theta}, \mathbf{y})$ is a loss function, and $\lambda \mathcal{R}(\boldsymbol{\theta})$ is the regularizer or the penalty term with $\mathcal{R}(\boldsymbol{\theta})$ as its norm function, and λ as the magnitude of the penalty.

Kernel Ridge Regression (KRR) and Support Vector Regression (SVR) [94] are two regression models in the class of linear smoothers that can be derived by (1.13). The regularization problem associated with KRR and SVR in its primal variables formulation is defined as

$$\arg\min_{\mathcal{F} \in \mathcal{H}} \sum_{i \in [N]} (y_i - \mathcal{F}(\mathbf{x}_i))^2 + \lambda ||\mathcal{F}||^2_{\mathcal{H}}, \tag{1.14}$$

where $\mathcal{H}$ is a mathematical space of functions known as Reproducing Kernel Hilbert Space (RKHS) [111] with an associated kernel $\mathcal{K}(\cdot,\cdot)$, and $||\cdot||_{\mathcal{H}}$ is the norm operator on $\mathcal{H}$. With the assumption that $\mathcal{F} \in \mathcal{H}$, the minimizer of optimization (1.14) can be found by the well known *representer theorem* [93], which is one of the most important theorems in statistical learning. It turns out that the optimal regression function found by (1.14) results in exactly the same point predictor as that of GPR, i.e., $\hat{f}_* = \mathbf{k}_{\mathbf{x}_*\mathbf{X}}(\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$. In fact, this is an example of how a GP model can be derived from a completely different point of view as mentioned in Section 1.1. However, we note that what makes GPR a superior model over SVR and KRR is the ability of GPR to attain the full predictive distribution simply by its construction, which is not the case in the other two models.

1.2 Research projects and objectives

This dissertation includes three research projects on statistical modeling of three engineering systems. The common feature between all these three research projects is the presence of GPs and their related techniques. The first research project directly focuses on GPR and studies a set of treatments that enables GPR to handle a special class of systems known as *spatial systems*. The second research project uses GPs to address the problem of *calibration* of computational models. Finally, the last research project investigates the potential of non-parametric linear smoothers in statistical modeling of *spatio-temporal systems*. The rest of this section is dedicated to an overview of each of these research projects.

1.2.1 Approximating GPR for spatial systems

GPR is a powerful tool in the analysis of *spatial systems*, i.e., systems whose data is collected across space [2, 19]. However, GPR does not scale efficiently to large datasets because its implementation entails inverting matrices of size N with computational

complexity of order $\mathcal{O}(N^3)$. In addition, many spatial systems have highly non-stationary covariance structures, i.e., the covariance structure varies in different parts of the input domain, which cannot be modeled effectively with a single covariance function. This problem is exacerbated when the spatial system contains additional explanatory variables known as *exogenous variables*. Hence, despite the strengths of GPR mentioned in Section 1.1, this regression model becomes inefficient in dealing with most spatial systems.

In the first research topic of this dissertation, we propose an approximating model that effectively captures the non-stationarity, and at the same time, speeds up the GPR for large non-stationary spatial datasets with exogenous variables . The new model is based on a decomposition method that partitions the domain of data and builds a local model in each partition. Our study includes a solid theoretical framework that instructs how to efficiently decompose a GPR model to smaller sub-models, train the sub-models, and handle boundary conditions. We present a comprehensive experimental study by applying our model to various real world datasets to indicate its efficiency in handling massive spatial datasets comparing to other competing models.

1.2.2 Calibration of expensive computational models using GPs, Bayesian statistics, and combinatorial optimization

With the immense amount of computational power available in the twenty first century, analysis of complex physical systems through computational models has become an appealing approach for practitioners due to their computational and cost efficiency. The computational models are usually built upon mathematical models involving various variables that may or may not be controllable/observable in the physical systems. Specifying those uncontrollable/unobservable variables, which are known as *calibration variables*, is crucial for building computational models capable of resembling their

associated physical systems. This problem is referred to as the *calibration problem* in statistical learning. Most of the statistical/mathematical approaches in calibration involve numerical optimization techniques requiring the computational models to be executed a very large number of times. Therefore, in the case of having a limited budget for evaluating the computational models, i.e., expensive computational model assumption, these calibration approaches fail to perform efficiently.

In the second research topic of this dissertation, we address the calibration problem under the assumption of having an expensive computational model. The proposed model embraces this practical restriction by taking advantage of a unique combination of combinatorial optimization, GPs, and Bayesian statistics. Our study discusses how to model the calibration problem as a GP as well as a combinatorial optimization problem, and how to bridge them by Bayesian statistics to make inferences on the calibration variables. We include a set of numerical experiments on a few real and synthetic datasets to demonstrate the efficiency of the proposed model comparing to the other existing models under the assumption of expensive computational models.

1.2.3 An application of non-parametric linear smoothers in spatio-temporal forecasting with a focus on wind-speed forecasting

Spatio-temporal systems refer to those systems whose data is collected across space and time, such as weather and climate systems. Building statistical models for these systems capable of making prediction over space as well as forecasting over time has received a considerable attention in the last two decades in the literature of statistical learning [18, 19, 101].

In the third research topic of this dissertation, we develop a forecasting model for inference of complex spatio-temporal systems through the class of non-parametric linear smoothers. The proposed model specifically adapts the idea of similarity-based regression model introduced in Section 1.1.2 to the domain of forecasting to build

a flexible and fast, but accurate forecasting model that takes into account any type spatial and temporal information.

Our primary spatio-temporal systems of interest are wind farms, which are used to generate electricity out of wind energy. In particular, we are interested in short-term wind speed forecasting that plays a critical role in making the wind energy a reliable source of energy [30, 40]. However, we note that our methodology is general and can be efficiently applied to any other spatio-temporal system. We conduct an experimental study on a historical dataset of a wind farm to compare the ability of the proposed model to other competing models.

1.3 Organization of the dissertation

Chapters 2, 3,and 4, are dedicated to the three research topic explained in Section 1.2, respectively. Each chapter consists of a separate introduction followed by a methodology, experimental results, and a summary section. Each introduction section includes a literature review, stating the research gap, and an overview of the proposed methodology. The methodology sections contain the statements of the problems as well as all the theoretical arguments and computational analysis. The methodology sections are supported by the results in the experimental results sections. Finally, each research topic is summarized in the closing section.

The final chapter of this dissertation, Chapter 5, concludes the dissertation by summarizing the major contributions and presenting some interesting future research directions.

CHAPTER 2

SPARSE PSEUDO-INPUT LOCAL GAUSSIAN PROCESS REGRESSION FOR LARGE NON-STATIONARY SPATIAL DATASETS WITH EXOGENOUS VARIABLES

2.1 Introduction

Building flexible, accurate, and fast predictive models is of paramount importance in spatial systems that include environmental factors such as temperature or irrigation as dependent variables [26, 118]. We call these environmental factors exogenous variables to distinguish them from simple spatial information such as latitude, longitude, and altitude.

The challenge in building predictive models for such systems arise because the covariance structure changes for different locations in the space, i.e., the covariance is non-stationary, primarily when the behavior of the response variable strongly depends on the underlying geology [49]. This problem is exacerbated when the spatial data contains exogenous variables. It is also challenging to manage the large number of observations in a computationally efficient manner.

Theoretically, GPR can benefit from very large datasets since it is a non-parametric model whose flexibility and performance generally increase by having more data points [39]. However, the computational complexity of GPR is dominated by the inversion of covariance matrices which is of $\mathcal{O}(N^3)$, where N is the number of data points. Hence, approximation techniques are generally employed to reduce the computational cost and burden (see [115, 98, 99, 86, 83]).

One idea that improves GPR to tackle inhomogeneous covariance structures for

large datasets is through the class of local GPR methods that assume distinct covariance functions for each region of the domain of data. Local GPR methods use various partitioning policies that decompose the domain into smaller subdomains and apply local GPR in each subdomain [35, 70, 33]. Therefore, local GPR methods reduce the total computational complexity to $O(Nn^2)$, where n is the number of local data points. This idea, however, presents two related problems: devising an efficient partitioning policy, and ensuring continuity in prediction on the boundaries of the subdomains.

The Domain Decomposition Method (DDM) [70], for example, uses *uniform mesh*, which partitions the domain of the input data into rectangles, and then addresses the discontinuity of GPR on the boundaries by defining a few control points and imposing equality of neighboring local predictors at these points. While the superiority of DDM over similar methods such as weighted average techniques [107, 87, 33, 104] emerged as a result of its ability to effectively handle boundary conditions, it cannot handle spatial datasets with exogenous variables, i.e., higher than two-dimensional input domains. This limitation arises because the number of the boundaries required for DDM's implementation in higher dimensions reduces the efficiency of the method.

Motivated by these challenges, this study proposes a new method, Sparse Pseudo-input Local Gaussian Process (SPLGP), which preserves the continuity of GPR in higher dimensional spaces after partitioning the domain of data into subdomains. SPLGP employs parallel hyperplanes to partition domains, thus minimizing the number of boundaries and simplifying the boundary conditions. This allows us to impose the continuity constraints on boundaries of subdomains in higher dimensional spaces. Our partitioning policy comes at the price of large-size subdomains, which makes the application of the full GPR impractical in each subdomain. We overcome this limitation by using covariance approximation methods for each region. We develop an optimization algorithm that finds the optimal hyperplanes that result in lower

errors for the covariance approximations in each region. We provide the theoretical justification based on the analysis of covariance structure for the use of optimal parallel hyperplanes to create the subdomains. Therefore, our proposed model seamlessly integrates a partitioning policy into local approximation to improve prediction accuracy.

The computational complexity of SPLGP is in the order of $\mathcal{O}(Nm^2 + S(\Delta^3 + Q^3) + n^3)$, where m is the average number of local pseudo-inputs in each subdomain, Q is the average number of control points on each boundary, Δ is the average number of neighboring data points of each boundary, and n is the size of a small subset of the training dataset. However, for most practical applications, the complexity is dominated by Nm^2 (similar to the most efficient existing methods) in moderate dimensional spaces, since n and Δ are less than a few hundreds. Moreover, although SPLGP is motivated and developed to build predictive models for spatial data with exogenous variables, the methodology is general and can be efficiently applied to any large dataset with a moderate number of input variables. Our numerical studies demonstrate that SPLGP outperforms, or performs as well as, the competing models in terms of computation time or accuracy on spatial data with or without exogenous variables and moderate dimensional non-spatial data.

The remainder of this chapter is organized as follows. Section 2.2 briefly introduces few approximation methods that are relevant to SPLGP's implementation. Section 2.3 explains SPLGP including domain partitioning, training local models subject to boundary conditions, and choosing directions of partitioning. Section 2.4 compares the proposed model to commonly used models, and Section 2.5 summarizes the chapter.

2.2 A few GPR approximation methods

Low-rank covariance approximation methods [115, 86] reduce the computational complexity of GPR by approximating the original covariance matrix by the Nyström method,

$$\mathbf{K_{XX}} \approx \tilde{\mathbf{K}}_{\mathbf{XX}} = \mathbf{K}_{\mathbf{X}\tilde{\mathbf{X}}}\mathbf{K}^{-1}_{\tilde{\mathbf{X}}\tilde{\mathbf{X}}}\mathbf{K}_{\tilde{\mathbf{X}}\mathbf{X}}, \tag{2.1}$$

where $\tilde{\mathbf{X}}$ is either a subset of $\mathbf{X}$ or a set of unobserved *pseudo-inputs*, which are a new set of parameters used to approximate the original covariance matrix of GPR. This approximation reduces the computational complxity of GPR to the order of $\mathcal{O}(Nm^2)$, where m is the size of $\tilde{\mathbf{X}}$ and $m \ll N$.

In particular, Sparse Pseudo-input Gaussian Process (SPGP) [100] assumes that observations $\mathbf{y}$ are conditionally independent, given the *pseudo-outputs* $\tilde{\mathbf{f}} = [\tilde{f}_1, \ldots, \tilde{f}_m]^T$ defined on pseudo-input set $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_m\}$. This implies the joint Gaussian likelihood,

$$[\mathbf{y}, f_*]^T \sim \mathcal{N}\left(0, \begin{bmatrix} \tilde{\mathbf{K}}_{\mathbf{XX}} + \mathrm{diag}(\mathbf{K_{XX}} - \tilde{\mathbf{K}}_{\mathbf{XX}}) + \sigma^2\mathbf{I} & \tilde{\mathbf{k}}_{\mathbf{X}\mathbf{x}_*} \\ \tilde{\mathbf{k}}_{\mathbf{x}_*\mathbf{X}} & k_{\mathbf{x}_*\mathbf{x}_*} \end{bmatrix}\right), \tag{2.2}$$

and the predictive mean and variance,

$$\hat{\mu}(f_*|\mathbf{y}) = \tilde{\mathbf{k}}_{\mathbf{x}_*\mathbf{X}}(\tilde{\mathbf{K}}_{\mathbf{XX}} + \mathrm{diag}(\mathbf{K_{XX}} - \tilde{\mathbf{K}}_{\mathbf{XX}}) + \sigma^2\mathbf{I})^{-1}\mathbf{y}, \tag{2.3}$$

$$\hat{\sigma}^2(f_*|\mathbf{y}) = k_{\mathbf{x}_*\mathbf{x}_*} - (\tilde{\mathbf{K}}_{\mathbf{XX}} + \mathrm{diag}(\mathbf{K_{XX}} - \tilde{\mathbf{K}}_{\mathbf{XX}}) + \sigma^2\mathbf{I})^{-1}\tilde{\mathbf{k}}_{\mathbf{X}\mathbf{x}_*}, \tag{2.4}$$

where $\tilde{\mathbf{k}}_{\mathbf{X}\mathbf{x}_*}$ is the low-rank covariance vector between $\mathbf{X}$ and the test data point $\mathbf{x}_*$ calculated by (2.1).

Although low-rank approximation methods reduce the computational complexity of GPR, they do not address the heterogeneity in the covariance structure. One alternative is to use local GPR methods, which partition the domain of data into smaller pieces, or subdomains, and train local predictors independently. Training

multiple local models instead of one global model enables local GPR methods to use different covariance functions for each subdomain. Moreover, dealing with local covariance matrices reduces the computational complexity of GPR to $O(Nn^2)$, where n is the number of data points in each subdomain. Commonly used methods in this class include Bayesian committee machine [104], local probabilistic regression [107], mixture of Gaussian process experts [87], the treed Gaussian process model [33], and Domain Decomposition Method (DDM) [70].

We are interested in DDM since it effectively addresses the discontinuity in prediction on the boundaries of subdomains. DDM decomposes the domain of data into S disjoint subdomains Ω_s for all $s \in [S]$, and uses a set of control points on its boundaries to impose connectivity constraints. Denoting $\mathbf{X}_s$ as the set of training data points belonging to Ω_s, DDM obtains the local mean predictor,

$$\hat{\mu}_s(f_*|\mathbf{y}) = \mathbf{k}_{\mathbf{x}_*\mathbf{X}_s}(\sigma_s^2\mathbf{I} + \mathbf{K}_{\mathbf{X}_s\mathbf{X}_s})^{-1}\mathbf{y}_s + c_s(\mathbf{x}_*), \tag{2.5}$$

where the first term is the standard local GPR mean predictor and the second term, $c_s(\mathbf{x}_*)$, is a correction term that appears because of the connectivity constraints, and approaches to zero as the test data points move away from the boundaries. However, DDM is applied only to two-dimensional datasets, because the boundary conditions become more complex and calculating correction terms $c_s(\mathbf{x}_*)$ becomes more difficult for higher dimension problems.

2.3 Sparse pseudo-input local Guassian process

This section describes our proposed model, Sparse Pseudo-input Local Gaussian Process (SPLGP), where we partition the domain of data into smaller subdomains with simple boundary structures, train local predictors that utilize a low-rank covariance matrix in each subdomain, and connect neighboring local predictors on their joint boundaries to obtain a continuous global predictor. To partition the input domain,

we use parallel hyperplanes, i.e., $(p-1)$-dimensional linear spaces embedded in a p-dimensional space (see Section 2.3.3 for the details). This partitioning policy has two main advantages: it minimizes the number of boundaries, because for S subdomains, we only need $S-1$ parallel hyperplanes regardless of the dimension of the input space, and it creates simple boundary conditions (see Section 2.3.1), because each boundary is shared by exactly two subdomains. Hence, we only need to attach two local predictors on each boundary, however, the drawback is that the partitioning policy can result in very large subdomains, where a full GPR is inefficient. We overcome this problem by using covariance approximation techniques that utilize pseudo-inputs.

Among an infinite possible ways to partition a domain by parallel hyperplanes, we seek those that result in more accurate local predictors, i.e., the covariance approximation in each subdomain has the smallest error. We present two theorems that together determine the optimal policy for creating subdomains. We begin by presenting the local mean and variance calculations, assuming the subdomains have already been determined. Then we discuss justifications for the proposed parallel hyperplanes for creating subdomains. Finally, we explain practical aspects of SPLGP's implementation such as constructing hyperplanes, hyper-parameter learning, and selection of control points in Sections 2.3.3 and 2.3.4.

2.3.1 Mean and variance prediction

Let $\Omega \in \mathbb{R}^p$ denote the input domain, i.e., $\mathbf{x} \in \Omega$. We partition Ω into S subdomains Ω_s for $s \in [S]$ such that $\bigcup_{s=1}^{S} \Omega_s = \Omega$, and $\Omega_s \cap \Omega_{s'} = \emptyset$ for $s \neq s'$. We also denote $\mathbf{X}_s = \{\mathbf{x}_i \in \mathbf{X} \mid \mathbf{x}_i \in \Omega_s\}$ and $\mathbf{y}_s$ as the vector of observations corresponding to $\mathbf{X}_s$. The partitioning scheme explained in Sections 2.3.2 and 2.3.3 can lead to subdomains containing a large number of training data points, which makes the application of a full GPR inefficient. Therefore, for each Ω_s, we use m_s local pseudo-inputs $\tilde{\mathbf{X}}_s =$

$\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{m_s}\} \in \Omega_s$ to form the local and low-rank covariance approximation,

$$\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s} = \mathbf{K}_{\mathbf{X}_s\tilde{\mathbf{X}}_s}\mathbf{K}^{-1}_{\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s}\mathbf{K}_{\tilde{\mathbf{X}}_s\mathbf{X}_s}. \tag{2.6}$$

It is easy to check that among all the linear predictors $\mu(f_*|\mathbf{x}_*) = \mathbf{u}(\mathbf{x}_*)^T\mathbf{y}$, where $\mathbf{u}(\mathbf{x}_*) \in \mathbb{R}^n$ and $[\mathbf{y}, f_*]^T$ follows distribution (1.6), the GPR mean predictor minimizes the expected squared error, $\mathbb{E}\big((\mu(f_*|\mathbf{x}_*) - f_*)^2\big)$. We extend this idea to find the local and low-rank predictor for each subdomain by assuming that $[\mathbf{y}_s, f_*]^T$ follows the local version of SPGP's joint likelihood distribution (2.2). As such, we solve

$$\begin{aligned} \min_{\mathbf{u}_s(\mathbf{x}_*)} \quad & \mathbb{E}\big((\mathbf{u}_s(\mathbf{x}_*)^T\mathbf{y}_s - f_*)^2\big) \\ \text{s.t.} \quad & [\mathbf{y}_s, f_*]^T \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s} + \operatorname{diag}(\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s} - \tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s}) + \sigma_s^2\mathbf{I}_s & \tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} \\ \tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} & k_{\mathbf{x}_*\mathbf{x}_*} \end{bmatrix}\right), \end{aligned} \tag{2.7}$$

where $\mathbf{u}_s(\mathbf{x}_*)$ is the local version of $\mathbf{u}(\mathbf{x}_*)$, $\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*}$ is the covariance vector between the test data point $\mathbf{x}_* \in \Omega_s$ and $\mathbf{X}_s$ using low-rank approximation formula (2.6). Expanding the objective function with respect to the constraint in (2.7) and removing $k_{\mathbf{x}_*\mathbf{x}_*}$, which does not depend on $\mathbf{u}_s(\mathbf{x}_*)$, results in the unconstrained optimization problem for each Ω_s,

$$\min_{\mathbf{u}_s(\mathbf{x}_*)} \quad \mathbf{u}_s(\mathbf{x}_*)^T(\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s} + \operatorname{diag}(\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s} - \tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s}) + \sigma_s^2\mathbf{I}_s)\mathbf{u}_s(\mathbf{x}_*) - 2\mathbf{u}_s(\mathbf{x}_*)^T\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} \tag{2.8}$$

Note that setting the derivative of the objective function to zero gives

$$\mathbf{u}_s^{\text{opt}}(\mathbf{x}_*) = (\tilde{\mathbf{K}}_{\mathbf{X}_s\mathbf{X}_s} + \operatorname{diag}(\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s} - \tilde{\mathbf{K}}_{\mathbf{X}_s\mathbf{X}_s}) + \sigma^2\mathbf{I})^{-1}\tilde{\mathbf{k}}_{\mathbf{X}_s\mathbf{x}_*},$$

which is the SPGP's mean predictor for subdomain Ω_s. Next, we modify the optimization problem to alleviate the problem of discontinuity in the predictions on the boundaries.

To impose continuity on the boundaries, we use a small number of *control points* on the boundaries of each subdomain [69]. Let $\mathbf{B}_s$ be the set of all the control points

located on the boundaries of Ω_s. We intend to force local predictor $\mathbf{u}_s(\mathbf{x}_*)^T\mathbf{y}_s$ to be equal to the boundary values at the control point locations in $\mathbf{B}_s$,

$$\mathbf{u}_s(\mathbf{b}_i)^T\mathbf{y}_s = \mathcal{R}(\mathbf{b}_i) \quad \forall \mathbf{b}_i \in \mathbf{B}_s, \tag{2.9}$$

where $\mathcal{R}(\mathbf{b}_i)$ is a function that evaluates each $\mathbf{b}_i$ (see Section 2.3.3 for the details). Adding constraints (2.9) to local models (2.8) gives the constrained local optimization for each Ω_s,

$$\begin{aligned} \min_{\mathbf{u}_s(\mathbf{x}_*)} \quad & \mathbf{u}_s(\mathbf{x}_*)^T(\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s} + \mathrm{diag}(\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s} - \tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s}) + \sigma_s^2\mathbf{I}_s)\mathbf{u}_s(\mathbf{x}_*) - 2\mathbf{u}_s(\mathbf{x}_*)^T\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} \\ \text{s.t.} \quad & \mathbf{u}_s(\mathbf{b}_i)^T\mathbf{y}_s = \mathcal{R}(\mathbf{b}_i) \quad \forall \mathbf{b}_i \in \mathbf{B}_s. \end{aligned} \tag{2.10}$$

Note that the objective function is a convex function of $\mathbf{u}_s(\mathbf{x}_*)$ and that all of the boundary constraints in (2.10) are affine; therefore, the duality gap is zero [5], which allows maximizing the Lagrangian of (2.10) instead,

$$\begin{aligned} \mathcal{L}_s(\mathbf{u}_s(\mathbf{x}_*), \boldsymbol{\lambda}_s(\mathbf{x}_*)) = \mathbf{u}_s(\mathbf{x}_*)^T(\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s} + \mathrm{diag}(\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s} - \tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s}) + \sigma_s^2\mathbf{I}_s)\mathbf{u}_s(\mathbf{x}_*) \\ -2\mathbf{u}_s(\mathbf{x}_*)^T\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} - \sum_{i=1:|\mathbf{B}_s|} \lambda_{is}(\mathbf{x}_*)(\mathbf{u}_s(\mathbf{b}_i)^T\mathbf{y}_s - \mathcal{R}(\mathbf{b}_i)), \end{aligned} \tag{2.11}$$

where $|\mathbf{B}_s|$ is the number of all the control points located on the boundaries of subdomain Ω_s, and $\boldsymbol{\lambda}_s(\mathbf{x}_*) = [\lambda_{1s}(\mathbf{x}_*), \ldots, \lambda_{|\mathbf{B}_s|s}(\mathbf{x}_*)]^T$ is the vector of the Lagrange multipliers.

Assuming $\mathbf{u}_s(\mathbf{x}_*)$ depends on the covariance between $\mathbf{x}_*$ and $\mathbf{X}_s$, and $\lambda_{is}(\mathbf{x}_*)$ depends on the covariance of $\mathbf{b}_i$ and $\mathbf{x}_*$, we write $\mathbf{u}_s(\mathbf{x}_*) = \mathbf{H}_s\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*}$ and $\lambda_{is}(\mathbf{x}^*) = \beta_{is}\tilde{k}^s_{\mathbf{b}_i\mathbf{x}_*}$ as suggested in [70], where $\mathbf{H}_j$ is a squared matrix with size equal to the number of data points in Ω_s, and β_{is} is the Lagrange parameter associated with λ_{is} that does not depend on $\mathbf{x}_*$. Consequently, we rewrite Lagrangian (2.11) as

$$\begin{aligned} \mathcal{L}(\mathbf{H}_s, \boldsymbol{\beta}_s) = \tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s}\mathbf{H}_s^T(\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s} + \mathrm{diag}(\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s} - \tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s}) + \sigma_s^2\mathbf{I}_s)\mathbf{H}_s\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} \\ -2\tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s}\mathbf{H}_s^T\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} - \tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{B}_s}\boldsymbol{\beta}_s(\tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s}\mathbf{H}_s^T\mathbf{y}_s - \mathbf{r}_s), \end{aligned} \tag{2.12}$$

where $\boldsymbol{\beta}_s$ is a diagonal matrix with diagonal elements $\beta_{1s}, \ldots, \beta_{|\mathbf{B}_s|s}$, and $\mathbf{r}_s$ is the vector of boundary values of Ω_s, i.e., $\mathbf{r}_s = [\mathcal{R}(\mathbf{b}_1), \ldots, \mathcal{R}(\mathbf{b}_{|\mathbf{B}_s|})]^T$

Due to convexity of function (2.12) we can calculate the optimal values of $\mathbf{H}_s$ and $\boldsymbol{\beta}_s$ analytically by writing out the first order necessary conditions,

$$\frac{d\mathcal{L}(\mathbf{H}_s, \boldsymbol{\beta}_s)}{d\mathbf{H}_s} = 2(\mathbf{G}_s\mathbf{H}_s - \mathbf{I}_s)\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*}\tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s} - \mathbf{y}_s\tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{B}_s}\boldsymbol{\beta}_s\tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s} = 0, \quad (2.13)$$

$$\frac{d\mathcal{L}(\mathbf{H}_s, \boldsymbol{\beta}_s)}{d\beta_{is}} = \tilde{\mathbf{k}}_{\mathbf{b}_i\mathbf{X}_s}\mathbf{H}_s^T\mathbf{y}_j - r_{is} = 0 \quad \forall i \in [|\mathbf{B}_s|], \quad (2.14)$$

where $\mathbf{G}_s = (\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s} + \text{diag}(\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s} - \tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{X}_s}) + \sigma_s^2\mathbf{I}_s)$, and r_{is} is the i^{th} element of the vector $\mathbf{r}_s$. Reordering equation (2.13),

$$(\tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s} + 0.5(\tilde{\mathbf{k}}^j_{\mathbf{x}_*\mathbf{X}_s}\tilde{\mathbf{k}}^j_{\mathbf{X}_s\mathbf{x}_*})^{-1}\tilde{\mathbf{k}}^j_{\mathbf{x}_*\mathbf{X}_s}\tilde{\mathbf{K}}^j_{\mathbf{X}_s\mathbf{B}_s}\boldsymbol{\beta}_s\tilde{\mathbf{k}}^j_{\mathbf{B}_s\mathbf{x}_*}\mathbf{y}_s^T)\mathbf{G}_s^{-1}\mathbf{y}_s = \tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s}\mathbf{H}_s^T\mathbf{y}_s, \quad (2.15)$$

and evaluating it at the boundary locations give the system of equations with $|\mathbf{B}_s|$ equations and Lagrangian parameters,

$$(\tilde{\mathbf{k}}^s_{\mathbf{b}_i\mathbf{X}_s} + 0.5(\tilde{\mathbf{k}}^s_{\mathbf{b}_i\mathbf{X}_s}\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{b}_i})^{-1}\tilde{\mathbf{k}}^s_{\mathbf{b}_i\mathbf{X}_s}\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{B}_s}\boldsymbol{\beta}_s\tilde{\mathbf{k}}^s_{\mathbf{B}_s\mathbf{b}_i}\mathbf{y}_s^T)\mathbf{G}_s^{-1}\mathbf{y}_s = r_{is} \ \ \forall i \in [|\mathbf{B}_s|]. \quad (2.16)$$

After some simple matrix algebra, we obtain the solution to the system of linear equations (2.16),

$$\boldsymbol{\beta}_s = \frac{\mathbf{I}_s(\mathbf{r}_s - \tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s}\mathbf{G}_s^{-1}\mathbf{y}_s)\left(\left(\text{diag}^{-1}(\tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s}\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{B}_s})(\tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s}\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{B}_s})\right) \circ \mathbf{K}^s_{\mathbf{B}_s\mathbf{B}_s}\right)^{-1}}{0.5\mathbf{y}_s^T\mathbf{G}_s^{-1}\mathbf{y}_s} \quad (2.17)$$

Using the values of $\boldsymbol{\beta}_s$ from (2.17) easily obtains the solution to $\mathbf{u}(\mathbf{x}_*)$ from (2.13),

$$\mathbf{u}_s(\mathbf{x}_*) = \mathbf{H}_s\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} = \mathbf{G}_s^{-1}(\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} + \mathbf{w}_s), \quad (2.18)$$

where $\mathbf{w}_s = 0.5(\tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s}\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*})^{-1}\mathbf{y}_s\tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{B}_s}\boldsymbol{\beta}_s\tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s}\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*}$. Consequently, we obtain the local mean predictor for Ω_s,

$$\hat{\mu}_s(f_*|\mathbf{x}_*) = \tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s}\mathbf{G}_s^{-1}\mathbf{y}_s + \mathbf{w}_s^T\mathbf{G}_s^{-1}\mathbf{y}_s. \quad (2.19)$$

The objective function of local problem (2.8) is in fact the local variance predictor. Therefore plugging $\mathbf{u}_s(\mathbf{x}_*)$ into (2.8) obtains the predictive variance for Ω_s,

$$\begin{aligned}\hat{\sigma}_s^2(f_*|\mathbf{x}_*) &= k_{\mathbf{x}_*\mathbf{x}_*} - \tilde{\mathbf{k}}^s_{\mathbf{x}_*\mathbf{X}_s}\mathbf{G}_s^{-1}\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*} \\ &+\tilde{\mathbf{k}}^j_{\mathbf{x}_*\mathbf{X}_s}\mathbf{G}_s^{-1}\mathbf{w}_s + \mathbf{w}_s^T\mathbf{G}_s^{-1}\mathbf{w}_s - \mathbf{w}_s^T\mathbf{G}_s^{-1}\tilde{\mathbf{k}}^s_{\mathbf{X}_s\mathbf{x}_*},\end{aligned} \tag{2.20}$$

where $k_{\mathbf{x}_*\mathbf{x}_*}$ is the constant initially removed from the optimization. Note that in both (2.19) and (2.20), the first term is exactly the predictive mean and variance of local SPGP, and the following terms, which are amplified for local points close to the boundaries, appear to maintain the continuity of the global predictive function.

Equations (2.19) and (2.20) imply inverting matrices of size $n_s \times n_s$, which is time consuming for large n_s. However, using Woodbury, Sherman and Morrison matrix inversion lemma [36] reduces the computational complexity from $\mathcal{O}(n_s^3)$ to $\mathcal{O}(n_s m_s^2)$, where $m_s \ll n_s$. Consequently, the total computational complexity of training the local models becomes $\mathcal{O}(Nm^2)$, where m is the average number of pseudo-inputs in each subdomain. Moreover, solving the system of linear equations (2.16) involves inverting low rank covariance matrix $\mathbf{G}_s$ and boundary covariance matrix $[(\mathrm{diag}(\tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s}\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{B}_s}))^{-1}(\tilde{\mathbf{K}}^s_{\mathbf{B}_s\mathbf{X}_s}\tilde{\mathbf{K}}^s_{\mathbf{X}_s\mathbf{B}_s})] \circ \mathbf{K}^s_{\mathbf{B}_s\mathbf{B}_s}$, which have computational complexity of $\mathcal{O}(n_s m_s^2)$ and $\mathcal{O}(|\mathbf{B}_s|^3)$, respectively. Therefore, the total complexity of solving the system of linear equations (2.16) is $\mathcal{O}(S|\mathbf{B}|^3 + Nm^2)$, where $|\mathbf{B}|$ is the average number of control points for each subdomain. The next section describes our theoretical framework for creating subdomains Ω_s using parallel hyperplanes.

2.3.2 Subdomain selection

Recall that each subdomain Ω_s uses a low-rank approximation for its covariance matrix. Therefore, a natural criterion is to look for subdomains such that the error for this approximation is minimized. To this end, suppose the covariance function $\mathcal{K}(\cdot,\cdot)$ is a symmetric positive semidefinite kernel with the associated Reproducing

Kernel Hilbert Space (RKHS), $\mathcal{H}$. With such a $\mathcal{K}(\cdot,\cdot)$, optimization

$$\min_{\mathbf{c}} \mathcal{L} = \left\| \mathcal{K}(\mathbf{z},\cdot) - \sum_{i\in[m_s]} c_i \mathcal{K}(\tilde{\mathbf{x}}_i,\cdot) \right\|_{\mathcal{H}}, \tag{2.21}$$

where $\mathbf{c} = [c_1,\ldots,c_{m_s}]^T$, $\|\cdot\|_{\mathcal{H}}$ is the norm in $\mathcal{H}$, and $\mathbf{z} \in \Omega_s$, has solution $\mathbf{c}^* = \mathbf{K}^{-1}_{\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s}\mathbf{k}_{\tilde{\mathbf{X}}_s\mathbf{z}}$ [98]. Using $\mathbf{z} = \mathbf{x}_i$ for all $\mathbf{x}_i \in \mathbf{X}_s$ in (2.21) and minimizing the sum over all terms obtains $\mathbf{K}_{\mathbf{X}_s\tilde{\mathbf{X}}_s}\mathbf{K}^{-1}_{\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s}\mathbf{K}_{\tilde{\mathbf{X}}_s\mathbf{X}_s}$, which is the low-rank approximation of $\mathbf{K}_{\mathbf{X}_s\mathbf{X}_s}$ in (2.6).

In fact, the optimal value of (2.21) measures the error of approximating $\mathcal{K}(\mathbf{z},\cdot)$, given $\tilde{\mathbf{X}}_s$. Assuming $\mathcal{K}(\mathbf{z},\mathbf{z}) = h$, this error is equal to $h - \mathbf{k}_{\mathbf{z}\tilde{\mathbf{X}}_s}\mathbf{K}^{-1}_{\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s}\mathbf{k}_{\tilde{\mathbf{X}}_s\mathbf{z}}$, which is similar in form to the power function [109]. Here, the objective is to create subdomain Ω_s for which the expected error, i.e.,

$$\mathbb{E}_{\Omega_s}(h - \mathbf{k}_{\mathbf{z}\tilde{\mathbf{X}}_s}\mathbf{K}^{-1}_{\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s}\mathbf{k}_{\tilde{\mathbf{X}}_s\mathbf{z}}), \tag{2.22}$$

where the expectation operator is with respect to all $\mathbf{z}$ and $\tilde{\mathbf{X}}_s$ over Ω_s, is minimized. Since the expected error has a complicated form and its direct calculation is a challenging task, we seek an upper bound for this term and minimize that instead.

Theorem 2.1 *Let $\mathcal{K}(\cdot,\cdot)$ denote a stationary covariance function, and $h = \mathcal{K}(\mathbf{t},\mathbf{t}) \in \mathbb{R}$ be the evaluation of kernel $\mathcal{K}(\cdot,\cdot)$ at an arbitrary point $\mathbf{t} \in \Omega_s$. Then, $h\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x},\mathbf{x}')) \leq \mathbb{E}_{\Omega_s}(\mathbf{k}_{\mathbf{z}\tilde{\mathbf{X}}_s}\mathbf{K}^{-1}_{\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s}\mathbf{k}_{\tilde{\mathbf{X}}_s\mathbf{z}})$, where $\mathbf{x},\mathbf{x}',\mathbf{z},\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_{m_s}$ are i.i.d random vectors sampled from subdomain Ω_s according to some probability distribution $\mathscr{P}$.*

Theorem 2.1 provides an upper bound, i.e., $h(1 - \mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x},\mathbf{x}')))$, on expected error (2.22) (See Appendix B for a simulation study showing that the relation between $\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x},\mathbf{x}'))$ and expected error (2.22) is more profound. In fact, under certain conditions by increasing $\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x},\mathbf{x}'))$, the expected error term itself monotonically decreases). Therefore, we seek to construct the subdomains such that the expected

covariance squared is maximized, i.e., the upper bound of expected error (2.22) is minimized.

For our theoretical framework, we consider a general scenario where, after standardizing the data, Ω is (or is inscribed in) a hypercube with edge length L, one vertex is on the origin, and all of the edges are parallel to one axis of $\mathbb{R}^p$. Also we assume that the data points are uniformly sampled from Ω, specifically,

$$x_1, \ldots, x_p \overset{i.i.d}{\sim} \mathcal{U}(0, L) \quad \forall \mathbf{x} \in \Omega. \tag{2.23}$$

We call such an Ω a *uniform straight hypercube*.

Moreover, we consider the anti-isotropic squared exponential function as the choice of the covariance function,

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp\big(- (\mathbf{x} - \mathbf{x}')^T \mathbf{\Gamma} (\mathbf{x} - \mathbf{x}')\big), \tag{2.24}$$

where $\mathbf{\Gamma}$ is a diagonal matrix with length-scale parameters $\gamma_1, \ldots, \gamma_p$ on the diagonal, and without loss of generality, assume $\gamma_1 \leq \ldots \leq \gamma_p$. We note that the squared function of (2.24), i.e., $\mathcal{K}^2(\mathbf{x}, \mathbf{x}')$, is a new squared exponential covariance function with the length scale parameters $2\gamma_1 \leq \ldots \leq 2\gamma_p$. Hence, as $\mathbb{E}_{\Omega_s}\big(\mathcal{K}(\mathbf{x}, \mathbf{x}')\big)$ increases, $\mathbb{E}_{\Omega_s}\big(\mathcal{K}^2(\mathbf{x}, \mathbf{x}')\big)$ increases.

Recall the discussion in Section 2.3 that for computational efficiency, we only consider the subdomains that are separated by parallel hyperplanes. We call these hyperplanes as "cutting hyperplanes", because each of them partitions or "cuts" Ω into two non-overlapping sets on different sides of the hyperplane. Note that creating S subdomains requires $S-1$ cutting hyperplanes. Assuming that the cutting hyperplanes are equidistant (with distant $W = L/S$ from each other), we can characterize the $\ell^{\text{th}} \in [S-1]$ cutting hyperplane on Ω with respect to k^{th} primary axis of $\mathbb{R}^p$ using the vector of angles $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_p\} \backslash \{\theta_k\}$,

$$H_{\boldsymbol{\theta},k,W,\ell} = \{\mathbf{x} \in \Omega \mid x_k - \sum_{j \in [p] \backslash \{k\}} \tan(\theta_j) x_j - \ell W = 0\} \quad \forall \ell \in [S-1]. \tag{2.25}$$

Note that this cutting hyperplane is orthogonal to the axis k only if $\boldsymbol{\theta} = \mathbf{0}$, that is $\theta_j = 0$ for $j \in [p] \backslash \{k\}$.

Denoting, respectively, the hyperplanes containing the "bottom" and the "top" faces of Ω as

$$H_{\boldsymbol{\theta},k,W,0} = \{\mathbf{x} \in \Omega \mid x_k = 0\} \quad \text{and} \quad H_{\boldsymbol{\theta},k,W,S} = \{\mathbf{x} \in \Omega \mid x_k - L = 0\},$$

we define the s^{th} subdomain as the intersection of area between two consecutive hyperplanes and Ω, specifically,

$$\Omega_{\boldsymbol{\theta},k,W,s} = \{\mathbf{x} \in \Omega \mid \min_{\mathbf{x}' \in H_{\boldsymbol{\theta},k,W,s-1}} ||\mathbf{x} - \mathbf{x}'||_2 \leq W \;\; \& \;\; \min_{\mathbf{x}' \in H_{\boldsymbol{\theta},k,W,s}} ||\mathbf{x} - \mathbf{x}'||_2 \leq W\}, \quad (2.26)$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

Theorem 2.2 *Let $\Omega \subset \mathbb{R}^p$ be a uniform straight hypercube with side length L, and let $\mathcal{K}(\cdot,\cdot)$ denote covariance function* (2.24)*. Then, for a fixed $W = L/S$, $s \in [S]$, and $k \in [p]$, $\Omega_{\mathbf{0},k,W,s}$ gives the maximum expected covariance, i.e.,*

$$\arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\Omega_{\boldsymbol{\theta},k,W,s}}\big(\mathcal{K}(\mathbf{x},\mathbf{x}')\big) = \mathbf{0}.$$

While Theorem 2.2 shows that cutting orthogonally to the given axis $k \in [p]$, i.e., $\boldsymbol{\theta} = \mathbf{0}$, maximizes the expected covariance compared to any other $\boldsymbol{\theta} > \mathbf{0}$, Theorem 2.3 further shows that among all the subdomains created by cutting orthogonally to a primary axis, the one created by cutting orthogonally to the axis associated with the fastest direction of change, i.e., the direction associated with the largest γ, has the maximum expected covariance

Theorem 2.3 *Let $\Omega \subset \mathbb{R}^p$ be a uniform straight hypercube with side length L, and let $\mathcal{K}(\cdot,\cdot)$ denote covariance function* (2.24)*. Then for a fixed $W = L/S$ and $s \in [S]$, among all the subdomains $\Omega_{\mathbf{0},k,W,s}$ for $k \in [p]$, $\Omega_{\mathbf{0},p,W,s}$ gives the maximum expected covariance, i.e.,*

$$\arg\max_{k} \mathbb{E}_{\Omega_{\mathbf{0},k,W,s}}\big(\mathcal{K}(\mathbf{x},\mathbf{x}')\big) = p.$$

Theorems (2.1), (2.2), and (2.3) along with the property of covariance function (2.24), i.e., larger values of $\mathbb{E}_{\Omega_s}\big(\mathcal{K}(\mathbf{x},\mathbf{x}')\big)$ imply larger $\mathbb{E}_{\Omega_s}\big(\mathcal{K}^2(\mathbf{x},\mathbf{x}')\big)$, provide a partitioning policy for the domain Ω. That is, cutting orthogonally to the direction of fastest covariance decay reduces the upper bound of expected error (2.22), and therefore, gives a more accurate covariance approximation in each subdomain.

However, we note that the direction of fastest covariance decay may not necessarily be a primary axis of the input domain. To overcome this drawback, we relax the restriction of choosing one of the primary axes as the direction of fastest covariance decay by using a general form of the squared exponential covariance function, $\mathcal{K}(\mathbf{x},\mathbf{x}') = \exp(-(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'))$, where $\mathbf{M}$ is a $p\times p$ positive definite matrix [88]. For the purpose of this discussion, we define $\mathbf{M}$ as $\mathbf{a}\mathbf{a}^T+\gamma\mathbf{I}_p$, where $\mathbf{a}$ is a unit direction vector in the input space with length p, and γ is a joint length-scale parameter, to obtain the following covariance function,

$$\mathcal{K}^{\mathbf{a}}(\mathbf{x},\mathbf{x}') = \exp(-(\mathbf{x}-\mathbf{x}')^T(\mathbf{a}\mathbf{a}^T+\gamma\mathbf{I}_p)(\mathbf{x}-\mathbf{x}')), \tag{2.27}$$

which involves a dot product $(\mathbf{x}-\mathbf{x}')^T\mathbf{a}$. This means that for a given distance $||\mathbf{x}-\mathbf{x}'||_2$, the angle between $\mathbf{x}-\mathbf{x}'$ and $\mathbf{a}$ determines the covariance. In particular, the direction $\mathbf{a}$ itself has the relatively highest rate of covariance decay.

Although in practice, direction $\mathbf{a}$ may not exist, fitting covariance function (2.27) to the data using Maximum Likelihood Estimation can find the best choice of $\mathbf{a}$ under the MLE criterion. Therefore, under the GP assumptions, we maximize the logarithm of the likelihood function to find the optimal value of vector $\mathbf{a}$,

$$\max_{\mathbf{a},\gamma,\sigma^2} -\mathbf{y}^T(\mathbf{K}^{\mathbf{a}}+\sigma^2\mathbf{I})^{-1}\mathbf{y} - \log|\mathbf{K}^{\mathbf{a}}+\sigma^2\mathbf{I}|, \tag{2.28}$$

where $\mathbf{K}^{\mathbf{a}}$ is the covariance matrix formed based on covariance function (2.27).

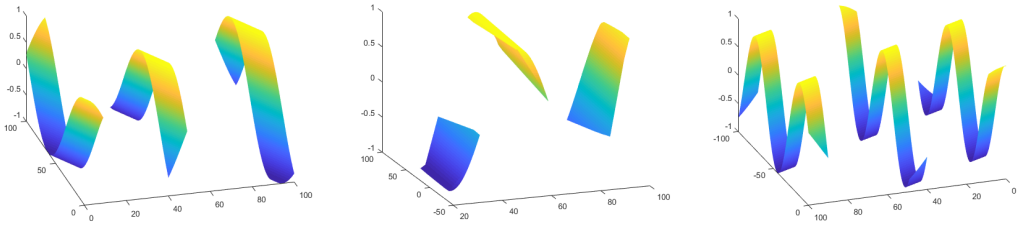
Here, since we only want to find direction $\mathbf{a}$, the nuisance parameters are the variance and the length scale parameters, σ^2 and γ. Therefore, to shrink the parameter space, we set σ^2 and γ to small values after standardizing the data.

Note that optimization problem (2.28) is of $\mathcal{O}(N^3)$, which is the same order of complexity as the original problem. However, since the output of optimization (2.28) is merely used to find a desired direction, and is not used for prediction, we utilize a small subset of data with size $n \ll N$. Further, since $\mathbf{a}$ is a unit direction vector, we write $\mathbf{a} = [\bar{\mathbf{a}}^T, \sqrt{1 - \bar{\mathbf{a}}^T\bar{\mathbf{a}}}]^T$, where $\bar{\mathbf{a}} = [a_1, \ldots, a_{p-1}]^T$, and add the unity constraint, $\bar{\mathbf{a}}^T\bar{\mathbf{a}} \leq 1$, to the optimization problem. Consequently,

$$\begin{aligned} \min_{\bar{\mathbf{a}}} \quad & \mathcal{L}(\bar{\mathbf{a}}) = \mathbf{y}_n^T(\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{y}_n + \log|\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n| \\ \text{subject to} \quad & \bar{\mathbf{a}}^T\bar{\mathbf{a}} \leq 1, \end{aligned} \tag{2.29}$$

where $\mathbf{y}_n$ is the response vector of the small subset of data and $\mathbf{K}_n^{\bar{\mathbf{a}}}$ is the covariance matrix evaluated by covariance function (2.27) on the same small subset (See Appendix C for solving optimization problem (2.29) by using Projected Gradient Descent [66]). Also note that the computational complexity of optimization (2.29) is dominated by matrix inversion $(\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n)^{-1}$, which has the order $\mathcal{O}(n^3)$.

Finally, the intuition behind the optimal direction $\mathbf{a}$ is that it results in subdomains with "simpler" or "smoother" functions which can be more easily approximated. The example in Figure 2.1 shows the 3-D presentations of three local functions created by cutting global function $f(\mathbf{x}) = \cos(0.05x_1 + 0.1x_2)$ orthogonally to each direction $[1, 0]$, $[0.43, 0.9]$, and $[0, 1]$. Direction $[0.43, 0.9]$ is the direction of fastest covariance decay obtained by optimizing (2.29) on a small dataset sampled from the global function. Observe that the local functions created by cutting orthogonally to the optimal solution have smoother structures compared to directions $[1, 0]$ and $[0, 1]$.



(a) $\mathbf{a} = [1, 0]^t$ (b) $\mathbf{a} = [0.43, 0.9]^t$ (c) $\mathbf{a} = [0, 1]^t$

Figure 2.1: Local functions created by cutting orthogonally to directions $[1, 0]$, $[0.43, 0.9]$ (optimal solution of (2.28)), and $[0, 1]$ on a synthetic dataset

2.3.3 Creating boundaries, control points, and boundary functions

The focus of this section is on the practical implementation of SPLGP, and therefore, the characterization of cutting hyperplanes differs from the discussion in Section 2.3.2. Here, instead of using a vector of angles corresponding to primary axes of input space, we use a given direction, which can be the solution to optimization (2.29) or any other arbitrary direction, to define the cutting hyperplanes.

Recall that in our partitioning policy all the cutting hyperplanes are parallel to each other, and therefore, orthogonal to a unique direction, which is characterized by a vector $\mathbf{a} = [a_1, \ldots, a_p]^T$. Let $\mathbf{Z} = \{\mathbf{x}_i^T \mathbf{a} \mid \mathbf{x}_i \in \mathbf{X}\}$ denote the projection of all the input vectors onto $\mathbf{a}$. Next, consider the ordered set $\{z_1, \ldots, z_{S-1}\}$, where $\min \mathbf{Z} < z_1$ and $z_{S-1} < \max \mathbf{Z}$, and $z_\ell < z_{\ell+1}$ for $\ell \in [S-1]$.

Given the set $\{z_1, \ldots, z_{S-1}\}$ and direction $\mathbf{a}$, which is in fact the normal vector of all of the cutting hyperplanes, we define the ℓ^{th} cutting hyperplane orthogonal to $\mathbf{a}$ as $H_{\ell,\mathbf{a}} = \{\mathbf{x} \in \Omega \mid \ a_1 x_1 + \ldots + a_p x_p = z_\ell\}$ for $\ell \in [S-1]$. We use the data points close to $H_{\ell,\mathbf{a}}$ to locate the control points. To this end, we first define $\mathbf{\Delta}_\ell = \{\mathbf{x}_i \in \mathbf{X} | \ |\mathbf{x}_i^T \mathbf{a} - z_\ell| < \delta\}$ as the set of training data points whose Euclidean distance to $H_{\ell,\mathbf{a}}$ is less than a predefined constant δ. Then, calculate the maximum

and minimum of the k^{th} dimension of the data points in $\boldsymbol{\Delta}_\ell$, respectively,

$$\tau_{1,k,\ell} = \max_{\mathbf{x}_i \in \boldsymbol{\Delta}_\ell} \mathbf{x}_i^T \mathbf{e}_k \qquad \text{and} \qquad \tau_{0,k,\ell} = \min_{\mathbf{x}_i \in \boldsymbol{\Delta}_\ell} \mathbf{x}_i^T \mathbf{e}_k, \tag{2.30}$$

where $\mathbf{e}_k$ is the unit vector along the k^{th} primary axis of the space for $k \in [p]$. As such, the set $\mathbf{V}_\ell = \left\{ [\tau_{b,1,\ell}, \ldots, \tau_{b,p,\ell}]^T \,|b = 0, 1 \right\}$ characterizes the vertices of the hyper-rectangle inscribing $\boldsymbol{\Delta}_\ell$. Next, we uniformly sample $Q > 0$ points from $\mathbf{V}_\ell$ and denote the set of all these points as $\mathbf{U}_\ell$. We obtain the set of control points on $H_{\ell,\mathbf{a}}$ denoted as $\mathbf{C}_\ell$ by projecting the points in $\mathbf{U}_\ell$ on $\mathcal{H}_{\ell,\mathbf{a}}$,

$$\mathbf{C}_\ell = \{(z_\ell - \mathbf{u}^T\mathbf{a})\mathbf{a} + \mathbf{u} \mid \forall \mathbf{u} \in \mathbf{U}_\ell\}. \tag{2.31}$$

There are several ways to choose the width of each subdomain, i.e., $z_{\ell+1} - z_\ell$ for $\ell \in [S-1]$. One way is to choose a fixed width for the subdomains; however, this approach results in subdomains with different numbers of local data points depending on their distribution on the domain. Also *adaptive mesh generation* techniques [6] can be used to vary the widths to balance the error among the subdomains. In Section 2.4, we use varying widths for the subdomains to balance the numbers of local data points across the subdomains. This approach helps us to control the computation time of the model, because it is evenly distributed among the subdomains.

Furthermore, to impose connectivity on the optimization procedure discussed in Section 2.3.1, we need to specify the boundary values for each control point $\mathbf{c} \in \mathbf{C}_\ell$. To this end, we fit a boundary GPR over the hyper-rectangle defined by $\mathbf{V}_\ell$ using the data points in $\boldsymbol{\Delta}_\ell$. We then use the predictive mean function of this GPR to determine the boundary values. Letting $\mathcal{R}_\ell(.)$ denote as the predictive mean function of the GPR constructed by $\boldsymbol{\Delta}_\ell$, the boundary value for each $\mathbf{c} \in \mathbf{C}_\ell$ is

$$\mathcal{R}_\ell(\mathbf{c}) = \mathbf{k}_{\mathbf{c}\boldsymbol{\Delta}_\ell}(\mathbf{K}_{\boldsymbol{\Delta}_\ell\boldsymbol{\Delta}_\ell} + \sigma_\ell^2 \mathbf{I}_\ell)^{-1} \mathbf{y}_{\boldsymbol{\Delta}_\ell}, \tag{2.32}$$

where $\mathbf{k}_{\mathbf{c}\boldsymbol{\Delta}_\ell}$ is the covariance vector between the control point $\mathbf{c} \in \mathbf{C}_\ell$ and the neighboring data points in $\boldsymbol{\Delta}_\ell$, and $\mathbf{K}_{\boldsymbol{\Delta}_\ell\boldsymbol{\Delta}_\ell}$ is the covariance matrix between the neighboring

data points in $\mathbf{\Delta}_\ell$ themselves. In Section 2.3.1, with a slight abuse of notation, we denote $\mathcal{R}(.)$ as a function that takes a control point as an input and returns $\mathcal{R}_\ell(.)$, depending on the location of the control point.

Since the set of neighboring data points $\mathbf{\Delta}_\ell$ is a small set, we use a full GPR to obtain functions (2.32). This results in $\mathcal{O}(S\Delta^3)$ operations in training the boundary functions, where Δ is the average size of all $\mathbf{\Delta}_\ell$. Moreover, due to our partitioning policy each subdomain has two boundaries at most. Therefore, the number of control points for each subdomain is twice the values of Q on average, i.e., $|\mathbf{B}| = 2Q$. Hence, we update the computational complexity derived in Section 2.3.1 as $\mathcal{O}(SQ^3 + Nm^2)$. Considering the complexity of subdomain selection discussed in Section 2.3.2, we calculate the total computational complexity of SPLGP as $\mathcal{O}(Nm^2+S(\Delta^3+Q^3)+n^3)$. Note that this complexity is dominated by $\mathcal{O}(Nm^2)$ for moderate dimensional spaces.

2.3.4 Hyper-parameter learning

In SPLGP, instead of one global marginal likelihood function (1.8), there are S local functions, each of which can be trained independently. Recall that our local predictors are in fact SPGP predictors that consider pseudo-inputs as parameters of the model. Therefore, we have two types of parameters: one is the location of local pseudo-inputs and the other is the hyper-parameters of the underlying covariance function. Maximizing the logarithm of the local SPGP marginal likelihood functions using gradient descent with respect to local pseudo-inputs and hyper-parameters provides local optimal locations. Specifically, the logarithm of the marginal likelihood of SPLGP's s^{th} local model is

$$\log\big(p(\mathbf{y}_s)\big) = -\frac{1}{2}\log|\mathbf{G}_s| - \frac{1}{2}\mathbf{y}_s^T\mathbf{G}_s^{-1}\mathbf{y}_s - \frac{n_s}{2}\log 2\pi, \tag{2.33}$$

where $\mathbf{G}_s$ is the same as that of Section 2.3.1. [99] shows that the cost of finding the derivatives and maximizing (2.33) is in the order of $\mathcal{O}(n_s m_s^2)$ for not very high

dimensional problems, and therefore, the total complexity of training the subdomains becomes $\mathcal{O}(Nm^2)$.

2.4 Experimental results

In this section, we apply SPLGP to four real datasets and compare its performance with local probabilistic regression (LPR) [107] and Bayesian committee machine (BCM) [104], bagged Gaussian process (BGP) [13], partial independent conditional GP (PIC) [100], and DDM [71]. LPR and BCM are local GPR models, BGP is a fast bootstrapping model, and PIC is a localized version of SPGP. We use the BGP, LPR, BCM, and DDM implementations in the GPLP toolbox [71] and use the BCM implementation provided by [96]. We also conduct sensitivity analysis of the parameters in SPLGP and propose some guidelines for their selection.

2.4.1 Datasets and evaluation criteria

We implement SPLGP in MATLAB and test it on four real datasets:

1. The spatial dataset, TCO, which contains 65000 observations, collected by the NIMBUS7 satellite for NASA's Total Ozone Mapping Spectrometer (TOMS) project (`https://www.nodc.noaa.gov`). The global measurement was conducted on a two dimensional grid, i.e., latitude and longitude, from 1978 to 2003 on a daily basis. We select the measurements of "total column of ozone" on this grid for the data collected on January 1, 2003. The dataset is highly non-stationary and an appropriate dataset for comparing SPLGP and DDM because it is constructed on a two-dimensional input space,

2. The spatial dataset, Levitus, which contains 56000 observations, is a part of the world ocean atlas that measures the annual means of major ocean parameters (`http://iridl.ldeo.columbia.edu/SOURCES/.LEVITUS94`). The global mea-

surement was conducted on a three dimensional grid, i.e., latitude, longitude, and depth, in 1994. We select the "apparent oxygen utilization" as the response variable on this grid.

Recalling that handling exogenous variables in spatial datasets also motivates this study, we use a third real dataset.

3. The spatial dataset, Dasilva, which contains 70000 observations, is a part of a five-volume atlas series of Surface Marine Data (`http://iridl.ldeo.columbia.edu/SOURCES/.DASILVA/.SMD94/.halfbyhalf/.climatology/`). The global measurement was conducted on a two dimensional grid, i.e., latitude and longitude, on a monthly basis in 1994. We select three exogenous variables, "constrained outgoing heat flux", "zonal heat flux", and "sea minus air temperature", and the objective is to predict "long wave Chi sensitivity" based on the data collected on January 1994.

Although SPLGP was developed to handle spatial datasets, the model is general and can be efficiently applied to non-spatial data that have a moderate number (say ten or fewer) of exogenous variables. We use a forth dataset to demonstrate the performance of SPLGP on non-spatial data.

4. The non-spatial dataset, Protein, which contains 46000 measurements, is a collection of Physicochemical Properties of Protein Tertiary Structure (`http://archive.ics.uci.edu/ml/datasets`). This dataset contains nine "physicochemical properties" of proteins as explanatory variables and "size of the residue" as the response variable.

We randomly partition each dataset into 90% for training and 10% for testing. We use three measures to evaluate the performance of each method. The first one is the measure of prediction accuracy, which is assessed by the Mean Squared Error

(MSE),

$$\text{MSE} = \frac{1}{c}\sum_{i=1}^{c}(y_*^i - \hat{\mu}_*^i)^2, \tag{2.34}$$

where y_*^i is the noisy observation of the test location $\mathbf{x}_*^i$ and $\hat{\mu}_*^i$ is the mean prediction of this test location. The second measure is the Negative Log Predictive Density (NLPD) that takes into account uncertainty in prediction in addition to accuracy, specifically

$$\text{NLPD} = \frac{1}{c}\sum_{i=1}^{c}\frac{(y_*^i - \hat{\mu}_*^i)^2}{2\hat{\sigma}_*^{i^2}} + 0.5\log(2\pi\hat{\sigma}_*^{i^2}), \tag{2.35}$$

where $\hat{\sigma}_*^{i^2}$ is variance of the predictor at the test location $\mathbf{x}_*^i$. The third measure is the computation time, i.e., training plus testing time, that evaluates the SPLGP's success in speeding up GPR. Note that the computation time is not an appropriate measure on its own and the corresponding MSE or NLPD should be taken into account simultaneously, because a reduction in training time without an accurate prediction is not useful.

2.4.2 Computation time and prediction accuracy

Here, we compare the computation time and the prediction accuracy of SPLGP with those of the competing models. Specifically, we consider the MSE and NLPD as functions of the computation time and plot the set of best results for each model. Under this criterion, the model associated with the curve closest to the origin will be superior. The parameter selection for each model is as follows.

The tuning parameters in SPLGP are Q, the number of control points on each boundary; S, the number of subdomains; and m, the number of pseudo-inputs in each subdomain. We set Q proportional to the dimension of the boundary,

$$Q = q^{p-1}, \tag{2.36}$$

where p is the dimension of the domain of data, and q determines the density of control points on each boundary space. Our experiments in Section 2.4.3 suggest that setting q to small values results in a satisfactory performance and increasing it does not significantly affect the model's accuracy. Therefore, we set $q = 3$, for datasets TCO, Levitus, and Dasilva, and $q = 2.2$ for dataset Protein. We also set m proportional to the average number of local data points,

$$m = \kappa\sqrt{\frac{N}{S}}, \tag{2.37}$$

where κ is a parameter that determines the density of pseudo-inputs in each subdomain. Moreover, we choose the direction of one of the primary axes of the datasets' domains as the direction of the cuts (see Section 2.4.3 for a discussion of cuts in other directions). We set κ to a fixed value and choose the size of the subdomains from the set $\{20, 30, 40, 50, 60\}$, except for the dataset Levitus. For the latter, since we cut the domain of data from the third direction with 33 distinct levels, we choose S from the set $\{8, 11, 16, 33\}$. Note that as S increases, computation time decreases, so the points with smaller computation times belong to larger values of S in Figures 2.2 and 2.3.

The tuning parameters for DDM are Q, the number of control points on each boundary; and S, the number of subdomains. For the two-dimensional dataset TCO, we set $Q = 3$ and choose the values of S from the set $\{100, 200, 300, 400, 500, 600\}$ to keep the average size of the subdomains between 100 to 600 as instructed in [70]. As expected, for smaller values of S, i.e., larger subdomains, the efficiency of the model deteriorates in terms of computation time; therefore, the points with higher computation times belong to smaller values of S in Figures 2.2 and 2.3.

PIC, which is the localized version of SPGP, has two tuning parameters, the number of local models, S, and the number of pseudo-inputs, m. We use k-means clustering to partition the domain of data into S local models. We note that m is the major tuning parameter that affects the model's computation time. Therefore, we fix

S to a reasonable value and choose the values of m from the set $\{100, 200, 300, 400, 500, 600\}$. After testing various values of S in the range of 100 to 800, we find that $S = 500$ is a reasonable choice for our experiments. Therefore, we set $S = 500$ for all the four datasets. In Figures 2.2 and 2.3, those points with higher computation times belong to larger values of m.

For BCM, we use k-means clustering to partition the domain of data into S local experts similar to PIC and choose the values of S from the set $\{200, 300, 400, 500, 600, 700\}$. The points with higher computation times belong to larger values of S in Figures 2.2 and 2.3.

LPR has three major tuning parameters, which are S, the number of local experts; m, the size of each local expert; and R, the size of the subset used for local hyper-parameter learning. The location of R data points used for local hyper-parameter learning can be chosen randomly or by clustering; however, for the sake of fair comparison, we use clustering to choose these locations. Moreover, we choose the values of S, m, and R from the sets $\{5, 10, 15, 20\}$, $\{100, 200, 300\}$, and $\{500, 1000, 1500\}$, respectively. For each dataset, we fix S to a value that results in better performance in terms of computation time and MSE, and choose five combinations out of the nine possible combinations of m and R that have different computation times.

Last, BGP has two tuning parameters, the number of bags, S, and the number of data points assigned to each bag, m. Based on our experiments, m is the major tuning parameter affecting the model's computation time; therefore, we vary the values of m from the set $\{500, 600, 700, 800, 900\}$ and fix the value of S to a reasonable number. After varying the values of S in range 10 to 80, we chose 40 as the fixed value of S. In Figures 2.2 and 2.3, those data points with higher computation times belong to larger size bags.

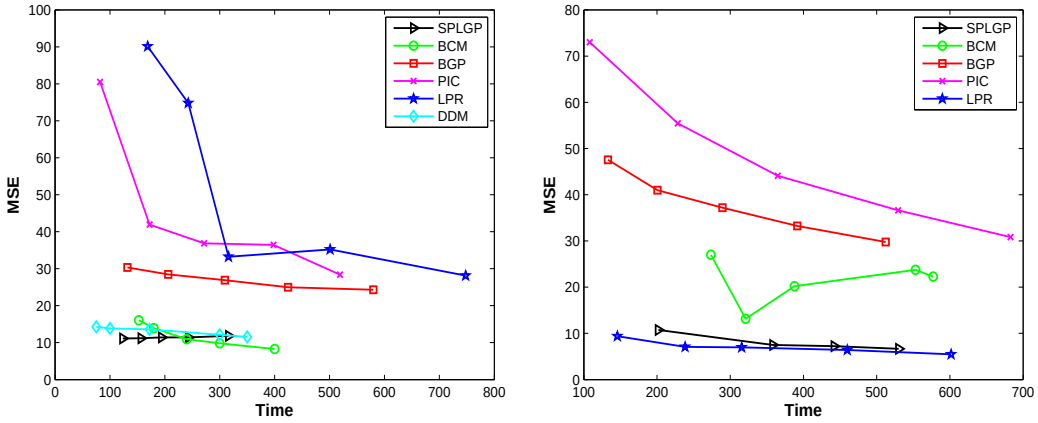
For two-dimensional dataset TCO, SPLGP, DDM, and BCM perform almost the same but they are faster and more accurate than the other models as shown in

Figure 2.2a. However, in terms of NLPD, SPLGP and DDM perform better than BCM as shown in Figure 2.3a. We attribute the BCM's higher NLPD values to underestimating the predictive variance in the BCM model. Also, despite the fact that SPLGP uses a low-rank covariance approximation, it outperforms DDM, mainly because it creates fewer boundaries thus compensating for the inaccuracy of the low-rank approximations in the subdomains. Note that for the other datasets, we cannot compare the performance of DDM with the other competing models, because DDM's implementation is restricted to one- or two-dimensional spaces.

For three-dimensional dataset Levitus, LPR and SPLGP outperform the other models in terms of MSE as shown in Figure 2.2b. However, in terms of NLPD, SPLGP's performance is superior (Figure 2.3b) meaning that SPLGP obtains a better goodness of fit compared to LPR.

A significantly better performance of SPLGP can be observed for five-dimensional dataset Dasilva. For this dataset, SPLGP reaches much lower values of MSE while keeping the values of NLPD low compared to all the other competing models as shown in Figures 2.2c and 2.3c.

Finally, for nine-dimensional dataset Protein, PIC and SPLGP perform much better than the other models as shown in Figures 2.2d and 2.3d. However, similar to the analysis of TCO and Levitus, the lower NLPD values of SPLGP makes our model more desirable than PIC. We note that unlike the other datasets in this study, we do not set the density parameter κ to 3, since 3^8 control points on each boundary slow down the SPLGP's performance without having a significant effect on accuracy (see Section 2.4.3). Therefore, we set q to a smaller value of 2.2.



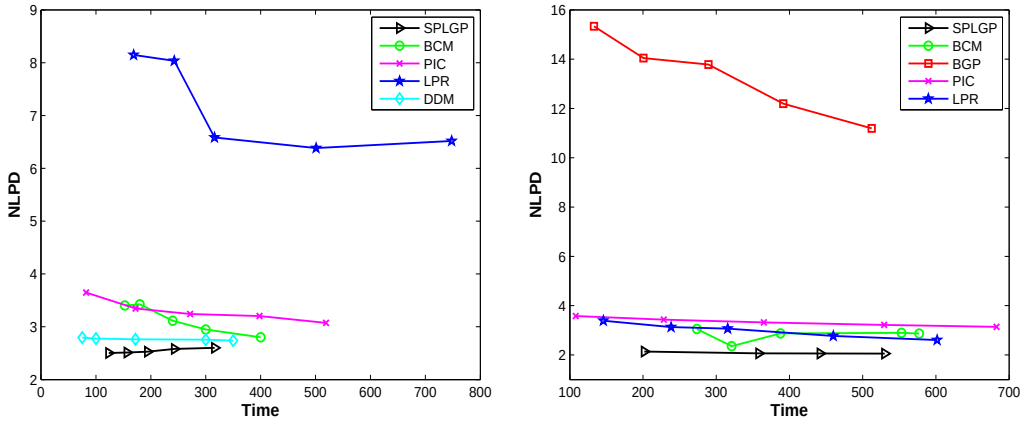
(a) TCO:$\kappa = 6, S \in \{20, 30, 40, 50, 60\}, q = 3$

(b) Levitus:$\kappa = 8, N \in \{8, 11, 16, 33\}, q = 3$

(c) Dasilva:$k = 8, N \in \{20, 30, 40, 50, 60\}, q = 3$

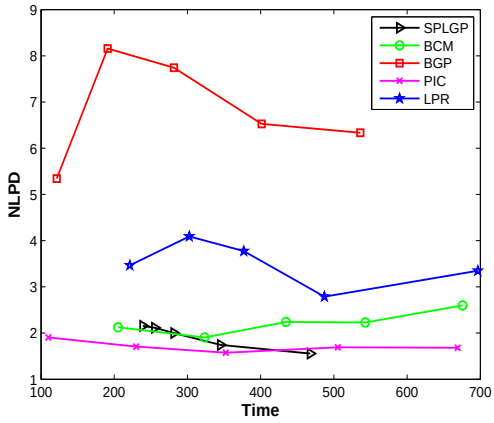
(d) Protein:$k = 8, N \in \{20, 30, 40, 50, 60\}, q = 2.2$

Figure 2.2: MSE versus computation time. For DDM, $Q = 3$ and $S \in \{100, 200, 300, 400, 500\}$; for PIC, $S = 500$ and $m \in \{100, 200, 300, 400, 500, 600\}$; for BCM, $S \in \{200, 300, 400, 500, 600, 700\}$; for LPR, $(S, m, R) \in \{5, 10, 15, 20\} \otimes \{100, 200, 300\} \otimes \{500, 1000, 1500\}$; and for BGP, $S = 40$ and $m \in \{500, 600, 700, 800, 900\}$

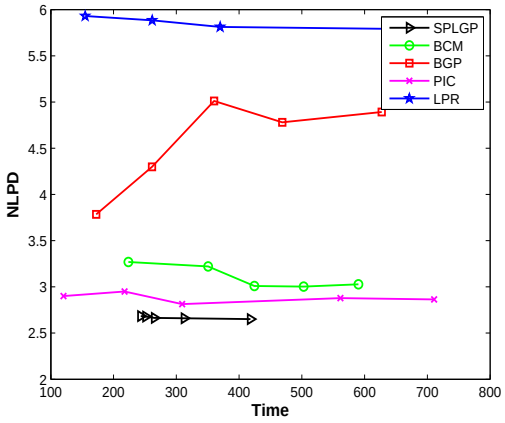


(a) TCO:$\kappa = 6, S \in \{20, 30, 40, 50, 60\}, q = 3$

(b) Levitus:$\kappa = 8, N \in \{8, 11, 16, 33\}, q = 3$



(c) Dasilva:$k = 8, N \in \{20, 30, 40, 50, 60\}, q = 3$



(d) Protein:$k = 8, N \in \{20, 30, 40, 50, 60\}, q = 2.2$

Figure 2.3: NLPD versus computation time. For DDM, $Q = 3$ and $S \in \{100, 200, 300, 400, 500\}$; for PIC, $S = 500$ and $m \in \{100, 200, 300, 400, 500, 600\}$; for BCM, $S \in \{200, 300, 400, 500, 600, 700\}$; for LPR, $(S, m, R) \in \{5, 10, 15, 20\} \otimes \{100, 200, 300\} \otimes \{500, 1000, 1500\}$; and for BGP, $S = 40$ and $m \in \{500, 600, 700, 800, 900\}$

2.4.3 Sensitivity analysis

This section describes the sensitivity analysis we conduct on the tuning parameters of SPLGP. Section 2.4.3 discusses some guidelines for selecting the size of the subdomains and the density of local pseudo-inputs. Section 2.4.3 explains the significance

of cutting from various directions. Finally, Section 2.4.3 shows how the choice of the number of the control points affects the model's performance.

Number of cuts and local pseudo-inputs

A trade-off exists between accuracy and computation time. For SPLGP, having a larger number of subdomains reduces computation time, but the local approximation in each subdomain can be less accurate. This section provides some guidelines for selecting the parameters of SPLGP that help to balance computation time and accuracy.

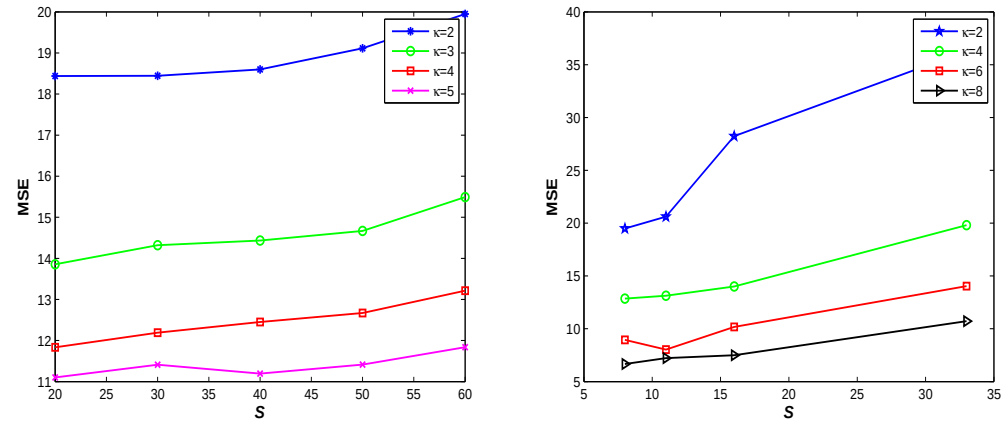
In our experiment, for each dataset, we vary the number of subdomains, S, and the density of local pseudo-inputs, κ, and use the values of MSE, NLPD, and computation time as the measures of efficiency. To illustrate the effect of various settings on the model's efficiency, we plot the values of MSE, NLPD, and computation time for varying S and a fixed κ as shown by the curves in Figures 2.4 and 2.5.

Figures 2.4e, 2.4f, 2.5e, and 2.5f show that as S increases, i.e., the size of the subdoamins decreases, SPLGP performs faster for a fixed value of κ. Moreover, the curves belonging to smaller values of κ are always below the curves with larger values of κ, meaning that as the density of the pseudo-inputs increases, the model becomes slower. Consequently, the model takes longer to run by increasing the size of subdomains or the number of local pseudo-inputs.

On the other hand, Figures 2.4a, 2.4b, 2.5a, and 2.5b show a positive correlation between S and MSE, i.e., by fixing the value of κ, SPLGP performs more accurately in terms of MSE, as the size of the subdomains increases. Moreover, the curves belonging to larger values of κ are always above the curves with lower values of κ, i.e., as κ increases, SPLGP becomes more accurate for a fixed value of S. Figures 2.4c, 2.4d, 2.5c, and 2.5d show the same trend for the values of NLPD. Therefore, we conclude that our model attains higher accuracy in terms of MSE and NLPD by increasing the

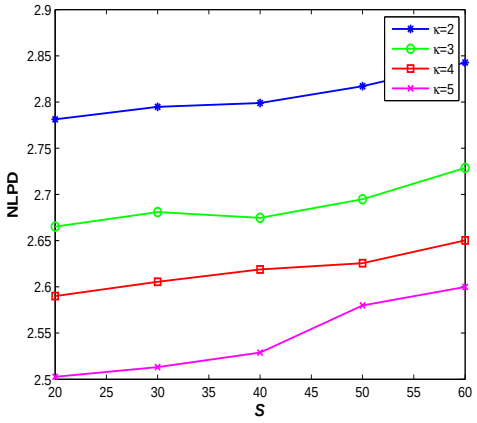
density of local pseudo-inputs or enlarging the size of the subdomains.

In summary, by increasing the size of the subdomains or the density of local pseudo-inputs, the models accuracy improves, but computation time increases. Therefore, we suggest using sufficiently large values of κ in smaller subdomains, because, as shown in Figures 2.4 and 2.5, the MSEs are small even with a large number of subdomains and computation times stay relatively low.

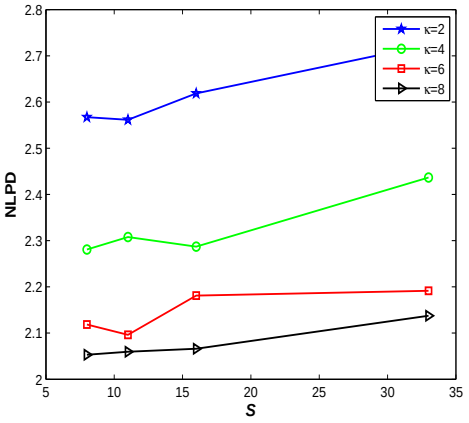


(a) TCO (MSE)

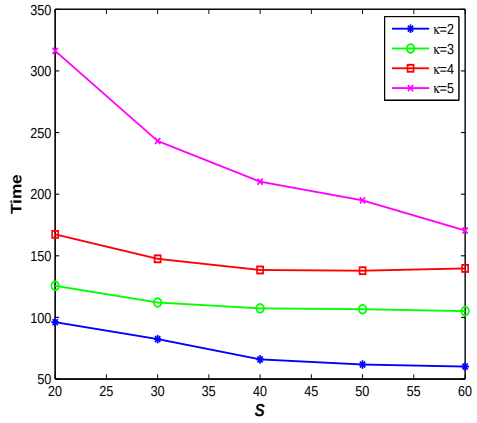
(b) Levitus (MSE)



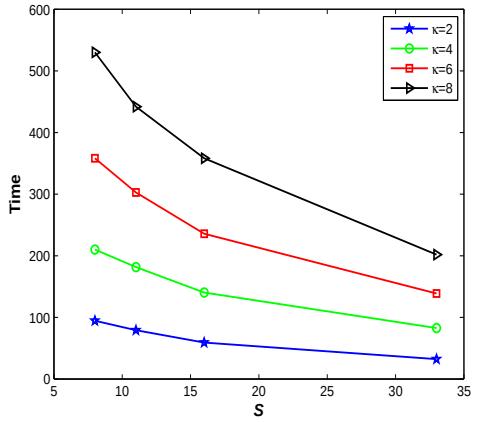
(c) TCO (NLPD)



(d) Levitus (NLPD)

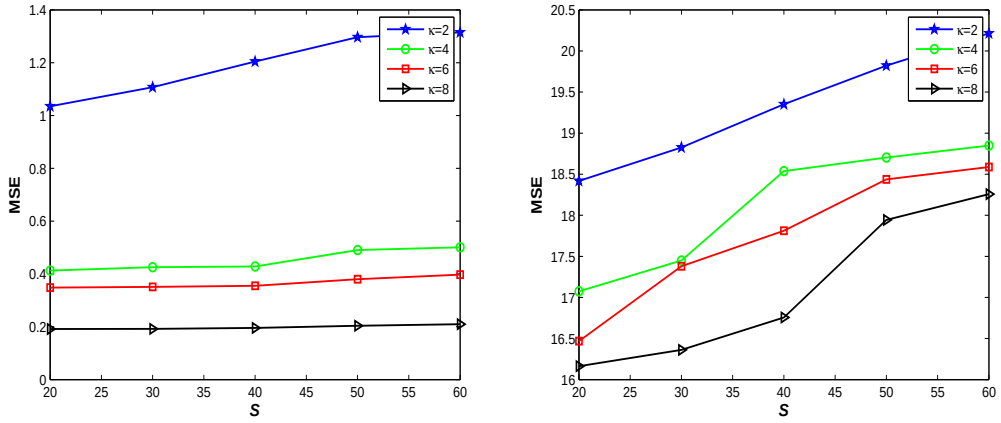


(e) TCO (Time)



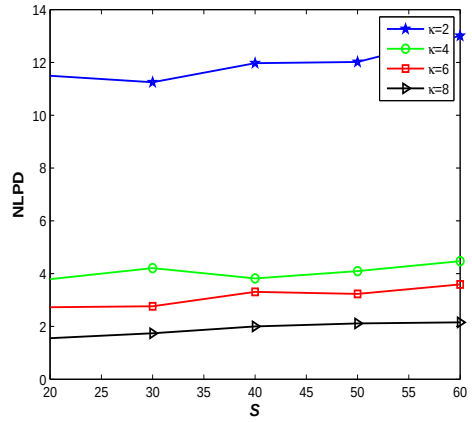
(f) Levitus (Time)

Figure 2.4: MSE, NLPD, and computation time versus S. Each curve associates with a particular value of κ

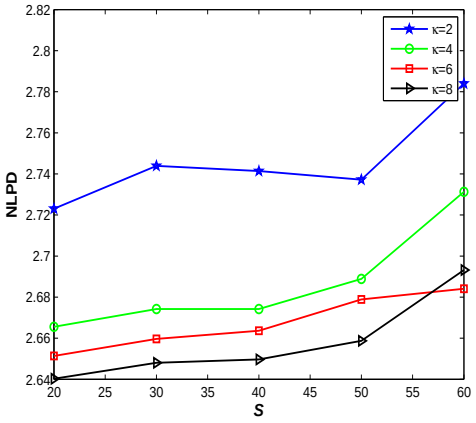


(a) Dasilva (MSE)

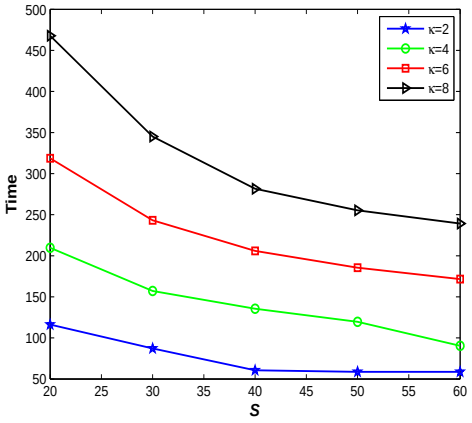
(b) Protein (MSE)



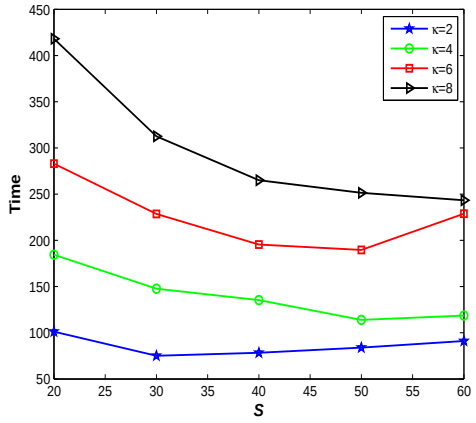
(c) Dasilva (NLPD)



(d) Protein (NLPD)



(e) Dasilva (Time)



(f) Protein(Time)

Figure 2.5: MSE, NLPD, and computation time versus S. Each curve associates with a particular value of κ

Direction of cuts

This section demonstrates how cutting from different directions affects SPLGP's performance. To discuss the significance of cutting from the optimal direction obtained from optimization (2.29), we fix the value of S and vary the values of κ and the direction of cuts for each dataset, and measure the accuracy of prediction in terms of MSE. In Figure 2.6, each curve shows the trend of changes in MSE for a particular direction and the varying values of κ.

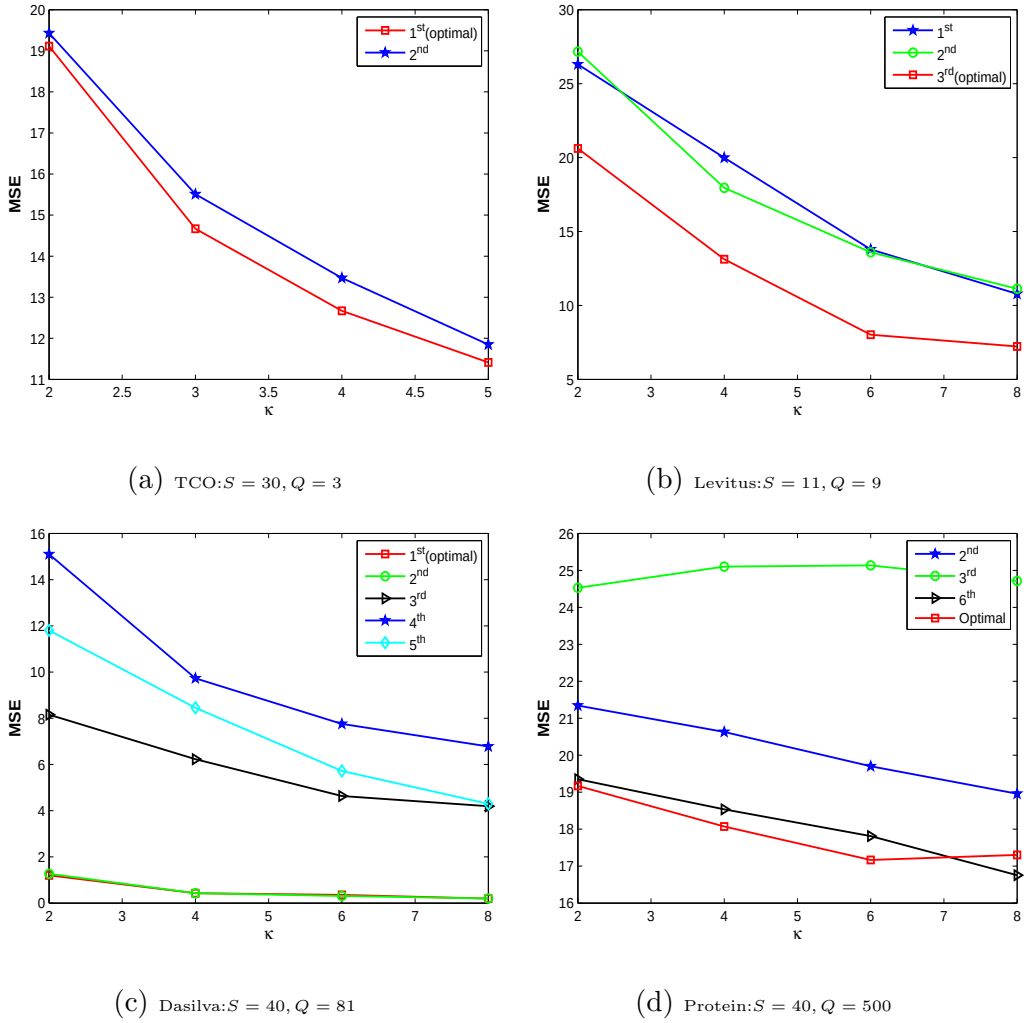
For dataset TCO, the optimal direction is the direction of the first primary axis as shown in Figure 2.6a. Cutting from this optimal direction attains higher accuracy for the varying values of κ compared to the direction of the second primary axis.

For dataset Levitus, the optimal direction is the direction of the third primary axis as shown in Figure 2.6b. Cutting from this optimal direction attains higher accuracy compared to the directions of the other primary axes.

For dataset Dasilva, the optimal direction is the direction of the first primary axis as shown in Figure 2.6c. Cutting from this direction attains a much higher accuracy compared to the directions of the third, fourth, and fifth primary axes. However, the performance of cutting from the direction of the second primary axis is almost the same as the optimal direction. This can be justified by considering the objective values of optimization (2.29) for these two directions. In fact, the objective values for the directions of the first and the second primary axes are very close and much smaller than the other directions. Therefore, we observe such a similar and much more accurate performance by cutting from these two directions compared to the other directions.

Finally, for dataset Protein, the optimal direction, which is not the direction of one of the primary axes of the input domain, is compared with the directions of the second, third, and sixth primary axes as shown in Figure 2.6d. Cutting from the optimal direction attains a much higher accuracy compared to the directions of the

second and the third primary axes, and slightly better than the direction of the sixth primary axis.



(a) TCO:$S = 30, Q = 3$ (b) Levitus:$S = 11, Q = 9$

(c) Dasilva:$S = 40, Q = 81$ (d) Protein:$S = 40, Q = 500$

Figure 2.6: Effects of cutting directions on MSE for the four datasets

Control points density

The last parameter needing to be tuned in SPLGP is the number of control points uniformly distributed on the boundaries of the subdomains. As the dimension of the domain of data increases, we need to locate more control points to efficiently handle the boundary conditions. As discussed in Section 2.4.2, we use a density parameter and the dimension of the boundary space, i.e., q and $p-1$ in (2.36), to determine the

Dataset	q	Time	MSE	NLPD
TCO	3	145.50	12.18	2.61
	4	145.61	12.15	2.61
	5	146.06	11.98	2.60
Levitus	3	134.48	25.50	2.60
	4	134.47	25.44	2.59
	5	135.36	25.25	2.59
Dasilva	3	157.62	0.42	4.01
	4	159.98	0.38	3.30
	5	167.79	0.38	3.05
Protein	2.2	147.53	17.41	2.67
	2.5	202.09	17.39	2.66
	3	3651.33	17.38	2.65

Table 2.1: Effect of q on efficiency of SPLGP. $S = 30$ and $\kappa = 4$ across all the datasets

number of control points to be uniformly located on each boundary.

Notably, our experiments show that setting q to small values usually results in satisfactory performance, while increasing it does not significantly affect the prediction accuracy, but increases the computational burden, particularly in higher dimensional domains. The results of testing SPLGP on our four datasets with varying values of q and all other parameters fixed are reported in Table 2.1. An increase in the value of q slightly improves the prediction accuracy in terms of NLPD and MSE. Moreover, as the dimension of the domain of data increases, an increase in the value of q results in much longer computation time.

2.5 Summary

We proposed Sparse Pseudo-input Local Gaussian Process (SPLGP) that simultaneously addressed scalability and heterogeneity problems of GPR by partitioning the data's domain into subdomains. We used parallel hyperplanes in our partitioning to

create non-overlapping subdomains and fitted a sparse GPR to the data within each subdomain. This allowed us to select subdomains having large numbers of data points and as a result reduce the number of the boundaries. We addressed the problem of discontinuity of the overall prediction surface by imposing connectivity constraints through putting a few control points on the boundaries of the neighboring subdomains. The parallel cuts, the sparse approximation scheme, and the procedure of handling boundary conditions allowed us to apply SPLGP to spatial datasets with higher dimensions, i.e., spatial data that included exogenous variables in addition to the location information. We proposed three theorems, which together characterized the optimal direction of parallel hyperplanes in the sense that they resulted in more accurate sparse approximation in each subdomain. We also developed an algorithm to find these optimal hyperplanes. Our numerical experiments demonstrated that SPLGP outperformed, or was comparable to, the competing models commonly applied to spatial datasets.

CHAPTER 3

A BAYESIAN FRAMEWORK FOR LOCAL CALIBRATION OF EXPENSIVE COMPUTATIONAL MODELS WITH A NON-ISOMETRIC CURVE TO SURFACE MATCHING INTERPRETATION

3.1 Introduction

Experimenting on computational models instead of actual physical systems has become a popular practice ever since computers have become advanced enough to handle complex mathematical models and intense computational procedures [22, 92]. This popularity is due to the fact that a computational model can obtain the output of an experiment in a cost-effective and timely manner in comparison to its associated physical system. However, one challenge in utilizing computer models is their "adjustment." In fact, computational models usually incorporate features which cannot be observed or measured in physical systems, but they need to be correctly specified, so that the computational model can faithfully resemble the physical system [48]. We refer to these unobservable/unmeasurable features as *calibration variables*, and adjusting their values as *calibration procedure*. In addition, we call the input features which are common between the computational models and the physical systems as *control variables*.

For example, in buckypaper fabrication requiring Poly-vinyl Alcohol (PVA) treatment, we are interested in understanding the relation between the response, the tensile strength, and the control variable, the PVA amount [81]. Here, the calibration variable is the percentage of PVA absorbed, which cannot be measured in the physical

system, but is required in the computational model.

Past studies on the calibration problem generally assume unique values for calibration variables, some times referred to as *global calibration*, and use different statistical frameworks to estimate these values. For instance, authors of [48, 16, 89, 42, 41, 113, 4, 32] have devised various Bayesian models. Whereas authors of [55, 85] have used Maximum Likelihood estimation, while [45, 38] have developed mixed models by combining frequentist and Bayesian methodologies. More recently authors of [105, 106] have developed models based on L_2 distance projection to estimate true values of global calibration variables.

By contrast, there are a few studies that assume the values of calibration variables to depend on control variables, which we refer to as *local calibration*. [81] have shown that an approach that considers a parametric functional relationship between the amount of PVA and the percentage absorbed can outperform the global calibration approach presented in [48]. Similarly, authors of [116] have used a simple linear relationship to improve the calibration accuracy in a thermal challenge problem. Local calibration approaches that are more general use non-parametric methods to build functional relationships between the calibration and the control variables. These non-parametric functional relationships have been constructed using Reproducing Kernel Hilbert Spaces [93] and GPs by various authors [84, 74, 10].

All the aforementioned studies in local calibration and most studies in global calibration require computational models that are "cheaply executable." This assumption is required since computational models need to be evaluated thousands of times either to draw samples from posterior distributions in Bayesian approaches, or to numerically optimize a loss or a likelihood function in other approaches. A common way to deal with expensive computational models is to initially draw a limited number of random samples from the computational model, then fit a surrogate function based on these random samples and replace it with the computational model in

the calibration procedure. However, surrogate modeling itself creates an additional source of error due to restrictions in drawing enough number of random samples from the expensive computational model that results in deterioration of the calibration accuracy.

In this study, we develop a new framework for the local calibration of expensive computational models. Unlike conventional surrogate modeling that replaces the computational model with a static, approximated surface, we employ a "dynamic" GP distribution over the computational model. We call our GP distribution dynamic as the hyper-parameters of the GP's covariance function are trained during our calibration procedure. In our calibration procedure, we use Bayesian statistics to simultaneously construct posterior distributions for the hyper-parameters of the GP's covariance function and the calibration variables associated with each of the physical control vectors.

In fact, the focus of local calibration approaches available in the literature [10, 84] are on the accurate estimation of the functional relationship between the control and the calibration variables. Accurate estimation however, requires the evaluation of the computational models a large number of times, which may not be feasible for expensive computational models. This results in a poor retrieval of the relationship between the control and the calibration variables. Therefore, in our approach, we compensate this inaccurate retrieval by allowing the GP to tune its hyper-parameters along with the calibration variables in a way that the computational model responses become as close as possible to the physical responses.

We note that our Bayesian model suffers from unidentifiablity due to high dimensionality of parameter space, therefore, we need to use informative prior distributions to make the model identifiable. To obtain informative prior distributions we take advantage of an alternate *geometric interpretation* of calibration as non-isometric curve to surface matching. We explain this in the case of a single control variable and a

single calibration variable. From a geometric perspective, all the possible values for the control variable and the physical response constitute a plane curve in the control-response space. By contrast, in the computational model, we can specify the values of both the control and the calibration variables. Consequently, all the possible values of the control and calibration variables, and the computational response together form a surface. The plane physical curve we observe in the control-response space is a projection of a space curve in the three-dimensional control-calibration-response space (see Figure 3.1).

The explanation lies in the nature of calibration variable in a physical process: for each value of control variable there exists an (unknown) value for the calibration variable, and these two features determine one single response. Since we do not observe the actual value of the calibration variable in the physical process, we only see a projected curve in control-response space. Hence, calibration aims to recover the true physical curve, or in other words, determine a non-isometric match of a curve to a surface.

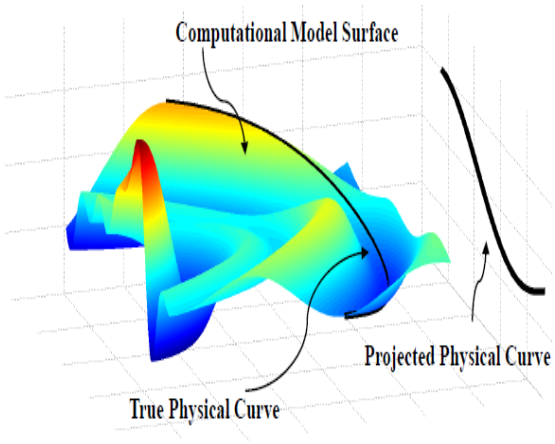


Figure 3.1: Non-isometric curve to surface matching perspective of local calibration. The computational model responses correspond to a two-dimensional surface. The observed physical curve is the projection of the true physical curve that lies on this surface.

The remainder of this chapter is organized as follows: In Section 3.2, we explain our Bayesian model for handling expensive computational models. In Section 3.3, we formally describe how the calibration problem can be interpreted as a non-isometric curve to surface matching problem. In the same section, we show how we can utilize this geometric perspective to construct informative prior distributions for our Bayesian model using a graph-theoretic approach. We generalize the idea of non-isometric curve to surface matching to higher dimensions in Section 3.4 and introduce integer programming techniques to tackle the problem. In Section 3.5, we use the solution approaches presented in Sections 3.3 and 3.4 to construct informative prior distributions for our Bayesian model. In Section 3.6, we show how we draw samples from the posterior distribution using Markov Chain Monte Carlo technique [28], and we explain the prediction procedure in Section 3.7. Finally, we present our experimental and comparative results in Section 3.8.

3.2 General Setting: A Bayesian model for the calibration of expensive computational models

Consider a physical system that operates according to a set of (possibly unknown) physical laws. In this system, there is a functional relationship between a group of features and the response (output). We call those features of the system which can be measured and specified as inputs of the physical system as *control variables*, and denote by $\mathbf{x} \in \mathbb{R}^{d^x}$, a vector containing specified values of these variables. We also assume that once the control variables are set (either observed or specified) in the physical system, the physical process $\mathcal{F}^p$ generates a real-valued response y^p. That is

$$y^p = \mathcal{F}^p(\mathbf{x}).$$

Although the response is a function of *all* the features of the physical system, we write y^p explicitly as a function of $\mathbf{x}$ as the rest of the features are hard to measure

or control, and hence we have no control over them in the physical system. We call such features, *calibration variables*, and categorize them into the following two groups: (i) *global calibration variables*, which have unique values regardless of the values of control variables, and (ii) *local calibration variables* which are functions of control variables.

We denote the vector of global calibration variables by $\boldsymbol{\psi} \in \mathbb{R}^{d^{\psi}}$, the vector of local calibration variables by $\boldsymbol{\theta} \in \mathbb{R}^{d^{\theta}}$, and the vector of functions that map $\mathbf{x}$ to each element of $\boldsymbol{\theta}$ by $\mathcal{F}^{\theta} = [\mathcal{F}^{\theta}_1, ..., \mathcal{F}^{\theta}_{d^{\theta}}]^T$. With a slight abuse of notation, we denote the vector map from $\mathbf{x}$ to $\boldsymbol{\theta}$ using the vector of functions $\mathcal{F}^{\theta}$ as $\boldsymbol{\theta} = \mathcal{F}^{\theta}(\mathbf{x})$.

Suppose we have a computational model constructed according to the laws governing the physical system. Similar to the physical system, the response of our computational model is determined by the interactions between the control and the calibration variables. Since in a computational model we are not restricted to measuring any physical features, we can set the values of all $\mathbf{x}$, $\boldsymbol{\theta}$, and $\boldsymbol{\psi}$ arbitrarily. If we denote the computational process as $\mathcal{F}^s$, then the response of the computational model can be written as

$$y^s = \mathcal{F}^s(\mathbf{x}, \boldsymbol{\psi}, \boldsymbol{\theta}). \tag{3.1}$$

The goal of *calibration* is to adjust the variables $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$, such that the computational model represents the physical system in the sense that the computational model can predict the physical response at any input location $\mathbf{x}_*$.

Mathematically, calibration can be translated as the estimation of vectors $\mathcal{F}^{\theta}$ and $\boldsymbol{\psi}$, such that for any given $\mathbf{x}_*$, the function $\mathcal{F}^s : \mathbb{R}^{d^x} \times \mathbb{R}^{d^{\psi}} \times \mathbb{R}^{d^{\theta}} \longrightarrow \mathbb{R}$ generates a response close to y^p_* up to an error:

$$y^p_* = \mathcal{F}^s(\mathbf{x}_*, \tilde{\boldsymbol{\psi}}, \tilde{\mathcal{F}}^{\theta}(\mathbf{x}_*)) + \epsilon_*, \tag{3.2}$$

where $\tilde{\boldsymbol{\psi}}$ and $\tilde{\mathcal{F}}^{\theta}$ are estimations of $\mathcal{F}^{\theta}$ and $\boldsymbol{\psi}$, and ϵ_* exists due to assumptions and simplifications made in the computational model and also due to the estimation of

the control variables.

To estimate $\tilde{\boldsymbol{\psi}}$ and $\tilde{\mathcal{F}}^{\theta}$ in (3.2) we initially obtain m responses from $\mathcal{F}^{p}$ at a set of physical system inputs $\{\mathbf{x}_1^p, ..., \mathbf{x}_m^p\}$ to create a dataset corresponding to that physical system

$$\mathbf{P} = \left\{ p_i = (\mathbf{x}_i^p, y_i^p) \mid \mathbf{x}_i^p \in \mathbb{R}^{d^x}, y_i^p \in \mathbb{R}, i \in [m] \right\}.$$

We also create the counterpart of $\mathbf{P}$ in the computational model, i.e., the computational dataset, as

$$\mathbf{S} = \{ s_j = (\mathbf{x}_j^s, \boldsymbol{\psi}_j^s, \boldsymbol{\theta}_j^s, y_j^s) \mid \mathbf{x}_j^s \in \mathbb{R}^{d^x}, \boldsymbol{\psi}_j^s \in \mathbb{R}^{d^{\psi}}, \boldsymbol{\theta}_j^s \in \mathbb{R}^{d^{\theta}}, j \in [n] \},$$

based on a set of computational model inputs $\{(\mathbf{x}_1^s, \boldsymbol{\psi}_1^s, \boldsymbol{\theta}_1^s), \ldots, (\mathbf{x}_n^s, \boldsymbol{\psi}_n^s, \boldsymbol{\theta}_n^s)\}$.

Denoting $\boldsymbol{\theta}_i^p = \mathcal{F}^{\theta}(\mathbf{x}_i^p)$ and $\boldsymbol{\psi}^p$ as true values of the calibration variables, which we do not know yet, and assuming errors are i.i.d standard normal give the following model

$$y_i^p = \mathcal{F}^s(\mathbf{x}_i^p, \boldsymbol{\psi}^p, \boldsymbol{\theta}_i^p) + \epsilon_i^p, \text{ where } \epsilon_i^p \sim \mathcal{N}(0, \sigma^2), \quad \forall i \in [m]. \tag{3.3}$$

Note that applying Bayesian statistics to construct posterior distributions for parameters of model (3.3), i.e., $\boldsymbol{\psi}^p, \sigma^2$ and $\boldsymbol{\theta}_i^p$ for all $i \in [m]$, requires a large number of evaluations of $\mathcal{F}^s$, which is not practical for expensive computational models. Therefore, we model a GP distribution over $\mathcal{F}^s$,

$$\mathcal{F}^s \sim \mathcal{GP}\big(0, \mathcal{K}(\cdot, \cdot)\big), \tag{3.4}$$

where $\mathcal{K}(\cdot, \cdot)$ is a covariance function. Here we use the following anti-isotropic squared exponential covariance function

$$\mathcal{K}(\mathbf{z}, \mathbf{z}') = \alpha \exp\big(-(\mathbf{z} - \mathbf{z}')^T \boldsymbol{\Gamma} (\mathbf{z} - \mathbf{z}') \big), \tag{3.5}$$

where α is the magnitude parameter and $\boldsymbol{\Gamma}$ is a diagonal matrix of length-scale parameters. We denote the vector of the diagonal elements of $\boldsymbol{\Gamma}$ as $\boldsymbol{\gamma}$ with the length equal to length of the input vectors $\mathbf{z}$ and $\mathbf{z}'$.

Subsequently, we can obtain the likelihood of model (3.3) by the GP distribution defined on $\mathcal{F}^s$ in (3.4) as a multivariate normal distribution,

$$\mathbf{y}^p \mid \mathbf{X}^p, \mathbf{\Theta}^p, \mathbf{\Psi}^p, \boldsymbol{\gamma}, \alpha, \sigma^2 \sim \mathcal{N}(0, \mathbf{\Sigma} + \sigma^2 \mathbf{I}_m), \tag{3.6}$$

where $\mathbf{y}^p = [y_1^p, ..., y_m^p]^T$ is the vector of physical responses, $\mathbf{X}^p = [\mathbf{x}_1^p, ..., \mathbf{x}_m^p]^T$, $\mathbf{\Psi}^p = [\boldsymbol{\psi}^p, ..., \boldsymbol{\psi}^p]^T$, and $\mathbf{\Theta}^p = [\boldsymbol{\theta}_1^p, ..., \boldsymbol{\theta}_m^p]^T$ are matrices of size $m \times d^x$, $m \times d^\psi$, and $m \times d^\theta$ respectively, and $\mathbf{\Sigma}$ is the $m \times m$ covariance matrix whose elements are calculated by covariance function (3.5) with $[\mathbf{x}_i^{p^T}, \boldsymbol{\theta}_i^{p^T}, \boldsymbol{\psi}^{p^T}]^T$ as input vectors with length $(d^x + d^\theta + d^\psi)$.

Although in the process of deriving likelihood (3.6), we consider the columns of $\mathbf{\Theta}^p$ and $\mathbf{\Psi}^p$ as the input variables of model (3.3), we do not know the values of these input variables, and we intend to estimate them. Therefore, in order to distinguish the calibration variables, $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$ in (3.1) from the parameters in model (3.3), we call $\mathbf{\Theta}^p$ and $\mathbf{\Psi}^p$ as the *calibration parameters*.

We can estimate the parameters of model (3.3), i.e., the calibration parameters, the parameters of covariance function (3.5), and the variance of error, simultaneously using Bayesian statistics with the following posterior

$$\begin{aligned} &\pi(\mathbf{\Theta}^p, \boldsymbol{\psi}^p, \boldsymbol{\ell}^2, \gamma^2, \alpha, \sigma^2 \mid \mathbf{y}^p, \mathbf{X}^p) \\ &\propto \pi(\mathbf{y}^p \mid \mathbf{X}^p, \mathbf{\Theta}^p, \boldsymbol{\psi}^p, \boldsymbol{\gamma}, \alpha, \sigma^2)\pi(\mathbf{\Theta}^p)\pi(\boldsymbol{\psi}^p)\pi(\boldsymbol{\gamma})\pi(\alpha)\pi(\sigma^2), \end{aligned} \tag{3.7}$$

where we substitute $\mathbf{\Psi}^p$ by $\boldsymbol{\psi}^p$, since all columns of $\mathbf{\Psi}^p$ are essentially the same and $\mathbf{\Psi}^p$ can be easily recovered by the following transformation: $\mathbf{\Psi}^p = \mathbb{1}_{m \times d_\psi} \text{diag}(\boldsymbol{\psi}^p)$, where $\mathbb{1}_{m \times d_\psi}$ is a $m \times d_\psi$ matrix of ones.

Bayesian model (3.7) would be completed by specifying prior distributions for the parameters. However, our model suffers from unidentifiablity in the absence of week or uninformative priors due to the high-dimensionality of the parameter space. Therefore, in Section 3.5 we explain our approach to constructing informative priors

for $\boldsymbol{\Theta}^p$ and $\boldsymbol{\psi}^p$ by viewing the calibration as a non-isometric curve to surface matching problem.

Note that one might misinterpret the replacement $\mathcal{F}^s$ by $\mathcal{GP}\big(0, \mathcal{K}(\cdot,\cdot)\big)$ as a surrogate modeling approach used in the literature. However, our approach is fundamentally different from surrogate modeling, wherein the computational model is replaced by a fixed surrogate surface. This is in turn, trained based on a set of limited samples drawn from the computational model prior to any calibration procedure. By contrast, in our model, building and training the GP is a part of the calibration process.

3.3 Calibration as a non-isometric curve to surface matching problem: A special case

In this section, we explain how the calibration problem can be viewed as a non-isometric curve to surface matching problem for the special case when $\mathbf{x} \in \mathbb{R}, \boldsymbol{\theta} \in \mathbb{R}$, and $\boldsymbol{\psi} \in \emptyset$, i.e. no global calibration variable. From a geometric perspective, all the possible values for $\mathbf{x}$ and $\mathcal{F}^p(\mathbf{x})$ constitute the curve $(\mathbf{x}, \mathcal{F}^p(\mathbf{x}))$ in a two-dimensional space. In the computational model however, we can specify the values of both $\mathbf{x}$ and $\boldsymbol{\theta}$. Consequently, all the possible values of $\mathbf{x}$, $\boldsymbol{\theta}$, and $\mathcal{F}^s(\mathbf{x}, \boldsymbol{\theta})$ together form a surface $(\mathbf{x}, \boldsymbol{\theta}, \mathcal{F}^s(\mathbf{x}, \boldsymbol{\theta}))$ in a three-dimensional space. As we noted in Section 3.1, the true physical curve lies on the three-dimensional computational model surface, i.e., $(\mathbf{x}, \mathcal{F}^\theta(\mathbf{x}), \mathcal{F}^p(\mathbf{x}))$. However, since we do not observe the actual values of the calibration variables in the physical process, we only see a projected curve in $\mathbf{x} - y$ space (see Figure (3.1)). Hence, the calibration problem is to recover the true physical curve, or in other words, determine a non-isometric match of a curve to a surface.

The non-isometry is due to the fact that the curve $(\mathbf{x}, \mathcal{F}^\theta(\mathbf{x}, \mathcal{F}^p(\mathbf{x}))$ on the three-dimensional space $\mathbf{x}-\boldsymbol{\theta}-y$ has a different length than the projected curve $(\mathbf{x}, 0, \mathcal{F}^p(\mathbf{x}))$ on a two-dimensional space $\mathbf{x} - y$. Therefore, this is in principle, different from isometric matching problems [34, 9, 1].

Furthermore, in practice we only have the finite physical system dataset $\mathbf{P}$ along with a finite computational model dataset $\mathbf{S}$, as we do not observe a complete curve or surface, but a scatter of data points from $\mathbf{P}$ and $\mathbf{S}$. Ideally, the data points from the former lie on the projected curve we observe and the data from the latter lie on the computational model surface (see Figure 3.2). Hence, what we observe is incomplete data and we aim to match non-isometrically an incomplete curve to an incomplete surface, which is equivalent to solving the calibration problem.

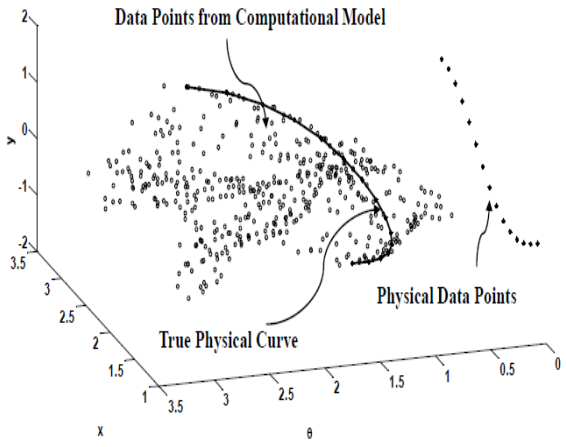


Figure 3.2: Non-isometric curve to surface matching perspective of local calibration with *incomplete data*; compare with Figure 3.1. In practice, we observe a scatter of data points sampled from the complete curve and surface, which is depicted in this plot.

This geometric perspective that the calibration problem is a non-isometric curve to surface matching problem, and that we are provided with a finite set of data points from the curve and the surface, motivates us to view the problem through a discrete combinatorial lens and model the problem using graph-theoretic approaches. Our graph-based solution to the non-isometric curve to surface matching problem provides us with a set of computational model data points, which carry information about the calibration parameters. We call this set of computational data points *anchor points*.

These anchor points will then be used in Section 3.5 to construct prior distributions for our Bayesian model.

We seek to identify a set of anchor points among the computational data points, which intuitively are "close" to the points on the true physical curve. In other words, the anchor points are positioned such that the true physical curve passes through neighborhoods of those points. To find these anchor points, we desire two properties: (i) the computational model response should be close to the physical response for a given input $\mathbf{x}$; and (ii) the parameter values for two consecutive anchor points should be close to each other. The former drives our method to identify the anchor points that have similar responses as that of the physical system, and the latter aims to encourage the smoothness of the physical curve.

Note that we are only interested in identifying these "optimal" anchor points that provide us with information about the true physical curve to be used in our prior distributions, and not the true physical curve itself. However, one could also directly use the selected anchor points to approximate the true physical curve via interpolation. Given our focus on expensive computational models wherein the number of computational model data points is limited, the approximation of the true physical curve may not be accurate. In the next section, we formally define and address the problem of finding anchor points with the desired properties using a rigorous graph-theoretic approach, for the special case when $\mathbf{x} \in \mathbb{R}$ and $\boldsymbol{\theta} \in \mathbb{R}$ and $\boldsymbol{\psi} \in \emptyset$.

3.3.1 A graph-theoretic approach for finding anchor points

Without loss of generality, we assume that all the data points in the physical system and the computational model datasets are strictly ordered such that $\mathbf{x}_i^p < \mathbf{x}_{i+1}^p$, for all $i \in [m-1]$, and $\mathbf{x}_j^s < \mathbf{x}_{j+1}^s$, for all $j \in [n-1]$. Then, we construct an edge-weighted directed graph $G = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V} = \mathbf{V}^0 \cup \{0, n+1\}$, and the vertices in $\mathbf{V}^0 = [n]$ correspond to the computational model data points in $\mathbf{S}$. We refer to G as

the *calibration digraph* (originally introduced in [79]).

We partition the $\mathbf{V}^0$ into m clusters $\mathbf{C}_1, \ldots, \mathbf{C}_m$ as follows: Consider any vertex $j \in \mathbf{V}^0$ corresponding to $s_j \in \mathbf{S}$. Then, j is assigned to a unique cluster $\mathbf{C}_i$ for some $i \in [m]$ as follows

$$j \in \mathbf{C}_i \iff i = \min \left\{ \arg\min_{\ell \in [m]} \{ ||\mathbf{x}_\ell^p - \mathbf{x}_j^s||_2 \} \right\}. \tag{3.8}$$

Note that, if the inner minimum in (3.8) is not unique, then the tie is broken by choosing the smallest index, by the outer minimum. As a consequence, each cluster $\mathbf{C}_i$ is in 1-to-1 correspondence with the physical data point p_i. We can now describe the set of directed edges $\mathbf{E}$ as follows

$$\mathbf{E} = \bigcup_{i \in [m-1]} \{(u,v) \mid u \in \mathbf{C}_i, v \in \mathbf{C}_{i+1}\} \bigcup \{(0,u) \mid u \in \mathbf{C}_1\} \bigcup \{(u, n+1) \mid u \in \mathbf{C}_m\}. \tag{3.9}$$

This construction is illustrated in Figure 3.3.

The final critical step is assigning weights w_{uv} to each edge $(u,v) \in \mathbf{E}$. Consider two consecutive clusters $\mathbf{C}_i$ and $\mathbf{C}_{i+1}$ for any $i \in [m]$ and let $u \in \mathbf{C}_i$ and $\mathbf{v} \in \mathbf{C}_{i+1}$. Define w_{uv} as follows

$$w_{uv} = |y_u^s - y_u^p| + \lambda ||\boldsymbol{\theta}_u^s - \boldsymbol{\theta}_v^s||_2, \tag{3.10}$$

where $\lambda > 0$ is a scaling parameter. The weights $w_{0,u}$ for all $u \in \mathbf{C}_1$ and $w_{u,n+1}$ for all $u \in \mathbf{C}_m$ are identically zero. The edge-weight for any edge between two consecutive clusters i and $i+1$ consists of two parts: the first part represents the difference between the model response and physical response; the second part represents the difference between the calibration parameters of i and $i+1$. On this digraph G, we intend to solve the shortest (simple, directed) path problem from origin vertex 0 to destination vertex $n+1$. Every path from 0 to $n+1$ in G has exactly $m+1$ edges by construction. Therefore, the m internal vertices on the shortest path that is found will serve as the anchor points.

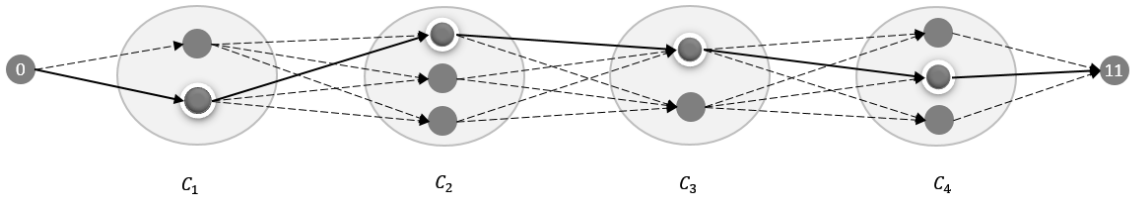


Figure 3.3: Illustration of the calibration digraph for the case where $m = 4$ and $n = 10$. The vertices represent data points from the computational model and the clusters $\mathbf{C}_1$ through $\mathbf{C}_m$ correspond to physical system data points p_1 through p_m. Vertices denoted by dark circles with a white border represent the anchor points, and the solid arrows identify the edges in the shortest path found.

Lemma 3.1 *Calibration digraph* $G = (\mathbf{V}, \mathbf{E})$ *is acyclic with a topological ordering* $\langle 0, 1, \ldots, n, n+1 \rangle$.

Proof. It suffices to show that if $(u, v) \in \mathbf{E}$, then $u < v$, which is trivially true when $u = 0$ and $v = n + 1$. For distinct $u, v \in \mathbf{V}^0$, note that $u < v \iff \mathbf{x}_u^s < \mathbf{x}_v^s$ as we have assumed the $\mathbf{S}$ to be strictly ordered. Suppose, $u \in \mathbf{C}_i$ and $v \in \mathbf{C}_{i+1}$ for some $i \in [m-1]$. Hence, $\mathbf{x}_i^p < \mathbf{x}_{i+1}^p$, and by (3.8) we can conclude that the distinct points $\mathbf{x}_u^s$ and $\mathbf{x}_v^s$ satisfy,

$$\mathbf{x}_u^s \leq \frac{1}{2}(\mathbf{x}_i^p + \mathbf{x}_{i+1}^p) \leq \mathbf{x}_v^s.$$

■

Since G is a directed acyclic graph, we can solve the shortest path problem using an $\mathcal{O}(|\mathbf{E}|)$ algorithm that scans out-going edges from each vertex in the topological order and updating distance-labels as needed [7, 51] . Suppose $0 - v_1 - v_2 - \cdots - v_m - (n+1)$ is the shortest path identified, then those s_j corresponding to each v_1 to v_m serve as the anchor points.

3.4 Generalization of the non-isometric curve to surface matching problem to higher dimensions

In Section 3.3, we introduced the curve to surface matching interpretation of calibration with $\mathbf{x} \in \mathbb{R}$ and $\boldsymbol{\theta} \in \mathbb{R}$. This special case allowed us develop a graph-theoretic approach to anchor point selection that admitted a linear-time algorithm. The geometric perspective underlying that approach can be generalized to arbitrary dimensions as a hyper-curve to hyper-surface matching problem. However, in the general setting, there is no straightforward extension of the directed acyclic graph model. Recall that the model hinges on the natural ordering of the computational and the physical data points on the real line, which does not exist in higher dimensions. So in this section, we introduce a different calibration graph model and an associated combinatorial optimization problem to find the anchor points in any arbitrary dimension. As with the special case, the anchor points will subsequently be used in Section 3.5 to construct prior distributions for our Bayesian model.

For the general case, we construct a *calibration graph* $G = (\mathbf{V}, \mathbf{E})$ that is undirected and edge-weighted, where $\mathbf{V} = [n]$. We partition $\mathbf{V}$ into m clusters, $\mathbf{C}_1, \ldots, \mathbf{C}_m$, in correspondence with the m physical data points and assign vertex j to a unique cluster $\mathbf{C}_i$ for some $i \in [m]$ by the same rule in (3.8). The graph G is a complete m-partite graph with partitions $\mathbf{C}_1, \ldots, \mathbf{C}_m$ with the edge set given by,

$$\mathbf{E} = \bigcup_{\substack{i,\ell \in [m] \\ i<\ell}} \{\{u, v\} \mid u \in \mathbf{C}_i, v \in \mathbf{C}_\ell\}.$$

Figure 3.4 illustrates this construction.

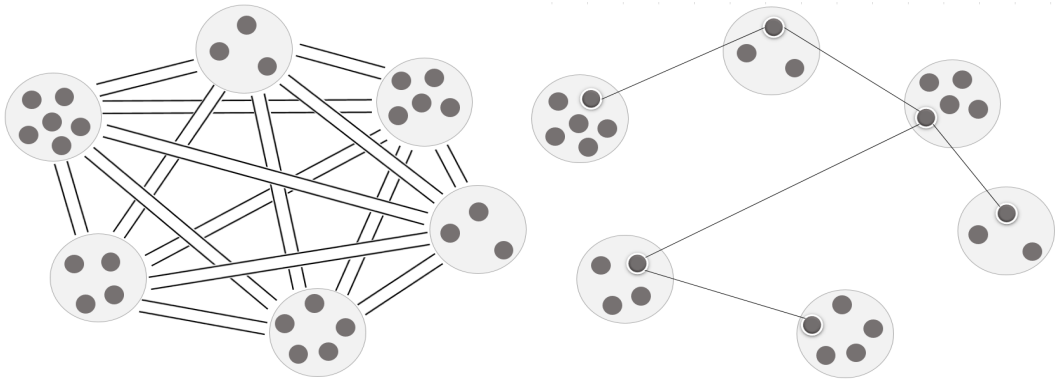
Finally, before defining the edge-weights, we introduce two required concepts. The *calibration vector* of the data point s_j is given by $[\boldsymbol{\theta}_j^{s^T}, \boldsymbol{\psi}_j^{s^T}]^T$. It is desirable for the calibration vectors of two neighboring anchor points to be close. For any two data points $s_j, s_{j'} \in \mathbf{S}$, the computational dataset, are said to be neighbors if the Euclidean distance of their control vectors is smaller than a predefined radius r, i.e.,

$||\mathbf{x}_j^s - \mathbf{x}_{j'}^s||_2 < r$.

We assign the weight w_{uv} to the edge $\{u, v\} \in \mathbf{E}$, where $u \in \mathbf{C}_i$ and $v \in \mathbf{C}_\ell$, as follows

$$w_{uv} = \begin{cases} |y_u^s - y_i^p| + |y_v^s - y_\ell^p| + \lambda||[\boldsymbol{\theta}_u^{s^T}, \boldsymbol{\psi}_u^{s^T}]^T - [\boldsymbol{\theta}_v^{s^T}, \boldsymbol{\psi}_v^{s^T}]^T||_2 & \text{if } ||\mathbf{x}_u^s - \mathbf{x}_v^s||_2 \leq r \\ |y_u^s - y_i^p| + |y_v^s - y_\ell^p| + M||\mathbf{x}_u^s - \mathbf{x}_v^s||_2 & \text{if } ||\mathbf{x}_u^s - \mathbf{x}_v^s||_2 > r, \end{cases} \tag{3.11}$$

where λ is a scaling parameter and M is a sufficiently large number. Note that the weights assigned in (3.11) extend the ideas behind equation (3.10). Here, the edge-weight between vertices $u \in \mathbf{C}_i$ and $v \in \mathbf{C}_\ell$, where s_u and s_v are neighbors, consists of two parts, similar to (3.10): the first part measures the distance between each vertex's response and the physical system response associated with the cluster to which it belongs, i.e., $|y_u^s - y_i^p|$ and $|y_v^s - y_\ell^p|$; the second part measures the distance between the vertex's calibration vectors, i.e., $||[\boldsymbol{\theta}_u^{s^T}, \boldsymbol{\psi}_u^{s^T}]^T - [\boldsymbol{\theta}_v^{s^T}, \boldsymbol{\psi}_v^{s^T}]^T||_2$. Let $\mathbf{E}_1$ denote the set of these edges that join neighbors. The remainder of the edges, $\mathbf{E}_2 = \mathbf{E} \backslash \mathbf{E}_1$ correspond to edges between computational data points that are not neighbors. We assign relatively large weights to these edges by setting M to a large value. Furthermore, the weight on such edges increases as the distance between the control vectors of the end-points increases.



(a) Calibration graph (b) Generalized minimum spanning tree

Figure 3.4: (a) A calibration graph where each black circle represents a vertex and each two parallel lines represent edges between vertices of two clusters. (b) A generalized spanning tree in the calibration graph

To identify the "optimal" anchor vertices on this calibration graph we use a minimum weight tree that selects exactly one vertex from each cluster to connect all the clusters. In the combinatorial optimization literature, this problem is known as the *Generalized Minimum Spanning Tree* (GMST) problem [63]. By construction, a GMST will tend to include edges in $\mathbf{E}_1$ as they are lighter. However, if no GMST exists that only uses edges in $\mathbf{E}_1$, it will be forced to include edges in $\mathbf{E}_2$.

3.4.1 Integer programming approach to GMST problem

GMST problem was introduced by the authors of [63] who showed that it is NP-hard and does not admit a polynomial-time constant-factor approximation algorithm unless P=NP. Since, comprehensive studies of the problem were undertaken in [78, 23]. For instance, the problem is known to be NP-hard even when restricted to trees and admits a dynamic programming algorithm that is exponential in the number of clusters, but quadratic in the number of vertices [76].

Consequently, considerable amount of work developing integer programming (IP)

formulations for this problem and studying the associated LP relaxations has been undertaken by various authors [63, 24, 76, 77]. In addition to integer programming approaches, heuristic approaches for solving the GMST problem are proposed in [78, 31]. Our focus in this section is on two strong formulations, one with exponentially many constraints and the other compact, with polynomially many constraints and variables, and using them to solve the anchor point selection problem in general dimensions.

Cutset and subtour elimination formulations for this problem, analogous to those available for the traveling salesperson problem and the minimum spanning tree problem (see for instance [8]), were first introduced by [63] and subsequently advanced by others [78, 23]. These formulations also have directed counterparts, based on the observation that GMST problem is a special case of the *generalized minimum spanning arborescence* (GMSA) problem, wherein we seek an arborescence of minimum weight in a directed graph that is rooted at some vertex in a specified cluster and containing exactly one vertex per cluster. We can transform the GMST problem to the GMSA problem by replacing each undirected edge $\{i, j\}$ of graph G with anti-parallel arcs (i, j) and (j, i), also known as the *complete orientation* of G, with each arc assigned the same weight as the undirected edge, and by arbitrarily choosing one of the clusters to contain the root.

In addition to the aforementioned formulations with exponentially many constraints, multi-commodity flow based compact formulations for the problem are also available that use polynomially many constraints and variables [63, 24, 76, 77]. For a detailed review of the IP formulations available for this problem and a comparison of the strength of the LP relaxations, we refer the reader to [24].

Authors of [24] showed that a strengthening of the directed counterpart of the subtour elimination formulation, what they call the *directed cluster subpacking* (DCSUB) formulation, is among the strongest in terms of the tightness of the LP relaxation, but

uses exponentially many constraints. For the remainder of this discussion we assume we have the complete orientation of the weighted calibration graph $G = (\mathbf{V}, \mathbf{E})$, that we denote by $\vec{G} = (\mathbf{V}, \mathbf{A})$, on which we wish to formulate the GMSA problem. The arc weights w_{uv} for each $(u, v) \in \mathbf{A}$ are assigned by duplicating the corresponding undirected edge-weights in G as described above and we require the arborescence to be rooted at some vertex in the first cluster, $\mathbf{C}_1$. Recall that the vertex set $\mathbf{V}$ is partitioned into clusters $\mathbf{C}_1, \ldots, \mathbf{C}_m$. In the formulations that follow, we use the notations: $\mathbf{A}(\mathbf{Q}) = \{(u, v) \in \mathbf{A} \mid u, v \in \mathbf{Q} \subset \mathbf{V}\}$, $\boldsymbol{\delta}^+(u) = \{a \in \mathbf{A} \mid u \text{ is the tail of } a\}$, and $\boldsymbol{\delta}^-(u) = \{a \in \mathbf{A} \mid u \text{ is the head of } a\}$. We use binary decision vectors $q \in \{0, 1\}^{|\mathbf{A}|}$ and $b \in \{0, 1\}^{|\mathbf{V}|}$ to denote the incidence vectors of the arcs and vertices included in the arborescence.

$$\min \quad \sum_{a \in \mathbf{A}} w_a q_a \tag{3.12a}$$

$$\text{s.t.} \quad \sum_{v \in \mathbf{C}_i} b_v = 1 \qquad \forall i \in [m] \tag{3.12b}$$

$$q_{uv} + q_{vu} \leq 1 \qquad \forall \{u, v\} \in \mathbf{E} \tag{3.12c}$$

$$\sum_{a \in \mathbf{A}} q_a = m - 1 \tag{3.12d}$$

$$\sum_{a \in \mathbf{A}(\mathbf{Q})} q_a \leq \sum_{v \in \mathbf{Q}} b_v - 1 \quad \forall \mathbf{Q} \subset \mathbf{V} \text{ such that } \mathbf{Q} \supset \mathbf{C}_i \text{ for some } i \in [m] \tag{3.12e}$$

$$\sum_{a \in \boldsymbol{\delta}^-(v)} q_a = b_v \qquad \forall v \in \mathbf{V} \backslash \mathbf{C}_1 \tag{3.12f}$$

$$b \in \{0, 1\}^{|\mathbf{V}|}, q \in \{0, 1\}^{|\mathbf{A}|} \tag{3.12g}$$

Constraints (3.12b) enforce that the model choose exactly one vertex from each cluster, constraints (3.12d) ensure that exactly $m - 1$ arcs from $\mathbf{A}$ are selected and constraints (3.12c) ensure that these correspond to $m-1$ distinct edges in $\mathbf{E}$. Cluster subpacking constraints (3.12e) prevent solutions that contain cycles and were shown by the authors of [24] to dominate the more familiar subtour elimination constraints

introduced in [63]:

$$\sum_{a \in \mathbf{A}(\mathbf{Q})} q_a \leq \sum_{v \in \mathbf{Q} \setminus \{u\}} b_v \quad \forall u \in \mathbf{Q} \subset \mathbf{V} \text{ such that } 2 \leq |\mathbf{Q}| \leq |\mathbf{V}| - 1.$$

Finally, constraints (3.12f) ensure that every non-root vertex selected by the solution has exactly one incoming edge and every vertex outside $\mathbf{C}_1$ that is not selected will have no incoming arcs; along with the requirement that we choose exactly m vertices and $m-1$ arcs without creating cycles, this ensures that we obtain an arborescence rooted at some vertex inside $\mathbf{C}_1$.

Next we present the multi-commodity flow (MCF) formulation for the GMSA problem that avoids using exponentially many constraints, but uses an additional set of variables. The MCF formulation treats every vertex $v \in \mathbf{C}_1$ to have supply b_v for each commodity $i \in \{2, \ldots, m\}$ corresponding to the remaining clusters; it treats every $v \in \mathbf{C}_i$ to have a demand of b_v for commodity i. Suppose $b_v = 1$ for some $v \in \mathbf{C}_i$, then a path must be traced from the root selected in $\mathbf{C}_1$ to deliver commodity i. We use the additional set of commodity-flow variables f_a^i to denote the amount of commodity $i \in \{2, \ldots, m\}$ flowing on arc $a \in \mathbf{A}$.

$$\min \quad \sum_{a \in \mathbf{A}} q_a w_a \tag{3.13a}$$

$$\text{s.t.} \quad (3.12b), (3.12c), (3.12d), (3.12g)$$

$$\sum_{a \in \boldsymbol{\delta}^+(v)} f_a^i - \sum_{a \in \boldsymbol{\delta}^-(v)} f_a^i = \left\{ \begin{array}{rl} b_v, & v \in \mathbf{C}_1 \\ -b_v, & v \in \mathbf{C}_i \\ 0, & v \notin \mathbf{C}_1 \cup \mathbf{C}_i \end{array} \right\} \quad \forall i \in \{2, \ldots, m\} \tag{3.13b}$$

$$0 \leq f_a^i \leq q_a \qquad \forall a \in \mathbf{A}, i \in \{2, \ldots, m\} \tag{3.13c}$$

Constraints (3.13b) are flow balance constraints for each commodity and constraints (3.13c) prevent flows on the edges that are not selected.

Formulations (3.13) and (3.12) are both equally good in terms of the tightness of the LP relaxations as the projection of the LP relaxation of the former on to the (b, q)-

space is the same as the LP relaxation of the latter [24]. As noted before, the former is compact while the latter has exponentially many cycle elimination constraints. Consequently, a direct monolithic implementation of formulation (3.12) is impractical even for small scale problems.

Nonetheless, a delayed constraint generation approach could be effective in practice [12, 61]. This approach starts by relaxing formulation (3.12) by omitting constraints (3.12e). During the normal progress of an LP relaxation based branch-and-bound algorithm to solve the relaxed IP, whenever integral solutions are detected, it is necessary to verify if any violated constraint exists among those that were initially excluded, and add them to the model. This ensures the overall correctness of the algorithm. An effective implementation of such an algorithm is possible using the "lazy cut" feature of most state-of-the-art commercial IP solvers as long as separation of the violated cycle constraints can be accomplished quickly. In Section 3.8, we use DCSUB formulation along with delayed constraint generation for selecting anchor points.

3.5 Prior distributions

In this section we describe how the information about the true physical curve carried by the anchor points, which were found in Sections 3.3 and 3.4, can be utilized to construct our prior distributions for the calibration parameters.

Suppose $\boldsymbol{\theta}_i^a, \boldsymbol{\psi}_i^a$ be the calibration vectors of the anchor point associated with the i^{th} physical data point, i.e., the anchor vertex selected from the i^{th} cluster. Define the matrix $\boldsymbol{\Theta}^a = [\boldsymbol{\theta}_1^a, ..., \boldsymbol{\theta}_m^a]^T$ of size $m \times d^\theta$ and the mean vector $\boldsymbol{\psi}^a = \frac{1}{m}\sum_{i=1}^m \boldsymbol{\psi}_i^a$ of length d^ψ. Note that for the mean vector, we take the average since we assume that the global calibration parameters are constant regardless of the values of control vectors.

For each component of $\boldsymbol{\psi}^p$, we consider a univariate normal distribution centered

at the corresponding element in $\boldsymbol{\psi}^a$ with an unknown variance as the choice of the prior distribution. Therefore, we have the following prior distribution for $\boldsymbol{\psi}^p$:

$$\boldsymbol{\psi}^p \mid \boldsymbol{\psi}^a, \boldsymbol{\tau}^2 \sim \mathcal{N}(\boldsymbol{\psi}^a, \mathrm{diag}(\boldsymbol{\tau}^2)), \tag{3.14}$$

where $\boldsymbol{\tau}^2$ is the vector of variances in the normal distributions with length d^{ψ}.

Applying the same procedure for constructing prior distributions for the local calibration parameters increases the dimension of the parameter space, since we need to define md^{θ} variance parameters, while these parameters are nuisance parameters and not of interest to our model. Therefore, in order to shrink the parameter space, we use the fact that the k^{th} column of $\boldsymbol{\Theta}^p$, i.e. $\boldsymbol{\Theta}^p_k$, is actually a realization of the functional relationship $\mathcal{F}^{\theta}_k$. Therefore, k^{th} column of $\boldsymbol{\Theta}^a$, i.e., $\boldsymbol{\Theta}^a_k$, is a rough estimator of this realization. With this logic, we use a single variance parameter for all the elements in $\boldsymbol{\Theta}^p_k$, and construct the prior distribution for $\boldsymbol{\Theta}^p_k$ as:

$$\boldsymbol{\Theta}^p_k \mid \boldsymbol{\Theta}^p_k, \nu^2_k \ \sim \mathcal{N}(\boldsymbol{\Theta}^a_k, \nu^2_k \mathbf{I}_m), \tag{3.15}$$

where ν^2_k is the k^{th} element of the vector of variances $\boldsymbol{\nu}^2$ with length d^{θ}.

The correctness of the normality assumptions in (3.14) and (3.15) is a legitimate concern. In fact, there is no guarantee that the anchor points embrace the true physical curve because of the limited number data points from the computational model. However, we note that we only make the normality assumptions in (3.14) and (3.15) for constructing the prior distributions, and the Bayesian model will adjust these priors by likelihood (3.6). Moreover, in order to construct stronger and more accurate prior distributions we recommend drawing samples from the computational model only at the locations of the physical system inputs $\{\mathbf{x}^p_1, ..., \mathbf{x}^p_m\}$, as this increases the likelihood of finding anchor points close to the true physical curve on the computational model surface.

3.6 Posterior distribution

In this section we complete posterior distribution (3.7) using the prior distributions constructed in Section 3.5, and define proper prior distributions for the rest of the parameter of the model as

$$p(\boldsymbol{\Theta}^p, \boldsymbol{\psi}^p, \boldsymbol{\nu}^2, \boldsymbol{\tau}^2, \boldsymbol{\gamma}, \alpha, \sigma^2 \mid \mathbf{y}^p, \mathbf{X}^p, \boldsymbol{\Theta}^a, \boldsymbol{\psi}^a) \propto$$

$$p(\mathbf{y}^p \mid \mathbf{X}^p, \boldsymbol{\Theta}^p, \boldsymbol{\psi}^p, \boldsymbol{\gamma}, \alpha, \sigma^2) p(\boldsymbol{\Theta}^p \mid \boldsymbol{\Theta}^a, \boldsymbol{\nu}^2) p(\boldsymbol{\psi}^p \mid \boldsymbol{\psi}^a, \boldsymbol{\tau}^2) p(\boldsymbol{\nu}^2) p(\boldsymbol{\tau}^2) p(\boldsymbol{\gamma}) p(\alpha) p(\sigma^2),$$

where

$$\begin{aligned}
&\mathbf{y}^p \mid \mathbf{X}^p, \boldsymbol{\Theta}^p, \boldsymbol{\psi}^p, \boldsymbol{\gamma}, \alpha, \sigma^2 \sim \mathcal{N}(0, \boldsymbol{\Sigma} + \sigma^2 \mathbf{I}_m) && \\
&\boldsymbol{\Theta}^p_k \mid \boldsymbol{\Theta}^a_k, \nu^2_k \ \sim \mathcal{N}(\boldsymbol{\Theta}^a_k, \nu^2_k \mathbf{I}_m) && \forall k \in [d^\theta] \\
&\boldsymbol{\psi}^p \mid \boldsymbol{\psi}^a, \boldsymbol{\tau}^2 \sim \mathcal{N}(\boldsymbol{\psi}^a, \mathrm{diag}(\boldsymbol{\tau}^2)) && \\
&\nu^2_k \sim \frac{1}{\nu^2_k} && \forall k \in [d^\theta] \\
&\tau^2_h \sim \text{Inv-Gamma}(\alpha_\tau, \beta_\tau) && \forall h \in [d^\psi] \\
&\gamma_j \sim \text{Log-Gamma}(\alpha_\gamma, \beta_\gamma) && \forall j \in [d^x + d^\theta + d^\psi] \\
&\alpha \sim \text{Log-Uniform} && \\
&\sigma^2 \sim \text{Log-Uniform}. &&
\end{aligned}$$

For each ν^2_k we use a flat Jeffreys prior [44], which is an inverse gamma distribution with zero value for both of the shape and the scale parameter. For each τ^2_h we choose a weak inverse gamma distribution, i.e., an inverse gamma with large variance, with $\alpha_\tau = 2.1$ and $\beta_\tau = 10$ as its parameters. Note that both of these priors are conjugate for their associated parameters in the posterior distribution. Moreover, as recommended in [28], to improve the identifiability of the model, we use the prior distributions on the logarithmic scale for parameters of the GP part of the model. Therefore, for σ^2 and α we use a flat log-uniform distribution and for each γ^2_j we

use a log-gamma distribution with the parameters $\alpha_\ell = \beta_\ell = 2$. Hence, we have the following full posterior distribution

$$
\begin{aligned}
& p(\boldsymbol{\Theta}^p, \boldsymbol{\psi}^p, \boldsymbol{\nu}^2, \boldsymbol{\tau}^2, \boldsymbol{\gamma}, \alpha, \sigma^2 \mid \mathbf{y}^p, \mathbf{X}^p, \boldsymbol{\Theta}^a, \boldsymbol{\psi}^a) \\
& \propto |\boldsymbol{\Sigma} + \sigma \mathbf{I}_m|^{-0.5} \exp\{\frac{-1}{2} \mathbf{y}^{pT} (\boldsymbol{\Sigma} + \sigma \mathbf{I}_m)^{-1} \mathbf{y}^p\} \\
& \times \prod_k (\nu_k^2)^{-m/2-1} \exp\{\frac{-1}{2\nu_k^2} (\boldsymbol{\Theta}_k^p - \boldsymbol{\Theta}_k^a)^T (\boldsymbol{\Theta}_k^p - \boldsymbol{\Theta}_k^a)\} \\
& \times \prod_h (\tau_h^2)^{-1/2-\alpha_\tau-1} \exp\{\frac{-1}{2\tau_h^2} (\psi_h - \psi_h^a)^2\} \exp\{\frac{-\beta_\tau}{\tau_h^2}\} \\
& \times \prod_j \frac{1}{\gamma_j^2} \log(\gamma_j^2)^{\alpha_\gamma - 1} \exp\{-\beta_\gamma \log(\gamma_j^2)\} \\
& \times \frac{1}{\alpha} \times \frac{1}{\sigma^2}
\end{aligned} \tag{3.16}
$$

We use Gibbs sampling [27] to sequentially sample from the full conditional posterior distributions. Note that the full conditional posterior distributions of all the parameters except ν_k^2 and τ_h^2, which have inverse gamma distributions, require taking Metropolis-Hastings [58] steps due to their unknown forms. In next section we explain how the samples drawn from the posterior distribution (3.16) can be used for prediction of the calibration and response values of a new control vector.

3.7 Prediction of the calibration and the response variables

In order to make prediction for a new control vector $\mathbf{x}_*$, first we need to predict its associated local calibration vector $\boldsymbol{\theta}^* = \mathcal{F}^\theta(\mathbf{x}_*)$. We estimate the k^{th} element of $\boldsymbol{\theta}^*$, i.e., θ_k^*, by fitting a GPR on $\mathcal{F}_k^\theta$ using the samples drawn form posterior distribution (3.16). First let the $\boldsymbol{\Theta}^p(t)$ be t^{th} draw from posterior distribution (3.16) after some burn in period, where $t \in [T]$. Since $\boldsymbol{\Theta}_k^p(t)$ is in fact one realization of $\mathcal{F}_k^\theta$ at design locations $\{\mathbf{x}_1^p, ..., \mathbf{x}_m^p\}$, so we can write

$$
\boldsymbol{\Theta}_k^p(t) = [\mathcal{F}_k^\theta(\mathbf{x}_1), ..., \mathcal{F}_k^\theta(\mathbf{x}_m)]^T + [\epsilon_1^\theta, ..., \epsilon_m^\theta]^T, \tag{3.17}
$$

where $[\epsilon_1^\theta, ..., \epsilon_m^\theta]^T \sim \mathcal{N}(0, \sigma_k^\theta \mathbf{I}_m)$.

By assuming a GP distribution on each $\mathcal{F}_k^\theta$, i.e., $\mathcal{F}_k^\theta \sim \mathcal{GP}\big(0, \mathcal{K}(\cdot,\cdot)\big)$, we have the following normal distribution for $\mathbf{\Theta}_k^p(t)$

$$\mathbf{\Theta}_k^p(t) \sim \mathcal{N}(0, \mathbf{\Sigma}_{\mathbf{X}^p\mathbf{X}^p} + \sigma_k^\theta \mathbf{I}_m), \tag{3.18}$$

where we use the notation $\mathbf{\Sigma}_{\mathbf{ZZ}'}$ as the covariance between columns of matrices $\mathbf{Z}$ and $\mathbf{Z}'$.

Moreover, we can write the joint distribution of $\mathbf{\Theta}_k^p(t)$ and the prediction of θ_k^* for the t^{th} draw, i.e., $\theta_k^*(t)$, as

$$\begin{bmatrix} \mathbf{\Theta}_k^p(t) \\ \theta_k^*(t) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{\Sigma}_{\mathbf{X}^p\mathbf{X}^p} + \sigma_k^\theta \mathbf{I}_m & \mathbf{\Sigma}_{\mathbf{X}^p\mathbf{x}_*} \\ \mathbf{\Sigma}_{\mathbf{x}_*\mathbf{X}^p} & \mathbf{\Sigma}_{\mathbf{x}_*\mathbf{x}_*} \end{bmatrix}\right). \tag{3.19}$$

By conditioning on $\mathbf{\Theta}_k^p(t)$ in (3.19), the point prediction of $\theta_k^*(t)$ is obtained,

$$\theta_k^*(t) = \mathbf{\Sigma}_{\mathbf{x}_*\mathbf{x}^p}(\mathbf{\Sigma}_{\mathbf{X}^p\mathbf{X}^p} + \sigma_k^\theta \mathbf{I}_m)^{-1}\mathbf{\Theta}_k^p(t). \tag{3.20}$$

Note that for each t^{th} prediction in (3.20), we need to tune the hyper-parameters of the covariance function used in (3.18), which can be obtained by maximizing the following logarithm of likelihood corresponding to (3.18) [88],

$$\log\left(p\big(\Theta_k^p(t)\big)\right) \propto -\log|\mathbf{\Sigma}_{\mathbf{X}^p\mathbf{X}^p} + \sigma_k^\theta \mathbf{I}_m| - \mathbf{\Theta}_k^{p^T}(t)(\mathbf{\Sigma}_{\mathbf{X}^p\mathbf{X}^p} + \sigma_k^\theta \mathbf{I}_m)^{-1}\mathbf{\Theta}_k^p(t). \tag{3.21}$$

To find point and interval predictions for the new response y_*, we make T predictions based on the T samples we drew from posterior (3.16) and the T predictions we made for the vector $\boldsymbol{\theta}^*$ using (3.20). To this end let $\boldsymbol{\psi}^p(t), \boldsymbol{\gamma}(t), \alpha(t), \sigma^2(t)$ be t^{th} samples from posterior (3.16), and $\boldsymbol{\theta}^*(t) = [\theta_1^*(t), ..., \theta_{d^\theta}^*(t)]^T$ be the prediction of the local calibration vector obtained from (3.20). Since we have assumed a GP distribution on $\mathcal{F}^s$, we can use the GP predictive distribution to derive the t^{th} prediction,

$$y_*(t) \sim \mathcal{N}\Big(\mathbf{\Sigma}_{\mathbf{v}^*(t)\mathbf{V}(t)}\big(\mathbf{\Sigma}_{\mathbf{V}(t)\mathbf{V}(t)} + \sigma^2(t)\mathbf{I}_m\big)^{-1}\mathbf{y}^p,$$
$$\mathbf{\Sigma}_{\mathbf{v}^*(t)\mathbf{v}^*(t)} - \mathbf{\Sigma}_{\mathbf{v}^*(t)\mathbf{V}(t)}\big(\mathbf{\Sigma}_{\mathbf{V}(t)\mathbf{V}(t)} + \sigma^2(t)\mathbf{I}_m\big)^{-1}\mathbf{\Sigma}_{\mathbf{V}(t)\mathbf{v}^*(t)}\Big), \tag{3.22}$$

where $\mathbf{v}^*(t) = [\mathbf{x}_*^T, \boldsymbol{\theta}^{*^T}(t), \boldsymbol{\psi}^{p^T}(t)]^T$, $\mathbf{V}(t) = [\mathbf{X}^P, \boldsymbol{\Theta}^P(t), \mathbb{1}_{m\times d_\psi}\text{diag}(\boldsymbol{\psi}^p(t))]^T$, and the covariance matrices are calculated by the t^{th} sample of the covariance parameters, i.e., $\boldsymbol{\gamma}(t), \alpha(t)$. Note that the trasformation $\mathbb{1}_{m\times d_\psi}\text{diag}(\boldsymbol{\psi}^p(t))$ uses the $m \times d_\psi$ matrix of ones, $\mathbb{1}_{m\times d_\psi}$, to construct a $m \times d_\psi$ matrix whose rows are $\boldsymbol{\psi}^{p^T}(t)$.

Finally, we derive our prediction using predictive distribution (3.22),

$$\hat{\mu}_* = \frac{1}{T}\sum_{t=1}^{T}\left(\boldsymbol{\Sigma}_{\mathbf{v}^*(t)\mathbf{V}(t)}\big(\boldsymbol{\Sigma}_{\mathbf{V}(t)\mathbf{V}(t)} + \sigma^2(t)\mathbf{I}_m\big)^{-1}\mathbf{y}^p\right),$$

$$\hat{\sigma}_*^2 = \frac{1}{T^2}\sum_{t=1}^{T}\left(\boldsymbol{\Sigma}_{\mathbf{v}^*(t)\mathbf{v}^*(t)} - \boldsymbol{\Sigma}_{\mathbf{v}^*(t)\mathbf{V}(t)}\big(\boldsymbol{\Sigma}_{\mathbf{V}(t)\mathbf{V}(t)} + \sigma^2(t)\mathbf{I}_m\big)^{-1}\boldsymbol{\Sigma}_{\mathbf{V}(t)\mathbf{v}^*(t)}\right).$$

3.8 Experimental results

We evaluate the performance of the proposed model by testing it on two synthetic and two real calibration problems and compare the results with the competing local calibration models, i.e., Non-parametric Functional Calibration (NFC) [84], Parametric Functional Calibration (PFC) [81], and Non-parametric Bayesian Calibration (NBC) [10]. We also refer to our model as Bayesian Non-isometric Matching Calibration (BMNC), and summarize the properties of all the four models in Table 3.1:

Model	Main method	Confidence interval	Surrogate modeling
NFC	Risk minimization in Reproducing Kernel Hilbert Spaces (RKHS)	No	Yes
PFC	Risk minimization in Reproducing Kernel Hilbert Spaces (RKHS)	No	Yes
NBC	Bayesian inference and GPs	Yes	Yes
BMNC	Bayesian inference and GMST	Yes	No

Table 3.1: Properties of different local calibration models

Moreover, for evaluating the accuracy of predictions, we use the Root Mean

Squared Error (RMSE),

$$\text{RMSE} = \sqrt{\frac{1}{c}\sum_{i=1}^{c}(y_*^i - \hat{\mu}_*^i)^2} \tag{3.23}$$

where y_*^i is the true response for a given $\mathbf{x}_*^i$ and $\hat{\mu}_*^i$ is its predicted value.

3.8.1 Synthetic problems

The first synthetic problem we examine has two control variables and one local calibration variable. We define

$$\mathcal{F}^s(\mathbf{x}, \theta) = 0.4(x_1^2 + x_2^2)\sin^2(0.7x_2)\frac{x_1 + x_2}{\theta^2 + 1}$$

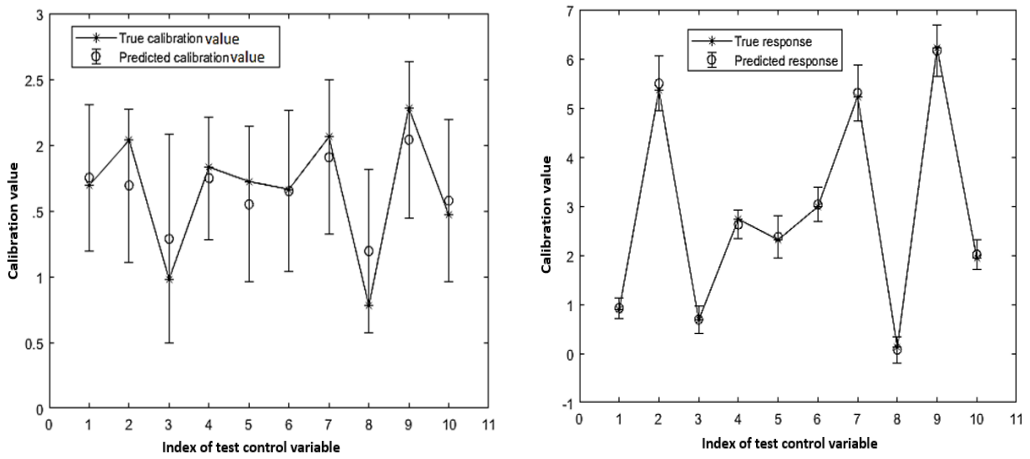
and

$$\mathcal{F}^p(\mathbf{x}) = 0.4(x_1^2 + x_2^2)\sin^2(0.7x_2),$$

where $\mathcal{F}^\theta(\mathbf{x}) = (x_1 + x_2 - 1)^{0.5}$. We locate $m = 16$ control vectors, i.e., $\mathbf{x}_i^p$, uniformly on the square $[0, 3.5] \times [0, 3.5]$. Then for each $\mathbf{x}_i^p$, we sample 10 calibration variables randomly from the interval $[0, 5]$; therefore, we have total of $n = 160$ computational data points. Finally, we sample 10 random $\mathbf{x}_*$ from the same square $[0, 3.5] \times [0, 3.5]$ to form a test dataset. We also set $\lambda = 0.5$ and $r = 0.9$.

Figure 3.5 shows the 95% confidence interval predictions for the responses and the calibration variables for the test dataset of the first synthetic problem. Since $\mathbf{x}^p \in \mathbb{R}^2$, we plot the predicted values against their indices, and connect the data points to each other for better a demonstration in Figure 3.5.

As noted in Section 3.2, due to the limited number of samples we have from the computational model, we cannot accurately recover $\mathcal{F}^\theta$, but the way we train the hyper-parameters of the GP compensates for this inaccuracy. We can observe this fact in Figure 3.5, where prediction of responses have more accuracy and tighter confidence intervals comparing to those of calibration variables.



(a) (b)

Figure 3.5: 95% confidence interval predictions for calibration variables and responses of the test dataset of the first synthetic problem

Finally, Table 3.2 compares the accuracy of our results with the other competing models, and shows that BNMC obtains the best prediction accuracy. Also note that overall, the Bayesian-based models, i.e., NBC and BNMC, perform better than RKHS based models.

Model	RMSE
NFC	0.2387
PFC	0.4122
NBC	0.1317
BNMC	0.0756

Table 3.2: RMSE of different models for the first synthetic problem

The second synthetic problem was originally used in [10] and has one control, one local calibration, and one global calibration variables. The computational and the physical models are defined as

$$\mathcal{F}^s(x, \theta, \psi) = \theta + \psi x^2,$$

and

$$\mathcal{F}^p(x) = 2\sqrt{x} + 2.5x^2.$$

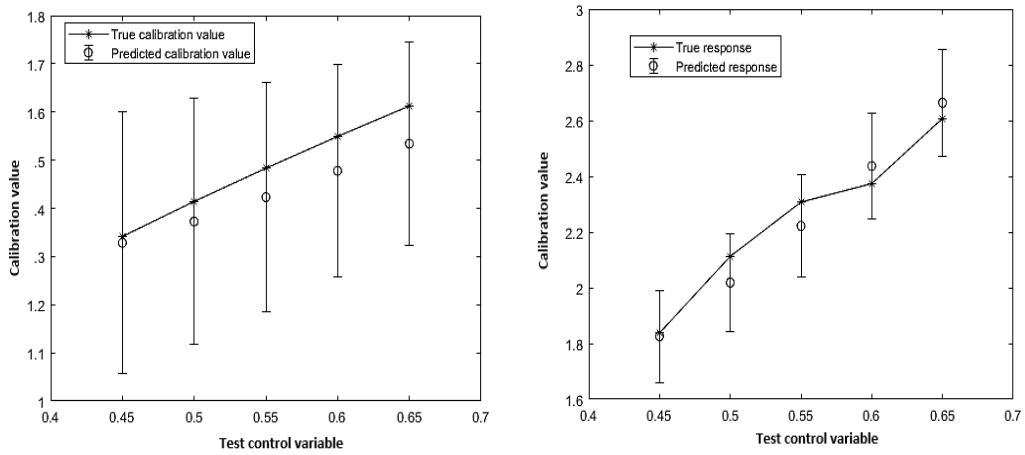
Therefore, the true calibration variables are $\psi = 2.5$ and $\mathcal{F}^\theta(x) = 2\sqrt{x}$. We locate $m = 15$, and five physical control variables for training and testing at locations $\{0, 0.05, 0, 1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$ and $\{0.45, 0.50, 0.55, 0.60, 0.65\}$, respectively. Following the same procedure as the previous example, we sample 10 calibration vectors for each physical control variable randomly from the square $[0, 5] \times [0, 5]$; therefore, we have total of $n = 150$ computational data points. We also set $\lambda = 1$ and $r = 0.16$.

Figure 3.6 shows the 95% confidence interval predictions for the responses and the calibration variables for the test dataset of the second synthetic problem. Observe that similar to the previous example, confidence intervals are tighter for the responses compared to those of calibration variables.

For the second synthetic problem, we just compare the results of NBC and BNMC, since NFC and PFC only apply to univariate calibration variables. Table 3.3 shows that BNMC outperforms NBC in terms of RMSE. Note that the reported RMSE in [10] for this problem under the cheap computational code assumption, which we refer to as NBC(cheap), has a better accuracy than BNMC, however, here BNMC is superior when NBC uses surrogate modeling.

Model	RMSE
NBC	0.1732
NBC(cheap)	0.0538
BNMC	0.0631

Table 3.3: RMSE of different models for the second synthetic problem



(a) (b)

Figure 3.6: 95% confidence interval predictions for calibration variables and responses of the test dataset of the second synthetic problem

3.8.2 Real problems

The first real problem, PVA, has one control variable and one calibration variable [81] . We use $m = 11$ physical and $n = 150$ computational data points, and use a 4-fold cross validation to form the training and test datasets. Also, we set $\lambda = 0.5$ and choose $r = 0.5$.

Table 3.4 shows the results of applying all the local calibration models to the PVA problem, where BNMC and NBC perform better than the RKHS-based models.

Model	RMSE
NFC	0.379
PFC	0.450
NBC	0.281
BNMC	0.296

Table 3.4: RMSE of different models for the PVA problem

Moreover, since we do not know the true calibration values, in Figure 3.7 we only

show the 95% confidence interval predictions for the responses and compare them with the true values of responses.

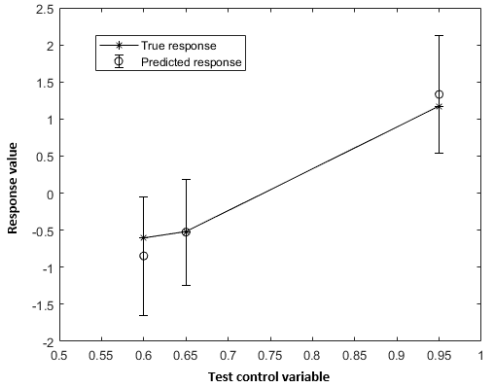


Figure 3.7: 95% confidence interval predictions for the test responses of the PVA problem

Spot welding is the second real problem originally used in [84] with three control variables and one calibration variable. This dataset contains $n = 35$ computational and $m = 12$ physical data points. We use 4-fold cross validation to form the training and test datasets. Also, we set $\lambda = 2$ and $r = 4$

Table 3.5 shows the prediction accuracy of all the models, where BNMC outperforms all the competing models. In fact, since BNMC is designed to handle expensive computational models with small number of computational data points, it performs much better on the small spot welding training dataset in comparison to the competing models.

Model	RMSE
NFC	0.683
PFC	1.115
NBC	0.516
BNMC	0.409

Table 3.5: RMSE of different models for the spot welding problem

Finally, similar to PVA, since we do not know the true calibration values, we only illustrate the 95% confidence interval predictions for the responses and compare them with the true response values in Figure 3.8. Note that similar to the first synthetic problem, we plot the prediction values against their indices, due to the dimensionality of $\mathbf{x}^p$.

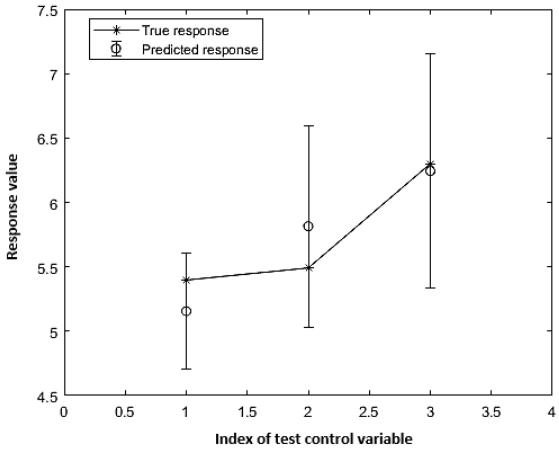


Figure 3.8: 95% confidence interval predictions for the test responses of the spot welding problem

3.9 Summary

We proposed a Bayesian Non-isometric Matching Calibration model for calibration of expensive computational models. We addressed the restriction of having a limited budget to evaluate computational models by replacing the computational function by a GP, which was trained during the calibration procedure. We used a Bayesian framework to simultaneously train the hyper-parameters of the GP's covariance function and make inferences on the calibration variables associated with the physical data points. To construct informative prior distributions for BNMC we used a geometric interpretation of calibration based on non-isometric curve to surface matching. This point of view enabled us to use graph-based approaches to address the problem of finding a set of anchor points used in constructing informative prior distributions. For

a special case, i.e., single control and calibration variables, we introduced the shortest path model on a directed acyclic calibration graph to tackle the problem of finding anchor points, while for the general case, we introduced the GMST model. Our numerical experiments conducted on four calibration problems showed that BNMC outperformed the existing calibration models under the assumption of expensive computational models.

CHAPTER 4

A SIMILARITY-BASED FORCASTING MODEL FOR SPATIO-TEMPORAL SYSTEMS WITH A FOCUS ON SHORT-TERM WIND SPEED FORECASTING IN WIND FARMS

4.1 Introduction

Wind energy has been hailed as a promising source of renewable energy. The increasing consumption of wind energy over the past two decades in developed countries asserts the significance of this source (`http://energy.gov/eere/wind/wind-program`). Although advances in technology have enabled wind farms to harness wind power extensively, a significant potential exists to fully utilize the available wind power. One challenge for utilizing this potential is caused by a lack of cost effective technologies capable of saving the wind energy for future demands. Hence, as wind blows, we need to convert it to wind energy and flow it to power grids. However, real time integration of wind farms into power grids has been hindered by the volatility of wind, which in turn results in a high degree of uncertainty for power generated by wind farms. This uncertainty can exert additional costs on power grids by failing to meet the power demand, through underestimating or overestimating the available wind power. Hence, wind power systems will greatly benefit from models that can accurately forecast the available wind power, specifically, in short intervals.

Physical and data-driven modeling are two main approaches for wind power forecasting. Physical approaches use physical specifications of wind farms and surrounding areas to build computational fluid dynamic (CFD) models based on physical principles [97, 73]. These CFD models are then used to simulate wind farms and

forecast the wind speed and the wind power in future. On the other hand data-driven models use historical wind speed, wind power, and other weather conditions data to build statistical forecasting models [14].

One key element in both physical and data-driven wind power forecasting in short intervals is forecasting the wind speed. There is a rich body of literature in statistical short-term wind speed forecasting, which can be generally categorized into two classes of autoregressive-based and kernel-based models. In autoregressive-based models, conventional time series analysis techniques, i.e., techniques developed by autoregrssion, are utilized [11, 103, 68]. These models have been improved by incorporating spatial information to predictive models using vector autoregressive approaches [80, 119]. The state-of-the-art autoregressive-based models, Regime-switching Space-Time Diurnal (RSTD)[30], Trigonometric Direction Diurnal (TDD), and Bivariate Skew-T (BST) [40], are used in this chapter as the competing models. The second class of models, kernel-based models, use supervised machine learning approaches to improve the forecasting accuracy [62, 54, 21, 52]. These models first convert the time stamped observations to regression type datasets and then apply modified versions of kernel-based learning algorithms for wind speed forecasting.

In this study, we take a similarity-based approach to tackle the problem of short-term wind speed forecasting. As mentioned in Section 1.1.2, similarity-based models are a group of non-parametric linear smoothers introduced and fully axiomatized in the field of economics [29]. These models make prediction by taking weighted average of historical observations based on the *similarity* of the input variables. However, in our similarity-based forecasting model, we redefine the notion of similarity as degree of resemblance between two sequences of consecutive observations, which we call *pattern vectors*, in two periods of time. Under the assumption that future observations of similar patterns resemble each other, we use an average of historical observations weighted based on the similarity of their pattern vectors to forecast the future obser-

vation.

Similarity-based models use *similarity functions* to assign weights to observations in the weighted averaging procedure. These functions are in fact biargumental, symmetric, and positive valued functions that take two input vectors and measure the similarity between them by a positive value. Similarity functions can take any reasonable form under the axioms stated in [53]. Following [29], we use the *empirical similarity function*, which is in fact a version of the squared exponential kernel function (1.3), to non-linearly search for similar pattern vectors and weight observations.

Moreover, our similarity-based forecasting model allows the pattern vectors to be more complex than just what was defined. In fact, in addition to prior consecutive observations at each time, these vectors can contain any necessary explanatory variables such as spatial information in other locations and temporal component. Therefore, we use a variable selection procedure to select efficient variables for the pattern vectors.

The proposed model can also be viewed as an autoregressive approach whose order is equal to number of historical observations, and the parameters are estimated by the similarity of the patterns. However, a fundamental difference between these two approaches is the following: autoregressive models assume only recent observations suffice to forecast the future, but in the similarity-based approach, observations in a distant past may be very informative for prediction due to the similarity in patterns. Note that this interpretation does not mean that we disregard time dependence of observations in our model, since the variable selection procedure gives us the flexibility to add any necessary temporal components to pattern vectors.

The idea of similarity-based forecasting for short-term wind forecasting was initially proposed in [91] using a Bayesian paradigm to estimate the magnitude parameters. However, the use of Markov Chain Monte Carlo simulation techniques make this approach computationally inefficient. Here, we take a frequentist approach (akin to that of [117]) and estimate the parameters of the model using the method of Least

Squared Errors, which is computationally much more efficient. Moreover, the proposed model is equipped with a variable selection procedure which makes it flexible to be easily applied to other spatio-temporal systems.

We apply our model to the dataset used in [30] and [40] to forecast 2-hours-ahead wind speed. The numerical results show a great improvement in forecasting accuracy compared with competing vector autoregressive-based models, i.e., TDD, BST, and RSTD.

The remainder of this chapter is organized as follows. Section 4.2 discusses what makes the similarity-based models suitable for spatio-temporal analysis. Section 4.3 briefly explains the U.S. pacific northwest wind farm dataset used in this study. Section 4.4 explains our similarity-based forecasting model including the parameter training and variable selection procedures. Section 4.5 illustrates the experimental results, and Section 4.6 summarizes the chapter.

4.2 Why similarity-based models suit analysis of spatio-temporal systems better than the kernel-based models

Despite the relationship between the similarity-based models and the kernel-based models, specifically GPR and NWR, in the class of nonparametric linear smoothers discussed in Section 1.1.2, here we point out two fundamental differences that make the similarity-based model more suitable for spatio-temporal forecasting than the kernel-based models.

First, kernel-based models assume that an underlying functional relationship between inputs and outputs exists, whereas similarity function used in the simlarity-based model is itself a part of the data generating process, which has been rigorously justified by the axioms of [29]. The latter is in fact a more realistic assumption in time series analysis, since there might be no valid functional relation between previous and future observations in a stochastic process changing over time.

Second, parameters in kernel functions have statistical interpretations that determine the behavior of the underlying function being estimated. For instance, in squared exponential kernel (1.3), parameter γ determine smoothness of the underlying function. On the other hand, the values of magnitude parameters in the empirical similarity function represent the extent of contribution of each variable in searching for similarities. In other words, similar to parametric regression models [59], magnitude parameters signify the importance of the variables used in modeling; therefore, these parameters allow us to rank and select significant variables as well as carry out hypothesis testing.

4.3 U.S. pacific northwest wind farm dataset

In this study, we use the same meteorological dataset used and fully explained in [30]. This dataset contains wind speed and wind direction measured every 10 minute at three meteorological towers along the Columbia river in the U.S. Pacific Northwest: Vansycle, Kennewick and Goodnoe Hills, during a period of nine months from March 1, 2003 to November 30, 2003. In addition to these two pieces of information for each location, we add three more attributes to the dataset by computing the hourly average of wind speed using the 10-minute measurements for each meteorological tower, and our goal is to predict the hourly average of wind speed in 2-hours-ahead at Vansycle. In Section 4.5, we apply our model explained in Section 4.4 to this dataset and compare the results with competing methods.

4.4 Similarity-based model for forecasting

Given a training dataset containing N observations, $\mathbf{D} = \{(\mathbf{x}_i, y_i) \mid i \in [N], \mathbf{x}_i \in \mathbb{R}^p, y \in \mathbb{R}\}$, authors of [29] proposed and fully axiomatized a similarity-based prediction model, wherein for a new input vector $\mathbf{x}_*$, the corresponding y_* can be predicted by taking the weighted average of the training data points weighted by a similarity

function,

$$\widehat{y}_* = \frac{\sum_{i=1}^{n} \mathcal{S}(\mathbf{x}_i, \mathbf{x}_*) y_i}{\sum_{i=1}^{n} \mathcal{S}(\mathbf{x}_i, \mathbf{x}_*)}, \tag{4.1}$$

where $\mathcal{S} : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is an *empirical similarity function* that measures the closeness between two observations in terms of their input vectors. Empirical similarity function can take any reasonable form under a few conditions stated in [53]. Following [29], we choose the anti-isotropic squared exponential function used in Chapters 2 and 3 as our similarity function,

$$\mathcal{S}(\mathbf{x}, \mathbf{x}') = \exp\Big(-\sum_{r=1}^{p} w_r (x_r - x'_r)^2 \Big). \tag{4.2}$$

In the context of similarly-based modeling, we distinguish the parameters of the empirical similarity function from the length-scale parameters in anti-isotropic squared exponential function, by denoting them as *magnitude parameters*, $\mathbf{w} = [w_1, \ldots, w_p]^T$ (See Section 4.4.1 for further discussion on the magnitude parameters).

We modify the idea of similarity-based prediction to use it for forecasting by introducing pattern vectors as follows: Given a *time series data*, which is a sequence of observations of a system indexed by time labels $[t]$, i.e., $\{z_i^0 \mid i \in [t]\}$, we preliminarily define a *pattern vector* corresponding to time i as a vector of l consecutive observations ending at i, i.e, $[z_{i-l+1}^0, \ldots, z_i^0]$. Subsequently, we define *similarity* as the rate of resemblance between two pattern vectors corresponding to time i and j that is measured by similarity function (4.2), $\mathcal{S}([z_{i-l+1}^0, \ldots, z_i^0]^T, [z_{j-l+1}^0, \ldots, z_j^0]^T)$.

In fact, pattern vectors indicate most recent trend of observations up to a specific time, so it is reasonable to assume that if two pattern vectors resemble each other, then their immediate futures are similar. Based on this assumption, to forecast the future observation in q time units ahead at time t, i.e., z_{t+q}^0, we find the patterns similar to the most recent pattern that is $[z_{t-l+1}^0, \ldots, z_t^0]^T$, and use q time units ahead observations of the similar patterns to forecast z_{t+q}^0. In other words, in this approach, we associate each observation z_i^0 with a unique pattern vector ending at q time units

before i, i.e., $[z^0_{i-q-l+1}, \ldots, z^0_{i-q}]^T$; subsequently, the weighted average (4.1) is modified to

$$\hat{z}^0_{t+q} = \frac{\sum_{i=l+q}^{t} \mathcal{S}([z^0_{i-l-q+1}, \ldots, z^0_{i-q}]^T, [z^0_{t-l+1}, \ldots, z^0_t]^T) z^0_i}{\sum_{i=l+q}^{t} \mathcal{S}([z^0_{i-l-q+1}, \ldots, z^0_{i-q}]^T, [z^0_{t-l+1}, \ldots, z^0_t]^T)}. \tag{4.3}$$

Moreover, to take the time dependence between the observations into account, we expand the pattern vector corresponding to time i by adding temporal components, such as time index, periodical components, etc., as $[z^0_{i-l+1}, \ldots, z^0_i, \mathcal{T}^T_{i-l+1}, \ldots, \mathcal{T}^T_i]^T$, where each $\mathcal{T}_i$ is a vector of temporal component belonging to time i.

The pattern vectors can be further augmented in the presence of additional variables. For instance, if the spatio-temporal dataset, $\{\mathbf{z}_i \mid i \in [t]\}$, is available, where each $\mathbf{z}_i$ is a vector of observations at time i in multiple locations, i.e., $\mathbf{z}_i = [z^0_i, z^1_i, ..., z^r_i]^T$, we can replace the single observation z^0_i by the vector of observation $\mathbf{z}_i$ in the pattern vector. We call the pattern vectors containing all the temporal and additional variables *initial pattern vector* and denote it by $\mathbf{x}_i$,

$$\mathbf{x}_i = [\mathbf{z}^T_{i-l+1}, \ldots, \mathbf{z}^T_i, \mathcal{T}^T_{i-l+1}, \ldots, \mathcal{T}^T_i]^T. \tag{4.4}$$

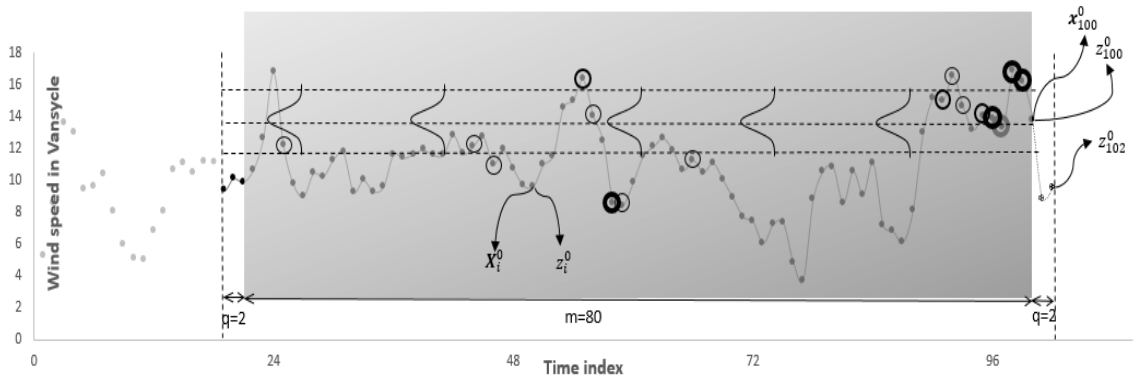
Considering all the variables in (4.4) can result in large pattern vectors causing lengthy training time (see discussed Section 4.4.1), thus, we select an effective subset of variables among the variables included in the initial pattern vector through a variable selection procedure (see Section 4.4.2). We call the vector of the selected variables as *truncated pattern vector* and denote it as $\mathbf{x}^0_i$.

Finally, since the observations have a chronological order, we can use a search length parameter m to search for similar patterns in the last m observations, as such equation (4.3) is modified to

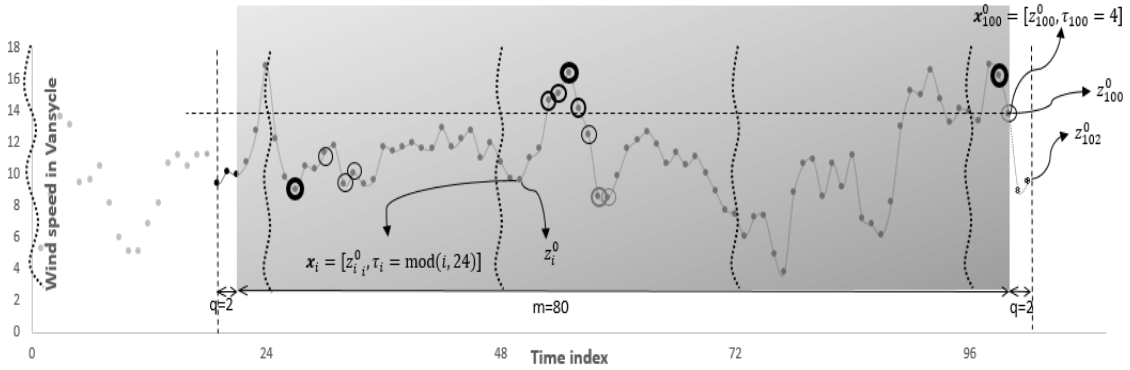
$$\hat{z}^0_{t+q}(m) = \frac{\sum_{i=t-m+1}^{t} \mathcal{S}(\mathbf{x}^0_{i-q}, \mathbf{x}^0_t) z^0_i}{\sum_{i=t-m+1}^{t} \mathcal{S}(\mathbf{x}^0_{i-q}, \mathbf{x}^0_t)}. \tag{4.5}$$

Figures 4.1a and 4.1b illustrate two simple scenarios based on a univariate time series from the dataset discussed in Section 4.3. In both figures, we set $q = 2$, $m = 80$, $t = 100$, so the target is z^0_{102}. Circles in each of the figures show those z^0_is whose assocciated patterns are close to the last pattern $\mathbf{x}^0_{100}$, and the thickness of each circle indicates the weight assigned to each data point. For Figure 4.1a, we construct $\mathbf{x}^0_i = z^0_i$ which is a univariate pattern; subsequently, the empirical similarity function with an arbitrary w can be superimposed on the center line going through the last pattern $\mathbf{x}^0_{100}$, so those data points whose patterns are closer to the center line gain higher weights, and those outside the two threshold lines are assigned almost zero weights. Note that the choice of w determines the width of weighting area. Moreover, for Figure 4.1b, we add a diurnal component with the period of 24, which is the inherent diurnal period for this time series, to the pattern vectors, i.e., $\mathbf{x}^0_i = [z^0_i, \mathcal{T}_i = \text{mod}(i, 24)]^T$, and assign arbitrary but relatively equal magnitudes to each of the dimensions, so that both the dimensions have equal impact on the pattern searching process.

We can observe that selected z^0_is and their assigned weights are different in the two figures. To determine which pattern vector is more effective and what the optimal values of the magnitude parameters are for each model, we need to devise procedures for variable selection and training parameters which will be discussed in the next two section, respectively.



(a) First Scenario: $\mathbf{x}_i^0 = z_i^0$. The height of the bell curve indicates the weight assigned to each data point by the empirical similarity function with an arbitrary magnitude w. The weights assigned to the data points outside the threshold lines are almost zero.



(b) Second scenario: $\mathbf{x}_i = [z_i^0, \mathcal{T}_i = \mathrm{mod}(i, 24)]^T$. Curvy lines indicate the beginning of each diurnal period. Magnitude of both of the dimensions are selected arbitrarily but relatively equal, so both of the dimensions in the pattern vectors have equal impact on the weighting process

Figure 4.1: Illustration of searching and weighting similar historical observation using two simple scenarios based on a univariate time series data with the following settings: $q = 2$, $m = 80$, $t = 100$, and target z_{102}^0. Thickness of the circles indicates the weights assigned to each observation

4.4.1 Training the magnitude parameters

Magnitude parameters show the relative significance of the variables included in the pattern vectors. A relatively large value of a w_r magnifies the weighted Euclidean distance in the exponent of (4.2) even with a small deviation of the variable associated with w_r. This large value of the weighted Euclidean distance rapidly takes the similarity value to zero regardless of proximity of other variables. In other words, the variables with relatively small magnitude parameters become important given the variables with larger magnitude parameters are close enough. Consequently, Magnitude parameters play key roles in finding the real similar observations, and should be trained prior to forecasting.

For accurate forecasting of z^0_{t+q} at time t, we train the model based on the n last observations prior to t, i.e., $\{z^0_{t-n+1}, \ldots, z^0_t\}$, by using the method of Least Squared Errors (LSE). In this method, we minimize the sum of squared errors in forecasting of n last observations, with a constant search length period parameter (m), and with respect to magnitude vector $\mathbf{w}$. We first define the sum of squared errors function as

$$\mathrm{SSE}(t, m, n, \mathbf{w}) = \sum_{j=t-n+1}^{t} (\hat{z}^0_j(m) - z^0_j)^2, \tag{4.6}$$

where z^0_j is the actual observation at time j and $\hat{z}^0_j(m)$ is the forecasted value of this observation with respect to the magnitude vector $\mathbf{w}$,

$$\hat{z}^0_j(m) = \frac{\sum_{i=t-m-q+1}^{j-q} \mathcal{S}(\mathbf{x}^0_{i-q}, \mathbf{x}^0_{j-q}) z^0_i}{\sum_{i=t-m-q+1}^{j-q} \mathcal{S}(\mathbf{x}^0_{i-q}, \mathbf{x}^0_{j-q})}. \tag{4.7}$$

Consequently, we define the following LSE optimization problem,

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathrm{SSE}(t, m, n, \mathbf{w}). \tag{4.8}$$

Optimization problem (4.8) can result in negative optimal weights which is meaningless in terms of the similarity between observations. These negative optimal

weights can occur either by collinearity between variables, i.e., redundancy, or irrelevancy of some variables to the response variable z_j^0. In both cases including such variables to the model causes not only difficulty to interpret the model, but also overfitting. For instance, in the wind speed dataset, the last 10-minute and the last hourly average wind speed are highly collinear, where the R-square statistics is about 95%; therefore, if both variables enter the pattern vector, one of them obtains a negative magnitude in the optimization procedure. Consequently, to prevent the negative weights problem, we force the model to assign a positive value to each w_r by adding the following constraint to (4.8),

$$\begin{aligned} \min_{\mathbf{w}} \quad & \text{SSE}(t, m, n, \mathbf{w}) \\ \text{s.t.} \quad & \mathbf{w} \geq \mathbf{0}, \end{aligned} \tag{4.9}$$

where $\mathbf{w} \geq \mathbf{0}$ means every components of $\mathbf{w}$ should be non-negative.

There are several ways to solve constrained optimization problem (4.9). For instance, one can use projected gradient descent [66] to find the optimal solution. However, here we convert problem (4.9) to an unconstrained optimization by introducing a penalty term [5]. In this method we relax the constraints by adding a penalty term to the objective function that penalizes the objective for negative values of the magnitude parameters. By this transformation, we can use unconstrained optimization techniques to obtain the optimal values of the magnitude parameters in a time efficient manner. The penalty term used here has the exponential form $\sum_{r=1}^{|\mathbf{x}^0|} \exp(-\lambda w_r)$, where λ is a sufficiently large positive number, and $|\mathbf{x}^0|$ is the length of the pattern vectors. This penalty function assumes large values for negative values of w_r, and therefore, increases the objective value, whereas positive values of w_r does not affect the objective (4.6), since the penalty term drops rapidly to zero. Therefore, we minimize the following penalized function instead,

$$\text{PSSE}(t, n, m, \mathbf{w}, \lambda) = \text{SSE}(t, n, m, \mathbf{w}) + \sum_{r=1}^{|\mathbf{x}^0|} \exp(-\lambda w_r). \tag{4.10}$$

To minimize the function (4.10) we use the method of gradient descent, which is an iterative numerical optimization procedure [56]. In this technique, starting from an arbitrary point in the domain of the function, we gradually reach a local optimum by moving along the negative gradient vector at each step. Therefore, calculating the partial derivatives at each point is necessary.

$$\frac{\partial}{\partial\, w_r}(\text{PSSE}(t, n, m, \mathbf{w}, \lambda)) =$$
$$-\lambda \exp(-\lambda w_r) + 2 \sum_{j=t-n+1}^{t} (\hat{z}_j^0(\mathbf{w}, m) - z_j^0) \frac{\partial}{\partial\, w^r} \hat{z}_j^0(\mathbf{w}, m),$$

where

$$\frac{\partial}{\partial\, w_r} \hat{z}_j^0(\mathbf{w}, m) =$$
$$\frac{\sum_{i=t-m-q+1}^{j-q} \mathcal{S}_{ij}^r z_i^0 \sum_{i=t-m-q+1}^{j-q} \mathcal{S}_{ij} - \sum_{i=t-m-q+1}^{j-q} \mathcal{S}_{ij} z_i^0 \sum_{i=t-m-q+1}^{j-q} \mathcal{S}_{ij}^r}{(\sum_{i=t-m-q+1}^{j-q} \mathcal{S}_{ij})^2},$$

$$\mathcal{S}_{ij} = \mathcal{S}(\mathbf{x}_{i-q}^0, \mathbf{x}_{j-q}^0),$$

$$\mathcal{S}_{ij}^r = \mathcal{S}(\mathbf{x}_{i-q}^0, \mathbf{x}_{j-q}^0)(x_{i-q}^r - x_{j-q}^r)^2,$$

and x_{i-q}^r is the r^{th} element of vector $\mathbf{x}_{i-q}^0$.

4.4.2 Variable selection

Along with the negative magnitude parameters and over-fitting problems discussed in Section 4.4.1, presence of many variables in the similarity-based forecasting model increases the training time, since calculating each similarity weight is order of $\mathcal{O}(|\mathbf{x}^0|)$ so evaluating the SSE function in (4.6) takes asymptotically $\mathcal{O}(mn|\mathbf{x}^0|)$ for each iteration of the optimization procedure. Consequently, by shrinking the initial pattern vectors prior to running the training procedure, we can reduce the training time. In this section, we use a stepwise variable selection procedure to truncate the initial pattern vectors.

We follow the stepwise variable selection method used in ordinary least squares regression for our purpose. As such, if we denote the SSE of the full model with p_f number of parameters as SSE_f, and SSE of the nested reduced model with p_r number of parameters as PSSE_r, under the $\epsilon \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ assumption, SSE_f, SSE_r, and $(\text{SSE}_r - \text{SSE}_f)$ follow the chi-square distribution with $(n - p_f)$, $(n - p_r)$, and $(p_f - p_r)$ degrees of freedom, respectively [60]; consequently, $(\frac{\text{SSE}_r - \text{SSE}_f}{p_f - p_r})/(\frac{\text{SSE}_f}{p_f})$ follows the F distribution with $(p_f - p_r)$ and $(n - p_f)$ degrees of freedom. In the forward selection, if the calculated F statistic exceeds the predetermined F value, the extra variables in the full model are significant and can enter the model. Table 4.1 summarizes this procedure.

Name	SSE	df	MSE
Full model	SSE_f	$n - p_f$	$\text{MSE}_f = \frac{\text{SSE}_f}{n - p_f}$
Reduced model	SSE_r	$n - p_r$	$\text{MSE}_r = \frac{\text{SSE}_r}{n - p_r}$
Difference	$\text{SSE}_d = \text{SSE}_r - \text{SSE}_f$	$p_f - p_r$	$\text{MSE}_d = \frac{\text{SSE}_d}{(p_f - p_r)}$
F statistics		$(p_f - p_r, n - p_f)$	$\text{MSE}_d/\text{MSE}_f$

Table 4.1: Sum of squares analysis.

Authors of [3] showed that although the calculated sum of squares ratio does not follow the exact F distribution for non-linear models, we still can apply the same setting, since the effect of the non-linearity in parameters is negligible in the F ratio; therefore, Table 4.1 can be used in our non-linear model by replacing SSE by PSSE values.

In practice, we take the following steps for variable selection:

1. Consider the initial pattern vector containing all the potential variables, i.e., $\mathbf{x}_i = [\mathbf{z}_{i-l+1}^T, \ldots, \mathbf{z}_i^T, \mathcal{T}_{i-l+1}^T, \ldots, \mathcal{T}_i^T]^T$. This vector can be very large depending on the dimension of each, $\mathbf{z}_i$, the number of lags, l, and choice of the temporal

component, $\mathcal{T}$.

2. Decide on the type of the stepwise approach, i.e., forward, backward, or hybrid, and the level of significance, α, for allowing variable to enter or leave the pattern. Note that in the forward selection, $\mathbf{x}_i^0 = [\,]$, and in backward selection, $\mathbf{x}_i^0 = \mathbf{x}_i$, initially.

3. Calculate the PSSE_r and PSSE_f and the subsequent F statistics at each step.

4. Decide which variable(s) should leave or enter the model by comparing the F statistics and the F value determined in step 3.

5. Recalculate PSSE_r and PSSE_f with the updated patter vectors.

6. Continue until no variable can leave or enter the pattern.

We note that due to the dynamic nature of many spatio-temporal systems the importance of the variables changes over time. Therefore, these systems benefit from executing the variable selection procedure before each forecasting. However, the variable selection is a time consuming procedure, which makes this approach impractical for the systems in which forecasting in short intervals is required.

Moreover, in addition to the variable selection procedure, the exponential penalty term in the PSSE function (4.10) can be used for variable selection as well, since it sets the magnitude parameters of many of the potentially irrelevant and redundant variables, which could obtained negative values, exactly to zero. However, it is not reasonable to merely rely on the penalty function to find the significant variables, because we need to train a larger number of parameters which result in lengthy training times.

Consequently, based on the foregoing discussion, we suggest running the variable selection procedure to refresh the truncated pattern vectors on a daily, weakly or

monthly basis, depending on the system of interest, while using the penalty term to avoid the problem of negative magnitudes.

4.5 Experimental results

In this section, we apply our similarity-based forecasting model to the wind dataset explained in Section 4.3. Section 4.5.1 introduces the notations and the variables involved in the initial pattern vectors, Section 4.5.2 explains the numerical procedure of choosing proper values of the training length and the search length parameters, Section 4.5.3 discusses the variables selected for the pattern vectors through the variable selection procedure, and Section 4.5.4 compares the results with the competing models.

4.5.1 Notations

Let $\mathbf{z}_i$ denote the vector of observations during the i^{th} hour that contains the following information associated with each of the locations Vansycle (v), GoodnoeHills (g), and Kennewick (k):

- Hourly wind speed average: $A_i^v, A_i^k,$ and A_i^g, where $A_i^v = z_i^0$.
- Last 10-minute wind speed: $L_i^v, L_i^k,$ and L_i^g
- Hourly wind speed standard deviation : $\sigma_i^v, \sigma_i^k,$ and σ_i^g
- Trigonometric transformation of the last 10-minute wind direction: $D_i^v, D_i^k,$ and D_i^g, where each D_i is in fact a pair of variables. For instance, if we denote the last 10-minute wind direction at Vansycle as d_i^v, then $D_i^v = [\sin(d_i^v), \cos(d_i^v)]^T$
- Interaction between the last 10-minute of the wind speed and the wind direction: $LD_i^v, LD_i^k,$ and LD_i^g, where for instance, $LD_i^v = [L_i^v * \sin(d_i^v), L_i^v * \cos(d_i^v)]^T$

Note that we could include more variables to $\mathbf{z}_i$ such as rest of the 10-minute wind speeds and direction measurements or other interaction terms; however, these variables are highly correlated with the variables mentioned in the above list; therefore, adding them does not help to improve the performance of the model.

Moreover, as discussed in [30], a diurnal periodicity exists in the data. This diurnality can be illustrated by taking the average wind speed in Vanscycle during each hour of day over the nine months of measurement. Figure 4.2 shows that during afternoon and evening wind speed is lower than the night and morning on average. Note that in the absence of diurnality, this curve would be a flat line. This phenomena has been taken into account in the models developed in [30] and [40] through approximating a diurnal curve by a trigonometric transformation and removing it from the actual observations.

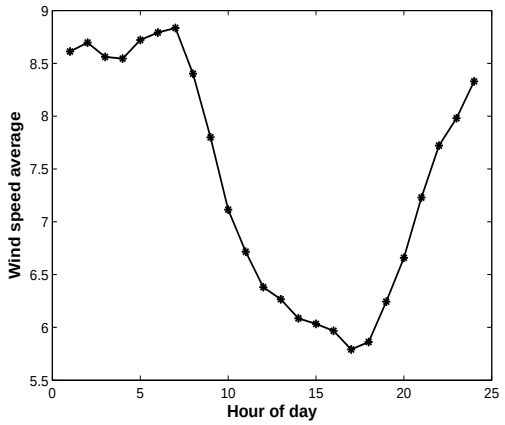


Figure 4.2: Average wind speed in Vanscycle during each hour of day over the nine months of measurement

However, here, we consider the diurnality of the data by adding a duirnal component to the pattern vectors. This diurnal component assigns the same diurnal value to observations at a same time of day. We use a trigonometric transformation of function $\mod(i, 24)$ to give the model the flexibility of having two parameters

associated with diurnality,

$$\mathcal{T}_i = [\sin(\frac{2\pi \bmod(i,24)}{24}), \cos(\frac{2\pi \bmod(i,24)}{24})]^T. \tag{4.11}$$

Finally, we construct the initial pattern vector $\mathbf{x}_i$ for 2-hours-ahead forecasting by considering only the last vector of observations $\mathbf{z}_i$ and the associated $\mathcal{T}_i$,

$$\mathbf{x}_i = [\mathbf{z}_i^T, \mathcal{T}_i^T]^T, \tag{4.12}$$

which contains 23 variables including 9 single variables $A_i^v, A_i^k, A_i^g, L_i^v, L_i^k, L_i^g, \sigma_i^v, \sigma_i^k, \sigma_i^g$ and seven pairs of variables $D_i^v, D_i^k, D_i^g, LD_i^v, LD_i^k, LD_i^g, \mathcal{T}_i$. In Section 4.5.3, we shrink this vector $\mathbf{x}_i$ to a smaller vector $\mathbf{x}_i^0$ with respect to the target variable A_i^v by the variable selection procedure introduced in Section 4.4.2.

4.5.2 Choices of m and n

The training length, n, and the search length, m, are the critical parameters of our similarity-based forecasting model that should be specified prior to variable selection and training the magnitude parameters. As mentioned in Section 4.4.2, evaluating the SSE function in (4.10) takes asymptotically $\mathcal{O}(mn|\mathbf{x}_i^0|)$ for each iteration of the optimization procedure; therefore, choosing large values for m and n causes lengthy training time, while small m and n may lead to a bias in estimation of the magnitude parameters.

To choose proper values for m and n we take an empirical approach and compare the forecasting power of the model with different values of m and n while keeping the other factors constant. To this end, we consider a constant pattern vector, $\mathbf{x}_i^0 = [L_i^v, L_i^k, L_i^g]^T$, and train the magnitude parameters with different values of m and n as explained in Section 4.4.1. Then, we make hourly forecasts for the last three months of the dataset, i.e., $91 * 24 = 2184$ forecasts, using the magnitude parameters trained for each pair of m and n . Different pairs of m and n obtain different forecasting

accuracy, which is measured by the Root Mean Squared Error,

$$\text{RMSE} = \sqrt{\frac{1}{2184} \sum_{i \in \text{last three months}} (A_i^v - \hat{A}_i^v)^2}. \tag{4.13}$$

Note that the reason why we choose the last three months of the dataset for this empirical procedure is to have enough historical observations to increase the values of m and n to large numbers. Figure 4.3 shows the RMSE values for different values of n, where each curve corresponds to a fixed m.

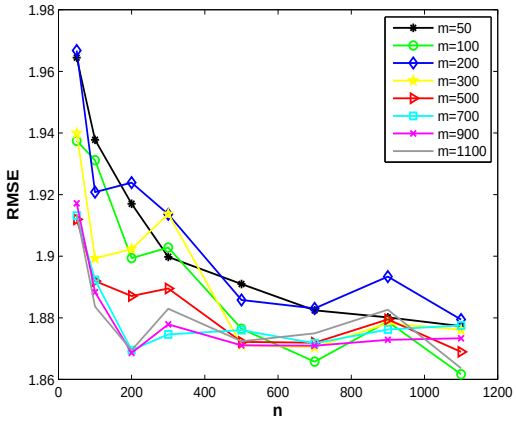


Figure 4.3: RMSE for different values of n during the last three months of the dataset using constant pattern vector $\mathbf{x}_i^0 = [L_i^v, L_i^k, L_i^g]^T$. Each curve corresponds to a fixed m.

Observe that for the large values of n, i.e, $n \geq 500$, all the curves converge to a constant RMSE, and also, the curves associated with larger m, i.e., $m \geq 500$, converge faster with less fluctuation; consequently, we set $n = m = 500$ for the variable selection procedure and training the magnitude parameters

4.5.3 Forward variable selection

As mentioned in Section 4.4.2, considering all the variables in (4.12) deteriorates the efficiency of the model; therefore, we use the first two months of the dataset, March

and April, which are not used in testing the performance in Section 4.5.4, for variable selection.

Here, we take the forward selection approach by applying the six steps procedure explained in Section 4.4.2 with level of significance $\alpha = 1\%$. The selected pattern vector obtained from this variable selection contains six variables (two single and two pairs of variables),

$$\mathbf{x}_i^0 = [L_i^v, A_i^g, LD_i^{k^T}, \mathcal{T}_i^T]^T. \tag{4.14}$$

Observe that in addition to the diurnal component $\mathcal{T}_i$, the variable selection procedure has selected one variable corresponding to each meteorological location, i.e., last 10-minute wind seed in Vansycle, L_i^v, hourly wind speed average in GoodnoeHills, A_i^g, and interaction between the last 10-minute wind speed and the wind direction in Kennewick, LD_i^k.

We note that in the variable selection procedure, we either let both of the variables in a pair enter the model or keep them out of the model. Consequently, a full model containing a component with a pair of variables has two degrees of freedom less than the nested reduced model.

4.5.4 Performance

To assess the forecasting power we apply our model to the last seven months of the dataset from May to November, 2003, which were disregarded in the variable selection procedure, and compare the results with the Persistence approach as the benchmark and the competing models, i.e., RSDT, TDD, and BST.

For each 2-hours-ahead wind speed forecasting during May to November, 2003, we train the magnitude parameters associated with the selected pattern vector (4.14) using the last $n = 500$ hours with $m = 500$ as discussed in Section 4.5.2. Also, we set $m = t - 2$ for forecasting, since calculating equation (4.5) is not computationally

expensive even with large values of m, whereas as m increases, more similar points can be found.

For measuring the prediction accuracy, we use Mean Absolute Error (MAE), $\frac{1}{c}\sum_{i=1}^{c}|A_i^v - \hat{A}_i^v|$, and Rout Mean Squared Error (RMSE),$\sqrt{\frac{1}{c}\sum_{i=1}^{c}(A_i^v - \hat{A}_i^v)^2}$.

Table 4.2 shows the values of RMSE and MAE for Persistence, RSDT, TDD, BST, and the proposed similarity-based forecasting model separated by month, in which the similarity-based forecasting model obtains more accurate results than the competing autoregressive-based models.

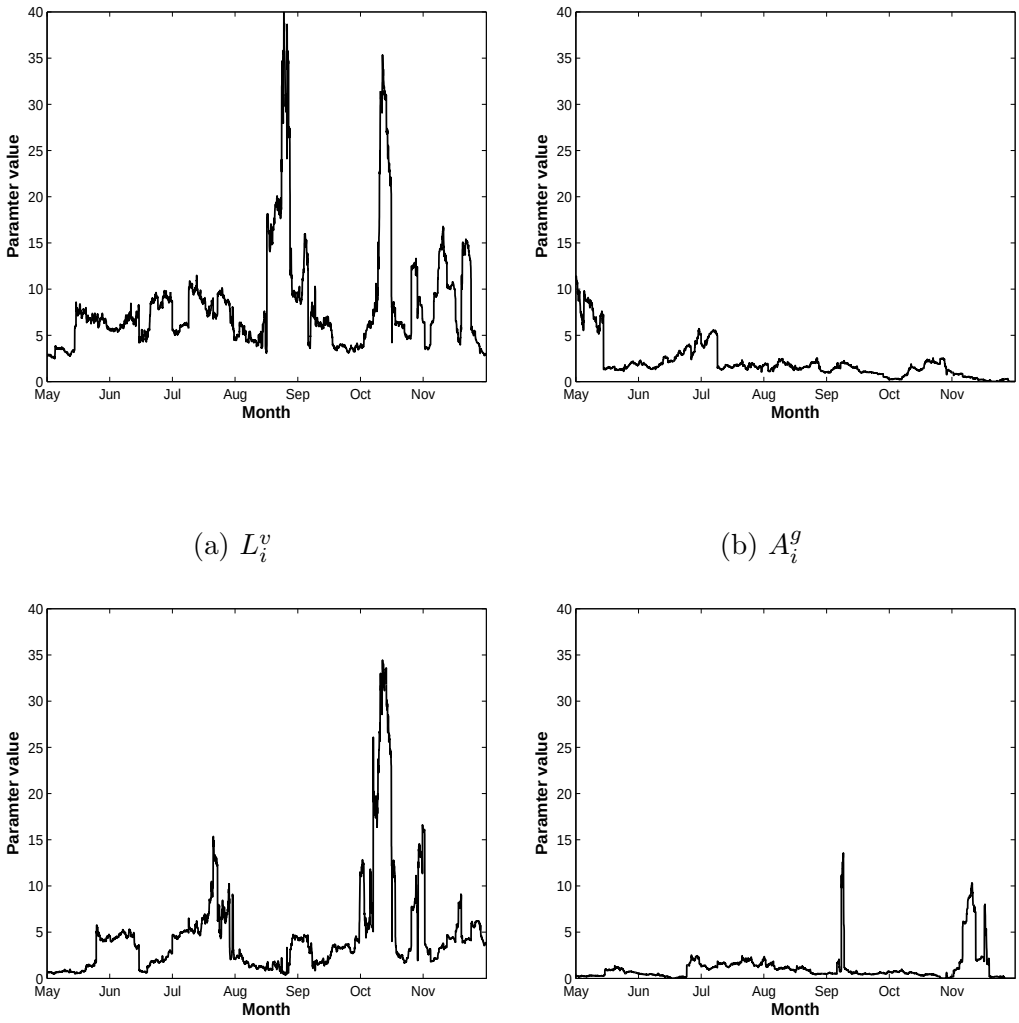
Measure	Model	May	Jun	Jul	Aug	Sep	Oct	Nov	Average
RMSE	Persistence	2.14	1.97	2.37	2.27	2.17	2.38	2.11	2.21
	RSTD	1.73	1.56	1.68	1.78	1.77	2.07	1.87	1.79
	TDD	1.74	1.56	1.68	1.78	1.75	2.03	1.86	1.78
	BST	1.69	1.59	1.64	1.81	1.85	2.09	2.00	1.82
	Similarity-based	**1.66**	**1.48**	**1.48**	**1.71**	**1.67**	**1.89**	**1.84**	**1.68**
MAE	Persistence	1.60	1.45	1.74	1.68	1.59	1.68	1.51	1.61
	RSTD	1.31	1.19	1.32	1.31	1.36	1.48	1.38	1.34
	TDD	1.34	1.18	1.31	1.33	1.33	1.48	1.38	1.34
	BST	1.26	1.19	1.27	1.37	1.42	1.51	1.50	1.36
	Similarity-based	**1.24**	**1.12**	**1.10**	**1.23**	**1.31**	**1.39**	**1.39**	**1.26**

Table 4.2: Accuracy of the different models for last seven months of the dataset

In addition, the importance of the variables involved in the pattern vectors vary over time due to the dynamic nature of the wind system. This is the reason for training the magnitude parameters prior to each hourly forecast to increase the forecasting power of our model. We visualize the trends of the magnitude parameter estimates during the testing period in Figure 4.4. Note that to make these values comparable we standardize each variable in the pattern vectors, and also for each of the variables LD_i^k and $\mathcal{T}_i$, we unify their associated magnitude parameters using the Euclidean

norm.

Moreover, to compare the importance of the variables we aggregate the data shown in Figure 4.4 by taking the monthly average of each of the parameters. Figure 4.5 shows that L_i^v is the most valuable variable and the significance of the A_i^g and LD_i^k decreases and increases, respectively, during the testing period. Note that although we do not apply the variable selection procedure dynamically during the testing period to ensure a fair comparison with the other competing models, Figure 4.5 indicates that the significance of some of the variables might not hold over these nine months. For instance, very small values of the parameter estimates of the variable A_i^g in the last month, i.e., November, signal that this variable can potentially be replaced by new variables if we run the variable selection procedure.



(a) L_i^v (b) A_i^g

(c) LD_i^K (d) $\mathcal{T}_i$

Figure 4.4: Magnitude parameter estimates from May to November 2003

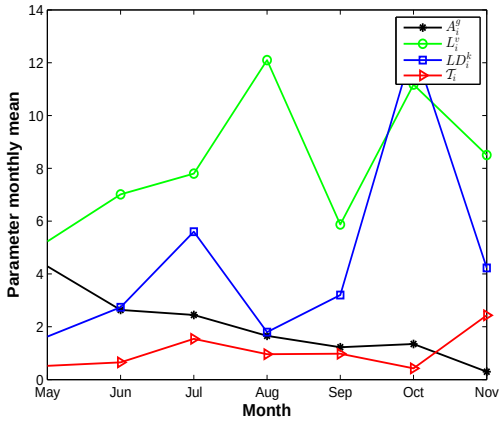


Figure 4.5: Monthly mean of the magnitude parameter estimates

Finally, to observe how the penalty function introduced in (4.4.2) selects the important variables without the stepwise variable selection procedure, we rerun our model once again using the initial pattern vector (4.12). Table 4.3 shows the results of this experiment for each month of the testing period. It can be observed that the average size of the pattern vectors in Table 4.3 is larger than the pattern vector (4.14). This means that although the penalty function shrinks the initial pattern vector (4.12) by 42.5% on average, it cannot perform as parsimoniously as the stepwise variable selection procedure. On the other hand, the accuracy of both the similarity-based forecasting models in Tables 4.2 and 4.3 is almost the same. Consequently, we prefer the model with fewer variables, i.e., pattern vector (4.14), due to its lower training time that is crucial to short-term wind speed forecasting. Moreover, Table 4.3 contains the five most important variables in each month, where L_i^v has always the highest value, while the rest of the variables vary from month to month.

		May	Jun	Jul	Aau	Sep	Oct	Nov	Average
Average size of the profile		8.7	8.5	10.3	8.9	11.4	9.4	6.8	9.1
RMSE		1.66	1.46	1.46	1.71	1.67	1.94	1.81	1.68
MAE		1.23	1.11	1.08	1.23	1.29	1.42	1.36	1.25
Important variables	1	L_i^v	L_i^v	L_i^v	L_i^v	L_i^v	L_i^v	L_i^v	
	2	L_i^g	A_i^g	L_i^k	A_i^g	L_i^k	L_i^k	LD_i^k	
	3	D_i^v	L_i^k	LD_i^v	$\mathcal{T}_i$	A_i^g	LD_i^v	D_i^k	
	4	L_i^k	LD_i^v	LD_i^k	LD_i^v	$\mathcal{T}_i$	D_i^k	LD_i^v	
	5	LD_i^v	$\mathcal{T}_i$	$\mathcal{T}_i$	LD_i^k	LD_i^k	LD_i^k	A_i^k	

Table 4.3: Performance of the Similarity-based forecasting model without the stepwise variable selection

4.6 Summary

We proposed a similarity-based forecasting model for spatio-temporal forecasting with the primary focus on wind speed forecasting in wind farms. We constructed the vectors of spatial and temporal information belonging to each time, which we called pattern vectors. Then we used a similarity function to weight the observations based on their associated pattern vectors and made forecasts by a weighted averaging technique. We discussed a constrained least squared error optimization approach to train the parameters of the model, and a variable selection procedure to construct parsimonious pattern vectors for the purpose of fast forecasting. Our numerical experiment conducted on a historical dataset collected from a wind farm showed that the proposed model outperformed commonly used forecasting methods.

CHAPTER 5

CONCLUSION

This dissertation discussed three well known problems in statistical modeling, namely, prediction, calibration, and forecasting, using GPs and their variations in the context of three engineering systems .

In Chapter 2, a Sparse Speudo-input Local Gaussian Process model was developed to simultaneously address the problems of scalability and covariance heterogeneity of GPR in dealing with spatial systems with exogenous variables. SPLGP used parallel hyperplanes to partition the domain of data into smaller subdomains, and then constructed independent local predictors in each subdomain. This partitioning policy resulted in a reduction of the number of boundaries and also simplified the boundary structures. However, this could lead to large size subdomains, which made the local application of the full GPR impractical. Therefore, a sparse approximation of the full GPR was applied to the subdomains. Moreover, independent training of the local predictors results in discontinuity of the global predictor on the boundaries. The small number of the boundaries and their simple structure enabled SPLGP to address this discontinuity by imposing connectivity constraints on the boundaries of the neighboring subdomains in a cost effective manner. SPLGP was also equipped with an optimization procedure capable of finding the best direction for the parallel hyperplanes. The optimization procedure was based on three theorems that developed an upper bound for covariance approximation error. SPLGP was applied to a spatial dataset with exogenous variables, two spatial datasets without exogenous variables, and a non-spatial dataset. The latter was used to demonstrate that the methodol-

ogy was general and was not restricted to spatial systems. The results showed that SPLGP maintained a good balance between prediction accuracy and computation time for data with stationary or non-stationary covariance structures. We list the major contributions of this study as follows:

- Proposed a local GPR model capable of maintaining the continuity of the global predictor for spatial datasets with exogenous variables in a time efficient manner.
- Addressed the discontinuity of the global predictor by transforming the GPR model to an optimization problem and solving it analytically.
- Proposed three theorems to characterize the best direction for the parallel hyperplanes based on the covariance structure of the domain of data.
- Developed an optimization procedure to find the best direction of applying parallel hyperplanes characterized by the theorems.

In Chapter 3, a Bayesian Non-isometric Matching Calibration for local calibration of expensive computational models was developed. The proposed calibration model replaced the computational function by a dynamic GP whose parameters were trained in the calibration producer, unlike conventional surrogate modeling techniques. BNMC utilized a Bayesian framework which used the constructed GP as its likelihood term to simultaneously make inferences on the parameters of the GP and the calibration parameters. Making inferences by the Bayesian framework was impractical without informative prior distributions on the calibration parameters. To construct informative prior distributions, the calibration problem was viewed as a non-isometric curve to surface problem. With this point of view, the concept of calibration graph was developed to find a set of computational data points which carried the information required for constructing the prior distributions. Two formulations

were introduced for the associated combinatorial optimization problem on the calibration graph, and necessary techniques for solving the formulations were discussed. With the construction of the prior distributions, the Bayesian framework was complete, and MCMC techniques were employed to build the posterior distribution for the calibration parameters. BNMC was applied to two real and two synthetic calibration problems. The results demonstrated that the proposed calibration model outperformed all the existing models under the assumption of expensive computational models. We list the major contributions of this study as follows:

- Proposed a novel model for local calibration of expensive computational models without any use of surrogate modeling.
- Presented a unique approach for combining combinatorial optimization, including graph theory and integer programming, with Bayesian statistics and GPs.

In Chapter 4, a similarity-based forecasting model was developed for short-term wind speed forecasting. The similarity-based forecasting model adapted the idea of weighted averaging prediction used by non-parametric linear smoothers, specifically similarity-based regression models, to the domain of forecasting. The proposed model introduced the concept of pattern vector as the vector of spatial and temporal information belonging to each time label, and redefined the notion of similarity as the degree of resemblance between two pattern vectors at two different times. The similarity-based forecasting model used a similarity function to search for similar pattern vectors and made forecasts by taking a weighted average of these similar patterns. Due to the need for forecasting in short intervals in many applications, the model was equipped with a variable selection procedure to choose only significant temporal and spatial information for forecasting. Also, a constrained least squared errors optimization problem was introduced to train the parameters of the proposed model. This optimization problem was solved by transforming it to an unconstrained

optimization problem by using a penalty function. Although the model was primarily developed for short-term wind speed forecasting, it could be efficiently applied to any spatio-temporal system. The similarity-based forecasting model was applied to a wind-farm dataset. The results demonstrated that the model outperformed the forecasting models that were based on conventional time series forecasting methods. We list the major contributions of the study as follows:

- Proposed a forecasting model for spatio-temporal systems capable of forecasting in short intervals by utilizing any type of spatial and temporal information.
- Equipped the forecasting model with a variable selection and an optimization procedure to choose significant variables and train parameters, respectively.

5.1 Future research

We suggest the following research paths to further improve the methodologies proposed in Chapters 2, 3 and 4.

In Chapter 2, SPLGP only considered parallel hyperplanes for partitioning the domain of data, however, more flexible cuts could be considered. Parallel hyper-curves or concentric hyper-spheres that would give the same number of boundaries created by the parallel hyperplanes could be alternative choices. Moreover, from a theoretical perspective, theories should be developed to establish a relationship between the expected covariance function and the expected error in the low-rank covariance approximation. This would be an improvement over the lower bound and simulation study developed for SPLGP.

In Chapter 3, BNMC only considered a single computational data point to construct a prior distribution for each calibration parameter, however, information of multiple computational data points could be taken into account. An implementation of this idea, of course, requires developing new combinatorial optimization techniques

capable of choosing an appropriate number of computational data points. Another interesting research path would be to consider uncertainty of data within the calibration graph instead of using a deterministic calibration graph model and then using Bayesian inference to deal with the uncertainty in the data.

In Chapter 4, the similarity-based forecasting model made minimal distributional assumptions to keep the model as general as possible. However, probability distributions on some parameters and variables of the model could be assumed to construct confidence intervals for forecasts. Furthermore, the current model was unable to handle very large values for m and n or very long pattern vectors due to computational inefficiencies in short intervals. This could be improved by using stochastic gradient descent techniques to train the magnitude parameters in the optimization procedure. Also, an analytical procedure should be developed for choosing proper values of m and n to replace the empirical procedure used in the similarity-based forecasting model.

BIBLIOGRAPHY

[1] Emmanuel Baltsavias, Armin Gruen, Henri Eisenbeiss, Li Zhang, and Lars T Waser. High-quality image matching and automated generation of 3D tree models. *International Journal of Remote Sensing*, 29(5):1243–1259, 2008.

[2] Sudipto Banerjee, Alan E Gelfand, Andrew O Finley, and Huiyan Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society*, 70(4):825–848, 2008.

[3] Douglas M Bates and Donald G Watts. *Nonlinear Regression: Iterative Estimation and Linear Approximations*. John Wiley & Sons, New York, 1988.

[4] Maria J Bayarri, James O Berger, Rui Paulo, Jerry Sacks, John A Cafeo, James Cavendish, Chin-Hsu Lin, and Jian Tu. A framework for validation of computer models. *Technometrics*, 49(2):138–154, 2007.

[5] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, New York, 2013.

[6] Roland Becker and Rolf Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, 2001.

[7] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[8] D. Bertsimas and R. Weismantel. *Optimization Over Integers*. Dynamic Ideas, Belmont, Massachusetts, 2005.

[9] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Three-dimensional face recognition. *International Journal of Computer Vision*, 64(1):5–30, 2005.

[10] Andrew Brown and Sez Atamturktur. Nonparametric functional calibration of computer models. *arXiv preprint arXiv:1602.06202*, 2016.

[11] Barbara G Brown, Richard W Katz, and Allan H Murphy. Time series models to simulate and forecast wind speed and wind power. *Journal of Climate and Applied Meteorology*, 23(8):1184–1195, 1984.

[12] Austin Buchanan, Je Sang Sung, Sergiy Butenko, and Eduardo L. Pasiliao. An integer programming approach for fault-tolerant connected dominating sets. *INFORMS Journal on Computing*, 27(1):178–188, 2015.

[13] Tao Chen and Jianghong Ren. Bagging for Gaussian process regression. *Neurocomputing*, 72(7-9):1605–1610, 2009.

[14] Ilhami Colak, Seref Sagiroglu, and Mehmet Yesilbudak. Data mining and wind power prediction: A literature review. *Renewable Energy*, 46:241–247, 2012.

[15] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[16] Peter S Craig, Michael Goldstein, Jonathan C Rougier, and Allan H Seheult. Bayesian forecasting for complex systems using computer simulators. *Journal of the American Statistical Association*, 96(454):717–729, 2001.

[17] Noel Cressie. The origins of kriging. *Mathematical Geology*, 22(3):239–252, 1990.

[18] Noel Cressie and Hsin-Cheng Huang. Classes of nonseparable, spatio-temporal stationary covariance functions. *Journal of the American Statistical Association*, 94(448):1330–1339, 1999.

[19] Noel Cressie and Christopher K Wikle. *Statistics for Spatio-temporal Data*. John Wiley & Sons, New York, 2011.

[20] Jianqing Fan. Design-adaptive nonparametric regression. *Journal of the American statistical Association*, 87(420):998–1004, 1992.

[21] Shu Fan, James R Liao, Ryuichi Yokoyama, Luonan Chen, and Wei-Jen Lee. Forecasting the wind generation using a two-stage network based on meteorological information. *IEEE Transactions on Energy Conversion*, 24(2):474–482, 2009.

[22] Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and Modeling for Computer Experiments*. CRC Press, Boca Raton, FL, 2005.

[23] Corinne Feremans. *Generalized Spanning Trees and Extensions*. PhD thesis, Universite Libré de Bruxelles, Belgium, 2001.

[24] Corinne Feremans, Martine Labbé, and Gilbert Laporte. A comparative analysis of several formulations for the generalized minimum spanning tree problem. *Networks*, 39(1):29–34, 2002.

[25] Gerald B Folland. *Real Analysis: Modern Techniques and Their Applications*. John Wiley & Sons, New York, 2013.

[26] Shengguo Gao, Zhongli Zhu, Shaomin Liu, Rui Jin, Guangchao Yang, and Lei Tan. Estimating the spatial distribution of soil moisture based on Bayesian maximum entropy method with auxiliary data from remote sensing. *International Journal of Applied Earth Observation and Geoinformation*, 32:54–66, 2014.

[27] Alan E Gelfand, Susan E Hills, Amy Racine-Poon, and Adrian FM Smith. Illustration of Bayesian inference in normal data models using gibbs sampling. *Journal of the American Statistical Association*, 85(412):972–985, 1990.

[28] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, London, 1995.

[29] Itzhak Gilboa, Offer Lieberman, and David Schmeidler. Empirical similarity. *The Review of Economics and Statistics*, 88(3):433–444, 2006.

[30] Tilmann Gneiting, Kristin Larson, Kenneth Westrick, Marc G Genton, and Eric Aldrich. Calibrated probabilistic forecasting at the stateline wind energy center: The regime-switching space–time method. *Journal of the American Statistical Association*, 101(475):968–979, 2006.

[31] Bruce Golden, Subramanian Raghavan, and Daliborka Stanojević. Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17(3):290–304, 2005.

[32] Michael Goldstein and Jonathan Rougier. Reified Bayesian modelling and inference for physical systems. *Journal of Statistical Planning and Inference*, 139(3):1221–1239, 2009.

[33] Robert B Gramacy and Herbert K H Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103:1119–1130, 2008.

[34] Armin Gruen and Devrim Akca. Least squares 3D surface and curve matching. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(3):151–174, 2005.

[35] Timothy C Haas. Kriging and automated variogram modeling within a moving window. *Atmospheric Environment*, 24(7):1759–1769, 1990.

[36] William W Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[37] James Douglas Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, London, 1994.

[38] Gang Han, Thomas J Santner, and Jeremy J Rawlinson. Simultaneous determination of tuning and calibration parameters for computer experiments. *Technometrics*, 51(4):464–474, 2009.

[39] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The Elements of Statistical Learning*. Springer, New York, 2009.

[40] Amanda S Hering and Marc G Genton. Powering up with space-time wind forecasting. *Journal of the American Statistical Association*, 105(489):92–104, 2010.

[41] Dave Higdon, James Gattiker, Brian Williams, and Maria Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

[42] Dave Higdon, Marc Kennedy, James C Cavendish, John A Cafeo, and Robert D Ryne. Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 26(2):448–466, 2004.

[43] Dave Higdon, Charles Nakhleh, James Gattiker, and Brian Williams. A Bayesian calibration approach to the thermal problem. *Computer Methods in Applied Mechanics and Engineering*, 197(29):2431–2441, 2008.

[44] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proc. R. Soc. Lond. A*, 186(1007):453–461, 1946.

[45] Roshan Joseph and Shreyes N Melkote. Statistical adjustments to engineering models. *Journal of Quality Technology*, 41(4):362, 2009.

[46] Jaesung Jung and Robert P Broadwater. Current status and future advances for wind speed and power forecasting. *Renewable and Sustainable Energy Reviews*, 31:762–777, 2014.

[47] Lulu Kang and V Roshan Joseph. Kernel approximation: From regression to interpolation. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):112–129, 2016.

[48] Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society*, 63(3):425–464, 2001.

[49] Hyoung-Moon Kim, Bani K Mallick, and Chris C Holmes. Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association*, 100(470):653–668, 2005.

[50] Andrey Kolmogoroff. Interpolation und extrapolation von stationaren zufalligen folgen. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 5(1):3–14, 1941.

[51] Eugene L Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York, 1976.

[52] Gong Li and Jing Shi. On comparing three artificial neural networks for wind speed forecasting. *Applied Energy*, 87(7):2313–2320, 2010.

[53] Offer Lieberman. Asymptotic theory for empirical similarity models. *Econometric Theory*, 26(04):1032–1059, 2010.

[54] Heping Liu, Jing Shi, and Ergin Erdem. Prediction of wind speed time series using modified Taylor kriging method. *Energy*, 35(12):4870–4879, 2010.

[55] Jason L Loeppky, Derek Bingham, and William J Welch. Computer model calibration or tuning in practice. Technical report, University of British Columbia, Vancouver, BC, CA, 2006.

[56] David G Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer, New York, 1984.

[57] Georges Matheron. The intrinsic random functions and their applications. *Advances in Applied Probability*, 5(3):439–468, 1973.

[58] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[59] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to Linear Regression Analysis*. John Wiley & Sons, New York, 2015.

[60] Douglas C Montgomery and George C Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, New York, 2010.

[61] Esmaeel Moradi and Balabhaskar Balasundaram. Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters*, November 2015. See also: Y. Lu, E. Moradi, and B. Balasundaram. Correction to: Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters*, DOI: 10.1007/s11590-018-1273-7, 2018.

[62] Hiroyuki Mori and Eitaro Kurata. Application of Gaussian process to wind speed forecasting for wind power generation. In *Sustainable Energy Technologies, 2008. ICSET 2008. IEEE International Conference on*, pages 956–959. IEEE, 2008.

[63] Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241, 1995.

[64] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[65] Radford M Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media, New York, 2012.

[66] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*. SIAM, Philadelphia, 1994.

[67] Anthony O'Hagan and John Frank Charles Kingman. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B (Methodological)*, 40(1):1–42, 1978.

[68] Jose C Palomares-Salas, Juan J G De la Rosa, Jose G Ramiro, Jose Melgar, Agustn Aguera, and Antonio Moreno. Arima vs. neural networks for wind speed forecasting. In *Computational Intelligence for Measurement Systems and Applications, 2009. CIMSA'09. IEEE International Conference on*, pages 129–133. IEEE, 2009.

[69] Chiwoo Park and Daniel Apley. Patchwork kriging for large-scale Gaussian process regression. *arXiv preprint arXiv:1701.06655*, 2017.

[70] Chiwoo Park, Jianhua Z. Huang, and Yu Ding. Domain decomposition approach for fast Gaussian process regression of large spatial data sets. *Journal of Machine Learning Research*, 12:1697–1728, 2011.

[71] Chiwoo Park, Jianhua Z Huang, and Yu Ding. GPLP: Alocal and parallel computation toolbox for Gaussian process regression. *Journal of Machine Learning Research*, 13:775–779, 2012.

[72] Johan Philip. The probability distribution of the distance between two random points in a box. Technical report, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden, 2007.

[73] Rui Filipe Carneiro Barbosa Pinto. *Wind power forecasting uncertainty and unit commitment*. PhD thesis, Electrical and Computers Engineering, University of Porto, Porto, Portugal, 2014.

[74] Matthew Plumlee, Roshan Joseph, and Hui Yang. Calibrating functional parameters in the ion channel models of cardiac cells. *Journal of the American Statistical Association*, 111(514):500–509, 2016.

[75] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[76] Petrica C. Pop. New models of the generalized minimum spanning tree problem. *Journal of Mathematical Modelling and Algorithms*, 3(2):153–166, Jun 2004.

[77] Petrica C. Pop, W. Kern, and G. Still. A new relaxation method for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 170(3):900–908, 2006.

[78] Petrica Claudiu Pop. *The generalized minimum spanning tree problem*. PhD thesis, University of Twente, 2000.

[79] Arash Pourhabib and Balabhaskar Balasundaram. Non-isometric curve to surface matching with incomplete data for functional calibration. *arXiv preprint arXiv:1508.01240*, 2015.

[80] Arash Pourhabib, Jianhua Z Huang, and Yu Ding. Short-term wind speed forecast using measurements from multiple turbines in a wind farm. *Technometrics*, 58(1):138–147, 2016.

[81] Arash Pourhabib, Jianhua Z Huang, Kan Wang, Chuck Zhang, Ben Wang, and Yu Ding. Modulus prediction of buckypaper based on multi-fidelity analysis involving latent variables. *IIE Transactions*, 47(2):141–152, 2015.

[82] Arash Pourhabib, Jianhua Z Huang, Kan Wang, Chuck Zhang, Ben Wang, and Yu Ding. Modulus prediction of buckypaper based on multi-fidelity analysis involving latent variables. *IIE Transactions*, 47(2):141–152, 2015.

[83] Arash Pourhabib, Faming Liang, and Yu Ding. Bayesian site selection for fast Gaussian process regression. *IIE Transactions*, 46(5):543–555, 2014.

[84] Arash Pourhabib, Rui Tuo, Shiyuan He, Yu Ding, and Jianhua Z Huang. Local calibration of computer models, 2014. Manuscript.

[85] Matthew T Pratola, Stephan R Sain, Derek Bingham, Michael Wiltberger, and E Joshua Rigler. Fast sequential computer model calibration of large nonstationary spatial-temporal processes. *Technometrics*, 55(2):232–242, 2013.

[86] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[87] Carl Edward Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems*, 2:881–888, 2002.

[88] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[89] Shane Reese, Alyson G Wilson, Michael Hamada, Harry F Martz, and Kenneth J Ryan. Integrated analysis of computer and physical experiments. *Technometrics*, 46(2):153–164, 2004.

[90] Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Phil. Trans. R. Soc. A*, 371(1984):20110550, 2013.

[91] Maximilian Ruhland. *Similarity-based probabilistic forecasting of wind speed*. Bachelor's thesis, University of Heidelberg, 2014.

[92] Thomas J Santner, Brian J Williams, and William I Notz. *The Design and Analysis of Computer Experiments*. Springer Science & Business Media, New York, 2013.

[93] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, pages 416–426. Springer, 2001.

[94] Bernhard Schölkopf and Alexander J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, Cambridge, MA, 2001.

[95] Alexander Schrijver. *Combinatorial optimization: Polyhedra and Efficiency*, volume 24. Springer Science & Business Media, New York, 2002.

[96] Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate Gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 977–984, Cambridge, MA, 2003. MIT Press.

[97] Nils Siebert. *Development of methods for regional wind power forecasting*. PhD thesis, École Nationale Supérieure des Mines de Paris, Paris, France, 2008.

[98] Alex J Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

[99] Edward Snelson. *Flexible and Efficient Gaussian Process Models for Machine Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, London, England, 2007.

[100] Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In *International Conference on Artifical Intelligence and Statistics 11*. Society for Artificial Intelligence and Statistics, 2007.

[101] Michael Stein. A simple model for spatial-temporal processes. *Water Resources Research*, 22(13):2107–2110, 1986.

[102] Charles J Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5(4):595–620, 1977.

[103] Jose Luis Torres, Almudena Garcia, Marian De Blas, and Adolfo De Francisco. Forecast of hourly average wind speed with arma models in Navarre (Spain). *Solar Energy*, 79(1):65–77, 2005.

[104] Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

[105] Rui Tuo and Chien-Fu J Wu. Efficient calibration for imperfect computer models. *The Annals of Statistics*, 43(6):2331–2352, 2015.

[106] Rui Tuo and Chien-Fu J Wu. A theoretical framework for calibration in computer models: parametrization, estimation and convergence properties. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):767–795, 2016.

[107] Raquel Urtasun and Trevor Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[108] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer science & business media, New York, 1999.

[109] Wenjia Wang, Rui Tuo, and CF Wu. Universal convergence of kriging. *arXiv preprint arXiv:1710.06959*, 2017.

[110] Geoffrey S Watson. Smooth regression analysis. *The Indian Journal of Statistics, Series A*, pages 359–372, 1964.

[111] Holger Wendland. *Scattered Data Approximation*, volume 17. Cambridge University Press, Cambridge, MA, 2004.

[112] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press, Cambridge, MA, 1949.

[113] Brian Williams, Dave Higdon, Jim Gattiker, Leslie Moore, Michael McKay, and Sallie Keller-McNulty. Combining experimental data and computer simulations, with an application to flyer plate experiments. *Bayesian Analysis*, 1(4):765–792, 2006.

[114] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, pages 514–520. MIT Press, 1996.

[115] Christopher KI Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. MIT Press, 2001.

[116] Ying Xiong, Wei Chen, Kwok-Leung Tsui, and Daniel W Apley. A better understanding of model updating strategies in validating engineering models. *Computer Methods in Applied Mechanics and Engineering*, 198(15):1327–1337, 2009.

[117] Zhuohua Xu. *Similarity based probabilistic forecasting with an application to wind speeds*. Diploma thesis, University of Heidelberg, Germany, 2011.

[118] Jialin Zhang, Xiuhong Li, Qiang Liu, Long Zhao, and Baocheng Dou. An extended kriging method to interpolate soil moisture data measured by wireless sensor network. *Sensors*, 17(6):1390, 2017.

[119] Xinxin Zhu, Kenneth P Bowman, and Marc G Genton. Incorporating geostrophic wind information for improved space–time short-term wind speed forecasting. *The Annals of Applied Statistics*, 8(3):1782–1799, 2014.

APPENDIX A

Proof of Theorems

A.1 Proof of Theorem 2.1

Proof. For any $i \in [m_s]$, let $\mathbf{u}_i$ denote the covariance vector between $\mathbf{z}$ and the first i members of $\tilde{\mathbf{X}}_s$, and let $\mathbf{v}_i$ denote the covariance vector between the $(i+1)^{\text{th}}$ member of $\tilde{\mathbf{X}}_s$ and the first i members of $\tilde{\mathbf{X}}_s$. That is, $\mathbf{u}_i = [\mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}_1}), \ldots, \mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_i)]^T$, and $\mathbf{v}_i = [\mathcal{K}(\tilde{\mathbf{x}}_{i+1}, \tilde{\mathbf{x}}_1), \ldots, \mathcal{K}(\tilde{\mathbf{x}}_{i+1}, \tilde{\mathbf{x}}_i)]^T$. Also let $\mathbf{K}_i$ denote the covariance matrix between the first i members of $\tilde{\mathbf{X}}_s$ themselves. We now prove by induction on i. For the base case, i.e, $i = 1$, the claim clearly holds,

$$\mathbb{E}_{\Omega_s}(\mathbf{u}_1^T \mathbf{K}_1^{-1} \mathbf{u}_1) = \mathbb{E}_{\Omega_s}(\mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_1)\mathcal{K}(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_1)\mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_1)) = h\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{z}, \tilde{\mathbf{x}}_1)) \tag{A.1}$$

Suppose the claim holds for $m_s - 1$, we show that it also holds for m_s. Expanding $\mathbf{u}_{m_s}^T \mathbf{K}_{m_s}^{-1} \mathbf{u}_{m_s}$ gives

$$\mathbf{u}_{m_s}^T \mathbf{K}_{m_s}^{-1} \mathbf{u}_{m_s} = \begin{bmatrix} \mathbf{u}_{m_s-1}^T & \mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_{m_s}) \end{bmatrix} \begin{bmatrix} \mathbf{K}_{m_s-1} & \mathbf{v}_{m_s-1} \\ \mathbf{v}_{m_s-1}^T & h \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{u}_{m_s-1} \\ \mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_{m_s}) \end{bmatrix} \tag{A.2a}$$

$$= \begin{bmatrix} \mathbf{u}_{m_s-1}^T & \mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_{m_s}) \end{bmatrix} \begin{bmatrix} \mathbf{K}_{m_s-1}^{-1} + c\mathbf{K}_{m_s-1}^{-1}\mathbf{v}_{m_s-1}\mathbf{v}_{m_s-1}^T\mathbf{K}_{m_s-1}^{-1} & -c\mathbf{K}_{m_s-1}^{-1}\mathbf{v}_{m_s-1} \\ -c\mathbf{v}_{m_s-1}^T\mathbf{K}_{m_s-1}^{-1} & c \end{bmatrix} \begin{bmatrix} \mathbf{u}_{m_s-1}^T \\ \mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_{m_s}) \end{bmatrix} \tag{A.2b}$$

$$= \mathbf{u}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{u}_{m_s-1} + \frac{(\mathbf{v}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{u}_{m_s-1})^2 + \mathcal{K}^2(\mathbf{z}, \tilde{\mathbf{x}}_{m_s}) - 2\mathbf{v}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{u}_{m_s-1} \mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_{m_s})}{c} \tag{A.2c}$$

$$= \mathbf{u}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{u}_{m_s-1} + \frac{(\mathbf{v}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{u}_{m_s-1} - \mathcal{K}(\mathbf{z}, \tilde{\mathbf{x}}_{m_s}))^2}{c} \tag{A.2d}$$

$$\geq \mathbf{u}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{u}_{m_s-1}. \tag{A.2e}$$

where equality (A.2b) follows from the block matrix inversion lemma [36], and $c = (h - \mathbf{v}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{v}_{m_s-1})^{-1}$, which is always non-negative.

By (A.2) and the induction step,

$$\mathbb{E}_{\Omega_s}(\mathbf{u}_{m_s}^T \mathbf{K}_{m_s}^{-1} \mathbf{u}_{m_s}) \geq \mathbb{E}_{\Omega_s}(\mathbf{u}_{m_s-1}^T \mathbf{K}_{m_s-1}^{-1} \mathbf{u}_{m_s-1}) \geq h\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x}, \mathbf{x}')). \tag{A.3}$$

■

A.2 Proof of Theorem 2.2

First, we prove the following lemma

Lemma A.1 *For the random variables $z_1 \sim \mathcal{U}(a, a+d)$ and $z_2 \sim \mathcal{U}(b, b+d)$, where $a \leq b$ and $a, b, d \geq 0$, define $v = (z_1 - z_2)^2$. Let v_0 denote a special case of v when $a = b$. Then $\mathbb{E}_v(\exp(-cv)) \leq \mathbb{E}_{v_0}(\exp(-cv_0))$ for any $c > 0$.*

Proof. Let $z = z_1 - z_2$, then by convolution of probability distributions, we derive the PDF of z as

$$f_z(t) = \begin{cases} \frac{1}{d^2}(t + b - a + d) & a - b - d \leq t < a - b, \\ \frac{-1}{d^2}(t + b - a - d) & a - b \leq t \leq a - b + d. \end{cases} \tag{A.4}$$

Hence, $F_v(t) = p(v \leq t) = p(z^2 \leq t) = p(\sqrt{t} \leq z \leq \sqrt{t})$, which can be written as

$$F_v(t) = \begin{cases} \frac{2\sqrt{t}}{d^2}(a - b + d) & 0 \leq \sqrt{t} < b - a, \\ \frac{1}{d^2}(2\sqrt{t}d - t - (a-b)^2) & b - a \leq \sqrt{t} < a - b + d, \\ 1 - \frac{1}{2d^2}(\sqrt{t} + a - b - d)^2 & a - b + d \leq \sqrt{t} \leq b - a + d. \end{cases} \tag{A.5}$$

Note that for the case of v_0, CDF (A.5) reduces to

$$F_{v_0}(t) = \frac{1}{d^2}(2\sqrt{t}d - t) \;\; 0 \leq \sqrt{t} \leq d. \tag{A.6}$$

Comparing $F_{v_0}(t)$ and $F_v(t)$ for all possible values of t gives

- $\sqrt{t} < 0$: $F_{v_0}(t) = F_v(t) = 0$.

- $0 \leq \sqrt{t} < b - a$: then $F_v(t) - F_{v_0}(t) = \frac{1}{d^2}(2\sqrt{t}(a-b) + t)$. Since $\sqrt{t} < b - a \Rightarrow t < \sqrt{t}(b-a) \Rightarrow t + \sqrt{t}(a-b) < 0 \Rightarrow t + 2\sqrt{t}(a-b) < 0 \Rightarrow F_v(t) - F_{v_0}(t) < 0 \Rightarrow F_v(t) < F_{v_0}(t)$.

- $b - a \leq \sqrt{t} < a - b + d$: then $F_v(t) - F_{v_0}(t) = -\frac{(a-b)^2}{d^2} < 0 \Rightarrow F_v(t) - F_{v_0}(t) < 0 \Rightarrow F_v(t) < F_{v_0}(t)$.

- $a - b + d \leq \sqrt{t} < d$: then $F_v(t) - F_{v_0}(t) = 1 - \frac{1}{2d^2}(\sqrt{t} + a - b - d)^2 + \frac{1}{d^2}(t - 2\sqrt{t}d)$. Note that $\frac{d(F_v(t) - F_{v_0}(t))}{dt} = \frac{1}{2d^2}(1 - \frac{a-b+d}{\sqrt{t}}) > 0$, and therefore, $F_v(t) - F_{v_0}(t)$ is a monotonically increasing function. Due to the monotonicity of $F_v(t) - F_{v_0}(t)$, the maximum occurs at d, so $\max_t F_v(t) - F_{v_0}(t) = F_v(d) - F_{v_0}(d) = -\frac{(a-b)^2}{d^2} < 0$. Therefore, $F_v(t) - F_{v_0}(t) \leq F_v(d) - F_{v_0}(d) < 0 \Rightarrow F_v(t) \leq F_{v_0}(t)$.

- $d \leq \sqrt{t} < b - a + d$: in this case $F_{v_0}(t)$ is always 1, hence, $F_v(t) \leq F_{v_0}(t)$.

- $b - a + d \leq \sqrt{t}$: in this case $F_{v_0}(t) = F_v(t) = 1$.

We conclude that

$$p(v \leq t) \leq p(v_0 \leq t)\ \forall t \in \mathbb{R} \Rightarrow p(-cv \geq t') \leq p(-cv_0 \geq t')\ \forall t' \in \mathbb{R} \text{ and } c > 0,$$

which implies $-cv$ is stochastically less than $-cv_0$, i.e., $-cv \preceq_{st} -cv_0$. Consequently, the expectation of any non-decreasing function of these two variables are ordered, i.e., $\mathbb{E}_v(\exp(-cv)) \leq \mathbb{E}_{v_0}(\exp(-cv_0))$ for any $c > 0$. ■

Proof. [**Proof of Theorem 2.2**] Let $\mathbf{x}_{\{k\}} = \{x_1, \ldots, x_p\} \backslash \{x_k\}$ for any $\mathbf{x} \in \Omega$. Then, based on how each $\Omega_{\boldsymbol{\theta},k,W,s}$ in (2.26) is constructed and considering the distribution of the data points in Ω according to (2.23), all variables $x_j \in \mathbf{x}_{\{k\}}$ are independent and have the uniform distribution $\mathcal{U}(0, L)$. Moreover, by the definition of the hyperplanes in (2.25), and given $\mathbf{x}_{\{k\}}$, the corresponding values of the variable x_k on the

hyperplanes $H_{\boldsymbol{\theta},k,W,s-1}$ and $H_{\boldsymbol{\theta},k,W,s}$ are

$$\sum_{j\in[p]\setminus\{k\}} \tan(\theta_j)x_j + (s-1)w \quad \& \sum_{j\in[p]\setminus\{k\}} \tan(\theta_j)x_j + sw. \tag{A.7}$$

Therefore, the conditional distribution $x_k|\mathbf{x}_{\{k\}}$ in the parallelogram subdomain $\Omega_{\boldsymbol{\theta},k,W,s}$ has a uniform distribution whose support is bounded by the values calculated in (A.7). Consequently, given a parallelogram subdomain $\Omega_{\boldsymbol{\theta},k,W,s}$, for any $\mathbf{x} \in \Omega_{\boldsymbol{\theta},k,W,s-1}$,

$$x_j \sim \mathcal{U}(0,L) \quad \forall j \in [p]\setminus\{k\}, \tag{A.8a}$$

$$x_k|\mathbf{x}_{\{k\}} \sim \mathcal{U}\Big(\sum_{j\in[p]\setminus\{k\}} \tan(\theta_j)x_j + (s-1)w, \sum_{j\in[p]\setminus\{k\}} \tan(\theta_j)x_j + sw\Big). \tag{A.8b}$$

Now that we have the distribution (A.8) we can expand $\mathbb{E}_{\Omega_{\boldsymbol{\theta},k,W,s}}\big(\mathcal{K}(\mathbf{x},\mathbf{x}')\big)$ by conditioning, that is

$$\mathbb{E}_{\Omega_{\boldsymbol{\theta},k,W,s}}\big(\mathcal{K}(\mathbf{x},\mathbf{x}')\big) = \mathbb{E}_{\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}}\Big(\mathbb{E}_{x_k,x'_k}\Big(\mathcal{K}(\mathbf{x},\mathbf{x}') \mid \mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}\Big)\Big) \tag{A.9a}$$

$$= \mathbb{E}_{\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}}\Big(\exp\Big(-\sum_{j\in[p]\setminus\{k\}} \gamma_j(x_j-x'_j)^2\Big)\mathbb{E}_{x_k,x'_k}\Big(\exp\big(-\gamma_k(x_k-x'_k)^2\big) \mid \mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}\Big)\Big) \tag{A.9b}$$

$$= \mathbb{E}_{\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}}\big(g(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}})h(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}})\big). \tag{A.9c}$$

Note that the function $g(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}})$ is always positive and independent of $\boldsymbol{\theta}$, and function $h(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}})$ is positive that attains its maximum for any given $\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}$ at $\boldsymbol{\theta}=\mathbf{0}$ by Lemma (A.1). Therefore, by denoting h_0 as the special case of function h, where $\boldsymbol{\theta}=\mathbf{0}$,

$$g(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}})h(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}) \le g(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}})h_0(\mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}}) \quad \forall \mathbf{x}_{\{k\}},\mathbf{x}'_{\{k\}},$$

which results in

$$\mathbb{E}_{\Omega_{\boldsymbol{\theta},k,W,s}}\big(\mathcal{K}(\mathbf{x},\mathbf{x}')\big) \le \mathbb{E}_{\Omega_{\boldsymbol{\theta},k,W,s}}\big(\mathcal{K}(\mathbf{x},\mathbf{x}') \mid \boldsymbol{\theta}=\mathbf{0}\big)$$

$$\Rightarrow \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\Omega_{\boldsymbol{\theta},k,W,s}}\big(\mathcal{K}(\mathbf{x},\mathbf{x}')\big) = \mathbf{0}.$$

■

A.3 Proof of Theorem 2.3

First, we prove the following lemma

Lemma A.2 $\mathbb{E}_{z_1,z_2}\Big(\exp\big(-c(z_1-z_2)^2\big)\Big) = \int_0^{b^2}\exp(-ct)(\frac{1}{b\sqrt{t}}-\frac{1}{b^2})dt$, *where* $z_1, z_2 \overset{i.i.d}{\sim} \mathcal{U}(a, a+b)$.

Proof. Let $v = (z_1 - z_2)^2$, then [72] shows that v has the following PDF:

$$f_v(t) = \frac{1}{\sqrt{t}b} - \frac{1}{b^2} \quad \forall\, 0 \le t \le b^2;$$

therefore,

$$\mathbb{E}_{z_1,z_2}\Big(\exp\big(-c(z_1-z_2)^2\big)\Big) = \mathbb{E}_v\Big(\exp(-cv)\Big) =$$
$$\int_0^{b^2}\exp(-ct)f_s(t)dt = \int_0^{b^2}\exp(-ct)(\frac{1}{b\sqrt{t}}-\frac{1}{b^2})dt.$$

■

Proof. [**Proof of Theorem 2.3**] By the assumptions of uniform distribution of points in Ω (2.23), and independence of the dimensions due to geometry of $\Omega_{\mathbf{0},k,W,s}$, for any $\mathbf{x} \in \Omega_{\mathbf{0},k,W,s}$,

$$x_k \sim \mathcal{U}\big((s-1)W, sW\big) \quad \& \quad x_j \sim \mathcal{U}\big(0, L\big) \quad \forall j \in [p]\backslash\{k\}. \tag{A.10}$$

Letting $G_k = \mathbb{E}_{\Omega_{\mathbf{0},k,W,s}}(\mathcal{K}(\mathbf{x},\mathbf{x}'))$, and using distribution (A.10),

$$G_k = \mathbb{E}_{x_k}\Big(\exp\big(-\gamma_k(x_k-x_k')^2\big)\Big)\prod_{j\in[p]\backslash\{k\}}\mathbb{E}_{x_j}\Big(\exp\big(-\gamma_j(x_j-x_j')^2\big)\Big) \tag{A.11a}$$
$$= \mathbb{E}_{v_k}\Big(\exp(-\gamma_k v_k)\Big)\prod_{j\in[p]\backslash\{k\}}\mathbb{E}_{v_j}\Big(\exp(-\gamma_j v_j)\Big) \tag{A.11b}$$
$$= \Big(\int_0^{W^2}\exp(-\gamma_k t)(\frac{1}{W\sqrt{t}}-\frac{1}{W^2})dt\Big)\Big(\prod_{j\in[p]\backslash\{k\}}\Big(\int_0^{L^2}\exp(-\gamma_j t)(\frac{1}{L\sqrt{t}}-\frac{1}{L^2})dt\Big)\Big) \tag{A.11c}$$
$$= \Big(\int_0^{W^2} g_k^W(t)dt\Big)\Big(\prod_{j\in[p]\backslash\{k\}}\Big(\int_0^{L^2} g_j^L(t)dt\Big)\Big), \tag{A.11d}$$

where equality (A.11a) follows from the independence of dimensions in each $\Omega_{\mathbf{0},i,W,s}$, equalities (A.11b) and (A.11c) follow from Lemma (A.2) with $f_{v_k}(t) = \frac{1}{\sqrt{t}W} - \frac{1}{W^2}$ $0 \leq t \leq W^2$ and $f_{v_j}(t) = \frac{1}{\sqrt{t}L} - \frac{1}{L^2}$ $0 \leq t \leq L^2$, and $g_\ell^m(t) = \exp(-\gamma_\ell t)(\frac{1}{m\sqrt{t}} - \frac{1}{m^2})$ in (A.11d).

To show that $G_p - G_k \geq 0$ for any $k \in [p]$, We first expand $G_p - G_k$,

$$G_p - G_k = \Big(\int_0^{W^2} g_p^W(t)dt\Big)\Big(\prod_{j\in[p]\setminus\{p\}}\Big(\int_0^{L^2} g_j^L(t)dt\Big)\Big) - \Big(\int_0^{W^2} g_k^W(t)dt\Big)\Big(\prod_{j\in[p]\setminus\{k\}}\Big(\int_0^{L^2} g_j^L(t)dt\Big)\Big)$$

$$= \Big(\prod_{j\in[p]\setminus\{k,p\}}\Big(\int_0^{L^2} g_j^L(t)dt\Big)\Big)\Big(\int_0^{W^2} g_p^W(t)dt\int_0^{L^2} g_k^L(t)dt - \int_0^{W^2} g_k^W(t)dt\int_0^{L^2} g_p^L(t)dt\Big) = A * B.$$

Note that A is always positive, since each $\int_0^{L^2} g_j^L(t)dt$ is the expectation of the random variable $\exp(-\gamma_j v_j)$ which is positive. Hence, it is enough to show that B is positive. Expanding B further,

$$B = \Big(\int_0^{W^2} g_p^W(t)dt\Big)\Big(\int_0^{W^2} g_k^L(t)dt + \int_{W^2}^{L^2} g_k^L(t)dt\Big) - \Big(\int_0^{W^2} g_k^W(t)dt\Big)\Big(\int_0^{W^2} g_p^L(t)dt + \int_{W^2}^{L^2} g_p^L(t)dt\Big) \tag{A.12a}$$

$$= \int_{t_k:0}^{W^2}\int_{t_p:0}^{W^2} g_p^W(t_k)g_k^L(t_p)dt_k dt_p + \int_{t_k:0}^{W^2}\int_{t_p:W^2}^{L^2} g_p^W(t_k)g_k^L(t_p)dt_k dt_p$$
$$- \int_{t_k:0}^{w^2}\int_{t_p:0}^{w^2} g_k^w(t_k)g_p^L(t_p)dt_k dt_p - \int_{t_k:0}^{w^2}\int_{t_p:w^2}^{L^2} g_k^w(t_k)g_p^L(t_p)dt_k dt_p \tag{A.12b}$$

$$= \int_{t_k:0}^{W^2}\int_{t_p:0}^{W^2} \big(\exp(-\gamma_p t_k - \gamma_k t_p) - \exp(-\gamma_k t_k - \gamma_p t_p)\big)\big(\frac{1}{W\sqrt{t_k}} - \frac{1}{W^2}\big)\big(\frac{1}{L\sqrt{t_p}} - \frac{1}{L^2}\big)dt_k dt_p$$
$$+ \int_{t_k:0}^{W^2}\int_{t_p:W^2}^{L^2} \big(\exp(-\gamma_p t_k - \gamma_k t_p) - \exp(-\gamma_k t_k - \gamma_p t_p)\big)\big(\frac{1}{W\sqrt{t_k}} - \frac{1}{W^2}\big)\big(\frac{1}{L\sqrt{t_p}} - \frac{1}{L^2}\big)dt_k dt_p \tag{A.12c}$$

$$= \int_{t_k:0}^{W^2}\int_{t_p:0}^{W^2} c(t_k,t_p)dt_k dt_p + \int_{t_k:0}^{W^2}\int_{t_p:W^2}^{L^2} c(t_k,t_p)dt_k dt_p. \tag{A.12d}$$

Note that for any member of set

$$\{(W,L,t_p,t_k,\gamma_p,\gamma_k) \mid 0 < W < L,\ 0 < \gamma_k < \gamma_p,\ 0 \leq t_k \leq W^2,\ W^2 \leq t_p \leq L^2\}, \tag{A.13}$$

we have

$$\big(\frac{1}{w\sqrt{t_k}} - \frac{1}{w^2}\big)\big(\frac{1}{L\sqrt{t_p}} - \frac{1}{L^2}\big) > 0, \tag{A.14}$$

and also

$$(-\gamma_p t_k - \gamma_k t_p) - (-\gamma_k t_k - \gamma_p t_p) = (\gamma_p - \gamma_k)(t_p - t_k) > 0, \tag{A.15}$$

where the latter results in

$$\exp(-\gamma_p t_k - \gamma_k t_p) - \exp(-\gamma_k t_k - \gamma_p t_p) > 0. \tag{A.16}$$

Therefore, by (A.14) and (A.16), the integrand $c(t_k, t_p)$ in (A.12d) is positive for any member of set (A.13), so is integral $\int_{t_k:0}^{W^2} \int_{t_p:W^2}^{L^2} c(t_k, t_p) dt_k dt_p$. Hence, to complete the proof we need to show integral $\int_{t_k:0}^{w^2} \int_{t_p:0}^{w^2} c(t_k, t_p) dt_k dt_p$ in (A.12d) is also positive. To show this, we expand the integral,

$$\int_{t_k:0}^{W^2} \int_{t_p:0}^{W^2} c(t_k, t_p) dt_k dt_p = \int_{t_k:0}^{W^2} \int_{t_p:t_k}^{W^2} c(t_k, t_p) dt_k dt_p + \int_{t_p:0}^{W^2} \int_{t_k:t_p}^{W^2} c(t_k, t_p) dt_p dt_k \tag{A.17a}$$

$$= \int_{t_k:0}^{W^2} \int_{t_p:t_k}^{W^2} c(t_k, t_p) dt_k dt_p + \int_{t_k:0}^{W^2} \int_{t_p:t_k}^{W^2} c(t_p, t_k) dt_k dt_p = \int_{t_k:0}^{W^2} \int_{t_p:t_k}^{W^2} \big(c(t_k, t_p) + c(t_p, t_k)\big) dt_k dt_p \tag{A.17b}$$

$$= \frac{1}{wL} \int_{t_k:0}^{W^2} \int_{t_p:t_k}^{W^2} \big(\exp(-\gamma_p t_k - \gamma_k t_p) - \exp(-\gamma_k t_k - \gamma_p t_p)\big)(\frac{1}{\sqrt{t_k}} - \frac{1}{\sqrt{t_p}})(\frac{1}{W} - \frac{1}{L}) dt_k dt_p. \tag{A.17c}$$

Similar to (A.13)-(A.16), for any member of set

$$\{(W, L, t_p, t_k, \gamma_p, \gamma_k) \mid 0 < W < L,\ 0 < \gamma_k < \gamma_p,\ 0 \le t_k \le W^2,\ t_k \le t_p \le W^2\} \tag{A.18}$$

we have

$$(\frac{1}{\sqrt{t_k}} - \frac{1}{\sqrt{t_p}})(\frac{1}{W} - \frac{1}{L}) > 0 \tag{A.19}$$

and

$$\big(\exp(-\gamma_p t_k - \gamma_k t_p) - \exp(-\gamma_k t_k - \gamma_p t_p)\big) > 0. \tag{A.20}$$

Hence the integrand in (A.17c) is positive for any member of set (A.18), so is integral (A.17c), and the proof is complete.

■

APPENDIX B

A simulation study on the relation between expected error (2.22) and $\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x}, \mathbf{x}'))$

Consider the squared exponential Gaussian kernel $\mathcal{K}(x, x') = exp(-\gamma(x - x')^2)$ with $\gamma > 0$ defined on

$$\Omega_s = \{x \in \mathbb{R} | a \leq x \leq a + b\} \tag{B.1}$$

with uniform sampling distribution

$$x \sim \mathcal{U}(a, a + b) \quad \forall x \in \Omega_s. \tag{B.2}$$

To have a general simulation study, we need the following lemma.

Lemma B.1 $\mathbb{E}_{z_1,z_2}\Big(\exp\big(-c(z_1 - z_2)^2\big)\Big)$, *where* $z_1, z_2 \overset{i.i.d}{\sim} \mathcal{U}(a, a + b)$, *is a monotonically decreasing function of* c *and* b.

Proof. We need to show that $\nabla g(b, c) = [\frac{\partial g(b,c)}{\partial b}, \frac{\partial g(b,c)}{\partial c}]^T < 0$ for all $[b, c]^T > 0$, where

$$g(b, c) = \mathbb{E}_{z_1,z_2}\Big(\exp\big(-c(z_1 - z_2)^2\big)\Big) = \int_0^{b^2} \exp(-ct)(\frac{1}{b\sqrt{t}} - \frac{1}{b^2})dt$$

by Lemma A.2.

We can write $\frac{\partial g(b,c)}{\partial b}$ as

$$\frac{\partial g(b, c)}{\partial b} = \frac{1}{b^2} \int_0^{b^2} \exp(-ct)(\frac{2}{b} - \frac{1}{\sqrt{t}})dt \tag{B.3a}$$

$$= \frac{1}{b^2}\left(\left[\exp(-ct)(\frac{2t}{b} - 2\sqrt{t})\right]_0^{b^2} - \int_0^{b^2} -c \exp(-ct)(\frac{2t}{b} - 2\sqrt{t})\right) \tag{B.3b}$$

$$= \frac{2c}{b^2} \int_0^{b^2} \exp(-ct)(\frac{t}{b} - \sqrt{t}), \tag{B.3c}$$

where equalities (B.3a) and (B.3b) follow from the Leibniz integral differentiation and the integration by part rules, respectively. It is easy to check that integrand $\exp(-ct)(\frac{t}{b} - \sqrt{t})$ is always negative for any member of set $\{(b, c, t) \mid 0 < b, 0 < c, 0 \leq t \leq b^2\}$; therefore, we always have $\frac{\partial g(b,c)}{\partial b} < 0$.

Moreover, for $\frac{\partial g(b,c)}{\partial c}$,

$$\frac{\partial g(b,c)}{\partial c} = \int_0^{b^2} -t\exp(-ct)(\frac{1}{b\sqrt{t}} - \frac{1}{b^2})dt = \frac{-1}{b}\int_0^{b^2} t\exp(-ct)(\frac{1}{\sqrt{t}} - \frac{1}{b})dt.$$

It is again easy to check that the integrand $t\exp(-ct)(\frac{1}{\sqrt{t}} - \frac{1}{b})$ is positive for any member of set $\{(b, c, t) \mid 0 < b, 0 < c, 0 \leq t \leq b^2\}$. Therefore, $\frac{\partial g(b,c)}{\partial c}$ is always negative. ■

By Lemma B.1, expectation function

$$\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x}, \mathbf{x}')) = \mathbb{E}_{x,x'}(\exp(-2\gamma(x - x')^2) \tag{B.4}$$

is a monotonically decreasing function of γ and b. This means that there are only two ways to increase expectation $\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x}, \mathbf{x}'))$, which are either decreasing γ or decreasing b. The approximation of expected error function (2.22) on domain (B.1) and sampling distribution (B.2) for varying values of γ and b and a fixed value of m_s using a heat map plot is shown in Figure B.1. We observe that as the values of γ or b decrease, or equivalently, $\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x}, \mathbf{x}'))$ increases, the approximation of the expected error function decreases.

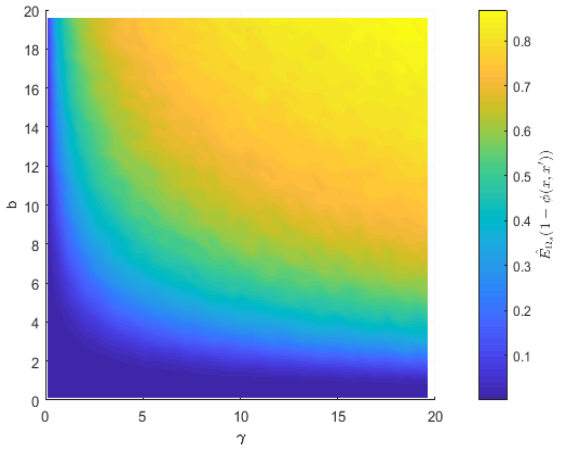


Figure B.1: Heat map of the approximation of expected error function (2.22) on domain (B.1) and sampling distribution (B.2) for varying values of γ and b and a fixed value of m_s

Our simulation study can be used to infer a more general case. Consider the covariance function as $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp(-\sum_{k=1}^{p} \gamma_k(x_k - x'_k))$ defined on Ω_s as a p-dimensional hyper-rectangle with side lengths $b_1, \ldots, b_p$ with a uniform sampling distribution, i.e., $x_k \sim \mathcal{U}(a_k, a_k + b_k) \quad \forall \mathbf{x} \in \Omega_s$. With this setup, we can write

$$\mathbb{E}_{\Omega_s}(\mathcal{K}^2(\mathbf{x}, \mathbf{x}')) = \prod_{k=1}^{p} \mathbb{E}_{x_k, x'_k}(\exp(-2\gamma_k(x_k - x'_k)), \tag{B.5}$$

which is a monotonic function in each b_k and γ_k by lemma B.1. Therefore, our simulation results are valid for this generalized case as well.

APPENDIX C

Solving optimization problem (2.29)

Let first write the partial derivatives of objective function in (2.29),

$$\frac{\partial \mathcal{L}(\bar{\mathbf{a}})}{\partial a_k} = -\mathbf{y}_n^T(\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n)^{-1}\frac{\partial \mathbf{K}_n^{\bar{\mathbf{a}}}}{\partial a_k}(\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{y}_n + \mathrm{tr}((\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n)^{-1}\frac{\partial \mathbf{K}_n^{\bar{\mathbf{a}}}}{\partial a_k}), \quad \text{(C.1)}$$

where $\frac{\partial \mathbf{K}_n^{\bar{\mathbf{a}}}}{\partial a_k}$ is the matrix of element-wise derivatives with respect to the k^{th} element of $\bar{\mathbf{a}}$. Note that each element of $\frac{\partial \mathbf{K}_n^{\bar{\mathbf{a}}}}{\partial a_k}$ involves the term $\frac{1}{\sqrt{1-\bar{\mathbf{a}}^T\bar{\mathbf{a}}}}$. Therefore, the gradient of the objective function in (2.29) does not exist on the boundary of the feasible region, i.e., $\nabla\mathcal{L}(\bar{\mathbf{a}}) \to \infty$ as $\bar{\mathbf{a}}^T\bar{\mathbf{a}} \to 1$. Therefore, to avoid an undefined gradient on the boundary, we modify the optimization by making the feasible region slightly tighter, i.e.,

$$\begin{aligned} \min_{\bar{\mathbf{a}}} \quad & \mathcal{L}(\bar{\mathbf{a}}) = \mathbf{y}_n^T(\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{y}_n + \log|\mathbf{K}_n^{\bar{\mathbf{a}}} + \sigma^2\mathbf{I}_n| \\ \text{subject to} \quad & \bar{\mathbf{a}}^T\bar{\mathbf{a}} \leq 1 - \epsilon, \end{aligned} \quad \text{(C.2)}$$

where ϵ is a very small number. In our experiments, we set $\epsilon = 0.001$.

Due to the simple convex structure of constraint $\bar{\mathbf{a}}^T\bar{\mathbf{a}} \leq 1 - \epsilon$, i.e., a $p-1$-dimensional hypersphere, optimization (C.2) can be solved by the Projected Gradient Descent algorithm [66]. In this projection algorithm, the $(j+1)^{\text{th}}$ decent step is defined by

$$\bar{\mathbf{a}}^{j+1} = \mathcal{P}\big(\bar{\mathbf{a}}^j - \frac{\alpha}{||\nabla\mathcal{L}(\bar{\mathbf{a}}^j)||}\nabla\mathcal{L}(\bar{\mathbf{a}}^j)\big), \quad \text{(C.3)}$$

where $\frac{\alpha}{||\nabla\mathcal{L}(\bar{\mathbf{a}}^j)||}$ is a normalized length step, and

$$\begin{aligned} \mathcal{P}(\mathbf{z}) = \quad & \mathrm{argmin}_{\mathbf{w}}||\mathbf{w} - \mathbf{z}|| \\ \text{subject to} \quad & \mathbf{w}^T\mathbf{w} \leq 1 - \epsilon. \end{aligned} \quad \text{(C.4)}$$

$\mathcal{P}(\mathbf{z}) = \mathbf{z}$, when $\mathbf{z}^T\mathbf{z} \leq 1-\epsilon$, otherwise the solution to $\mathcal{P}(\mathbf{z})$ occurs at the point that the line defined by $\mathbf{z}$ and the center of the hypersphere, ($\mathbf{0}$), crosses the boundary of the hypersphere, i.e, intersection of $\frac{w_1}{z_1} = \frac{w_2}{z_2} = \ldots = \frac{w_{p-1}}{z_{p-1}}$ and $\mathbf{w}^T\mathbf{w} = 1 - \epsilon$. Therefore, the solution to $\mathcal{P}(\mathbf{z})$ has the closed form,

$$\mathcal{P}(\mathbf{z}) = \begin{cases} \mathbf{z} & \mathbf{z}^T\mathbf{z} \leq 1 - \epsilon \\ [\frac{z_1}{\sqrt{\mathbf{z}^T\mathbf{z}}}, \ldots, \frac{z_{p-1}}{\sqrt{\mathbf{z}^T\mathbf{z}}}]^T & \mathbf{z}^T\mathbf{z} > 1 - \epsilon. \end{cases} \tag{C.5}$$

VITA

Babak Farmanesh

Candidate for the Degree of

Doctor of Philosophy

Dissertation: EFFICIENT TECHNIQUES FOR STATISTICAL MODELING OF CALIBRATION AND SPATIO-TEMPORAL SYSTEMS USING GAUSSIAN PROCESSES

Major Field: Industrial Engineering and Management

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Industrial Engineering and Management at Oklahoma State University, Stillwater, Oklahoma in August, 2018.

Completed the requirements for the Bachelor of Science in Industrial Engineering at Sharif University of Technology, Tehran, Iran in 2014.

Experience:

Research Assistant, Industrial Engineering and Management, Oklahoma State University, Stillwater, Oklahoma, Fall 2014-Summer 2018

Intern, Support and services, DELL EMC, Austin, Texas, Summer 2017

Professional Memberships:

Institute of Operations Research and the Management Sciences (INFROMS)

Institute of Industrial and System Engineering (IISE)

Alpha Pi Mu