GAUSSIAN PROCESS MODELING AND COMPUTATION IN ENGINEERING APPLICATIONS

A Dissertation

by

ARASH POURHABIB

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Yu Ding
Co-Chair of Committee,	Faming Liang
Committee Members,	Jianhua Z. Huang
	Satish T.S. Bukkapatnam
Head of Department,	César O. Malavé

August 2014

Major Subject: Industrial Engineering

ABSTRACT

Big Data refers to the complexity, high-dimensionality, and high volume of information which are common features in many contemporary engineering applications. In the context of Big Data, however, specific treatments are required to successfully apply and implement Gaussian processes. This dissertation discusses new methodologies for solving three critical problems: analysis of spatial-temporal systems for wind energy applications; multi-fidelity analysis for nano-manufacturing systems; and predictive modeling for large datasets.

First, we develop a spatial-temporal model for local wind fields in a wind farm with more than 200 wind turbines. Our framework utilizes the correlation among the derivatives of wind speeds to find a neighborhood of predictors. We extend the model to incorporate the wind direction as a variable to define regimes and fit a separate model for each regime. We consider other meteorological measurements, such as air pressure and temperature, by calculating a theoretical wind called the geostrophic wind to enhance the model's predictive power. We present the model in an optimization framework and solve it through numerical techniques. We compare the model's performance with some alternatives in order to demonstrate its prediction accuracy.

Second, we consider a multi-fidelity analysis for predicting the Young's modules of buckypaper, a nano-manufactured product. The data for this problem derive from expensive, but accurate, physical experiments and an inexpensive, but less accurate, simulation model. The practice of integrating such data with different levels of accuracy is called multi-fidelity analysis. The challenge is that some of the input variables in the physical experiments are difficult to measure. We formulate the problem by introducing latent variables and then imputing unobserved latent variables in a two-step process: defining the functional relationship between observed and latent variables, and finding the optimal relationship by minimizing the distance between them. We demonstrate that this problem can be understood as a case of non-isometric curve to surface matching.

Third, we apply Gaussian process regression to large datasets. We propose a Bayesian Site Selection (BSS) approach which approximates the likelihood of the Gaussian process by using unobserved variables called pseudo-inputs. The BSS framework enables us to learn both the number and the location of the pseudo-inputs simultaneously through reversible jump Markov chain Monte Carlo methods. Testing the proposed method on both real and artificial datasets shows that the BSS approach provides a sensible trade-off between the prediction accuracy and computation time.

ACKNOWLEDGEMENTS

I express gratitude to my advisor, Dr. Yu Ding, for the guidance he has provided. This dissertation would not have been possible without your encouragement, advice, and assistance and I remain deeply indebted to you.

I also thank my co-advisor, Dr. Faming Liang, and my committee members, Dr. Jianhua Huang and Dr. Satish Bukkapatnam. All of you have given me support and helped me to develop the skills I needed for independent research. My special thanks to Dr. Huang, for the time you devoted to assisting me throughout the course of my research.

I acknowledge Dr. Abhishek Shrivastava, Dr. Eunshin Byon, Dr. Chiwoo Park, and Dr. Giwhyn Lee, who are former members of the Advanced Metrology Laboratory, and Yanjun Qian and Hoon Hwangbo, who are current members. Thank you all for your insights and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION: GAUSSIAN PROCESSES

The sheer immensity and complexity of the Big Data used to model and analyze contemporary engineering systems present system engineers and data analysts with unique challenges, e.g., determining how to extract only the pertinent knowledge. This study examines the use of Gaussian processes (GPs), which are based on the prevalent concept of normal, or Gaussian, distribution for natural phenomena.

This section serves as an introduction to GPs. We present a short history and give the basic definitions. Here, we focus on two classes of models involving GPs: GP regression that mainly concerns predictive modeling for systems with datasets having arbitrary dimensions, and Gaussian Markov processes for the special case of one-dimensional data, especially systems that evolve over time.

1.1 Short History of Gaussian Stochastic Processes

Modeling natural phenomena using Gaussian distribution has long been used in science and engineering. In the eighteenth century, de Moivre gave the first formal definition of normal distribution, which was named "Gaussian" after Johan Carl Friedrich Gauss, who studied it extensively in the nineteenth century (Le Cam and Yang, 2000). Informally, a GP can be understood as a sequence of random variables each and jointly following the Gaussian distribution. Originally, GPs were developed to study one-dimensional time-series models (Wiener, 1964) and later they became central to statistical machine learning. Krige (1951) introduced an application of GPs as an interpolation technique in his attempt to identify the potential locations of gold mines. This brought attention to, and popularity, for GPs beyond time-series applications, e.g., the rigorous mathematical theorems and proofs proposed by Matheron (1973). O'Hagan (1978) studied GPs for curve fitting and optimal predictions.

A new era began for GPs when their application was introduced into the design and analysis of computer experiments (Sacks et al., 1989). As GPs became a core part of machine learning, several books were published on different aspects of GPs, such as Rasmussen and Williams (2006), Santner et al. (2003), and Kleijnen (2007). Today, GPs prevail as a powerful tool in the era of Big Data. Emerging applications and variations include (Hensman et al., 2013; Damianou and Lawrence, 2013; Clifton et al., 2013; Zhu and Dunson, 2013; Pérez-Cruz et al., 2013).

1.2 Gaussian Process Regression

In this study, we are interested in the application of GPs in statistical learning. *Statistical learning* concerns identifying patterns, or building predictive models, from collected data (Hastie et al., 2001). *Supervised learning* estimates the functional relationship between a set of inputs and its corresponding set of outputs. If the outputs come from a discrete set, we call the problem classification, whereas if they potentially come from a continuous set, such as real variables, we call the problem regression. Here, we briefly review regression based on GPs adopted from Rasmussen and Williams (2006) (see Rasmussen and Williams (2006); Santner et al. (2003) and Kleijnen (2007) for detailed discussions).

In a regression context, suppose we observe data points in the form of $(\mathbf{x}_i, y_i)$, where $\mathbf{x}$ is the vector of input variables or explanatory/predictive variables, y is the output, and i is the index of observations. If we observe N data points, we denote the inputs by $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, and the outputs by $\mathbf{Y} = \{y_1, y_2, \ldots, y_N\}$, which we assume are independent and identically distributed (i.i.d). We call the combined $(\mathbf{X}, \mathbf{Y})$ the observed data, or simply data, as denoted by $\mathcal{D}$. In the regression, we assume there exists a function, or the ground truth, f, such that $y_i = f(\mathbf{x}_i) + \epsilon_i$, for

$i = 1, 2 \ldots, N$, where ϵ_i denotes the observation noise, or any other discrepancy that cannot be captured by the model, usually assumed to follow a normal distribution with mean 0 and variance σ^2. We consider the case where each $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ for some $d \geq 1$ and $y_i \in \mathbb{R}$.

To find function f, we impose extra constraints on the model in order to obtain a well-defined problem. GP regression is a powerful tool for inferring such functional relationships while maintaining good interpretability and model flexibility. There are many ways to describe the GP regression, such as conceiving the weight-space view or the function-space view. The former is derived by extending the simple linear regression into a high-dimensional feature space, whereas the latter is derived by directly defining distribution over functions (we note that the function-space view provides a more general framework). Specifically, from the function-space view, a GP is a continuous stochastic process in which any finite number of variables have a joint Gaussian distribution (Rasmussen and Williams, 2006, p. 13). Based on the function-space view, the random variables are simply $f(\mathbf{x})$ – the "true process"– as we defined above. In other words, in the context of stochastic processes, we have an index set, $\mathcal{X} \subset \mathbb{R}^d$, that is partially ordered according to the observed input set, $\mathbf{X}$, and the value of $f(\mathbf{x})$ is a random variable. An analogy to the Gaussian distribution demonstrates that we can specify a GP by its mean and covariance functions

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbb{E}\left\{f(\mathbf{x})\right\}, \\ K(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left\{(f(\mathbf{x}) - \mu(\mathbf{x}))\left(f(\mathbf{x}') - \mu(\mathbf{x}')\right)\right\}, \end{aligned} \tag{1.1}$$

where $\mathbb{E}\{.\}$ is the expectation operator. Note that the mean and the covariance functions are specified in terms of the inputs, $\mathbf{x}$ and $\mathbf{x}'$. It is also possible to define the parametric forms for the covariance function, e.g., a simplified form of the squared exponential covariance

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\eta}\right), \tag{1.2}$$

where η and σ_f^2 are, in fact, the "parameters" of the model which we can collectively denote by $\boldsymbol{\theta}$.

In practice, the ultimate goal in regression is to find the predictive distribution of the process at some test point, $\mathbf{x}_* \in \mathcal{X}$. From a decision theory perspective, we can minimize the expected loss, specifically, the predicted value at $\mathbf{x}_*$ is

$$y_* = \arg\min_y \int \mathcal{L}\left(y_*, y\right) p(y_*|\mathbf{x}_*, \mathcal{D})\, dy_*, \tag{1.3}$$

where $p(y_*|\mathbf{x}_*, \mathcal{D})$ is the predictive distribution of the response at $\mathbf{x}_*$ given the data, and $\mathcal{L}(.,.)$ is the loss function.

First, however, we want to fit the model to the data, or from a statistical learning perspective, we want to "learn" the parameters in the model. Generally in the GP regression, we estimate the parameters of the covariance function by maximizing the marginal log-likelihood $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$. Having done so, we use the fully specified model to obtain the probabilistic prediction at any test point, $\mathbf{x}_* \in \mathcal{X}$.

1.3 Gaussian Markov Processes

Recall that we are interested in the special case of one-dimensional data, especially systems that evolve over time. Examples include the temperature of an object undergoing a chemical reaction, or the wind speed at a specific site. In many practical applications for such processes, the current response is a function of the past responses of the process. We can study this by using autoregressive models which assume linear dependency between the system's responses plus some discrepancy modeled through a random noise. For example, if we denote the system's response at time t by y_t, a (discrete-time) autoregressive model of order p is

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \epsilon_t, \tag{1.4}$$

where ϵ_t is a "white noise" following $\mathcal{N}(0, \sigma^2)$, and α_i, $i = 1, 2, \ldots, p$ are the autoregressive coefficients.

Knowing that many of the systems that change over time are continuous prompts us to try modifying equation (1.4) to explicitly model a *continuous* system like our case example of a wind farm. The extension of equation (1.4) for the continuous case is not trivial, so we determine the correct generalization by solving a Stochastic Differential Equation (SDE), which gives rise to a continuous autoregressive model (Rasmussen and Williams, 2006). Thus, the SDE we need to solve is

$$0 = \sum_{i=1}^{p} \alpha_i y_t^{(i)} + \alpha_0 y_t + \epsilon_t, \tag{1.5}$$

where $y_t^{(i)}$ denotes the ith derivative of the process $y(.)$ at time t. Note that because $y(.)$ is a stochastic process, we cannot define its derivative, and as a result, equation (1.5), using the classical concept of the derivative for a (deterministic) function, yet we need separate theories in order to rigorously define such operators and processes (Øksendal, 2003). Therefore, we review a heuristic way of building a derivative function for a special stochastic process known as the standard Brownian motion. We also can apply similar treatments to the other (one-dimensional) stochastic processes we consider, but for the purpose of this study, a heuristic construction is sufficient.

First, we want to know if we can use model (1.4) to describe a system that follows equation (1.5), but we only observe its status at discrete times. We note that the answer generally appears to be negative. Moreover, deriving a rigorous relationship between continuous and discrete autoregressive models (equations (1.5) and (1.4), respectively), entails Fourier Analysis, and certain regulatory conditions need to hold in order for the discrete sampling of a continuous autoregressive system to engender a discrete autoregressive model (see Rasmussen and Williams (2006, pp. 207-219) for details). However, we assume that a discrete sampling of the continuous process described by (1.5) indeed will result in a discrete autoregressive model (1.4). We note that this statement holds if $p = 1$ (Rasmussen and Williams, 2006).

The special case of equation (1.5) when $p = 1$ gives

$$\alpha_1 y_t^{(1)} = \alpha_0 y_t + \beta \epsilon_t, \tag{1.6}$$

which is a continuous counterpart for an autoregressive model of order $p = 1$. We can use this very simple case to define the Gaussian Markov processes for modeling the wind speeds at our wind farm. Therefore, we investigate equation (1.6) in more detail. Recall that for a continuous stochastic process to be a Markov process we need to have

$$\mathbb{E}\{y_t|\mathcal{F}_s\} = \mathbb{E}\{y_t|y_s\}, \quad \forall\, 0 \leq s < t, \tag{1.7}$$

where $\mathcal{F}_s$ is a sigma-algebra generated by y_s for all $s < t$ (Hunter, 2009). This statement means that given the current value of the process, the “upcoming” value is independent of the history of the process. A Gaussian Markov process is simply a (one-dimensional) GP that also satisfies the Markovian property.

For better understanding, we can express equation (1.6) in another equivalent form. First, note that we can define white noise, ϵ_t, as a stochastic derivative of the standard Brownian motion. Since the standard Brownian motion on $\mathbb{R}$ is a GP whose increments are independent Gaussian random variables, this motion is a Gaussian Markov process, and the Markovian property is due to the fact that the increment should follow the Gaussian distribution. However, the standard Brownian motion is *not* stationary and, in fact, the covariance between two values at times t and s is $\min(t, s)$. Heuristically, we build the relation between the standard Brownian motion and ϵ_t by first defining a new stochastic process as the difference between two "very close in time" values of a Brownian motion (Hunter, 2009). Let B denote a standard Brownian motion, and then define

$$\epsilon_{\Delta t}(t) = \frac{B(t + \Delta t) - B(t)}{\Delta t}; \tag{1.8}$$

therefore $\epsilon_{\Delta t}(t)$ is a GP with mean 0 and variance $(\Delta t)^{-1}$. If we find the limit of $\epsilon_{\Delta t}(t)$ as $\Delta t \to 0$, we get a stochastic process with mean 0 and variance between two different values at times t and s as $\delta(t - s)$, where δ is the Kronecker delta function. This description and the white noise we described above are the same. This approach also demonstrates a heuristic way to build the derivative of the stochastic processes. Therefore, $\epsilon = \frac{dB}{dt}$. For more rigorous construction of the Brownian motion and the white noise see (Øksendal, 2003).

Having established the relation between the Brownian motion and the white noise, we can express equation (1.6) as

$$dy = -aydt + bdB, \tag{1.9}$$

which is the Ornstein-Uhlenbeck (OU) process (Øksendal, 2003). As $t \to \infty$, the solution of this process approaches a stationary Gaussian Markov process. Its expression in stochastic form allows us to intuitively understand equation (1.9). Specifically, the rate of change in the response consists of one part that is explained by the current value of the process, and a second part following a random path that accounts for the measurement noise or any other parts not captured by the model.

In Section 2 we will utilize an extension of equation (1.9) to model the local wind fields. Specifically, if y denotes the wind speed a a target location, we want to utilize the information in its vicinity, which is captured through other wind speed measurements. Therefore, we have

$$dy = -aydt + f(\mathbf{u}(y)) + bdB, \tag{1.10}$$

where $\mathbf{u}(y)$ consists of wind speed measurements around the target location and f is a function. We are not concerned about the theoretical properties of equation (1.10); instead we utilize it towards devising a predictive model for local wind fields.

1.4 Research Objective and Outline

This study addresses three problems characterized by having two features in common: (1) Big Data, i.e., large datasets or complex data; (2) Gaussian processes or a variation, in their modeling and analysis. Below, we describe the three problems and our research objectives.

1.4.1 Problem 1: Spatial-temporal Dynamics of Local Wind Fields

The latest IPCC findings (https://www.ipcc.ch/report/ar5/index.shtml) add urgency to global efforts to increase wind power generation. Expanded use of this important renewable resource, however, will not occur until we develop more effective methods, e.g., improving turbine capacity, to drive down the cost of operations and maintenance (O&M) (Blanco, 2009). Solving significant technological issues requires addressing wind dynamics including intermittency and volatility that result in high variability of the energy produced.

Wind forecasters typically develop their wind speed/power predictions based on aggregating all of the data collected from a wind farm. In this study, we argue that collecting data "turbine by turbine" rather than aggregating data from the whole farm will produce relatively accurate forecasts, and that with such knowledge we can tailor very specific, and less costly, O&M strategies. To solve the problem of analyzing the dynamics of local wind fields, we construct a set of spatial-temporal models that relate closely to Gaussian Markov processes. We use a dataset of near ground wind speeds collected over two years from a wind farm with 200 turbines in Illinois, and consider both historical wind speed data and other meteorological measurements. We present the models in the form of optimization problems for large datasets and solve them through efficient parametrization.

1.4.2 Problem 2: Multi-fidelity Analysis Based on Latent Variables

Analysis of many Big Data problems generally uses data drawn from many sources, e.g., a simulation model and physical observations, or a variety of sensing devices. Multi-fidelity analysis, which refers to the analysis of such data structures, requires addressing each specific application and its associated considerations in order to obtain a comprehensive analysis. We are interested in analyzing a multi-fidelity system having some input variables that cannot be measured, leaving only a physical dataset with partially observed inputs. The lack of observability of some variables implies that we cannot appeal to existing multi-fidelity analyses, since virtually all of them assume that all variables are observable. We utilize GP regression to construct a simulation model that is the low-fidelity experiment. We apply our model to analyze buckypaper, a nano-manufactured product.

1.4.3 Problem 3: Approximating GP for Large Datasets

Traditionally, GP regression has been popular for modeling complex engineering systems, due to its flexibility and non-parametric nature. However, its application in large datasets is limited, because it does not scale well. To approximate GP regression, current approaches infuse sparsity into the model, or use likelihood approximation. We are interested in improving the likelihood approximation in order to obtain a a sensible balance between prediction accuracy and computation time. We propose a Bayesian Site Selection (BSS) model for approximating the GP regression. BSS utilizes a set of unobserved variables to approximate the likelihood of the GP model. We solve BSS by employing reversible jump Markov chain Monte Carlo. Applying BSS to several large scale datasets, we demonstrate that our model is efficient in tackling Big Data while simultaneously producing reasonably accurate prediction results with less computation.

1.5 Organization of Dissertation

The remainder of this dissertation is organized as follows. Section 2 describes the first problem, modeling the spatial-temporal dynamics of local wind fields. We explain how to model the problem by utilizing a variation of Gaussian Markov processes with discrete sampling. We propose to use spatial-temporal parametrization to solve it efficiently while maintaining the interpretability of the model. We test the model by using two years of data compiled for ground level wind speeds and analyze the results.

Section 3 describes the second problem, calibration in the nano-manufacturing process of buckypaper fabrication. We use the GP regression to model an existing simulation model so that the GP surface serves as a surrogate model. We demonstrate how to calibrate such simulation models in the context of partially observable inputs.

Section 4 describes the third problem, employing the GP regression on large datasets. We propose a Bayesian framework to improve the accuracy of existing likelihood approximation techniques. We solve the problem using reversible jump Markov chain Monte Carlo. We test the model on datasets having more than 50,000 data points and describe the results.

Section 5 summarizes our findings and suggests some future research pursuits.

2. SHORT-TERM WIND SPEED FORECAST USING MEASUREMENTS FROM MULTIPLE TURBINES IN A WIND FARM

2.1 Introduction

Interest in methods for achieving more economic wind power production (Blanco, 2009) hinges upon an enhanced comprehension of the behavior of near ground wind, the major input force to wind power systems. Modeling the spatial-temporal aspects of wind can facilitate many practical aspects related to operations including the analysis of wake effect (Crespo et al., 1999), maintenance and control, and optimal siting of turbines.

In particular, accurate prediction of wind speeds can lead to more confident estimations of the wind energy produced, which energy producers and regulators can utilize to improve overall wind energy supply. An improved understanding of wind dynamics also can allow system operators to incorporate greater amounts of wind energy into the grid. In this section, we focus on wind speed prediction due to its significance and practical importance. In brief, we categorize wind speed forecasting by short-term, medium-term, and long-term. There is no sharp division between medium- and long-term forecasts, which can range from days to years. Short-term forecasts rely on efficiently utilizing past wind speed measurements due to the high volatility of short-term wind and the computational cost of running physics-based numerical weather prediction (NWP) methods (Hering and Genton, 2007). A common consensus is that within six hours of a prediction horizon, data-driven, statistical models can outperform the physics-based NWPs, and that beyond six hours, forecasts without considering atmospheric physics cannot be trusted (Giebel et al., 2011). Therefore, the six-hour boundary is the separator that differentiates short-term from medium- and long-term forecasts.

Producers use short-term wind forecasts as guides for adjusting their wind energy supply in order to meet the demand in a short horizon. The significance of the short-term wind forecast has encouraged substantial research by engineers and statisticians (see Giebel et al. (2011) and Zhu (2013) for comprehensive reviews of the work done in the past three decades). Despite numerous efforts to build predictive models for wind speed forecasting, spatial information is much less frequently used than temporal information in most short-term forecasts.

This section presents our spatial-temporal models for analyzing the behavior of near-ground wind. Our technical objective is to use the *in-situ* measurements at multiple turbine locations within the same wind farm, which constitute a multiple time-series. We model the relationship of the wind speed at any given turbine location with surrounding wind turbines in the farm. One contribution of our work is a novel way to adaptively determine the neighborhood for a target site, leading to a sparse representation of the spatial dependency unique to each turbine. We also model the variance of the wind speed as a function of volatility in its neighborhood, and use the predictive distribution to quantify the uncertainty associated with the prediction. We incorporate two existing ideas into the model to enhance its performance: (1) a regime-switching approach to account for the dominant wind directions; and (2) meteorological measurements, such as temperature and air pressure, as inputs. We test our proposed models on real data from a wind farm with more than 200 turbines and multiple meteorological mast towers.

The remainder of this section is organized as follows. Section 2.2 reviews some of the widely used models for short-term wind forecasting. Section 2.3 presents the proposed model and discusses the selection of appropriate loss functions. Section 2.4 applies the proposed methods and compares their performance with some other practices. Section 2.5 summarizes the research findings in this section.

2.2 Review of Existing Short-term Wind Forecast Methods

To start, we briefly review some widely used models for short-term wind forecasting (see Giebel et al. (2011) and Zhu (2013) for comprehensive reviews). Let $Y_i(t) = Y(\mathbf{s}_i; t)$ denote the wind speed at time t measured at location $\mathbf{s}_i$ for $i = 1, 2, \ldots, I$. Use the vector notation $\mathbf{Y}(t) = [Y_1(t), Y_2(t), \ldots, Y_I(t)]^T$. Assume hour for the resolution for time. Now, suppose we observe the wind speed at locations $\mathbf{s}_i$ for $i = 1, 2, \ldots, I$ for times $t = 1, 2, \ldots, T$ and then we want to make an h-step ahead prediction that is $\widehat{Y}_i(t+h)$. In general, h is between one and six hours.

Statistical models developed for short-term wind forecasts can be categorized into temporal and into spatial-temporal models. The basic idea behind temporal models is that wind speed at each time is affected by the wind speed in its near past and that knowledge can be employed to build time-series modelsfor the short-term forecasts. The plethora of time-series models has created remarkable and rich literature for short-term wind forecasts, particularly, autoregressive models (Schlink and Tetzlaff, 1998; Brown et al., 1984; Huang and Chalabi, 1995), autoregressive moving average (Torres et al., 2005; Erdem and Shi, 2011), and autoregressive integrated moving average (Palomares-Salas et al., 2009) have all been used on wind data for the short-term forecast. Spatial-temporal models hinge upon the idea that the wind characteristics of a region resemble the characteristics of neighboring regions. This idea has encouraged researchers to take into account the spatial dependency of wind in their model building (Gneiting et al., 2006; Hering and Genton, 2010). In this part, we present some of the main ideas in each of these two classes.

2.2.1 Temporal Models

This class of methods build individual models for each respective time series $\{Y_i(t) : t = 1, 2, \ldots, T\}$, for $i = 1, 2, \ldots, I$. The simplest case known as the persistence forecasting simply assumes

$$\widehat{Y}_i(t+h) = Y_i(t), \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.1}$$

that is, the wind speed "persists" over time for the following h hours. The persistence method has been considered as a reference model in the literature (Giebel et al., 2011). Despite its simplicity, we will later show that the persistence method performs relatively well for short forecast horizon such as 2 or 3-hour.

More sophisticated methodologies can be employed in an attempt for better forecasting results. For instance, an autoregressive moving average model of order (p, q), denoted by ARMA(p,q), is a popular choice. The model can be expressed as:

$$Y_i(t) = c + \sum_{\ell=1}^{p} \phi_\ell Y_i(t-\ell) + \sum_{\ell=1}^{q} \theta_\ell \epsilon(t-\ell) + \epsilon(t), \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.2}$$

where c is a constant, ϕ_ℓ and θ_ℓ are the autoregressive and moving average parameters, respectively, and $\epsilon(t) \sim (0, \sigma^2)$ for $t = 1, 2, \ldots, T$. A special case is when $q = 0$; this results in an autoregressive model of order p, denoted by AR(p). AR models have been popular in short-term wind forecast. Particularly Brown et al. (1984), by explicitly considering non-Gaussian distribution and diurnal non-stationarity, found that AR(1) and AR(2) models outperform the persistence method for hourly data. In general, low-order AR models are considered the most suitable type of AR models for short-term forecast (Katz and Skaggs, 1981; Huang and Chalabi, 1995). However, successful implementation of higher-order AR models for short-term forecast has also been reported, for instance, by Schlink and Tetzlaff (1998).

ARMA models have also been used for short-term forecast. Torres et al. (2005) showed that ARMA(p,q) with $p = 1$ and $q \leq 4$ outperforms the persistence method for most cases over five sites with different terrain features. Specifically, they found that the root mean squared error for one hour ahead forecast is, on average, 2 to 5% less than that of the persistence method, and when the prediction horizon is larger, the difference is more pronounced (up to 20% for ten-hour ahead forecast). Kamal and Jafri (1997) showed that ARMA(p,q) with a higher order for p (up to $p = 5$) can produce accurate results for minutes ahead forecast using data with resolution of minutes. Daniel and Chen (1991) use the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) to choose the values of p and q in an ARMA(p,q) model and found that $p = 2$ and $q = 0$ would be selected based on those criteria, which in fact reduces to an AR(2) model.

Another linear model as an alternative to AR(p) is Kalman Filter (KF) (Kalman, 1960), applications of which in short term wind forecast can be found in Louka et al. (2008) and Crochet (2004). KF has also been used in combinations with other methods such as the Kalman Filtering of Numerical Weather Predictions output (Cassola and Burlando, 2012) as well as hybrid time series Kalman Filter models (Liu et al., 2012). Also, more sophisticate AR models to handle handle seasonal fluctuations such as Markov-Switching Autoregressive models has been used to model wind speed time-series (Ailliot and Monbet, 2012). Furthermore, there is research that focuses on forecasting the wind power directly include (Pinson, 2012; Bessa et al., 2012; Wan et al., 2014). Nevertheless, these works essentially follow a similar approach with the distinction that they entail modeling the relation between the wind speed (and other meteorological variables) and wind power.

2.2.2 Spatial-temporal Models

While the use of spatial information is a standard treatment in physics-based NWPs, temporal models dominate the statistical approaches for short-term wind forecast. Spatial-temporal models developed in the past decade were encouraged by the fact that wind speed at some location is informative of wind speed at its vicinity. Of course the physical properties of a region dictate how the dependency manifests itself. In general, for flat regions we expect wind characteristics to be positively correlated, while, on the other hand, even in small regions, erratic changes in altitude would render spatial information for wind speed and direction useless (Zhu and Genton, 2012). Therefore, spatial-temporal models not only consider the historical wind data at the reference site, but also explicitly employ the data at other locations.

To take spatial dependency into account, a straightforward extension of the AR(p) model that yields Vector AR(p) (Johansen, 1995), or simply VAR(p), which is defined as

$$\mathbf{Y}(t) = \mathbf{c} + \sum_{\ell=1}^{p} \mathbf{\Psi}_\ell \mathbf{Y}(t-\ell) + \boldsymbol{\epsilon}(t), \tag{2.3}$$

where $\mathbf{c}$ is an $I \times 1$ constant vector, $\mathbf{\Psi}_\ell$ is an $I \times I$ matrix of autoregressive coefficients for $\ell = 1, 2, \ldots, p$, and $\boldsymbol{\epsilon}(t)$ is the $I \times 1$ error vector such that $\mathbb{E}\{\boldsymbol{\epsilon}(t)\} = \mathbf{0}$, $\mathbb{E}\{\boldsymbol{\epsilon}(t)\boldsymbol{\epsilon}(t)^T\} = \mathbf{\Omega}$, where $\mathbf{\Omega}$ is a diagonal matrix with non-negative entries, and $\mathbb{E}\{\boldsymbol{\epsilon}(t)\boldsymbol{\epsilon}(t-k)^T\} = \mathbf{0}$, for $k \neq 0$. One can generalize the VAR(p) model to a Vector AR with a moving average part, which becomes a Vector ARMA, or VARMA (Mauricio, 1995)

$$\mathbf{Y}(t) = \mathbf{c} + \sum_{\ell=1}^{p} \mathbf{\Psi}_\ell \mathbf{Y}(t-\ell) + \sum_{\ell=1}^{q} \mathbf{\Xi}_\ell \boldsymbol{\epsilon}(t-\ell), \tag{2.4}$$

$\mathbf{\Xi}_\ell$ is an $I \times I$ matrix of moving-average coefficients for $\ell = 1, 2, \ldots, q$. To explicitly include spatial-temporal dependency, a parametrized from of model (2.4) yields spatio-temporal autoregressive moving-average (STARMA) (Cressie and Wikle, 2011, p. 344):

$$\mathbf{Y}(t) = \sum_{k=0}^{p} \left(\sum_{j=1}^{\lambda_k} f_{kj} \mathbf{U}_{kj} \right) \mathbf{Y}(t-k) + \sum_{\ell=0}^{q} \left(\sum_{j=1}^{\mu_\ell} g_{\ell j} \mathbf{V}_{kj} \right) \mathbf{W}(t-\ell) \tag{2.5}$$

where $\mathbf{U}_{kj}$ and $\mathbf{V}_{kj}$ are known weight matrices, and p and q are the orders of the autoregressive part and the moving average part, respectively, f_{kj} and $g_{\ell j}$ are the model parameters, and $\{\mathbf{W}_t\}$ are i.i.d random vectors with mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma}_W$. Estimating the parameters in such models is, however, not a straightforward undertaking.

An application of VAR(p) for wind speed forecast can be found in de Luna and Genton (2005). Other models utilizing spatial information include the regime-switching approach proposed by Gneiting et al. (2006) which utilizes data at three different sites in northern Oregon and southern Washington. The regime in this context refers to the dominant direction of the wind; in their particular case it is either towards the east or towards the west. Hering and Genton (2010) propose an extension of this model in which the wind direction also serves as a predicting variable, as opposed to merely defining the regime. Other works considering using data at nearby sites include Alexiadis et al. (1999) and Kusiak and Li (2010) where both use Artificial Neural Networks (ANNs). In addition, models have been proposed which directly address the problem of forecasting the wind power generation using spatial information (Tastu et al., 2014).

In the most aforementioned spatial-temporal approaches for the wind speed forecast, multiple measurements come from relatively large regions for a small number of locations, i.e. geographically dispersed information. This is different from the problem of our interest in which we explicitly consider a set of measurements in a relatively small region for a large number of locations, specifically within a wind farm. Therefore, the problem resembles some of the cases investigated by Alexiadis et al. (1999) and Kusiak and Li (2010); however, we try to model spatial dependency using a continuous and interpretable field unlike utilizing ANNs which lack physical interpretation. Furthermore, our goal is to make individual speed predictions at individual turbine locations, as we argue the impact of this knowledge can transcend mere power generation forecast and can be utilized for better control of wind power systems; specifically, we need turbine-specific wind forecasts to be able to effectively control the damage created by the fatigue on the structural components (Santos, 2007). Hence, this work is different from those that consider individual turbine measurement for an overall power forecast of the whole wind farm (He et al., 2013). Hence, selection and modeling such spatial dependency in local wind fields for individual turbine present unique challenges which will be discussed in detail in Section 2.3.

2.3 Spatial-temporal Auto Regression For Short-term Wind Forecast

In this part, we present the details of our approach for modeling the wind speed in a local wind field with wind measurements available at multiple locations. To explicitly consider the variability in the wind speed, we assume the wind speed at each location is a random variable. Different distributions have been proposed in the literature to model this random variable, including Weibull (Yu and Tuzuner, 2008) and truncated normal (Gneiting et al., 2006), among which the truncated normal has proven powerful for the purpose of short-term wind forecast. In particular, a truncated normal can model non-negativity of the wind speed and also its quantiles can be easily computed. Therefore, we assume wind speed $Y_i(t) = Y(\mathbf{s}_i; t)$ follows a truncated normal distribution $\mathcal{N}^+(\mu_i(e_t), \sigma^2(e_t))$ at time t at location $\mathbf{s}_i$ for $i = 1, 2, \ldots, I$ (location $\mathbf{s}_i$ will be often shortened as location i), where e_t denotes the "epoch" at time t, i.e. a section of days in a period of time in which the wind speed can be assumed stationary (He et al., 2013). Epochs are introduced as a mean to model the nonstationarity of wind speed. For example, we may consider 6a.m. to 12p.m. in the month of January as an epoch. The main goal here is to develop a model for the parameters of the truncated normal distribution by considering both temporal and spatial dependency in the field. Here, we first present three models to analyze the dynamics of wind in local regions, and then address the loss function issue.

2.3.1 Gaussian Spatial-temporal Auto Regression

Towards devising a model for the wind speed, we note that single time-series models can be derived from the assumption that wind speed is some (parametric or non-parametric) function of the past wind speed values. Considering the spatial dependency of the wind speed, we extend this understanding by assuming that the mean of the wind speed is a function of the means of the wind speeds at not only the target site but also other locations in that region. Of course, to handle the problem in practice we need to be more specific about this relationship. A simple extension of single time-series modeling for efficiently incorporating the spatial dependency suggests a linear form for this dependency, namely that the mean of the wind speed over time at location i is expressed as a linear combination of a group of mean speed values, estimated by observed wind speeds, at locations $J_i \subset \{1, 2, \ldots, I\}$.

Specifically,

$$\mu_i(e_t) = c + \sum_{\ell=0}^{p} \sum_{j \in J_i} a_{ij\ell} Y_j(t-\ell), \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.6}$$

where c is a constant, p represents the history of time which can be informative to model the mean of the distribution, $a_{ij\ell}$ are the parameters which show the spatial-temporal dependency, and h is the prediction horizon. Here, the order of temporal part p is fixed and we will discuss its determination in Section 2.4.4. Furthermore, we delay the discussion about the selection of time epochs, e_t, to Section 2.4.

Dealing with large-scale datasets which influence the model through J_i, instead of directly estimating all the unknown coefficients in equation (2.6) we proceed by imposing a natural structure on the spatial-temporal coefficients through parameterization. Furthermore, we adaptively select the neighborhood size such that most important information is captured via employing a smaller number of locations. These tasks, while maintaining the model interpretability, facilitate the solution procedure enormously as will be explained in detail.

We assume the spatial-temporal parameters $a_{ij\ell}$ can be decomposed into the respective spatial and temporal parts. That is

$$a_{ij\ell} = a^s_{ij} a^t_{i\ell} \qquad \text{for} \quad i = 1, 2, \ldots, I, \quad j \in J_i, \quad \ell = 1, 2, \ldots, T. \tag{2.7}$$

A key observation in modeling the spatial parameter a^s_{ij} is that wind speeds at closer geographic proximity contribute more in explaining the change in the wind speed at the target site, while wind speeds at faraway locations could also be informative but to a lesser degree. One way to model this type of dependency is through a Gaussian kernel. Specifically,

$$a_{ij}^s = \exp\left[-\left(\mathbf{s}_i - \mathbf{s}_j\right)^T \boldsymbol{\Lambda}_i \left(\mathbf{s}_i - \mathbf{s}_j\right)\right], \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.8}$$

where $\boldsymbol{\Lambda}_i = \mathrm{diag}\{\lambda_{i1}, \lambda_{i2}\}$, and λ_{i1} and λ_{i2} are parameters modeling the spatial decay in the longitudinal and latitudinal directions, respectively. In other words, this Gaussian kernel assigns "weights" to different locations and the weight continuously diminishes as the distance increases. Therefore, this modeling strategy creates a contentious spatial field; this spatial field is achieved by replacing the spatial part of the coefficients in equation (2.6) by new location-specific parameters $\boldsymbol{\Lambda}_i$.

For the temporal part $a_{i\ell}^t$ we can make a similar argument, i.e., an exponential delay in weighting but in terms of time distance. This leads to the following equation

$$a_{i\ell}^t = \exp\left[-\lambda_{i3}\ell\right], \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.9}$$

where λ_{i3} is a parameter modeling the temporal decay. The subscript "3" means to include the time as the third dimension in the input space.

Our goal is to use data to decide both J_i and $a_{ij\ell}$ for each location i. Let $\mathbf{A}_i$ denote an $I \times p$ matrix of spatial dependency for location i. This means if $j \in J_i$ the (k, ℓ) entry of the matrix is a_{ij}^s, otherwise it is zero. We also define matrix $\mathbf{D}_i$ as an $p \times p$ diagonal matrix whose $(\ell, \ell)^{\text{th}}$ entry is $a_{i\ell}^t$. Then, equation (2.6) can be written as

$$\mu_i(e_t) = c + \mathrm{tr}\left(\mathbf{A}_i \mathbf{D}_i \mathcal{Y}^T(t)\right), \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.10}$$

where $\mathcal{Y}$ is an $I \times p$ matrix whose ℓ^{th} column is $\mathbf{Y}(t-\ell)$, and the superscript ^{T} denotes the transpose. We call model (2.10) Gaussian Spatial-temporal Auto-regression of order p and denote it by GSTAR(p).

To estimate the parameters in equation (2.10) we follow a regularized least square estimation procedure. The regularized least square estimation, as the name suggests, is an optimization framework consisting of two parts: we want to find the parameters which minimize the discrepancy between the observations and the model prediction through, and also, we add a penalty to avoid overfitting in both time and space. In fact, one purpose of considering a penalty term is to find the "best" neighborhood J_i while keeping the size of the optimization problem tractable.

Let us first present the optimization formulation and delay the discussion of how the optimization selects the neighborhood to Section 2.3.2. Specifically, given p, we consider the optimization problem

$$\min U(\lambda_{i1}, \lambda_{i2}, \lambda_{i3}) = \sum_{\ell=1}^{T} \mathcal{L}\left\{Y_i(\ell+h) - \bar{Y}_i, \mathrm{tr}\left(\mathbf{A}_i \mathbf{D}_i \mathcal{Y}^T(\ell)\right)\right\} + \gamma \mathrm{Pen}\left(\mathbf{A}_i\right), \quad (2.11)$$

where $\bar{Y}_i = \frac{1}{T}\sum_{\ell=1}^{T} Y_i$, $\mathcal{L}\{.,.\}$ is a loss function (to be explained in Section 2.3.5), γ is a penalty coefficient, and $\mathrm{Pen}\left(\mathbf{A}_i\right)$ is a penalty term. The purpose of including $\mathrm{Pen}\left(\mathbf{A}_i\right)$ is to obtain a sparse representation of the informative neighborhood based on the similarity in the rate of change of the wind speed between the target site and other locations in the wind farm. In Section 2.3.2 we will discuss in detail the role and definition of this penalty term.

The variance of wind speed can be modeled as a linear combination of volatility, which presents the size of recent changes in wind speed (Gneiting et al., 2006). Specifically,

$$\sigma_i^2(e_t) = b_0 + b_1 \nu_i(t), \qquad \text{for} \quad i = 1, 2, \ldots, I, \quad (2.12)$$

where

$$\nu_i(t) = \left[\frac{1}{2|J_i|} \sum_{j \in J_i} \sum_{\ell=0}^{1} \left\{ (Y_j(t-\ell) - Y_j(t-\ell-1))^2 \right\} \right]^{\frac{1}{2}}, \tag{2.13}$$

$|J_i|$ is the number of elements in J_i, and b_0 and b_1 can be estimated through least square estimation using the past sample variance of $\sigma_i^2(e_t)$ and $\nu_i(t)$.

The predicted value for location i at h-step ahead will be the α-quantile of the truncated normal distribution;

$$\widehat{Y}(t+h) = \widehat{\mu}_i(t+h) + \alpha \Phi^{-1} \left[\alpha + (1-\alpha)\Phi(-\frac{\widehat{\mu}_i(t+h)}{\widehat{\sigma}_i(t+h)}) \right], \tag{2.14}$$

where $\widehat{\mu}_i(\cdot)$ is the estimated mean found through equation (2.10) (likewise, $\widehat{\sigma}_i(\cdot)$ can be found through equation (2.12)), and the value of α should be decided based on the loss function $\mathcal{L}(\cdot,\cdot)$. In Section 2.4.2 we discuss the optimal choice for α.

2.3.2 Selecting Informative Neighborhoods

One important aspect while making use of multiple-site measurements is which sites, among all possible locations, can be genuinely "informative". Despite the fact that a Gaussian kernel as in equation (2.8) has already been used to weigh locations based on their relative distance from the target site, our analysis reveals that a pure distance-based determination of informative sites is insufficient and ineffective. The key to identify the informative neighborhood is, based on our study, to observe that if two locations have similar rates of change in wind speed for a given period, they can be informative for each other. In other words, the spatial dependency can be determined using the correlation among the rate of change of the wind speed.

Specifically, let $Z_i(t) = \frac{dY_i^s(t)}{dt} \approx Y_i^s(t) - Y_i^s(t-1)$, for $i = 1, 2, \ldots, I$, where $Y_i^s = \frac{Y_i}{m(Y_i)}$, where $m(Y_i) = \max\{Y_i(t); t = 1, 2, \ldots, T; i = 1, 2, \ldots, I\}$. Then, define a sample covariance matrix for Z as:

$$\boldsymbol{\rho} = \frac{1}{T}\sum_{\ell=1}^{T}\left(\mathbf{Z}(\ell) - \bar{\mathbf{Z}}\right)\left(\mathbf{Z}(\ell) - \bar{\mathbf{Z}}\right)^T, \tag{2.15}$$

where $\mathbf{Z}(\ell) = [Z_1(\ell), Z_2(\ell), \ldots, Z_I(\ell)]^T$, for $\ell = 1, 2, \ldots, T$ and $\bar{\mathbf{Z}} = \frac{1}{T}\sum_{\ell=1}^{T}\mathbf{Z}(\ell)$.

Next, we discuss the role of penalty $\text{Pen}\left(\mathbf{A}_i\right)$. The inclusion of this penalty is to find a sparse representation of the informative neighborhood by utilizing the information embedded in $\boldsymbol{\rho}$. This task is done by performing several operations in turn. These operations are to ensure that we select a small neighborhood which has a high correlation in the rate of change with the target site. Specifically, $\text{Pen}\left(\mathbf{A}_i\right)$ does three operations: (a) it thresholds the entries of $\boldsymbol{\rho}$ with respect to $\beta \in [0, 1]$; (b) it creates a new matrix whose entries are the inverse of the entries of the matrix obtained in step (a) (with the convenience that inverse of zero is ∞); and (c) it returns the product between the matrix obtained after step (b) and $\mathbf{A}_i$ with the convention that $0 \times \infty = 0$. Specifically, let $\boldsymbol{\rho}^\beta$ denote the matrix $\boldsymbol{\rho}$ after thresholding with respect to β:

$$\rho^\beta_{jk} = \rho_{jk} \qquad \text{if} \quad \rho_{jk} \geq \beta \quad \text{otherwise} \quad \rho^\beta_{jk} = 0, \tag{2.16}$$

where ρ^β_{jk} and ρ_{jk} are the $(j,k)^{\text{th}}$ entries of $\boldsymbol{\rho}^\beta$ and $\boldsymbol{\rho}$ respectively for $j, k \in \{1, 2, \ldots, I\}$. Then, let $\boldsymbol{\rho}^\beta_{\text{inv}}$ define the entrywise inverse of matrix $\boldsymbol{\rho}^\beta$, that is

$$\rho^\beta_{\text{inv},jk} = \frac{1}{\rho^\beta_{jk}}, \tag{2.17}$$

where $\rho^\beta_{\text{inv},jk}$ is the $(j,k)^{\text{th}}$ entry of $\boldsymbol{\rho}^\beta_{\text{inv}}$ for $j, k \in \{1, 2, \ldots, I\}$. We assume $\rho^\beta_{jk} = 0$ implies $\rho^\beta_{\text{inv},jk} = \infty$. Finally, we define

$$\text{Pen}\left(\mathbf{A}_i\right) = \|\mathbf{A}_i^T \boldsymbol{\rho}^\beta_{\text{inv}}\|_F, \tag{2.18}$$

where $\|.\|_F$ denotes the Frobenius norm and we use the notational convention that $0 \times \infty = 0$.

In other words, $\text{Pen}\left(\mathbf{A}_i\right)$ is a scalar obtained by imposing a sparse structure on $\mathbf{A}_i$ in which entries with associated sample correlation of the derivative smaller than β is 0. If the sample correlation of the derivative is small (but larger than β) the associated entries in $\mathbf{A}_i$ are penalized more; on the other hand, if the sample correlation of the derivative is large the associated entries in $\mathbf{A}_i$ are slightly penalized. Therefore, $J_i = \{j : \rho_{ij}^{\beta} \neq 0\}$. In Section 2.4.2 we present how this approach can be used to select neighborhoods.

2.3.3 Regime Switching Gaussian Spatial-temporal Auto Regression

Following the regime switching approach developed by Gneiting et al. (2006), we extend model (2.10) to take the wind direction into account. Regimes are determined according to the wind direction, denoted by θ. This means we consider a partition of the interval $[0^o, 360^o)$, with 0^o representing due north, where each segment of the partition defines a regime. For example, an east-west two-regime partition can be represented as $\boldsymbol{r} = \{[0^o, 180^o)\}$, meaning that when $0^o \leq \theta(t) < 180^o$, it is the east regime, whereas when $180^o \leq \theta < 360^o$, it is the west regime. Then, we fit a separate model for each regime. Specifically, we can model the mean of the wind speed as

$$\mu_i(e_t) = c + \text{tr}\left(\mathbf{A}_i(\theta(t), \boldsymbol{r})\mathbf{D}_i\mathcal{Y}^T(t)\right), \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.19}$$

where $\boldsymbol{r}$ denotes the forecast regimes, and $\theta(t)$ is the current wind direction at location i. The matrix $\mathbf{A}_i(\theta(t), \boldsymbol{r})$ is similar to $\mathbf{A}_i$ as defined in Section 2.3.1, however, the dependency on $(\theta(t), \boldsymbol{r})$ means that in the training stage, for each regime, we only consider those observations that fall in the specific range determined by $(\theta(t), \boldsymbol{r})$. Then, based on the regime at time t, we make a prediction for the wind speed at time $t+h$, according to the specific trained model for that θ. We call model (2.19) Regime-switching GSTAR of order p and denote it by RGSTAR(p). Specifically, we solve

$$\min U(\lambda_{i1}, \lambda_{i2}, \lambda_{i3}) = c + \sum_{\ell=1}^{T} \mathcal{L}\left\{Y_i(\ell+h) - \bar{Y}_i, \operatorname{tr}\left(\mathbf{A}_i(\theta(t), \boldsymbol{r})\mathbf{D}_i \mathcal{Y}^T(\ell)\right)\right\} + \gamma \operatorname{Pen}\left(\mathbf{A}_i\right), \tag{2.20}$$

which, given a regime, is solved similar to optimization problem (2.11).

In our analysis, to find the regimes in each calendar month, we use the data in the previous year. First, we select a group of candidate regimes, for example, east-west, or north-south. Then, for each of the candidate regimes, we fit the model (2.19), and then we choose a regime which yields the smallest training error. In other words, we choose a set of regimes based on the following:

$$\boldsymbol{r}^* = \arg\min_{\boldsymbol{r}} \mathcal{E}_m(\boldsymbol{r}), \qquad \text{for } m_o = 1, 2, \ldots, 12, \tag{2.21}$$

where $\mathcal{E}_m(\boldsymbol{r})$ denotes the prediction error based on some loss function for month m_o. This simply means for each month we choose the regime which yields a smaller prediction error; however, based on our analysis, selecting a regime with too many partitions does not improve the prediction accuracy as it reduces the number of data points needed for training in each regime. Therefore, in general, regimes with two or three partitions work the best.

2.3.4 Using Geostrophic Wind in Regime Switching Gaussian Spatial-temporal Auto Regression

In addition to the information related to the wind speed and wind direction, one can utilize measurements of temperature and air pressure for the purpose of short term wind forecast. One way to effectively employ temperature and air pressure measurements for the predictive modeling of wind speed is through geostrophic wind (Zhu, 2013). Geostrophic wind is a type of theoretical wind obtained by assuming an exact balance between the air pressure gradient force and the Coriolis force (Focken and Lange, 2006). Geostrophic wind can be obtained after some simple calculations on temperature and air pressure measurements (Zhu, 2013, pp.77-81). The actual value of geostrophic wind is in general assumed to be in good accordance with the wind speed close to the ground.

Let $\omega_i(t)$ denote the geostrophic wind at location i at time t. We can extend model (2.19) to incorporate geostrophic wind. Specifically,

$$\mu_i(e_t) = \operatorname{tr}\left(\mathbf{A}_i(\theta(t), \boldsymbol{r})\mathbf{D}_i\mathcal{Y}^T(t)\right) + \sum_{\ell=1}^{w} \psi(t-\ell)\omega_i(t-\ell), \qquad \text{for} \quad i = 1, 2, \ldots, I \tag{2.22}$$

where $\psi(\ell)$ denotes the coefficient of the geostrophic wind at time ℓ and w is the order of the model associated with the geostrophic wind. We call model (2.22) RGSTAR Geostrophic Wind of order (p, w) and denote it by RGSTARGW(p,w). To find the optimal values of the parameters in model (2.22), we use a two-step approach. First, given p and w, we fit data to the model RGSTAR, for each respective regime, based on the model discussed in Section 2.3.3, namely optimization problem (2.20). Then we regress the residuals on the geostrophic wind using least square estimation.

$$\widehat{Y}_i(t+h) - \operatorname{tr}\left(\widehat{\mathbf{A}_i}(\theta(t), \boldsymbol{r})\widehat{\mathbf{D}_i}\mathcal{Y}^T(t)\right) = c + \sum_{\ell=1}^{w} \psi(t-\ell)w_i(t-\ell), \qquad \text{for} \quad i = 1, 2, \ldots, I, \tag{2.23}$$

where $\widehat{\mathbf{A}_i}(\theta(t), \boldsymbol{r})$ and $\widehat{\mathbf{D}_i}$ are the estimated values for the matrix $\mathbf{A}_i(\theta(t), \boldsymbol{r})$ and $\mathbf{D}_i$ respectively, and $\widehat{Y}_i(t+h)$ is the predicted value obtained in the first step.

In all the above discussion, we focus on how to incorporate the spatial information into our model for a given order of temporal dependency, namely for fixed p, q and/or

w. We delay the discussion of the selection of temporal order Section 2.4.4.

2.3.5 Choice of Loss Functions and Handling Missing Data

The evaluation of prediction should be based on suitable loss functions for the wind energy industry. If $\widehat{Y}_i(t+h)$ denotes the predicted wind speed for h-step ahead forecast at location i for $i = 1, 2, \ldots, I$, a common choice to evaluate the prediction is the mean absolute difference

$$\mathrm{MAD} = \frac{1}{I}\sum_{i=1}^{I}\left|\widehat{Y}_i(t+h) - Y_i(t+h)\right|. \tag{2.24}$$

However, as Hering and Genton (2010) argued, this loss function (or similar loss functions like root mean squared error) is not suitable for the wind energy industry. The reason can be explained by recalling that the ultimate goal in short term wind forecast is to predict the amount of wind energy produced. To estimate the wind energy produced by a turbine, one can use the power curve associated with a turbine converting wind speed into power production.

As shown in Figure 2.1, for a wind speed smaller than cut-in speed the energy produced by the turbine is zero. For wind speed between cut-in speed and rated speed, the energy produced is monotonically increasing. If the wind speed is between the rated speed and the cut-out speed, the output will not exceed the maximum capacity of the turbine. If the wind speed exceeds the cut-out speed the turbine is shut down for safety purposes and no energy is produced. To explicitly consider the energy produced, Hering and Genton (2010) proposed the power curve error which is defined as

$$\mathrm{PCE}(Y, \hat{Y}) = \begin{cases} \alpha\left(g(Y) - g(\hat{Y})\right) & \text{if } \hat{Y} \leq Y, \\ (1-\alpha)\left(g(\hat{Y}) - g(Y)\right) & \text{if } \hat{Y} > Y, \end{cases} \tag{2.25}$$

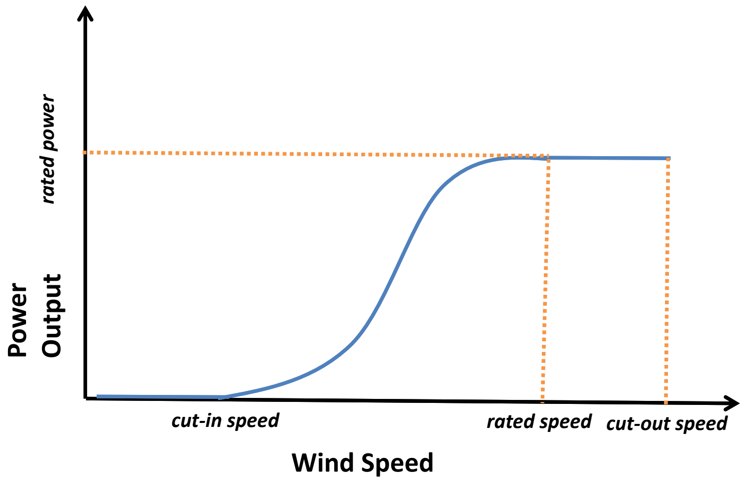

Figure 2.1: Power curve for a wind turbine

where $g(.)$ is the power curve and $\alpha \in (0,1)$. The parameter of α is introduced to penalize underestimation and overestimation differently. The reason can be attributed to the fact that, in practice, the cost incurred by underestimating the wind energy is more than that of overestimating. Therefore, for practical purposes $\alpha > 0.5$. Furthermore, the value of α in equation (2.25) is the quantile of the optimal predictor, given the PCE is used as the loss function, and is the same as the α shown in equation (2.14).

In solving optimization problem (2.11), the loss function $\mathcal{L}$ is determined based on the forecast objective. For example, if we want to make a prediction based on the power curve error, optimization problem (2.11) can be written as

$$\min U(\lambda_{i1}, \lambda_{i2}, \lambda_{i3}) = \sum_{\ell=1}^{T} \text{PCE}\left\{Y_i(\ell+h) - \bar{Y}_i, \mathbf{A}_i \mathbf{D}_i \mathcal{Y}^T(\ell)\right\} + \gamma \text{Pen}\left(\mathbf{A}_i\right), \quad (2.26)$$

where we simply replace $\mathcal{L}(.,.)$ with PCE. Note that PCE possesses practical significance as it explicitly consider the cost associated with wind speed forecast for power generation. Nevertheless, we also consider the continuous rank probability score (CRPS) which is akin to PCE, but it evaluates the forecast from a distributional perspective. Specifically, CRPS is defined as

$$\text{CRPS} = \frac{1}{I}\sum_{i=1}^{I}\int_0^1 \left(\hat{F}(X) - \mathbb{I}(X \geq g_i)\right)^2, \tag{2.27}$$

where $\hat{F}(X)$ is the distributional forecast of power (assuming g is known), g_i is the (normalized) power generated by turbine i, and $\mathbb{I}(\xi)$ is equal to 1 if ξ is true otherwise it is 0. and Unlike Gneiting et al. (2006) we do not use CRPS for estimation, but for model evaluation (He et al., 2013).

For the data sets we use in this paper, there exist missing observations for both turbines and meteorological masts. To address the missing data problem, we impute missing values using a singular value decomposition (SVD) method (Golub and Van Loan, 2012). Let $\mathbf{X}$ denote the complete data matrix, where data refer to any of the following: wind speed, wind direction, temperature, or air pressure. Assume that the dimension of $\mathbf{X}$ is $n \times m$, and let $\mathbf{I}$ be a matrix of the same size where if (i, j) element of $\mathbf{X}$ is observed then (i, j) element of $\mathbf{I}$ is 1, otherwise it is 0. A truncated SVD decomposition of $\mathbf{X}$ is defined as

$$[\mathbf{U}_d, \mathbf{\Sigma}_d, \mathbf{V}_d] = \text{SVD}(\mathbf{X}, d), \tag{2.28}$$

such that

$$\mathbf{X} = \mathbf{U}_d\mathbf{\Sigma}_d\mathbf{V}_d', \tag{2.29}$$

where $\mathbf{U}_d$ is an $m \times d$ matrix, $\mathbf{\Sigma}_d$ is a diagonal $d \times d$ matrix and $\mathbf{V}_d$ is an $n \times d$ matrix, where $d < \min(m, n)$. Then, for imputing the missing values, we use iterative SVD decomposition (Maadooliat et al., 2013; Beckers and Rixen, 2003). This procedure leaves the observed data intact and only interpolates the missing values. For the wind speed data, we have an average of 2 to 3 percent of missing data in each month which is relatively small. For other meteorological data, in particular temperature and pressure we observe relatively higher percentage of missing data, which for some months reaches up to 30 percent. The adverse effect of this high amount of missing data will be discussed in Section 2.4.2.

2.4 Results

The wind farm we are interested in, consists of more than 200 wind turbines and a few meteorological mast towers. The data used in this paper are randomly selected at 200 sites. We used 200 turbines rather than all of them because our industrial partner deems the exact number of turbines confidential and asks that number not to be disclosed.

This wind farm is on a reasonably flat terrain, so the elevation differences between the highest and lowest locations are less than ten meters, over a stretch of approximately 160 square kilometers. The data we have were obtained in 2008 through 2010. Each wind turbine has the measurements of wind speed for every 10 minutes and the standard deviation of the wind speed during that 10-minute period. The meteorological mast towers provide us with the measurements of temperature, air pressure and the wind direction, all measured as a 10-minute average. Arranging data in 10-minute blocks is a common practice in the wind industry. For our forecast purpose, we further average the each data set over each hour, i.e. an average over the 10-min observations taken at 10, 20, 30, 40, 50, and 60 minutes; therefore we obtain datasets with a measurement resolution of one hour.

We consider three proposed methods: GSTAR, RGSTAR, and RGSTARGW. GSTAR(p) is simply Gaussian Spatial-temporal Model as described in Section 2.3.1, RGSTAR(p) is the regime-switching GSTAR in which the regime in each calendar month is decided based on the data in the previous year. Finally, RGSTARGW(p,w) utilizes the temperature and pressure measurements in forms of geostrophic wind as described in Section 2.3.4. For each model, we define four epochs for each day in a calendar months: (1) 12:00 am to 6:00 am, (2) 6:00 am to 12:00 pm, (3) 12:00 pm to 6:00 pm, and (4) 6:00 pm to 12:00 am. This means, for example, the time sections 12:00 am to 6:00 am in each day for the month of January comprise an epoch during which the wind speed is assumed a stationary stochastic process. Consequently, we need to fit individual models for each epoch, depending on the epoch to which the forecast horizon belongs. In each month, we randomly assign each turbine (out of a total of 200 turbines) to one of the four regimes, and fit a respective model for that case. Then, we report the average prediction error for each month which is a good approximation of the overall prediction error as epochs are uniformly assigned to turbines for model evaluation. Table 2.1 summarizes features of each of the models.

Table 2.1: Features of proposed models

Model	Epochs	Regimes	Geostrophic Wind
GSTAR	Yes	No	No
RGSTAR	Yes	Yes	No
RGSTARGW	Yes	Yes	Yes

The competing algorithms will be ARMA(p,q), ARMA*(p,q), Vector AR of order p (VAR(p)), and Persistence method (PER). ARMA*(p,q) is the same as ARMA(p,q) but the analysis is performed on the residuals after removing a diurnal trend as will be discussed in Section 2.4.4. For the VAR(p) we select the neighborhood based on the geographical distance smaller than 5km. For these models we select $p = 1$, $q = 2$, and $w = 1$. We delay the discussion on the rational and criteria for selection of p, q, and w, and also, addressing the diurnal trend in wind speed to Section 2.4.4. We compute h-step ahead predictions for $h = 2, 3, \ldots, 5$. Specifically, for each h-step ahead prediction we train the model for 30 days using hourly data and then make predictions for the next h hours, i.e. we fit separate models for each prediction horizon; based on our experiments, this creates better predictions compared to multi-step prediction in which we use fit a single model and then make a prediction step-by-step for the next h hours. The loss functions will be PCE as defined in Section 2.3.5.

2.4.1 Model Setup

An important aspect regarding the GSTAR models is the way they select the informative neighborhood for each target site. It will be elucidative to take a look at how neighborhoods are selected. Figure 2.2 shows a sample of results for three different sites using data of January of 2009. As evident in the figure, the selection is not necessarily based on geographical proximity. The neighborhood is selected according to historical similarity in the rate of the change of the wind speed as explained in Section 2.3.2.

Next we discuss the actual values used for β in equation (2.16). Smaller values for β imply a larger neighborhood for a target site since it makes the thresholded covariance matrix of the derivatives, $\boldsymbol{\rho}^{\beta}$, have less zero entries. On the other hand, larger values for β create a smaller informative neighborhood. As will be discussed in Section 2.4.3, the suitable size of the neighborhood is related to the forecast horizon. Nevertheless, for the range of 2-hour ahead to 5-hour ahead the weather-related

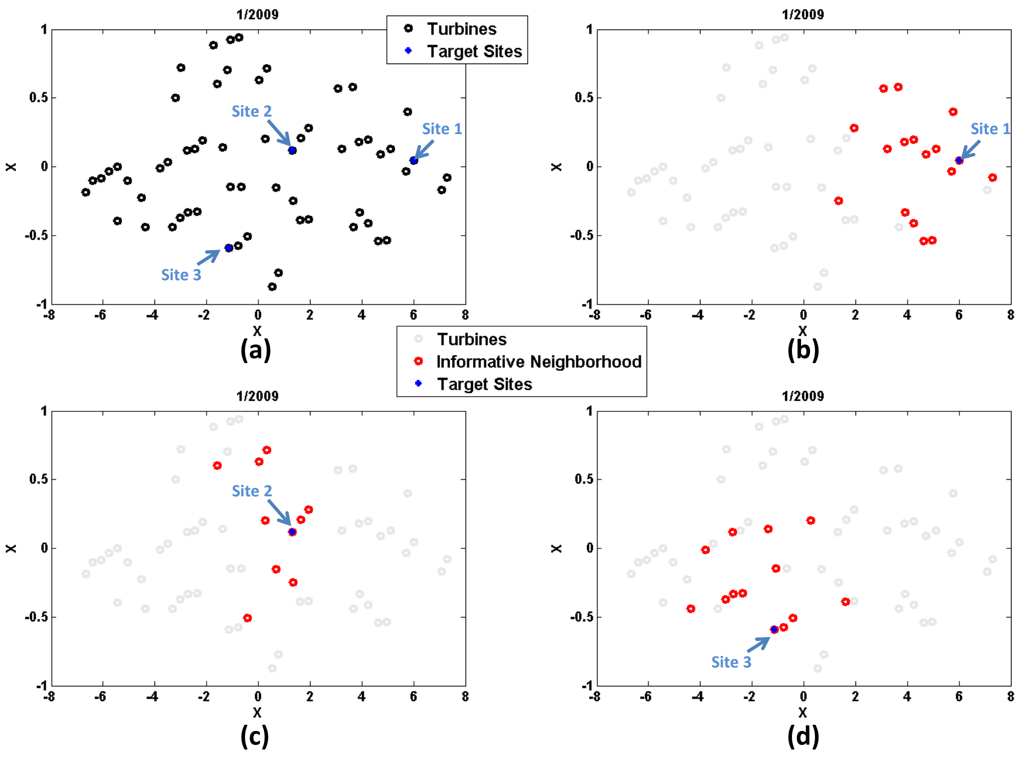

Figure 2.2: Neighborhood selection in GSTAR models for a sample data in January 2009: (a) three different sites and turbines in their neighborhood; (b)-(d) informative neighborhood selection for each site.

information, which travels through the wind speed, contributes little to the model's performance. Therefore, we choose similar values for each horizon but in a descending order, i.e. for 2-hour ahead forecast we choose $\beta = 0.85$ and for 5-hour ahead $\beta = 0.92$ and we linearly interpolate the values of βs for 3 and 4-hour ahead. The rational for this descending trend is that for longer horizons, a smaller neighborhood is deemed sufficient since the information conveyed through the weather system will be to little avail. This phenomenon will be further discussed in Section 2.4.3.

2.4.2 Forecasts and Comparisons

Here we present the prediction results for the array of methods selected, including different variants of the GSTAR models, VAR model, ARMA models, and PER. We train each model using one month of data and then make an h-step ahead forecast for $h = 2, 3, \ldots, 5$ hours. For each of year 2009 and year 2010, we do this task for turbine $i = 1, 2, \ldots, 200$, therefore for each method we get 200×12 prediction results per year, and we report the average values and the standard deviations of the associated loss functions. The data in year 2008 is used to determine the regimes for year 2009. For the loss function PCE, we normalize the power curve so the rated power is transformed into 1, also we choose $\alpha = 0.73$ as suggested by Hering and Genton (2010). The point forecast for GSTAR models is α-quantile of the predictive normal distribution, since Gneiting (2011) notes that for the PCE error, the α-quantile is the optimal predictor.

Tables 2.2 shows the results for the GSTAR methods (i.e. GSTAR, RGSTAR, and RGSTARGW) and four competing methods, which are PER, VAR(1), ARMA(1,2), and ARMA*(1,2). The results suggest that in general GSTAR models perform better than the persistence method, the vector autoregressive and the autoregressive moving average. Another piece of information which is worth considering is the relative performance of ARMA and VAR. ARMA performs better than VAR: this indicates the significance of choosing the appropriate neighborhood as discussed in Section 2.3. A poor selection of neighborhood, such as simply based on the geographical distance, can mislead the model into making worse predictions for longer horizons. We want to note that the GSTAR methods consistently outperform the VAR and ARMA methods, sometimes by more than 10% reduction in PCE. The relative poor performance of RGSTARGW 2009 can be attributed to the high percentage of missing meteorological data in a few months, consequently affecting the performance of the model.

Table 2.2: Prediction results for 2009 and 2010 using PCE. The values in parenthesis are the standard deviations of the corresponding predictions. Last row denotes improvement of the best method over PER.

2009				
Method	2-h	3-h	4-h	5-h
PER	0.054 (0.013)	0.066 (0.17)	0.076 (0.022)	0.089 (0.025)
VAR(1)	0.109 (0.053)	0.115 (0.044)	0.126 (0.042)	0.127 (0.037)
ARMA(1,2)	0.058 (0.016)	0.077 (0.020)	0.088 (0.028)	0.100 (0.028)
ARMA*(1,2)	0.066 (0.018)	0.085 (0.022)	0.094 (0.027)	0.104 (0.028)
GSTAR(1)	**0.050 (0.012)**	**0.058 (0.016)**	**0.066 (0.019)**	**0.080 (0.025)**
RGSTAR(1)	0.055 (0.018)	0.061 (0.020)	0.070 (0.022)	0.085 (0.025)
RGSTARGW(1,1)	0.054 (0.016)	0.059 (0.019)	0.069 (0.021)	0.087 (0.027)
Imp. % over PER	7.4	12.1	13.2	10.1
2010				
PER	0.047 (0.012)	0.053 (0.015)	0.069 (0.019)	0.074 (0.023)
VAR(1)	0.110 (0.068)	0.129 (0.075)	0.138 (0.058)	0.148 (0.057)
ARMA(1,2)	0.055 (0.014)	0.068 (0.017)	0.085 (0.023)	0.094 (0.026)
ARMA*(1,2)	0.096 (0.015)	0.104 (0.017)	0.118 (0.020)	0.121 (0.022)
GSTAR(1)	**0.042 (0.013)**	0.052 (0.016)	**0.063 (0.019)**	**0.071 (0.020)**
RGSTAR(1)	0.043 (0.017)	**0.051 (0.017)**	0.067 (0.022)	0.073 (0.021)
RGSTARGW(1,1)	**0.042 (0.014)**	0.054 (0.016)	0.066 (0.021)	**0.071 (0.021)**
Imp. % over PER	10.6	3.8	8.7	4.1

To achieve a better understanding of the relative performance of the GSTAR models, we present more detailed results. Table 2.3 shows the average improvement in prediction made on individual turbines. Specifically, the first number in parentheses is the percentage of turbines for which the proposed methods works better than the persistence method and the second number is the average percentage of improvement over the persistence method for those turbines. We observe that for those best methods identified in Tables 2.2, the GSTAR methods always perform better than the persistent method on more than half of the turbines. In fact, for many of the case, the GSTAR methods perform better on nearly two-thirds of the turbines. A second observation is that whenever the GSTAR methods perform better, it makes an appreciable improvement: the improvements are always more than 15%, but could be as much as 29%. We do acknowledge that for some other cases, due to the sudden change in the wind speed or other forms of volatility, the GSTAR models cannot capture such trends, and could perform worse than the persistent model. Hence, such cases suggest that more sophisticated framework is also needed to address those patterns and trends the GSTAR methods currently fall short of capturing. We also

Table 2.3: Percentage of improvement over PER in 2009 and 2010, measured by PCE on individual turbines. The data pair in a parenthesis are the percentage of turbines that see an improvement and the average improvement percentage for those turbines. The bold values correspond to the best method in Table 2.2.

Method	2-h	3-h	4-h	5-h
GSTAR(1) (2009)	(66, 18.0)	(**75**,**20.1**)	(**77**,**20.3**)	(**70**,**19.8**)
RGSTAR(1) (2009)	(53, 23.9)	(60,27.6)	(58,30.1)	(57,27.3)
RGSTARGW(1,1) (2009)	(**49**,**24.9**)	(60,28.9)	(60,29.1)	(49,30.8)
GSTAR(1) (2010)	(**63**,**22.1**)	(54,18.6)	(**63**,**19.5**)	(**59**,**15.9**)
RGSTAR(1) (2010)	(61,31.9)	(**56**,**27.1**)	(54,28.7)	(53,26.5)
RGSTARGW(1,1) (2010)	(**66**,**29.2**)	(46,26.3)	(54,28.1)	(**53**,**26.8**)

present a sample of results using CRPS as the loss function. We randomly select twenty turbines and apply the GSTAR model using the data in the year 2009. The results, shown in Table 2.4, compare the value of CRPS for GSATR with taht of ARMA(1,2). As the table suggests similar messages, we will not discuss the CRPS loss function. Finally, we want to investigate if the model performance is sensitive to

Table 2.4: Prediction results for 2010 using CRPS for twenty randomly selected turbines.

Method	2-h	3-h	4-h	5-h
ARMA(1,2)	1.166	1.480	1.845	2.353
GSTAR	0.997	1.198	1.478	1.532

the value of α used in the PCE. Note that the PCE loss function and the associated value of $\alpha = 0.73$ are suggested by Hering and Genton (2010) as a "practical" measure for the accuracy of the prediction. Considering values of α ranging from 0 to 1 would be of little use as it considers arbitrary measures for over and underestimation. However, we present sensitivity analysis for the value of the parameter α. Our sensitivity analysis is to change the value of α between 0.6 and 0.8 and we average the PCE error over this range. We present the results of this sensitivity analysis for a sample of 100 turbines and the data in the year 2009. The analysis suggests despite the fact that $\alpha = 0.73$ is used for model fitting and evaluation, the performance of the model is not sensitive to that specific value.

Table 2.5: Sensitivity analysis for the value of α ranging in $[0.6, 0.8]$ for the year 2009. The values in parenthesis are the average of standard deviations for each case.

Method	2-h	3-h	4-h	5-h
PER	0.071 (0.007)	0.073 (0.008)	0.064 (0.009)	0.077 (0.010)
GSTAR	0.057 (0.007)	0.066 (0.007)	0.053 (0.009)	0.074 (0.008)

2.4.3 Informative Neighborhoods and Propagation of Information

Here we discuss the question of why the GSTAR models can outperform the competing models. This question can be approached from an information propagation perspective. Recall that the region we are considering is relatively small, around 160 square kilometers. Therefore, the information related to the weather system, which is assumed to travel by the wind speed, travels the entire region in a short amount of time generally between half an hour to an hour. Consequently, for a forecast horizon greater than 2 hours, which is the focus of this paper, such information cannot have a role in the prediction performance. This shows the contrast between this problem and other spatial modeling for wind speed/power forecast such as those of Tastu et al. (2014) or Gneiting et al. (2006) in which the weather information at a relatively distant point, in some cases more than 100 kilometers away, is used for the prediction at the target site. In fact, the best relative performance for the model proposed by Tastu et al. (2014) occurs at 1-hour ahead forecast, and that can be explained by noting the average wind speed in the region (30-50 km/h) and the average distance of their target site and other wind farms in their study(around 50 km).

The performance of the GSTAR models relies on different aspects, which are neighborhood selection, regime switching, the loss function, and epochs. In fact, the GSTAR models select a neighborhood which has a similar rate of change with that of the target site, and as a result and combined with the other aspects of the methods, they improve the prediction power. Note that this does not imply the prediction error for longer horizons are better than that of the shorter horizons: as tables in Section 2.4.2 demonstrate, in general, the prediction error increases as the prediction horizon gets longer. However, the deterioration for the GSTAR models, compared to other methods, is less significant.

In other words, the GSTAR models find a subset of turbines, which have a similar rate of change with that of the target turbine, and then build, loosely speaking, an "ensemble learner" using those turbines. Hence, the GSTAR models do not use the data in the sense that Tastu et al. (2014) use, but more closely to Alexiadis et al. (1999) and Kusiak and Li (2010), in which a machine learning algorithm using the data in close-by region has been built to predict the wind speed.

To test this hypothesis, i.e. the performance of the GSTAR models lies, partially, in utilizing an effective machine learning algorithm and not that the weather information is employed, we design an experiment. We want to see if there exists a neighborhood which is informative in the sense described in the paper. To this end, we compare the change in the neighborhood selected by the GSTAR for two cases, (i) when the GSTAR performs better than the persistence method, and (ii) when the GSTAR performs worse than the persistence method. Our hypothesis suggests that for the former, the selected (informative) neighborhood should remain pretty much the same as we move to the following month, while for the latter the informative neighborhood should change dramatically. To this end, we define a metric change in the neighborhood $ch(i,m)$ for $i = 1, 2, \dots, 240$ and for $m = 1, 2, \dots, 12$

$$ch(i,m) = \frac{|J_i(m) \cap J_i(m+1)|}{|J_i(m)|}, \tag{2.30}$$

where $J_i(m)$ denotes the informative neighborhood selected by the GSTAR at month m for turbine i and $|J_i(m)|$ is the number of elements in the set $J_i(m)$. Note that $ch(i,m)$ is a number greater than or equal to 0 and smaller than or equal to 1. To obtain a robust statistic, we set a threshold, $t^{ch} \in (0,1)$, and find the proportion of time the change in the informative neighborhood is larger than that threshold. In other words, we define

$$S_c = \mathbb{E}\left\{\mathbb{I}\left(ch(\cdot,\cdot) \geq t^{ch}\right)\right\}, \tag{2.31}$$

where $\mathbb{I}$ is the indicator function defined similarly to what we had in equation (2.27) and $\mathbb{E}$ is the expectation operator. Simply speaking, an estimate for the value defined by (2.31) denotes the probability of a "significant" change in the informative neighborhood. Therefore, our proposed hypothesis means that if GSTAR outperforms the persistence method, S_c should be relatively larger compared with the cases that GSTAR performs worse. To test this hypothesis, we select 60 turbines and calculate S_c metric for the year 2010, for the two cases where the GSTAR produces a better h-hour ahead forecast compared with the cases when the persistence method produces a better h-hour ahead forecast, for $h = 2, 3, 4, 5$. We choose $t^{ch} = 0.8$ and estimate the value of S_c by counting the proportion of time $ch(i, m)$ is greater than 0.8. The results are shown in Table 2.6. As the results display, for smaller horizons, when the GSTAR model performs better, with a higher probability we observe a significant change in the neighborhood selected. However, for longer horizons, this rule does not apply.This suggests that when predicting for shorter horizons, the role of an informative neighborhood is more important. However, that functionality becomes less significant for longer horizons.

Table 2.6: Proportion of significant changes in the informative neighborhood S_c.

S_c	2-h	3-h	4-h	5-h
GSTAR performs better	0.68	0.59	0.53	0.53
Persistence performs better	0.56	0.56	0.60	0.72
Difference	0.12	0.03	-0.07	-0.19

This brings us to the role of some other aspects in the GSTAR models, specifically using epochs for modeling non-stationarity and the use of the PCE loss function in model training for GSTARs. Particularly, the latter, using the PCE, helps the GSTAR models adapt to the data based on the loss function used in the evaluation. In particular, the GSTAR models are less sensitive for the wind speed less than the cut-in speed or greater than the cut-out speed. That contributes to an overall better performance once we evaluate the models mainly in the range that falls between the cut-in and cut-out speed.

2.4.4 Temporal Dependency for Wind Speed

Here we revisit some issues regarding specification of temporal parameters and diurnal trends in spatial-temporal models and discuss them in more detail. One important issue regarding autoregressive moving average models is deciding the orders of the model, p and q. For autoregressive models, one approach to find the order p is the partial autocorrelation function (Brockwell and Davis, 2009). Partial autocorrelation of lag k is defined as the autocorrelation between the terms with indices t and the $t+k$ where the linear dependency of the terms with indices $t+1$ to $t+k-1$ are not accounted for, i.e. removed. Katz and Skaggs (1981) found that in general, for meteorological data, a low order for AR models, often $p = 1$, would suffice; for our data, the results seem to be in accordance with this claim.

Specifically, Figure 2.3 displays a sample of results for partial autocorrelation for turbine#1 in year 2009. As evident in the figure the order of 1 for autoregressive models deems adequate. The results for other cases agree with this figure. Therefore, for autoregressive term in VAR, GSTAR, RGSTAR, and RGSTARGW models, we use $p = 1$. Also for RGSTARGW, as the partial autocorrelation of order 1 for the geostrophic wind has the largest value and higher orders are negligible, we use $w = 1$. One can also use criteria such as the Bayesian Information Criterion to identify the

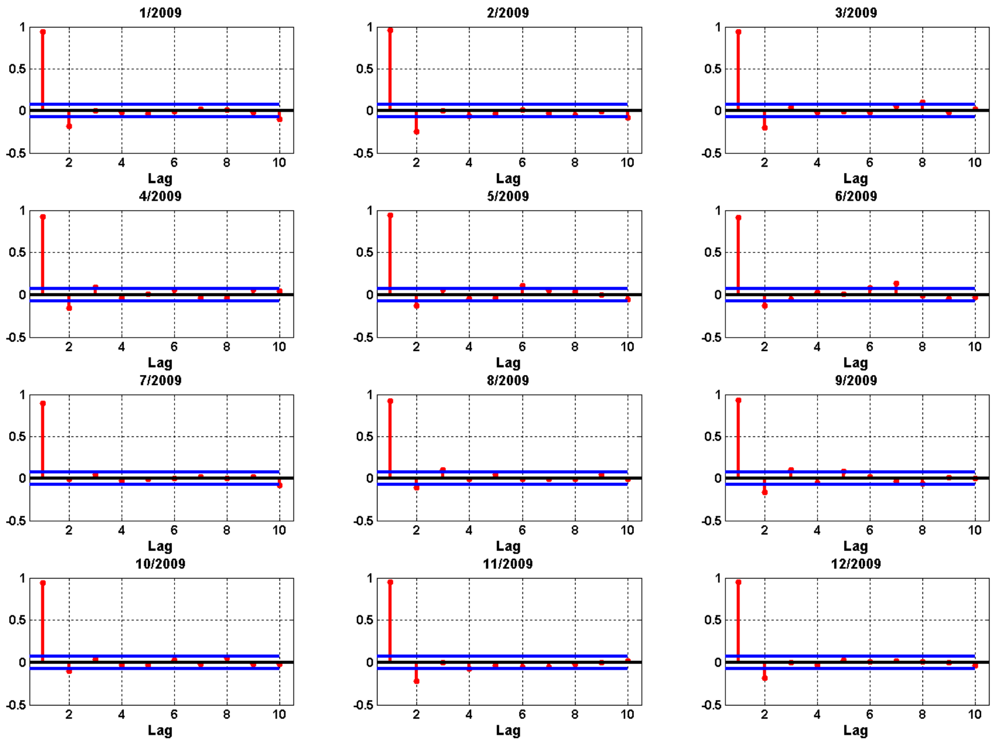

Figure 2.3: Sample partial autocorrelation for turbine#1 in 2009: each subplot denotes the sample partial autocorrelation for a month in 2009; the x-axis is the lag (in hour) and the y-axis is the sample partial autocorrelation. The two parallel lines are approximate 95% upper and lower confidence bounds.

order for the autoregressive part p or moving average parts q. In general, this criterion suggests that low order ARMA models are adequate, such as $p = 1$ and $q \leq 2$ for most cases (Erdem and Shi, 2011; Torres et al., 2005). For our data, ARMA(1,2) provides the smallest BIC for most cases and therefore we choose it as a competing algorithm.

Another important issue for wind speed modeling is dealing with diurnal trends. The existence of diurnal trends in the wind speed is a well-known fact. However, addressing this diurnal trend is not a straightforward task. For example, Gneiting et al. (2006) proposed to use summation of trigonometric functions to model the diurnal trend:

$$D(s) = d_0 + d_1 \sin\left(\frac{2\pi s}{24}\right) + d_1 \cos\left(\frac{2\pi s}{24}\right) + d_1 \sin\left(\frac{4\pi s}{24}\right) + d_1 \cos\left(\frac{4\pi s}{24}\right), \quad (2.32)$$

where $s = 1, 2, \ldots, 24$. Based on the regime-switching approach proposed by Gneiting et al. (2006), however, removing a diurnal trend does not always provide a better forecast results in all regimes, even for the same data set. In Gneiting et al. (2006), for the westerly regime the authors removed the diurnal trends and performed the subsequent analysis on the residuals, whereas for the easterly regime, they use the original time series as removing the trends did not improve predictive performance. Using equation (2.32) to model the diurnal trend for our data set does not appear to improve the prediction in general.

Other approaches for removing the diurnal trends have been proposed in the literature. For example, Torres et al. (2005) standardize the data according to

$$Y_i(t)^* = \frac{Y_i(t) - \mu^*(s)}{\sigma^*(s)}, \qquad \text{for } s = 1, 2, \ldots, 24;\ i = 1, 2, \ldots I, \quad (2.33)$$

where $Y_i(t)^*$ denotes the standardized speed (i.e. speed after removing the diurnal trend) and

$$\mu^*(s) = \frac{\sum_{j=0}^{d-1} Y_i(24j + s)}{d}, \qquad \text{for } s = 1, 2, \ldots, 24;\ i = 1, 2, \ldots I, \quad (2.34)$$

$$\sigma^*(s) = \left[\frac{\sum_{j=0}^{d-1} \left(Y_i(24j + s) - \mu^*(s)\right)^2}{d}\right]^{\frac{1}{2}}, \qquad \text{for } s = 1, 2, \ldots, 24;\ i = 1, 2, \ldots I, \quad (2.35)$$

where d is the number of days in each month. Due to large day-to-day variability in our datasets, using this formulation for our wind speed data leads to an almost constant $\mu^*(s)$ and a very large $\sigma^*(s)$ which will make the standardized data nearly zero. As such, what we would have done using this formula is to simply remove a constant c as the diurnal trend, which is in fact what we actually did in Section 2.3.1.

Not removing a diurnal trend seems to be at odds with standard time-series practices. This appears one unique aspect in handling wind speed data that have a very significant day-to-day variability. Consequently, it becomes challenging, and currently still an elusive objective, regarding how to fit a suitable temporal trend and extrapolate the trend into the prediction horizon. Empirical evidence suggests, at least in our analysis, that for many cases regarding wind speed forecast, one would be better off to conduct the analysis on the original data.

The relative poor performance of ARMA*(1,2), as demonstrated in Section 2.4.2, indicates the issue associated with removing a diurnal trend and then adding it back. To better illustrate this point we present in Figure 2.4 the wind speed data for turbine#1 for seven consecutive days. If one looks at an individual day, there appears a diurnal trend. On the other hand, if one looks at the seven days together, there is a large day-to-day variability that renders the practice of deseasonalizing the trend different and current methods of doing so ineffective.

We suspect that this complicated trend in wind speed data can be ascribed to nonlinearity embedded in some meteorological structures (Giannakis and Majda, 2012), and we believe that on such data sets, any linear attempt to capture the temporal effect would be to little avail. Together with our comparison results in Section 2.4.2, we present the take-away messages from this study as:

- The spatial information incorporated in the proposed GSTAR models helps the wind forecast objective;

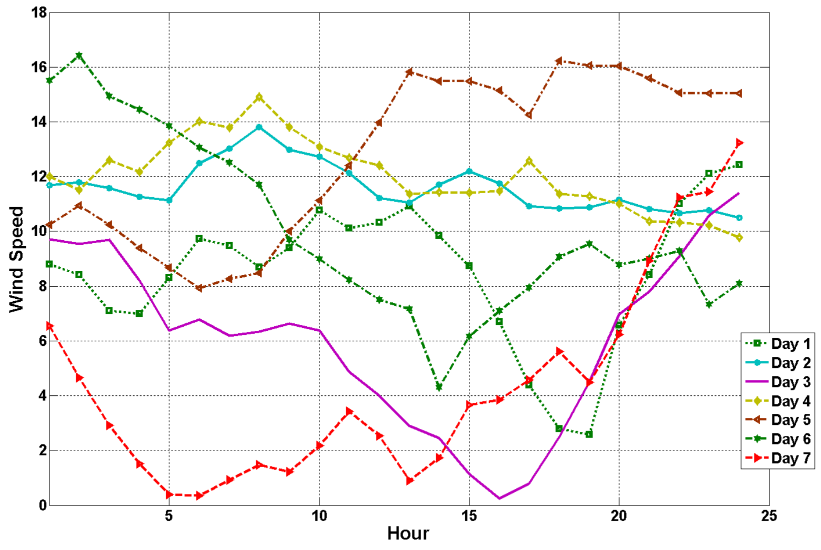

Figure 2.4: Wind speed for turbine#1 for a sample week

- Selection of the informative neighborhood is important for making effective use of the spatial information;
- We suggest using the rate of change in wind speeds as the criterion to select the informative neighborhood, instead of basing the selection on distance only.
- The temporal information and trend is difficult to model, so that many time series models in and by themselves cannot beat the persistent method.

The results reported in Table 2.3 indicate that our proposed methods do not always outperform the persistent model on the 200 turbines. This is in contrast to the large body of literature using a single time series dataset, claiming to make improvement over the persistent model. Our study suggests that it is unlikely that any temporal-only model can consistently outperform the persistent model in a comprehensive comparative study.

2.5 Summary

In this section we presented a spatial-temporal model for short-term forecasting of wind speeds in local wind fields. Our contributions included a novel approach to determine the informative neighborhood based on the rate of change in wind speeds. To produce a robust predictive outcome, especially for a longer prediction horizon, we identified three important components: (a) using the wind speed measurements in the informative neighborhood, as just mentioned; (b) using the regime-switching approach to account for the wind direction effect; and (c) incorporating other environmental measurements, such as temperature and air density. Empirical comparisons using 200 turbines and real two-year data demonstrated the practical utility of the model. From the results, we concluded that our spatial-temporal model outperformed the persistent model for three-hour to six-hour forecast horizons in most cases.

3. MODULUS PREDICTION OF BUCKYPAPER BY MULTI-FIDELITY ANALYSIS INVOLVING LATENT VARIABLES*

3.1 Introduction

As mentioned in Section 1, GP regression may serve as a surrogate model for a computer experiment. To make the surrogate model match the real process, we need to calibrate that model. This section presents the details of such a procedure for a nano-manufacturing problem (Pourhabib et al., 2014).

Carbon nanotubes (CNTs) are a type of carbon structure made of nano-scale tubes (Iijima, 1991). Possessing exceptional thermal and mechanical properties, CNTs hold promise for a wide range of applications (Tsai et al., 2011). One method to fabricate CNT-based products is by manufacturing buckypaper, which are thin layers of CNTs (Wang et al., 2004). However, since buckypaper does not necessarily possess the desirable properties for industrial applications (Tsai et al., 2011), one treatment is to add polyvinyl alcohol (PVA) (Zhang et al., 2011), which results in the high stiffness product, PVA-treated buckypaper.

It is important to understand how the addition of PVA in the presence of other noise variables affects the stiffness of the buckypaper as measured in terms of the Young's modulus. Thus, practitioners use a standard approach to conduct a set of physical experiments, namely to fabricate a number of buckypaper strips with varying amounts of the PVA added, measure the Young's modulus of the resulting buckypaper, and fit a functional relationship between the PVA input and the stiffness output. Because these physical experiments are both time-intensive and costly,

*Reprinted with permission from "Modulus prediction of buckypaper based on multi-fidelity analysis involving latent variables" by Pourhabib, A., J. Z. Huang, K. Wang, C. Zhang, B. Wang, and Y. Ding, 2014. *IIE Transactions*, in press, Copyright 2014 by Taylor & Francis.

and measuring the Young's modulus can damage the buckypaper being tested, a simulation model based on finite element approximation has been developed to numerically calculate the Young's modulus of the buckypaper under a given amount of PVA additive and a few specifications of CNTs (Wang et al., 2013). Unfortunately, the co-existence of the physical and simulation outputs of the buckypaper fabrication process gives rise to another issue, the multi-fidelity analysis problem.

As discussed in Section 1, the GP regression may serve as a surrogate model for a computer experiment. In this section, we discuss an application of the GP models to the buckypaper nano-manufacturing problem. To make the surrogate model (i.e. the GP model) fit the real process, we need to calibrate that model. In our case, we consider that the physical outputs provide the ground truth, and are therefore the high-fidelity outputs, whereas the simulation is an approximation, and understandably, provides the low-fidelity responses.

Multi-fidelity analysis uses the following datasets from: (a) A physical experiment and a simulation model, (i.e., our buckypaper fabrication) (also see Qian and Wu (2008); Kennedy and O'Hagan (2001); Reese et al. (2004); Kennedy and O'Hagan (2000); Joseph and Melkote (2009); Higdon et al. (2004); Han et al. (2009); Bayarri et al. (2007)); (b) Two physical processes of different measurement resolutions (Xia et al., 2011); or (c) Simulation models of different degrees of accuracy (Qian et al., 2006; Xiong et al., 2013).

Regardless of the origin of the data, we have the situation where one experiment provides more accurate data (high-fidelity) at a relatively higher cost, whereas the other provides less accurate data (low-fidelity) at a lower cost. Of course, if we could collect an adequate number of data points from the high-fidelity experiments, we would not need the low-fidelity data. In practice, the higher cost prohibits us from running the high-fidelity experiments/simulations that we need for covering a sufficient number of input conditions, nor can we rely on the high fidelity responses with their inferiority in numbers.

The methods developed to tackle the multi-fidelity fall into two broad categories: those based on building respective models for each dataset; and those based on building a model for one of datasets (e.g., a low-fidelity dataset) and then employing a linkage model to connect both datasets. The methods in the first category is based on the concept that the same underlying physical mechanism generates the data from each of the corresponding experiments and therefore creates similar models for describing the datasets that connect implicitly via the underlying physics. The literature reports a variety of modeling strategies. For example, to combine spatial data with different levels of accuracy, Wikle and Berliner (2005) devise a hierarchical Bayesian framework for making an inference at some predetermined level. Reese et al. (2004) use the low-fidelity data as a prior for making the model fit using the high-fidelity data.

The methods in the second category assume that the responses in one of the datasets can be re-constructed by including correction terms to the responses in the other dataset, and then using a calibration model to explicitly link the two datasets. Generally, the existing methods employ GP to model the low-fidelity experiment and a linear calibration function to connect the two datasets (Kennedy and O'Hagan, 2000, 2001; Qian and Wu, 2008; Xia et al., 2011; Joseph and Melkote, 2009; Bayarri et al., 2007; Goldstein and Rougier, 2009, 2006; Han et al., 2009; Higdon et al., 2004; Xiong et al., 2009). Irrespective of the specific details, they all implicitly assume that the output in each dataset is a function of the same set of input variables for both the high-fidelity and low-fidelity experiments. More important, we can directly measure these inputs so that a response from the high-fidelity experiment can be matched to its low-fidelity counterpart.

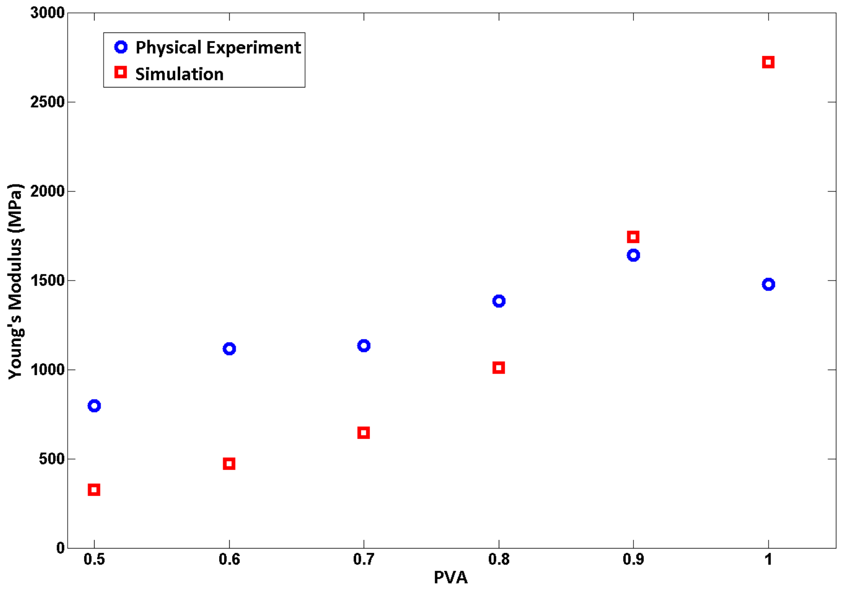

Figure 3.1: Young's modulus of simulation model and physical experiments.

As mentioned, the problem of predicting the Young's modulus of the PVA-treated buckypaper presents some extra challenges. To illustrate, Figure 3.1 above shows that our simulation model tends to underestimate the Youngs modulus for the small amounts of PVA and to overestimate the modulus for the larger amounts of PVA. Our understanding of the physical process suggests that such a mismatch in the trend line is caused by assuming in the simulation that the effectiveness of PVA, i.e., the amount of the PVA absorbed in the process, does not change as its amount varies. In other words, our assumption makes the simulation responses continue to rise at a rapid rate, as the amount of PVA addition increases, whereas the physical responses increase slowly, or even level off somewhat.

Theoretically, it should be possible for the simulation outputs to track the physical responses by using this extra variable with appropriate input values, but since we cannot directly measure this PVA effectiveness in the physical process, it becomes difficult to set in our simulation model. In other words, we have a multi-fidelity analysis problem having some unobservable input variables in the physical experiment and we have to represent them by using latent variables in the corresponding response model. As a result, our problem becomes a multi-fidelity analysis involving latent variables.

We also want to emphasize the difference between latent variables and parameters in the physical experiment. Several multi-fidelity methodologies explicitly consider the existence of some unobserved or uncontrollable features in the physical experiments, generally referred to as calibration parameters (Higdon et al., 2004; Han et al., 2009; Bayarri et al., 2007; Xiong et al., 2009; Goldstein and Rougier, 2006, 2009); however, a calibration parameter is internal to the physical experiments, rather than correlating with the inputs to another response. The role of the latent inputs here allows us to link the two experiments, which is an important difference between our proposed model and the existing multi-fidelity analyses.

In this section, we introduce a solution approach targeting the specific application as described above. We assume that the latent input variables are correlated with and can be imputed from the observable variables. Our strategy entails the following elements: (a) For the low-fidelity simulation responses, we resort to a GP model; this is the same as in the existing multi-fidelity analyses; (b) Based on the aforementioned assumption, we introduce a functional relationship connecting the latent variables with the observed ones; (c) We formulate the combined models as a non-linear optimization problem which we solve in turn by using numerical techniques.

The remainder of this section is organized as follows. Section 3.2 defines the problem and describes our mathematical setup. Section 3.3 discusses our choices of specific model components for the PVA-treated buckypaper fabrication process. Section 3.4 demonstrates how our method outperforms existing methods in terms of prediction accuracy. In particular, the advantage of the proposed method becomes more obvious when we perform extrapolation. Section 3.5 summarizes the research findings in this section.

3.2 Latent Variable Multi-fidelity Analysis with Correlated Inputs

We first introduce the notations and symbolism to define the latent variable multi-fidelity analysis problem in the context of the Young's modulus prediction in the PVA-treated buckypaper fabrication process. Consider two data sets available in such a process, the physical experiment denoted by $\mathcal{P}$ and the simulation denoted by $\mathcal{S}$. We assume there exists a degree of similarity between the simulation responses and the physical responses so their integration is justified. The degree of similarity can be easily checked by computing the correlation between the two datasets.

Let $\mathbf{x} \in \mathcal{X}$ be the input vector, then $\mathcal{P} = \{(\mathbf{x}, P(\mathbf{x})) : \mathbf{x} \in \mathcal{X}\}$ where $(\mathbf{x}, P(\mathbf{x}))$ is an input-output pair for the physical experiment. Similarly, we have $\mathcal{S} = \{(\mathbf{x}, S(\mathbf{x})) : \mathbf{x} \in \mathcal{X}\}$ where $(\mathbf{x}, S(\mathbf{x}))$ is an input-output pair for the simulation. Assume we can decompose the components of the vector $\mathbf{x}$ into two parts such that $\mathbf{x} = (\mathbf{x}^o, \mathbf{x}^m)$, where the subscript o and m stand for "observable" and "missing," respectively. Then, we can express the functional relation between inputs and outputs in the two experiments as $P = P(\mathbf{x}^o, \mathbf{x}^m)$ and $S = S(\mathbf{x}^o, \mathbf{x}^m)$. In other words, both physical experiment and simulation are functions of $\mathbf{x} = (\mathbf{x}^o, \mathbf{x}^m)$. In the physical experiment, however, only a subset of components of the input, i.e., $\mathbf{x}^o$, can be specified, while in the simulation, both $\mathbf{x}^o$ and $\mathbf{x}^m$ can be specified.

In order to handle the latent variables, we believe that their values need to be in some way informed by those of the observable inputs in $\mathbf{x}^o$, because if $\mathbf{x}^m$s are completely uncorrelated to anything we can observe, it becomes impossible to make an inference about them. Based on this understanding, we assume that $\mathbf{x}^m$s can be described by using the observations in $\mathbf{x}^o$, through a relationship $g(\cdot)$ and subject to a prescribed level of discrepancy. Specifically, we intend to find the relationship $g(\cdot)$ by minimizing the difference between the simulation outputs and the physical experiment outputs, such as:

$$\begin{aligned} &\min_{g\in\mathcal{G}} \mathcal{L}\left(P\left(\mathbf{x}^o,\mathbf{x}^m\right), S\left(\mathbf{x}^o,\mathbf{x}^m\right)\right) \\ &\text{s.t.} \qquad \int_{\mathcal{X}} \left[\mathbf{x}^m - g(\mathbf{x}^o)\right]^2 \mu(d\mathbf{x}) \leq \delta, \quad g \in \mathcal{G}, \end{aligned} \tag{3.1}$$

where $\mathcal{L}(.,.)$ is a loss function, $\mathcal{G}$ is a class of functions to which g is deemed to belong and δ is the predetermined discrepancy allowance in terms of some metric induced by a measure $\mu(\cdot)$. The integral constraint is to connect the unobservable variables $\mathbf{x}^m$ with the observed variables $\mathbf{x}^o$, by minimizing the average difference between the latent variables and the fitted values based on the estimated relationship.

This formulation is in general difficult to solve. To make it tractable, we would like to introduce a few simplifications. Since we care about the mean prediction, the loss function $\mathcal{L}(.,.)$ is chosen to be a squared error loss function. An alternative choice is the absolute error loss, and using the absolute error loss leads to optimality in median estimation. The absolute error loss is more robust to the existence of outliers, while the squared error loss is easier to optimize. In our application, the outlier problem is not a source of worry so we choose the squared error loss here.

Being multi-fidelity means that the simulation responses generally differ from the physical responses by a noticeable bias. Without bias, people can simply run the low-fidelity simulation a good number of times and average the responses to produce a result comparable to the high-fidelity source. In reality, the low-fidelity data sources are inherently inferior because the bias cannot be reduced or eliminated through averaging. When using the squared error loss function, we would like to include a bias term $B(\mathbf{x}^o, \mathbf{x}^m)$, the value of which may depend on the input conditions in general. Under this general circumstance, we assume that $B(\mathbf{x}^o, \mathbf{x}^m)$ can be parameterized by a set of parameters Θ_B. One example of such parametrization is to use a Gaussian process to model the bias $B(\mathbf{x}^o, \mathbf{x}^m)$ as a function of the input conditions; as such, Θ_B contains the parameters in the GP model.

The loss function will be evaluated using a set of training data. Suppose that we execute n runs of high-fidelity experiments, having their input conditions as $\mathbf{x}_1^o, \mathbf{x}_2^o, \ldots, \mathbf{x}_n^o$, and the ith experiment was replicated n_i times. Then, the noise contaminated responses of the high fidelity experiments are

$$y_{ij} = P(\mathbf{x}_i^o) + \epsilon_{ij}, \quad i = 1, 2, ..., n, \text{ and } j = 1, ..., n_i. \tag{3.2}$$

where $\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ captures variability in y due to both measurement errors and uncertainty associated with unknown latent variable $\mathbf{x}_i^m$.

In parallel, we also execute a set of low-fidelity simulations. Here we are primarily concerned with the so-called *deterministic simulations* that yield the same response, when run repeatedly under the same input condition. The deterministic simulations are usually referred to as computer experiments (Santner et al., 2003). The simulation, being low cost computationally, can be run in a large number. Suppose there are a total of $N(>> n)$ runs for the observable variables and L runs for the unobserved variables (recall that both of the variables can be specified in the computer experiment), then

$$S_{i\ell} = S(\mathbf{x}_i^o, \mathbf{x}_\ell^m), \quad i = 1, 2, ..., N, \text{ and } \ell = 1, ..., L. \tag{3.3}$$

Understandably, when planning for the two sets of experiments, we would like the input conditions used in the physical experiment to be a subset of those used in the computer experiment.

The simulation code has to be run at specific values of the input variables, so that including the simulation directly in an optimization formulation creates a continuous-discrete mixed optimization problem that is usually harder to solve. To alleviate this problem, We use a Gaussian process to model the simulation responses $\{S_{i\ell}\}$ and denote the resulting GP model as $\widehat{S}(\mathbf{x}^o, \mathbf{x}^m)$. The GP model provides a smooth and continuous response over the design space, and using the GP model in the objective function makes the problem easier. We want to note that modeling the low-fidelity response using GP models is a standard practice in the existing multi-fidelity analysis literature, for example, Kennedy and O'Hagan (2000, 2001); Qian and Wu (2008); Xia et al. (2011), among others, but the motivation of doing so here is slightly different.

We believe that the choice of $\mathcal{G}$ will have to be decided according to specific applications. Generally the governing physics of a process should indicate whether $\mathbf{x}^o$ and $\mathbf{x}^m$ are related, and if so, how. Here we assume that the class of $\mathcal{G}$ can be parameterized through a set of parameters in $\Theta_{\mathcal{G}}$.

Provided all the above simplifications and specifications, and moreover, choosing a counting measure for μ, the original optimization formulation can be rewritten, for a given δ, as

$$\min_{\Theta_B, \Theta_{\mathcal{G}}} \quad \sum_i \sum_j \left(y_{ij} - \widehat{S}(\mathbf{x}_i^o, \mathbf{x}_i^m) - B(\mathbf{x}_i^o, \mathbf{x}_i^m; \Theta_B) \right)^2 \tag{3.4}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \left| \mathbf{x}_i^m - g(\mathbf{x}_i^o; \Theta_{\mathcal{G}}) \right|^2 \leq \delta, \tag{3.5}$$

where the parameters of the bias and the linkage function g are explicitly mentioned to demonstrate how the decision variables impact the optimization problem. However, for simplicity of notation, hereafter we drop the explicit notational dependencies, namely using $B(\mathbf{x}_i^o, \mathbf{x}_i^m)$ for $B(\mathbf{x}_i^o, \mathbf{x}_i^m; \Theta_B)$ and $g(\mathbf{x}_i^o)$ for $g(\mathbf{x}_i^o; \Theta_{\mathcal{G}})$.

Solving the optimization problem (3.4)-(3.5) requires imposing additional constraints on the relation between observed and latent variables. The reason can be mainly attributed to the fact that we cannot observe $\mathbf{x}_i^m$ and we need to impute those values in the optimization procedure. Therefore, depending on the nature of the application, one needs to make pertinent assumptions to solve (3.4)-(3.5). For example, if the particular application permits and g is selected to be flexible enough, one may assume $\delta = 0$ which in essence implies $\mathbf{x}_i^m$ can be imputed by $g(\mathbf{x}_i^o)$. In Section 3.3 we proceed by considering a similar approach and demonstrate how one can utilize such dependency towards devising a tractable optimization problem for buckypaper fabrication. In Section 3.3.3, we choose the appropriate g function, while in Section 3.3.4 we present additional regulations to be used for the buckypaper fabrication process, and finally solve the above optimization problem.

3.3 PVA-treated Buckypaper Fabrication Process Model

In this part, we specify the model components for the PVA-treated buckypaper fabrication process. In this application, $\mathbf{x}^o$ represents the PVA amount, denoted as p and measured by the weight ratio of the PVA additive versus the raw carbon nanotubes (see also the x-axis of Figure 3.1), and $\mathbf{x}^m$ is the absorption rate of the PVA, i.e., the effectiveness of the PVA, denoted as α and expressed in percentage, so that $0 \leq \alpha \leq 1$.

3.3.1 Design of Experiments

Since $\mathbf{x}^o$ is one dimensional, the design of physical experiments is straightforward. Our material scientist partners set the PVA amount range to be from 0.4 to 1.2, and conduct a total of $n = 17$ physical experiments with p_is evenly spread over the input range. Under each p_i level, there were five replications, namely $n_i = 5$ for all $i = 1, ..., n$. There are therefore a total of 85 physical experiments conducted. We want to note that in this study, the number of physical experiments is relatively large because we need extra data for the validation purpose. In practice, it is usually difficult to afford this level of luxury.

The simulation code takes two inputs p and α. The simulation code does involve a group of randomly generated parameters associated with CNTs, such as a CNT's diameter, length, and orientation, so its response is not entirely deterministic. But the simulation code generates a large quantity of the CNTs to mimic the underlying structure in a buckypaper, and the resulting Young's modulus is affected mostly by the two inputs mentioned above. The randomness of the response, under a given setting of p, is much smaller as compared to the randomness in the physical experiments. So we believe that the simulation can be reasonably approximated by a deterministic computer experiment.

The computer experiment is designed to cover the PVA amount in the range of $0.5 \leq p \leq 1$. The physical responses outside this range are reserved for validating the quality of extrapolation. The simulation code we use has a restriction on the product of $p \times \alpha$. This product indicates the effective PVA level and cannot be smaller than 0.40 in the simulation code (Wang et al., 2013); otherwise the simulation returns a Young's modulus virtually zero. This is one of the limitations of the current simulation code for the computer experiments that the material scientists are working to improve. Given this restriction, our design input space for the computer experiment is no longer a rectangular region.

This type of design problems is generally solved through a space-filling design formulation (Johnson et al., 1990). The basic idea is to find the design points that minimize the maximum inter-point distance; this is the so-called minimax design criterion. Alternatively, a maximin criterion can be used as well (Stinstra et al., 2003). Suppose we choose the minimax criterion. The design problem can be expressed as follows—for a fixed number of design points n_s, find a set of design points $D \subset T$ that solves the following optimization formulation:

$$\begin{aligned} &\inf\nolimits_D \sup\nolimits_{t \in T} \rho(t, D) \\ s.t. \quad &|D| = n_s, \quad D \subset T, \end{aligned} \tag{3.6}$$

where $\rho(t, D) = \inf_{d \in D} \rho(t, d)$ is the inter-point distance, $|D|$ denotes the cardinality of the set D, and T is the feasible region from which a candidate design point is chosen. Specifying T differentiates the non-regular designs of arbitrary shape from the regular designs of a rectangular design region. When T is a bounded polytope, Draguljć et al. (2012) developed an efficient algorithm that finds the optimal design. The feasibility constraint for a polytope T is specified as:

$$\begin{aligned} &\mathbf{A}\mathbf{t} \leq \mathbf{r}, \\ &\mathbf{l} \leq \mathbf{t} \leq \mathbf{u}, \end{aligned} \tag{3.7}$$

for some matrix $\mathbf{A}$ and vectors $\mathbf{r}$, $\mathbf{l}$ and $\mathbf{u}$, where the inequality should hold pointwise between the corresponding vectors. Using this set of constraints, together with the minimax design criterion, Draguljć et al. (2012) showed that it can be solved using a sequential algorithm entailing mainly linear operations. For other alternatives regarding space-filling designs, for example the sliced Latin hypercube designs (SLHD), the readers may refer to Qian (2012) and Qian and Wu (2009).

The design area of our computer experiments can be duly represented by a polytope. Specifically, let $\mathbf{t} = (p, \alpha)^T$, then the design space can be represented in terms of (3.7) using the following values

$$\mathbf{A} = [-0.8, -1], \quad \mathbf{r} = -1.2,$$
$$\mathbf{l} = [0.5, 0.4]^T, \quad \mathbf{u} = [1, 1]^T.$$

Note that as only one of the constraints is non-parallel to an axis, the matrix $\mathbf{A}$ degenerates to a 1×2 vector and $\mathbf{r}$ to a real number. The number of points n_s (i.e. $|D| = n_s$) is decided such that the subsequent surrogate GP model for the low-fidelity data suitably represents the corresponding response surface. Using a few rounds of trial and error, we settle at $n_s = 150$. Note that the number of low-fidelity input settings is about one order of magnitude higher than that of the high-fidelity physical experiment (150 versus 17). Figure 3.2 displays the selected design points for (p, α) in this computer experiment.

3.3.2 Gaussian Process Model and Bias Term

Once the experimental designs are finished and data are collected, we are ready to train a GP model for the low-fidelity responses, and if needed, for the bias correction term.

The key aspect in training a GP model is to specify a covariance function which, loosely speaking, determines the similarity of the response surface at different locations. Here we choose a squared exponential (SE) covariance function as defined in

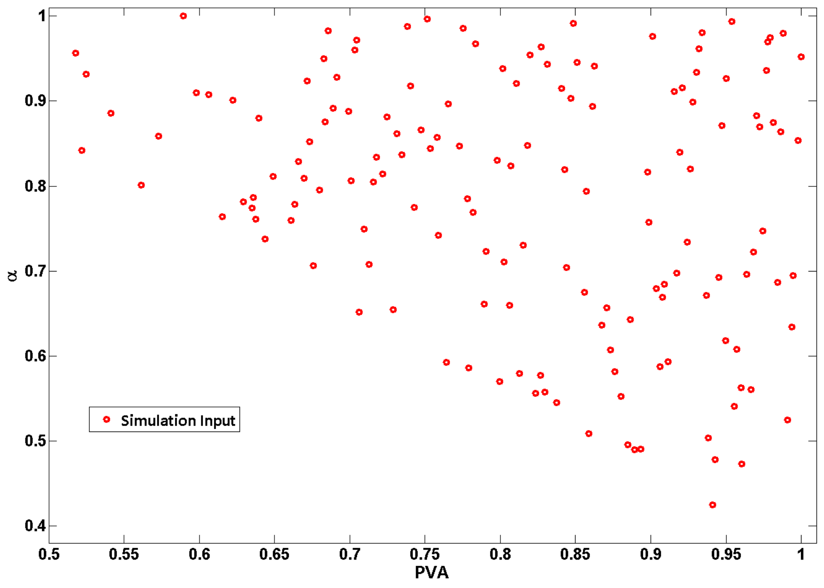

Figure 3.2: The design layout for the computer experiment.

(1.2) whose parameters are estimated by using the low-fidelity data obtained from the previous subsection. The SE covariance function is arguably the most widely used form in many applications and it is isotropic.

Concerning the choice of the bias term, we believe it is adequate to use a constant in this buckypaper fabrication process because the resulting response surface is not complicated. Making the bias term more flexible does not bring in much added value. Given this choice, the parameter $\Theta_B = \{B\}$.

3.3.3 Choice of Linkage Function

Based on our understanding of the physical process, the absorption rate of the PVA appears to be in a monotonically decreasing relation with the PVA amount (Zhang et al., 2011). This implies that when $\mathbf{x}^o = 0.7$, the corresponding absorption rate = 75%, then when $\mathbf{x}^o = 0.8$, the corresponding absorption rate is smaller than

75%. This intuitively explains why the physical responses do not increase with a rapid rate as in the simulation responses in which the absorption rate is set constant for all PVA levels. For this reason we choose $\mathcal{G}$ as the class of smooth monotonically decreasing functions.

For the PVA-treated buckypaper fabrication process, we propose to model function $g(\cdot)$ as a sum of monotone splines. Specifically,

$$g = \sum_{q=1}^{Q} g_q, \tag{3.8}$$

$\log(-Dg_q)$ is differentiable and

$$D\{\log(-Dg_q)\} = \frac{D^2 g_q}{Dg_q} \tag{3.9}$$

is Lebesgue square integrable, where D^m represents taking the derivative of order $m > 0$. These conditions guarantee that g_i is smooth and strictly monotonically decreasing (Ramsay, 1998). For different choices of q, $g(\cdot)$ can take a variety of forms which results in a rich and flexible set of functions.

In the buckypaper fabrication process, since the observable and unobservable variables both have a single element, the function form of $g(\cdot)$ can be greatly simplified. In a one-dimensional space, one solution to differential equation (3.9) is $g_q(p) = a^{b \cdot p}$, provided that $a \cdot b < 0$, thereby, assuming $Q = 1$, $g(p) = a^{b \cdot p}$ is an option for the linkage function. This simple form is desirable as it facilitates the subsequent optimization problem for linking the two experiments without sacrificing the flexibility of the model. We conduct numerical analysis later in Section 3.4, comparing this simple choice of g function with a few other alternatives and showing that this choice suits our problem well. Given this choice, the function $g(\cdot)$ can be parametrized by

$\Theta_{\mathcal{G}} = \{a, b\}$.

3.3.4 Solution Approach

The final step is to solve optimization (3.4)-(3.5). Based on our understanding of the buckypaper fabrication process, we believe it is reasonable to assume that the relation between the latent variables can be expressed as a nonlinear function of the observed variables plus a residual difference, such that

$$\mathbf{x}^m = g(\mathbf{x}^o) + e, \tag{3.10}$$

where $e \sim \mathcal{N}(0, \tilde{\sigma}^2)$. Then, in order to solve optimization (3.4)-(3.5), we can simply replace the unknown latent variables $\mathbf{x}_i^m$ with its sample mean, $g(\mathbf{x}_i^o)$, for $i = 1, 2, \ldots, n$, and plug the sample mean in the optimization formulation. When taking this approach, we can express the optimization problem as

$$\min_{\Theta_B, \Theta_{\mathcal{G}}} \sum_i \Big(\bar{y}_i - \widehat{S}(\mathbf{x}_i^o, g(\mathbf{x}_i^o)) - B(\mathbf{x}_i^o, g(\mathbf{x}_i^o)) \Big)^2, \tag{3.11}$$

where $\bar{y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij}$. The resulting optimization problem can be solved by standard nonlinear optimization techniques.

Once solving the above optimization problem in (3.11), the multi-fidelity analysis yields a linkage function $g(\cdot)$, determined by $\hat{a}$ and $\hat{b}$ (estimated parameters in $\Theta_{\mathcal{G}}$), and a bias function $B(\mathbf{x}_i^o, \mathbf{x}_i^m)$, determined by $\widehat{B}$ (estimated parameter in Θ_B). For any given test case which has an observable $\mathbf{x}_*^o$, the linkage function $g(\cdot)$ would determine an associated unobservable input component $\mathbf{x}_*^m$ which represents the average value of unobserved latent variables for the input $\mathbf{x}_*^o$. With both $\mathbf{x}_*^o$ and $\mathbf{x}_*^m$, the corresponding low-fidelity simulation response (or its GP surrogate model response) as well as the bias correction can be computed. Adding the low fidelity simulation response (or its GP model response) and the bias correction together produces the multi-fidelity prediction for input $\mathbf{x}_*^o$. Specifically, given $\mathbf{x}_*^o$, we have $\mathbf{x}_*^m = g(\mathbf{x}_*^o)$ and the predicted value $y_* = \widehat{S}(\mathbf{x}_*^o, \mathbf{x}_*^m) + B(\mathbf{x}_*^o, \mathbf{x}_*^m)$. Furthermore, as the optimization yields the functional relationship $g(.)$, one can utilize that information for a better understanding of the process. Indeed understanding how the latent and observed variables connect can provide insight into the physical process. This fact could be of significant importance for engineers who design or operate the application process.

Using the notations and specific models presented in Sections 3.3.1, 3.3.2 and 3.3.3, we can further simplify the optimization problem in (3.11) as

$$\begin{aligned} &\min\nolimits_{\theta\in\Theta} u(\theta) = \textstyle\sum_{i=1}^{n} \Big(\bar{y}_i - \widehat{S}(p_i, g(p_i)) - B\Big)^2, \\ \text{s.t.} \qquad &\Theta = \{(a, b, B) \in \mathbb{R}^3 \quad | \quad a \cdot b < 0\}, \end{aligned} \tag{3.12}$$

where Θ is used as a collection of the parameters in both Θ_B and $\Theta_{\mathcal{G}}$.

We solve this constrained optimization problem in (3.12) numerically using a steepest descent algorithm. We sequentially update the parameter values by moving opposite the gradient direction for each parameter. The steps for this procedure are summarized in Algorithm 1. The parameter ω^* in the algorithm determines the length of each optimization step. Specifically, to find the value of ω^* at each step, we discretize the interval $(0, 1)$ and choose a value which provides the largest decrease in the objective function:

$$\omega^* = \underset{\omega\in(0,1)}{\arg\min}\, u(\theta_\ell^\omega), \qquad \text{for} \quad \ell = 1, 2, 3, \tag{3.13}$$

where θ_ℓ is the ℓth parameter in Θ. Here we have three θ parameters, namely

$\theta_1 = a$, $\theta_2 = b$ and $\theta_3 = B$. In the above expression, $\theta_\ell^\omega = \theta_\ell - \omega \frac{\partial u(\theta)}{\partial \theta_\ell}$ and $u(\theta)$ is defined in (3.12). The derivative of $\widehat{S}(p_i, g(p_i))$ with respect to a and b are computed numerically. Also, to ensure the relation $a \cdot b < 0$ holds, the step to update b is performed only if the resulting b has a different sign with the current value for a. As the value of objective function decreases at each stage, the algorithm continues until the change in the objective function is negligible where the algorithm determines that location as a local optimum. The parameters of the covariance function (1.2) for the GP remain unchanged as the algorithm proceeds, because those values were estimated using merely the low-fidelity data prior to the iterations of the algorithm.

Algorithm 1 Sequential Update For Optimization Problem 3.12

```
 1: Set $\theta = (1, -0.1, 500)$
 2: repeat
 3:     Calculate $\omega^*$ according to equation (3.13)
 4:     $a \leftarrow a + 2\omega^* \sum_{i=1}^n \left\{ \left( \bar{y}_i - \widehat{S}(p_i, g(p_i)) - B \right) \frac{\partial}{\partial a} \widehat{S}(p_i, g(p_i)) \right\}$
 5:     $b \leftarrow b + 2\omega^* \sum_{i=1}^n \left\{ \left( \bar{y}_i - \widehat{S}(p_i, g(p_i)) - B \right) \frac{\partial}{\partial b} \widehat{S}(p_i, g(p_i)) \right\}$
 6:     $B \leftarrow B + 2\omega^* \sum_{i=1}^n \left( \bar{y}_i - \widehat{S}(p_i, g(p_i)) - B \right)$
 7:     Re-evaluate $\widehat{S}(p_i, g(p_i))$ based on $(a, b)$
 8:     $\theta \leftarrow (a, b, B)$
 9: until Local minima are found
10: $\widehat{\theta} \leftarrow \theta$
```

In fact our multi-fidelity analysis problem can be seen as a special case of matching a 1D curve to a 2D surface in the 3-D Euclidean space. Here the Euclidean space is generated by (p, α) together with the Young's modulus, while the 2D surface is the response surface generated by the simulation model and the 1D curve is formed by the responses of the physical experiment. Once the 2D surface is constructed, one can choose to position the 1D curve such that the response values associated at

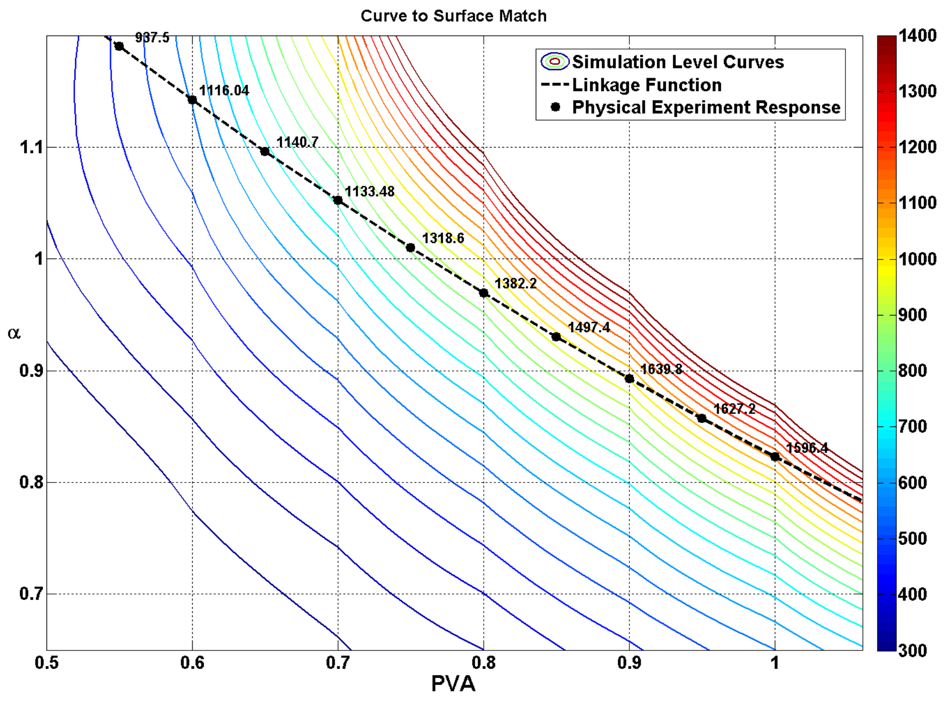

Figure 3.3: The curves are the level sets for the simulation surface with step size of 50. The colorbar represents the Young's modulus from the simulation model. The dashed curve shows the linkage function. The values close to the dark circles are the Young's modulus from the physical experiment given the corresponding PVA values. The linkage function is decided such that the overall difference between the physical experiment responses and the simulation responses, plus some constant bias, is minimized.

different locations (i.e., PVA levels) on the curve can be matched to those on the 2D surface as close as possible, after a bias adjustment. Once such a match is found, it reveals the linkage function between the two variables, as illustrated in Figure 3.3. In our solution procedure, the manipulation of the position of the 1D curve is in fact done through specifying and solving for the linkage function, as we presented in the preceding parts of this section.

3.4 Results

In this part, we evaluate the performance of the proposed multi-fidelity analysis method. In the first subsection, we compare the performance of the proposed method with two alternatives, while in the two following subsections, we investigate the impact of the high-fidelity data amount on the multi-fidelity analysis and the effect of different choices of the linkage function.

3.4.1 Performance Comparison

Concerning the multi-fidelity analysis problem involving latent variables, we see two alternatives to what we present in this part: (a) since the effectiveness of PVA is not observable, one may argue that we should simply ignore its existence and use whatever is observable to conduct a multi-fidelity analysis following the procedure, say, in Reese et al. (2004) or in Kennedy and O'Hagan (2001). (b) Because the low-fidelity computer experiment response, while using the observable variable alone (i.e., the PVA amount), could possibly mislead us, it may be most appropriate to rely solely on the physical experiment data for making predictions at an input level where experimental data were not available. To do this, a Gaussian process model can be used to fit the physical data and make predictions. We refer to option (a) as the multi-fidelity analysis without considering the latent variables ("MFA w/o LA"), option (b) as a single-fidelity analysis (SFA), and our proposed method as "MFA with LA".

More specifically, in MFA w/o LA, we assume the physical experiment value for run i (i.e. $P(p_i)$) could be modeled after a bias and scale change on the simulation response $\widehat{S}(p_i)$. Here, $\widehat{S}(p_i)$ is the average of the surrogate model $\widehat{S}(p_i, \alpha)$ over all possible values of α. The calibration model can be expressed as

$$P(p_i) = \beta_0 + \beta_1 \widehat{S}(p_i) + \gamma_i, \tag{3.14}$$

where β_0 and β_1 are constants and $\gamma_i \sim \mathcal{N}(0, \sigma_\gamma^2)$. Then the model can be readily solved following the procedure in Kennedy and O'Hagan (2001).

On the other hand, when choose option (b), i.e., SFA, we simply train a one-dimensional GP using the training data $\{(p_i, y_{ij}), i = 1, 2, \ldots, n; j = 1, 2, \ldots, n_i\}$.

To evaluate the performance of a method, we divide the physical experiment data into the training set and test set: use the training set to fit a model during the analysis step and use the test set to compute a performance measure. Note that the low-fidelity data are only used in the training (model fitting) stage not in the testing stage, because the outcome from a multi-fidelity analysis is supposed to be better than the low-fidelity response; otherwise, it is of no value to conduct the multi-fidelity analysis. One performance measure we use is the standardized root mean squared errors (SRMSEs):

$$\text{SRMSE} = \sqrt{\frac{\sum_{i=1}^{n_t} \left[(\hat{y}_i - \bar{y}_i) / \bar{y}_i \right]^2}{n_t}}, \tag{3.15}$$

where $\hat{y}_i$ denotes the predicted value (i.e., a method's output) when given the ith observable input $\mathbf{x}_i^o$ in the test set and n_t is the number of data points in the test set. Besides, as suggested by a reviewer, we also consider standardized maximum absolute deviation

$$\text{SMAD} = \max \left\{ (\hat{y}_i - \bar{y}_i) / \bar{y}_i \right\}; \quad i = 1, \ldots, n_t. \tag{3.16}$$

Depending on how the training/test data sets are generated, we produce the following three types of performance measures:

- Leave-One-Out (LOO): For the details of LOO cross validation, please refer to Hastie et al. (2001). The reported LOO SRMSE is the average of the 13 SRMSEs computed when one of the physical data points is left out during the training stage for the physical data in the range of $0.5 \leq p \leq 1.1$ (therefore $n = 12$ for each case).

- Extrapolation (EXT): Under this circumstance, the training dataset contains all the physical data in the range of $0.5 \leq p \leq 1.1$. Four pairs of data points outside this range, two having $p < 0.5$ and two having $p > 1.1$, are used as the test set (therefore $n = 13$).

- Interpolation (INT): Under this circumstance, we select eight of the physical data points, evenly spread over the input region as the training set and use the remaining as the test set (therefore $n = 8$).

Table 3.1 shows the comparison of results from the three different methods, where under "Improvement" column, the numerics are the reduction of SRMSE or SMAD, expressed in percentages, when the proposed MFA method is compared to the other two methods. As evident in the table, the proposed method outperforms the other two algorithms for all evaluation measures. When the latent variable is present so that the low-fidelity response deviates significantly from the high-fidelity response over certain area of the input space, the existing multi-fidelity analysis ("MFA w/o LA") performs even worse than the single-fidelity analysis. This outcome suggests that without a new methodology handling the latent variables, people would be better off by ignoring the low-fidelity responses.

It is interesting to know that the proposed MFA performs much better than the SFA and MFA w/o LA when they are used for extrapolation. Extrapolation is considered more valuable for product development and process control purposes because a good extrapolation tool can save the time and cost while exploring a large response surface. It is a common understanding that SFA does not have good extrapolation ability since it is purely data driven. The multi-fidelity analysis is supposed to improve SFA over the extrapolation ability, because the low-fidelity model is supposed to be still physics-based and can guide its response when doing extrapolation. Of course, this is only true when the low-fidelity model uses the right physics to guide its response. We believe this is one critical reason to understand the role of the latent variables and then incorporate them into the multi-fidelity analysis.

Table 3.1: Comparison of methods: the two rightmost columns show the improvement percentage of the proposed method over the other two methods.

	SRMSE			Improvement (%)	
	MFA with LA	SFA	MFA w/o LA	over SFA	over MFA w/o LA
LOO	0.0032	0.0045	0.0140	29%	77%
EXT	0.0392	0.0806	0.1501	51%	73%
INT	0.0383	0.0547	0.0681	30%	43%

	SMAD			Improvement (%)	
	MFA with LA	SFA	MFA w/o LA	over SFA	over MFA w/o LA
LOO	0.0092	0.0097	0.0195	5%	20%
EXT	0.0885	0.0993	0.1666	11%	47%
INT	0.0612	0.0818	0.0923	25%	34%

3.4.2 Impact of High-fidelity Data Amount

We are interested in knowing how the high-fidelity data amount may impact the quality of the multi-fidelity analysis. Here our benchmark is the SFA, since the previous subsection establishes that with the presence of latent variables the SFA outperforms the multi-fidelity analysis that does not consider the latent variables.

To this end, we select a subset of data points from the physical experiment and conduct both SFA and multi-fidelity analysis (with LA) using the same set of data. Here we keep the same number of replications per input level as before but choose a subset of the PVA amounts. We start off with four PVA levels, which are randomly selected, as the training data, and then we append one extra PVA level at a time and observe the difference between the SRMSEs when using the two methods; the SRMSEs are obtained by comparing the predicted values at the PVA levels not used in the training data with their counterparts from the physical experiment.

Figure 3.4 displays the results. If we look at the right side of the figure when the high-fidelity physical data are plenty, there is not much difference between SFA and MFA. This is expected, as we argued before that with a sufficient amount of high-fidelity data, SFA can do an adequate job of making predictions, and consequently, the low-fidelity data may no longer be needed. As we move along the horizontal axis to the left as the amount of the high-fidelity physical data gets smaller, the benefit of using MFA becomes obvious because MFA can borrow strength from the simulation responses.

As the high-fidelity data points get fewer, the difference between MFA and SFA once again diminishes. We believe that there are two reasons behind. The first reason is common to all multi-fidelity analysis problems. When the high-fidelity data points are too few, the dominance in data amount as presented by the low-fidelity data is much more pronounced and the benefit of using a combined MFA could be compromised. This reason alone, however, cannot explain the trend shown in Figure 3.4, in which when the high-fidelity data points are three or four, MFA makes slight improvement over SFA. The previous study in the literature concerning multi-fidelity analysis without latent variables showed a somewhat different insight – when the high-fidelity data points become substantially small, the benefit of using MFA, albeit compromised, remains significant; for an example, please see Table VII of Xia (2008, page 90).

That is why we believe that for the problems of MFA with latent variables, the second reason is more important. The existence of latent variables forces us to include another layer of estimation action, which is to use the multi-fidelity data to find out the linkage function between the observable and unobservable variables. The quality of this estimation action suffers when the high-fidelity data points are too few. In turn, a poorly estimated linkage function does not do due service for making the combined predictions better than that from SFA.

This analysis tells us that a multi-fidelity analysis with latent variables will be effective only for the right range of the amount of high-fidelity data points. The low bound of this range depends on the amount of data points that can provide a quality estimation of the linkage function, and the upper bound is decided by the amount of data points that can make SFA self-sufficient. Our experience indicates that there is generally a considerable gap between the two bounds for practical problems, thereby rendering the multi-fidelity analysis with latent variables a useful methodology.

3.4.3 Impact of Linkage Functions

We compare different linkage functions g which could potentially be used in the proposed method. Our aim is to investigate the effect of the functional form specified

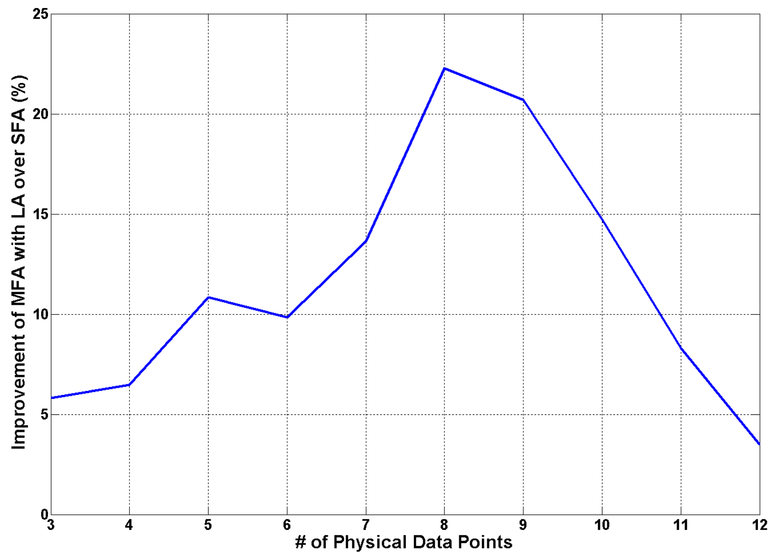

Figure 3.4: Improvement of MFA with LA over SFA under different number of high-fidelity data points.

for linking the two sets of data sources and substantiate the specific choice of the linkage function made in the previous parts of this section.

We consider two sets of alternatives. The first is a more complex class $\mathcal{G}$ whose elements are expressed as the sum of two decreasing splines. Specifically, for the form defined in (3.8), we let $Q = 2$ which means each function in $\mathcal{G}$ is the sum of two exponential functions. Comparing the choice between $Q = 1$ versus $Q = 2$ is intended to provide some insights into the question whether a more complex class of functions would improve the prediction accuracy. The second set is the polynomial functions that are popularly used in curve fitting. Specially, we consider the linear and quadratic functions. Our experience with the buckypaper fabrication process indicates that using a very complex form for the linkage function does not make the final model effective because as the number of parameters to be estimated in the subsequent optimization problem increases, the efficiency of the subsequent optimization procedure deteriorates.

Table 3.2 compares the different linkage functions in terms of SRMSE for both extrapolation and interpolation cases. As evident in the table, using the class $\mathcal{G}$ with $Q = 1$, which was the linkage function chosen in Section 3.4.1, produces the best results among all, while using a more complex function does not appear to benefit the final prediction objective. This is not only true from the $Q = 1$ versus $Q = 2$ comparison but also true from the linear versus quadratic comparison (namely a linear function works better).

We believe that the reason that the simple linkage function is favored in our problem is rooted in the fact that the problem has only one observable and one unobservable variable, and that the two variables appear to have a rather monotonic relationship. This may not be true for other problems. We stress that the linkage function should be chosen based on the structure of a specific problem and the availability of data. One can choose other classes of functions in the case of a viable justification for the problem of interest. In addition, more data points also offer the opportunity to employ a linkage function comprising more parameters for handling a linkage relationship of complicated form.

Table 3.2: Comparing different linkage functions in terms of SRMSE: the rightmost column denotes the linkage function used in Section 3.4.1

	SRMSE			SRMSE
	Linear	Quadratic	$Q = 2$	$Q = 1$
EXT	0.0490	0.0570	0.0898	0.0392
INT	0.0409	0.1753	0.0504	0.0383

3.5 Summary

In this section we developed a new method for predicting the Young's modulus in PVA-treated buckypaper. We aggregated the information from two datasets, (physical experiments and an FEA-based simulation model) by introducing a latent variable which represented the level of effectiveness of the PVA in each sample. The latent variable in turn helped to determine the functional relation between the effectiveness and the PVA level. Solving for the linkage function led to a multi-fidelity model allowing predictions to be made at any untried levels of the PVA.

Applying the model to the PVA-treated buckypaper fabrication process showed that it outperformed both the existing multi-fidelity analysis that does not consider the latent variables and the single-fidelity analysis that ignores the low-fidelity data. Notably, in the problems of multi-fidelity analysis with latent variables, the proposed method appeared effective when the amount of high fidelity data was in the correct range, i.e., too few high-fidelity data points did not allow a quality estimation of the linkage function, whereas too many high-fidelity rendered the single-fidelity analysis self-sufficient. From the results, we concluded that our proposed multi-fidelity analysis method can exploit the valuable information in the low-fidelity (simulation) data and make an overall better prediction.

4. BAYESIAN SITE SELECTION FOR FAST GAUSSIAN PROCESS REGRESSION*

4.1 Introduction

This section presents an approximation algorithm for GP regression (Pourhabib et al., 2014). Since its introduction, the GP regression has gained popularity among experts ranging from computer scientists and statisticians to engineers. GP's flexibility, nonlinearity, and inherent nonparametric structure are the key features which have made it of use to a wide range of researchers (Rasmussen and Williams, 2006). GP regression has proliferated in recent years owing to the widespread availability of data. On the other hand, however, the vast amount of data, while furnishing adequate information to train the model, could hamper computationally-efficient implementation of GP regression. As the Gaussian distribution is central to the GP regression, in almost all methods of full GP regression one needs to invert matrices of size equal to the number of data points; this could be a burdensome task as its complexity is of order $\mathcal{O}(N^3)$, noting that most methods require executing this matrix inversion many times to guarantee successful implementation of the algorithms.

For example, the surrogate GP model used for the buckypaper fabrication process in Section 3 utilizes a modest number of data points in the training stage. However, if we wanted to use a very large number of data points, to achieve a more reliable surrogate model, it would become practically impossible to utilize a full GP. Another example that discussed in Section 2 can also convey a similar message: Due to the existence of very large datasets we cannot afford to rely on a full implementation of a

*Reprinted with permission from "Bayesian site selection for fast Gaussian process regression" by Pourhabib, A., F. Liang, and Y. Ding, 2014. *IIE Transactions*, 46(5), 543-555, Copyright 2014 by Taylor & Francis.

GP model. In the case of the GP regression, spurred on by GP's popularity, research has been conducted in recent years to address the computation issue of handling large datasets. There are two main schools of thought: sparse approximation and low-rank approximation. The sparse approximation methods employ a compactly supported covariance function in a way that it results in a sparse covariance matrix, still of size N, but inverting this sparse matrix using the sparse matrix algorithms (Furrer et al., 2006; Gneiting, 2002) can lead to a substantial reduction in computation. Although the theoretical complexity of this method is difficult to determine, Furrer et al. (2006) observed through a number of numerical case studies that the training computation increases almost linearly in N. This class of algorithms, however, suffers from a high order of complexity during the test stage, which is also linear in N, whereas the low-rank approximation, as we will briefly review below, can do faster than $\mathcal{O}(N)$ during testing.

The second school of thought, the low-rank approximation, tries to reduce the computational complexity by inverting a matrix of reduced rank instead of the original covariance matrix. Utilizing different techniques to produce the reduced-rank matrix, low-rank approximation can be categorized into three groups: matrix approximation, localized regression, and likelihood approximation. The algorithms based on the matrix approximation seek substitutions for the original covariance matrix, which has truncated bases (e.g. Nyström method), resulting in a rank reduction, and can therefore be handled less expensively (Quiñonero-Candela and Rasmussen, 2005). Localized regression assumes that the data points far from each other do not share any measure of similarity and one can employ a GP on a local region by merely taking the data points in the very region into account (Park et al., 2011). The likelihood approximation methods (Seeger et al., 2003; Snelson and Ghahramani, 2007; Snelson, 2007) try to reduce the computation cost by making use of a set of unobserved latent variables called pseudo points. Assuming the conditional independence of the observed variables given the pseudo outputs, one needs to invert matrices of the size equal to that of the pseudo points M, and doing so can save significant time if one chooses $M \ll N$ (Seeger et al., 2003; Snelson and Ghahramani, 2007; Snelson, 2007). Specifically, assuming that each training and test point in the dataset is independent from others given the pseudo points, one can achieve a computational complexity of $\mathcal{O}(NM^2)$ for the training stage and $\mathcal{O}(M^2)$ for testing (Snelson and Ghahramani, 2006). This complexity expression is generally true for other methods in the school of low-rank approximation methods, although the meaning of M in a specific method differs.

While the methods in the low-rank approximation class typically mitigate the computational burdens, they suffer from other problems. For instance, the matrix approximation algorithms may lead to poor estimation and lack of interpretability (Snelson and Ghahramani, 2006), and its prediction variance is not guaranteed to be positive (Park et al., 2011). In localized regression algorithms, it is not very straightforward to select independent subsets. Localized regression predictions, in general, lack continuity on boundaries, and the existing methods to address this problem cannot handle datasets other than those from one- or two-dimensional spaces (Park et al., 2011). In likelihood approximation, despite the fact that the accuracy and computation of the algorithm rely heavily on the number of pseudo points selected, there are no strict guidelines regarding how to choose them, and currently, M is typically fixed *a priori*.

In this section, we choose to make a further improvement for the likelihood approximation methods because they are easy to use and do not have problems such as higher complexity in testing (associated with the spare approximation) or discontinuity in prediction (with the localized GP). But as we mentioned above, one major improvement needed for the likelihood approximation methods is a more flexible way of deciding the number of pseudo points. The current inflexibility often causes the likelihood approximation methods to have a higher prediction error (measured by the mean squared errors using a testing data set).

For the purpose of improving the likelihood approximation, we propose a Bayesian Site Selection (BSS) method that allows the data to decide simultaneously the number and locations of pseudo inputs. Specifically, BSS considers the pseudo inputs as a new set of parameters in the model and selects them from a subspace of the training data. Then, BSS tries to estimate the posterior predictive distribution via a Markov Chain Monte Carlo (MCMC) method. We call the new set of parameters in the BSS "sites," which are the counterparts of the pseudo inputs in the likelihood approximation methods. We generate an artificial GP defined on the sites, and condition our real response on those artificially-generated outputs to reduce the order of computational complexity.

Comparing with the current likelihood approximation methods, BSS tries to systematically discover the number of sites used as the pseudo inputs. The efforts spent in finding the appropriate sites are valuable as those lead to a less subjective algorithm and produce more accurate results. In addition, BSS chooses the locations of sites based on an MCMC algorithm, and by applying MCMC, BSS employs more than one GP approximation, and can thereby provide more accurate prediction results through the mechanism of Bayesian model averaging (Hoeting et al., 1999).

Understandably, doing all these in BSS comes with a higher computation cost than the existing likelihood approximation methods. The theoretical computational complexities of the likelihood approximation methods and BSS, if using the same number of sites, are at the same order; for training, it is at $\mathcal{O}(LNM^2)$, where L is the number of iterations a method employs to fit the model – the model fitting process is also known as *hyperparameter learning* in GP research (Rasmussen and Williams, 2006). Because the current likelihood approximation methods use a deterministic gradient-based optimization method, its L in practice is smaller than that in BSS; L in BSS is the number of iterations of MCMC. But we would like to point out that BSS, with appropriate priors chosen, can produce a practically sensible balance between computation time and prediction accuracy: it is fast enough to handle large datasets that a full GP is unable to handle, while it improves, quite often remarkably, the prediction accuracy, as compared to deterministic likelihood approximations.

The remainder of the section is organized as follows. Section 4.2 gives the GP regression formulation and uses the method in Snelson and Ghahramani (2006) to explain the basic thoughts behind the likelihood approximation. Section 4.3 presents the details of the BSS approach, including discussions about the implementation of the method. In Section 4.4, we implement our method on several simulated and real datasets and show that the BSS method outperforms the existing methods for several test cases. Finally, we conclude the research findings regarding GP approximation methods in Section 4.5 with additional discussions and comments.

4.2 Likelihood Approximation Based on Pseudo Inputs

Recall that in Section 1 we defined the GP as a continuous stochastic process such that any finite number of those variables follow a joint Gaussian distribution. Following the notation we introduced in Section 1, define $f_i := f(\mathbf{x}_i)$. Then if $\boldsymbol{f} = \{f_1, f_2, \ldots, f_N\}$ represents a finite collection of these variables, we have

$$\pi(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}), \tag{4.1}$$

where $\mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ denotes a multivariate Gaussian distribution over $\mathbf{x} \in \mathbb{R}^d$ with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{K}$ whose entries are defined by the covariance function

$K(.,.)$ as in equation (1.1). One particular case of interest for the covariance function is a generalization of the squared exponential function (1.2), also known as automatic relevance determination (ARD) which can be represented in the following form

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_f^2 \exp\left[-\frac{1}{2}\sum_{\ell=1}^{d}\left(\frac{x_{i\ell} - x_{j\ell}}{\eta_\ell}\right)^2\right], \tag{4.2}$$

where η_ℓ is the ℓ^{th} component of the vector of the length-scale parameter $\boldsymbol{\eta} = \{\eta_1, \eta_2, \ldots, \eta_d\}$, $x_{i\ell}$ and $x_{j\ell}$ are the ℓ^{th} components of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, and d is the dimension of the input space. As we can always subtract a constant from the response values before using the data, without loss of generality we can assume $\boldsymbol{\mu} = 0$.

As mentioned in Section 1, for the task of regression we need to minimize the expected loss according to equation (1.3). It is not difficult to show (see Rasmussen and Williams (2006)) if we choose the loss function as the squared loss $(y_* - y)^2$, the joint Gaussian distribution implies that the predictive distribution of the response value at $\mathbf{x}_*$ is also Gaussian with mean μ_* and variance σ_*^2, where

$$\mu_* = \mathbf{K}_{*N}\left(\mathbf{K}_N + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y},$$

$$\sigma_*^2 = K_* - \mathbf{K}_{*N}\left(\mathbf{K}_N + \sigma^2\mathbf{I}\right)^{-1}\mathbf{K}_{N*} + \sigma^2. \tag{4.3}$$

Here, we need to elaborate on the notations used for the covariance matrix. In the above, the subscript of the covariance matrix implies the data points for which the covariance matrix is formed. For example, $\mathbf{K}_N$ is the $N \times N$ covariance matrix of all training data points, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. For a covariance matrix between a single point and a set of points, such as the test point $\mathbf{x}_*$, we explicitly denote the point in the covariance notation. Therefore, $\mathbf{K}_{*N}$ denotes the $1 \times N$ covariance matrix between the test input $\mathbf{x}_*$ and the training points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Finally, K_* is used to denote $K(\mathbf{x}_*, \mathbf{x}_*)$, the prior variance associated with the test site $\mathbf{x}_*$. The same symbolism is used throughout this section.

The computational issue mentioned earlier is related to the inversion of $(\mathbf{K}_N + \sigma^2\mathbf{I})$. This inversion happens during the learning stage of the parameters at the order of $\mathcal{O}(N^3)$. Note that once the parameter learning is done, the calculation of the mean and variance of a test point $\mathbf{x}_*$ costs $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ respectively.

The likelihood approximation method we aim at improving in this section is the one using sparse pseudo input Gaussian process (SPGP) (Snelson and Ghahramani, 2006). The SPGP method works as follows. Instead of using the N observations of y directly, which are too numerous and cause the computational problem, one can consider using a much smaller set of inputs of size $M \ll N$ to approximate the full GP model. If this smaller set of inputs is a subset of the actual N data points, that method is called the Subset of Data Approximation (Quiñonero-Candela and Rasmussen, 2005, SDA). But researchers have realized that the simple SDA usually does poorly in approximating the full GP because the possible subsets are restricted to the locations where the data were observed. If that restriction is lifted, meaning that if the input locations can be strategically selected to be at places where there are not necessarily any observations, the resulting GP approximation can be much improved, and the unrestricted inputs are then called pseudo inputs.

Following the notations in Snelson (2007), let $\bar{\mathbf{X}} = (\bar{\mathbf{x}})_{m=1}^{M}$ denote the pseudo inputs and $\bar{\boldsymbol{f}} = \left(\bar{f}\right)_{m=1}^{M}$ denote the pseudo outputs. The bar notation shows the pseudo inputs (and outputs) reside in the same spaces as those of real data, but they cannot be observed. Another important point is that since pseudo outputs are not actually observed, it does not make sense to include observation noise (i.e. ϵ) in them, which is why $\bar{\boldsymbol{f}}$ is used instead of $\bar{\mathbf{y}}$. Based on the same reasoning presented for the selection of the GP prior in equation (4.1), we can assume the following prior for pseudo outputs

$$\pi(\bar{\boldsymbol{f}}) = \mathcal{N}\left(\mathbf{0}, \mathbf{K}_M\right), \tag{4.4}$$

and if assuming the outputs are independently, identically distributed (i.i.d) given the inputs, one can have

$$\begin{aligned} \pi\left(\mathbf{y} \mid \bar{\boldsymbol{f}}, \mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right) \quad &= \prod_{n=1}^{N} \pi\left(y_n \mid \bar{\boldsymbol{f}}, \mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right) \\ &= \mathcal{N}\left(\mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\boldsymbol{f}}, \operatorname{diag}\left(\mathbf{K}_N - \mathbf{Q}_N\right) + \sigma^2\mathbf{I}\right), \end{aligned} \tag{4.5}$$

where $\mathbf{K}_{NM}$ is the $N \times M$ covariance matrix between the N training points and the M pseudo inputs (Snelson, 2007, p. 38). The matrix $\mathbf{Q}_N$ is the low-rank covariance matrix whose entries are defined by the low-rank covariance function $Q(.,.)$

$$Q\left(\mathbf{x}, \mathbf{x}'\right) = \mathbf{K}_{\mathbf{x}M}\mathbf{K}_M^{-1}\mathbf{K}_{M\mathbf{x}'}. \tag{4.6}$$

Next, integrating out the pseudo outputs produces the marginal likelihood

$$\begin{aligned} \pi(\mathbf{y} \mid \mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}) \quad &= \int \pi\left(\mathbf{y} \mid \bar{\boldsymbol{f}}, \mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right) \pi(\bar{\boldsymbol{f}} \mid \bar{\mathbf{X}}, \boldsymbol{\theta}) \mathrm{d}\bar{\boldsymbol{f}} \\ &= \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_N + \operatorname{diag}\left(\mathbf{K}_N - \mathbf{Q}_N\right) + \sigma^2\mathbf{I}\right). \end{aligned} \tag{4.7}$$

The predictive distribution can be obtained by first writing the joint distribution of $\pi(y_*, \mathbf{y})$, which is multivariate normal and takes the same form as in equation (4.7). From the joint distribution of $\pi(y_*, \mathbf{y})$, the prediction distribution $\pi(y_*|\mathbf{y})$ can be attained using the conditional normal distribution formula (Rasmussen and

Williams, 2006, p. 200). The resulting $\pi(y_*|\mathbf{y})$ is as follows:

$$\pi(y_* \mid \mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}) = \mathcal{N}\left(\mu_*, \sigma_*^2\right)$$

$$\mu_* = \mathbf{Q}_{*N}\left(\mathbf{Q}_N + \operatorname{diag}\left(\mathbf{K}_N - \mathbf{Q}_N\right) + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y}$$

$$\sigma_*^2 = K_* - \mathbf{Q}_{*N}\left(\mathbf{Q}_N + \operatorname{diag}\left(\mathbf{K}_N - \mathbf{Q}_N\right) + \sigma^2\mathbf{I}\right)^{-1}\mathbf{Q}_{N*} + \sigma^2. \tag{4.8}$$

Despite entailing $N \times N$ matrix inversions in equation (9), using the matrix inversion lemma, one can show that the computation complexity is actually $\mathcal{O}(NM^2)$ (Snelson, 2007, p. 40). The reason is simply that after using the matrix inversion lemma, the $N \times N$ matrices will become diagonal whose inversion is $\mathcal{O}(N)$, and consequently, the computation is no longer dominated by inverting those matrices.

The parameters in the above model can be categorized into two groups: (a) The so-called hyperparameters that are also used by other GP models, usually denoted by $\boldsymbol{\theta}$. Here $\boldsymbol{\theta} = \{\sigma, \sigma_f, \boldsymbol{\eta}\}$. (b) The locations of pseudo inputs $\bar{\mathbf{X}}$. To estimate all the model parameters ($\boldsymbol{\theta}$, $\bar{\mathbf{X}}$) together, one can use gradient ascent methods to optimize the marginal likelihood in (4.7). The details of the optimization procedure as well as how to take the gradients can be found in Snelson (2007, pp. 126-129). For the SPGP method, the cost for computing the marginal likelihood in (4.7) once is $\mathcal{O}(NM^2)$, due to the fact that $\mathbf{Q}_N$ is of rank M (lower than N). If the optimization method takes l steps to converge, then the training cost, i.e. that for hyperparameter learning, is at $\mathcal{O}(lNM^2)$. We will refer to the SPGP method as the Deterministic Site Selection (DSS) hereafter. This name is chosen because it helps highlight the difference between the existing likelihood approximation and our proposed Bayesian method.

4.3 Bayesian Site Selection

One drawback of the DSS mechanism in SPGP is that the number of pseudo inputs, i.e. the cardinality of $\bar{\mathbf{X}}$, is fixed at M. Given the important role that the number of pseudo inputs plays in both computation and prediction accuracy, it would be desirable that the number of pseudo inputs can change in the algorithm and be decided by the data. That is indeed the objective of the research presented in this section, through a method we label as Bayesian Site Selection.

To establish a Bayesian framework for this problem, we begin by emphasizing that the goal is to make an inference about the posterior predictive distribution $\pi(y_*|\mathbf{y}, \mathbf{X}, \theta)$. As we are interested in utilizing the information inherent in the pseudo inputs, we consider the pseudo inputs as a set of new parameters through which we can represent the predictive distribution, namely

$$\pi(y_*|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) = \int \pi\left(y_*|\mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right) \pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) d\bar{\mathbf{X}}. \tag{4.9}$$

Equation (4.9) requires specifying the posterior distribution $\pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right)$ which can be expressed as

$$\pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta\right) \propto \pi\left(\bar{\mathbf{X}}\right) \pi\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right). \tag{4.10}$$

Note that $\pi\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right)$ follows a normal distribution according to (4.7). Moreover, as the new observation y_* and $\mathbf{y}$ are jointly normally distributed, $\pi\left(y_*|\mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right)$ is also normally distributed with parameters shown in (4.2). Therefore, the only term to be determined in order to fully specify the model is the prior distribution of the pseudo inputs. A suitable prior distribution should take into account the number of pseudo inputs so that we can update our belief about their number in light of the observed data. Considering $\bar{\mathbf{X}}$ as an $M \times d$ matrix, one reasonable choice for the prior could be

$$\pi\left(\bar{\mathbf{X}}\right) \propto \frac{\lambda^{|\bar{\mathbf{X}}|}}{|\bar{\mathbf{X}}|} \times I\left(k_l \leq |\bar{\mathbf{X}}| \leq k_u\right) \tag{4.11}$$

where $|\bar{\mathbf{X}}|$ denotes the number of rows in $\bar{\mathbf{X}}$ (i.e. the number of pseudo inputs), and $I(.)$ is the indicator function. The prior considers a range for the number of locations $\left(k_l \leq |\bar{\mathbf{X}}| \leq k_u\right)$ which describes the smallest and the largest number of pseudo inputs we would like to consider in our model. The new hyperparameter λ reflects our belief regarding the average number of pseudo inputs in the prior.

Having specified the terms constituting the integrand in equation (4.9), we want to evaluate the value of the integral. Unfortunately, the integral in (4.9) cannot be solved analytically, and consequently, we need to appeal to numerical methods to approximate the integral. In fact, if we can generate Markov samples $\{\bar{\mathbf{X}}_1, \bar{\mathbf{X}}_2, \ldots, \bar{\mathbf{X}}_T\}$ from $\pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta\right)$, then we can approximate the integral in (4.9) by

$$\hat{\pi}\left(y_*|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) = \sum_{t=1}^{T} \pi\left(y_*|\mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}_t, \boldsymbol{\theta}\right). \tag{4.12}$$

Then, the problem is reduced to how to draw samples from $\pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right)$.

The difficulty associated with drawing samples from $\pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right)$ is that a direct application of MCMC fails as it requires the state space of the Markov chain to be of a fixed dimension, but the dimension of $\bar{\mathbf{X}}$ may actually vary. To overcome this issue, we can use the Reversible Jump Markov Chain Monte Carlo (RJMCMC) algorithm (Green, 1995) which allows the dimension of the state space of the Markov chain to vary. The idea behind is that RJMCMC introduces three types of moves: exchange, birth, and death. Exchange means that the chain remains in the space with the same dimension, but moves into a new state. Birth and death are the moves which change the dimension of the state space. Intuitively, a birth step augments the state space by adding new states, while a death step reduces the dimension of the state space. At each iteration, the type of move, be it an exchange, birth, or death, is randomly chosen, and one accepts the new state using a Metropolis-Hastings rule. To see full details and examples, refer to Green (1995).

4.3.1 Algorithm

Specifically, to generate the sample $\{\bar{\mathbf{X}}_1, \bar{\mathbf{X}}_2, \ldots, \bar{\mathbf{X}}_T\}$, we build a Markov chain whose space is a subset of $\mathbb{R}^{d \times s}$ for varying values of s, so that at stage t, the corresponding set of pseudo inputs $\bar{\mathbf{X}}_t$ may have a number of elements different from previous stages. This allows the number of pseudo inputs to change so that we can seek simultaneously the number and location of the pseudo inputs.

To employ the RJMCMC, we need to introduce some new notations and make some extra assumptions. We restrict the space from which the pseudo inputs can be chosen by imposing it to be finite. Specifically, let $\mathbf{S}$ denote the whole space of explanatory variables, so if $\mathbf{x}$ is an element in $\mathbf{X}$ or $\bar{\mathbf{X}}$, then $\mathbf{x} \in \mathbf{S}$. Let $\tilde{\mathbf{S}} \subset \mathbf{S}$ denote a finite discretized subspace in the sense that $|\tilde{\mathbf{S}}| < \infty$. For an $\mathbf{x} := \{x_1, x_2, \ldots, x_d\} \in \tilde{\mathbf{S}}$, $x_i \in \{\tilde{x}^i_{min}, \tilde{x}^i_{min} + \xi, \tilde{x}^i_{min} + 2\xi, \ldots, \tilde{x}^i_{max}\}, \forall i \in \{1, 2, \ldots, d\}$, where $\tilde{x}^i_{min}$ and $\tilde{x}^i_{max}$ are the minimum and maximum values to consider in the i^{th} dimension of $\mathbf{S}$, respectively, and ξ is the discretization step. We choose the locations of the pseudo inputs from $\tilde{\mathbf{S}}$ instead of from the original $\mathbf{S}$ for the sake of computation easiness. As such, $\bar{\mathbf{X}} \subset \tilde{\mathbf{S}}$ and we refer to $\bar{\mathbf{X}}$ as "sites" in our approach. Note that if ξ is taken to be a small number and $|\tilde{x}^i_{min}|$ and $|\tilde{x}^i_{max}|$ are large enough, then $\tilde{\mathbf{S}}$ can approximate $\mathbf{S}$ reasonably well.

Same as in other RJMCMC-based algorithms, we also use three types of moves: Birth, Death and Exchange. Birth and Death are used to add or remove points to the current $\bar{\mathbf{X}}$, and Exchange is used to update the locations of sites while the numbers are kept unchanged. Recall that the set of parameters for this model is ($\boldsymbol{\theta}$, $\bar{\mathbf{X}}$), where $\boldsymbol{\theta} = \{\sigma, \sigma_f, \boldsymbol{\eta}\}$. In order to optimize $\boldsymbol{\theta}$, a full Bayesian approach is to incorporate both $\boldsymbol{\theta}$ and $\bar{\mathbf{X}}$ in the RJMCMC algorithm and update them both as the chain evolves. Based on our numerical analysis, this approach, although theoretically appealing, does not provide stable results numerically; at least we have not found a robust enough numerical procedure attaining that outcome. Therefore, we choose to employ the gradient ascent method to find a $\boldsymbol{\theta}$ that maximizes the marginal likelihood (4.7) for a fixed set of $\bar{\mathbf{X}}$. That is to say, after every few iterations we seek for the optimal value of $\boldsymbol{\theta}$ conditioned on the current value of $\bar{\mathbf{X}}$, and then we update $\bar{\mathbf{X}}$ using RJMCMC moves, conditioned on the newly found value of $\boldsymbol{\theta}$. We denote the number of RJMCMC iterations between two consecutive optimizations of $\boldsymbol{\theta}$ by κ. We also want to note that the algorithm used here to maximize the marginal likelihood (4.7), for a given $\bar{\mathbf{X}}$, is the same as that in the full GP.

Next, we present the specific formulations of the RJMCMC moves. In the following, $q(A \to B)$ is the proposal distribution denoting the probability of going from a set $A \subset \tilde{\mathbf{S}}$ to $B \subset \tilde{\mathbf{S}}$, and a and b are the probabilities of performing Birth and Death, respectively; both will be explained after the formulations.

Birth: Choose a point from $\tilde{\mathbf{S}} \backslash \bar{\mathbf{X}}$, say $\mathbf{x}^*$, and add it to the current site set $\bar{\mathbf{X}}$ with probability p so that the new site set is $\bar{\mathbf{X}} \cup \{\mathbf{x}^*\}$, where

$$p = \min\left(1, \frac{\pi\left(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\} | \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) q\left(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\} \to \bar{\mathbf{X}}\right)}{\pi\left(\bar{\mathbf{X}} | \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) q\left(\bar{\mathbf{X}} \to \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\right)} \times \frac{b}{a}\right);$$

Death: Choose a point from $\bar{\mathbf{X}}$, say $\mathbf{x}^*$, and remove it from the current site set $\bar{\mathbf{X}}$ with probability p so that the new site set is $\bar{\mathbf{X}} \backslash \{\mathbf{x}^*\}$, where

$$p = \min\left(1, \frac{\pi\left(\bar{\mathbf{X}} \backslash \{\mathbf{x}^*\} | \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) q\left(\bar{\mathbf{X}} \backslash \{\mathbf{x}^*\} \to \bar{\mathbf{X}}\right)}{\pi\left(\bar{\mathbf{X}} | \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) q\left(\bar{\mathbf{X}} \to \bar{\mathbf{X}} \backslash \{\mathbf{x}^*\}\right)} \times \frac{a}{b}\right);$$

Exchange: Choose a point from $\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}$, say $\mathbf{x}^*$, and a point from $\bar{\mathbf{X}}$, say $\mathbf{x}^{**}$, and exchange the two points with probability p where

$$p = \min\left(1, \frac{\pi\left(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) q\left(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\} \to \bar{\mathbf{X}}\right)}{\pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right) q\left(\bar{\mathbf{X}} \to \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\}\right)}\right).$$

Regarding the choice of the proposal distribution $q\,(A \to B)$ used in the moves, we choose to use a uniform function which assigns equal weights to all the points in the sets $\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}$. For example, $q\left(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\} \to \bar{\mathbf{X}}\right)$ can be expressed as $\frac{1}{|\bar{\mathbf{X}}|+1}$ and $q\left(\bar{\mathbf{X}} \to \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\right)$ is equal to $\frac{1}{|\tilde{\mathbf{S}}|-|\bar{\mathbf{X}}|}$. Choosing the uniform proposal and expressing the posteriors as the product of the priors and the likelihoods, we will get the following forms for the acceptance probabilities of Birth, Death, and Exchange, respectively:

$$p_B = \min\left(1, \frac{\lambda|\bar{\mathbf{X}}|\left(|\tilde{\mathbf{S}}| - |\bar{\mathbf{X}}|\right) f\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}, \boldsymbol{\theta}\right)}{\left(|\bar{\mathbf{X}}| + 1\right)^2 f\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right)} \times \frac{b}{a}\right);$$

$$p_D = \min\left(1, \frac{|\bar{\mathbf{X}}|^2 f\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}\backslash\{\mathbf{x}^*\}, \boldsymbol{\theta}\right)}{\lambda\left(|\tilde{\mathbf{S}}| - |\bar{\mathbf{X}}| - 1\right)\left(|\bar{\mathbf{X}}| - 1\right) f\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right)} \times \frac{a}{b}\right);$$

$$p_E = \min\left(1, \frac{f\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\}, \boldsymbol{\theta}\right)}{f\left(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \boldsymbol{\theta}\right)}\right).$$

Other valid proposal distributions could be used as well; for a few other proposal distributions, see Liang et al. (2010).

Algorithm (2) presents the procedure of this RJMCMC, which generates a Markov chain of sites whose stationary distribution is $\pi\left(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}\right)$. Once we have $\left\{\bar{\mathbf{X}}_\mathbf{t}\right\}_{t=1}^{T}$ from the posterior distribution of $\bar{\mathbf{X}}_\mathbf{t}$, we can use equation (4.12) to make a prediction at any untried points. In Algorithm (2), the values a and b can be selected as

$a = b = \frac{1}{3}$, which means the probability of performing each of the operations Birth, Death, and Exchange is equal.

Algorithm 2 Bayesian Site Selection for Sparse Pseudo Input Gaussian Processes

```
Choose $\bar{\mathbf{X}}_\mathbf{0}$ as a uniformly random draw from $\tilde{\mathbf{S}}$ where $|\bar{\mathbf{X}}_\mathbf{0}| = M$
Optimize the value of $\boldsymbol{\theta}$ for a fixed $\bar{\mathbf{X}}_\mathbf{0}$
repeat
  Draw $u$ uniformly from $[0, 1]$
  if $u \leq a$ then
    Perform Birth
  else
    if $a < u \leq a + b$ then
      Perform Death
    else
      Perform Exchange
    end if
  end if
  After each $\kappa$ steps optimize the value of $\boldsymbol{\theta}$ for a fixed $\bar{\mathbf{X}}_\mathbf{t}$
  After burn-in steps store the values of $\bar{\mathbf{X}}_\mathbf{t}$
until The number of RJMCMC iterations reaches the pre-specified value of MCMC Length
```

To choose the optimal number and locations of the pseudo inputs, BSS's order of complexity for the training stage is $\mathcal{O}(LMN^2)$, and for the test stage is $\mathcal{O}(LM^2)$, where L is the length of the MCMC chain. Comparing the BSS's computation complexity with that of the full GP, which is $\mathcal{O}(N^3)$ for the training stage and $\mathcal{O}(N^2)$ for the test stage, one can see considerable cost reduction as long as M is chosen such that $M \ll N$. Comparing the BSS with the DSS, BSS costs more because L in the MCMC is longer than its counterpart in a gradient-based optimization, namely the number of optimization iterations l. However, BSS generally produces results with better accuracy than those of DSS, as will be demonstrated in Section 4.4.

Finally, it should be noted that since the BSS method is a Bayesian approach, it naturally provides the posterior distribution that inherently contains the information about uncertainty associated with the method. Once the method is established, one only needs to sample from its posterior distribution enough times to get the mean prediction and the confidence intervals.

4.3.2 Computational Details

As the new algorithm employs GP and MCMC methods, the reader can consult Rasmussen and Williams (2006) for general advice related to GP implementation and Liang et al. (2010) concerning MCMC.

The bottleneck for the computation comes from inverting the matrix of size N: (**i**) when we evaluate the likelihood in the Birth, Death and Exchange steps in RJMCMC, and (**ii**) when we evaluate the posterior predictive distribution after burn-in steps. The computation of evaluating the likelihood in RJMCMC is at the order of $\mathcal{O}(NM^2)$ (Snelson, 2007, p. 40), but the computation can be further reduced. Since the low-rank covariance in equation (4.6) after each move changes moderately, we can exploit the similar structures of the very past matrix to calculate the new one. To illustrate, assume we want to perform the Birth step. Let $\mathbf{x}_b$ be the newly-added point to the site set $\bar{\mathbf{X}}$. As such, the new low-rank covariance is

$$\mathbf{Q}^{new} = \mathbf{K}_{N(M+1)}\mathbf{K}_{M+1}^{-1}\mathbf{K}_{(M+1)N}, \tag{4.13}$$

which can be written as

$$\mathbf{Q}^{new} = \begin{bmatrix} \mathbf{K}_{NM} & \mathbf{K}_{N\mathbf{x}_b} \end{bmatrix} \begin{bmatrix} \mathbf{K}_M & \mathbf{K}_{M\mathbf{x}_b} \\ \mathbf{K}_{\mathbf{x}_b M} & K_{\mathbf{x}_b\mathbf{x}_b} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{K}_{MN} \\ \mathbf{K}_{\mathbf{x}_b N} \end{bmatrix}. \tag{4.14}$$

Let $k = K_{\mathbf{x}_b\mathbf{x}_b}$ and use the matrix inversion formula for a partitioned matrix (Rasmussen and Williams, 2006, p. 201), we get

$$\mathbf{Q}^{new} = \begin{bmatrix} \mathbf{K}_{NM} & \mathbf{K}_{N\mathbf{x}_b} \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_M & -\frac{1}{c}\mathbf{K}_M^{-1}\mathbf{K}_{M\mathbf{x}_b} \\ -\frac{1}{c}\mathbf{K}_{\mathbf{x}_b M}\mathbf{K}_M^{-1} & \frac{1}{c} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{MN} \\ \mathbf{K}_{\mathbf{x}_b N} \end{bmatrix} \tag{4.15}$$

where $c = k - \mathbf{K}_{\mathbf{x}_b M}\mathbf{K}_M{}^{-1}\mathbf{K}_{M\mathbf{x}_b}$ and $\mathbf{\Sigma}_M = \mathbf{K}_M{}^{-1} + \frac{1}{c}\mathbf{K}_M{}^{-1}\mathbf{K}_{M\mathbf{x}_b}\mathbf{K}_{\mathbf{x}_b M}\mathbf{K}_M{}^{-1}$. Therefore, to evaluate $\mathbf{Q}^{new}$ instead of inverting $\mathbf{K}_{M+1}$ as in equation (4.13), equation (4.15) allows us to use the inverse of $\mathbf{K}_M$ from the very previous step. Similar actions can be taken to facilitate the computation in the Death and Exchange steps.

4.3.3 Choices of Other Parameters

We share our thoughts in choosing the parameters when initializing the BSS algorithm. The first is what to choose for λ in the site prior (4.11). The effect of λ can be understood as follows. A small value of λ forces the algorithm to choose a smaller number of sites, while a large value of λ has the reverse effect. The latter typically results in a more accurate prediction but at the expense of longer computation time. Taking this trade-off into consideration, one can decide the value of λ by selecting a subset of data to train the model for different values of λ. Then the trained model can be used to predict the responses of another unused subset of data to observe the prediction accuracy. As the computation time can approximately be extrapolated over the whole data set, this method can provide us with the information about the accuracy and computation time trade-off as a result of choosing different values for λ. In our implementation, the value of λ is selected to be $1.5\frac{M}{10^t}d$ where M is the initial number of sites, d is the dimension of the input space, and t is selected from the interval $[1, 4]$.

The second parameter is the size of the discretized subspace, $\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}$, from which we choose the sites. The discretized subspace should be constructed based on the trade-off between computation time and accuracy. Larger $|\tilde{\mathbf{S}} \backslash \bar{\mathbf{X}}|$ yields more accurate results but could slow the computation. As a rule of thumb, one can choose $|\tilde{\mathbf{S}} \backslash \bar{\mathbf{X}}| = 10N$.

The next set of parameters is the initial locations of sites, decided in three steps: first, we randomly choose a subset of the training points; second, we find the new locations (while the number of pseudo inputs are kept fixed) by maximizing the marginal likelihood; and finally, we find the closest points in the discretized steps to those locations.

The other set of parameters is the range of the pseudo inputs, namely the lower and upper bounds k_l and k_u used in the site prior of equation (4.11). Our analysis shows that the values of k_l and k_u do not affect the method as long as the range is selected wide enough. However, if a user wants to prevent the algorithm from choosing a large number of locations (for instance, due to the time constraint), he/she can choose a relatively small number for the upper bound k_u. On the other hand, if the user wants to make sure the accuracy of the method is over some threshold, he/she can choose a relatively large number for the lower bound k_l, so that the algorithm will not choose a number of locations less than that limit. In the current implementation, the range is decided based on the initial number of pseudo inputs M and the number of data points in the training set N. Specifically, $k_l = \frac{M}{t_l}$, and $k_u = t_{u_1} M + \frac{N}{t_{u_2}}$ for some constants t_l, t_{u_1} and t_{u_2}.

Finally, we want to note that the stopping criterion for this algorithm is decided based on the trade-off between computation time and accuracy, not necessarily based on the probabilistic convergence of a Markov Chain. Doing so can be justified by noting two facts: first, the BSS is a Bayesian model averaging approach and, theoretically speaking, as the chain evolves, the results get more accurate. Second, letting the chain run for a very long time is counterproductive to the original purpose of the algorithm, which is to approximate the GP regression and provide a reasonable result in a relatively short period of time. As such, we would recommend that the stopping criterion be decided based on the presumed reduction in the MSE in a specified period of time, which can be evaluated through cross validation.

4.4 Experimental Results

In this section we present the results of the proposed method on some real and simulated datasets. First, we compare BSS with the full GP (FGP) and Treed Gaussian Processes (TGP) (Gramacy and Lee, 2008) for some small- to moderate-sized datasets. Then, we compare BSS with the DSS (Snelson, 2007) for some large datasets. Through both comparisons, we want to reinforce our claim that BSS provides a good trade-off: on one hand, it can handle the large datasets that full GP and TGP usually could not, while on the other hand, it is more accurate than DSS in terms of mean squared prediction errors. All the numerical studies were performed on a computer with two 3.16 GHz quadcore CPUs.

4.4.1 Datasets and Performance Criterion

We use four real datasets: the first two datasets are available at the UCI Machine Learning Repository (UCI, 2010), and the next two are NASA's satellite data. The first set is the *Abalone* data, which consists of 4,177 points and each data record has an input vector $\mathbf{x}$ of dimension $d = 7$. The response in the Abalone dataset is the abalone age, and its inputs in $\mathbf{x}$ are related to different properties in an abalone's body. The second set is the *Sarcos* data, which consists of 48,933 data points and $d = 27$. The data are related to the dynamics of a robot. The two NASA datasets, the third and fourth real datasets, are spatial data; both have $d = 2$. The third dataset is TCO, which consists of 48,331 measurements of the total column of ozone around the globe, collected by NIMBUS-7/TOMS satellite on October 1, 1988. The fourth data set is MOD08-CL, which is the data collected by the Moderate Resolution Imaging Spectroradiometer (MODIS) on NASA's Terra satellite. The data points, 64,800 in total, are the measurements of the average of cloud fractions around the globe from January to September in 2009.

We also test the proposed method on a set of simulated datasets generated using the revised Ackley's path function (Joseph and Kang, 2011), which is defined as

$$f(\mathbf{x}) = -\alpha \exp\left(-\beta\sqrt{\sum_{\ell=1}^{d}\frac{x_\ell^2}{d}}\right) - \exp\left(\sum_{\ell=1}^{d}\frac{\cos(\gamma x_\ell)}{d}\right) + \alpha + \exp(1), \quad \mathbf{x} \in [-2,2]^d, \tag{4.16}$$

where $\alpha = 2d$, $\beta = 0.2$, and $\gamma = 2\pi$. To generate a set of data, d and N need to be specified. Additionally, the locations of the data points need to be selected, and following Joseph and Kang (2011), we use the Latin Hypercube designs (Wu and Hamada, 2009) for this purpose.

To evaluate the performance of a method, we partition each simulated data set so that 80% is for training and the remaining 20% is for testing, and for the real datasets, we use a five-fold cross validation.

The primary evaluation criterion is the mean squared error (MSE), defined as

$$\text{MSE} = \sum_{i=1}^{N_t}\frac{(y_i - \hat{y}_i)^2}{N_t},$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value, and N_t is the number of test cases. When comparing BSS and DSS on the real datasets, in addition to MSE, which measures the accuracy of the mean prediction, we employ a predictive log score measure (PLSM) (Hoeting et al., 1999), which takes into account the predictive uncertainty of a method. PLSM is formally defined as

$$\text{PLSM} = -\sum_{\delta \in D^T} \log \left\{ \sum_{M \in \mathcal{A}} Pr(\delta | M, D^B) pr(M | D^B) \right\}, \tag{4.17}$$

where D^B and D^T are the build data (i.e. training data) and test data respectively. Specifically, $D^B = \{(\mathbf{x}_i, y_i); i = 1, 2, \ldots, N\}$ and $D^T = \{(\hat{\mathbf{x}}_i, \hat{y}_i); i = 1, 2, \ldots, N_t\}$, where $(\mathbf{x}_i, y_i)$ is the i^{th} input-output pair for training and $(\hat{\mathbf{x}}_i, \hat{y}_i)$ is the i^{th} input-output prediction pair in which $\hat{\mathbf{x}}_i$ is the test input and $\hat{y}_i$ is the corresponding prediction. The set $\mathcal{A}$ contains all the models used in prediction; for the BSS, $\mathcal{A}$ is the set of all pseudo input-based approximations based on the set of pseudo inputs $\bar{\mathbf{X}}_t$ for $t >$ burn-in, and for the DSS $\mathcal{A}$ is simply one approximation based on a fixed number of pseudo inputs. A smaller PLSM indicates an overall better predictive performance of a method, considering all uncertainties involved.

4.4.2 A One-dimensional Example

To illustrate how the algorithm works, we first generate a simulated dataset from equation (4.16) for $d = 1$ and $N = 5,000$ and apply the BSS algorithm with λ chosen to be 0.1. Figure 4.1 presents four plots showing the results. Figure 4.1 (a), the top-left plot, displays half of the 4,000 points selected as the training data set. Here we plot only the positive half as the revised Ackley's path is symmetric around the y-axis. Figure 4.1 (b), the bottom left plot, illustrates how the sites (locations of the pseudo inputs) change as the MCMC chain evolves. Figure 4.1 (c), the top-right plot, displays the number of sites versus the number of MCMC iterations. Considering the plots in Figure 4.1 (b) and (c), one can observe how the number and locations of sites change; during the first 1,000 iterations, the number of sites increase to around 110. After that, the number of sites oscillates around 110, and the locations change as a combined result of the Birth, Death and Exchange operations. Although the plot indicates that the number of sites at a given iteration might be the same as that of a few iterations before, the very locations of sites are not necessarily the same. The last plot, Figure 4.1 (d), displays how the MSE changes after the 1,000 burn-in iterations. Initially, we observe a significant reduction in MSE, then the decrease levels off. This example provides insights regarding how the BSS works: the method is initialized by selecting a number of sites, and through RJMCMC, BSS chooses different sites both in locations and quantity. In general, the predictive MSE has a decreasing trend and the number of sites converges toward a specific range, depending on the value of λ used.

4.4.3 Performance Comparison

We first use the simulated dataset from the revised Ackley's path to compare BSS with FGP, TGP and DSS. The dimension of the revised Ackley's path is fixed at $d = 10$ but two dataset sizes are used: $N = 1,000$ and $N = 5,000$. These dataset sizes are moderate, so that FGP and TGP can handle them. Figure 4.2 illustrates the results of the proposed method as well as that of the other three methods.

For BSS, three MSEs are presented in Figure 4.2, each of which corresponds to some pre-determined RJMCMC chain length. As the length of the chain increases, the MSE in general decreases. For DSS, the different results are due to the different initial locations (randomly selected) used in the gradient-based optimization. For TGP, we use the default parameter settings in its `R` package. The top part of the figure presents the results for $N = 1,000$. As evident in the figure, BSS produces MSE results close to those of FGP while spending less time. When we allow BSS to spend nearly the same time as FGP, it can produce smaller MSEs. DSS here uses a fixed number of sites, the same as the initial number of sites used in BSS,

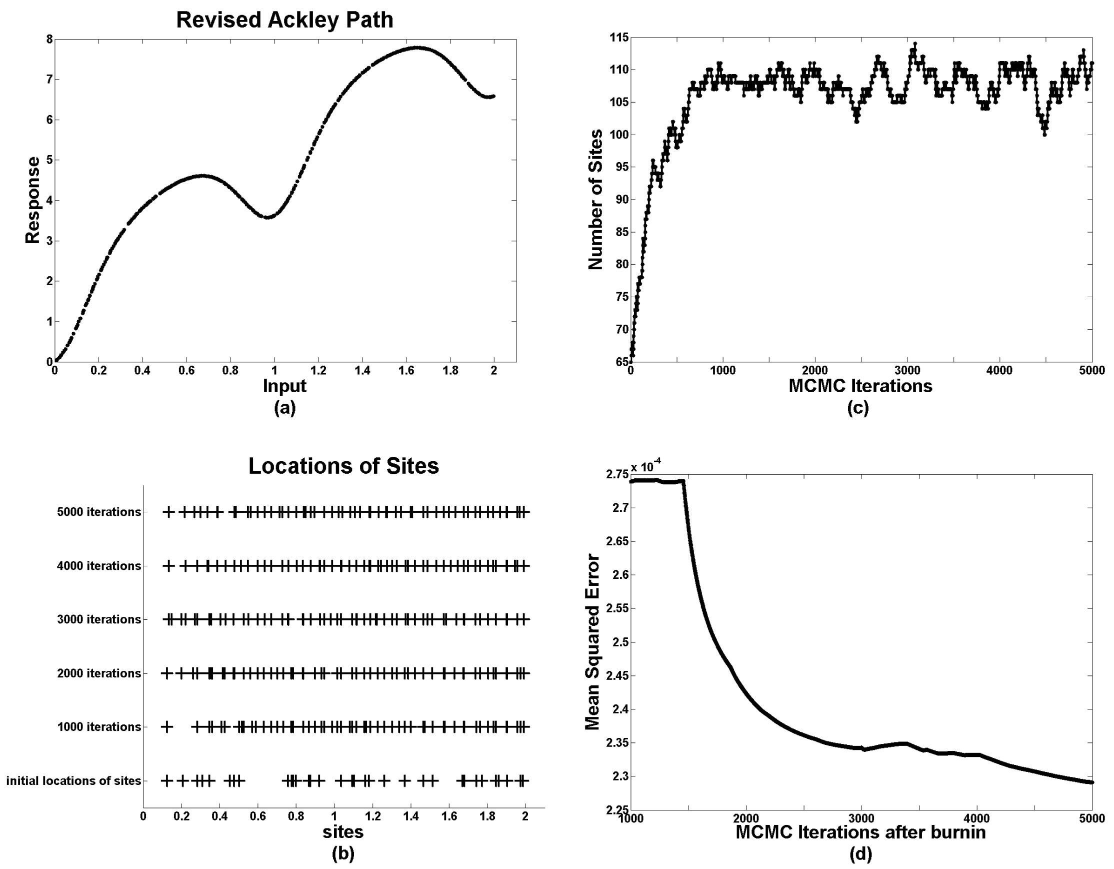

Figure 4.1: Left: (a) The revised Ackley's path when $d = 1$, and $N = 5,000$ (the plot only shows 2,000 training points whose inputs are positive). (b) The initial locations of sites and their new locations after every 1,000 MCMC iterations (on the positive side of the axis). Right: (c) Number of sites vs. MCMC iterations. (d) MSE vs. MCMC iterations after the burn-in period.

and then tries to find the optimal locations of those sites. As we see for this case, BSS obviously spends more time but produces more accurate predictions than DSS does. TGP produces very competitive MSE results but takes the longest time. The bottom part of the figure presents the results for $N = 5,000$, at which dataset size TGP takes too long to run, so the figure includes only the results of BSS, DSS, and FGP. The observations made earlier apply here too. In fact, compared to FGP, BSS performed noticeably better in less time. This could be due to the fact that the

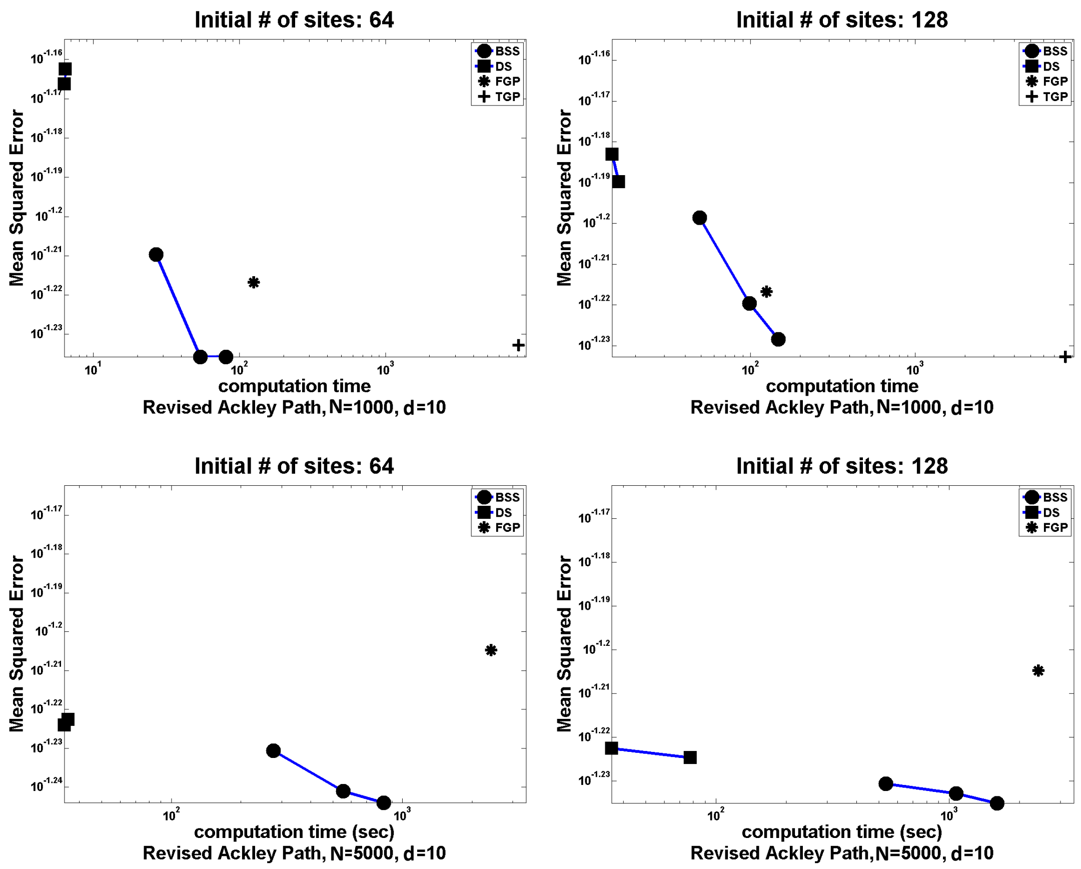

Figure 4.2: Top: The results of BSS comparing with DSS, FGP, and TGP for the revised Ackley's path with $d = 10$, and $N = 1,000$. In the left side plot, the number of initial sites for BSS and the (fixed) number of sites for DSS are 64, and on the right side they are 128. Bottom: The results of the BSS comparing with DSS and FGP for the revised Ackley's path with $d = 10$, and $N = 5,000$. In the left side plot, the number of initial sites for BSS and the (fixed) number of sites for DSS are 64, and on the right side they are 128.

dimension of the dataset ($d = 10$) is relatively high, which makes it difficult for FGP to learn the hyperparameters, and consequently, a poor estimate of hyperparameter hinders its performance. A summary of these results is also presented in Table 4.1.

We also compare BSS with DSS using the real, generally larger-sized data sets; for the datasets larger than 1,000 data points, FGP and TGP are too computationally expensive to run, so that we do not include FGP and TGP in the subsequent

Table 4.1: Summary of the results presented in Figure 4.2

	$N = 1,000, d = 10$		$N = 5,000, d = 10$	
Algorithm	Computation Time (sec)	MSE	Computation Time (sec)	MSE
BSS (64)	26.9	0.0617	277.1	0.0588
	53.9	0.0581	554.1	0.0574
	80.8	0.0581	831.1	0.0570
DSS (64)	6.4	0.0685	14.4	0.0656
	6.3	0.0688	15.8	0.0646
BSS (128)	49.2	0.0633	534.3	0.0588
	98.5	0.0603	1,068.5	0.0585
	147.6	0.0591	1,602.8	0.0582
DSS (128)	35.8	0.0597	77.6	0.0596
	35.8	0.0599	78.2	0.0593
FGP	125.2	0.0607	2,417.5	0.0624
TGP	8,145.0	0.0585	N/A	N/A

comparisons. To reach a more definite conclusion, we use a five-fold cross validation. The five-fold cross validation provides average MSE/PLSM values as well as their standard deviations from the five trials.

Table 4.2 presents the MSE results of BSS and DSS when both are applied to the four real datasets. As shown in the table, BSS always produces a smaller average MSE than DSS: on the two spatial data cases the reduction in MSE is impressive, around two-fold smaller than that of DSS; on *Sarcos* data, BSS provides a remarkable 35% decrease in the average MSE; and on *Abalone* data, the two methods performed similarly, especially considering the standard deviation of the MSE.

Table 4.2: Comparing BSS with DSS in terms of MSE. The numbers in the parentheses are standard deviations.

Data Set	Dimension	Number of data points	BSS	DSS
Abalone	7	4,177	4.4081 (0.2018)	4.4454 (0.2008)
Sarcos	27	48,933	0.0558 (0.0082)	0.0754 (0.0059)
MOD08-CL	2	64,800	0.0058 (0.0004)	0.0147 (0.0010)
TCO	2	48,331	197.8 (37.4)	337.2 (37.5)

Table 4.3 shows the PLSMs of the two methods for the data sets used in the section. It is observed that except for the Abalone data set, the BSS has a significantly smaller average PLSM than that of the DSS. Recall that PLSM measures the combined effect from the accuracy of the mean prediction and the predictive variance. Smaller PLSM and MSE values for *Sarcos*, TCO, and MOD08-CL datasets are strong indicators that BSS outperforms DSS not only in terms of mean prediction but also with lower overall uncertainty. On *Abalone* data, the two methods perform similarly: BSS and DSS have almost indistinguishable MSEs but BSS has a slightly worse PLSM.

Table 4.3: Predictive log score measure (PLSM). $M = 32$, and for BSS, the MCMC chain runs 2,000 iterations with 1,000 burn-in iterations. The numbers in the parentheses are standard deviations.

Data	PLSM - BSS	PLSM - DSS
Abalone	$8,600.1(40.13)$	$8,461.1(23.1)$
Sarcos	$-7,039.3(1,155.8)$	$-2,123.2(1,007.6)$
MOD08-CL	$-92,458.3(2,608.6)$	$-44,563.2(1,793.5)$
TCO	$177,695.2(4,112.2)$	$206,905.6(3,040.9)$

Admittedly, the improvement in accuracy by BSS comes with the cost of more computation time. The computation times of BSS, however, still reside in a region desirable for practical purposes. For example, BSS produces the results in less than 8 minutes for the *Abalone* data, 230 minutes for the *Sarcos*, and less than 100 minutes for both MOD08-CL and TCO data sets. Corresponding computation times for DSS are 10 seconds for *Abalone*, 8 minutes for *Sarcos* and around 3 minutes for MOD08-CL and TCO data sets. Should FGP be applied to a data set of a size similar to TCO, based on extrapolation from FGP's run times of solving smaller datasets, it would take more than 20 days.

4.4.4 Sensitivity Analysis

Recall that we embed an optimization procedure within the RJMCMC moves. Here we investigate how the running of the optimization would affect the results of BSS.

There are two parameters involved; one is the number of MCMC iterations between two consecutive optimizations of hyperparameters, which we denote by κ, while the second parameter, intuitively speaking, concerns how "well" we perform the optimization, which can be characterized by the number of gradient steps used in the optimization procedure. The number of gradient steps was denoted by l in DSS; here we adopt the same notation. Using the revised Ackley's path case with $d = 2$ and $N = 10,000$, we run an ANOVA taking κ and l as the factors. The value of κ is chosen from seven levels $\{1, 5, 10, 25, 50, 75, 100\}$ and l is chosen from six levels $\{10, 20, 40, 100, 150, 200\}$. Note that in DSS, the number of gradient search iterations is generally fixed around 200. The response values are the mean squared errors under each combination of factors. The value of λ is randomized so it would not have a significant effect on the responses. We run three replications in a full factorial design. Tabel 4.4 shows the resulting ANOVA table.

Table 4.4: ANOVA analysis for revised Ackley path with $d = 2$, $N = 10,000$ and 5,000 MCMC iterations

Source	Sum of squares	Degree of freedom	Mean of squares	F	Prob>F
κ	0.0046	6	0.0008	0.9850	0.4350
l	0.0053	5	0.0011	1.3397	0.2467
$\kappa \times l$	0.0156	30	0.0005	0.6630	0.9140
Error	0.2963	378	0.0008		
Total	0.3218	419			

The ANOVA results are presented in Table 4.4. This set of results is based on 5,000 MCMC iterations. This ANOVA table suggests none of the factors is significant under $\alpha = 0.05$. However, among the parameters investigated, the parameter l has the smallest p-value. Based on other experiments conducted on the same data set, l may be found significant under $\alpha = 0.05$ when shorter MCMC chains were used. We believe that this analysis suggests that if the length of the MCMC chain is long enough, the BSS method becomes less sensitive (or insensitive) to the change in parameters. This conclusion is supported by most of the other data sets we used.

To gain more insight concerning the effect of the parameters, we decide to look further into the behavior of the algorithm for different values of κ and l. Figure 4.3 shows the change in the MSE, the normalized computation time and the number of sites for different values of κ and l as the chain evolves, based on a MCMC chain truncated at 2,000 for the revised Ackley's path with $d = 2$ and $N = 10,000$. The normalized computation time is the computation time under a combination of κ and l, normalized by dividing the longest computation time among all possible κ-l combinations.

We would like to make a number of observations in Figure 4.3: (1) Intuitively people might think a larger l leads to a smaller MSE since a larger l means a deeper optimization of $\boldsymbol{\theta}$ at each iteration. In reality, it turns out a smaller l helps reduce the MSE more. The reason behind this is because a model's MSE depends much more on the number and locations of the sites than the optimization of $\boldsymbol{\theta}$. Long iterations in the optimization routine could overfit the data with a smaller number of sites and a refined $\boldsymbol{\theta}$, which may very well end up with a higher MSE. (2) Given the observation in (1), in order to get smaller MSE values, people would understandably use a smaller l (e.g., $l = 10$) . The other factor of consideration is the computation. On the surface, a smaller l could mean a fast computation, and pairing with the small l, a large κ should be chosen to further reduce the computation. Although the large κ choice is generally correct, a small l does not necessarily lead to fast computation. For example, for the combination $\kappa = 1$ and $l = 10$ the algorithm has longer computation time and results in a smaller MSE, as compared to $\kappa = 1$ and $l = 100$. The reason is again the number of sites the algorithm chooses, and the computation depends much more on the number of sites than the value of l. Generally speaking, a smaller l causes the selection of a larger number of sites, leading to a smaller MSE but causing a longer computation. This can be seen by comparing Figure 4.3 (b) and (c). Our experience indicates that a general practice is to choose a relatively large l, say $l = 40$, paired with a large κ, say $\kappa = 25$, that can arrive at a good compromise between prediction quality and computation expense.

4.5 Summary

This section presents an approximation algorithm to reduce the computation time of GP regression when dealing with large datasets. We tackle the problem by trying to approximate the likelihood function using a set of artificial data points and labeling them as "sites." We devise a Bayesian site selection method and solve it using the reversible jump MCMC algorithm, to find simultaneously the locations and number of the sites. Our method can handle large datasets with general dimensions and outperforms the deterministic site selection method, which decides the locations of sites but with the site number fixed *a priori*.

As evident in the case studies presented in Section 4.4, the proposed BSS method produces similar or even smaller MSEs as compared to full GP, while being able to do so faster. It can handle large datasets that full GP is not practically able to handle, while producing MSEs smaller than the deterministic site selection method. The computation time related to BSS can be reduced further by reducing the number of

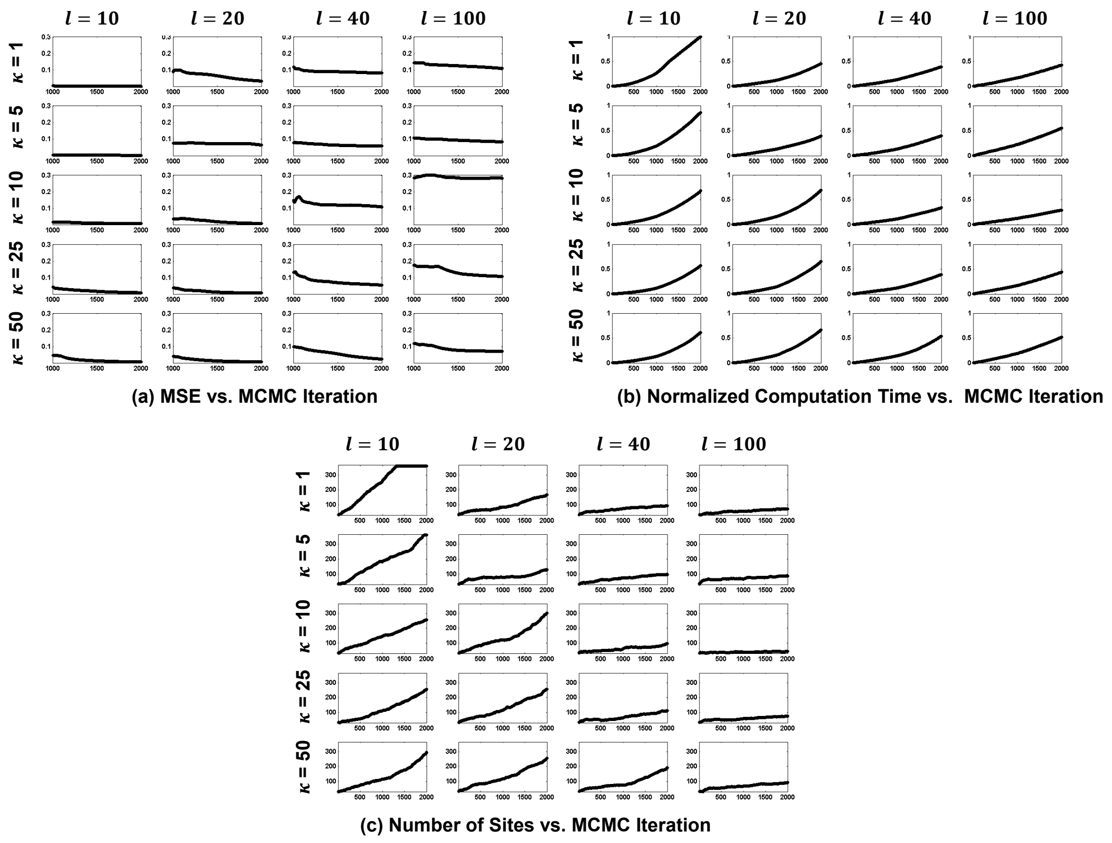

Figure 4.3: (a) MSE vs. MCMC iterations (after burn-in period) for different values of κ and l for the revised Ackley's path with $d = 2$ and $N = 10,000$. (b) Normalized computation time vs. MCMC iterations. (c) Number of sites vs. MCMC iterations.

MCMC iterations in RJMCMC. Understandably, this may come at the expense of a decrease in prediction accuracy. Our current analysis indicates that with appropriate priors chosen, BSS generally provides a good trade-off between the two conflicting objectives.

5. CONCLUSION

This study has discussed applications of Gaussian processes to solve three problems that involve Big Data: local wind field modeling in a wind farm, buckypaper nano-manufacturing, and applying GP regression to large datasets. This section presents the insights we gained from applying Gaussian processes (GPs) to solve these three problems that involve Big Data. Below, we summarize our findings and propose some new areas for future research.

5.1 Summary

To solve the first problem related to improving the short-term forecasting of wind speed, we constructed three different spatial-temporal models: GSTAR, RGSTAR, and RGSTARGW. All three used a Gaussian kernel to model spatial dependency in a local region and autoregressive components for linking past observations to the current observation. RGSTAR also incorporated wind direction by introducing regimes. RGSTARGW incorporated meteorological measurements by calculating the geostrophic wind in the region under study. We showed how the three models closely related to Gaussian Markov processes and could be interpreted as a parametrized case of vector autoregressive models. We presented each model as an optimization problem that could be solved using numerical techniques.

To solve the second problem related to predicting the mechanical properties of buckypaper, a nano-manufacturing product, we calibrated an existing simulation model which used accurate, but costly, physical experiments. To overcome the challenge of some unobserved input variables in the physical experiments, we proposed a calibration framework introducing latent variables which were imputed using a functional relationship between the observed and unobserved variables. We developed an algorithm which sequentially updated the parameters in the model. Empirical results demonstrated that the proposed model outperformed existing single-fidelity or multi-fidelity analysis without latent variables.

To solve the third problem of approximating the GP regression to strike a sensible balance between prediction accuracy and computation time, we proposed to improve the prediction accuracy of existing likelihood approximation methods. Our framework modeled the covariance of the GP using latent variables called pseudo-inputs. The pseudo-inputs were treated as an extra parameter in the model, and their number and locations were determined simultaneously using a reversible jump MCMC method. Empirical results showed the method achieves a sensible balance between prediction accuracy and computation time.

We summarize the major contributions of this study as follows:

- Spatial-temporal analysis of systems with dense measurements: This study proposes a methodology for eliciting pertinent information from a network of dense observations in time and space. The novelty of the idea is how to identify the information that propagates in a small region through efficient space-time modeling. The method can serve as the basis for improving O&M and planning strategies for a wind power system, because the proposed models allow single turbines to be isolated, which in turn implies that each turbine can be treated separately for control and maintenance. The model also can be applied to other spatial-temporal systems having a set of time-series that are highly dense in space.

- Local calibration of complex computer codes: The proposed methodology for the calibration of simulation experiments involving latent variables can be understood as a task of the local calibration of parameters in physical experiments.

In fact, the latent variables introduced in the methodology can be conceived as parameters in the physical experiments which are dependent on other variables and therefore need to be locally calibrated. The proposed method can be applied to other engineering applications requiring local calibration with less cost and time.

- Predictive modeling for large-scale and complex systems: The BSS method is an efficient algorithm to use as a predictive model for any complex system. The proposed model was constructed upon a GP-based framework, therefore making weak assumptions about the structure of the system of interest. There is no restriction for the dimension of the dataset. The BSS method can be used for spatial data and as a surrogate model for complex and large-scale computer codes. Allowing the user to decide the trade-off between computation time and prediction accuracy makes the BSS method appealing for many practical purposes.

5.2 Suggestions for Future Research

Based on each separate problem addressed in this study, we present the following suggestions for future research:

- Short-term Wind Speed Forecast Using Measurements from Multiple Turbines in a Wind Farm:

  GSTAR models do not always outperform the persistent model for very short horizons, such as for a two-hour ahead prediction. This suggests the need to develop more sophisticated modeling to capture such temporal dependency, instead of simple linear relationships as manifested in the low order of temporal process parameter, p. For the highly volatile near-ground wind field, we conjecture that any attempts to model the temporal dependency as a linear function will fail. Nonlinear models in the mode of Giannakis and Majda (2012) may be more capable of handling the nonlinear dependency for meteorological data, but how this type of method can be transformed to the wind field modeling is not straightforward and thus represents an interesting research pursuit. We suggest that the utility of the proposed methods for short-term wind forecasts can be employed for other important applications, such as reconstructing the wind field and analyzing the wake effect. We suggest using the GSTAR models for simulating local wind fields, based on point-wise measurements at turbine sites, and providing certain understanding of how the rotor motion of one turbine can affect the performance of neighboring turbines. This analysis should prove useful for designing the layout of wind farms.

- Modulus Prediction of Buckypaper based on Multi-fidelity Analysis Involving Latent Variables:

  We have modeled the relationship between unobserved and latent variables via equation (3.10). An interesting alternative approach would be to utilize an EM algorithm (Dempster et al., 1977) to impute the unobserved variables. We note that doing so would not be straightforward, since it would require making assumptions on the distribution of unobserved input variables and expressing the optimization problem in (3.1) in terms of likelihood maximization. In addition, for practical purposes, we suggest it would be useful to develop guidelines to evaluate the similarity between the simulation outputs and the physical responses, which then could be used to justify the action of integrating the simulation and physical responses. One work alluding to this aspect is Xiong et al. (2013), which sets a threshold on testing the cross-validation error for continuation in a sequential design. Using this cross-validation measure sheds light on how a multi-fidelity model improves the predictive outcome, but leaves open the issue of whether a multi-fidelity design is “worth it” or not until the cross-validation error is computed (which can only be done after the multi-fidelity model is established). We believe that this is an unsettled issue needing attention from the academic community.

- Bayesian Site Selection for Fast Gaussian Process Regression:

  One possible improvement is a full Bayesian treatment that updates $\boldsymbol{\theta}$ the same time as the site locations $\bar{\mathbf{x}}$s vs. using the gradient method for $\boldsymbol{\theta}$ within the RJMCMC iterations. That is to say, in each Birth, Death, or Exchange step, in addition to proposing a value for $\bar{\boldsymbol{x}}$, we also could propose a value for $\boldsymbol{\theta}$ and use the ratio test to either accept or reject it. Not yet resolved is the proposal distribution to use. The typical distributions we have tested have been ineffective. Another research path involves making BSS choose the sites from a continuous subspace vs. from a discretized subspace in its current version. We suggest that lifting the site selection restriction should enhance the performance of BSS in high-dimensional data problems.

REFERENCES

Ailliot, P. and V. Monbet (2012). Markov-switching autoregressive models for wind time series. *Environmental Modelling & Software 30*, 92–101.

Alexiadis, M. C., P. S. Dokopoulos, and H. S. Sahsamanoglou (1999). Wind speed and power forecasting based on spatial correlation models. *Energy Conversion, IEEE Transactions on 14*(3), 836–842.

Bayarri, M. J., J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu (2007). A framework for validation of computer models. *Technometrics 49*(2), 138–154.

Beckers, J.-M. and M. Rixen (2003). EOF calculations and data filling from incomplete oceanographic datasets. *Journal of Atmospheric and Oceanic Technology 20*(12), 1839–1856.

Bessa, R. J., V. Miranda, A. Botterud, J. Wang, and E. M. Constantinescu (2012). Time adaptive conditional kernel density estimation for wind power forecasting. *IEEE Transactions on Sustainable Energy 3*(4), 660–669.

Blanco, M. I. (2009). The economics of wind energy. *Renewable and Sustainable Energy Reviews 13*(6), 1372–1382.

Brockwell, P. J. and R. A. Davis (2009). *Time Series: Theory and Methods.* Springer.

Brown, B. G., R. W. Katz, and A. H. Murphy (1984). Time series models to simulate and forecast wind speed and wind power. *Journal of Climate and Applied Metereology 23*, 1184–1195.

Cassola, F. and M. Burlando (2012). Wind speed and wind energy forecast through kalman filtering of numerical weather prediction model output. *Applied Energy 99*, 154–166.

Clifton, L., D. A. Clifton, M. A. Pimentel, P. J. Watkinson, and L. Tarassenko (2013). Gaussian processes for personalized e-health monitoring with wearable sensors. *IEEE Transactions on Biomedical Engineering 60*(1), 193–197.

Crespo, A., J. Hernandez, and S. Frandsen (1999). Survey of modelling methods for wind turbine wakes and wind farms. *Wind Energy 2*(1), 1–24.

Cressie, N. and C. K. Wikle (2011). *Statistics for Spatio-Temporal Data*. New York City: Wiley.

Crochet, P. (2004). Adaptive Kalman filtering of 2-metre temperature and 10-metre wind-speed forecasts in Iceland. *Meteorological Applications 11*(2), 173–187.

Damianou, A. and N. Lawrence (2013). Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 207–215.

Daniel, A. and A. Chen (1991). Stochastic simulation and forecasting of hourly average wind speed sequences in jamaica. *Solar Energy 46*(1), 1–11.

de Luna, X. and M. G. Genton (2005). Predictive spatio-temporal models for spatially sparse environmental data. *Statistica Sinica 15*(2), 547–568.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B 39*(1), 1–38.

Draguljć, D., A. M. Dean, and T. J. Santner (2012). Non-collapsing space-filling designs for bounded nonrectangular regions. *Technometrics 54*(2), 169–178.

Erdem, E. and J. Shi (2011). ARMA based approaches for forecasting the tuple of wind speed and direction. *Applied Energy 88*(4), 1405–1414.

Focken, U. and M. Lange (2006). *Physical Approach to Short-Term Wind Power Prediction*. Springer.

Furrer, R., M. G. Genton, and D. Nychka (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics 15*, 502–523.

Giannakis, D. and A. J. Majda (2012). Nonlinear laplacian spectral analysis for time series with intermittency and low-frequency variability. *Proceedings of the National Academy of Sciences 109*(7), 2222–2227.

Giebel, G., R. Brownsword, G. Kariniotakis, M. Denhard, and C. Draxl (2011). The state-of-the-art in short-term prediction of wind power: A literature overview. Technical report, Risø DTU National Laboratory for Sustainable Energy, Roskilde, Denmark.

Gneiting, T. (2002). Compactly supported correlation funtion. *Journal of Multivariate Analysis 83*(2), 493–508.

Gneiting, T. (2011). Quantiles as optimal point forecasts. *International Journal of Forecasting 27*(2), 197–207.

Gneiting, T., K. Larson, K. Westrick, M. G. Genton, and E. Aldrich (2006). Calibrated probabilistic forecasting at the stateline wind energy center. *Journal of the American Statistical Association 101*(475), 968–979.

Goldstein, M. and J. Rougier (2006). Bayes linear calibrated prediction for complex systems. *Journal of the American Statistical Association 101*(475), 1132–1143.

Goldstein, M. and J. Rougier (2009). Reified bayesian modelling and inference for physical systems. *Journal of Statistical Planning and Inference 139*(3), 1221–1239.

Golub, G. H. and C. F. Van Loan (2012). *Matrix Computations, 3rd edition.* Johns Hopkins University Press.

Gramacy, R. B. and H. K. H. Lee (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association 103*, 1119–1130.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and

Bayesian model determination. *Biometrika 82*, 711–732.

Han, G., T. J. Santner, and J. J. Rawlinson (2009). Simultaneous determination of tuning and calibration parameters for computer experiments. *Technometrics 51*(4), 464–474.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer.

He, M., L. Yang, J. Zhang, and V. Vittal (2013). A spatio-temporal analysis approach for short-term forecast of wind farm generation. *IEEE Transactions on Power Systems*. in press.

Hensman, J., N. Fusi, and N. D. Lawrence (2013). Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pp. 282–290.

Hering, A. S. and M. G. Genton (2007). Blowing in the wind. *Significance 4*, 11–14.

Hering, A. S. and M. G. Genton (2010). Powering up with space-time wind forecasting. *Journal of the American Statistical Association 105*(489), 92–104.

Higdon, D., M. Kennedy, J. C. Cavendish, J. A. Cafeo, and R. D. Ryne (2004). Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing 26*(2), 448–466.

Hoeting, J. A., D. Madigan, A. E. Raftery, and C. T. Volinsky (1999). Bayesian model averaging: A tutorial. *Statistical Science 14*(4), 382–417.

Huang, Z. and Z. S. Chalabi (1995). Use of time-series analysis to model and forecast wind speed. *Journal of Wind Engineering and Industrial Aerodynamics 56*(2), 311–322.

Hunter, J. K. (2009). Lecture Notes on Applied Mathematics, https://www.math.ucdavis.edu/ hunter/m280-09/ch.pdf. last time accessed March 2014.

Iijima, S. (1991). Helical microtubules of graphitic carbon. *Nature 354*(6348), 56–58.

Johansen, S. (1995). *Likelihood-based Inference in Cointegrated Vector Autoregressive Models*. Cambridge Univ Press.

Johnson, M., L. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference 26*(2), 131–148.

Joseph, V. and L. Kang (2011). Regression-based inverse distance weighting with applications to computer experiments. *Technometrics 53*, 254 –265.

Joseph, V. R. and S. N. Melkote (2009). Statistical adjustments to engineering models. *Journal of Quality Technology 41*(4), 362–375.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering 82*(1), 35–45.

Kamal, L. and Y. Z. Jafri (1997). Time series models to simulate and forecast hourly averaged wind speed in Quetta, Pakistan. *Solar Energy 61*(1), 23–32.

Katz, R. W. and R. H. Skaggs (1981). On the use of autoregressive-moving average processes to model meteorological time series. *Monthly Weather Review 109*(3), 479–484.

Kennedy, M. C. and A. O'Hagan (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika 87*(1), 1–13.

Kennedy, M. C. and A. O'Hagan (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63*(3), 425–464.

Kleijnen, J. P. (2007). *Design and Analysis of Simulation Experiments*. Springer.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa 52*(6), 119–139.

Kusiak, A. and W. Li (2010). Estimation of wind speed: A data-driven approach.

Journal of Wind Engineering and Industrial Aerodynamics 98(10), 559–567.

Le Cam, L. and G. L. Yang (2000). *Asymptotics in Statistics: Some Basic Concepts* (second ed.). Springer.

Liang, F., C. Liu, and R. Carroll (2010). *Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples*. John Wiley and Sons.

Liu, H., H.-Q. Tian, and Y.-F. Li (2012). Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction. *Applied Energy 98*, 415–424.

Louka, P., G. Galanis, N. Siebert, G. Kariniotakis, P. Katsafados, I. Pytharoulis, and G. Kallos (2008). Improvements in wind speed forecasts for wind power prediction purposes using kalman filtering. *Journal of Wind Engineering and Industrial Aerodynamics 96*(12), 2348–2362.

Maadooliat, M., J. Z. Huang, and J. Hu (2013). Integrating data transformation in principal components analysis. *Journal of Computational and Graphical Statistics*. Under Review.

Matheron, G. (1973). The intrinsic random functions and their applications. *Advances in Applied Probability 5*(3), 439–468.

Mauricio, J. A. (1995). Exact maximum likelihood estimation of stationary vector ARMA models. *Journal of the American Statistical Association 90*(429), 282–291.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B (Methodological) 40*(1), 1–42.

Øksendal, B. (2003). *Stochastic Differential Equations*. Springer.

Palomares-Salas, J. C., J. J. G. De la Rosa, J. G. Ramiro, J. Melgar, A. Agera, and A. Moreno (2009). ARIMA vs. Neural networks for wind speed forecasting. In *Proceedings of the 2009 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, Hong Kong, pp. 129–133.

Park, C., J. Z. Huang, and Y. Ding (2011). Domain decomposition approach for fast Gaussian process regression of large spatial data sets. *Journal of Machine Learning Research 12*, 1697–1728.

Pérez-Cruz, F., S. Van Vaerenbergh, J. J. Murillo-Fuentes, M. Lázaro-Gredilla, and I. Santamaria (2013). Gaussian processes for nonlinear signal processing: An overview of recent advances. *Signal Processing Magazine, IEEE 30*(4), 40–50.

Pinson, P. (2012). Very-short-term probabilistic forecasting of wind power with generalized logit-normal distributions. *Journal of the Royal Statistical Society: Series C (Applied Statistics) 61*(4), 555–576.

Pourhabib, A., J. Z. Huang, K. Wang, C. Zhang, B. Wang, and Y. Ding (2014). Modulus prediction of buckypaper based on multi-fidelity analysis involving latent variables. *IIE Transactions*. in press.

Pourhabib, A., F. Liang, and Y. Ding (2014). Bayesian site selection for fast Gaussian process regression. *IIE Transactions 46*(5), 543–555.

Qian, P. Z. (2012). Sliced latin hypercube designs. *Journal of the American Statistical Association 107*(497), 393–399.

Qian, P. Z. and C. J. Wu (2009). Sliced space-filling designs. *Biometrika 96*(4), 945–956.

Qian, P. Z. G. and C. F. J. Wu (2008). Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. *Technometrics 50*(2), 192–204.

Qian, Z., C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. F. J. Wu (2006). Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design 128*(4), 668–677.

Quiñonero-Candela, J. and C. E. Rasmussen (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research 6*, 1939–1959.

Ramsay, J. (1998). Estimating smooth monotone functions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 60*(2), 365–375.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning.* MIT Press.

Reese, C. S., A. G. Wilson, M. Hamada, H. F. Martz, and K. J. Ryan (2004). Integrated analysis of computer and physical experiments. *Technometrics 46*(2), 153–164.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science 4*(4), 409–423.

Santner, T. J., B. J. Williams, and W. I. Notz (2003). *The Design and Analysis of Computer Experiments.* Springer Verlag.

Santos, R. A. (2007). *Damage Mitigation Control for Wind Turbines.* Dissertation, University of Colorado, Boulder, CO.

Schlink, U. and G. Tetzlaff (1998). Wind speed forecasting from 1 to 30 minutes. *Theoretical and Applied Climatology 60*(1-4), 191–198.

Seeger, M., C. K. I. Williams, and N. D. Lawrence (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Workshop on Artificial Intelligence and Statistics 9.* Society for Artificial Intelligence and Statistics.

Snelson, E. (2007). *Flexible and Efficient Gaussian Process Models for Machine Learning.* Dissertation, Gatsby Computational Neuroscience Unit University College London, London, United Kingdom.

Snelson, E. and Z. Ghahramani (2006). Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems 18*, 1257–1264.

Snelson, E. and Z. Ghahramani (2007). Local and global sparse Gaussian process approximations. In *International Conference on Artifical Intelligence and Statistics 11.* Society for Artificial Intelligence and Statistics.

Stinstra, E., P. Stehouwer, D. den Hertog, and A. Vestjens (2003). Constrained maximin designs for computer experiments. *Technometrics 45*(4), 340–346.

Tastu, J., P. Pinson, P. J. Trombe, and H. Madsen (2014). Probabilistic forecasts of wind power generation accounting for geographically dispersed information. *IEEE Transactions on Smart Grid* (5), 480–489.

Torres, J., A. Garca, M. De Blas, and A. De Francisco (2005). Forecast of hourly average wind speed with ARMA models in Navarre Spain. *Solar Energy 79*(1), 65–77.

Tsai, C., C. Zhang, J. A. David, B. Wang, and R. Liang (2011). Elastic property prediction of single-walled carbon nanotube buckypaper/polymer nanocomposites: stochastic bulk response modeling. *Journal of Nanosience and Nanotechnology 11*(3), 2132–2141.

UCI (2010). UCI Machine Learning Repository, http://archive.ics.uci.edu/ml. last time accessed January 2014.

Wan, C., Z. Xu, P. Pinson, Z. Y. Dong, and K. P. Wong (2014). Optimal prediction intervals of wind power generation. *IEEE Transactions on Power Systems 29*(3), 1166–1174.

Wang, K., A. Vanli, C. Zhang, B. Wang, and Z. Liang (2013). Model calibration and adjustment of 2D Truss model for predicting youngs modulus of Poly(vinyl alcohol)-enhanced carbon nanotube sheet. Working paper, Submitted to ASME Manufacturing Science and Engineering.

Wang, Z., Z. Liang, B. Wang, C. Zhang, and L. Kramer (2004). Processing and property investigation of single-walled carbon nanotube (swnt) buckypaper/epoxy resin matrix nanocomposites. *Composites Part A: Applied Science and Manufacturing 35*(10), 1225–1232.

Wiener, N. (1964). *Extrapolation, Interpolation, and Smoothing of Stationary Time*

Series: With Engineering Applications, Volume 8. MIT press.

Wikle, C. K. and L. M. Berliner (2005). Combining information across spatial scales. *Technometrics 47*, 80–91.

Wu, C. F. J. and M. S. Hamada (2009). *Experiments: Planning, Analysis, and Optimization, 2nd Edition.* John Wiley and Sons.

Xia, H. (2008). *Bayesian Hierarchical Model for Combining Two-resolution Metrology Data.* Dissertation, Texas A&M University, College Station, TX, USA.

Xia, H., Y. Ding, and B. Mallick (2011). Bayesian hierarchical model for combining misaligned two-resolution metrology data. *IIE Transactions 43*(4), 242–258.

Xiong, S., P. Z. G. Qian, and C. F. J. Wu (2013). Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics 55*(1), 37–46.

Xiong, Y., W. Chen, K.-L. Tsui, and D. W. Apley (2009, March). A better understanding of model updating strategies in validating engineering models. *Computer Methods in Applied Mechanics and Engineering 198*(15-16), 1327–1337.

Yu, Z. and A. Tuzuner (2008). Wind speed modeling and energy production simulation with weibull sampling. In *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pp. 16.

Zhang, C., K. Wang, Z. Liang, and B. Wang (2011). Fast filtration synthesis method for buckypaper using highly concentrated carbon nanotube slurry. Informs Annual Meeting, Charlotte, NC, Nov. 13-16, 2011.

Zhu, B. and D. B. Dunson (2013). Locally adaptive bayes nonparametric regression via nested gaussian processes. *Journal of the American Statistical Association 108*(504), 1445–1456.

Zhu, X. (2013). *Wind Forecasting for Power System Operation.* Dissertation, Texas A&M University, College Station, TX.

Zhu, X. and M. G. Genton (2012). Short-term wind speed forecasting for power system operations. *International Statistical Review 80*(1), 2–23.