UNIVERSITY OF CALIFORNIA
SANTA CRUZ

COMBINING MULTIVARIATE STOCHASTIC PROCESS MODELS WITH FILTER METHODS FOR CONSTRAINED OPTIMIZATION

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

STATISTICS AND APPLIED MATHEMATICS

by

Tony Pourmohamad

March 2016

The Dissertation of Tony Pourmohamad
is approved:

Professor Herbert K. H. Lee, Chair

Professor Bruno Sansó

Professor Qi Gong

Dr. Stefan Wild

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Combining Multivariate Stochastic Process Models with Filter Methods for Constrained Optimization

by

Tony Pourmohamad

Expensive black box systems arise in many engineering applications but can be difficult to optimize because their output functions may be complex, multi-modal, and difficult to understand. The task becomes even more challenging when the optimization is subject to constraints and no derivative information is available. In this dissertation, we combine response surface modeling, sequential Monte Carlo, and filter methods in order to solve problems of this nature. Furthermore, we propose a new model for correlated outputs of mixed type. Our modeling framework extends Gaussian process methodology for modeling of continuous multivariate spatial outputs by adding a latent process structure that allows for joint modeling of a variety of types of correlated outputs. In addition, we implement fully Bayesian inference using particle learning, which allows us to conduct fast sequential inference. By employing a filter algorithm for solving constrained optimization problems, we also establish two novel probabilistic metrics for guiding the filter. We extend these ideas to a multidimensional filter that outperforms the traditional filter method. Overall, this hybridization of statistical modeling and nonlinear programming efficiently utilizes both global and local search in order to converge to a global solution to the constrained optimization problem. To demonstrate the effectiveness of the proposed methods, we perform

numerical tests on both synthetic examples and real-world hydrology computer experiment optimization problems.

To Josephine and Sophia, my loves and inspirations.

Acknowledgments

Completing this dissertation, and ultimately surviving graduate school, would not have been possible, or as much fun, without the support of many people.

First and foremost, I would like to thank my advisor, Herbie Lee, for his constant support and guidance throughout my Ph.D. studies. Herbie has truly been the best advisor I could have ever asked for, and I am very lucky to have been able to work with him all of these years. Herbie has taught me a tremendous amount about being a good graduate student, writing, conferences, and of course, Bayesian statistics. I am forever grateful to Herbie for everything he has taught me, and for his guidance in finishing my Ph.D. His continued mentorship and friendship will be something that I will forever cherish.

Outside of my advisor, no other professor in the AMS department has impacted me more than Bruno Sansó. Thank you Bruno for always reminding me that 1) there is always a bug in the code, and 2) that there is a special pot of boiling olive oil awaiting me in a certain place. But seriously, thank you Bruno for teaching me a ton about spatial statistics, for being a great mentor and a great friend. I will especially miss our 30 minute "work discussions" in the hall about baseball.

Thank you to the rest of my committee members, Qi Gong and Stefan Wild, for their many insightful comments during my advancement that contributed to much of the work in this dissertation. Thank you two for your willingness to serve on my defense committee. This dissertation has greatly benefited from it.

Ever since arriving at UC Santa Cruz I have felt a sense of community and family within the AMS department. All of the faculty members and graduate students in the

department have left a lasting impact on me both intellectually and personally. I am forever indebted to every member of the AMS department for everything they have taught me about statistics and life outside of statistics. Especial thanks to Marc Mangel for the many thoughtful conversations over B&C and for allowing me *squatters rights* in his lab.

I could not have had as much fun during my graduate studies without the friendship of Robert Richardson and Juan Lopez. Thank you Robert for your willingness to always lend a helping hand when I was stuck on any problem and for always agreeing to be our 3rd in a game of cards. Thank you Juan for teaching me that beer boosts creative thinking and that coffee allows you to focus once you have the creative idea. The endless hours we spent arguing in front of a white board has played a critical component in helping me finish and improve this dissertation.

Thank you to all my friends and family that have been a never ending source of support. In particular, thank you to my parents Houshang and Marilyn for their unconditional love and support, and for fostering my scientific curiosity at a young age. You two have always allowed me to pursue my dreams and I am eternally grateful for it.

Last but not least, thank you to my amazing wife Josie. You are my best friend and greatest supporter, and everything I have achieved up until this point could not have been done without you. I cannot fathom what it is like to be married to a graduate student, but somehow, you were always able to love, support and tolerate me. Thank you for not killing me.

Chapter 1

Introduction

Computer models have become a ubiquitous tool for the study of complex scientific phenomena. A computer model (or code) is a mathematical model that simulates the complex phenomena, or system, under study via a computer program. Controlled experiments, once considered to be a de facto standard in statistics, are not a viable means for studying complex phenomena when the systems under study are either too expensive, too time consuming, or physical experimentation is simply not possible. For example, weather phenomena, such as hurricanes or global warming, are not reproducible physical experiments, therefore, computer models based on climatology are used to study these events. Thus, researchers hoping to better understand and model complex phenomena should consider computer modeling as a possible solution.

Computer models have enjoyed a wide range of use, spanning disciplines such as physics, astrophysics, climatology, chemistry, biology, and engineering. At its simplest, a

computer model is a mathematical model of the form

$$y = f(x_1, \ldots, x_d) = f(\mathbf{x}), \quad \mathbf{x} = (x_1, \ldots, x_d)^T \in \mathcal{X}, \tag{1.1}$$

where $\mathbf{x}$ is an input variable to the computer model, y is a (possibly multivariate) output from the computer model, and $\mathcal{X}$ is the domain of the input variable. Here, f may or may not have a known analytical representation, and thus the computer model describing the complex system under study may itself also be very complex. Therefore, understanding the computer model can be as challenging of a task as understanding the original physical system it represents. Although possibly stochastic, in this dissertation we focus on the case of deterministic computer models where running the computer model for the same input yields the same output always. Much like the design of controlled experiments, one can also construct designed experiments in order to better understand computer models. These experiments, or *computer experiments*, consist of running the computer model at different input configurations in order to build up an understanding of the possible outcomes of the computer model.

For example, there is an area in Billings, Montana, near the Yellowstone river, called the Lockwood site, where two plumes of contaminated groundwater have developed in the area due to industrial practices. The two plumes (plume A and B) are slowly migrating towards the Yellowstone river and of primary concern is keeping the chlorinated contaminants from leaking into the river (Figure 1.1). In order to prevent the flow of the contaminants into the river, a pump-and-treat remediation is proposed, in which wells are placed to pump out contaminated water, purify it, and then return the treated water, at six locations (A_1, A_2, B_1, B_2, B_3 and B_4).

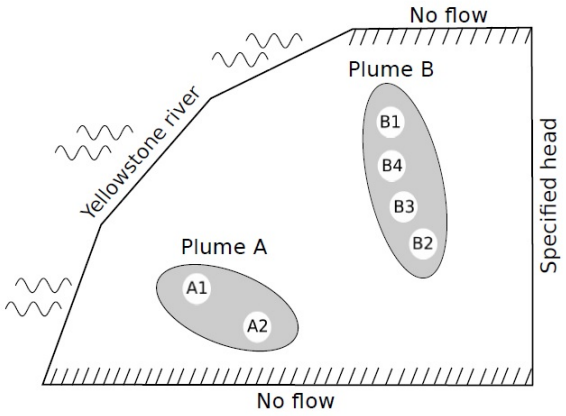


Figure 1.1: Lockwood site and its contaminant plumes located near Billings, Montana. A_1, A_2, B_1, B_2, B_3 and B_4 are the locations of the six proposed remediation wells (Lindberg & Lee, 2015a).

Of course the naive scientist could go out to the Lockwood site, turn on the pumps, adjust the respective pumping rates, and wait to see if any of the contaminants leak into the Yellowstone river. However, this would be a terrible experiment since, as one could imagine, the scientist either would be contaminating or not contaminating the river and thus would have to learn about the complex physical system through their mistakes, and, worst of all, at the expense of the environment. Instead, the savvy scientist could construct a computer model of this physical process in order to understand the dynamics of this complex system. Here the scientist could run a computer experiment, where the inputs to the computer model would be the associated pumping rates for the six pumps, and the output(s) could be the cost of running the pumps or the status of whether or not the contaminants leaked into the Yellowstone river, or both.

A common feature of high fidelity computer models is that they tend to be compu-

tationally expensive. Computer models can be extremely complex mathematical programs, and the evaluation of different input configurations may take, minutes, hours or even days to calculate a single output. This computational expense makes it prohibitive to try and run the computer experiment at every possible input configuration in order to understand the system. Thus, a common theme of computer experiments is to try to find an appropriate "cheap-to-compute" model, or *surrogate model*, that resembles the true computer model very closely but is much faster to run (Figure 1.2).

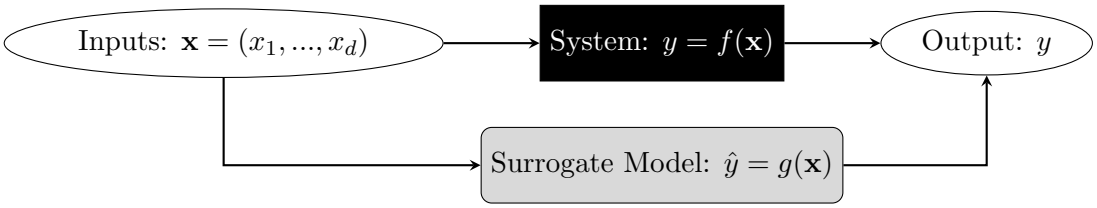


Figure 1.2: The framework for surrogate modeling for computer experiments.

Traditionally, the canonical choice for modeling computer experiments has been the Gaussian process (GP) (Sacks et al., 1989; Santner et al., 2003). Gaussian processes are distributions over functions such that the joint distribution at any finite set of points is a multivariate Gaussian distribution. Gaussian processes make for convenient surrogate models because they are conceptually straightforward (a form of nonparametric regression). Gaussian processes have a number of desirable properties such as being flexible, being able to closely approximate most functions, and often being much cheaper/faster to evaluate than the actual computer model. More importantly, using Gaussian processes for surrogate modeling allows for uncertainty quantification of computer outputs at untried (or unobserved) inputs, and also provides a statistical framework for efficient design and analysis

of computer experiments (Santner et al., 2003). It is for these reasons that all of the work done in this dissertation will be built upon the foundations of Gaussian process modeling.

1.1 Objectives and Contributions

This dissertation consists of four chapters spanning topics in statistics, stochastic modeling, and optimization. As a natural progression, each chapter builds upon the previous chapter's work, with the overarching goal of building an efficient framework for constrained optimization of computer experiments.

Consider again the pump-and-treat hydrology problem at the Lockwood site (Figure 1.1). This problem can be thought of as the entire motivation for this dissertation. There is a physical process that needs to be studied, but environmental standards do not allow for physical experiments. Thus a computer model is built, and computer experiments can be conducted to better understand the physical system. Now, recall that the inputs to the Lockwood computer model are the pumping rates and the outputs are the cost of running the pumps and the contamination status. A natural question that arises is can we minimize the cost of running the pumps while also containing the contaminants from leaking into the river? Thus, using the computer model can be seen as an exercise in constrained optimization. Here, the objective function we wish to minimize describes the cost of running the pumps, subject to the constraint that we do not allow contaminants into the Yellowstone river. But how can we find the optimal configuration of pumping rates that solves this constrained optimization problem? The solution to this question will be found through computer modeling and is the overarching goal of this dissertation.

In this dissertation, we introduce a new methodology that combines statistical modeling and filter methods (Chapter 3) for solving nonlinear optimization problems of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{c}(\mathbf{x}) \leq 0 \\ & \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{1.2}$$

where $\mathcal{X} \subset \mathbb{R}^d$ is a known, bounded region such that $f : \mathcal{X} \to \mathbb{R}$ denotes a scalar-valued objective function and $\mathbf{c} : \mathcal{X} \to \mathbb{R}^m$ denotes a vector of m constraint functions. We further complicate the problem by assuming that (1.2) is an expensive black box optimization problem where both the objective, f, and constraint, $\mathbf{c}$, require black box simulation (i.e., running a computer model revealing little about the functional form of the objective and constraints). Furthermore, we focus on the *derivative-free* situation where no information about the derivatives of the objective and constraint functions is available (Conn et al., 2009).

Traditionally, methods for solving (1.2) have been based on Newton's methods and are iterative (Fletcher et al., 2006). Typically one starts with an initial estimate $\mathbf{x}_k$ that is reasonably close to the true solution, then a linear or quadratic approximation of (1.2) is locally solved for a new $\mathbf{x}_{k+1}$. The method iterates successively in this fashion until convergence is achieved. Local convergence is almost guaranteed with such algorithms, unfortunately, these algorithms may not be convergent from an arbitrary starting point $\mathbf{x}_k$. To remedy this downfall, penalty functions are often used to ensure global convergence properties, meaning convergence to a local optimum for every possible starting point $\mathbf{x}_k$. For

example, a linear combination of the objective function and a constraint violation function, say $h(\mathbf{x}) = ||\max\{\mathbf{0}, \mathbf{c}(\mathbf{x})\}||$ for some norm, can define the penalty function

$$p(\mathbf{x}; \pi) = f(\mathbf{x}) + \pi h(\mathbf{x}), \tag{1.3}$$

where $\pi > 0$ is the penalty parameter. The new penalized objective function (1.3) transforms the constrained problem in (1.2) into an unconstrained optimization problem. Some of the main drawbacks to the unconstrained optimization of (1.3) is that the rate of convergence is dependent upon a suitable choice of π, and that the unconstrained optimization cannot distinguish infeasible points (where $h(\mathbf{x}) > 0$) from feasible points. Furthermore, from a statistical point of view, there is a substantial amount of valuable information lost in compressing the constraint values $c(\mathbf{x})$ into a single scalar $h(\mathbf{x})$.

Unfortunately, traditional methods based on Newton's method that leverage information about the derivatives of the objective and/or constraint functions are of no use to us in the derivative-free scenario. However, a considerable amount of effort has been put forth by the optimization community for solving constrained problems like (1.2) when no derivative information is available (Conn et al., 2009). The class of local derivative-free optimization algorithms can be loosely partitioned into two classes of derivative free methodologies: direct-search and model-based methods. Direct-search methods are methods that sample the objective function at a finite number of points at each iteration and decide which actions to take next solely based on those function values and without any explicit or implicit derivative approximation or model building (Conn et al., 2009). For example, the mesh adaptive direct search algorithm (Audet & Dennis, 2006) generates trial points on a spatial discretization, called a mesh, and then decides which actions to take

based on the function values evaluated on the mesh. On the other hand, model-based methods are more akin to the traditional response surface modeling of statistics (Box & Draper, 1987). For example, model-based methods based on nonlinear kernels such as radial basis functions in trust regions (Wild et al., 2008) or based on local polynomials (Conn et al., 2009) are directly relatable to Gaussian processes.

Additionally, model-based methods have become increasingly popular in the statistical literature for solving constrained optimization problems. Primarily, model-based methods have utilized Gaussian processes as a means of building surrogate models for the black box simulators (computer models) that are being optimized and have relied upon heuristics for handling constraints. For example, Wilson et al. (2001) proposed exploring the Pareto frontier using surrogate approximations in order to solve a biobjective optimization problem, and following the same suit, Parr et al. (2012) took it a step further by incorporating constrained expected improvement into building the Pareto frontier. Following the earlier works of Jones et al. (1998), Svenson & Santner (2012) extended the idea of expected improvement to the multiobjective optimization case by exploring Gaussian process surrogate modeling and Pareto frontiers as well. Sasensa et al. (2002) used surrogate models based on Gaussian processes and handled the constraints by transforming the problem into an unconstrained problem by use of a penalty function. Similarly, Gramacy et al. (2015) used a penalty function approach based on augmented Lagrangians to handle the constraints and searched the objective space using particle learning Gaussian process methods with expected improvement techniques. Lee et al. (2011) and Lindberg & Lee (2015b) solved black box optimization problems using constrained expected improvement;

the former made use of calculations of the probability of satisfying the constraint and the latter used asymmetric entropy as a guiding measure of satisfying the constraints.

Although the aforementioned authors have made significant contributions to solving problems such as (1.2), there are still avenues of research for improvement, some of which we aim to address in this dissertation. Similarly to the work described above, we advocate the use of model-based methods to solve constrained optimization problems of the same variety. In particular, in Chapter 3, we rely on an established nonlinear optimization technique, known as a filter method, and combine the method with statistical surrogate modeling. Combining these two techniques in an innovative fashion results in an efficient algorithm to solve problems like (1.2). Like many of the prior model-based methods, we utilize Gaussian processes to develop our surrogate models. The current methodology in surrogate modeling makes the simplification of modeling the objective and constraint function independently. While this approach has proven fruitful, it may be advantageous to model them jointly; in Chapter 2, we employ Gaussian processes to develop this methodology. As a result of this idea, we explore the problem of developing a multivariate modeling framework for fast sequential inference for computer experiments. The fact that black box models are often costly to evaluate calls for the development of an efficient methodology for these problems. Gramacy & Polson (2011) developed a methodology based on sequential Monte Carlo methods to tackle this problem in the univariate case. In this setting we propose to extend their work to the multivariate setting.

The remainder of this dissertation is organized as follows. We first review Gaussian process modeling and the necessary theoretical foundations for understanding the rest of

this dissertation. Chapter 2 develops a novel methodology for modeling correlated outputs of different types (e.g., continuous, binary, categorical). For example, in the motivating Lockwood problem, one could think of the cost of running the pumps as a continuous output and the contamination status as a binary output and that these outputs are correlated with one another. It is problems like these that motivate the need for a model that can handle joint modeling of correlated outputs of different types, and so, we introduce a new stochastic model capable of joint inference and prediction for correlated outputs of different types. We build the modeling framework with expensive computer models in mind, and thus we develop a technique for fast sequential inference for this model. Through the application of our methodology on both synthetic and real-world computer models we show that joint modeling of outputs leads to much better results than independent modeling.

Chapter 3 combines filter methods with the statistical models of Chapter 2 for solving constrained optimization problems in computer experiments. We develop two novel metrics for guiding the sequential search for a global minimum of (1.2) and validate our new methodology on a suite of synthetic test problems as well as the real-world pump and treat hydrology problem. Furthermore, we compare our methodology to existing methods and show our method to be superior to the comparators. Chapter 4 extends the methodology of Chapter 3 to accommodate higher fidelity solutions that outperform the current two dimensional filter method for constrained optimization. Chapter 5 establishes arguments for proving global convergence of the methods in Chapter 3. Lastly, Chapter 6 concludes with some discussions and conclusions about the work of this dissertation, as well as future avenues for further research.

1.2 Background

The foundations of this dissertation are based on stochastic modeling, and in particular, on the use of Gaussian processes (GPs) as efficient surrogate models. Readers familiar with the concepts of modeling, inference, and prediction for Gaussian processes are encouraged to skip ahead to Chapter 2.

1.2.1 Gaussian Process Models

Traditionally, the canonical choice for modeling computer experiments has been the Gaussian process (Sacks et al., 1989; Santner et al., 2003). Gaussian processes are commonly used in probabilistic modeling when priors over functions, without reference to an underlying parametric representation, are needed. Gaussian processes are distributions over functions such that the joint distribution at any finite set of points is a multivariate Gaussian distribution.

More rigorously, for any index set $\mathcal{X}$, the real-valued stochastic process $\{Y(\mathbf{x}), \mathbf{x} \in \mathcal{X}\}$, is a Gaussian process if all the finite-dimensional distributions, say, $F(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, are multivariate normal distributions, for any choice of $n \geq 1$ and $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$. The fundamental characterization of a Gaussian process thus requires a specification of a mean function, $m(\mathbf{x}) = \mathbb{E}(Y(\mathbf{x}))$, $\forall \mathbf{x} \in \mathcal{X}$, and a covariance function, $C(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(Y(\mathbf{x}), Y(\mathbf{x}'))$, $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$. There are no restrictions on the functional form of $m(\mathbf{x})$ other than that $\mathbb{E}(Y(\mathbf{x}))$ exist and is finite for all $\mathbf{x} \in \mathcal{X}$. However, in order for a valid covariance function to exist, $C(\mathbf{x}, \mathbf{x}')$ must satisfy

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j C(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for all integer $n \geq 1$, all $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and all real numbers $c_1, \ldots, c_n$. Any covariance function satisfying the above property is said to be positive semidefinite. Covariance matrices need only be positive semidefinite, however, for all of the calculations that follow in this chapter, and subsequent chapters, we assume the form of the covariance matrix to be positive definite. This positive definite assumption is typically imposed by the specification of the form of the kernel of the covariance matrix, and is critical in allowing us to be able to calculate and invert inverse matrices that will arise in many calculations.

Stationarity and Isotropy

Although not a necessary condition, a simplifying condition is to assume stationarity of the Gaussian process. Assuming stationarity is akin to assuming that the probabilistic structure of the Gaussian process, in some sense, looks similar in different parts of $\mathcal{X}$ (Stein, 1999). More formally, the Gaussian process is said to be *strongly stationary* provided that for any finite collection of $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{X}$, the joint distributions of $(Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_n))$ and $(Y(\mathbf{x}_1+\mathbf{u}), .., Y(\mathbf{x}_n+\mathbf{u}))$ are the same. Thus, strong stationarity requires that arbitrary translations do not change the distribution of the process. The strongly stationary condition can be weakened by requiring that only the mean and covariance functions be invariant under translations, i.e., $m(\mathbf{x}) = m, \forall \mathbf{x} \in X$ and $C(\mathbf{x}, \mathbf{x}' + \mathbf{u}) = C(\mathbf{u})$. This type of stationarity is usually referred to as being *weakly stationary* (Cressie, 1993). Conveniently, in the case of the Gaussian process, weak stationarity will also imply strong stationarity. Another simplifying condition is to assume that the Gaussian process is invariant under rotations, a property called isotropy. A stationary Gaussian process can be shown to be isotropic if the covariance function depends on distance alone, i.e., $C(\mathbf{x}, \mathbf{x}') = C(\tau)$

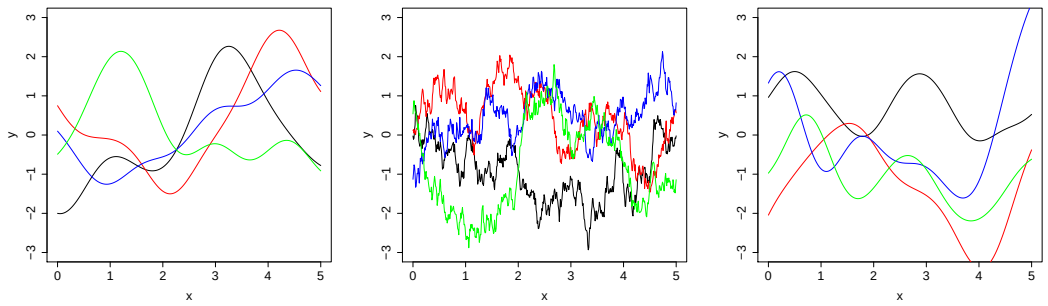
where $\tau = ||\mathbf{x} - \mathbf{x}'||$ is Euclidean distance. This is a very strong condition on the radial symmetry of the covariance function.

Correlation Function

In most applications of Gaussian processes, it is often more convenient to separately model the process variance $\sigma^2(\mathbf{x}) = C(\mathbf{x}, \mathbf{x})$ and the process correlation function $\rho(\cdot, \cdot)$. The correlation function is defined as

$$\rho(\mathbf{x}, \mathbf{x}') = \frac{C(\mathbf{x}, \mathbf{x}')}{\sqrt{C(\mathbf{x}, \mathbf{x})C(\mathbf{x}', \mathbf{x}')}} \tag{1.4}$$

for $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. The correlation function plays a key role in determining the dependence structure and smoothness of the Gaussian process (Figure 1.3), and thus, much care is needed when specifying it. Many families of parametric forms exist for the correlation



(a) Squared Exponential (b) Powered Exponential (c) Matérn

Figure 1.3: Realizations of Gaussian processes under three different correlation functions. Different correlation functions can produce drastically different processes.

function, such as the powered exponential or Matérn for example, and thus specifying the parametric form of the correlation function is not a trivial task (Abrahamsen, 1997).

However, restricting to the space of all stationary and isotropic Gaussian processes, the correlation function simplifies to

$$\rho(\tau) = C(\tau)/\sigma^2 \tag{1.5}$$

where $\sigma^2 = C(0)$ for all $\mathbf{x} \in \mathcal{X}$. More intuitively now, requiring that a Gaussian process be stationary and isotropic is equivalent to requiring that the correlation between any $\mathbf{x}$ and $\mathbf{x}'$ be measured identically throughout $\mathcal{X}$ and that the correlation should depend only on a function of the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$.

The Nugget

The nugget was first introduced by Matheron (1962) in the geostatistics literature. The nugget is considered as random noise and typically represents measurement error or short scale variability (Cressie (1993), Diggle & Ribeiro Jr. (2007)) in the Gaussian process. Thus, the nugget provides a mechanism for introducing measurement error into the Gaussian process. The nugget, termed jitter in the machine learning literature (Neal, 1997), also serves the practical purpose of preventing the correlation matrix from becoming numerically singular. Lastly, inclusion of a nugget parameter in the Gaussian process can also lead to models with better statistical properties, such as predictive accuracy and coverage (Gramacy & Lee, 2012).

More formally, considering the covariance function $C(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2\rho(\mathbf{x}_j, \mathbf{x}_k)$, the nugget term η is introduced into the model by extending the new covariance function to be

$$C(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2\rho^*(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2[\rho(\mathbf{x}_j, \mathbf{x}_k) + \eta\delta_{j,k}] \tag{1.6}$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta function and $\eta > 0$. For example, considering the squared exponential covariance function with nugget leads to a covariance function of the following form

$$C(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2 \rho^*(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2 \left[\exp\left(-\sum_{i=1}^{p} \frac{|x_{ij} - x_{ik}|^2}{\phi_i} \right) + \eta \delta_{j,k} \right]. \tag{1.7}$$

As an illustrating example, consider the deterministic function $f(x) = e^{-1.4x} \cos\left(\frac{7\pi x}{2}\right)$ where for the nine inputs $\{x_1, \ldots, x_9\} \in (0, 1)$ we have data of the nine deterministic outputs $f(x_1), \ldots, f(x_9)$. Modeling the data using a Gaussian process (as will be shown in subsequent sections) with and without a nugget leads to the following posterior predictive inference shown in Figure 1.4. The main feature to take away from Figure 1.4 is that the

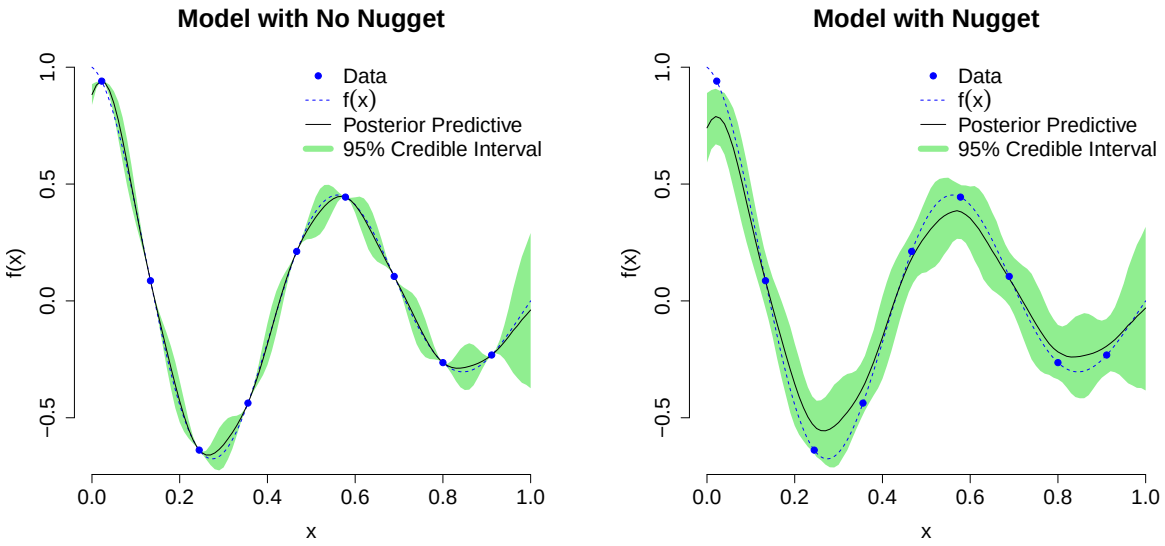


Figure 1.4: Models with (right) and without (left) the nugget term. Models without a nugget lead to interpolation while those with a nugget result in smoothing.

posterior predictive distribution under the model with no nugget will interpolate the data while the model with no nugget will not. In fact, under the model with no nugget, the

prediction variance at observed points is exactly zero and increases away from zero as the distance from the predicted points increases from the observed points. On the other hand, the model with nugget displays the same trend as the model with no nugget except that the prediction variance will be equal to $\sigma^2\eta$ at observed points rather than zero.

Model Building

A popular approach, especially in the computer modeling literature, is to allow the mean function of the stochastic process $Y(\mathbf{x})$ to be a function of $\mathbf{x}$ in the usual regression framework while assuming the process residual variation follows a stationary Gaussian process. Modeling the stochastic process $Y(\mathbf{x})$ this way leads to models of the form

$$Y(\mathbf{x}) = \sum_{j=1}^{p} f_j(\mathbf{x})\beta_j + Z(\mathbf{x}) \tag{1.8}$$

where $f_1(\cdot), \ldots, f_p(\cdot)$ are known regression functions, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is a vector of unknown regression coefficients, and $Z(\cdot)$ is a zero mean stationary Gaussian process over $\mathcal{X}$ (Santner et al., 2003). Restricting attention to the class of stationary isotropic processes greatly reduces the possible model space, however, the stationarity assumption may be too restrictive. In particular, when the underlying true stochastic process $Y(\mathbf{x})$ is indeed nonstationary, other more flexible approaches may be needed to model $Y(\mathbf{x})$. Several approaches exist in order to model nonstationarities in $Y(\mathbf{x})$. For example, Higdon et al. (1999) used a process convolution approach convolving a common latent white noise Gaussian process with a smoothing kernel to arrive at a nonstationary process. Likewise, Gramacy & Lee (2008) developed treed Gaussian processes to model nonstationary processes by modeling smaller partitions of the space with stationary processes. Of course, models of the form

(1.8) can also be nonstationary.

Estimation

The parameters in the model (1.8) can be estimated through both frequentist and Bayesian methods, here we focus on the latter. From a Bayesian perspective, the standard linear model priors (Gelman et al., 2013) can be placed on $\boldsymbol{\beta}$ and σ^2. Placing a diffuse prior for β, $p(\boldsymbol{\beta}) \propto 1$, and a conjugate inverse-gamma prior, for σ^2, allows Gibbs samples to be obtained for these parameters. Priors must also be placed on the parameters of the correlation matrix $\mathbf{R}$, unfortunately for these parameters, Gibbs samples will not be available.

Now, for clarification, let $\mathbf{Y} = (Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_n))^T$ where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$ and $\mathbf{R}(\boldsymbol{\psi})$ be a correlation matrix based on correlation parameters $\boldsymbol{\psi} = (\boldsymbol{\phi}, \eta)$ where $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_p)$ and nugget η. Also, define $\mathbf{F}$ as the traditional $n \times p$ linear model design matrix corresponding to (1.8). Placing the following priors

$$p(\boldsymbol{\beta}) \propto 1 \quad \text{and} \quad \sigma^2 \sim \text{IG}(a/2, b/2)$$

leads to the following posterior distribution

$$\begin{aligned} p(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\psi}|\mathbf{Y}) \propto |\sigma^2 \mathbf{R}(\boldsymbol{\psi})|^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{Y} - \mathbf{F}\boldsymbol{\beta})^T \mathbf{R}(\boldsymbol{\psi})^{-1}(\mathbf{Y} - \mathbf{F}\boldsymbol{\beta})\right) \\ \times (\sigma^2)^{-(a/2+1)} \exp\left(-\frac{b}{2\sigma^2}\right) p(\boldsymbol{\phi}) p(\eta). \end{aligned} \tag{1.9}$$

Furthermore, accounting for normalizing constants and integrating over $\boldsymbol{\beta}$ and σ^2 in (1.9),

the marginal posterior distribution for $\boldsymbol{\psi}$ is available in closed form:

$$p(\boldsymbol{\psi}|\mathbf{Y}) = \left(\frac{|\mathbf{F}^T\mathbf{R}(\boldsymbol{\psi})\mathbf{F}|}{|\mathbf{R}(\boldsymbol{\psi})|}\right)^{1/2} \frac{(b/2)^{a/2}\Gamma((n-p+a)/2)}{(2\pi)^{(n-p)/2}\Gamma(a/2)} \left(\frac{b+S(\boldsymbol{\psi})}{2}\right)^{-(n-p+a)/2} p(\boldsymbol{\psi}), \tag{1.10}$$

where

$$S(\boldsymbol{\psi}) = \mathbf{Y}^T\mathbf{R}(\boldsymbol{\psi})^{-1}\mathbf{Y} - \hat{\boldsymbol{\beta}}\mathbf{V}_\beta\hat{\boldsymbol{\beta}}, \quad \hat{\boldsymbol{\beta}} = \mathbf{V}_\beta(\mathbf{F}^T\mathbf{R}(\boldsymbol{\psi})^{-1}\mathbf{Y}), \quad \mathbf{V}_\beta = (\mathbf{F}^T\mathbf{R}(\boldsymbol{\psi})^{-1}\mathbf{F})^{-1},$$

and $\Gamma(\cdot)$ is the standard gamma function defined as $\Gamma(t) = \int_0^\infty x^{t-1}e^{-x}dx$. A caveat is that proper priors must be placed on the correlation parameters $\boldsymbol{\phi}$ and nugget η in order to ensure a proper marginal posterior distribution. Working with the integrated likelihood, Berger et al. (2000) showed that a reference prior for $\boldsymbol{\phi}$ could be derived. In either case, Markov Chain Monte Carlo methods will need to be used in order to obtain posterior samples of $\boldsymbol{\phi}$ and η.

Prediction

Spatial prediction under the Gaussian process, also known as kriging (Matheron, 1963) in the geostatistics literature in honor of the pioneering geostatistican D. G. Krige (Krige, 1951), is a direct application of standard multivariate Normal conditioning rules (Omre (1987), Omre et al. (1989), Omre & Halvorsen (1989), Hjort & Omre (1994)). Working with the marginal posterior (1.10), the predictive distribution at new data points $\tilde{\mathbf{Y}} = (Y(\tilde{\mathbf{x}}_1), \ldots, Y(\tilde{\mathbf{x}}_m))$, conditional on previous data $\mathbf{Y}$, is a multivariate t-distribution with $\nu = n - p - 1$ degrees of freedom, mean

$$\mathbb{E}(\tilde{\mathbf{Y}}|\mathbf{Y}, \boldsymbol{\psi}) = \mathbf{f}(\tilde{\mathbf{X}})^T\hat{\boldsymbol{\beta}} + \mathbf{r}_{\mathbf{X},\tilde{\mathbf{X}}}^T\mathbf{r}_{\mathbf{X},\mathbf{X}}^{-1}(\mathbf{Y} - \mathbf{F}\hat{\boldsymbol{\beta}}), \tag{1.11}$$

and scale matrix

$$\mathbb{V}(\tilde{\mathbf{Y}}|\mathbf{Y},\boldsymbol{\psi}) = \frac{[b + S(\boldsymbol{\psi})]\left(\mathbf{r}_{\tilde{\mathbf{X}},\tilde{\mathbf{X}}} - \mathbf{r}^{T}_{\mathbf{X},\tilde{\mathbf{X}}}\mathbf{r}^{-1}_{\mathbf{X},\mathbf{X}}\mathbf{r}_{\mathbf{X},\tilde{\mathbf{X}}}\right)}{a+\nu} \tag{1.12}$$

where $\mathbf{r}_{\mathbf{X},\tilde{\mathbf{X}}} = \mathbf{R}(\mathbf{X},\tilde{\mathbf{X}};\boldsymbol{\psi})$ is the $m \times n$ correlation matrix and $\mathbf{f}(\tilde{\mathbf{X}}) = (\mathbf{f}(\tilde{\mathbf{x}}_1),\ldots,\mathbf{f}(\tilde{\mathbf{x}}_m))$. Interestingly, prediction under (1.11) and (1.12) leads to interpolation at prediction points $\tilde{\mathbf{X}} = \mathbf{X}$ when $\eta = 0$.

Chapter 2

Multivariate Stochastic Process Models for Correlated Responses of Mixed Type

The problems of regression and classification are both well-studied individually, but there has been limited work on the problem of combined regression and classification when these outputs are correlated, particularly in the nonparametric setting. Only recently has the literature moved beyond traditional parametric assumptions. We are motivated by the problem of constrained optimization where both the objective function and the constraints are unknown and potentially expensive to evaluate. Thus we seek an efficient statistical model to serve as a fast approximation to the true objective function and constraints, which in our applications are computer simulation experiments. In the case that the simulator only returns whether a constraint is satisfied, and not any measure of distance to satisfaction, we

need to jointly model a continuous objective function and one or more binary constraints. Constrained optimization is typically difficult because at least one of the constraints operates in opposition to the objective function, i.e., they are negatively correlated. We propose here a nonparametric model to jointly model continuous and binary outputs, and the framework is flexible enough to include a wide variety of other types of outputs. Our approach builds upon the standard computer emulation approach in the literature of using Gaussian process (GP) models (Santner et al., 2003).

Joint modeling of outputs of different types, also called multiway or mixed type responses, can be a difficult task. When the outputs are known or suspected to be correlated, it is common practice to use latent processes to induce correlation between them (Sammel et al., 1997; Moustaki & Knott, 2000). However, most of these latent methods rely on either simple linear models or restrictive parametric assumptions. Recent papers have started to utilize more robust nonparametric models, such as infinite mixtures of linear models (Zhe et al., 2015). Typically in computer simulation experiments, constrained optimization is very challenging because the outputs of the simulators arise from highly nonlinear functions. This lack of linearity is what makes nonparametric methods so desirable.

An active area of research in machine learning, multi-task learning builds predictive models based on the learning of multiple tasks, in our case learning mixed type outputs, at the same time. The performance of multi-task learning methods is highly dependent upon the sharing of information, or induced correlation, across each task. A nonparametric extension, Yu et al. (2005), used multi-task GPs for sharing information across multiple tasks of the same type. However, the growing literature on multi-task GPs (Bonilla et al.,

2008; Hayashi et al., 2012) does not seem to suggest that the model can facilitate the modeling of outputs of mixed types. Similar to the approach we will take in this chapter, Liu et al. (2013) makes an attempt to address this problem, however, a major limitation of that work is that they only consider the case of two correlated outputs.

We propose here a flexible nonparametric model capable of handling $p \geq 2$ correlated outputs to address this gap. The Gaussian process framework has proved to be an effective tool for modeling both regression and classification (Neal, 1999). Multivariate regression GPs (Wackernagel, 2003) provide a basis for modeling correlated outputs. We build upon these ideas to create a new GP-based model for correlated outputs of mixed type, where each output uses a transformation function to map back to the regression setting. Chan (2013) explored this same idea, but was limited by only considering correlated outputs of the same type and by assuming that the likelihood function takes the generic form of the multivariate exponential family distribution. We bypass these limitations by allowing for a more general likelihood function and create a fully generalized family of models that utilize standard link functions, including but not limited to the identity for regression and the logistic for classification, but more general links also fit into our framework. Similar approaches, Xu et al. (2012) and Zhe et al. (2013), utilize latent tensor-valued Gaussian process models to model mixed type outputs from a Bayesian point of view, however, both works employed only variational techniques for inference. Another key innovation of our work is fully Bayesian inference, through particle learning, whereas Liu et al. (2013), Xu et al. (2012), Chan (2013) and Zhe et al. (2013) provide for only approximate Bayesian inference.

We take a fully Bayesian approach, which can require significant computational effort. To address this concern, we build upon recent work on sequential Monte Carlo methods for GPs (Gramacy & Polson, 2011; Montagna & Tokdar, 2013). Particle learning allows for fast inference, and also allows for sequential inference when the data arrive over time, as is the case in computer experiments. As a special case, we believe this is the first implementation of particle learning for a multivariate regression GP.

The remainder of this chapter is organized as follows. In the next section, we review the separable multivariate Gaussian process and set up the necessary modeling framework for the rest of the chapter. Section 2.2 introduces our novel joint regression and classification model. We review the sequential Monte Carlo technique, particle learning, in Section 2.3, and explain how fast sequential inference can be conducted on our joint regression and classification model with an extension to a similar stochastic process model. Section 2.4 demonstrates the applicability of the models presented with a number of illustrative examples and comparisons with previous work. Section 2.5 concludes with some discussion.

2.1 Multivariate Gaussian Process

We consider a stochastic process returning a p-dimensional output $\mathbf{y} \in \mathbb{R}^p$ for a given d-dimensional input $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$. We think of the stochastic process as a function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^p$ for some (possibly high dimensional) input space $\mathcal{X}$. Similar to Conti & O'Hagan (2010) and Fricker et al. (2013), from a Bayesian perspective, we regard $\mathbf{f}(\cdot)$ as an unknown function and represent the uncertainty surrounding it through the use of the

p-dimensional multivariate Gaussian process

$$\mathbf{f}(\cdot) \sim \mathrm{GP}_p\left(\boldsymbol{\mu}(\cdot), \mathbf{C}(\cdot,\cdot)\right), \tag{2.1}$$

where $\boldsymbol{\mu}$ is a mean function and $\mathbf{C}$ is a covariance function. The existence of the multivariate Gaussian process depends on the specification of a valid cross-covariance function $\mathbf{C}(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}'))$ for $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ (Wackernagel, 2003). Generating valid, as well as tractable, cross-covariance functions is not a simple task. Many methods have been proposed, such as: separable models (Mardia & Goodall, 1993; Banerjee & Gelfand, 2002), convolution of covariance functions (Gaspari & Cohn, 1999; Majumdar & Gelfand, 2007), and the linear model of coregionalization (Goulard & Voltz, 1992; Wackernagel, 2003; Gelfand et al., 2004). The approaches in this chapter work for more general covariance structures beyond the separable model; for example, we have tried them with the linear model of coregionalization, but we focus on the separable model as we find it works well in practical applications, providing sufficient flexibility without too much additional computational expense. In the section that follows, we briefly discuss separable models.

2.1.1 Separable Model

One of the simplest ways of achieving a valid cross-covariance function is to take a valid univariate correlation function $\rho(\mathbf{x}, \mathbf{x}')$ and a valid $p \times p$ positive semidefinite matrix $\mathbf{T}$ so that

$$\mathbf{C}(\mathbf{x}, \mathbf{x}') = \rho(\mathbf{x}, \mathbf{x}')\mathbf{T}. \tag{2.2}$$

The cross covariance function in (2.2) is said to be a separable model. Letting $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be the collection of all inputs observed so far in $\mathcal{X}$, the resulting $np \times np$

covariance matrix for $\mathbf{Y}^T = (\mathbf{y}_1^T, \ldots, \mathbf{y}_n^T)$, where $\mathbf{y}_i^T = (f_i(\mathbf{x}_1), \ldots, f_i(\mathbf{x}_n))$, is

$$\mathbf{C}(\mathbf{X}, \mathbf{X}) = \mathbf{R} \otimes \mathbf{T} = \begin{pmatrix} \rho(\mathbf{x}_1, \mathbf{x}_1)\mathbf{T} & \cdots & \rho(\mathbf{x}_1, \mathbf{x}_n)\mathbf{T} \\ \vdots & \vdots & \vdots \\ \rho(\mathbf{x}_n, \mathbf{x}_1)\mathbf{T} & \cdots & \rho(\mathbf{x}_n, \mathbf{x}_n)\mathbf{T} \end{pmatrix}, \tag{2.3}$$

where we denote $\mathbf{\Sigma} = \mathbf{C}(\mathbf{X}, \mathbf{X})$ and $\mathbf{R}$ is the $n \times n$ correlation matrix with $\mathbf{R}_{ij} = \rho(\mathbf{x}_i, \mathbf{x}_j)$. Clearly, $\mathbf{\Sigma}$ is positive semidefinite since $\mathbf{R}$ and $\mathbf{T}$ are. There are some clear advantages to using a separable model, for instance, $|\mathbf{\Sigma}| = |\mathbf{R}|^p |\mathbf{T}|^n$ and $\mathbf{\Sigma}^{-1} = \mathbf{R}^{-1} \otimes \mathbf{T}^{-1}$ which means that working with $\mathbf{\Sigma}$ requires working with a $p \times p$ and $n \times n$ matrix instead of a $np \times np$ matrix. Additionally, from a Bayesian perspective, using a separable model allows for placing a conjugate prior on $\mathbf{\Sigma}$ (Banerjee et al., 2004). Conti & O'Hagan (2010) placed an improper inverse-Wishart prior on $\mathbf{\Sigma}$, which leads to a proper inverse-Wishart posterior for $\mathbf{\Sigma}$ and allows for $\mathbf{\Sigma}$ to be analytically integrated out of the posterior predictive process.

2.1.2 Model Building and Prediction

We treat the unknown function $\mathbf{f}(\cdot)$ as a multivariate stochastic process and model $\mathbf{f}(\cdot)$ as

$$\begin{aligned} \mathbf{f}(\cdot) &= \boldsymbol{\mu}(\cdot) + \boldsymbol{\omega}(\cdot) \\ \boldsymbol{\mu}(\cdot) &= (\mathbf{I}_p \otimes \mathbf{H}) \operatorname{vec}(\mathbf{B}) \\ \boldsymbol{\omega}(\cdot) | \mathbf{T}, \boldsymbol{\phi}, \eta &\sim \mathrm{GP}_p(\mathbf{0}, \mathbf{C}(\cdot, \cdot)), \end{aligned} \tag{2.4}$$

where $\operatorname{vec}(\cdot)$ is the "vec" operator that stacks the columns of its matrix argument from left to right into a single vector. Here, $\mathbf{I}_p$ denotes the $p \times p$ identity matrix, $\mathbf{H}^T = [\mathbf{h}(\mathbf{x}_1) \cdots \mathbf{h}(\mathbf{x}_n)] \in \mathbb{R}^{q \times n}$ is a matrix of regression functions, $\mathbf{B} = [\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_p] \in \mathbb{R}^{q \times p}$ is a

matrix of regression coefficients and the matrix valued covariance function, $\mathbf{C}(\cdot,\cdot)$, depends on covariance parameters $\mathbf{T}$ and $\boldsymbol{\psi} = \{\boldsymbol{\phi}, \eta\}$ where $\boldsymbol{\psi}$ represents parameters governing the correlation function ρ, which we take in this chapter to be the length-scale parameter $\boldsymbol{\phi}$ and nugget parameter η. The length-scale parameter, $\boldsymbol{\phi}$, plays the role of determining how fast the spatial correlation decays throughout the input space, while the nugget, η, is considered as random noise and typically represents measurement error or short scale variability (Cressie, 1993; Diggle & Ribeiro Jr., 2007) in the Gaussian process. Thus, the nugget provides a mechanism for introducing measurement error into the Gaussian process. Assuming separability of the covariance function in (2.4) allows us to write the likelihood for the data as the following matrix Normal distribution

$$\mathbf{D}|\mathbf{B}, \mathbf{T}, \boldsymbol{\phi}, \eta \sim \mathrm{N}_{n,p}\left(\mathbf{HB}, \mathbf{R}, \mathbf{T}\right), \tag{2.5}$$

(Rowe, 2003) where we arrange the data vector $\mathbf{Y}$ into the output matrix $\mathbf{D}$ such that $\mathrm{vec}(\mathbf{D}) = \mathbf{Y}$. From a Bayesian point of view, all that is left is to place prior distributions on the unknown parameters of the model and to update the posterior distribution of the unknown parameters via Bayes' theorem. Lacking strong prior information for $\mathbf{B}$ and $\mathbf{T}$, we follow Conti & O'Hagan (2010) and place the following joint improper prior for $\mathbf{B}$ and $\mathbf{T}$

$$p(\mathbf{B}, \mathbf{T}|\boldsymbol{\psi}) \propto |\mathbf{T}|^{-(p+1)/2}. \tag{2.6}$$

Specifying an arbitrary choice of prior $p(\boldsymbol{\psi})$ for $\boldsymbol{\psi}$ we obtain the posterior distribution

$$\begin{aligned} p(\mathbf{B}, \mathbf{T}, \boldsymbol{\psi}|\mathbf{D}) &\propto |\mathbf{R}|^{-p/2}|\mathbf{T}|^{-(n-q+p+1)/2}p(\boldsymbol{\psi}) \\ &\times \exp\left\{-\frac{1}{2}\left[\mathrm{tr}\left(\mathbf{D}^T\mathbf{G}\mathbf{D}\mathbf{T}^{-1}\right) + \mathrm{tr}\left(\left(\mathbf{B}-\hat{\mathbf{B}}\right)^T\left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)\left(\mathbf{B}-\hat{\mathbf{B}}\right)\mathbf{T}^{-1}\right)\right]\right\}, \end{aligned} \tag{2.7}$$

where $\mathbf{G} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{H}\left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}$ and $\hat{\mathbf{B}} = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{D}$ is the generalized least squares estimator of $\mathbf{B}$. Our choice of prior in (2.6) allows us to integrate out $\mathbf{B}$ and $\mathbf{T}$ from the above posterior distribution (2.7) resulting in the marginal posterior distribution

$$p(\boldsymbol{\psi}|\mathbf{D}) \propto |\mathbf{R}|^{-p/2}|\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}|^{-p/2}|\mathbf{D}^T\mathbf{G}\mathbf{D}|^{-(n-q)/2}p(\boldsymbol{\psi}). \tag{2.8}$$

Eliciting prior distributions for the correlation parameters $\boldsymbol{\psi}$ is, in general, a difficult task. We enforce the caveat that proper priors must be placed on the correlation parameters $\boldsymbol{\phi}$ and nugget η in order to ensure a proper marginal posterior distribution. In either case, Monte Carlo methods will need to be used in order to obtain posterior samples of $\boldsymbol{\psi}$.

Of main concern is deriving the posterior distribution of $\mathbf{f}(\cdot)$ given the output data $\mathbf{D}$ since this distribution will allow us to make predictions, and quantify our uncertainties, for outputs at new inputs $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_m)$. Conditional on $\mathbf{B}, \mathbf{T}, \boldsymbol{\psi}$, and $\mathbf{D}$, the posterior distribution of $\mathbf{f}(\cdot)$ is

$$\mathbf{f}(\cdot)|\mathbf{B}, \mathbf{T}, \boldsymbol{\psi}, \mathbf{D} \sim \mathrm{GP}_{mp}\left(\mathrm{vec}(\boldsymbol{\mu}^*(\cdot)), \mathbf{T} \otimes \mathbf{C}^*(\cdot,\cdot)\right) \tag{2.9}$$

where for $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_m) \in \mathcal{X}$

$$\boldsymbol{\mu}^*(\tilde{\mathbf{X}}) = \tilde{\mathbf{H}}\mathbf{B} + \mathbf{F}\mathbf{R}^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right), \tag{2.10}$$

$$\mathbf{C}^*(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) = \mathbf{C}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) - \mathbf{F}\mathbf{R}^{-1}\mathbf{F}^T, \tag{2.11}$$

and $\tilde{\mathbf{H}}^T = [\mathbf{h}(\tilde{\mathbf{x}}_1), \ldots, \mathbf{h}(\tilde{\mathbf{x}}_m)] \in \mathbb{R}^{q\times m}$ is a matrix of regression functions and $\mathbf{F} = \rho(\tilde{\mathbf{X}}, \mathbf{X}) \in \mathbb{R}^{m\times n}$. Typically, integrating the correlation parameters $\boldsymbol{\psi}$ out of (2.9) cannot be done analytically and so one instead works with the posterior distribution of $\mathbf{f}(\cdot)$ conditional on

the output data $\mathbf{D}$ and correlation parameters $\boldsymbol{\psi}$. Integrating $\mathbf{B}$ and $\mathbf{T}$ out of (2.9) yields the multivariate $\mathcal{T}$ process (Gupta & Nagar, 2000)

$$\mathbf{f}(\cdot)|\boldsymbol{\psi}, \mathbf{D} \sim \mathcal{T}_{mp}\left(\text{vec}(\boldsymbol{\mu}^*(\cdot)), \hat{\mathbf{T}} \otimes \mathbf{C}^*(\cdot,\cdot), n-q\right) \tag{2.12}$$

with $n-q$ degrees of freedom, where for $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_m) \in \mathcal{X}$

$$\boldsymbol{\mu}^*(\tilde{\mathbf{X}}) = \tilde{\mathbf{H}}\hat{\mathbf{B}} + \mathbf{F}\mathbf{R}^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right), \tag{2.13}$$

$$\begin{aligned}\mathbf{C}^*(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) = \mathbf{C}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) - \mathbf{F}\mathbf{R}^{-1}\mathbf{F}^T \\ + \left(\tilde{\mathbf{H}} - \mathbf{F}\mathbf{R}^{-1}\mathbf{H}\right)\left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\left(\tilde{\mathbf{H}} - \mathbf{F}\mathbf{R}^{-1}\mathbf{H}\right)^T,\end{aligned} \tag{2.14}$$

and $\hat{\mathbf{T}} = (n-q)^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right)^T \mathbf{R}^{-1}\left(\mathbf{D} - \mathbf{H}\hat{\mathbf{B}}\right)$ denotes the generalized least squares estimator of $\mathbf{T}$. A fully Bayesian approach can proceed by sampling from $p(\boldsymbol{\psi}|\mathbf{D})$ in order to average the conditional posterior in (2.12) with respect to $\boldsymbol{\psi}$.

2.1.3 Linear Model of Coregionalization

An alternative to separable modeling, in the field of geostatistics, the linear model of coregionalization (LMC) was developed as a tool to model multivariate spatial processes (Journel & Huijbregts (1978), Goulard & Voltz (1992), Wackernagel (2003), Gelfand et al. (2004)). Recently, Fricker et al. (2013) applied the LMC to multivariate emulators in the computer modeling literature. The idea behind the LMC is to construct output processes, $\mathbf{z}(\cdot)$, as linear combinations of a number of building-block processes $\mathbf{w}(\cdot)$. That is, dependent multivariate processes, $\mathbf{z}(\cdot)$, are obtained through a linear transformation of independent processes $\mathbf{w}(\cdot)$. The LMC is written as

$$\mathbf{z}(\cdot) = \mathbf{A}\mathbf{w}(\cdot),$$

where the p components of $\mathbf{w}(\cdot)$ are independent stationary Gaussian processes with mean μ_j, variance 1, and correlation function $\rho(\mathbf{x}, \mathbf{x}'; \phi_j, \eta)$, for $j = 1, ..., p$. The $p \times p$ full-rank matrix $\mathbf{A}$ completely defines the linear transformation of $\mathbf{w}(\cdot)$ to $\mathbf{z}(\cdot)$. The associated mean and covariance function for $\mathbf{z}(\cdot)$ are $\mathbb{E}(\mathbf{z}(\cdot)) = \boldsymbol{\mu}$ and

$$\begin{aligned}\mathbf{C}_{\mathbf{z}}(\mathbf{x}, \mathbf{x}') &= \mathbf{A}[\text{diag}\{\rho_1(\mathbf{x}, \mathbf{x}'; \phi_1, \eta), ..., \rho_p(\mathbf{x}, \mathbf{x}'; \phi_p, \eta)\}]\mathbf{A}^T \\ &= \sum_{j=1}^{p} \mathbf{a}_j \mathbf{a}_j^T \rho_j(\mathbf{x}, \mathbf{x}'; \phi_j, \eta),\end{aligned}$$

where $\mathbf{a}_j$ represents the j^{th} column of $\mathbf{A}$. Thus, the covariance function can be interpreted as a weighted sum of the individual correlation functions and $\mathbf{z}(\cdot)$ is a stationary nonseparable process.

The multivariate output $\mathbf{Y}$ can be considered as a realization from the multivariate stochastic process $\mathbf{f}(\cdot)$, and modeled, using the LMC, as

$$\begin{aligned}\mathbf{f}(\cdot) &= \boldsymbol{\mu}(\cdot) + \mathbf{z}(\cdot) \\ \boldsymbol{\mu}(\cdot) &= (\mathbf{I}_p \otimes \mathbf{H})\,\text{vec}(\mathbf{B}) \\ \mathbf{z}(\cdot)|\mathbf{A}, \boldsymbol{\phi}, \eta &\sim \text{GP}_p\,(\mathbf{0}, \mathbf{C}(\cdot, \cdot))\end{aligned} \tag{2.15}$$

where all of the parameters of the model, with the exception of $\mathbf{A}$, are the same as the parameters of the model in (2.4). Here, $\mathbf{C}(\cdot, \cdot)$ is a matrix valued covariance function that depends on parameters $\mathbf{A}$, $\boldsymbol{\phi} = (\phi_1, ..., \phi_p)$, and nugget η. Letting $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathcal{X}$ be the collection of all inputs observed so far, the covariance structure of the multivariate

Gaussian process is given by

$$\mathbf{C}(\mathbf{X},\mathbf{X}) = \begin{pmatrix} \mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{Q}_{1,2}\mathbf{A}^T & \cdots & \mathbf{A}\mathbf{Q}_{1,m}\mathbf{A}^T \\ \mathbf{A}\mathbf{Q}_{2,1}\mathbf{A}^T & \mathbf{A}\mathbf{A}^T & \cdots & \mathbf{A}\mathbf{Q}_{2,m}\mathbf{A}^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}\mathbf{Q}_{m,1}\mathbf{A}^T & \mathbf{A}\mathbf{Q}_{m,2}\mathbf{A}^T & \cdots & \mathbf{A}\mathbf{A}^T \end{pmatrix} = \sum_{j=1}^{p} \mathbf{R}(\phi_j,\eta) \otimes \mathbf{a}_j\mathbf{a}_j^T, \quad (2.16)$$

which we denote by $\mathbf{\Sigma} = \mathbf{C}(\mathbf{X},\mathbf{X})$. Also, note that $\mathbf{Q}$ is a $p \times p$ diagonal matrix with elements $\rho(\mathbf{x},\mathbf{x}';\phi_j,\eta)$ and that $\mathbf{R}(\phi_j)$ is an $m \times m$ matrix with $[\mathbf{R}(\phi_j)]_{ii'} = \rho(\mathbf{x}_i,\mathbf{x}_{i'};\phi_j,\eta)$.

Now, under the model in (2.15), if we let $\mathbf{\Theta} = (\mathbf{B},\mathbf{A},\boldsymbol{\phi},\eta)$ then the likelihood for the data is $\mathbf{Y}|\mathbf{\Theta} \sim N(\boldsymbol{\mu},\mathbf{\Sigma})$. Under a Bayesian framework we need to assign prior distributions to all of the parameters of $\mathbf{\Theta}$ in order to continue with posterior inference. As a typical choice, we place a diffuse prior, $p(\mathbf{B}) \propto 1$, on the mean function of the multivariate Gaussian process. Placing a diffuse prior on $\mathbf{B}$ is convenient from the point of view of Gibbs sampling, however, it also allows for $\mathbf{B}$ to be numerically integrated out of the posterior distribution. We found it simpler to let $\mathbf{T} = \mathbf{A}\mathbf{A}^T$ and to then place a prior distribution on $\mathbf{T}$ rather than $\mathbf{A}$; a similar approach is taken in Schmidt & Gelfand (2003). Realizing that $\mathbf{T}$ is a valid covariance matrix, we place an inverse Wishart prior on $\mathbf{T}$ with low precision and mean, and relate $\mathbf{A}$ to $\mathbf{T}$ through a suitable transformation.

Recall that we allow $\mathbf{T} = \mathbf{A}\mathbf{A}^T$, however, matrix square roots are not unique, i.e., $\mathbf{T}$ and $\mathbf{A}$ are not one-to-one, and different choices of $\mathbf{A}$ can lead to multiple models with the same $\mathbf{T}$. To ameliorate this problem, we need to specify a particular square root decomposition of $\mathbf{T}$ to obtain $\mathbf{A}$. Due to the computational convenience of working with lower triangular matrices, the Cholesky decomposition is a popular choice of square root decomposition (Gelfand et al., 2004). However, the lower triangular structure of the Cholesky

decomposition induces an ordering on the outputs $\mathbf{Y}$ and may be inappropriate when no such appropriate ordering exists. Instead, as proposed by Fricker et al. (2013), we use the eignendecomposition of $\mathbf{T}$ as an alternative to the Cholesky decomposition. The eigendecomposition is $\mathbf{A} = \mathbf{\Lambda}\sqrt{\mathbf{E}}\mathbf{\Lambda}^T$ where $\mathbf{\Lambda}$ is the orthogonal matrix of normalized eigenvectors of $\mathbf{T}$ and $\sqrt{\mathbf{E}}$ is the diagonal matrix of square roots of the eigenvalues of $\mathbf{T}$. One appeal to the eigendecomposition is that the decomposition preserves matrix permutations between $\mathbf{T}$ and $\mathbf{A}$, i.e., permuting the rows or columns of $\mathbf{T}$ will also permute the corresponding rows or columns of $\mathbf{A}$. Thus, the ordering of the outputs $\mathbf{Y}$ does not matter for the structure of the model. Lastly, proper priors for the correlation functions, $\rho_j(\mathbf{x}, \mathbf{x}'; \phi_j, \eta)$, need to be placed for each ϕ_j and nugget η. Then, the posterior distribution for $\mathbf{\Theta}$ becomes

$$p(\mathbf{\Theta}|\mathbf{Y}) \propto |\mathbf{\Sigma}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{Y}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{Y}-\boldsymbol{\mu})\right) p(\mathbf{T})p(\phi_1)\cdots p(\phi_p)p(\eta). \tag{2.17}$$

Posterior inference for the parameters of (2.17) can be carried out using MCMC methods (see Gelfand et al. (2004) and Banerjee et al. (2004) for further discussion) through a series of Gibbs sampling and Metropolis-Hastings steps. Fortunately, the full conditional distribution of $\mathbf{B}$ is obtainable in closed form as

$$p(\text{vec}(\mathbf{B})|\mathbf{T}, \boldsymbol{\phi}, \eta, \mathbf{Y}) \sim N\left(\text{vec}(\hat{\mathbf{B}}), \mathbf{H}^T\mathbf{\Sigma}\mathbf{H}\right), \tag{2.18}$$

where $\hat{\mathbf{B}} = (\mathbf{H}^T\mathbf{\Sigma}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{Y}$. Thus, posterior samples of $\mathbf{B}$ can be obtained through a Gibbs sampler. Unfortunately, the full conditional distributions for $\mathbf{T}$, $\boldsymbol{\phi}$, and η are intractable and so posterior samples of these parameters must be obtained through some other MCMC sampling strategy. Prediction under the LMC proceeds in a similar fashion to the predictive model of Section 2.1.2 with the caveat that we cannot integrate out $\mathbf{T}$ from the predictive process.

In practice we found no appreciable difference between using separable models versus the linear model of coregionalization. In the case of building surrogate models, each method provided us with models that had excellent predictive performance. Thus, we chose to use the simpler structure provided by the separable model for the rest of the theory and methodology developed in this dissertation.

2.2 Joint Modeling of Correlated Responses

Building upon the modeling framework of Section 2.1, we introduce a novel methodology for joint modeling of correlated outputs. We are particularly interested in the case of jointly modeling continuous and binary outputs that are correlated, but our framework is more general. Again denote the process of interest as $\mathbf{f}(\cdot)$, but we now allow its outputs to be of arbitrary form, which could include continuous, binary, categorical, ordinal, or other types of output.

Let $\mathcal{G}(\cdot)$ be a multivariate function that acts analogous to a link function for generalized linear models. The function $\mathcal{G}(\cdot)$ is a deterministic function that takes continuous inputs on the real line and maps them to the range of $\mathbf{f}(\cdot)$. We allow for $\mathcal{G}(\cdot)$ to be flexible in its specification and thus only impose the restriction that $\mathcal{G}(\cdot)$ be a function that preserves the range of $\mathbf{f}(\cdot)$. Where the domain and range of $\mathbf{f}(\cdot)$ are identical, it makes sense for $\mathcal{G}(\cdot)$ to be a one-to-one function. Our framework allows for quite general $\mathcal{G}(\cdot)$, however, it is often convenient to decompose $\mathcal{G}$ into $r \leq p$ independent transformations, i.e., $\mathcal{G}^T(\cdot) = (\mathcal{G}_1^T(\cdot), \ldots, \mathcal{G}_r^T(\cdot))$. Although $\mathcal{G}(\cdot)$ could be any arbitrarily complex multivariate function, we typically use a $\mathcal{G}(\cdot)$ that can be decomposed into r independent transforma-

tions such that r is equal to the number of unique output types. Thus, for $i = 1, .., r$, each $\mathcal{G}_i(\cdot)$ is a transformation of the inputs to a unique output space. When the outputs do not have the same form for all i we refer to the outputs as mixed type; for example, $\mathcal{G}_1$ might be regression outputs and $\mathcal{G}_2$ might be classification outputs. This rule of thumb is suggested in order to facilitate ease in specifying appropriate transformations that preserve the range of $\mathbf{f}(\cdot)$, as well as for model tractability in specifying the posterior distributions and the calculations that follow.

Following the model formulation in (2.4), we define $\mathbf{H}$ as before but now let $\mathbf{B} = [\mathbf{B}_1 \cdots \mathbf{B}_r] \in \mathbb{R}^{q \times p}$, where $\mathbf{B}_i \in \mathbb{R}^{q \times p_i}$ is a subset of $\mathbf{B}$, and similarly $\boldsymbol{\omega}(\cdot) = \text{vec}(\boldsymbol{\omega}_1(\cdot), \ldots, \boldsymbol{\omega}_r(\cdot))$, where $\boldsymbol{\omega}_i(\cdot) \in \mathbb{R}^{p_i}$ is a subset of $\boldsymbol{\omega}(\cdot)$, for $i = 1, \ldots, r$, and $p = p_1 + \ldots + p_r$. Now, under our proposed modeling framework, at any collection of inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{X} \subset \mathbb{R}^d$, we model the output with the following general likelihood

$$\mathcal{L}(\mathbf{B}, \mathbf{T}, \boldsymbol{\psi}; \mathbf{Y}) = \prod_{i=1}^{n} p(\mathbf{Y} | \mathcal{G}(\boldsymbol{\mu}(\mathbf{x}_i) + \boldsymbol{\omega}(\mathbf{x}_i))), \tag{2.19}$$

where we can write the full model as follows:

$$\begin{aligned} \mathbf{f}(\mathbf{X}) &= \mathcal{G}(\boldsymbol{\mu}(\mathbf{X}) + \boldsymbol{\omega}(\mathbf{X})) \\ \boldsymbol{\mu}(\mathbf{X}) &= (\mathbf{I}_p \otimes \mathbf{H}) \, \text{vec}(\mathbf{B}) \\ \boldsymbol{\omega}(\mathbf{X}) | \mathbf{T}, \boldsymbol{\phi}, \eta &\sim \text{GP}_p(\mathbf{0}, \mathbf{C}(\mathbf{X}, \mathbf{X})) \, . \end{aligned} \tag{2.20}$$

When we can decompose $\mathcal{G}$ into independent transformations, we can rewrite the component-

wise model as follows:

$$\begin{aligned}
\mathbf{f}_1(\mathbf{X}) &= \mathcal{G}_1((\mathbf{I}_{p_1} \otimes \mathbf{H})\text{vec}(\mathbf{B}_1) + \boldsymbol{\omega}_1(\mathbf{X})) \\
&\vdots \\
\mathbf{f}_r(\mathbf{X}) &= \mathcal{G}_r((\mathbf{I}_{p_r} \otimes \mathbf{H})\text{vec}(\mathbf{B}_r) + \boldsymbol{\omega}_r(\mathbf{X})),
\end{aligned} \tag{2.21}$$

where $\mathbf{f}^T(\mathbf{X}) = (\mathbf{f}_1^T(\mathbf{X}), \ldots, \mathbf{f}_r^T(\mathbf{X}))$ and

$$\boldsymbol{\omega}(\mathbf{X}) = \begin{pmatrix} \boldsymbol{\omega}_1(\mathbf{X}) \\ \vdots \\ \boldsymbol{\omega}_r(\mathbf{X}) \end{pmatrix} \sim \text{GP}_p\left(\mathbf{0}, \mathbf{C}(\mathbf{X}, \mathbf{X})\right). \tag{2.22}$$

Clearly, under the models in (2.20) and (2.21), we have induced a correlation structure between outputs by allowing $\boldsymbol{\omega}(\mathbf{X})$ to be modeled as a joint multivariate Gaussian process. Similar to Albert & Chib (1993), we introduce a latent process structure, $\boldsymbol{\ell}_i = (\mathbf{I}_{p_i} \otimes \mathbf{H})\text{vec}(\mathbf{B}_i) + \boldsymbol{\omega}_i(\mathbf{X})$, by augmenting the model with the unobservable output $\boldsymbol{\ell}_i \in \mathbb{R}^{p_i}$ that allows for a mapping between $\mathbf{f}_i(\mathbf{X})$ and $\boldsymbol{\ell}_i$. Here $\mathcal{G}^T(\boldsymbol{\ell}) = (\mathcal{G}_1^T(\boldsymbol{\ell}_1), \ldots, \mathcal{G}_r^T(\boldsymbol{\ell}_r))$ is a set of transformations that govern the mapping between the observed outputs and the latent parameters. Formulating our model this way allows for a lot of theoretical carry-over from Section 2.1 as well as model tractability. Clearly, when we allow all of the $\mathcal{G}_i(\boldsymbol{\ell}_i) = \boldsymbol{\ell}_i$ for $i = 1, \ldots, p$, akin to an identity link, we recover the multivariate Gaussian process model of Section 2.1.2. Thus, when $\mathcal{G}_i(\boldsymbol{\ell}_i)$ is the identity link, we obtain regression models, and likewise, when $\mathcal{G}_i(\boldsymbol{\ell}_i)$ involves the logistic or probit link, we obtain classification models.

For the applications in this chapter, we are interested in jointly modeling a multivariate continuous output and a binary output under our new modeling framework, and so, we let $\mathbf{f}^T(\mathbf{X}) = (\mathcal{G}_1^T(\boldsymbol{\ell}_1), \mathcal{G}_2^T(\boldsymbol{\ell}_2))$ be a multivariate stochastic process where $\mathcal{G}_1(\boldsymbol{\ell}_1) \in \mathbb{R}^{p-1}$

represents a real continuous output and $\mathcal{G}_2(\boldsymbol{\ell}_2) \in \{0, 1\}$ be a binary output. Now at any collection of inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{X} \subset \mathbb{R}^d$, we model the mixed type output as follows:

$$\mathbf{f}_1(\mathbf{X}) = \mathcal{G}_1(\boldsymbol{\ell}_1) \tag{2.23}$$

$$\mathbf{f}_2(\mathbf{X}) = \mathcal{G}_2(\boldsymbol{\ell}_2), \tag{2.24}$$

where

$$\begin{pmatrix} \boldsymbol{\omega}_1(\mathbf{X}) \\ \\ \boldsymbol{\omega}_2(\mathbf{X}) \end{pmatrix} \sim \mathrm{GP}_p \left(\mathbf{0}, \mathbf{C}(\mathbf{X}, \mathbf{X}) \right) \tag{2.25}$$

and $\mathcal{G}_1(\boldsymbol{\ell}_1) = \boldsymbol{\ell}_1$ and

$$\mathcal{G}_2(\ell_{2,i}) = \begin{cases} 1 & \text{if } \ell_{2,i} > 0 \\ \\ 0 & \text{if } \ell_{2,i} \leq 0 \end{cases} \tag{2.26}$$

where $\ell_{2,i}$ is the i^{th} element of $\boldsymbol{\ell}_2$. We define data and latent parameter matrices $\mathbf{D}$ and $\mathbf{L}$, respectively, such that $\mathrm{vec}(\mathbf{D}) = \mathbf{Y}$ and $\mathrm{vec}(\mathbf{L}) = (\boldsymbol{\ell}_1^T, \boldsymbol{\ell}_2^T)^T$, and paralleling the model in Section 2.1.2, we use the same joint prior in (2.6) for $\mathbf{B}$ and $\mathbf{T}$, and arrive at the following joint posterior density of our model:

$$\begin{aligned} &p(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \mathbf{B}, \mathbf{T}, \boldsymbol{\psi} | \mathbf{Y}) \propto |\mathbf{R}|^{-p/2} |\mathbf{T}|^{-(n-q+p+1)/2} p(\boldsymbol{\psi}) \\ &\exp\left\{ -\frac{1}{2} \left[\mathrm{tr}\left(\mathbf{L}^T \mathbf{G} \mathbf{L} \mathbf{T}^{-1} \right) + \mathrm{tr}\left(\left(\mathbf{B} - \hat{\mathbf{B}} \right)^T \left(\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \right) \left(\mathbf{B} - \hat{\mathbf{B}} \right) \mathbf{T}^{-1} \right) \right] \right\} \\ &\times \prod_{i=1}^{n(p-1)} \mathbf{1}(\mathbf{Y}_{1,i} = \boldsymbol{\ell}_{1,i}) \prod_{i=1}^{n} \left[\mathbf{1}(\ell_{2,i} > 0) \mathbf{1}(\mathbf{Y}_{2,i} = 1) + \mathbf{1}(\ell_{2,i} \leq 0) \mathbf{1}(\mathbf{Y}_{2,i} = 0) \right], \end{aligned} \tag{2.27}$$

where $\mathbf{Y}_1$ corresponds to the vector of $n(p-1)$ continuous outputs, $\mathbf{Y}_2$ corresponds to the vector of n binary outputs, $\mathbf{1}(\cdot)$ is an indicator function and $p(\boldsymbol{\psi})$ is an arbitrary proper prior distribution for the correlation parameters in (2.25). Working with the joint posterior

density in (2.27) is no simple feat, however, inference methods can be devised in order to obtain samples of all of the unknown parameters of the joint posterior density (see Section 2.3.2). Conveniently, integrating out $\mathbf{B}$ and $\mathbf{T}$ from our model is possible and thus we can derive the joint posterior predictive distribution of $(\tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2)$, conditional on $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_2$,$\boldsymbol{\psi}$, and $\mathbf{Y}$ as the following multivariate $\mathcal{T}$ process

$$\begin{pmatrix} \tilde{\boldsymbol{\ell}}_1 \\ \tilde{\boldsymbol{\ell}}_2 \end{pmatrix} \Big| \boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}, \mathbf{Y} \sim \mathcal{T}_{mp}\left(\boldsymbol{\mu}^*(\cdot), \hat{\mathbf{T}} \otimes \mathbf{C}^*(\cdot,\cdot), n-q\right) \tag{2.28}$$

where for $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_m) \in \mathcal{X}$ we have that $\boldsymbol{\mu}^*(\mathbf{X})$ and $\mathbf{C}^*(\mathbf{X}, \mathbf{X})$ are the same as (2.13) and (2.14). Obtaining samples of $\tilde{\boldsymbol{\ell}}_1$ and $\tilde{\boldsymbol{\ell}}_2$ from (2.28) proceeds immediately and allows us to quantify our uncertainty at unobserved inputs for new latent parameters. Likewise, predicting the binary value of $\mathcal{G}_2(\tilde{\boldsymbol{\ell}}_2)$ proceeds by evaluating the following integral (Rasmussen & Williams, 2006)

$$p(\mathbf{f}_2(\tilde{\mathbf{X}}) = 1 | \boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}, \mathbf{Y}) = \int\int p(\mathbf{f}_2(\tilde{\mathbf{X}}) = 1 | \tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2) p(\tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2 | \boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}, \mathbf{Y}) d\tilde{\boldsymbol{\ell}}_1 d\tilde{\boldsymbol{\ell}}_2. \tag{2.29}$$

The integral in (2.29) is analytically intractable and so we calculate the integral through Monte Carlo integration by first taking many samples of $\tilde{\boldsymbol{\ell}}_1$ and $\tilde{\boldsymbol{\ell}}_2$ from the multivariate $\mathcal{T}$ process in (2.28) and then passing those samples through the sigmoid function $p(\mathbf{f}_2(\tilde{\mathbf{X}}) = 1 | \tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2)$, which in our case is a probit function, and then averaging over those values. Classification of the binary outputs can then proceed by classifying outputs as a one if the posterior predictive probability is greater than 0.5 and otherwise classify the output as a zero.

2.3 Sequential Inference via Particle Learning

2.3.1 Particle Learning

Particle learning (PL), as established by Carvalho et al. (2010), is a type of sequential Monte Carlo (SMC) algorithm designed for online inference in dynamic models. SMC algorithms are an advantageous alternative to Markov chain Monte Carlo (MCMC) algorithms when the data arrive sequentially in time. MCMC algorithms suffer from a computational burden in online inference, due to the algorithm having to be restarted every time a new data point arrives; in SMC algorithms, this is not the case. SMC relies on particles, $\{\mathbf{S}_t^{(i)}\}_{i=1}^N$, containing the sufficient information, $\mathbf{S}_t$, about the uncertainties given data $\mathbf{z}^t = (z_1, \ldots, z_t)$ up to time t, with N denoting the total number of particles. These particles, at time t, are then used to approximate the posterior distribution, $p(\mathbf{S}_t|\mathbf{z}^t)$, where now online posterior inference continues by updating the particle approximation from time t to time $t+1$. We want to emphasize that the only thing changing in time here is the accumulation of additional data points; our model itself is not dynamic, in that the underlying parameter distributions are the same at all time steps, it is only our posterior estimates that get updated with the arrival of new data points. It can be shown (Gramacy & Polson, 2011) that the PL update to the particles can be derived from the following decomposition

$$\begin{aligned} p(\mathbf{S}_{t+1}|\mathbf{z}^{t+1}) &= \int p(\mathbf{S}_{t+1}|\mathbf{S}_t, z_{t+1}) d\mathbb{P}(\mathbf{S}_t|\mathbf{z}^{t+1}) \\ &\propto \int p(\mathbf{S}_{t+1}|\mathbf{S}_t, z_{t+1}) p(z_{t+1}|\mathbf{S}_t) d\mathbb{P}(\mathbf{S}_t|\mathbf{z}^t). \end{aligned} \tag{2.30}$$

The decomposition (2.30) suggests that we update the particle approximation in two steps:

1. *Resample*: Sample indices $\zeta(i) \sim \text{Multinomial}(w_t^{(i)}, N)$, where each index is given weight

$$w_t^{(i)} \propto p(z_{t+1}|\mathbf{S}_t^{(i)}) = \int p(z_{t+1}|\mathbf{S}_{t+1})p(\mathbf{S}_{t+1}|\mathbf{S}_t)d\mathbf{S}_{t+1}, \tag{2.31}$$

for $i = 1, \ldots, N$.

2. *Propagate*: Propagate $\mathbf{S}_t^{\zeta(i)}$ to $\mathbf{S}_t^{(i+1)}$ with a draw from $\mathbf{S}_{t+1}^{(i)} \sim p\left(\mathbf{S}_{t+1}|\mathbf{S}_t^{\zeta(i)}, z_{t+1}\right)$ to obtain a new collection of particles $\{\mathbf{S}_{t+1}^{(i)}\}_{i=1}^N \sim p(\mathbf{S}_{t+1}|\mathbf{z}^{t+1})$.

Although there exist a plethora of SMC algorithms with components similar to PL, we choose to explore particle learning based methods due to the convenient form of the posterior predictive distribution of Gaussian process models and past successes of particle learning in the univariate Gaussian process setting (Gramacy & Polson, 2011; Liang & Lee, 2014). A slight modification to the above PL steps, similar to the filter of Storvik (2002), would be to reverse the order and first propagate and then resample in similar fashion as above. In the sections that follow, we make use of both the propagate-resample framework as well as the resample-propagate framework, although both schemes could be applied in either case. In Section 2.3.2 we choose to use the propagate-resample scheme of Storvik (2002) in order to avoid lengthy MCMC steps, which we discuss in Section 2.3.2, for updating the latent parameters. In Section 2.3.3 we use the resample-propagate scheme that mimics the PL Gaussian process model of Gramacy & Polson (2011).

2.3.2 Particle Learning Joint Regression and Classification

As a first step, identifying the sufficient information, $\mathbf{S}_t$, is pivotal for any particle learning algorithm. Recalling the joint regression and classification (JRC) model in Section

2.2, we have that the sufficient information needed for the JRC model is all of the parameters of the correlation matrix in (2.3), namely, $\boldsymbol{\psi} = \{\boldsymbol{\phi}, \eta\}$. A necessary quantity, we tend to think of the latent parameters $\boldsymbol{\ell}_i$ as also part of the sufficient information. We are able to analytically integrate out $\boldsymbol{\beta}$ and $\mathbf{T}$ from the posterior predictive process in (2.28) and so we do not consider $\hat{\boldsymbol{\beta}}$ or $\hat{\mathbf{T}}$ as part of the sufficient information since they can be directly calculated from $\boldsymbol{\psi}$. However, for efficient bookkeeping and implementation we do include the correlation matrix from (2.3) as part of the sufficient information even though it can be directly calculated from $\boldsymbol{\psi}$ as well. Thus, we define the sufficient information as $\mathbf{S}_t = \{\boldsymbol{\psi}_t, \mathbf{R}_t, \mathbf{L}_t\}$ where $\mathbf{L}^t = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2]$ is the matrix of latent parameters up until time t. Similarly, we define $\mathbf{D}^t$ as the data matrix of all observed outputs up until time t, and $\mathbf{L}_{t+1} = [\tilde{\boldsymbol{\ell}}_1, \tilde{\boldsymbol{\ell}}_2]$ and $\mathbf{D}_{t+1}$ as the matrices for new latent parameters and observed outputs, respectively, at time $t+1$.

Taking a joint improper prior on $(\mathbf{B}, \mathbf{T})$ requires initializing the particles conditional on $t_0 > q+p$ data points to ensure a proper posterior. Recalling that we can integrate $\mathbf{B}$ and $\mathbf{T}$ out of (2.27), we obtain our initial particles using a Metropolis-Hasting scheme to sample $\boldsymbol{\psi}_{t_0}^{(i)}$, $\boldsymbol{\ell}_{1,t_0}^{(i)}$, and $\boldsymbol{\ell}_{2,t_0}^{(i)}$ from $p(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\psi}|\mathbf{D}^{t_0})$. Once sampled, we then deterministically calculate $\mathbf{R}_{t_0}^{(i)}$ from $\boldsymbol{\psi}_{t_0}^{(i)}$ and thereby obtain $\mathbf{S}_{t_0}^{(i)}$. After initialization of the particles, we can follow the *propagate* and *resample* steps for updating particles $\{\mathbf{S}_t^{(i)}\}_{i=1}^N$ to $\{\mathbf{S}_{t+1}^{(i)}\}_{i=1}^N$. We outline the three steps as the following:

1. *Propagate*: The first propagate step draws new latent parameters $\{\mathbf{L}_{t+1}^{(i)}\}_{i=1}^N$ from $p(\mathbf{L}_{t+1}|\mathbf{S}_t^{(i)})$ via sampling from the posterior predictive distribution in (2.28).

2. *Resample*: The resample step requires sampling indices $\zeta(i) \sim \text{Multinomial}(w_t^{(i)}, N)$

where each index is given weight

$$w_t^{(i)} \propto p(\mathbf{D}_{t+1}|\mathbf{L}_{t+1}^{(i)}, \mathbf{S}_t^{(i)}), \tag{2.32}$$

for $i = 1, \ldots, N$, which can be factored into a product of a multivariate normal distribution and a Bernoulli distribution.

3. *Propagate*: The second propagate step updates each resampled sufficient information $\mathbf{S}_t^{\zeta(i)}$ to $\mathbf{S}_{t+1}^{(i)}$ by accounting for the new data output $\mathbf{D}_{t+1}$. Since the correlation parameters of the JRC model are not dynamic, we may propagate deterministically each resampled $\boldsymbol{\psi}$ by copying $\boldsymbol{\psi}_t^{\zeta(i)}$ to $\boldsymbol{\psi}_{t+1}^{(i)}$. A key step in deterministically propagating components of $\mathbf{S}_t^{\zeta(i)}$ to $\mathbf{S}_{t+1}^{(i)}$ requires the calculation of the propagated correlation function $\mathbf{R}_{t+1}$. Once we have deterministically propagated $\boldsymbol{\psi}_t^{\zeta(i)}$ to $\boldsymbol{\psi}_{t+1}^{(i)}$, we calculate the propagated correlation matrix as follows

$$\mathbf{R}_{t+1}^{(i)} = \begin{bmatrix} \mathbf{R}_t^{(i)} & (\tilde{\mathbf{F}}^{(i)})^T \\ \tilde{\mathbf{F}}^{(i)} & \tilde{\mathbf{R}}^{(i)} \end{bmatrix} \tag{2.33}$$

where $\tilde{\mathbf{F}} = \rho(\mathbf{x}_{t+1}, \mathbf{X})$ and $\tilde{\mathbf{R}} = \rho(\mathbf{x}_{t+1}, \mathbf{x}_{t+1})$.

As noted in Section 2.3.1, we could instead have implemented a resample-propagate scheme, similar to Gramacy & Polson (2011) for GP classification models, but this would then make our propagate step dependent upon sampling the new latent parameters from $p(\mathbf{L}_{t+1}|\mathbf{S}_t, \mathbf{D}_{t+1})$, which would need to be done in an MCMC scheme that would require a large number of Metropolis-Hastings updates to ensure stationarity. This would be done within each particle, and thus considerably slowing the PL algorithm. Conversely, following a propagate-resample scheme requires no Metropolis-Hastings updates and hence no concerns over convergence within each propagate step of the latent parameters.

Like many sequential Monte Carlo algorithms, particle learning is susceptible to particle degeneracy in future resample steps. Particle degeneracy occurs when, after some iterations of the algorithm, all but one particle will have weights that are very close to zero. In particular, deterministically propagating components of $\mathbf{S}_t^{\zeta(i)}$ to $\mathbf{S}_{t+1}^{(i)}$ almost surely guarantees particle degeneracy in the future resampling steps. However, deterministic propagation does lead to significant speed gains in computation. In order to take advantage of the computational speed up of deterministic propagation, and to avoid degeneracy, we include a *rejuvenate* step inside of the *propagate* step that samples from the posterior distribution in (2.8) via MCMC for the sake of rejuvenating the particles (MacEachern et al., 1999). Following the lead of Gramacy & Polson (2011), we find a single Metropolis-Hastings step for the parameters of $\boldsymbol{\psi}_t$, for each particle, to be sufficient in avoiding particle degeneracy.

2.3.3 Particle Learning Multivariate Gaussian Process

Recall the fact that when we allow $\mathcal{G}_i(\boldsymbol{\ell}_i) = \boldsymbol{\ell}_i$, akin to an identity link, we recover the multivariate Gaussian process model of Section 2.1.2. As a special case, utilizing particle learning for inference in this model parallels the particle learning joint regression and classification (PLJRC) model, but in far more simplicity. Given the separable multivariate Gaussian process of Section 2.1.2, the sufficient information needed for this model is the same as the sufficient information of the JRC model, i.e., $\mathbf{S}_t = \{\boldsymbol{\psi}_t, \mathbf{R}_t, \mathbf{L}^t\}$. However, given the form of $\mathcal{G}(\boldsymbol{\ell})$, we no longer need to worry about $\mathbf{L}^t$ as part of the sufficient information, since we now effectively treat the latent parameters as the true observed data, and so we set $\mathbf{S}_t = \{\boldsymbol{\psi}_t, \mathbf{R}_t\}$ and $\mathbf{L}^t = \mathbf{D}^t$.

Similarly, taking a joint improper prior on $(\mathbf{B}, \mathbf{T})$ requires us to initialize our

particles conditional on $t_0 > q + p$ data points to ensure a proper posterior. We follow the same Metropolis-Hasting scheme described in Section 2.3.2. After initialization of the particles, we can follow the two step *resample* and *propagate* steps for updating particles $\{\mathbf{S}_t^{(i)}\}_{i=1}^N$ to $\{\mathbf{S}_{t+1}^{(i)}\}_{i=1}^N$. We argue, as Gramacy & Polson (2011) do, that the multivariate Gaussian process is not a dynamic model, i.e., its parameters do not change in time, and so our resample step only requires us to consider $\mathbf{L}_{t+1}$ conditional on $\mathbf{S}_{t+1}$ rather than the typical integration over $p(\mathbf{S}_{t+1}|\mathbf{S}_t)$. We outline the two steps as the following:

1. *Resample*: The resample step requires sampling indices $\zeta(i) \sim \text{Multinomial}(w_t^{(i)}, N)$, where each index is given weight

$$w_t^{(i)} \propto p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)}) = \frac{p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)})}{\sum_{i=1}^N p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)})}, \tag{2.34}$$

   for $i = 1, \ldots, N$. Conveniently, $p(\mathbf{D}_{t+1}|\mathbf{S}_t^{(i)})$ is just the probability of $\mathbf{D}_{t+1}$ under the multivariate $\mathcal{T}$ process (2.13–2.14) given $\mathbf{S}_t^{(i)}$.

2. *Propagate*: The multivariate Gaussian process is not a dynamic model, and so, we may propagate deterministically each resampled sufficient information by copying $\mathbf{S}_t^{\zeta(i)}$ to $\mathbf{S}_{t+1}^{(i)}$. Similar to before in Section 2.3.2, once we have deterministically propagated $\boldsymbol{\psi}_t^{\zeta(i)}$ to $\boldsymbol{\psi}_{t+1}^{(i)}$, we calculate the propagated correlation matrix following (2.33).

2.4 Illustrating Examples

As a proof of concept, we demonstrate the applicability of the models presented in Sections 2.3.2 and 2.3.3 with a number of illustrating examples and comparisons with previous work. Because our motivating example is a computer modeling problem, we follow

the standard approach in the computer modeling literature (Santner et al., 2003) by using a Gaussian correlation function with unknown length-scale parameter ϕ and nugget η for all of these examples, i.e.,

$$\rho(\mathbf{x}_j, \mathbf{x}_k; \boldsymbol{\phi}, \eta) = \exp\left(-\sum_{i=1}^{d} \frac{|x_{ij} - x_{ik}|^2}{\phi_i}\right) + \eta\delta_{j,k}, \tag{2.35}$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta function. Our methodology applies to any valid choice of correlation function, but here we follow the substantial literature in computer modeling. For the length-scale parameter ϕ, we use the prior suggested in Gramacy & Lee (2008) and let $\phi \sim \frac{1}{2}(\text{Gamma}(1, 20)+\text{Gamma}(10, 10))$. The prior for ϕ encodes our belief that about half of the particles should represent Gaussian process parameterizations with wavy surfaces while the other half should represent Gaussian process parameterizations that are quite smooth or approximately linear. We place a prior on the nugget parameter, $\eta \sim \text{Exp}(10)$, that allows for a moderate amount of noise, or provides robustness in fitting (Gramacy & Lee, 2012). Lastly, we specify our regression matrix $\mathbf{H}$, with regression functions $\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_n)$, such that it is equivalent to a linear regression model with an intercept term.

2.4.1 PLMGP Synthetic Data Example

To illustrate the particle learning multivariate Gaussian process (PLMGP) method, examples based on simulated data are presented. We first consider the one-dimensional synthetic sinusoidal data used originally by Higdon (2002) and later by Gramacy & Polson (2011). Now, in the case of the one-dimensional sinusoidal functions we have that

$$f(x) = \sin\left(\frac{\pi x}{5}\right) + \frac{1}{5}\cos\left(\frac{4\pi x}{5}\right) \quad \text{and} \quad g(x) = \sin\left(\frac{f(x)}{3}\right), \tag{2.36}$$

where $x \in [0, 10]$. We simulate data, $\mathbf{y}(x) = (z(x), g(x))$, from (2.36) with noise such that $z(x) \sim N(f(x), \sigma = 0.1)$. We start with an initial sample of five data points from the model (2.36) that are uniformly spaced throughout the domain of x, see Figure 2.1, and sequentially sample thirty more data points from a Uniform(0,10) distribution. Using our PLMGP algorithm with $N = 2000$ particles, we obtain posterior predictive surfaces for $f(x)$ and $g(x)$ that capture the true data generating models, without noise, very well (Figure 2.1).

Alternatively, we could ignore the correlation between the outputs of $f(x)$ and $g(x)$ and use the particle learning Gaussian processes (PLGP) (Gramacy & Polson, 2011) to model $f(x)$ and $g(x)$ independently. As an additional experiment, we randomly generated 450 data sets from the data generating model in (2.36). Each of the 450 data sets started with an initial sample of five data points and sequentially added thirty more data points as seen previously. For all 450 data sets we ran both the PLMGP and PLGP algorithms, with $N = 2000$ particles, and compared the performance of the algorithms based on two separate metrics: mean square error and coverage. When running the PLGP algorithm we defaulted to using the `R plgp` package library (Gramacy, 2012).

We calculate the distribution of the mean square errors of the fits under both the PLMGP and PLGP models (Figure 2.2). On average, the fits under the PLMGP framework do a better job and lead to smaller mean square error values. Likewise, for the 450 data sets we calculated the coverage as the percentage of time that the true output, $f(x)$ and $g(x)$, was covered (pointwise) by the 95% credible intervals. A summary of the numerical results are found in Figure 2.2. When modeling the data from $f(x)$, the PLMGP method

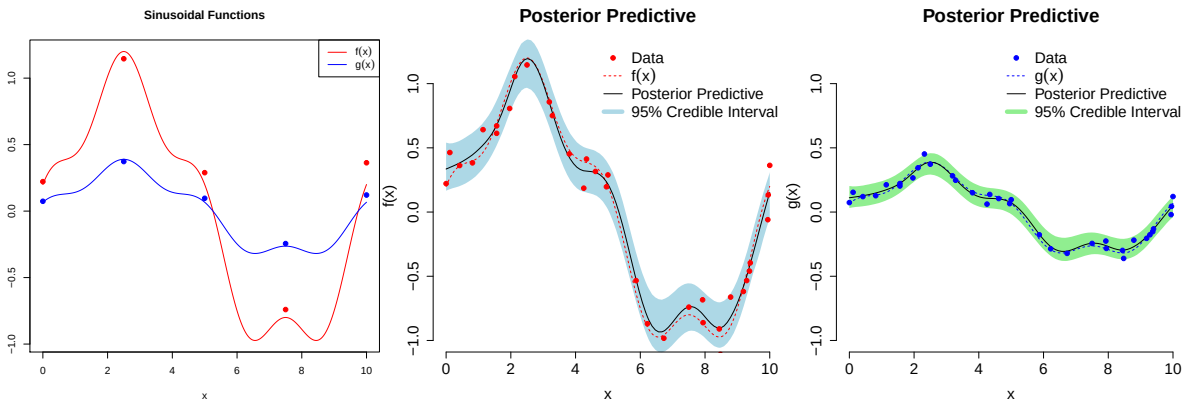


Figure 2.1: The true data generating models $f(x)$ and $g(x)$ with five observed data points (left). Also, posterior predictive surfaces of $f(x)$ (middle) and $g(x)$ (right) with 95% credible intervals.

does slightly better at being centered around the nominal 95% coverage rate while, on average, the PLGP method seemed to undercover. On the other hand, when modeling $g(x)$ both methods did a good job at being centered around the nominal 95% coverage rate. We attribute these facts to higher amount of nonstationarity needed to model $f(x)$ and to the lower amount of nonstationarity for $g(x)$. Sharing information by modeling data from $f(x)$ and $g(x)$ jointly is an advantage of the PLMGP that appears to help when modeling correlated functions that require differing amounts of nonstationary modeling. However, in both instances, the minimum observed coverage rate was lower for the PLMGP than for the PLGP.

Given our two metrics (mean square error and coverage), the PLMGP method appears to outperform the PLGP method when modeling functions that are clearly correlated. This fact should come as no surprise since the PLMGP is able to take advantage of the shared information that comes from correlated processes whereas the independent PLGP cannot. It is worth noting, however, that although the PLMGP method outperformed the

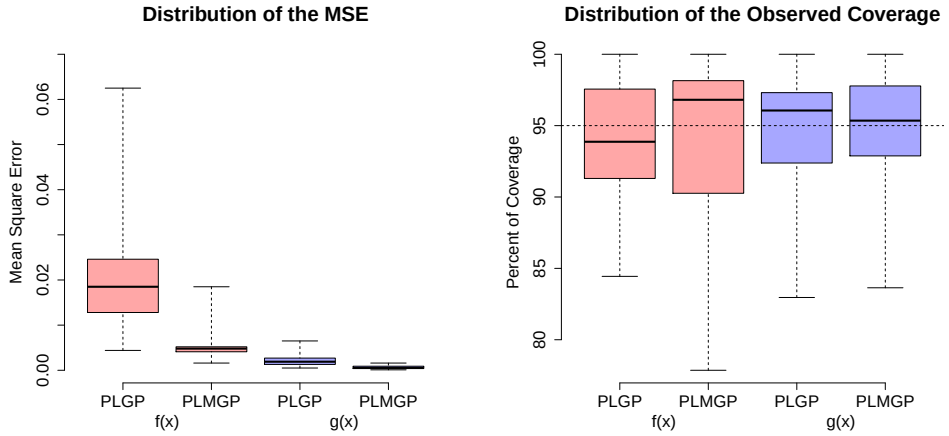


Figure 2.2: Summary of the mean square errors (left) and of the observed coverage (right) under both models for 450 repeated designs. The nominal coverage was taken to be 95% (horizontal dashed line).

PLGP method, the PLGP method still did very well at modeling the true functions.

As a further comparison, we contrasted fitting the model using MCMC versus the proposed PL scheme. Working under the same scenario as before, we generated 35 data points from (2.36), but treated the design as static and fixed, and so we started both the MCMC and PLMGP method with 5 data points and sequentially added 30 more data points to each. This was repeated 50 times in an `R` implementation on an Intel Core-i7 2.9GHz CPU computer. The efficiency of the two approaches was judged based on the average computing time and average mean square error over the 50 repetitions. Only slightly better, there did not seem to be a large difference in the prediction error between fitting the model with MCMC versus PLMGP (Table 2.1). It is worth noting that in theory, MCMC and PLMGP should achieve the same error rate. We can speculate that PLMGP might be doing slightly better here because it mixes better in a finite amount of time.

	$f(x)$		$g(x)$	
	PLMGP	MCMC	PLMGP	MCMC
Average MSE	0.0056	0.0062	0.0006	0.0007
Std. Deviation MSE	0.0015	0.0026	0.0001	0.0001

Table 2.1: Summary of the mean square errors for 50 repeated designs where the model is either fit using Markov chain Monte Carlo or particle learning.

There was however a stark contrast in average computing time where it took, on average, 25 minutes for one repetition based on 100,000 MCMC iterations and only 6 minutes, on average, for the PLMGP method to finish one repetition with $N = 4000$ particles. Clearly, the computational gain in speed is what sets the PLMGP method apart from the traditional MCMC methods when data arrives sequentially.

Although many sequential Monte Carlo methods have to account for the problem of particle depletion, in our application of PLMGP, we had effective sample sizes that were always above the commonly used threshold of $N/2$ (Prado & West, 2010).

2.4.2 PLJRC Synthetic Data Example

Now, consider data generated from the following one-dimensional dampened cosine functions, $f(x)$, used by Santner et al. (2003), $g(x)$, and the step function $h(x)$:

$$f(x) = \exp(-1.4x)\cos\left(\frac{7\pi x}{2}\right), \quad g(x) = \exp(-3x)\cos\left(\frac{7\pi x}{2}\right)$$
$$\text{and} \quad h(x) = \begin{cases} 1 & \text{if } f(x) + g(x) > 0 \\ 0 & \text{if } f(x) + g(x) \leq 0, \end{cases} \tag{2.37}$$

where $x \in [0, 1]$. Clearly, $f(x)$ and $g(x)$ are continuous functions for all x, while $h(x)$ is a binary function that is highly correlated with $f(x)$ and $g(x)$. We start with an initial sample

of ten data points from the functions in (2.37), where we sample our inputs $x \in [0, 1]$ from a Latin hypercube design (McKay et al. (1979); Urban & Fricker (2010)) and sequentially sample ten more inputs in [0,1] from the same Latin hypercube design. Using the particle learning joint regression and classification (PLJRC) algorithm, with $N = 4000$ particles, we initialize our particles and sequentially update our model using the remaining ten data points. When finished, we obtain posterior predictive surfaces for $f(x)$ and $g(x)$ that do an excellent job of capturing the true underlying data generating functions (Figure 2.3). Moreover, we are able to calculate the posterior predicted probability of $h(x) = 1$ by evaluating the expression in (2.29) via Monte Carlo integration. We classify $h(x)$ as 1 if the posterior predicted probability is greater than 0.5 and classify $h(x)$ as 0 otherwise. In this example (Figure 2.3), this method leads to a very low misclassification rate of 4%.

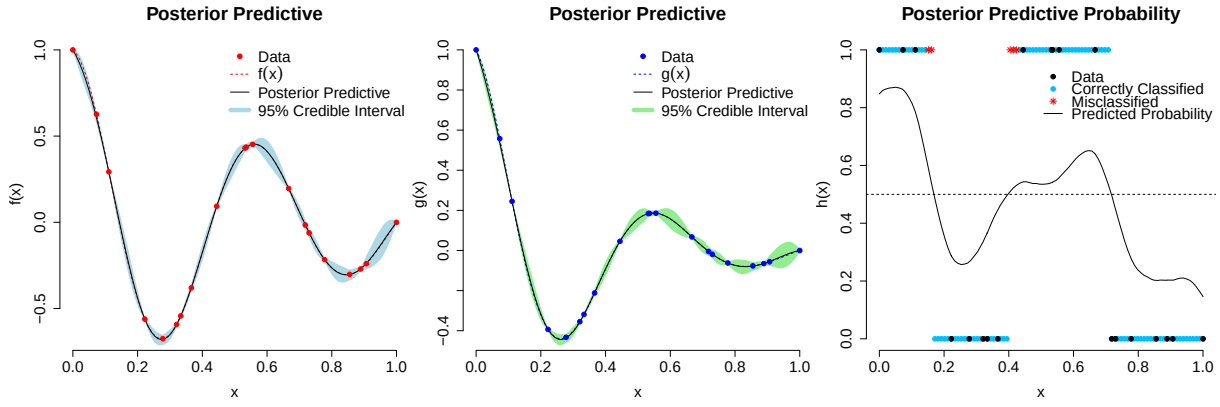


Figure 2.3: The posterior predictive surface and 95% credible intervals for $f(x)$ (left), posterior predictive surface and 95% credible intervals for $g(x)$ (middle), and posterior predictive probability associated with $h(x)$ (right). When the posterior predicted probability exceeds 0.5 we predict $h(x) = 1$ and $h(x) = 0$ otherwise.

In order to compare the performance of including correlation between outputs, we

can ignore the correlation structure between $f(x)$, $g(x)$, and $h(x)$ and use the PLGP method of Gramacy & Polson (2011) and model the posterior predictive probability associated with $h(x) = 1$ independently of $f(x)$ and $g(x)$. As a test, we randomly generated 450 data sets from the data generating functions in (2.37) and ran both the PLGP and PLJRC algorithms in order to compare the mean square error and classifications rates under each model. When running the PLGP algorithm we again defaulted to using the `R plgp` package library. Each of the 450 data sets started with an initial sample of ten data points and sequentially added ten more. The mean square error was always better for the PLJRC method than the PLGP (Figure 2.4), and the classification rate was overall better, suggesting the strength of modeling $f(x)$, $g(x)$, and $h(x)$ jointly when correlated.

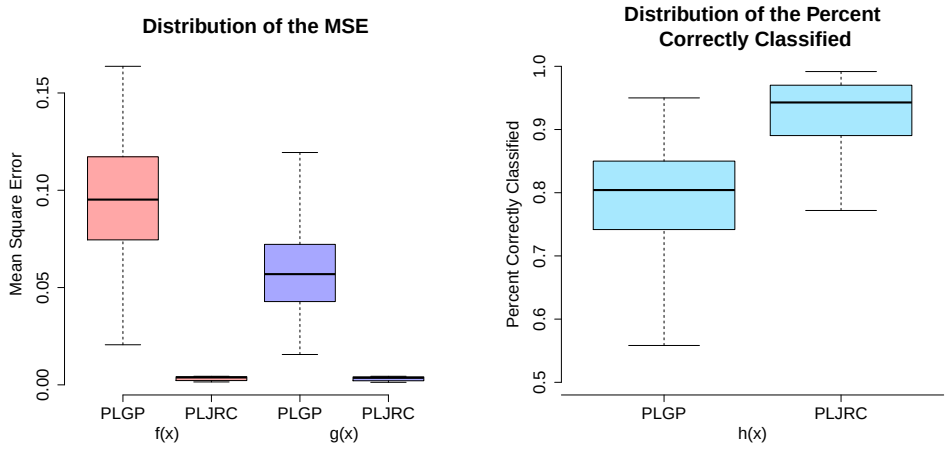


Figure 2.4: Summary of the mean square error (left) and classification rates (right) under both models for 450 repeated designs for data simulated using (2.37).

As a last comparison, we investigated the difference in fitting the model using MCMC versus the proposed PLJRC scheme. Working under the same scenario as before, we generated 20 data points from (2.37), but treated the design as static and fixed, and so we

started both the MCMC and PLJRC methods with 10 data points and sequentially added 10 more data points to each. This was repeated 50 times in an R implementation on an Intel Core-i7 2.9GHz CPU computer. The efficiency of the two approaches were judged based on the average computing time, average mean square error, and average correct classification rate over the 50 repetitions. As before, we only found slightly better differences in prediction error between fitting the model with MCMC versus PLJRC (Table 2.2).

	$f(x)$		$g(x)$		$h(x)$	
	PLJRC	MCMC	PLJRC	MCMC	PLJRC	MCMC
Average MSE	0.0038	0.0044	0.0034	0.0036	–	–
Std. Deviation MSE	0.0023	0.0019	0.0021	0.0026	–	–
Correctly Classified	–	–	–	–	0.9720	0.9590

Table 2.2: Summary of the mean square errors and classification rates for 50 repeated designs where the model is either fit using Markov chain Monte Carlo or particle learning on an Intel Core-i7 2.9GHz CPU computer.

On the other hand, classification was slightly better using MCMC rather than PLJRC with a misclassification rate of 2.8% versus 4.1%, respectively (Table 2.2). However, the difference in average computing time between the two methods is large enough to preclude the use of MCMC over PLJRC. On average, it took 22 minutes for one repetition based on 100,000 MCMC iterations and only 4 minutes, on average, for the PLMGP method to finish one repetition with $N = 4000$ particles. The computational burden of MCMC is a clear disadvantage in sequential inference as compared to the much quicker sequential Monte Carlo.

Comparable to Section 2.4.1, in our application of PLJRC, we had effective sample sizes which were always above the common threshold of $N/2$ (Prado & West, 2010).

2.4.3 Hydraulic Capture Problem

A real-world application, the hydraulic capture problem from the community problems (Mayer et al., 2002) involves a groundwater contamination scenario based on the U.S. Geological Survey computer simulator MODFLOW (McDonald & Harbaugh, 1996). The objective of the problem is to contain a plume of contaminated groundwater by installing up to four wells to control the direction of groundwater movement, and to do so at minimal cost. The MODFLOW simulator was built to model this physical process where the inputs to the computer simulator are the coordinates, (x_1, x_2), of the well and its pumping rate, x_3. We focus, as Lindberg & Lee (2015b) do, on the single well version of the problem and further constrain ourselves to the same initial conditions as Lindberg & Lee (2015b). Thus we focus our search of the input space on the region with $235 \leq x_1 \leq 270$, $580 \leq x_2 \leq 680$, and $-0.0064 \leq x_3 \leq -0.0051$, where negative rates indicate extraction. Lindberg & Lee (2015b) focused their efforts on searching this restricted area of the input space due to the fact that only about 2% of initial inputs, based on a Latin hypercube design (LHD), will yield a valid output in the original input space. Narrowing the search to a smaller region of the input space increases the number of valid outputs to about 5%.

We reformulate this problem in the framework of a constrained optimization problem as

$$\min_{\mathbf{x}} \{f(\mathbf{x}) : g(\mathbf{x}) = 1;\ 235 \leq x_1 \leq 270; 580 \leq x_2 \leq 680; -0.0064 \leq x_3 \leq -0.0051\} \quad (2.38)$$

where the objective f we wish to minimize describes the cost required to install and operate the wells. The contaminant plume is contained when the binary constraint, g, is met. The objective f and constraint g are both highly complex and nonlinear functions in which

outputs from each function can only be obtained from running the computer simulator. However, worst of all, when the constraint g is not met, the computer simulator only tells us that the constraint was not met but gives us no information about the output of the objective function f.

The time it takes to run the computer simulator is nontrivial and so it not feasible to run the computer simulator at every possible combination of inputs and find the one that optimizes the problem (2.38). Instead, we proceed by constructing a surrogate model (Sacks et al., 1989; Santner et al., 2003) sequentially while searching for the minimum of the response surface (Jones et al., 1998; Taddy et al., 2009). We construct our surrogate model by modeling both the continuous objective f and the binary constraint g jointly using our PLJRC model. Modeling f and g as correlated makes sense in this context, because extracting more fluid is more likely to meet the constraint (contain the plume of contamination) but will cost more; extracting less fluid will be cheaper but less likely to meet the constraint. The model is sequentially updated by selecting new candidate inputs, $\mathbf{x}_*$, that maximize the probability of finding a smaller objective function value f. Our approach toward this goal is that of expected improvement (Jones et al., 1998). We define the improvement statistic at a proposed input $\mathbf{x}$ to be

$$I(\mathbf{x}) = \max\{f_{\min} - f(\mathbf{x}), 0\}, \tag{2.39}$$

where, after N runs of the simulator, $f_{\min} = \min\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)\}$ is the current minimum value observed. Since the proposed input $\mathbf{x}$ has not yet been observed, $f(\mathbf{x})$ is unknown and, conditional on the observed inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$, the distribution of $f(\mathbf{x})$ can be represented by using the posterior predictive distribution of the PLJRC. Now that $I(\mathbf{x})$ can be regarded

as a random variable, we choose new candidate inputs, $\mathbf{x}_*$, by selecting those inputs that maximize the expected improvement

$$\mathbf{x}_* \in \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}\{I(\mathbf{x})\}. \tag{2.40}$$

Fortunately, conditional on a particular parameterization of the PLJRC, the expected improvement is available in closed form as

$$\mathbb{E}\{I(\mathbf{x})\} = (f_{\min} - \mu(\mathbf{x}))\Phi\left(\frac{f_{\min} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{f_{\min} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right), \tag{2.41}$$

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the mean and standard deviation of the posterior predictive distribution of $f(\mathbf{x})$ and $\Phi(\cdot)$ and $\phi(\cdot)$ are the standard normal cdf and pdf, respectively. The equation in (2.41) provides a combined measure of how promising a candidate point is, that trades off between local search ($\mu(\mathbf{x})$ under $f_{\min}$) and global search ($\sigma(\mathbf{x})$). However, in the presence of constraints, it makes no sense to search for candidate inputs that maximize the expected improvement yet violate the constraints. Thus, we go one step further and choose candidate inputs, $\mathbf{x}_*$, by selecting those inputs that maximize the following (Lindberg & Lee, 2015b)

$$\mathbf{x}_* \in \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}\{I(\mathbf{x})\}^{\alpha_1} S(\mathbf{x})^{\alpha_2}, \tag{2.42}$$

where $S(\mathbf{x})$ is the asymmetric entropy defined as

$$S(\mathbf{x}) = \frac{2p(\mathbf{x})(1-p(\mathbf{x}))}{p(\mathbf{x}) - 2wp(\mathbf{x}) + w^2} \tag{2.43}$$

and $p(\mathbf{x})$ is the predicted probability that the constraint is satisfied at the input $\mathbf{x}$. Here, α_1, α_2, and w are constants that we set equal to the default values suggested in Lindberg & Lee (2015b), and so, $\alpha_1 = 1, \alpha_2 = 5$, and $w = 2/3$. Thus, maximizing the quantity in (2.42)

is a trade-off between maximizing the expected improvement and the probability of meeting the constraints. Calculating the probability that an input $\mathbf{x}_*$ satisfies the constraint, i.e., $p(\mathbf{x}) = \Pr(g(\mathbf{x}) = 1)$, is not a trivial task; however, we make use of the posterior predictive probability calculations for $g(\mathbf{x})$ under our PLJRC model in order to make calculations of the probability of meeting the constraint. In many constrained minimization problems, the probability of the constraint being satisfied is correlated with the value of the objective function, so that the expected improvement is in opposition to the probability of meeting the constraint; this makes the problem difficult. Our JRC model can handle this correlation, in contrast to the existing models in the computer experiment literature.

To show the advantage of modeling the objective and constraint as correlated, we followed the same setup as Lindberg & Lee (2015b) and began with an initial sample size of 65 inputs, based on a LHD, to run the MODFLOW simulator at. The output from the MODFLOW simulator was then used to fit our PLJRC model where we chose to use $N = 2000$ particles. The search for the optimal solution then proceeded by sequentially selecting 300 more inputs to run the MODFLOW simulator at based on choosing points that maximized the expected constrained improvement in (2.42). We follow the strategy of Taddy et al. (2009) and select the candidate set of inputs from a LHD of size 500 times the input dimension augmented by an additional 10% of the candidate locations taken from a smaller LHD bounded to within 5% of the domain range of the current best point. This hybrid space filling design further ensures that our algorithm searches both locally as well as globally.

The best solution our algorithm found is a cost of \$22,952.8 by setting the co-

ordinates of the pump to $(x_1, x_2) = (258.8451, 638.3419)$ and the extraction rate to $x_3 = -0.005320973$. Our solution reached the same cost values found in Lindberg & Lee (2015b) and Lee et al. (2011). Although under slightly different initial conditions, our optimal cost found was much better than all eight of the solutions found in Fowler et al. (2008), with the best solution reported there being \$23,421 found using Implicit Filtering for Constrained Optimization (IFFCO). Likewise, we reran the same analysis 30 times, and the average value found after 300 runs reached by our algorithm (\$22,953.3), see Figure 2.5, was much lower than the best values found by eight competing optimization algorithms as reported in Fowler et al. (2008).

We ran the independent PLGP methodology 30 times under the same setup and conditions as the joint PLJRC methodology and found that both methods lead to comparable optimal solutions. Over those 30 Monte Carlo repetitions, the independent PLGP models also found an optimal cost of \$22,952.8. However, it was clear that using the independent PLGP model did not always result in an optimal solution within the 300 sequential updates, see Figure 2.5, whereas the PLJRC model consistently found the optimal solution in fewer than 150 sequential updates. Thus, it would seem that the PLJRC model was able to take advantage of the correlation structure between the objective and constraint functions that the independent PLGP model was not.

The framework of expected constrained improvement is heavily reliant on the models behind both the objectives and constraints. Here we demonstrate that our PLJRC model does a very good job at modeling the objective and constraint surfaces and allows the constrained expected improvement mechanism to outperform other established solutions. In

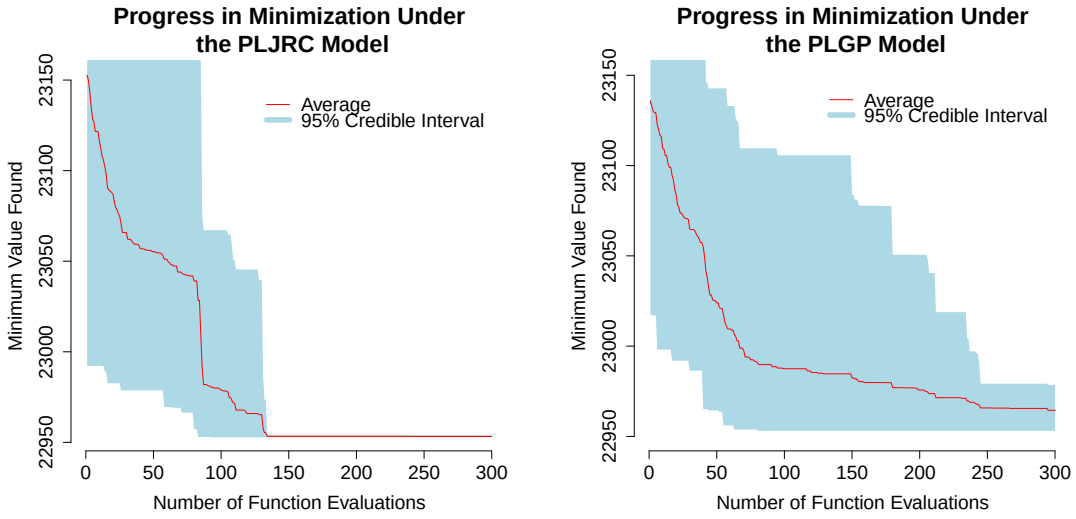


Figure 2.5: After 300 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint (left) and independent (right) models. The plot shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found.

constrained optimization, the problems are typically difficult because the objective function and the constraint satisfaction are in opposition, i.e., strongly negatively correlated. Our model performs better by fully modeling this correlation.

2.5 Discussion

We have established a novel methodological framework for modeling mixed type outputs with fully Bayesian inference. By introducing latent outputs we are able to jointly model correlated outputs. In particular, we showed how our framework could be applied in the case of jointly modeling continuous and binary outputs that were correlated, particularly for constrained optimization of computer simulation experiments. We combined the strengths of our joint regression and classification (JRC) model with the speed of sequential Monte Carlo methods to conduct fast inference. Sequential inference for the JRC model based on MCMC is inefficient because it requires rerunning the MCMC every time a new data point arrives. A considerable reduction in computation time is achieved by applying particle learning methodology to the JRC model. Thus, the marriage of the JRC model with particle learning (PLJRC) is indeed a powerful new technique. We highlighted the strengths of the PLJRC over other competing methodologies (such as the PLGP) with simulated examples and were able to show its practical usefulness in a real-world constrained optimization problem of a computer simulator.

The introduction of latent parameters for joint modeling underlies the innovation of our model. By directly allowing the latent parameters to be the observed outputs, in an identity link fashion, we can recover the multivariate Gaussian process model of Section

2.1. Furthermore, this also allows us to extend the particle learning algorithm to the model of Section 2.1, thereby creating a new stochastic modeling technique (PLMGP) comparable to that of Gramacy & Polson (2011). This framework also provides a novel implementation of particle learning for multivariate Gaussian processes.

An obvious extension within our framework would be to extend the model to a richer class of outputs. Using suitable transformations $\mathcal{G}(\cdot)$, we can envision an infinite continuum of correlated data types that could be modeled jointly.

2.5.1 Author Remarks

The work that appears in this chapter of the dissertation was recently accepted for publication (Pourmohamad & Lee, 2015) in the journal *Bayesian Analysis*.

Chapter 3

Statistical Filters

3.1 Filter Methods

Filter methods were introduced by Fletcher & Leyffer (2002) as a means of solving nonlinear programming problems without the use of a penalty function. Penalty functions suffer from the drawback of having to specify a suitable penalty parameter that balances the often-competing aims of minimizing f and h. Instead of combining the objective and constraint violation into a single function (1.3), filter methods take a biobjective optimization approach and try to minimize both f and h simultaneously. However, priority is placed on minimizing h since a feasible solution only exists when $h(\mathbf{x}) = 0$. Borrowing concepts from multiobjective optimization, filter methods solve the constrained optimization problem in (1.2) by locating the set of all nondominated inputs $\mathbf{x} \in \mathcal{X}$. A point $\mathbf{x}_i \in \mathcal{X}$ dominates a point $\mathbf{x}_j \in \mathcal{X}$ if and only if $f(\mathbf{x}_i) \leq f(\mathbf{x}_j)$ and $h(\mathbf{x}_i) \leq h(\mathbf{x}_j)$ with $(h(\mathbf{x}_i), f(\mathbf{x}_i)) \neq (h(\mathbf{x}_j), f(\mathbf{x}_j))$. Geometrically, $\mathbf{x}_i$ dominates $\mathbf{x}_j$ if $\mathbf{x}_i$ is to the "southwest" of $\mathbf{x}_j$. Fletcher & Leyffer (2002) defined the filter (similar to the Pareto front in the mul-

tiobjective literature), denoted $\mathcal{F}$, as the set of all pairs $(h(\mathbf{x}_i), f(\mathbf{x}_i))$ such that no pair dominates another pair. Given the definition of the filter $\mathcal{F}$, we summarize the generic filter method as follows:

Initialize the filter $\mathcal{F}$;

while *not terminated* **do**

  Solve a subproblem to obtain a candidate point $\mathbf{x}_*$;

  Evaluate $f(\mathbf{x}_*)$ and $c(\mathbf{x}_*)$;

  if $(h(\mathbf{x}_*), f(\mathbf{x}_*))$ *is acceptable to* $\mathcal{F}$ **then**

    Add $(h(\mathbf{x}_*), f(\mathbf{x}_*))$ to $\mathcal{F}$;

    Remove any entries in $\mathcal{F}$ dominated by $(h(\mathbf{x}_*), f(\mathbf{x}_*))$;

  end

  Check for termination;

end

Algorithm 1: Generic filter method

Typically termination of the algorithm requires that some tolerance, with respect to the solution, has been achieved or that all budgetary resources, for example time or money, have been exhausted. As simple as Algorithm 1 may look, extra care must be taken to avoid convergence to infeasible points ($h(\mathbf{x}) > 0$) or to feasible points that are not stationary points of (1.2) (Fletcher et al., 2006). One proposed way to avoid these pitfalls is an envelope, which is added around the current filter to avoid convergence to infeasible

points. A candidate point x_* is acceptable to the filter if

$$h(\mathbf{x}_*) \leq \beta h(\mathbf{x}_i) \text{ or } f(\mathbf{x}_*) \leq f(\mathbf{x}_i) - \gamma h(\mathbf{x}_*) \qquad \forall (h(\mathbf{x}_i), f(\mathbf{x}_i)) \in \mathcal{F}, \tag{3.1}$$

where $\beta, \gamma \in (0, 1)$ are constants. The envelope in (3.1) has stronger convergence properties (Chin & Fletcher, 2003) due to its sloping nature although an axis aligned envelope could be used if γ is allowed to equal zero. Likewise, an upper bound U may be placed on the constraint function in order to ensure a practical limit to the largest allowable constraint violation to the filter. An illustration of a filter with sloping envelope is given in Figure 3.1. Note that the orange shaded area in Figure 3.1, defined by U, is an upper bound on the acceptable constraint violation.

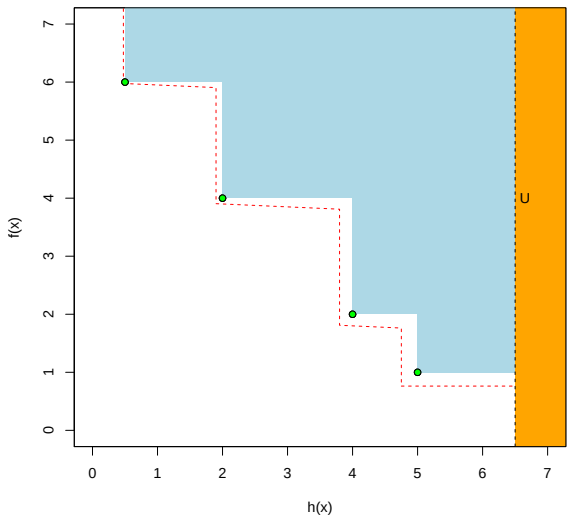


Figure 3.1: A typical filter. All pairs $(h(\mathbf{x}), f(\mathbf{x}))$ that are below and to the left of the envelope (red dashed line), and to the left of U, are acceptable to the filter.

Lastly, it is important to note that Algorithm 1 is a general outline of the filter method with no explicit explanation of how to obtain a candidate $\mathbf{x}_*$. Evaluating $(f(\mathbf{x}_*)$,

$h(\mathbf{x}_*)$) and updating the filter are relatively simple tasks, however, the main difficulty of Algorithm 1 is in specifying and solving the subproblem. Many different subproblems have been proposed within the filter method framework, however, we prefer to think of the subproblem as a challenge in statistical modeling. In particular, our preferred approach relies heavily on surrogate modeling via the particle learning multivariate Gaussian process (PLMGP) model of Section 2.3.3.

3.2 Statistical Methods

We combine the filter methods of Section 3.1 with statistical methods in order to solve constrained optimization problems of the form (1.2). Consider an expensive black box computer model with a d-dimensional input $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ that produces a p-dimensional vector of outputs $\mathbf{y} \in \mathbb{R}^p$ where the output $\mathbf{y}$ contains the values of the objective function f and constraint function $\mathbf{c}$ at input $\mathbf{x}$. We follow the traditional statistical modeling techniques from the computer modeling literature (Sacks et al., 1989; Santner et al., 2003; Conti & O'Hagan, 2010) and build surrogate models based on joint multivariate Gaussian processes for the objective and constraint functions. Due to the expense of evaluating our black box model, we use the particle learning methods developed in Section 2.3.3 to sequentially update our joint multivariate Gaussian processes. As a general guideline, we follow the rule of thumb put forth by Loeppky et al. (2009), and require that the number of initial runs (or samples) of the computer model to be about ten times the input dimension, i.e., $n = 10d$, in order to achieve reasonable surrogate model fits. Likewise, we sample inputs for our initial $n = 10d$ runs of the computer model using a Latin Hypercube Design

(LHD) (McKay et al., 1979; Santner et al., 2003; Fang et al., 2005). In our experience, we found LHD to be an adequate space-filling design for sampling our initial inputs and possible future candidate inputs under our PLMGP framework.

In order to combine the filter methodology with the surrogate modeling, it is important to understand the three different spaces we are working in. Probably the most important space, we denote the "design space" as the domain of the inputs of the computer experiment (Figure 3.2).

Being able to understand, and obtain a representative sample of, the design space is imperative for a good robust optimization algorithm. Furthermore, the design space directly influences the surrogate building in the other two spaces and can be critical in their design. The second space, denoted the constraint space, is built from the constraint function, $\mathbf{c}(\mathbf{x})$, and the objective function, $f(\mathbf{x})$, at the inputs from the design space (Figure 3.2). The constraint space is always at minimum two dimensions, i.e., $p \geq 2$, but can have dimension as high as the number of constraints plus one, i.e., $p \geq m + 1$, and can thus be extremely high dimensional. Lastly, the third space, denoted the filter space, is built from the objective function, $f(\mathbf{x})$, and the measure of feasibility

$$h(\mathbf{x}) = ||\max\{\mathbf{0}, \mathbf{c}(\mathbf{x})\}||_1 = \sum_{i=1}^{p-1} \max\{0, c_i(\mathbf{x})\} \tag{3.2}$$

The filter space is always two dimensional since the measure of feasibility, $h(\mathbf{x})$, is an aggregate measure of the constraint functions (Figure 3.2). Important to note, points will be distinct in the design space, but may not be so in the other two space, i.e., $\mathbf{x}_1 \neq \mathbf{x}_2$ with $c(\mathbf{x}_1) = c(\mathbf{x}_2)$ and/or $f(\mathbf{x}_1) = f(\mathbf{x}_2)$.

Moreover, each space also serves its own unique purpose. The design space is

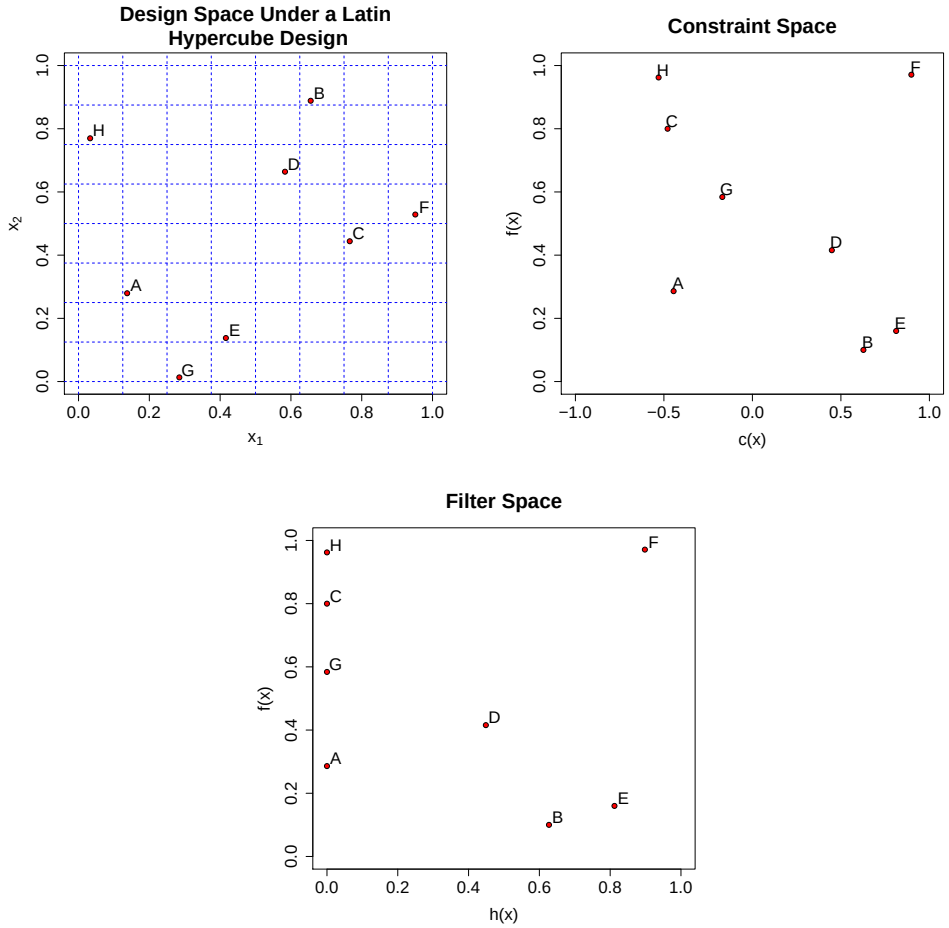


Figure 3.2: An example of a 2-D design space (top left) where eight points were sampled using a Latin hypercube design. Notice that no two points occupy the same row or column in the Latin hypercube design. Also, an example of the constraint space $(c(\mathbf{x}), f(\mathbf{x}))$ (top right) and the mapping to the filter space $(h(\mathbf{x}), f(\mathbf{x}))$ (bottom). Points in the constraint space such that $c(\mathbf{x}) \leq 0$ get mapped to $h(\mathbf{x}) = 0$ in the filter space.

responsible for the sampling of inputs and dictates where the optimization algorithm is able to explore. Bad representative sampling of the inputs in the design space can lead to poor performance of the optimization algorithm. The constraint space is used solely for the fitting of the surrogate models. The objective function, $f(\mathbf{x})$, and the constraint functions, $c(\mathbf{x})$, are modeled in the constraint space and mapped deterministically into the filter space based on the fitted surrogate models. Once in the filter space, the actual filter methods of Section 3.1 are applied and new candidate points are searched for until a minimum has been declared. Updating our generic filter Algorithm 1, we have the following new algorithm:

Sample initial inputs from a LHD;

Initialize the filter $\mathcal{F}$;

while *not terminated* **do**

  Fit surrogate models for $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$ using the joint PLMGP model;

  Map the surrogate model in the constraint space to the filter space;

  Solve a subproblem to obtain a candidate point $\mathbf{x}_*$;

  Evaluate $f(\mathbf{x}_*)$ and $c(\mathbf{x}_*)$;

  if $(h(\mathbf{x}_*), f(\mathbf{x}_*))$ *is acceptable to* $\mathcal{F}$ **then**

    Add $(h(\mathbf{x}_*), f(\mathbf{x}_*))$ to $\mathcal{F}$;

    Remove any entries in $\mathcal{F}$ dominated by $(h(\mathbf{x}_*), f(\mathbf{x}_*))$;

  end

  Check for termination;

end

Algorithm 2: Statistical filter method

Although we have an updated statistical filter method, we still need to solve a subproblem in order to obtain a new better candidate point $\mathbf{x}_*$. In the following two sections we address two subproblems for selecting candidate points and test their performance on many synthetic test problems (Section 3.3) and a real-world hydrology computer experiment (Section 3.6).

3.2.1 Probability Beyond the Filter

The first subproblem, dubbed "probability beyond the filter" (PBF), selects a new candidate point, $\mathbf{x}_*$, by maximizing the probability that the candidate point falls beyond (to the southwest of) the current filter. Stated more formally, we wish to find an $\mathbf{x}_*$ such that

$$\mathbf{x}_* = \max_{\mathbf{x}\in\mathcal{X}} \Pr\{(h(\mathbf{x}), f(\mathbf{x})) \text{ is acceptable to the filter } \mathcal{F}\}, \tag{3.3}$$

Having fit a joint surrogate model to $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$, using our joint PLMGP model, the joint predictive distribution of $(f(\mathbf{x}), \mathbf{c}(\mathbf{x}))$ is a multivariate $\mathcal{T}$ process with mean and covariance matrix given by (2.13) and (2.14) at input $\mathbf{x}$. Given this fact, we can obtain a best prediction for $(f(\mathbf{x}), \mathbf{c}(\mathbf{x}))$ from (2.13), and quantify the uncertainty around that prediction with a probability ellipse based on (2.14). For example, Figure 3.3 shows a typical filter, with envelope, and two candidate points A and B and their respective 95% probability ellipses based on their multivariate $\mathcal{T}$ process. Thus, finding a candidate point $\mathbf{x}_*$ satisfying (3.3) is equivalent to finding a candidate point with the largest area of its ellipse falling outside (to the left and below) of the filter's envelope. Although calculating this area of the ellipse is a well posed problem, solving for the area analytically is not a trivial task, and so we use

Monte Carlo integration to estimate the probability beyond the filter instead.

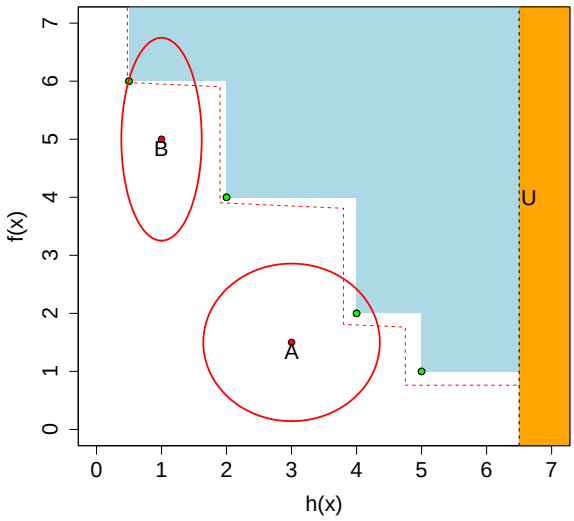


Figure 3.3: The probability beyond the filter subproblem. Selecting between candidate points A and B is determined by finding the area of their respective ellipses to the left of the filter envelope.

3.2.2 Maximum Expected Area

Denote the filter produced by a new candidate point, $\mathbf{x}_*$, as $\mathcal{F}_*$. The second subproblem, coined "maximum expected area" (MEA), selects a new candidate point, $\mathbf{x}_*$, by maximizing the area between the current filter's envelope and the new filter $\mathcal{F}_*$ that would be created by accepting $\mathbf{x}_*$. Candidate points $\mathbf{x}_*$ are predicted at the mean of the joint distribution of $(f(\mathbf{x}_*), \mathbf{c}(\mathbf{x}_*))$, which is given by the mean in (2.13) of the multivariate $\mathcal{T}$ process in Section 2.1 at input $\mathbf{x}_*$. Figure 3.4 gives a geometric representation of the MEA subproblem where the dark blue polygons correspond to the area created by choosing candidate point A or B. Clearly, in Figure 3.4 the area is maximized by selecting point A

over point B.

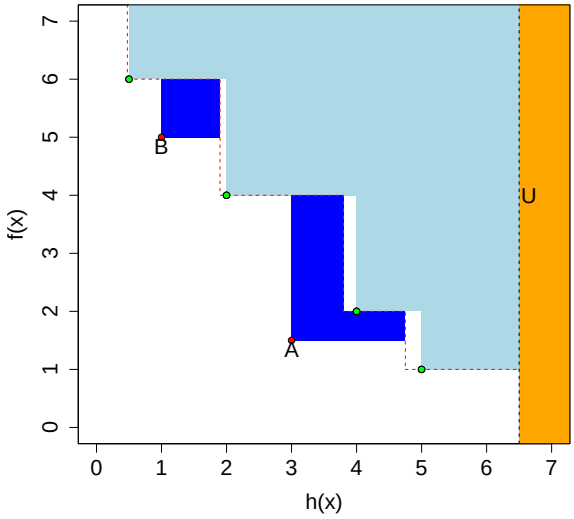


Figure 3.4: The maximum expected area subproblem. Candidate point A would be preferable to candidate point B since its dark blue area is larger.

3.2.3 Joint Modeling of the Objective and Constraints

Constrained optimization is typically difficult because at least one of the constraint violations operates in opposition to the objective function, i.e., they are negatively correlated. We model the objective and constraint functions jointly because we want to be able to capture this negative dependency assumption, however, we could also forgo this assumption and model the objective and constraint functions independently using independent particle learning Gaussian process (PLGP) models instead. There is an increase in the time and computational burden of joint modeling as compared to independent modeling of the objective and constraint functions, however, as seen in Chapter 2, there are significant gains

in predictive accuracy and coverage by use of joint modeling. Given that both the solutions of the subproblems in Sections 3.2.1 and 3.2.2 rely heavily on the accuracy of the prediction of new candidate points, it is imperative that we do a good job in modeling the objective and constraint functions. Furthermore, modeling the objective and constraint functions independently implicitly assumes that the objective function and constraint functions are not correlated. Clearly the independence assumption can be violated in real-world applications (for example the hydraulic capture problem in Section 2.4.3) and typically, a rational assumption would be to assume that the objective and constraint function are negatively correlated. Moreover, the benefits of using our PLMGP model in the statistical filter rather than the independent PLGP model is twofold. First off, the PLMGP model in the statistical filter could improve our probability beyond the filter (PBF) calculations immensely when the objective function and constraint functions are correlated. Under the independence assumption the ellipses in our probability calculations will be always axis aligned which could lead to an under (or over) statement in our probability calculation. Allowing the objective and constraint functions to be modeled jointly allows for the ellipses to be angled and non-axis aligned (Figure 4.1) resulting in more accurate probability calculations when correlations are present.

Secondly, when the objective and constraint function are correlated, the shared information in modeling the functions jointly can lead to better model fits, which would ultimately lead to far fewer function evaluations. Being able to obtain good surrogate models with few functions evaluations is especially important when modeling expensive black box functions where the time to evaluate function calls is a limiting factor. Thus,

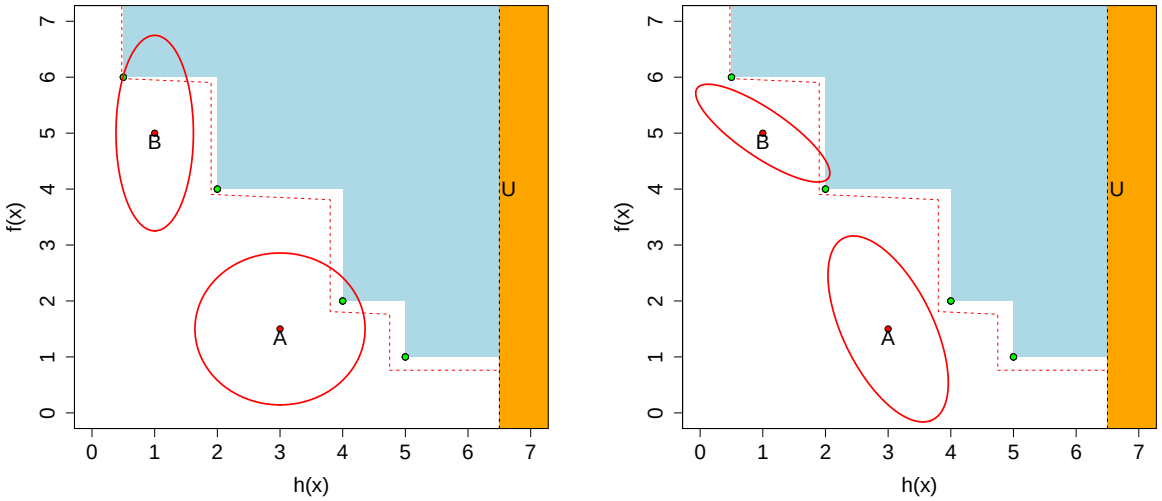


Figure 3.5: Modeling the objective and constraint functions jointly (right) versus independelty (left). Modeling the objective and constraint functions using PLMGP can lead to probability ellipses that are non-axis aligned.

we choose to implement the PLMGP model as the de facto choice for solving the filter subproblems instead of the PLGP model. We validate our choice to use joint modeling instead of independent modeling by exploring their differences in Section 3.3.

3.3 Synthetic Test Problems

We demonstrate the effectiveness of our statistical filter method on a suite of synthetic test problems. Each test problem was chosen to illustrate the many different kinds of black box computer models one could expect to encounter in the real-world. For all the test problems that follow, we employ the standard approach in the computer modeling literature (Santner et al., 2003) by using a Gaussian correlation function with unknown

length-scale parameter ϕ and nugget η for all of the test problems, i.e.,

$$\rho(\mathbf{x}_j, \mathbf{x}_k; \boldsymbol{\phi}, \eta) = \exp\left(-\sum_{i=1}^{d} \frac{|x_{ij} - x_{ik}|^2}{\phi_i}\right) + \eta\delta_{j,k} \tag{3.4}$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta function. For the length-scale parameter ϕ, we use the prior suggested in Gramacy & Lee (2008) and let $\phi \sim \frac{1}{2}(\text{Gamma}(1, 20) + \text{Gamma}(10, 10))$. The prior for ϕ encodes our belief that about half of the particles should represent Gaussian process parameterizations with wavy surfaces while the other half should represent Gaussian process parameterizations that are quite smooth or approximately linear. We place a prior on the nugget parameter, $\eta \sim \text{Exp}(10)$, that allows for a moderate amount of noise, or provides robustness in fitting (Gramacy & Lee, 2012). Lastly, we specify our regression matrix $\mathbf{H}$, with regression functions $\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_n)$, such that it is equivalent to a linear regression model with an intercept term.

The following test problems, P1–P6, are indicative of some of the types of real-world black box computer models that exist. The spectrum of problems ranged from those with single inputs and single constraints, to multiple inputs and multiple constraints, with a mix of everything in between. The goal of each test problem is to minimize the objective function f while satisfying the constraint that $c(\mathbf{x}) \leq 0$. We ran our statistical filter algorithm (Algorithm 2) on each test problem using both independent PLGP models and our joint PLMGP model and solving each using the PBF and MEA subproblems. When solving the PBF subproblem we use a sloping envelope for the filter and let $\beta = 0.95$ and $\gamma = 0.05$. For ease of calculation, we chose not to let the envelope be of the sloping form for the MEA subproblem and thus the parameters for the envelope were set to $\beta = 0.95$ and $\gamma = 0$. Furthermore, for each synthetic test problem, we evoked the rule of thumb proposed

by Loeppky et al. (2009) and started with an initial sample of inputs of size $n = 10d$ and then sequentially selected, one-at-a-time, an additional amount of candidate points based on maximizing either the probability beyond the front or the maximum expected area. The results are as follows:

Problem 1: P1

One of the simpler cases, this test problem, P1, represents the case of a single input x and single nonlinear constraint $c(x)$. Here, we would like to minimize the objective, f, while simultaneously satisfying the constraint, c, i.e.,

$$\min_{x} \quad f(x) = \cos\left(\frac{\pi x}{5}\right) + 0.2\sin\left(\frac{4\pi x}{5}\right) \tag{3.5}$$

$$\text{s.t.} \quad c(x) = \sin\left(\frac{\pi x}{5}\right) + 0.2\cos\left(\frac{4\pi x}{5}\right) \leq 0 \tag{3.6}$$

$$x \in [0, 10]. \tag{3.7}$$

The optimal solution to P1 is $f(x) = -0.8671835$ and occurs $x = 5.25585$. Common to many constrained optimization problems, the solution of P1 lies at the boundary of the constraint function (Figure 3.6).

Before we delve into the results of applying our statistical filter algorithm, it is worthwhile to take the extra time to step through the details of the setup of the problem. In general, all of the problems, P1–P6, have their own unique intricacies, but we shall spend extra time and care in explaining problem P1, with all the figures and analyses that follow being generalizable to the rest of the problems (P2–P6).

In Section 3.2, we alluded to the fact that it is very important to understand the different spaces of the optimization problem. For P1, the input space is the simple to

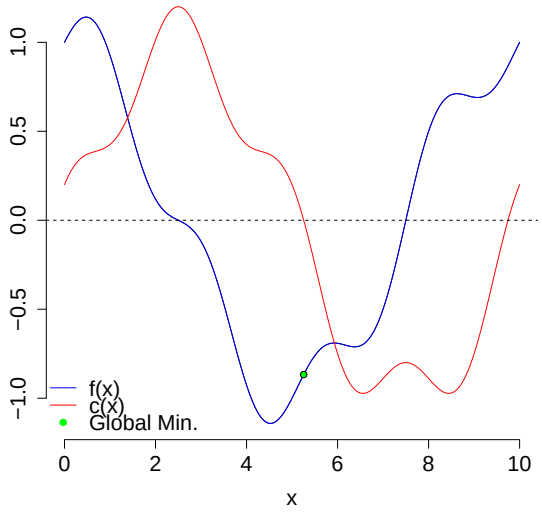


Figure 3.6: Problem 1: We want to minimize the objective, f, while satisfying the constraint, c. Values of the objective function (blue) are only feasible when the constraint function (red) is below the dashed line. The green dot corresponds to the global minimum solution to the problem and is found to lie along the constraint boundary $c(x) = 0$.

understand one dimensional closed interval on 0 to 10, however, the constraint and filter space are much more complicated and fascinating to look at. Recall that the constraint space consists of all pairs of points $(c(x), f(x))$. Interestingly, the constraint space, for P1, forms the shape of a star (Figure 3.7), while the filter space retains the same star shape but is truncated to the left at zero.

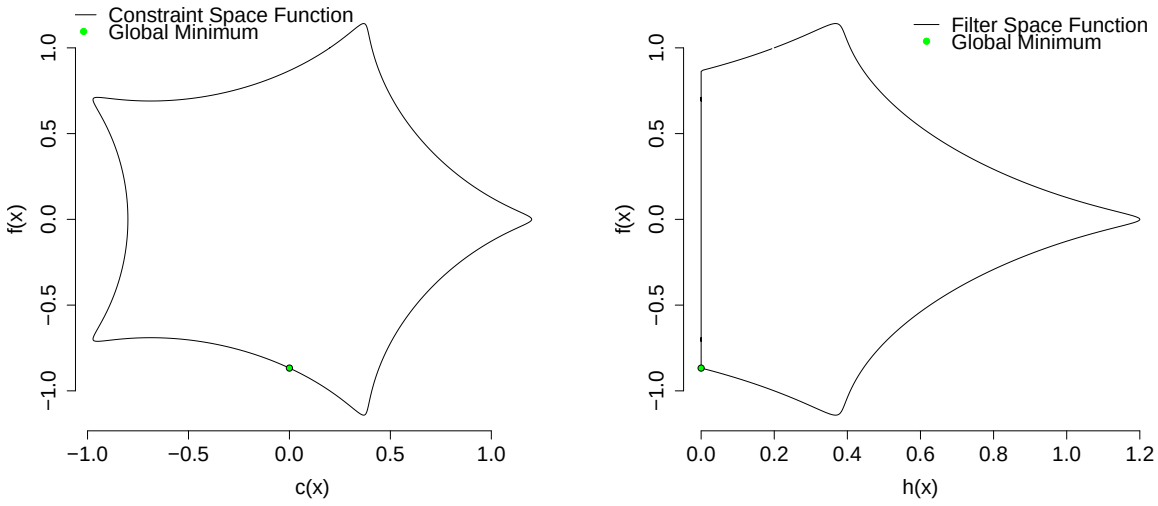


Figure 3.7: The constraint space (left) and filter space (right) for Problem 1.

Surrogate models, based on our PLMGP method, are built in the constraint space and then the actual filter steps are applied in the filter space. Although there appears to be no linear correlation between the objective and the constraint, in P1, (Pearson's correlation coefficient of $r = 0.0009$), there is still some structure between the objective and constraint functions in the constraint space that joint modeling may be able to capture. The constraint space in Figure 3.7 is mapped to the filter space by use of the feasibility measure h in (3.2).

The filter space is where the filter $\mathcal{F}$ is constructed, as well as where we solve our

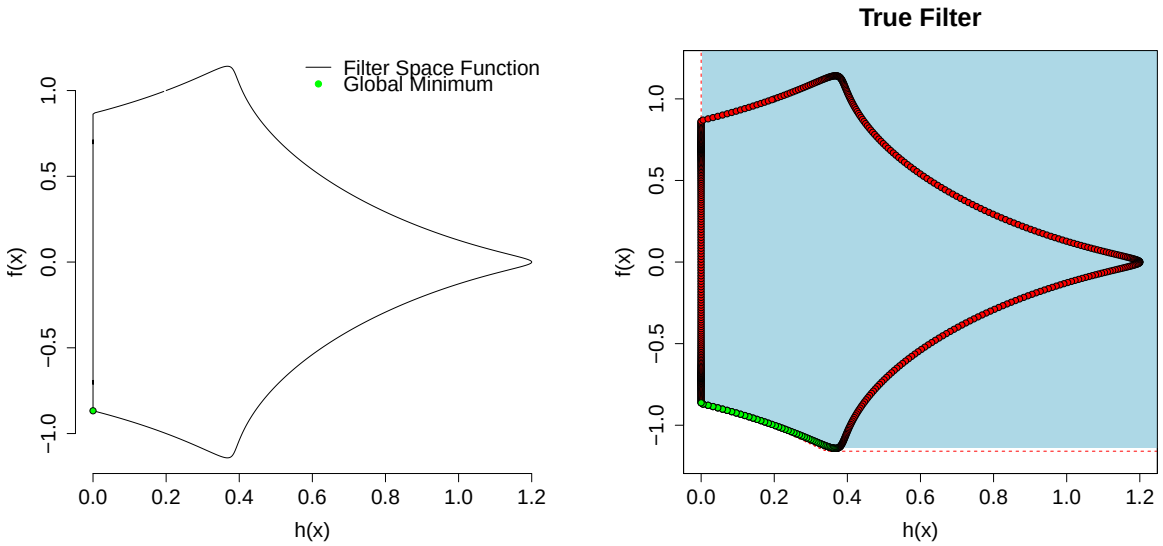


Figure 3.8: The true filter space (left) for Problem 1, and an example of a filter built from 1000 random inputs (right) for Problem 1. The red points denote output pairs $(h(\mathbf{x}), f(\mathbf{x}))$ not belonging to the filter $\mathcal{F}$, while green points represent output pairs that are part of the filter.

statistical filter algorithm subproblems. Recalling that filter methods take a biobjective approach to optimization, the aim of the subproblems is to advance the filter as far "south-west" as possible while driving the successive iterates towards feasibility, i.e., $h(x) = 0$. Clearly, from Figure 3.8, the input x that corresponds to $h(x) = 0$ in the filter $\mathcal{F}$, and has the smallest objective function value, is the current best solution to the constrained optimization problem (1.2).

We successfully solved P1 using our statistical filter algorithm, with respect to both the probability beyond the filter (PBF) and maximum expected area (MEA) subproblems. In both instances, we started by initializing our filter $\mathcal{F}$ with a sample of 10 points chosen using a Latin hypercube design on $[0, 10]$. Here, we place an upper bound on the maximum allowable constraint violation with $U = 2$. We then proceed to sequentially select 50

more points based on maximizing the PBF. For purpose of illustration, we step through a few iterations of the statistical filter algorithm while solving the PBF subproblem. As we should expect, from Figure 3.9, we see that as the number of successive iterations increases, the smaller our uncertainty about the predicted values (blue points) under our PLMGP surrogate model becomes. After initializing the filter $\mathcal{F}$, our uncertainty about the predicted values, based on our surrogate model, is at its highest. This large amount of uncertainty intuitively makes sense since, after initialization of the filter, we are trying to predict new outputs, at untried inputs, conditional on the smallest amount of data. As we update our filter, i.e., collect more data, our uncertainty in prediction decreases because we are conditioning on more knowledge about outputs in the constraint space. A feature of the Gaussian process, we note that uncertainty in our predictions should also be smaller when predicting near inputs where there already exists data. Gaussian processes rely on a correlation matrix that dictates how related inputs are in space, and so, our PLMGP method assumes that the outputs from inputs near each other should be more related, or correlated, than the outputs from inputs that are farther away. Hence, in Figure 3.9, we see that no matter what iteration of updating the filter we are at, the uncertainty around our predictions tends to be much lower where there already exists outputs than at places where we do not have much information. Interesting to note, in Figure 3.9 after 50 updates to the filter, we clearly see that our prediction of the upper left arm of the star shape is missing. Given the shape of the true filter in Figure 3.8, we should expect that our joint model would predict points throughout the space. However, the filter gives that region of the space little priority therefore not selecting any candidate points from this region. This

ultimately results in the joint model to have lowered predictive accuracy due to the small amounts of data in areas for the filter space.

A common feature of our statistical filter algorithm, as the number of iterations of the algorithm increases we see that the uncertainty around pairs of points in the filter tends to decrease much more rapidly than those outside it. We attribute this feature to the fact that as we observe more points in the region of the filter, our uncertainty becomes smaller the closer we are to previously observed points. Our statistical filter algorithm does a very good job at identifying pairs of points that should be added to the filter and so our predictive uncertainty tends to be higher around points that do not belong to the filter. This feature of our method is quite desirable in that it is a constant trade-off between searching locally where uncertainty is low, and searching globally where uncertainty is high.

Returning to solving P1, we ran our statistical filter algorithm utilizing both the PBF and MEA subproblem and find our optimal solution for P1 to be $f(x) = -0.86718$ occurring at $x = 5.25588$, which matches the true solution. Visually, from Figure 3.10, we see that the algorithm locks on to the optimal solution quite early, and only tends to select candidate inputs that would add pairs of points for inclusion to the filter. To illustrate the robustness of our method, we reran the statistical filter algorithm 30 times under different initial sample inputs from a LHD on $[0, 10]$. Table 3.1 shows that, on average, our statistical filter algorithm, under both the PBF and MEA subproblem, obtained the true global minimum for P1 after 50 iterations of the algorithm. Empirical convergence of our method was assessed by the fact that the average best feasible solution, and 95% credible intervals, converged towards the true solution as the number of updates to the

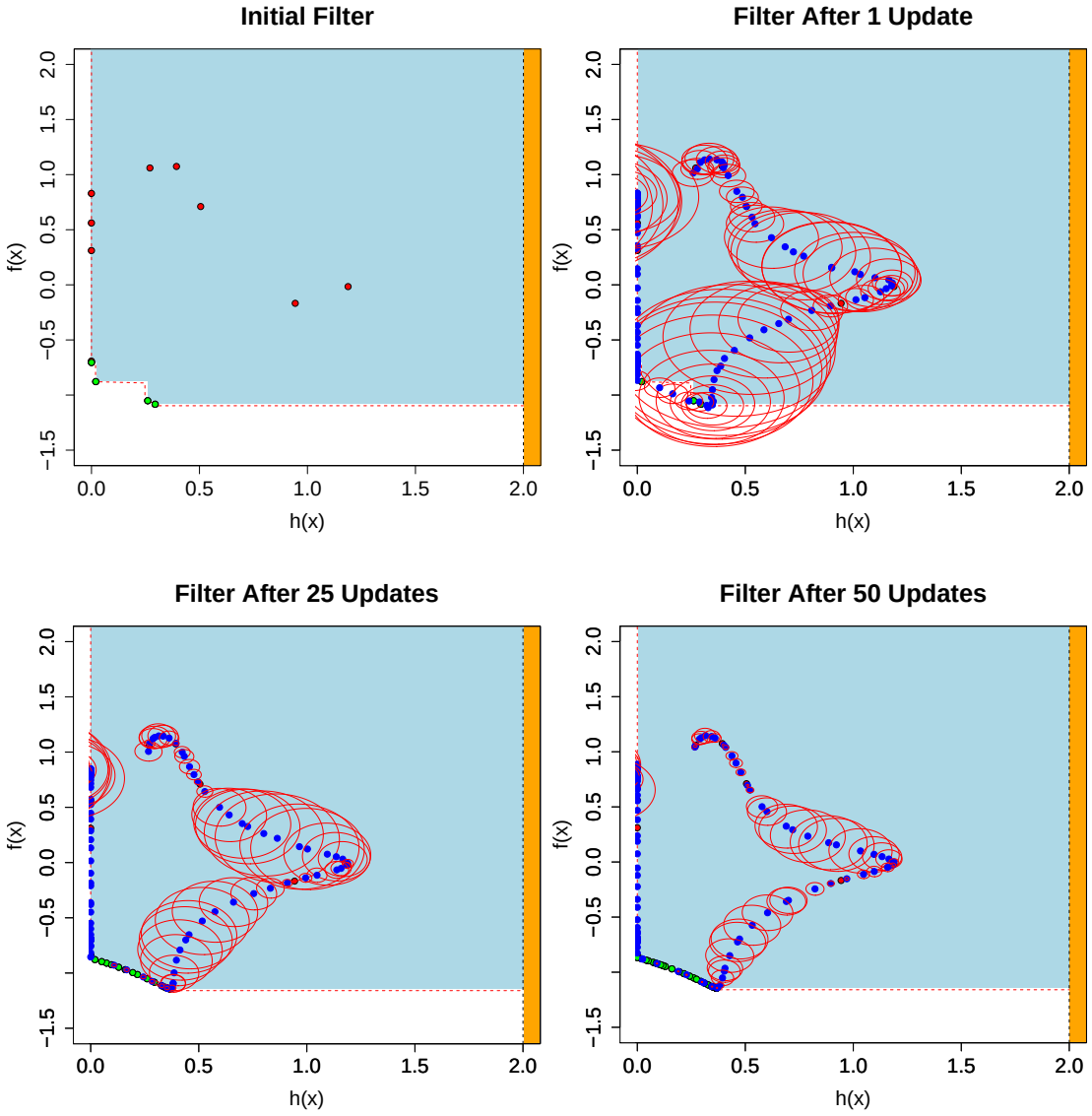


Figure 3.9: Progression of the filter space for Problem 1 under the PBF subproblem. Green points correspond to output pairs $(h(\mathbf{x}), f(\mathbf{x}))$ that belong to the filter $\mathcal{F}$, while red points correspond to output pairs that do not. The blue points and red ellipses are the predicted outputs and associated 95% probability ellipses under our PLGMP surrogate model.

algorithm increased (Figure 3.11).

In Section 3.2.3 we claim that it is advantageous to utilize the PLMGP model to build joint surrogate models for the objective and constraint functions rather than building independent ones based on PLGP models. In order to validate this claim, we also reran the statistical filter algorithm on P1, under the same conditions as before, but now we instead used independent surrogate models based on PLGP models rather than the joint models based on PLMGP. We assessed the performance and impact of independent modeling by rerunning the statistical filter algorithm 30 times under different initial sample inputs from a LHD on $[0, 10]$. We summarize the results of this exercise with Table 3.1 and Figure 3.11.

n		10	30	50
95%				
PBF	Independent	-0.76094	-0.84575	-0.86717
	Joint	-0.83574	-0.86432	-0.86717
MEA	Independent	-0.71076	-0.85332	-0.86717
	Joint	-0.83230	-0.86657	-0.86717
Average				
PBF	Independent	-0.83914	-0.86557	-0.86718
	Joint	-0.85676	-0.86687	-0.86718
MEA	Independent	-0.82160	-0.86157	-0.86718
	Joint	-0.85784	-0.86710	-0.86718
5%				
PBF	Independent	-0.86649	-0.86718	-0.86718
	Joint	-0.86668	-0.86718	-0.86718
MEA	Independent	-0.86558	-0.86713	-0.86718
	Joint	-0.86674	-0.86718	-0.86718

Table 3.1: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for P1. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using MEA performs the best for this problem after 50 additional evaluations.

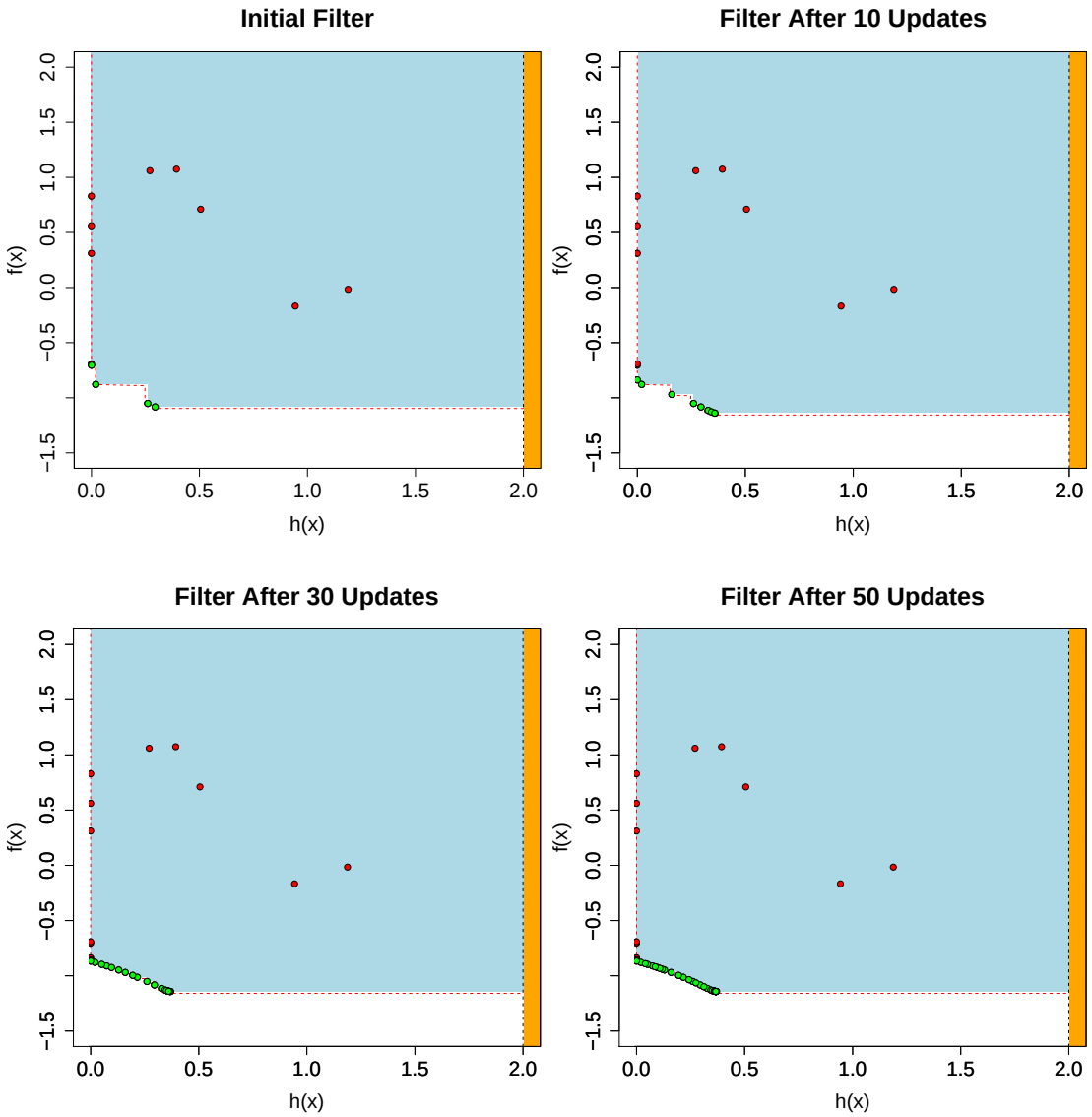


Figure 3.10: Progression of the filter space for Problem 1 under the PBF subproblem. Similar results hold for the MEA subproblem.

From both Table 3.1 and Figure 3.11 we see that the statistical filter algorithm, on average, found the optimal solution to P1 after 50 iterations for both joint and independent modeling. And although both methods found the optimal solution after 50 iterations, we can see visually from Figure 3.11, joint modeling tended to converge to the optimal solution much faster than did independent modeling. In practice, the functions that we are optimizing come from expensive black box computer experiments, and so, it is imperative that we work with algorithms that are able to seek out an optimal solution with fewer iterations. In the case of computer experiments, there may even be times when the computer models simply crash or end abruptly and so obtaining solutions that are much lower in fewer iterations is a desirable quality we seek.

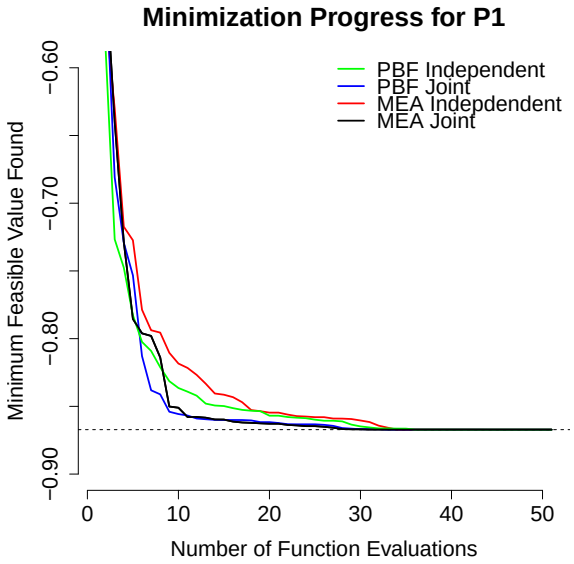


Figure 3.11: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent surrogate models for P1. The figure shows the average best feasible minimum over the 30 runs.

In P1, we saw our statistical filter algorithm did a good job of locating the optimal solution. Not surprisingly, using surrogate models built either from PLMGP or PLGP models converged to the same optimal solution in the end. However, it was the speed at which the two algorithms converged that highlights the additional benefit of joint modeling. Both models did a good job of finding the optimal solution because of the relatively easy input space and smooth well-behaved functions chosen. In the results that follow, we see that for simple functions this tended to be the case, although, as the number of constraints and/or inputs increased, the differences become appreciable.

For the rest of the test problems, P2–P6, we conducted similar analyses as in the case of P1, but omit the exhaustive illustrative discussion and focus on the performance and solutions of our statistical filtering algorithm.

Problem 2: P2

The simplest remaining problem we considered, P2, represents the case of a single input x and a single simple constraint $c(x)$. Here, we would like to minimize the objective, f, while simultaneously satisfying the constraint, c, i.e.,

$$\min_{x} \quad f(x) = 1 + (x-4)^2 \tag{3.8}$$

$$\text{s.t.} \quad c(x) = 1 - 0.5(x-3)^2 \leq 0 \tag{3.9}$$

$$x \in [0, 8]. \tag{3.10}$$

The optimal solution to P2 is $f(x) = 1.171522$ and occurs at $x = 3 + \sqrt{2}$ (Figure 3.12). Once again, the solution of P2 lies along the boundary of the constraint function. As before, we ran the statistical filter algorithm solving for both the PBF and MEA subprob-

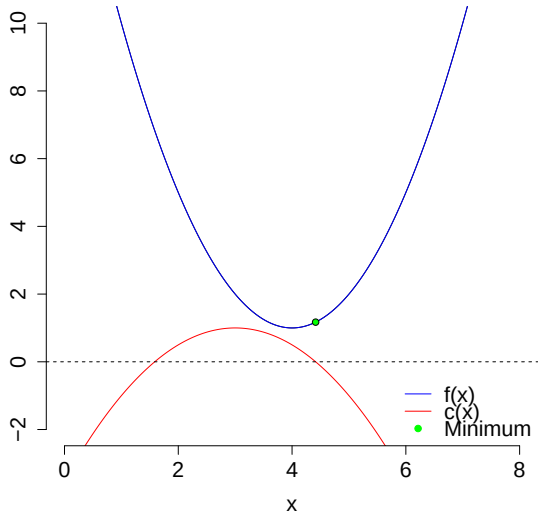


Figure 3.12: Problem 2: We want to minimize the objective, f, while satisfying the constraint, c. Values of the objective function (blue) are only feasible when the constraint function (red) is below the dashed line. The green dot corresponds to the global minimum solution to the problem and is found to lie along the constraint boundary. Making the problem more complicated, the feasible region for P2 is also disconnected.

lems and did so based on surrogate models built from joint PLMGP model and independent PLGP model. We first sampled 10 initial inputs from a LHD on $[0, 8]$ and then sequentially selected 50 candidate points from a LHD on $[0, 8]$ based on the appropriate subproblem. We reran this analysis 30 times under different initial sample inputs from a LHD on $[0, 8]$ and recorded the average best feasible solution (Figure 3.13) and 95% posterior intervals (Table 3.2).

n		10	30	50
95%				
PBF	Independent	1.6720	1.1777	1.1716
	Joint	1.3552	1.1775	1.1716
MEA	Independent	1.3848	1.1906	1.1716
	Joint	1.3552	1.1755	1.1716
Average				
PBF	Independent	1.2699	1.1722	1.1715
	Joint	1.2101	1.1720	1.1715
MEA	Independent	1.2435	1.1735	1.1715
	Joint	1.2033	1.1715	1.1715
5%				
PBF	Independent	1.1762	1.1715	1.1715
	Joint	1.1722	1.1715	1.1715
MEA	Independent	1.1735	1.1715	1.1715
	Joint	1.1726	1.1715	1.1715

Table 3.2: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for P1. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using PBF performs the best for this problem after 50 additional evaluations..

The general trend, as seen in P1, was that at the termination of the algorithm, all four different scenarios reached the optimal solution to P2. The use of joint surrogate modeling seemed to do slightly better in converging faster to the optimal solution than did

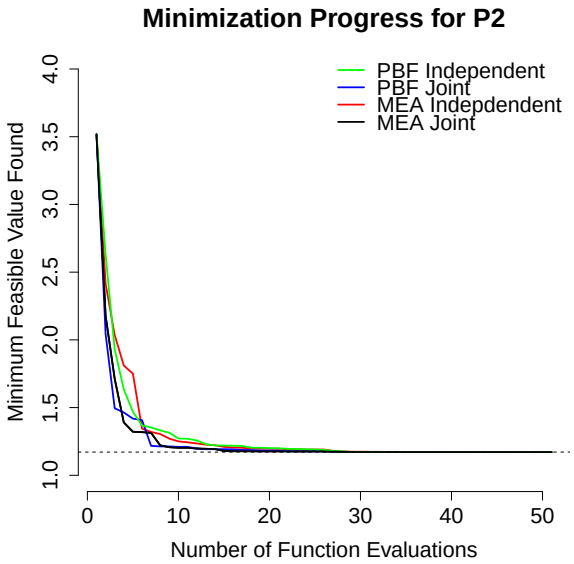


Figure 3.13: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent surrogate models for P2. The figure shows the average best feasible minimum over the 30 runs.

independent modeling, but on a problem as simple as this, the results for all four methods were almost indistinguishable.

Problem 3: P3

A harder problem (taken from Gramacy et al., 2015), this test problem represents the case of multiple inputs, x_1 and x_2, and a single, highly nonlinear constraint $c(\mathbf{x})$. Here, we would like to minimize the objective, f, while simultaneously satisfying the constraint, c, i.e.,

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = x_1 + x_2 \tag{3.11}$$

$$\text{s.t.} \quad c(\mathbf{x}) = 1.5 - x_1 - 2x_2 - 0.5\sin(2\pi(x_1^2 - 2x_2)) \leq 0 \tag{3.12}$$

$$x_1, x_2 \in [0, 1]. \tag{3.13}$$

The optimal solution to P3 is $f(\mathbf{x}) = 0.5998$ and occurs at $x_1 = 0.1954$ and $x_2 = 0.4044$. The solution of P3 lies along the boundary of the highly nonlinear constraint function (Figure 3.14).

Now that our input space is larger than the two previous problems we need to sample a larger initial set of inputs, and so, we sample 20 initial inputs from a LHD on $[0, 1]^2$ and then sequentially selected 60 candidate points from the same LHD based on the appropriate subproblem. We reran this analysis 30 times under different initial sample inputs from a LHD on $[0, 1]^2$ and recorded the average best feasible solution (Figure 3.15) and 95% posterior intervals (Table 3.3).

All four different scenarios reached the optimal solution of P3, with the joint surrogate modeling doing a better job in converging faster to the optimal solution than its

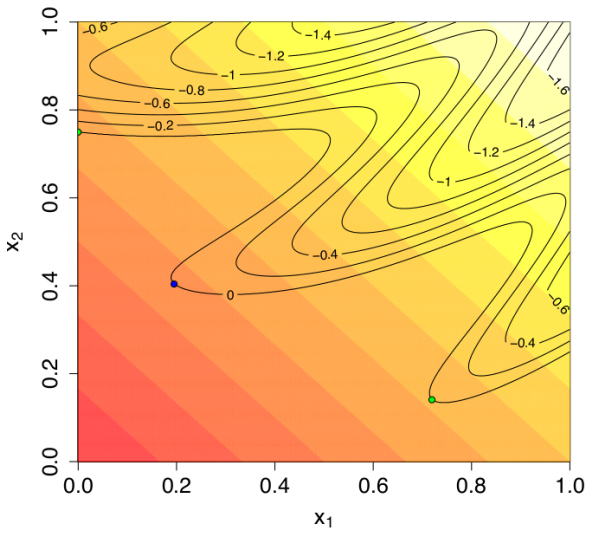


Figure 3.14: Problem 3: We want to minimize the objective, f, while satisfying the constraint, c. The objective function is a two-dimensional linear plane represented by the heat map where red and white corresponds to low and high values, respectively. The black contour curves define the region of the space where the constraint function is satisfied. Here, the infeasible region is all area to the left of the black contour curves. The blue dot corresponds to the global minimum of the problem, and the green dots correspond to two local minima. The global solution, as well as the two local solutions, to the problem is found to lie along the constraint boundary.

n		20	40	60
95%				
PBF	Independent	0.73142	0.63272	0.59987
	Joint	0.67042	0.59986	0.59984
MEA	Independent	0.82478	0.67151	0.59987
	Joint	0.65703	0.59998	0.59982
Average				
PBF	Independent	0.62587	0.60820	0.59981
	Joint	0.61492	0.59982	0.59980
MEA	Independent	0.64733	0.61345	0.59981
	Joint	0.61734	0.59984	0.59980
5%				
PBF	Independent	0.60141	0.60087	0.59980
	Joint	0.60081	0.59980	0.59980
MEA	Independent	0.60196	0.60089	0.59980
	Joint	0.60175	0.59980	0.59980

Table 3.3: After 60 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for P3. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using PBF performs the best for this problem after 60 additional evaluations.

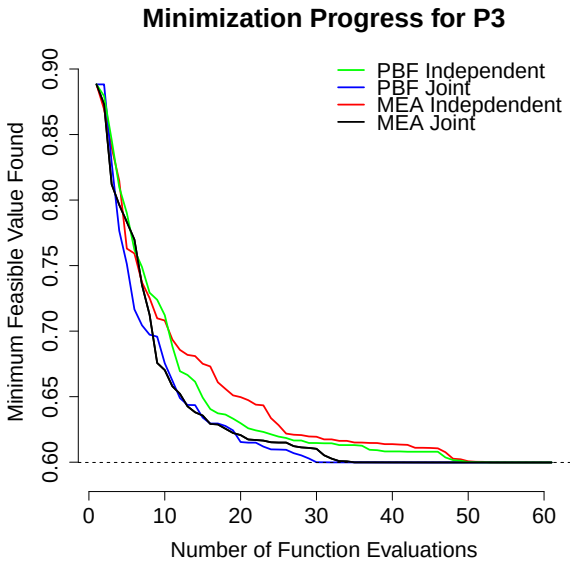


Figure 3.15: After 60 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent surrogate models for P3. The figure shows the average best feasible minimum over the 30 runs.

independent counterpart. In the joint case, it appeared that the PBF and MEA subproblems produced nearly indistinguishable results, while in the independent case the MEA subproblem seemed to do a worse job early on as compared to the PBF subproblem.

Problem 4: P4

This test problem, P4, represents the case of a single input x, and two nonlinear constraints $c_1(x)$ and $c_2(x)$. Here, we would like to minimize the objective, f, while simultaneously satisfying the constraints, c_1 and c_2, i.e.,

$$\min_x \quad f(x) = \cos\left(\frac{\pi x}{5}\right) + 0.2\sin\left(\frac{4\pi x}{5}\right) \tag{3.14}$$

$$\text{s.t.} \quad c_1(x) = \sin\left(\frac{\pi x}{5}\right) + 0.2\cos\left(\frac{4\pi x}{5}\right) \leq 0 \tag{3.15}$$

$$c_2(x) = \cos\left(\frac{\pi x}{5}\right) + 0.2\cos\left(\frac{4\pi x}{5}\right) \leq 0 \tag{3.16}$$

$$x \in [0, 10]. \tag{3.17}$$

The optimal solution to P4 is $f(x) = -0.8671835$ and occurs at $x = 5.25585$ (Figure 3.16).

We reran the prior analysis as before and first sampled 10 initial inputs from a LHD on $[0, 10]$ and then sequentially selected 50 candidate points from the same LHD based on the appropriate subproblem. Unsurprisingly, we found that using PBF versus MEA or joint versus independent did not make much of a difference in the final optimal solution found at the end of the iterations. All four combinations of the algorithm, on average, found the true solution (Table 3.4). However, on average, the statistical filter algorithm based on joint modeling and the MEA subproblem converged the fastest to the optimal solution (Figure 3.17), followed by the solution found from joint modeling and the PBF subproblem. Once again joint modeling was preferable to independent modeling.

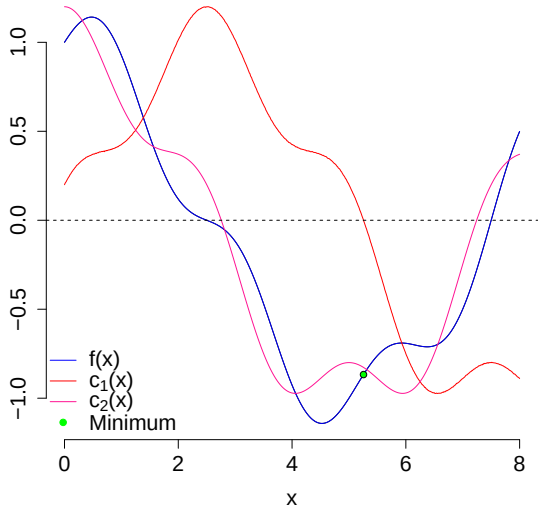


Figure 3.16: Problem 4: We want to minimize the objective, f, while satisfying the constraints, c_1 and c_2. Values of the objective function (blue) are only feasible when the constraint functions c_1 (red) and c_2 (pink) are below the dashed line. The green dot corresponds to the global minimum solution to the problem and is found to lie along the constraint boundary of $c_1(x) = 0$.

n		10	30	50
		95%		
PBF	Independent	-0.71541	-0.79150	-0.86715
	Joint	-0.74192	-0.84494	-0.86716
MEA	Independent	-0.70536	-0.82981	-0.86715
	Joint	-0.76704	-0.86434	-0.86717
		Average		
PBF	Independent	-0.80083	-0.84282	-0.86718
	Joint	-0.83765	-0.86000	-0.86718
MEA	Independent	-0.81051	-0.85041	-0.86718
	Joint	-0.84743	-0.86695	-0.86718
		5%		
PBF	Independent	-0.86558	-0.86558	-0.86718
	Joint	-0.86546	-0.86653	-0.86718
MEA	Independent	-0.86337	-0.86690	-0.86718
	Joint	-0.86674	-0.86718	-0.86718

Table 3.4: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for P4. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using MEA performs the best for this problem after 50 additional evaluations.

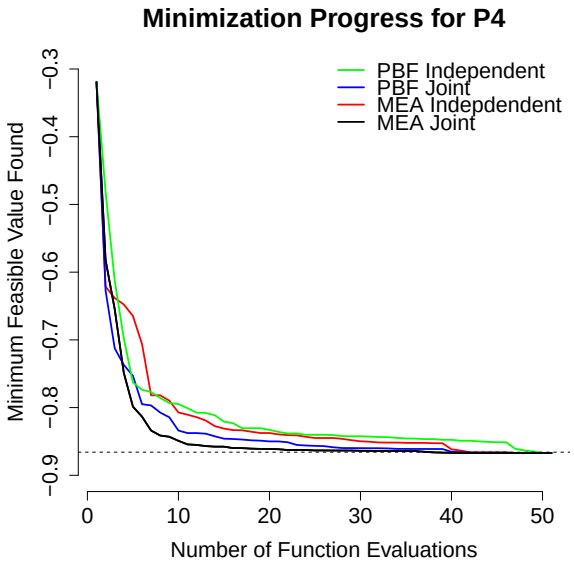


Figure 3.17: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent surrogate models for P4. The figure shows the average best feasible minimum over the 30 runs.

Problem 5: P5

A much harder example, this test problem, P5, comes from Floudas & Pardalos (1990) and represents the case of multiple inputs, x_1, x_2, x_3, x_4, and x_5, and a single constraint $c(\mathbf{x})$. Here, we would like to minimize the objective, f, while simultaneously satisfying the constraint, c, i.e.,

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = 42x_1 + 44x_2 + 45x_3 + 47x_4 + 47.5x_5 - 50\sum_{i=1}^{5} x_i^2 \tag{3.18}$$

$$\text{s.t.} \quad c(\mathbf{x}) = 20x_1 + 12x_2 + 11x_3 + 7x_4 + 4x_5 - 40 \leq 0 \tag{3.19}$$

$$\mathbf{x} \in [0,1]^5. \tag{3.20}$$

The optimal solution to P5 is $f(\mathbf{x}) = -17$ and occurs at $\mathbf{x} = (1, 1, 0, 1, 0)$. This problem was much harder to solve than P1–P4 because the solution to P5 lies exactly on the edge of the input space. This is a challenge because we are almost surely guaranteed to never sample a point exactly on the edge of the input space using a Latin hypercube design. In order to overcome this obstacle, we allowed ourselves to sample from a LHD that was slightly larger than the input space of $\mathbf{x}$, say for example $x_i \in [-0.05, 1.05]$ for $i = 1, \ldots, 5$, and in the event we sampled a value of x_i either less than 0 or greater than 1, we rounded to either 0 or 1 as appropriate. For clarity, if we sampled the input $\mathbf{x} = (1, 1.02, 0.3, -.01, 0.4)$ this would be adjusted to be $\mathbf{x} = (1, 1, 0.3, 0, 0.4)$. Sampling our inputs in this modified fashion allowed us to be able to obtain the edge case where the input $\mathbf{x}$ was located exactly on the edge of the input space.

In order to solve P5, we ran our statistical filter algorithm by first sampling 60 initial inputs from a LHD on the unit hypercube $[0, 1]^5$ and then sequentially selected 150

candidate points from a LHD on $[-0.05, 1.05]^5$ based on the appropriate subproblem. We reran this analysis 30 times under different initial sample inputs from a LHD on $[0, 1]^5$ and recorded the average best feasible solution (Figure 3.18) and 95% posterior intervals (Table 3.5).

n		50	100	150
		95%		
PBF	Independent	0.9645	-4.2586	-17.0000
	Joint	-2.4764	-17.0000	-17.0000
MEA	Independent	-2.0727	-2.42340	-17.0000
	Joint	-2.7603	-17.0000	-17.0000
		Average		
PBF	Independent	-6.1547	-7.3275	-17.0000
	Joint	-8.3218	-17.0000	-17.0000
MEA	Independent	-6.6835	-8.2955	-17.0000
	Joint	-8.1635	-17.0000	-17.0000
		5%		
PBF	Independent	-11.6561	-11.8828	-17.0000
	Joint	-12.7532	-17.0000	-17.0000
MEA	Independent	-12.2849	-14.1332	-17.0000
	Joint	-14.3213	-17.0000	-17.0000

Table 3.5: After 150 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for P5. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using MEA performs the best for this problem after 150 additional evaluations.

P5 is not only a challenging problem because the solution lies exactly at the edge of the input space, but moreover, P5 has many feasible solutions at the different edges of the input space that produce solutions that are near optimal. For example, recall that the optimal solution of -17 for P5 occurs at $\mathbf{x} = (1, 1, 0, 1, 0)$, however, other possible edge configurations, such as $\mathbf{x} = (1, 0, 1, 0, 1)$ or $\mathbf{x} = (1, 1, 0, 0, 1)$, produce feasible solutions that

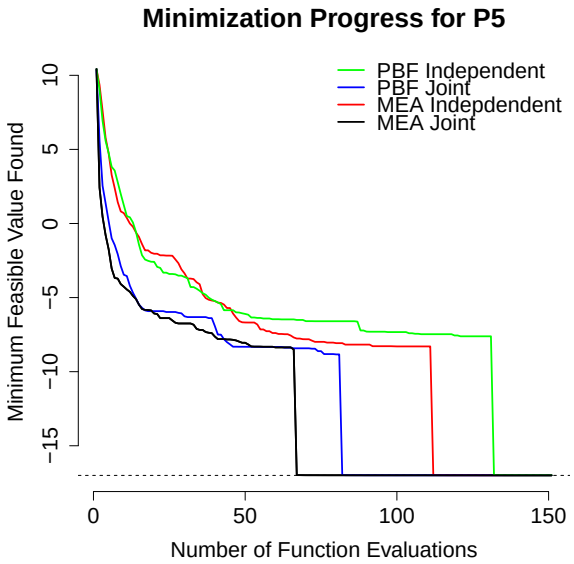


Figure 3.18: After 150 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent surrogate models for P5. The figure shows the average best feasible minimum over the 30 runs.

are close to the optimal solution, i.e., $f(1,0,1,0,1) = -15.5$ and $f(1,1,0,0,1) = -16.5$ respectively. Graphically, we cannot visualize the 5-dimensional input space but we can envision that these different feasible input configurations are along the edges and corners of a 5-dimensional hypercube that may be on opposite sides of the hypercube or not close to each other in the Euclidean sense. Thus, a method for solving P5 needs to be able to locate the edge solution while not getting stuck in local minima that may be far from the global minimum solution.

All four optimization scenarios, PBF independent/joint and MEA independent/joint, were able to find the global solution to P5. However, from Figure 3.18, it was clear that the algorithm that utilized the joint surrogate modeling framework performed much better than the independent models. In fact, the algorithm based on the joint models was able to converge to the global minimum before the 100^{th} iteration of the algorithm, whereas the independent model took substantially longer. In both instances, joint and independent, the statistical filter algorithms that solved the MEA subproblem appeared to converge to the global minimum much faster as compared to solving the PBF subproblem.

Problem 6: P6

The hardest synthetic problem we present in this chapter, test problem P6, represents the case of multiple inputs, $x_1, x_2, x_3, x_4, x_5, x_6, x_7$, and x_8, and two highly nonlinear constraints $c_1(\mathbf{x})$ and $c_2(\mathbf{x})$. Here, we would like to minimize the objective, f, while simul-

taneously satisfying the constraints, c_1 and c_2, i.e.,

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = \cos(\pi(x_1 + x_2x_3 + x_4)) + 0.2\sin\left(\frac{4\pi(x_5x_6 + x_7)}{x_8 + 1}\right) \tag{3.21}$$

$$\text{s.t.} \quad c_1(\mathbf{x}) = \sin(\pi(x_1 + x_2x_3 + x_4)) + 0.2\cos\left(\frac{4\pi(x_5x_6 + x_7)}{x_8 + 1}\right) \leq 0 \tag{3.22}$$

$$c_2(\mathbf{x}) = -\cos(\pi(x_1 + x_2x_3 + x_4)) + 0.2\cos\left(\frac{4\pi(x_5x_6 + x_7)}{x_8 + 1}\right) \leq 0 \tag{3.23}$$

$$\mathbf{x} \in [0, 1]^8. \tag{3.24}$$

The optimal solution to P6 is $f(\mathbf{x}) = -0.2828427$, which occurs at $x = (0.7909, 0.2670, 0.0798, 0.6426, 0.9161, 0.9377, 0.9645, 0.3893)$.

We ran the statistical filter algorithm solving for both the PBF and MEA subproblems and did so based on surrogate models built from both a joint PLMGP model and an independent PLGP model. We first sampled 80 initial inputs from a LHD on $[0, 1]^8$ and then sequentially selected 220 candidate points from a LHD on $[0, 1]^8$ based on the appropriate subproblem. We reran this analysis 30 times under different initial sample inputs from a LHD on $[0, 1]^8$ and recorded the average best feasible solution (Figure 3.19) and 95% posterior intervals (Table 3.6).

P6 was the most challenging synthetic problem due to the multiple constraints and large input space. As the number of inputs to the computer model increases, the harder the search for the optimal solution to the problem becomes. Both the PBF and MEA subproblems, under the joint surrogate model, were able find near optimal solutions to P6 while the independent models did not. Although trending towards the optimal solution of P6 (Figure 3.19), both independent models, on average, did not converge to the optimal solution for the total number of function evaluations allotted. P6 highlights the benefits

	n	50	150	220
		95%		
PBF	Independent	-0.091881	-0.097320	-0.107861
	Joint	-0.19613	-0.28055	-0.28208
MEA	Independent	-0.19082	-0.21673	-0.23412
	Joint	-0.10378	-0.28103	-0.28201
		Average		
PBF	Independent	-0.22548	-0.24420	-0.24924
	Joint	-0.24677	-0.28209	-0.28283
MEA	Independent	-0.23759	-0.25671	-0.26254
	Joint	-0.23837	-0.28230	-0.28284
		5%		
PBF	Independent	-0.27413	-0.28070	-0.28071
	Joint	-0.27765	-0.28282	-0.28284
MEA	Independent	-0.26902	-0.27647	-0.27670
	Joint	-0.27830	-0.28278	-0.28284

Table 3.6: After 220 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for P6. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best minima found. On average, the joint model using MEA performs the best for this problem after 150 additional evaluations.

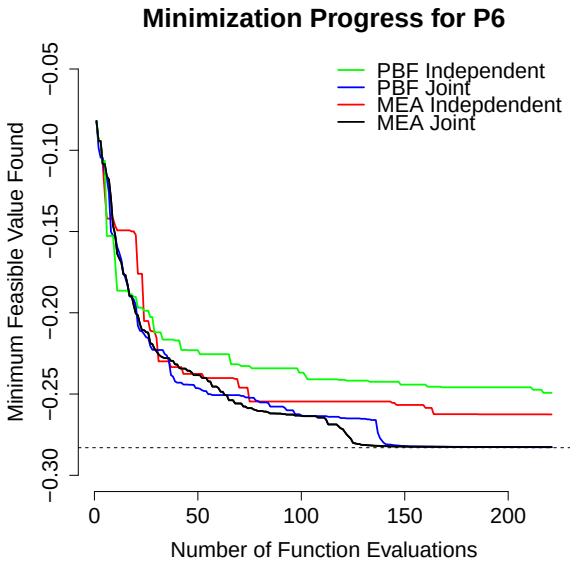


Figure 3.19: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent surrogate models for P6. The figure shows the average best feasible minimum over the 30 runs.

of joint modeling in that joint modeling tends to do a better job of predicting the outputs for the objective and constraint functions when the functions are complex and there is far fewer data available.

All six of the synthetic test problems showcased the fact that joint modeling of the objective and constraint functions lead to better results, as compared to independent modeling, in solving constrained optimization problems. The predictive accuracy of the joint models was usually much higher with fewer data points than the independent models and was ultimately the reason the joint models lead to faster convergence to the optimal solutions of the problems. As further evidence of the superiority of joint modeling over independent modeling, we summarize our findings for the synthetic test problems in Table 3.7. For each test problem we partition the number of additional function evaluations into three time points: beginning, middle, and end, based on the first, second, and third time point recorded in each problem's table of results. We note that in each test problem, the joint models always did better in the beginning and middle of running the statistical filter algorithm than did the independent models. Moreover, the joint models also did better more times than the independent models at the end time points as well for all test problems. The more interesting point however is that there appeared to be no clear cut joint model that did better when solving the two different subproblems. In most cases it appeared that the joint model solving the MEA subproblem tended to do better than the joint model solving the PBF subproblem (14 best iterative solutions found by the MEA subproblem as compared to 10 best iterative solutions found by the PBF subproblem), however, the reason for this still needs further investigation.

		Beginning	Middle	End	Total
Problem 1					
PBF	Independent	–	–	✓	1
	Joint	–	–	✓	1
MEA	Independent	–	–	✓	1
	Joint	✓	✓	✓	3
Problem 2					
PBF	Independent	–	–	✓	1
	Joint	–	–	✓	1
MEA	Independent	–	–	✓	1
	Joint	✓	✓	✓	3
Problem 3					
PBF	Independent	–	–	–	0
	Joint	✓	✓	✓	3
MEA	Independent	–	–	–	0
	Joint	–	–	✓	1
Problem 4					
PBF	Independent	–	–	✓	1
	Joint	–	–	✓	1
MEA	Independent	–	–	✓	1
	Joint	✓	✓	✓	3
Problem 5					
PBF	Independent	–	–	✓	1
	Joint	✓	✓	✓	3
MEA	Independent	–	–	✓	1
	Joint	–	✓	✓	2
Problem 6					
PBF	Independent	–	–	–	0
	Joint	✓	–	–	1
MEA	Independent	–	–	–	0
	Joint	–	✓	✓	2

Table 3.7: A summary, for all of the test problems, of the best iterative solution found based on choice of surrogate model and subproblem solved. Here, beginning, middle and end correspond to first, second, and third time point recorded in each problem's table of results. A check mark denotes which method, on average, found the smallest feasible value of the objective function at each time point.

3.4 Welded Beam Problem

A less artificial example, the welded beam problem (Coello Coello & Montes (2002); Hedar (2004)) has four inputs x_1, x_2, x_3, and x_4, and six constraints $c_1(\mathbf{x}), \ldots, c_6(\mathbf{x})$. The objective function, f, is the cost associated to construct a welded beam subject to constraints on sheer stress ($c_1(\mathbf{x})$, t), bending stress in the beam ($c_2(\mathbf{x})$, s), buckling load on the bar ($c_6(\mathbf{x})$, P_c), end deflection of the beam ($c_5(\mathbf{x})$, d), and side constraints ($c_3(\mathbf{x})$, $c_4(\mathbf{x})$). Here, we would like to minimize the objective, f, while simultaneously satisfying

the constraints, $c_1, \ldots, c_6$, i.e.,

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2) \tag{3.25}$$

$$\text{s.t.} \tag{3.26}$$

$$P = 6000,\ L = 14,\ E = 30 \times 10^6, G = 12 \times 10^6 \tag{3.27}$$

$$t_{\max} = 13600,\ s_{\max} = 30000,\ x_{\max} = 10,\ d_{\max} = 0.25 \tag{3.28}$$

$$M = P(L + x_2/2),\ R = \sqrt{0.25(x_2^2 + (x_1 + x_3)^2)} \tag{3.29}$$

$$J = \sqrt{2} x_1 x_2 (x_2^2/12 + 0.25(x_1 + x_3)^2) \tag{3.30}$$

$$P_c = \frac{4.013E}{6L^2} x_3 x_4^3 \left(1 - 0.25 x_3 \frac{\sqrt{E/G}}{L}\right) \tag{3.31}$$

$$t_1 = P/(\sqrt{2} x_1 x_2),\ t_2 = MR/J \tag{3.32}$$

$$t = \sqrt{t_1^2 + t_1 t_2 x_2 / R + t_2^2} \tag{3.33}$$

$$s = 6PL/(x_4 x_3^2) \tag{3.34}$$

$$d = 4PL^3/(E x_4 x_3^3) \tag{3.35}$$

$$c_1(\mathbf{x}) = (t - t_{\max})/t_{\max} \leq 0 \tag{3.36}$$

$$c_2(\mathbf{x}) = (s - s_{\max})/s_{\max} \leq 0 \tag{3.37}$$

$$c_3(\mathbf{x}) = (x_1 - x_4)/x_{\max} \leq 0 \tag{3.38}$$

$$c_4(\mathbf{x}) = (0.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0)/5.0 \leq 0 \tag{3.39}$$

$$c_5(\mathbf{x}) = (d - d_{\max})/d_{\max} \leq 0 \tag{3.40}$$

$$c_6(\mathbf{x}) = (P - P_c)/P \leq 0 \tag{3.41}$$

$$0.125 \leq x_1 \leq 10,\ 0.1 \leq x_i \leq 10 \text{ for } i = 2, 3, 4 \tag{3.42}$$

The optimal solution to the welded beam problem (WB), as reported by Hedar

(2004), is $f(\mathbf{x}) = 1.7250022$, which occurs at $\mathbf{x} = (0.2056, 3.4726, 9.0366, 0.2057)$.

One of the toughest parts of the welded beam problem is that the set of possible feasible points is very small as compared to the entire input space. Taking a Latin Hypercube sample (LHS) of size 1,000,000 yields only 972 feasible points, or rather, only 0.0972% of the design points in a LHD will be feasible for the welded beam problem. Having fewer than 1% of the initial set of inputs be feasible means that our PLMGP model needs to do a good job at prediction and uncertainty quantification of the predictions so that the filter method can direct the search towards areas where the chance of encountering feasible points is high.

To solve the WB problem, we start with an initial sample of 40 inputs from a LHD over the input space and sequentially sample 960 more inputs. Using a LHD to predict new outputs is an inefficient space filling design in the context of the WB problem, and so, because our LHD provides us with so few feasible inputs, we decided to follow the strategy of Taddy et al. (2009) and select the candidate set of inputs from a LHD of size 500 times the input dimension augmented by an additional 10% of the candidate locations taken from a smaller LHD bounded to within 5% of the domain range of the current best feasible point. Using the approach of Taddy et al. (2009) better ensures that our search should continue to predict at some feasible inputs once we have found at least one feasible input. Under these conditions, we ran our statistical filter algorithm using the joint PLMGP model and progressed the search for new inputs by solving both the PBF and MEA subproblem. Leading to near identical solutions, solving the MEA subproblem led to an optimal input configuration of $\mathbf{x} = (0.2057296, 3.470489, 9.036624, 0.2057296)$, which

yields a feasible value of 1.7248523. The solution found was only slightly better than the optimal solution reported in Hedar (2004) of 1.7250022.

We compared the performance of joint surrogate modeling to independent surrogate modeling by rerunning the analysis 30 times under different initial sample inputs from a LHD over the input space. We recorded the average best feasible solution, Figure 3.20, and its 95% posterior intervals, Table 3.8, for comparison.

n		300	600	960
95%				
PBF	Independent	1110.84	1107.9701	1.7385
	Joint	1111.61	1.7445	1.7361
MEA	Independent	1134.89	1120.9307	1.7424
	Joint	1097.26	1.7450	1.7356
Average				
PBF	Independent	844.28	423.5903	1.7314
	Joint	455.15	1.7354	1.7309
MEA	Independent	924.97	779.7948	1.7319
	Joint	413.64	1.7356	1.7308
5%				
PBF	Independent	1.7306	1.7284	1.7250
	Joint	4.4548	1.7372	1.7276
MEA	Independent	5.8341	4.5256	1.7249
	Joint	1.7334	1.7286	1.7276

Table 3.8: After 960 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for the welded beam problem. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using MEA performs the best for this problem for 960 addtional evaluations.

The joint surrogate models drastically outperformed the independent surrogate models in converging to the true optimal solution of the welded beam problem (Figure 3.20). Given the small percentage of feasible inputs, both the joint and independent models

took much longer to converge towards the optimal solution than seen in the prior synthetic test problems of Section 3.3. However, both models ended up converging (on average) quite closely to the optimal solution with the joint surrogate model beating the independent surrogate model by almost 300 iterations (Figure 3.20).

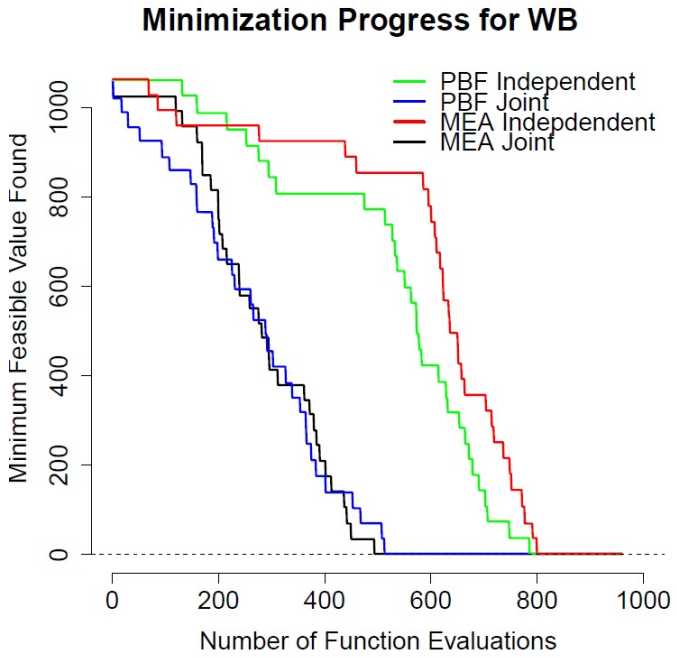


Figure 3.20: After 960 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent surrogate models for the welded beam problem. The figure shows the average best feasible minimum over the 30 runs.

3.5 Comparators

In Sections 3.3 and 3.4 we compared our statistical filter algorithm under joint and independent modeling of the objective and constraint functions. In all test problems, we

found that joint modeling was the better approach, as compared to independent modeling, where the statistical filter algorithm under the joint model always converged, empirically, to the optimal solution of the test problems. To further assess the practical usefulness of our statistical filter, we compared our method to three derivative free optimization (DFO) algorithms that are capable of handling the constrained optimization case. The three DFO comparators we used were constrained optimization by linear approximation (COBYLA; Powell (1994)), method of moving asymptotes (MMA; Svanberg (2002)), and sequential least squares quadratic programming (SLSQP; Kraft (1988)). COBYLA is an algorithm for derivative free optimization, with nonlinear inequality and equality constraints, where the algorithm works by constructing successive linear approximations of the objective function and constraints via a simplex of $n+1$ points (in n dimensions), and optimizes these approximations in a trust region at each step. The MMA algorithm works by generating a strictly convex approximating subproblem in each step of the iterative process. The generation of these subproblems is controlled by the so-called moving asymptotes, which both stabilize and speed up the convergence of the general process (Svanberg, 2002). Lastly, the SLSQP algorithm works by optimizing successive second-order (quadratic/least-squares) approximations of the objective function (via Broyden-Fletcher-Goldfarb-Shanno updates), with first-order (affine) approximations of the constraints.

We compared and contrasted our statistical filter algorithm to the three DFO algorithms by testing each DFO algorithm on all of the synthetic test problems, but for brevity, we only show the results for P1 and P6 of Section 3.3, and the welded beam problem. We implemented each of the three DFO algorithms using the `R nloptr` package

library (Ypma, 2014) and used the default hyper parameter values provided in `nloptr`. We compared the results of each DFO algorithm to the results already established by our statistical filter algorithm in Section 3.3 and 3.4. For each of the three comparative DFO algorithms we set the maximum number of iterative function evaluations (updates) to be the same number used previously by our statistical filter algorithm for each respective problem. Each of the DFO comparators required an initial starting input, and so, we always chose the initial input using a LHD from the appropriate domain of each problem. The initial starting input for each comparator was allowed to be either feasible or infeasible.

On the spectrum of difficulty of synthetic test problems, P1 was considered to be an easy test problem as it only contained one nonlinear constraint and one input. We chose to test the three DFO algorithms on P1 in order to compare how well our algorithm could stack up against established comparators for the "simple" test case scenario. We ran the three DFO comparators 30 times under different initial inputs from a LHD on $[0, 10]$ and recorded the average best feasible solution, Figure 3.21, and its 95% quantiles and posterior intervals, Table 3.9. In what follows, we simply refer to the statistical filter algorithm using the joint model, under both the PBF and MEA subproblems, as the PBF and MEA models, respectively.

Surprisingly, the simple case, P1, was actually much harder for the three comparative DFO algorithms to solve. On average, over the 60 iterations of each DFO algorithm, none of the comparators were able to converge to the optimal feasible solution of P1. In fact, all three DFO algorithms, on average, appeared to converge (or asymptote) to a much higher solution than those found by the statistical filter algorithm (Figure 3.21). This can

n	20	40	60
		95%	
PBF	-0.83574	-0.86432	-0.86717
MEA	-0.83230	-0.86657	-0.86717
COBYLA	0.69001	0.69001	0.69001
MMA	0.06491	0.05957	0.05957
SLSQP	0.69001	0.69001	0.69001
		Average	
PBF	-0.85676	-0.86687	-0.86718
MEA	-0.85784	-0.86710	-0.86718
COBYLA	-0.72562	-0.72562	-0.72562
MMA	-0.71842	-0.71904	-0.71904
SLSQP	-0.44908	-0.44908	-0.44908
		5%	
PBF	-0.85676	-0.86687	-0.86718
MEA	-0.86674	-0.86718	-0.86718
COBYLA	-0.86718	-0.86718	-0.86718
MMA	-0.86718	-0.86718	-0.86718
SLSQP	-0.86718	-0.86718	-0.86718

Table 3.9: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for P4. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found.

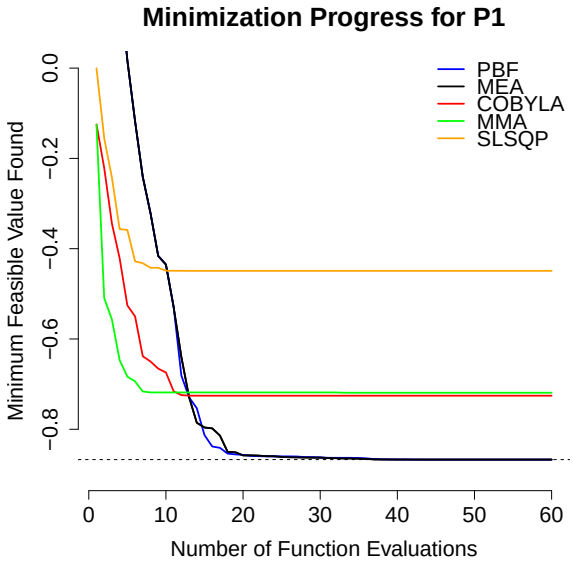


Figure 3.21: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the statistical filter algorithm (PBF and MEA) and the three DFO algorithms (COBYLA, MMA, and SLSQP) for P1. The figure shows the average best feasible minimum over the 30 runs.

be attributed to the fact that the average solution over the 30 runs for the three DFO comparators is highly susceptible to bad runs that converged to (or near) local minima. However, as we see by the lower 5% quantiles, Table 3.9, all of the DFO comparators were able to find the optimal solution to P1 at least once, and each of the DFO comparators were also able to do it much faster than the statistical filter algorithm.

One of the main disadvantages of the three DFO comparators is that they tend to get stuck in local minima. Each of the three DFO algorithms is very good at local search and so when the search begins near a stationary point the algorithms tend to converge quite quickly to it. However, when the stationary point is only a local minimum, the DFO algorithms tend to become stuck at that point, often declaring convergence prematurely, and not exploring the space globally. Thus, convergence to a global minimum is often contingent upon the comparators being started near the global minimum or far from local modes. In the feasible area, P1 has one global minimum and two local minima (Figure 3.22) and is responsible for the behavior of the solutions we see in Table 3.9 for the three comparators.

The values of the solutions for the three comparators in Table 3.9 tend to be either at a local minimum or averaged in between two local minima. Thus, being able to search the input space globally, as well as not get stuck in local minima, is a key success of our algorithm. Statistical surrogate modeling allows us to understand and quantify our uncertainty about the input space globally, while utilizing filter methods allows us to search well locally for solutions. The marriage of the two concepts is a powerful tool that the standard local search methods lack. Furthermore, in all 30 runs of the PBF and MEA

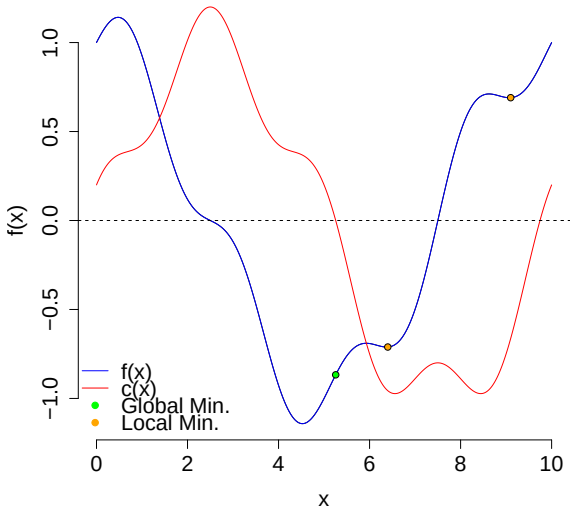


Figure 3.22: P1 has one global minimum and two local minima in the feasible region.

methods, each run found at least one feasible point over the 60 iterations. On the other hand, out of the 30 runs, there were 8 COBYLA runs, 11 MMA runs, and 10 SLSQP runs that did not find any feasible solutions. For a fair comparison we removed these runs from the prior analysis, however, it is worth noting that they did exist and moreover it highlights a superior aspect of the statistical filter algorithm in comparison.

Seeing how poorly the three comparators performed on a simple test case problem, we did not envision the comparators to do much better as the complexity of the problem increases. Still, comparing and contrasting our method with the comparators on a full range of different difficulties is an essential task. Testing the comparators on a harder problem, we applied each of the three DFO algorithms to P6, which was the hardest synthetic test problem in Section 3.3. P6 represented the case when there was more than one constraint, and multiple inputs. We ran the three DFO comparators 30 times under different initial

inputs from a LHD on $[0,1]^8$ and recorded the average best feasible solution (Figure 3.21) and 95% quantiles and posterior intervals (Table 3.9).

n	100	200	300
95%			
PBF	-0.08806	-0.20931	-0.28208
MEA	-0.07660	-0.23992	-0.28201
COBYLA	0.55907	0.23932	0.22039
MMA	0.80000	0.80000	0.80000
SLSQP	0.85264	0.53296	0.53296
Average			
PBF	-0.20790	-0.26479	-0.28283
MEA	-0.20154	-0.27330	-0.28284
COBYLA	-0.19576	-0.22608	-0.23347
MMA	-0.11287	-0.11287	-0.11287
SLSQP	-0.04981	-0.11867	-0.11867
5%			
PBF	-0.27604	-0.28231	-0.28284
MEA	-0.26490	-0.28225	-0.28284
COBYLA	-0.28280	-0.28284	-0.28284
MMA	-0.28284	-0.28284	-0.28284
SLSQP	-0.28284	-0.28284	-0.28284

Table 3.10: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for P4. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found.

Not deviating much from the results of P1, on average, the three DFO comparators performed rather poorly as compared to the PBF and MEA methods. In fact, on average, the PBF and MEA methods converged to the optimal solution of P6 while only the COBYLA method came close to reaching it. On average, COBYLA seemed to be getting closer to the optimal solution, and may have converged if run for longer, while MMA and SLSQP seemed to converge earlier to local minima (Figure 3.23). Once again, noting

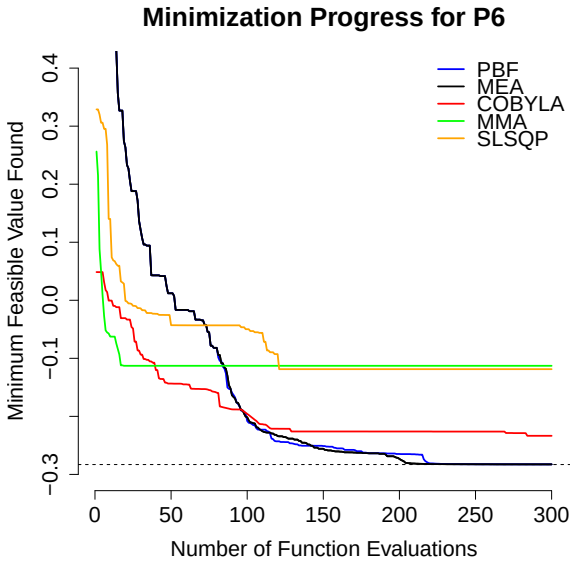


Figure 3.23: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the statistical filter algorithm (PBF and MEA) and the three DFO algorithms (COBYLA, MMA, and SLSQP) for P6. The figure shows the average best feasible minimum over the 30 runs.

the 5% quantiles of Table 3.10, each of the three comparators at least once converged to the optimal solution, and when they did, they did it at a much faster rate than the PBF and MEA methods. P6 echoes the fact that when started near a local minimum, the DFO comparators are quick to find a local minimum and do not explore the space globally. Likewise, out of the 30 runs of the DFO comparators, COBYLA had 6, MMA had 10, and SQP had 13 runs in which the entire run failed to find any feasible points. The fact that our PBF and MEA methods always find at least one feasible point is a desirable quality that the DFO comparators lack.

Lastly, we compared the performance of our PBF and MEA method to the DFO comparators using the welded beam (WB) problem. Recall that the WB problem resembled a more realistic real-world problem and had the additional challenge of not containing a large percentage of feasible points. Following the same strategy as before, we ran the three DFO comparators 30 times under different initial inputs from a LHD over the input space and recorded the average best feasible solution (Figure 3.24) and 95% quantiles and posterior intervals (Table 3.11).

Unlike the results for P1 and P6, the DFO comparators do a much better job in solving the welded beam problem. In fact, of the three comparators, the SLSQP algorithm clearly outperformed the PBF and MEA methods with all 30 runs of the SLSQP algorithm finding the optimal solution in fewer than 350 iterations (Table 3.11). In comparison, the SLSQP algorithm was able to converge to the true solution in more than half the time it took for the PBF and MEA algorithms to do so. Likewise, COBYLA and MMA also converged much faster to the optimal solution of WB in fewer than 350 iterations at least

n	350	700	1000
	95%		
PBF	1111.61	1.7392	1.7361
MEA	1097.26	1.7381	1.7356
COBYLA	157.85	2.5625	2.2842
MMA	3.7622	3.7622	3.7622
SLSQP	1.7249	1.7249	1.7249
	Average		
PBF	420.46	1.7340	1.7309
MEA	413.64	1.7339	1.7308
COBYLA	16.5037	1.8170	1.7758
MMA	1.9346	1.9346	1.9346
SLSQP	1.7249	1.7249	1.7249
	5%		
PBF	1.7306	1.7282	1.7249
MEA	1.7316	1.7276	1.7249
COBYLA	1.7249	1.7249	1.7249
MMA	1.7249	1.7249	1.7249
SLSQP	1.7249	1.7249	1.7249

Table 3.11: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for the welded beam problem. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found.

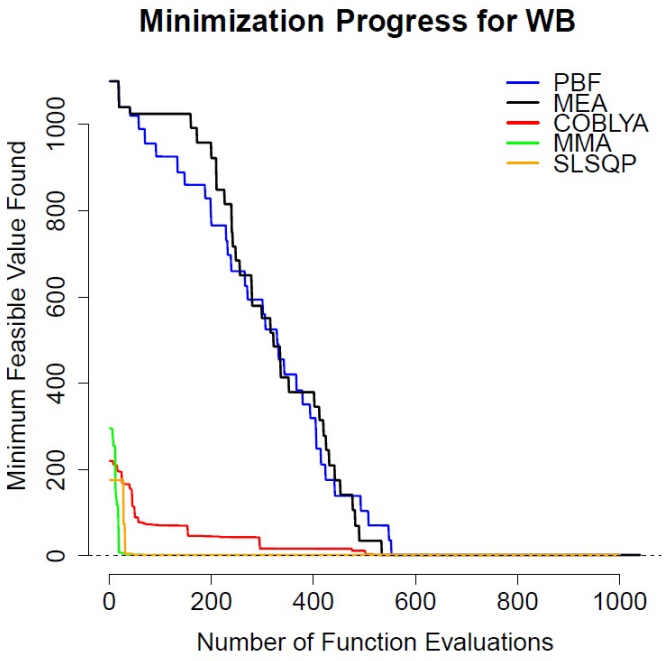


Figure 3.24: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the statistical filter algorithm (PBF and MEA) and the three DFO algorithms (COBYLA, MMA, and SLSQP) for the welded beam problem. The figure shows the average best feasible minimum over the 30 runs.

once (Table 3.11), although on average, these two algorithms did more poorly than the PBF and MEA methods. It would seem that once again premature convergence to possibly local minima were to blame for the poor average convergence results of the COBYLA and MMA algorithms. Interestingly, the DFO comparators did not seem to be troubled in this example by the lack of feasible points and were able to always find at least one feasible point much quicker than the PBF and MEA methods. The success of the DFO comparators can also be attributed to the fact that the WB's objective function is very smooth and unimodal. The DFO comparators do not rely on sampling the input space and so sequential descent steps are efficient when there is only one mode. On the other hand, the PBF and MEA methods rely on Latin hypercube samples as a means of generating sets of inputs to perform prediction at, and thus, is attributable to the reason why it takes much longer for the PBF and MEA methods to converge since we can only expect roughly 0.0972% of the points in our predictive set to be feasible.

In testing the three DFO comparators we have shown that they all possess an undesirable feature of getting stuck in local minima when a global solution is needed. The ability of the statistical filter algorithm to combine both global and local search is a powerful feature that differentiates it from the DFO comparators. Although convergence to a global minimum may take longer for the PBF and MEA methods when the objective function is flat or unimodal, as compared to the DFO comparators, the fact that the PBF and MEA methods consistently found the global minimum offsets this difference. However, performance of the statistical filter algorithm was generally better than the DFO comparators, and so, we feel comfortable in claiming that the statistical filter algorithm is superior to the

DFO comparators evaluated in regards to converging to the global minimum solution for constrained optimization problems of the form (1.2). We shall further evaluate this claim in Section 3.6 by testing the DFO comparators on a real-world computer experiment.

3.6 Pump-and-Treat Hydrology Problem

A real-world application, the pump-and-treat hydrology problem (Matott et al., 2011) involves a groundwater contamination scenario based on the Lockwood Solvent Groundwater Plume Site located near Billings Montana. Due to industrial practices in the area, two plumes (plume A and B) containing chlorinated contaminants have developed near the Yellowstone river (Figure 1.1). The two plumes are slowly migrating towards the Yellowstone river and of primary concern is keeping the chlorinated contaminants from leaking into the Yellowstone river. A pump-and-treat remediation is proposed, in which wells are placed to pump out contaminated water, purify it, and then return the treated water, at six locations (A_1, A_2, B_1, B_2, B_3 and B_4). A computer simulator was built to model this physical process where the inputs to the simulator are the pumping rates, $x_1, \ldots, x_6$ (which can be set between 0 and 20,000), for the six pump-and-treat wells and the output of the simulator is the cost of running the pump-and-treat wells and whether or not the plumes have been contained. Thus, the objective of the pump-and-treat hydrology problem is to minimize the cost of running the pump-and-treat wells while containing both plumes.

We reformulate this problem in the framework of a constrained optimization prob-

lem as

$$\min_{\mathbf{x}}\{f(\mathbf{x}) = \sum_{j=1}^{6} x_j : \mathbf{c}(\mathbf{x}) \leq 0,\, \mathbf{x} \in [0, 20 \cdot 10^4]^6\}, \tag{3.43}$$

where the objective f we wish to minimize is linear and describes the cost required to operate the wells. The two plumes are contained when the constraint, $\mathbf{c}$, is met. The time it takes to run the computer simulator is nontrivial and so it is not feasible to run the computer simulator at every possible combination of inputs and find the one that optimizes the problem (3.43). Instead, we proceed by applying our filter Algorithm 2 to solve for the optimal solution of the constrained optimization problem in (3.43). We choose to select candidate points based on solving the maximum expected area (MEA) and probability beyond the front (PBF) subproblems. Since the dimension of the input space is $d = 6$, we use our rule of thumb of $n = 10d = 60$ starting points to initialize the filter $\mathcal{F}$, from a LHD in $[0, 20000]^6$. We chose to use a sloping and a non-sloping envelope for the PBF and MEA subproblems, respectively, where we set $\beta = 0.95$ in both cases and let $\gamma = 0.05$ for the sloping envelope and $\gamma = 0$ for the non-sloping case. Additionally, we chose (arbitrarily) to place an upper bound of $U = 1,000$ on the constraint violation. We then proceed to sequentially select 440 more points based on maximizing the PBF and the MEA. At each of the 440 iterations we fit our joint PLMGP model for the objective and constraint functions with the PLGMP model using the same priors as Section 3.3 and $N = 4000$ particles. We follow the strategy of Taddy et al. (2009) and select the candidate set of inputs from a LHD of size 500 times the input dimension augmented by an additional 10% of the candidate locations taken from a smaller LHD bounded to within 5% of the domain range of the current best point.

The best solution our algorithm found is a cost of \$23,393 by setting the pumping rates to $(A_1, A_2, B_1, B_2, B_3, B_4) = (222, 5672, 13490, 2208, 1222, 580)$. Our solution was lower than that found in Lindberg & Lee (2015b) and Matott et al. (2011) of \$25,612 and \$23,714 respectively. In fact, according to Lindberg & Lee (2015b), of algorithms completing in 500 runs or less, the optimal cost found was also much higher than our solution at \$27,137. Additionally, rerunning the algorithm 30 times, the average best feasible minimum value found after 500 runs reached by our algorithm (\$23,502), Table 3.12, was much lower than the average best feasible minimum value found by nine competing optimization algorithms as reported in Gramacy et al. (2015).

We contrasted joint modeling to independent modeling by running the independent PLGP methodology 30 times under the same setup and conditions as the joint PLMGP methodology and found that both methods, over the 440 updates, lead to comparable optimal final solutions. However, it is clear from Figure 3.25 that joint modeling of the objective and constraint function did indeed lead to solutions that converged faster to an optimal solution than did independent modeling.

Lastly, we compared the performance of our statistical filter algorithm against the three derivative free optimization (DFO) comparators of Section 3.5 in solving the pump-and-treat hydrology problem. We ran the three DFO comparators 30 times under different initial inputs from a LHD on $[0, 20000]^6$ and recorded the average best feasible solution (Figure 3.26) and 95% quantiles and posterior intervals (Table 3.13).

Although the pump-and-treat problem has a very simple and flat objective function, the DFO comparators have a very hard time navigating the highly nonlinear, and

	n	150	300	440
		95%		
PBF	Independent	33202.91	28269.99	24554.38
	Joint	29515.83	26230.13	24506.18
MEA	Independent	32590.12	28500.84	24556.50
	Joint	29281.62	25535.59	24554.47
		Average		
PBF	Independent	31049.37	26749.41	23552.06
	Joint	27406.38	24649.97	23502.50
MEA	Independent	30468.24	27017.82	23555.88
	Joint	27110.29	23987.08	23552.58
		5%		
PBF	Independent	29995.04	26016.17	23450.54
	Joint	26342.30	23885.85	23400.96
MEA	Independent	29369.42	26215.70	23450.20
	Joint	26029.78	23472.18	23450.88

Table 3.12: After 440 updates: The table shows the average best feasible minimum over the 30 Monte Carlo repetitions with random initial conditions, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using PBF performs the best for this problem after 440 additional evaluations.

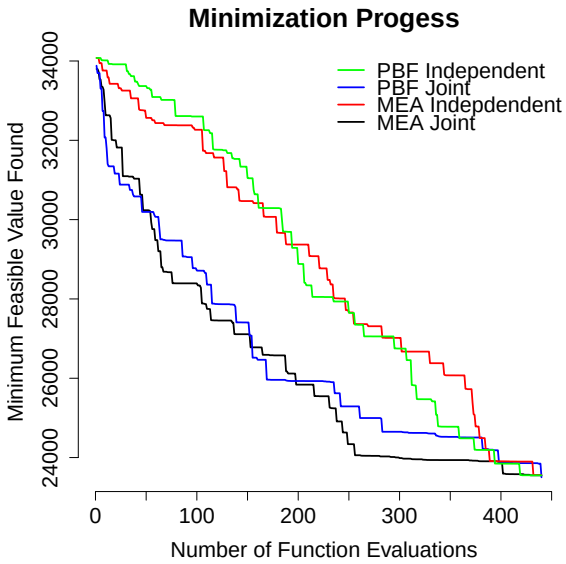


Figure 3.25: After 440 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the joint and independent models for the pump-and-treat hydrology problem. The figure shows the average best feasible minimum over the 30 runs.

n	150	300	500
		95%	
PBF	31418.14	27408.55	24506.18
MEA	30811.46	26750.99	24554.47
COBYLA	61048.09	32077.11	32052.22
MMA	68969.12	67317.69	67317.69
SLSQP	63365.13	58553.61	31542.80
		Average	
PBF	29058.67	25621.26	23502.50
MEA	28392.99	24910.76	23552.58
COBYLA	46976.67	28951.85	28838.67
MMA	52458.20	38617.13	38577.82
SLSQP	53657.24	35506.71	27295.09
		5%	
PBF	27868.03	24756.46	23400.96
MEA	27169.20	24006.54	23450.20
COBYLA	29421.68	27228.17	27218.27
MMA	30568.33	24418.86	24389.37
SLSQP	37068.28	26274.93	24029.45

Table 3.13: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for the pump-and-treat hydrology problem. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found.

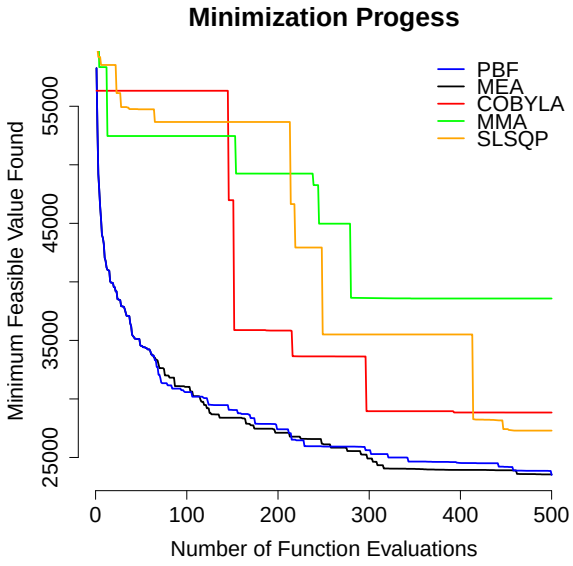


Figure 3.26: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the statistical filter algorithm (PBF and MEA) and the three DFO algorithms (COBYLA, MMA, and SLSQP) for the pump-and-treat hydrology problem. The figure shows the average best feasible minimum over the 30 runs.

possibly multimodal, constraint surface which can lead to multiple disconnected feasible regions. The statistical filter method, under both the PBF and MEA subproblem, does a much better job of finding a lower cost, on average, than does the DFO comparators over 500 runs of the computer model (Figure 3.26). In fact, the entire distribution of solutions, Table 3.13, is much lower for the statistical filter algorithm as compared to the DFO comparators. This real-world computer experiment once again highlights the fact that, if not started near the global solution, the DFO comparators tend to take longer to converge, or even become stuck in local modes. Undoubtedly better at converging towards the global minimum solution, the statistical filter algorithm validated our beliefs in the superiority of it over the DFO comparators in this real-world hydrology computer experiment.

3.7 Discussion

We have proposed a new approach for constrained optimization based on statistical models and filter methods. In particular, we have combined the particle learning multivariate Gaussian processes (PLMGP) model of Chapter 2 with flexible filter methods in order to solve problems in constrained optimization for expensive black box functions. In solving the filter method subproblem, we introduced two novel metrics, PBF and MEA, that both performed well in directing the filter to a global optimum. We validated our statistical filter algorithm on a suite of synthetic test problems as well as showed its applicability to a real-world computer experiment involving groundwater remediation.

One of the highlights of our statistical filter algorithm is the use of joint modeling of the objective and constraint functions. Through a series of experiments, we showed that

the joint model was able to do a much better job of converging to the global minimum of the constrained optimization problems as compared to independent modeling. It is not uncommon for the objective and constraint function to be negatively correlated, and the joint model was able to outperform the independent model by utilizing this fact. Likewise, as compared to three DFO comparators, the statistical filter method was a more robust method for finding the global minimum of the solutions. Being able to search globally as well as locally is a highlight of the statistical filter algorithm and underscores its usefulness when contrasted with the comparators.

Combining statistical models with filter methods proved to be a powerful tool for solving constrained optimization problems. In particular, the greatest novelty of Chapter 3 arose from the creation of the new subproblems responsible for selecting new candidate inputs. And although we did not augment nor change any particular feature of the filter, we think that there may be further avenues for improvement to it. What follows in Chapter 4 is a detailed exploration of what benefits there may or may not be in relaxing (or changing) some of the assumptions of the filter method.

Chapter 4

High Versus Low Dimensional Filters

From a statistical point of view, there is a substantial amount of valuable information lost in compressing the constraint values $\mathbf{c}(\mathbf{x})$ into a single nonnegative scalar $h(\mathbf{x})$. Under our statistical filters framework, we follow the accepted convention that we build our surrogate models in the constraint space and that we build the filter $\mathcal{F}$ in the two dimensional filter space. However, other than the fact that we saw little to no literature in the optimization community on filter methods that built the filter $\mathcal{F}$ in the higher dimensional constraint space, we see no apparent reason why this should not be explored.

In this chapter, we propose a method for constructing a multidimensional filter for solving the constrained optimization problem in (1.2). A natural extension of the work of Fletcher & Leyffer (2002), the multidimensional filter setting has not been readily explored. Gould et al. (2005a) and Gould et al. (2005b) investigate the use of a multidimensional

filter for solving systems of linear equations of the form $\mathbf{c}(\mathbf{x}) = \mathbf{0}$. However, finding solutions to these equations were of the utmost importance to the authors and the constrained optimization aspect of the filter method secondary. On the other hand, Shen et al. (2009) built a three dimensional filter for solving constrained optimization problems of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{c}_{\varepsilon}(\mathbf{x}) = 0 \\ & \mathbf{c}_{\mathcal{I}}(\mathbf{x}) \leq 0 \\ & \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{4.1}$$

where each component $f(\mathbf{x})$, and the feasibility measures $h_{\varepsilon}(\mathbf{x})$ and $h_{\mathcal{I}}(\mathbf{x})$ for $\mathbf{c}_{\varepsilon}(\mathbf{x})$ and $\mathbf{c}_{\mathcal{I}}(\mathbf{x})$, respectively, contributed to one of the dimensions of the three dimensional filter. Although there is additional benefit in extending the two dimensional filter to three dimensions, Shen et al. (2009) did not take potential advantage of building an even larger dimensional filter. It is here that we hope to extend and further explore the advantages of a higher multidimensional filter.

Clearly building a multidimensional filter $\mathcal{F}$ in the constraint space may break some already established convergence techniques, such as the envelope (Chin & Fletcher, 2003), since there will no longer be a guarantee that the current filter will avoid convergence to infeasible points, i.e., $h(\mathbf{x}) > 0$. However, there may be additional benefits yet to be discovered by building the filter $\mathcal{F}$ in the constraint space. On the other hand, we also argue that there may be additional benefits to building our surrogate models in the two dimensional filter space rather than the constraint space. For example, it would be much quicker/easier to build a single surrogate model for the feasibility measure $h(\mathbf{x})$ rather than

multiple constraint functions for $\mathbf{c}(\mathbf{x})$. Thus, for this chapter, we wish to explore the advantages (and disadvantages) of modeling $\mathbf{c}(\mathbf{x})$ versus $h(\mathbf{x})$ and the subsequent implications this will have on building the filter $\mathcal{F}$.

4.1 High Dimensional Filters

Introduced by Fletcher & Leyffer (2002), filter methods solve the constrained optimization problem (1.2) by taking a biobjective approach to optimization. Through the use of the aggregate constraint measure of feasibility

$$h(\mathbf{x}) = ||\max\{\mathbf{0}, \mathbf{c}(\mathbf{x})\}||_1 = \sum_{i=1}^{p-1} \max\{0, c_i(\mathbf{x})\} \tag{4.2}$$

the filter method transforms the original problem in (1.2) into the newly stated problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h(\mathbf{x}) = 0 \\ & \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{4.3}$$

The goal of the filter method is still to minimize the objective function f in (4.3), but now, emphasis is placed on satisfying the feasibility measure h rather than the constraint functions $\mathbf{c}$. Following the general filter algorithm in Section 3.1, Fletcher & Leyffer (2002) showed that global convergence results for the filter method could be established by solving a subproblem based on trust regions and sequential quadratic programming (SQP). Clearly, from an applied standpoint, filter methods have merit as a tool for solving constrained optimization problems. However, from a statistical point of view there is a loss in fidelity in the amount of information that is being processed by use of the feasibility measure h.

Akin to an identifiability issue in statistics, the feasibility measure h does not differentiate between which constraints are satisfied and which are not. The feasibility measure h is an aggregate statistic of constraint violation and so each constraint function contributes to building h but the feasibility measure h does not give us any indication of how much of a contribution each constraint contributes. For example, consider the case of the following constraint function, $\mathbf{c}(\cdot)$, where for some arbitrary inputs, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{X}$, we have that $\mathbf{c}(\mathbf{x}_1) = (1, 0, 0)$, $\mathbf{c}(\mathbf{x}_2) = (0, 0, 1)$ and $\mathbf{c}(\mathbf{x}_3) = (0, 0.5, 0.5)$. The identifiability issue becomes abundantly clear when calculating the feasibility measure h for each of the constraint functions. Here, $h(\mathbf{x}_1) = h(\mathbf{x}_2) = h(\mathbf{x}_3) = 1$, which tells us that at least one of the constraints were violated, but this feasibility measure h does not tell us which or how many constraints were violated, and it does not tell us by how much each of the constraints were violated or whether (and by how much) any constraint was satisfied. Clearly, in the example, $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$ each violate the constraint function in a different way and by a different amount, but to the feasibility measure h, each input has produced the same constraint violation in measuring feasibility. Intuitively, aggregating all of the constraint violation information into a single statistic seems like a waste of potentially useful information. And so, rather than compress all of the constraint violation information into a single feasibility measure h to build the filter with, we instead propose to construct one feasibility measure for each constraint function, i.e.,

$$h_i(\mathbf{x}) = ||\max\{\mathbf{0}, c_i(\mathbf{x})\}||_1 = \max\{0, c_i(\mathbf{x})\} \tag{4.4}$$

for $i = 1, \ldots, p-1$, and then proceed to build a new "higher-dimensional" filter based on f and the multiple feasibility measures h_i. The original filter algorithm of Fletcher & Leyffer

(2002) proposed solving the biobjective problem in (4.3), but we extend this approach to solving the following multiobjective problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h_i(\mathbf{x}) = 0 \text{ for all } i = 1, \ldots, p-1 \\ & \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{4.5}$$

In order to solve (4.5), using the filter approach, we need to redefine the concept of dominance. We borrow from the multiobjective optimization literature and say that a point $\mathbf{x}_i \in \mathcal{X}$ dominates a point $\mathbf{x}_j \in \mathcal{X}$ if and only if $f(\mathbf{x}_k) \leq f(\mathbf{x}_j)$ and $h_i(\mathbf{x}_k) \leq h_i(\mathbf{x}_j)$, with $(f(\mathbf{x}_k), h_i(\mathbf{x}_k)) \neq (f(\mathbf{x}_j), h_i(\mathbf{x}_j))$, for all $i = 1, \ldots, p-1$. Aligning with the original definition of the filter, we define the filter, denoted $\mathcal{F}$, as the set of all p-tuples $(h_1(\mathbf{x}_i), \ldots, h_{p-1}(\mathbf{x}_i), f(\mathbf{x}_i))$ such that no p-tuple dominates another p-tuple. The rest of the filter algorithm would then proceed in the same fashion as Algorithm 1, but we state the newly updated filter algorithm in Algorithm 3 for clarity. Clearly when $p = 2$, i.e., the case of only one constraint, we are back in the original filter problem of Fletcher & Leyffer (2002) and so for the rest of this chapter we assume that we are dealing with constrained optimization problems where $p > 2$.

Initialize the filter $\mathcal{F}$;

while *not terminated* **do**

Solve a subproblem to obtain a candidate point $\mathbf{x}_*$;

Evaluate $f(\mathbf{x}_*)$ and $c_1(\mathbf{x}_*), \ldots, c_{p-1}(\mathbf{x}_*)$;

if $(h_1(\mathbf{x}_*), \ldots, h_{p-1}(\mathbf{x}_*), f(\mathbf{x}_*))$ *is acceptable to* $\mathcal{F}$ **then**

Add $(h_1(\mathbf{x}_*), \ldots, h_{p-1}(\mathbf{x}_*), f(\mathbf{x}_*))$ to $\mathcal{F}$;

Remove any entries in $\mathcal{F}$ dominated by $(h_1(\mathbf{x}_*), \ldots, h_{p-1}(\mathbf{x}_*), f(\mathbf{x}_*))$;

end

Check for termination;

end

Algorithm 3: Upated generic filter method

Similar to our method, Shen et al. (2009) realized early on that each constraint may have its own behavior. For example, some constraints may be highly nonlinear, while some others are nearly linear. And so, Shen et al. (2009) proposed a new filter line search SQP method in which the violations of equality and inequality constraints were considered separately. Here, they constructed a three dimensional filter composed of all non-dominated triplets $(f(\mathbf{x}), h_\varepsilon(\mathbf{x}), h_\mathcal{I}(\mathbf{x}))$, where

$$h_\varepsilon(\mathbf{x}) = \sum_{i=1}^{k} |c_i(\mathbf{x})|, \quad \text{and} \quad h_\mathcal{I}(\mathbf{x}) = \sum_{j=k+1}^{p-1} \max\{0, c_i(\mathbf{x})\}. \tag{4.6}$$

However, Shen et al. (2009) was still severely limited in their analysis in that they still compressed a lot of valuable information about each individual constraint into only two feasibility measures.

From an algorithmic point of view, understanding the difference between Algo-

rithm 1 and Algorithm 3 is not very difficult nor is the implementation. However, conceptually the difference between Algorithm 1 and Algorithm 3 is best conveyed visually (Figure 4.1).

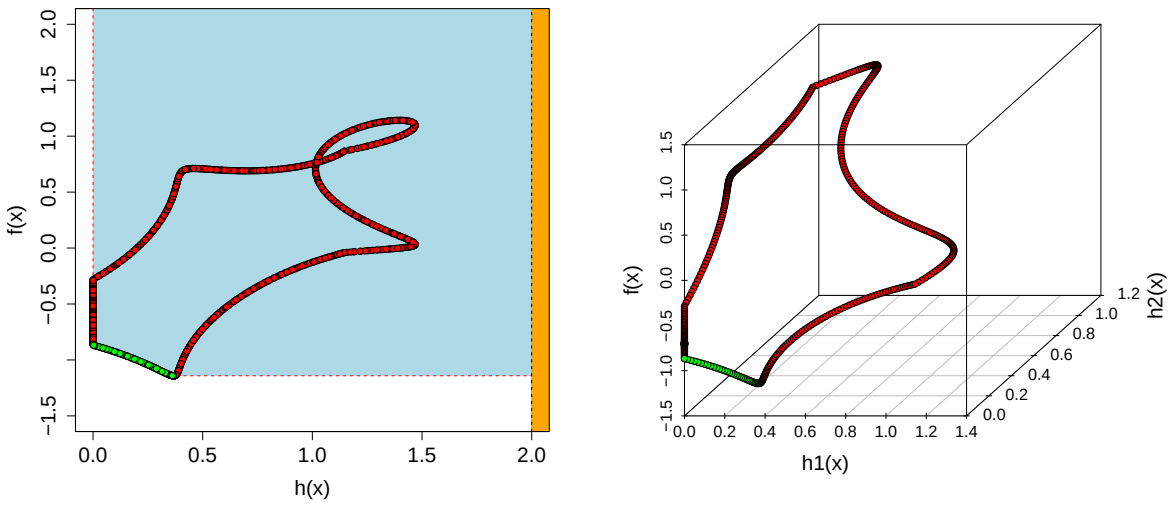


Figure 4.1: The filter space based on the original filter Algorithm 1 (left) and the newly updated filter Algorithm 3 (right). Green circles correspond to points in the filter while red circles correspond to points that are not.

Figure 4.1 represents the old and new filter space for synthetic test problem four (P4) of Section 3.3 based on Algorithm 1 and Algorithm 3, respectively. Important to note, in adapting the filter algorithm to have multiple feasibility measures we have not affected the input space at all and thus there are no new concerns over obtaining good space filling designs for the inputs. The added complexity of introducing multiple feasibility measures only shows up in the filter space where we must now solve a harder subproblem than before. We will still use the probability beyond the filter (PBF) subproblem for picking new candidate points $\mathbf{x}_*$, but first amend it to reflect the fact that we now utilize multiple

feasibility measures. Thus, we define the updated PBF criteria as selecting a $\mathbf{x}_*$ such that

$$\mathbf{x}_* = \max_{\mathbf{x}\in\mathcal{X}} \Pr\{(h_1(\mathbf{x}), \ldots, h_{p-1}(\mathbf{x}), f(\mathbf{x})) \text{ is acceptable to the filter } \mathcal{F}\}. \tag{4.7}$$

The added complexity in using (4.7) as compared to (3.3) is that the probability calculation no longer corresponds to calculating the area of an ellipse, but instead, we are now calculating the volume of a p-dimensional ellipsoid. This p-dimensional ellipsoid arises from the fact we can obtain a best prediction for $(f(\mathbf{x}), \mathbf{c}(\mathbf{x}))$ from (2.13), and quantify the uncertainty around that prediction with a probability ellipsoid based on (2.14). Solving the integral associated with (4.7) in two dimensions analytically is not a simple problem, and as the dimension of p increases, the problem becomes quite intractable. However, as before, we can simply use Monte Carlo integration to approximate the probability in (4.7).

We present here the newly updated statistical filter algorithm and discuss some of the limitations of it:

Sample initial inputs from a LHD;

Initialize the filter $\mathcal{F}$;

while *not terminated* **do**

Fit surrogate models for $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$ using the joint PLMGP model;

Map the surrogate model in the constraint space to the filter space;

Solve the PBF subproblem to obtain a candidate point $\mathbf{x}_*$;

Evaluate $f(\mathbf{x}_*)$ and $\mathbf{c}(\mathbf{x}_*)$;

if $(h_1(\mathbf{x}_*), \ldots, h_{p-1}(\mathbf{x}_*), f(\mathbf{x}_*))$ *is acceptable to* $\mathcal{F}$ **then**

Add $(h_1(\mathbf{x}_*), \ldots, h_{p-1}(\mathbf{x}_*), f(\mathbf{x}_*))$ to $\mathcal{F}$;

Remove any entries in $\mathcal{F}$ dominated by $(h_1(\mathbf{x}_*), \ldots, h_{p-1}(\mathbf{x}_*), f(\mathbf{x}_*))$;

end

Check for termination;

end

Algorithm 4: Updated statistical filter method

In our algorithm 4 we explicitly state to the solve the PBF subproblem and do not suggest solving the MEA subproblem. Although possible to solve, the metric associated with the MEA subproblem is much harder to calculate in higher dimensions as well as to conceptualize. The MEA subproblem in two dimensions was very simple to calculate because in two dimensions the problem came down to summing up the area of many rectangles where as in the higher dimensional case it would amount to summing up the volumes of many different types of polygons. Along the same lines as the MEA subproblem, we no longer have an envelope to calculate as we did in the original filter algorithm due to the

difficulty in specifying a well posed solution to what the envelope should look like in higher dimensions. Gould et al. (2005a), Gould et al. (2005b) and Shen et al. (2009) each had their own criteria for what an envelope should look like in higher dimensions. However, none of these options were appealing choices to us given the nature of our new statistical filter algorithm. This lack of an envelope is indeed a limitation of the new Algorithm 4 because we are no longer guaranteed to not get stuck selecting candidate points that are arbitrarily close to the current filter. We can however place upper bounds, U_i, on the each of the feasibility measures h_i that act as intersecting hyperplanes in the filter space.

4.2 Low Dimensional Filters

In opposition to Section 4.1, we explore the potential benefit of getting rid of the constraint space and only dealing with the original filter space of Fletcher & Leyffer (2002). A simpler idea, we propose to solve the constrained optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h(\mathbf{x}) = 0 \\ & \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{4.8}$$

using the original constraint measure of feasibility

$$h(\mathbf{x}) = ||\max\{\mathbf{0}, \mathbf{c}(\mathbf{x})\}||_1 = \sum_{i=1}^{m} \max\{0, c_i(\mathbf{x})\}, \tag{4.9}$$

however, we fit our joint PLMGP model to $(h(\mathbf{x}), f(\mathbf{x}))$ in the filter space rather than to $(\mathbf{c}(\mathbf{x}), f(\mathbf{x}))$ in the constraint space. We do not actually envision this to be a successful endeavor, however, we still undertake this experiment to gain empirical evidence of the

potential advantages and disadvantages of the idea. Important to note, once again, when $p = 2$, i.e., the case of only one constraint, we are back in the original filter problem of Fletcher & Leyffer (2002) and so for the rest of this chapter we assume that we are dealing with constrained optimization problems where $p > 2$.

Some of the potential benefits are easy to see right away in that joint surrogate modeling only consists of fitting the joint PLMGP model to the two dimensional outputs from $(h(\mathbf{x}), f(\mathbf{x}))$, rather than fitting the model to the, possibly high, p dimensional output $(\mathbf{c}(\mathbf{x}), f(\mathbf{x}))$. Thus model fitting becomes a much simpler and faster task under the proposed framework. A potential downside though is a loss in predictive accuracy due to the fact that we are compressing the constraint functions into a single feasibility measure h and then fitting a model to this lower fidelity output. Even worse, the PLMGP model, which places support on the entire real line for the feasibility measure h, would be inappropriate for modeling h at the boundary (i.e., $h(\mathbf{x}) = 0$) which is the area of interest where the feasibility measure, and ultimately the constraint, is met. However, with all this in mind, we are still able to adapt our statistical filter algorithm to model the feasibility measure rather than the constraints and thus summarize our new statistical filter algorithm as follows:

Sample initial inputs from a LHD;

Initialize the filter $\mathcal{F}$;

while *not terminated* **do**

  Fit surrogate models for $f(\mathbf{x})$ and $h(\mathbf{x})$ using the joint PLMGP model;

  Solve a subproblem to obtain a candidate point $\mathbf{x}_*$;

  Evaluate $f(\mathbf{x}_*)$ and $h(\mathbf{x}_*)$;

  if $(h(\mathbf{x}_*), f(\mathbf{x}_*))$ *is acceptable to* $\mathcal{F}$ **then**

    Add $(h(\mathbf{x}_*), f(\mathbf{x}_*))$ to $\mathcal{F}$;

    Remove any entries in $\mathcal{F}$ dominated by $(h(\mathbf{x}_*), f(\mathbf{x}_*))$;

  end

  Check for termination;

end

Algorithm 5: Updated statistical filter method

A key advantage of Algorithm 5 is that it still retains the capability to solve both PBF and MEA subproblems as before as well as all of the additional properties of the filter method, such as the envelope. In practical application we saw no difference between using the PBF and MEA subproblems for Algorithm 5, and so, for the rest of Chapter 4 we only use Algorithm 5 with the PBF subproblem.

4.3 Synthetic Test Problems

In order to test the new statistical filter algorithms in Sections 4.1 and 4.2, we use the synthetic test problems of Chapter 3 where $p > 2$. Thus, we examine Algorithms

4 and 5 on synthetic test problems P4 and P6. Where appropriate we use the exact same correlation function, priors, and starting parameters as detailed in Section 3.3.

Recall the fourth test problem, P4, in Chapter 3. P4, represents the case of a single input x, and multiple nonlinear constraints $c_1(x)$ and $c_2(x)$. Here, we would like to minimize the objective, f, while simultaneously satisfying the constraints, c_1 and c_2, i.e.,

$$\min_x \quad f(x) = \cos\left(\frac{\pi x}{5}\right) + 0.2\sin\left(\frac{4\pi x}{5}\right) \tag{4.10}$$

$$\text{s.t.} \quad c_1(x) = \sin\left(\frac{\pi x}{5}\right) + 0.2\cos\left(\frac{4\pi x}{5}\right) \leq 0 \tag{4.11}$$

$$c_2(x) = \cos\left(\frac{\pi x}{5}\right) + 0.2\cos\left(\frac{4\pi x}{5}\right) \leq 0 \tag{4.12}$$

$$x \in [0, 10]. \tag{4.13}$$

The optimal solution to P4 is $f(x) = -0.8671835$ and is obtained at $x = 5.25585$.

We first sampled 10 initial inputs from a LHD on $[0, 10]$ and then sequentially selected 50 candidate points from the same LHD based on Algorithm 2, 4, or 5 and the appropriate subproblem. To distinguish the results of running each different algorithm, we denote the results corresponding to the original Algorithm 2 as PBF and MEA, under the PBF and MEA subproblems respectively, and the results of running Algorithm 4 as "HDF" (high dimensional filter) and the results of running Algorithm 5 as "LDF" (low dimensional filter). As somewhat expected, the filter algorithm associated with LDF did not converge to the true solution although the algorithm does come close to it. We attribute the fact that LDF stayed the same (on average) from iteration 30 to 50 (Table 4.1) due to the PLMGP model having a hard time predicting values where the feasibility measure is zero since the PLMGP model places positive support on the entire real line. Additionally, there should

be a loss in fidelity in the predictive accuracy of LDF due to the extra lost information in not modeling c_1 and c_2. Interestingly, HDF (on average) did just as well as MEA and PBF at finding the optimal solution (Figure 4.2), with HDF doing slightly better than PBF and MEA at the beginning iteration of the search.

n	10	30	50
95%			
PBF	-0.74192	-0.84494	-0.86716
MEA	-0.76704	-0.86434	-0.86717
HDF	-0.86193	-0.86604	-0.86640
LDF	-0.70791	-0.82314	-0.83796
Average			
PBF	-0.83765	-0.86000	-0.86718
MEA	-0.84743	-0.86695	-0.86718
HDF	-0.86515	-0.86677	-0.86718
LDF	-0.80861	-0.85087	-0.85807
5%			
PBF	-0.86546	-0.86653	-0.86718
MEA	-0.86674	-0.86718	-0.86718
HDF	-0.86710	-0.86718	-0.86718
LDF	-0.86378	-0.86691	-0.86691

Table 4.1: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for P4. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found.

Recall the sixth test problem, P6, in Chapter 3. P6, represents the case of multiple inputs, $x_1, x_2, x_3, x_4, x_5, x_6, x_7$, and x_8, and multiple highly nonlinear constraints $c_1(\mathbf{x})$ and $c_2(\mathbf{x})$. Here, we would like to minimize the objective, f, while simultaneously satisfying the

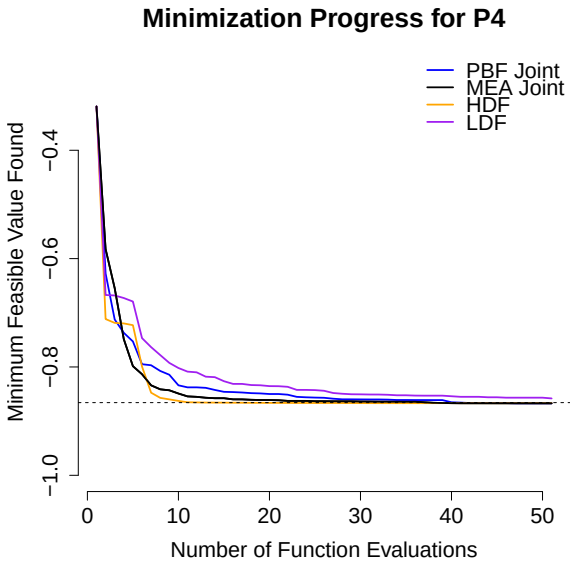


Figure 4.2: After 50 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for P4. The figure shows the average best feasible minimum over the 30 runs.

constraints, c_1 and c_2, i.e.,

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = \cos(\pi(x_1 + x_2x_3 + x_4)) + 0.2\sin\left(\frac{4\pi(x_5x_6 + x_7)}{x_8 + 1}\right) \tag{4.14}$$

$$\text{s.t.} \quad c_1(\mathbf{x}) = \sin(\pi(x_1 + x_2x_3 + x_4)) + 0.2\cos\left(\frac{4\pi(x_5x_6 + x_7)}{x_8 + 1}\right) \leq 0 \tag{4.15}$$

$$c_2(\mathbf{x}) = -\cos(\pi(x_1 + x_2x_3 + x_4)) + 0.2\cos\left(\frac{4\pi(x_5x_6 + x_7)}{x_8 + 1}\right) \leq 0 \tag{4.16}$$

$$\mathbf{x} \in [0, 1]^8. \tag{4.17}$$

The optimal solution to P6 is $f(\mathbf{x}) = -0.2828427$ which occurs at $x = (0.7909, 0.2670, 0.0798, 0.6426, 0.9161, 0.9377, 0.9645, 0.3893)$.

We ran the statistical filter algorithms fitting the joint PLGMP models and solving the subproblems associated with PBF, MEA, HDF, and LDF. We first sampled 80 initial inputs from a LHD on $[0, 1]^8$ and then sequentially selected 220 candidate points from a LHD on $[0, 1]^8$ based on the appropriate subproblem. We reran this analysis 30 times under different initial sample inputs from a LHD on $[0, 1]^8$ and recorded the average best feasible solution, Figure 4.3, and its 95% posterior intervals, Table 4.2.

On a much harder problem, as compared to P4, the results of running the statistical filter algorithm corresponding to HDF were far better than expected. Although PBF, MEA, and LDF all eventually converged to the optimal solution to P6, it was the rate at which HDF did it that was quite astounding. For over 100 iterations, HDF was far superior at the rate in which it converged towards to the true solution as compared to the other three competing metrics. In modeling multiple feasibility measures, HDF seems capable of creating a higher fidelity filter that is able to identify better candidate points much more quickly than the other algorithms. On the opposite end of the performance spectrum, not

n	50	150	220
		95%	
PBF	-0.19613	-0.28055	-0.28208
MEA	-0.10378	-0.28103	-0.28201
HDF	-0.27281	-0.28110	-0.28277
LDF	-0.079650	-0.215598	-0.257502
		Average	
PBF	-0.24677	-0.28209	-0.28283
MEA	-0.23837	-0.28230	-0.28284
HDF	-0.27919	-0.28213	-0.28284
LDF	-0.16587	-0.26503	-0.27609
		5%	
PBF	-0.27765	-0.28282	-0.28284
MEA	-0.27830	-0.28278	-0.28284
HDF	-0.28233	-0.28283	-0.28284
LDF	-0.28035	-0.28124	-0.28214

Table 4.2: After 220 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for P6. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using HDF performs the best for this problem after 220 additional evaluations.

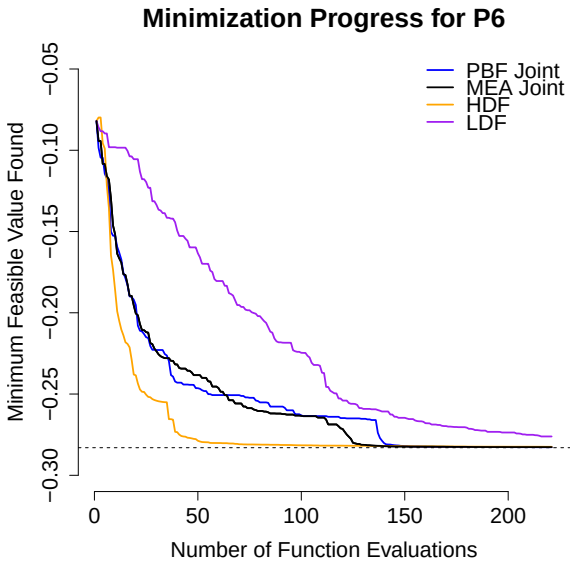


Figure 4.3: After 220 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for P6. The figure shows the average best feasible minimum over the 30 runs.

surprisingly, LDF did a terrible job of converging to the true solution, as compared to the other algorithms. In fact, LDF was significantly higher at all successive iterations of the search as compared to the other three metrics, and LDF did not converge to the true solution even after 220 updates.

4.4 Welded Beam Problem (WB)

Recall the pseudo black box computer model based on the fairly realistic real-world welded beam (WB) problem (Coello Coello & Montes (2002); Hedar (2004)). The WB problem has four inputs x_1, x_2, x_3, and x_4, and multiple constraints $g_1(\mathbf{x}), \ldots, g_6(\mathbf{x})$. The objective function, f, is the cost associated to construct a welded beam subject to constraints on sheer stress ($g_1(\mathbf{x})$, t), bending stress in the beam ($g_2(\mathbf{x})$, s), buckling load on the bar ($g_6(\mathbf{x})$, P_c), end deflection of the beam ($g_5(\mathbf{x})$, d), and side constraints ($g_3(\mathbf{x})$, $g_4(\mathbf{x})$). Here we would like to minimize the cost associated with constructing the welded beam, while also simultaneously satisfying the six physical constraint functions.

One of the main challenges of the welded beam problem is that the entire input space contains only a very tiny percentage (0.0972%) of feasible inputs. This scarcity of feasible points in the input space led our statistical filter algorithm, from Chapter 3, to converge at a much slower rate to the optimal solution. Thus, it is the hope that there will be a significant speed gain in convergence by applying either Algorithm 4 or 5 to the welded beam problem than seen before. We first sampled 40 initial inputs from a LHD over the appropriate input domain, and then sequentially selected 960 more candidate points from the same LHD based on the PBF, MEA, HDF and LDF methods. We reran this analysis

30 times under different initial sample inputs from the same LHD and recorded the average best feasible solution, Figure 4.4, and its 95% posterior intervals, Table 4.3.

n	300	600	960
		95%	
PBF	1111.61	1.7445	1.7361
MEA	1097.26	1.7450	1.7356
HDF	1072.92	1.7334	1.7334
LDF	1121.13	1121.13	1121.13
		Average	
PBF	455.15	1.7354	1.7309
MEA	413.64	1.7356	1.7308
HDF	209.59	1.7304	1.7304
LDF	862.72	613.83	471.53
		5%	
PBF	1.7306	1.7284	1.7250
MEA	1.7334	1.7286	1.7249
HDF	1.7309	1.7276	1.7249
LDF	4.1510	3.1659	3.1086

Table 4.3: After 960 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for the welded beam problem. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found.

Much like the prior analyses in Section 4.3, we see that HDF and LDF exhibit the same general tendencies in converging to the global minimum solution as seen before. HDF converged, on average, to the global minimum much quicker than PBF, MEA, while LDF did not come close to converging to the global minimum over the 960 updates to the filter (Figure 4.4). In fact, none of the runs of LDF came even close to converging to the global minimum solution of the WB problem (Table 4.3). A breakthrough, the rate at which HDF appears to converge to the global minimum, as compared to PBF and MEA, is

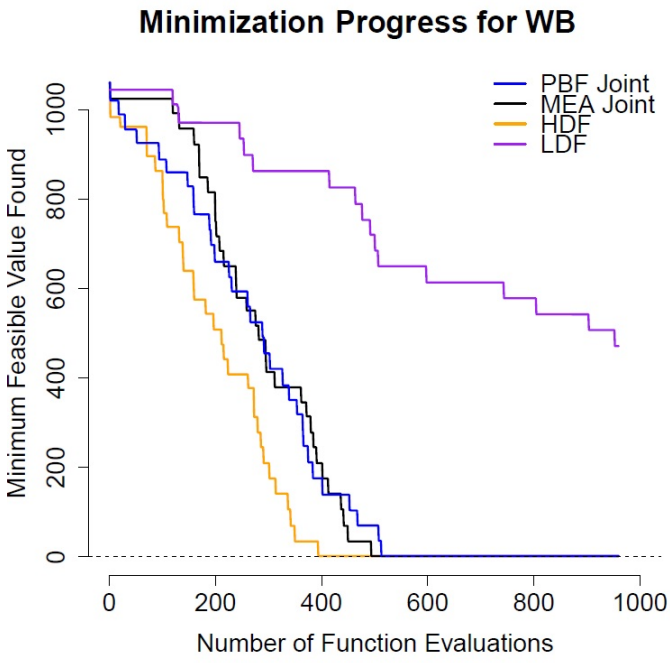


Figure 4.4: After 960 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for the welded beam problem. The figure shows the average best feasible minimum over the 30 runs.

a clear advantage of the HDF method when the percentage of feasible points is very small over the entire input space since PBF and MEA will tend to converge slower under this scenario. And although HDF, PBF, and MEA all eventually converge to the solution of the WB problem, the results of this experiment provide more empirical evidence in supporting the use of Algorithm 4, as well as more empirical evidence in opposition to using Algorithm 5.

4.5 Pump-and-Treat Hydrology Problem

We revisit the real-world hydrology computer model of Section 3.6, and test our newly proposed statistical filter algorithms, based on Algorithm 4 and 5, on it. Recall the formulation of the pump-and-treat hydrology problem as

$$\min_{\mathbf{x}}\{f(\mathbf{x}) = \sum_{j=1}^{6} x_j : \mathbf{c}(\mathbf{x}) \leq 0,\, \mathbf{x} \in [0, 20 \cdot 10^4]^6\} \tag{4.18}$$

where the objective f we wish to minimize is linear and describes the cost required to operate the wells. The two plumes are contained when the constraint, $\mathbf{c}$, is met. Here, $\mathbf{c}$ is a function of two constraints functions, $c_A(\mathbf{x})$ and $c_B(\mathbf{x})$, that describe when the contaminants in plume A and B have been contained, respectively. Collectively there are two constraints to the pump-and-treat hydrology problem, i.e., $p > 2$, and so Algorithm 4 and 5 are clearly applicable to use to solve this problem. Mirroring the problem setup in Section 3.6, we start with an initial sample of 60 inputs from a LHD in $[0, 20000]^6$ to initialize the filter $\mathcal{F}$. We then proceed to sequentially select 440 more points based on using Algorithms 2, 4, and 5. At each of the 440 iterations we fit our joint PLMGP model for the objective and constraint functions with the PLGMP model using the same priors as Section 3.3 and

$N = 4000$ particles. We follow the strategy of Taddy et al. (2009) and select the candidate set of inputs from a LHD of size 500 times the input dimension augmented by an additional 10% of the candidate locations taken from a smaller LHD bounded to within 5% of the domain range of the current best point. We reran this analysis 30 times under different initial sample inputs from a LHD on $[0, 20000]^6$ and recorded the average best feasible solution, Figure 4.5, and its 95% posterior intervals, Table 4.4.

n	150	300	440
95%			
PBF	29515.83	26230.13	24506.18
MEA	29281.62	25535.59	24554.47
HDF	28751.23	25365.94	24588.09
LDF	35203.67	31453.68	29643.20
Average			
PBF	27406.38	24649.97	23502.50
MEA	27110.29	23987.08	23552.58
HDF	26361.48	24581.32	23572.74
LDF	32844.21	29666.39	28396.38
5%			
PBF	26342.30	23885.85	23400.96
MEA	26029.78	23472.18	23450.88
HDF	25167.88	23639.61	23497.53
LDF	31653.56	28801.58	27768.41

Table 4.4: After 440 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for the pump-and-treat hydrology problem. The table shows the average best feasible minimum over the 30 runs, as well as 5^{th} and 95^{th} percentiles for the best feasible minima found. On average, the joint model using HDF performs the best for this problem after 440 additional evaluations.

Comparing the results of running PBF, MEA, HDF and LDF, we see that on average HDF does a better job than the other filter methods at converging towards an optimal solution to the pump-and-treat hydrology problem. Overall, the PBF method was

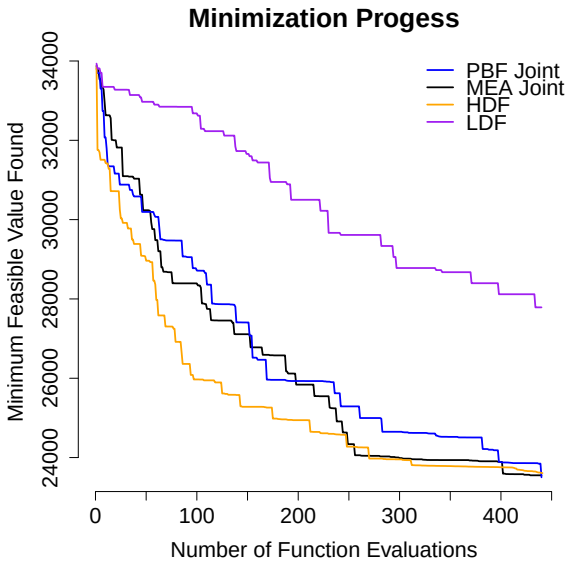


Figure 4.5: After 440 updates: The progress in minimization for 30 Monte Carlo repetitions with random initial conditions using the different statistical filter algorithms for the pump-and-treat hydrology problem. The figure shows the average best feasible minimum over the 30 runs.

still able to locate the best input configuration of pumping rates to minimize the constrained problem after 440 updates to the filter, however, HDF did a much better job at decreasing the objective function over a shorter span of iterations. Similar to Sections 4.3 and 4.4, we observed Algorithm 5 performing the worst in solving the pump-and-treat hydrology problem compared to the other two competing algorithms. In fact, LDF was the only method that did not come close to converging to the optimal solutions reported in the prior literature.

4.6 Discussion

In this chapter we proposed two different approaches for augmenting the original filter algorithm of Fletcher & Leyffer (2002). The two approaches relied on working with either a high dimensional filter (HDF) or a low dimensional filter (LDF). An exhaustive review of the constrained optimization literature has led us to believe that we are the first to explore using a filter with multiple feasibility measures for constrained optimization under the derivative free setting, and especially in the context of black box computer experiments. We hypothesized that using a higher dimensional filter would improve the original filter algorithm when there are multiple constraints to satisfy. We verified this hypothesis empirically by testing Algorithm 4 on a suite of synthetic and real-world computer experiment problems. In every case, the HDF method performed as well, if not better, than all of the other filter algorithms tested. It is here that the novelty and importance of the HDF method is so abundantly clear. On the other hand, as we also hypothesized, the LDF method performed terribly, as compared to the other filter algorithms, on all of the problems that we tested it

on. The LDF method was typically much slower to converge to the optimal solution, and in some cases, did not ever converge to the optimal solution under the allotted computing time.

The results of Chapter 4 highlight the need for higher fidelity algorithms. The compression of the constraint functions into a single feasibility measure was a clear disadvantage of the PBF, MEA, and LDF methods as compared to the HDF method. Compressing the constraints into a single feasibility measure implicitly weights vastly different constraint violations as being the same. This was not a problem for the HDF method as all constraints functions were viewed individually rather than as an aggregate measure. And thus, the higher resolution modeling of the HDF method led to much better solutions than its counterparts.

However, not everything about the HDF method was desirable. In particular, there is an added complexity in solving the subproblem in (4.7) that did not exist in (3.3). Although not insurmountable, a larger number of Monte Carlo iterations must be used in order to accurately calculate the volume of the ellipsoid as compared to the area of the ellipse. These extra iterations are not prohibitive, however, they do add additional computational cost to an already expensive problem. Secondly, the lack of an envelope used in the problems did not hinder the HDF method but is cause for concern. The envelope was introduced as a means to avoid convergence to infeasible points, and without it, we believe that provable convergence cannot be guaranteed for all problems. However, we think the creation of a "higher dimensional envelope" is an interesting and necessary problem to solve, and a definite avenue for future research.

Overall the HDF method showed great promise. On the problems considered, the HDF method was the clear best method, while the LDF method fared quite poorly and would not be recommended for use.

Chapter 5

Convergence of the Statistical Filter

Provable convergence is the hallmark of any good optimization method. Most optimization methods focus on proving global convergence, which is convergence to a local optimum given any arbitrary starting point. Demonstrating that an optimization method converges to a global optimum rather than a local one is a much harder problem. Traditionally, methods for optimization, both constrained and unconstrained, have relied on gradient information to ensure global convergence. Proving global convergence in the derivative free optimization (DFO) setting, i.e., no gradient information, is typically much harder.

As is often the case in practice, little to no derivative information may be available in a given optimization problem. There has been much work done in the field of derivative free optimization (see Conn et al. (2009) for a comprehensive list) when the problem is unconstrained. However, less focus has been given to the constrained DFO case.

Outside of derivative free optimization, there have been many algorithms and proofs for constructing a globally convergent filter algorithm. To name a few, Fletcher et al. (1998) and Fletcher et al. (2002) proved global convergence of a filter algorithm under sequential linear programming and sequential quadratic programming, respectively. Likewise, Ribiero et al. (2008) propose a globally convergent framework for filter methods based on step computations that are efficient, in the sense that, near a feasible nonstationary point, the reduction of the objective function is "large". On the other hand, Audet & Dennis (2006) incorporated a filter algorithm into a pattern-search method for derivative free optimization that was shown to be globally convergent. Here the authors were able to extend the usual pattern-search convergence results to the filter method. In similar vein, Ferreira et al. (2015) extended the work of Ribiero et al. (2008) to handle the case of derivative free optimization by utilizing an inexact restoration filter that was shown to be globally convergent.

A limitation of derivative free optimization is that convergence is typically rather slow (Conn et al., 2009). Even for a very large number of iterates, global convergence cannot always be assured. Empirical evidence, based on running the statistical filter algorithm of Chapter 3, from synthetic and real-world problems with known solutions seem to suggest that our algorithm is globally convergent. However, empirical evidence is not enough to confirm this belief. Thus, the aim of this chapter is to provide a proof of convergence for our statistical filter algorithm.

5.1 Necessary Conditions

For the remainder of this chapter, we shall simplify the notation, when appropriate, by using (h_k, f_k) to represent $(h(\mathbf{x}_k), f(\mathbf{x}_k))$. Now we shall state the necessary assumptions for the global convergence analysis of Algorithm 2.

A1. Algorithm 2 generates an infinite sequence of iterates, $(\mathbf{x}_k)_{k\in\mathbb{N}}$, that remains in a compact domain $\mathcal{X} \subset \mathbb{R}^d$.

A2. For some $\alpha > 0$, the objective function f and constraint functions $\mathbf{c}$ lie in the Hölder space of α-smooth functions, i.e., $f, \mathbf{c} \in C^{\alpha}[0,1]^d$ where $C^{\alpha}[0,1]^d$ is the Hölder space of α-smooth functions.

A1 is an assumption on the sequence generated by the algorithm that is typically enforced by including a bounded box into the problem constraints. However, A1 will always be true under our Algorithm 2 since the possible set of candidate points arise from a Latin hypercube design on $\mathcal{X}$. Even though we work under the derivative free optimization setting, we assume in A2 that the derivatives of the objective and constraint functions are not available to us but do exist. In other words, A2 is an assumption that the functions we are dealing with are smooth and well-behaved. Our statistical filter algorithm works well under far more general assumptions, however, A1 and A2, with $\alpha = 2$, i.e., working with twice continuously differentiable functions, are the standard assumptions, in the filter literature (Fletcher et al., 1998, 2002; Audet & Dennis, 2006; Ribiero et al., 2008; Ferreira et al., 2015), necessary for proving global convergence.

We now state the key theorem for proving global convergence of our statistical filter algorithm.

Theorem. As the number of iterations k gets arbitrarily large, the sequence of points, $(\mathbf{x}_k)_{k\in\mathbb{N}}$, that are accepted into the filter under Algorithm 2 converges almost surely to a local feasible solution for any choice of starting point.

We defer proof of this theorem until the end of the chapter after we have introduced and proved the necessary lemmas (1–7) that will be essential for proving the key theorem.

Recall that we conduct model based optimization by building surrogate models to approximate the true objective and constraint functions. Once built, these surrogate models can be used to predict values of the objective and constraint functions. Such approaches, however, are likely to require dense sampling to produce meaningful predictions and results. In order to achieve dense sampling, we use Latin hypercube designs for selecting points at which to make predictions. Latin hypercube designs are supposed to spread well the points in the sampling space by typically placing them at hypercube-type vertices (McKay et al., 1979). Furthermore, as we let the number of iterations k go to infinity, the method will explore the entire input space well.

Lemma 1. For any region R_0 of the input space with volume $\epsilon > 0$, there will be an infinite number of iterations where a candidate point is in R_0 with probability one.

Proof. By Assumption A1, the input domain is bounded, so its volume can be bounded above by some constant $V := \text{vol}(\mathcal{X})$. At each iteration, n candidate points are chosen via a Latin hypercube. Assuming ϵ is small relative to n, no more than one point could be chosen in R_0 from a particular Latin hypercube, and the probability of choosing a

point is bounded by $n\epsilon/V$. If R_0 is not sufficiently small, then this is a lower bound on the probability. Each iteration uses an independent Latin hypercube. Thus the probability of not choosing a point in R_0 in m iterations is $(1 - n\epsilon/V)^m$. For any given iteration m_i, the probability that there will never be another sample in R_0 goes to 0, thus with probability 1 there will be a future iteration $m_{i'}$ that contains a candidate point in R_0. This statement is true again for iteration $m_{i'}$, thus with probability 1 there will be an infinite number of interactions with a candidate point in R_0. □

Our argument for provable convergence of the statistical filter algorithm revolves around the use of asymptotics as the number of iterates, or sample size, k tends to infinity. Therefore, it is important to recognize the kind of asymptotics we are dealing with. In our case, the asymptotics we deal with are based on observations that get increasingly dense in some fixed and bounded region as their number increases. In other words, if one views $\mathcal{X} \subset \mathbb{R}^d$ as a bounded domain, an obvious way to increase k is to take observations at locations between the existing ones. Asymptotics of this type, where $k \to \infty$ but $0 < |\mathcal{X}| < \infty$, are known as *fixed-domain asymptotics* (Stein, 1999) or *infill asymptotics* (Cressie, 1993). If we define the quantity $\lambda = \max_{x_i} \min_{x_j} ||\mathbf{x}_i - \mathbf{x}_j||$, for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, as the distance between any two points in the input space, then, as a result *under infill asymptotics*, the minimum distance λ between all points in the input space tends to zero as the number of iterates, k, tends to infinity. Infill asymptotics, where more data are collected by sampling more densely in a fixed domain, will play a key role in helping us develop an argument for provable convergence of the statistical filter algorithm. In particular, infill asymptotics have been used to provide posterior consistency results for Gaussian processes (Stein (1999),Cressie

(1993)).

Lemma 2. Assume condition A1 holds, then as the sample size $k \to \infty$, our posterior predictive distribution converges in probability to the true underlying process.

Proof. Van Der Vaart & Van Zanten (2009) show that under infill asymptotics, the resulting posterior distribution converges in probability to the true distribution of the underlying process, w_0, as long as $w_0 \in C^{\alpha}[0,1]^d$, for some $\alpha > 0$, where $C^{\alpha}[0,1]^d$ is the Hölder space of α-smooth functions. Assumption $A2$ guarantees that our objective and constraint functions will lie in the Hölder space of α-smooth functions, and thus, our posterior predictive distributions for the objective and constraint functions will converge in probability to the true functions. See Van Der Vaart & Van Zanten (2009) for theorems, proofs, and discussion. □

Lemma 3. Assume condition A1 holds, then as the sample size $k \to \infty$, the area of our 95% posterior predictive probability contours converges in probability to 0.

Proof. This is a direct consequence of Lemma 2. Recall that our posterior predictive distribution, under the PLMGP, is a multivariate $\mathcal{T}$ process. As the resulting posterior predictive distribution converges in probability towards the true data generating process, by Lemma 2, our posterior predictive variances must converge in probability to 0 under infill asymptotics. Thus, as the posterior predictive variances converge in probability to 0, so must the area of our posterior predictive probability contours. □

Lemma 4. Consider an infinite sequence of iterations on which (h_k, f_k) is entered into the filter, and $\{f_k\}$ is bounded below. It follows that $h_k \to 0$.

Proof. We cite the proof of Lemma 1 of Chin & Fletcher (2003). The envelope in (3.1) provides an important *inclusion property* that if a new pair is added to the filter, then the set of unacceptable points for the new filter always includes the set of unacceptable points for the old filter.

If $h_{k+1} \leq \beta h_k$ for all k sufficiently large, then $h_k \to 0$. Otherwise we define a subsequence $\mathcal{S}$ as follows. The initial index in $\mathcal{S}$ is the first iteration k on which $h_{k+1} > \beta h_k$. For any $k \in \mathcal{S}$, its successor $k_+ \in \mathcal{S}$ is the least $j > k$ such that $h_j > \beta h_k$. It is a consequence of the inclusion property that (h_{k_+}, f_{k_+}) is acceptable to (h_k, f_k), even if the latter pair has been deleted from the filter on an intermediate iteration. Hence $f_k - f_{k_+} \geq \gamma h_{k_+} > 0$. Thus f_k is monotonically decreasing for $k \in \mathcal{S}$ and, because f_k is bounded below, it follows that $\sum_{k \in \mathcal{S}} h_k$ is bounded above, and hence that $h_k \to 0$ for $k \in \mathcal{S}$. Moreover, intermediate iterations j such that $k < j < k_+$ have the property that $h_j \leq \beta h_k$, so it follows that $h_k \to 0$ on the main sequence. □

Lemma 5. For infeasible points, the minimum vertical distance between the envelope and filter is positive.

Proof. Consider a candidate point $\mathbf{x}_*$. If $h(\mathbf{x}_*) > 0$, i.e., the point is infeasible, then the minimum vertical distance between the envelope and the filter is

$$\min_i \{f(\mathbf{x}_i) - (f(\mathbf{x}_i) - \gamma h(\mathbf{x}_*))\} = \gamma h(\mathbf{x}_*) > 0 \tag{5.1}$$

since, by definition of the envelope, $\gamma \in (0, 1)$. □

Lemma 6. For feasible points, the minimum vertical distance between the envelope and filter is zero.

Proof. Consider a candidate point $\mathbf{x}_*$. If $h(\mathbf{x}_*) = 0$, i.e., a feasible point, then the minimum vertical distance between the envelope and the filter is

$$\min_i \{f(\mathbf{x}_i) - (f(\mathbf{x}_i) - \gamma h(\mathbf{x}_*))\} \min_i \{f(\mathbf{x}_i) - (f(\mathbf{x}_i) - 0)\} = 0. \tag{5.2}$$

□

Lemmas 1–6 lay the necessary foundations for ensuring that our statistical filter algorithm is globally convergent. In what follows, we make use of all of the lemmas to build our argument for global convergence.

5.2 Subproblems

In this section, we contend that global convergence can be obtained by our statistical filter algorithm by solving either the probability beyond the filter (PBF) (Section 3.2.1) or the maximum expected area (MEA) (Section 3.2.2) subproblem. Before delving into the specifics of either subproblems use, we first give the intuition for why we must obtain global convergence in general. Typically, it is enough to show that if assumptions A1 and A2 are true, when $\alpha = 2$, then an assumption that successive iterates produce a *sufficient reduction* in the objective function is enough to conclude global convergence of the filter algorithm (Fletcher & Leyffer, 2002; Chin & Fletcher, 2003; Ribiero et al., 2008). Typically the envelope is used to ensure sufficient reduction of the objective function (Chin & Fletcher, 2003). In our case, the envelope coupled with Lemmas 1–6 ensure global convergence of our Algorithm 2.

In general, consider the case of an infinite sequence of iterates $(\mathbf{x}_k)$. Under Lemma 4, we have that $h_k \to 0$ and so at least one feasible point must be obtained. Given that a

feasible point is obtained, Lemmas 5 and 6 define the minimum vertical distance between the envelope and the filter. Important to note, wherever a point is infeasible, the minimum vertical distance has to be positive, and zero otherwise. The fact that there will be positive distance between the filter and the envelope will be critical in establishing global convergence. Now, Lemmas 2 and 3 provide us with the fact that as the number of successive iterates ($\mathbf{x}_k$) increases, the posterior predictive process, under our surrogate model, converges arbitrarily close, in probability, to the true data generating process. Thus, as the number of observed points tends to infinity, we can essentially regard the posterior predictive process as the true data generating process.

Now, with regard to the two subproblems, we claim that the envelope will ensure global convergence of our statistical filter algorithm. Consider again that Lemmas 2 and 3 guarantee arbitrarily accurate prediction of the objective function and feasibility measure. Thus, any candidate point that is predicted to be infeasible must be predicted to fall to the right, or to the northeast, of the current envelope. However, both the PBF and MEA metric selects candidate points based on finding a maximum positive quantity representing the distance to the left, or to the southwest, of the envelope. Since, by Lemma 5, there will always be positive area between the filter and the envelope for infeasible points, and, because of our accurate predictions, we can never pick candidate points that are infeasible, as k tends to infinity, based on PBF and MEA. Thus, the only way to select a candidate point, based on PBF and MEA, is to select one where $h(\mathbf{x}_*) = 0$ and $f(\mathbf{x}_*) < f(\mathbf{x}_j)$ for all $j \in \mathcal{F}$. As the final necessary component for ensuring global convergence, we state this property as a lemma.

Lemma 7. As $k \to \infty$ the only way to select a new candidate point $\mathbf{x}_*$, based on either PBF or MEA, is to select one where $h(\mathbf{x}_*) = 0$ and $f(\mathbf{x}_*) < f(\mathbf{x}_j)$ for all $j \in \mathcal{F}_k$, where $\mathcal{F}_k$ is the current filter at iteration k.

Proof. We first deal with the case of using the PBF subproblem. Consider an infinite sequence of iterations $(\mathbf{x}_k)$. As k grows large, we have that, by Lemmas 2–4, our predictions of the objective functions and infeasibility measure become highly accurate, and we have that $h_k \to 0$. Now, recall that the PBF subproblem selects candidate points that maximize the probability of being to the left of the envelope. By Lemma 5, all predictions of infeasible points will be to the right of the envelope. Furthermore, Lemma 3 also ensures that as the area of the posterior predictive probability contours shrink to zero, we will always have that the probability of infeasible points being selected tends to 0 since their respective posterior predictive probability contours will always have an area falling beyond (to the left of) the envelope of zero. Thus, we must consider the case when $h(\mathbf{x}_*) = 0$, i.e., a feasible point.

Now, given that a candidate point $\mathbf{x}_*$ is predicted to be feasible, there are two possible scenarios. Either the point is feasible and does not reduce the objective function, i.e., $f(\mathbf{x}_*) > f(\mathbf{x}_j)$, or the point is feasible and does reduce the objective function, i.e., $f(\mathbf{x}_*) < f(\mathbf{x}_j)$, for all j in the current filter $\mathcal{F}_k$. Recall the definition of the envelope in (3.1). Clearly, when $h(\mathbf{x}_*) = 0$ the only way for a candidate point to be acceptable to the filter is for $f(\mathbf{x}_*) < f(\mathbf{x}_j)$. But once again, by Lemma 3, the posterior predictive probability contour associated with the candidate point $\mathbf{x}_*$ essentially has area of 0 and so the PBF metric will never have a probability greater than 0 of selecting a candidate point that does

not reduce the objective function. Thus, the only way for a candidate point to be selected by the PBF metric is to have it be feasible and below the current best minimum objective value. Thus, as $k \to \infty$, the PBF subproblem will only choose candidate points such that $h(\mathbf{x}_*) = 0$ and $f(\mathbf{x}_*) < f(\mathbf{x}_j)$ for all $j \in \mathcal{F}_k$.

Now, consider the case of using the MEA subproblem. Consider again an infinite sequence of iterations $(\mathbf{x}_k)$. As k grows large, we have that, by Lemmas 2–4, our predictions of the objective function and infeasibility measure become highly accurate, and we have that $h_k \to 0$. Now, recall that the MEA subproblem selects candidate points that maximize the expected area of being to the left of the envelope. Similar as before, by Lemma 5, all predictions of infeasible points will be to the right of the envelope and thus, by definition of the MEA metric, all points predicted to the right of the envelope will have maximum expected areas of 0. Thus, we must consider the case when $h(\mathbf{x}_*) = 0$, i.e., a feasible point.

Now, given that a candidate point $\mathbf{x}_*$ is predicted to be feasible, there are two possible scenarios. Either the point is feasible and does not reduce the objective function, i.e., $f(\mathbf{x}_*) > f(\mathbf{x}_j)$, or the point is feasible and does reduce the objective function, i.e., $f(\mathbf{x}_*) < f(\mathbf{x}_j)$, for all j in the current filter $\mathcal{F}_k$. Recall the definition of the envelope in (3.1). Clearly, when $h(\mathbf{x}_*) = 0$ the only way for a candidate point to be acceptable to the filter is for $f(\mathbf{x}_*) < f(\mathbf{x}_j)$. This occurs because, if a feasible point is predicted such that $f(\mathbf{x}_*) > f(\mathbf{x}_j)$ then the MEA metric associated with that point will always be 0 and thus never selected. Therefore, the only way for a candidate point to be selected by the MEA metric is to have it be feasible and below the current best minimum objective value because that is the only way for it to have positive area under the MEA metric. Thus, as

$k \to \infty$, the MEA subproblem will only choose candidate points such that $h(\mathbf{x}_*) = 0$ and $f(\mathbf{x}_*) < f(\mathbf{x}_j)$ for all $j \in \mathcal{F}_k$. □

Now that we have outlined and discussed our seven lemmas, we return to proving the key theorem of this chapter for establishing global convergence.

Proof of Theorem. In order to prove this theorem, we must consider the following two cases: (i) there are only a finite number of points accepted into the filter and this finite sequence converges almost surely to a local feasible solution, or (ii) the infinite sequence converges almost surely to a local feasible solution. First we consider case (i). Here we will demonstrate that this case (i) holds by contradiction. Assume that there are only a finite number of points accepted into the filter and the finite sequence does not converge to a local feasible solution. We assume that k is large and so, by Lemmas 1, 2, and 3, our predictions of the objective function and feasibility measure become arbitrarily good as k grows large. However, by Lemma 4, as k grows large we must have that $h_k \to 0$, and so finally, by Lemma 7, there must be some positive probability at each iteration that we will find a better point to accept into the filter that would create a decrease in the objective function. Thus, an infinite sequence of iterations would find another point, with probability one, which would be acceptable to the filter. But this is a contradiction to our assumption that we have a finite number of points accepted into the filter. Thus, if we do have a finite number of points then this finite sequence must have converged almost surely to a local feasible solution.

Next consider case (ii). By Lemma 7, given an infinite sequence of iterates $(\mathbf{x}_k)$, for k large, the only way that the MEA and PBF subproblems can pick new candidate points is

to select those points that will be feasible and obtain a reduction in the objective function, i.e., $h(\mathbf{x}_*) = 0$ and $f(\mathbf{x}_*) < f(\mathbf{x}_j)$, for all j in the current filter $\mathcal{F}_k$. Thus, Lemma 7 can be seen as a way of imposing a sufficient reduction criteria. Moreover, given assumption A1 that the sequence remains in a compact bounded domain, we have that the infinite sequence must converge almost surely to a local feasible solution as the algorithm can only chose points, with probability one, that are feasible and that reduce the objective function. Equivalently, once feasible, i.e., $h_k \to 0$, the global minimum acts as a lower bound on the infinite sequence and since this infinite sequence only decreases, by Lemma 7, we must have that it converges almost surely to a local feasible solution.

We have proven that either case (i) or case (ii) must hold, and thus our statistical filter algorithm satisfies all of the necessary assumptions for global convergence and should be deemed globally convergent. □

5.3 Discussion

In this chapter we considered the problem of proving global convergence of the statistical filter algorithm of Chapter 3. Under some standard assumptions, we provide an argument for the global convergence of the statistical algorithm under both the PBF and MEA subproblems. This global convergence proof comes from that the fact that as the number of iterates tends to infinity, selection of candidate points, under the PBF and MEA subproblems cannot be done in a way that does not guarantee selecting feasible points that will decrease the objective function. We view this condition as being comparable to the typical sufficient decrease criteria and conclude that the statistical filter algorithm must be

globally convergent.

Although we have shown that our statistical filter possess global convergence properties, we do not give any indication as to how large the number of iterates k has to be. Recalling that convergence is typically rather slow for derivative free optimization algorithms (Conn et al., 2009), it would be a nice feature to have some sort of bound on the number of iterations or the rate of convergence of our statistical filter algorithm. Cartis & Scheinberg (2015) recently developed a method based on probabilistic models for proving global convergence of unconstrained optimization problems. Cartis & Scheinberg (2015) method not only proved global convergence but also provided a bound on the expected number of iterations that the algorithms take before they achieve a desired level of accuracy. The analysis was based on stochastic processes and submartingale properties and thus would seem like a nice extension to apply to the statistical filter algorithm given its roots in stochastic processes. Unfortunately, the analysis by Cartis & Scheinberg (2015) was for the unconstrained case and would need to be adapted to fit into the constrained optimization case.

Lastly, although we did not attempt a convergence analysis of the filter methods in Chapter 4, we think it would be a fruitful endeavor as well, and definitely part of future research.

Chapter 6

Conclusions

Many complex phenomena in the world cannot be studied physically and so there is an ever growing need for computer models to study these complicated systems. As science evolves and questions become more complex and expensive, the need for efficient computer models, and surrogate models, becomes even more abundant. An important topic, constrained optimization of computer experiments is a very challenging problem. Computer models may be difficult to optimize because their output functions may be complex, multi-modal, and difficult to understand. The novelty of the work presented in this dissertation is in proposing a new methodology for solving constrained optimization problems by combining statistical modeling and nonlinear programming. The marriage of stochastic process models with filter methods is a powerful combination for understanding and optimizing black box computer models.

In this dissertation we developed novel methodology for modeling of correlated outputs of different types. Although motivated by applications in computer experiments,

this new methodology presents a general purpose modeling framework for modeling correlated outputs that need not arise solely from computer models. However, with computer models in mind, this new joint stochastic process model proved to be an efficient surrogate model for emulating computer experiments. The joint stochastic process model utilized the sequential Monte Carlo method, particle learning, and was able to conduct fast sequential inference to help alleviate the computational burden that comes from working with computer models that are prohibitively expensive to run a large number of times.

In order to solve constrained optimization problems of expensive black box computer experiments we embedded the joint stochastic process model within a filter algorithm that outperformed many of the established comparators. By building a joint surrogate model for the objective and constraint functions we were able to establish two novel statistical metrics for guiding the filter efficiently. We demonstrated the success of our new statistical filter algorithm on a suite of synthetic examples and a real world hydrology computer optimization problem. Our newly designed statistical filter algorithm showed much promise as a globally convergent algorithm. We took our statistical filter algorithm one step further and extended it to allow for the incorporation of more information through what we describe as a multidimensional filter algorithm. This multidimensional filter algorithm capitalized on the inherent weaknesses of the original filter algorithm and was able to produce a better solution to the constrained optimization problem.

Although we conclude the dissertation here, there are always avenues for future work. In general, we hypothesize that we may be able to gain faster convergence to a global solution by incorporating a time dependent upper bound U_k on the constraint function that

shrinks (or slides) over time. Currently within the statistical filter algorithms we use an upper bound U on the constraint functions that is set arbitrarily large or by some real-word physical property of the problem. In applications, setting the upper bound U this way works well, however, we think that by allowing U to shrink over time towards $h(\mathbf{x}) = 0$ we may be able to force the filter to find feasible solutions more rapidly.

The multidimensional filter worked exceedingly well but in creating it we lost some of the desirable convergence properties that the envelope afforded us. With as much promise as the multidimensional filter showed, we think it a very important task to establish a multidimensional envelope equivalent that may even help facilitate faster convergence to a solution of the constrained optimization problem as well as allow for provable convergence results of the method to hold.

Lastly, and most importantly a personal goal, we would like to create an `R` package based on the joint stochastic model of Chapter 2 to disseminate to the statistics community at large.

Bibliography

Abrahamsen, P. (1997). A review of Gaussian random fields and correlation functions. Technical Report 917, Norwegian Computing Center, Box 114 Blindern, N-0314 Oslo, Norway.

Albert, J. H. & Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association* **88**, 669–679.

Audet, C. & Dennis, J. E., Jr. (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization* **17**, 188–217.

Banerjee, S. & Gelfand, A. E. (2002). Prediction, interpolation, and regression for spatial misaligned data points. *Sankhya* **64**, 227–245.

Banerjee, S., Gelfand, A. E. & Carlin, B. P. (2004). *Hierarchical Modeling and Analysis for Spatial Data*. New York: Chapman & Hall/CRC.

Berger, J. O., De Oliveira, V. & Sansó (2000). Objective bayesian analysis of spatially correlated data. *Journal of the American Statistical Association* **96**, 1361–1374.

Bonilla, E. V., Chai, K. M. & Williams, C. K. I. (2008). Multi-task gaussian process

prediction. In *Advances in Neural Information Processing Systems 20*, Eds. J. C. Platt, D. Koller, Y. Singer & S. Rowesi. Cambridge, MA: MIT Press.

Box, G. E. P. & Draper, N. R. (1987). *Empirical Model Building and Response Surfaces*. Oxford: Wiley.

Cartis, C. & Scheinberg, K. (2015). Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. Technical report, Mathematical Institute, University of Oxford.

Carvalho, C. M., Johannes, M. S., Lopes, H. F. & Polson, N. G. (2010). Particle learning and smoothing. *Statistical Science* **25**, 88–106.

Chan, A. B. (2013). Multivariate generlaized gaussian process models. Technical report, Department of Computer Science, City University of Hong Kong.

Chin, C. M. & Fletcher, R. (2003). On the global convergence of an slp-filter algorithm that takes eqp steps. *Mathematical Programming* **96**, 161–177.

Coello Coello, C. A. & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics* **16**, 2002.

Conn, A. R., Scheinberg, K. & Vicente, L. N. (2009). *Introduction to Derivative-Free Optimization*. Philadelphia: SIAM.

Conti, S. & O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference* **140**, 640–651.

Cressie, N. A. C. (1993). *Statistics for Spatial Data, revised edition.* John Wiley & Sons.

Diggle, P. J. & Ribeiro Jr., P. J. (2007). *Model-based Geostatistics.* Springer.

Fang, K., Li, R. & Sudjianto, A. (2005). *Design and Modeling for Computer Experiments.* Chapman & Hall/CRC.

Ferreira, P. S., Karas, E. W. & Sachine, M. (2015). Global convergence of a derivative-free inexact restoration filter algorithm for nonlinear programming. Technical report, Department of Mathematics, Federal University of Paraná.

Fletcher, R. & Leyffer, S. (2002). Nonlinear programming without a penalty function. *Mathematical Programming* **91**, 239–270.

Fletcher, R., Leyffer, S. & Toint, P. (1998). On the global convergence of a slp-filter algorithm. Technical report, Numerical Analysis Report NA/183, University of Dundee, UK.

Fletcher, R., Leyffer, S. & Toint, P. (2002). On the global convergence of a filter-sqp algorithm. *SIAM Journal on Optimization* **13**, 44–59.

Fletcher, R., Leyffer, S. & Toint, P. (2006). A brief history of filter methods. Technical report, ANL/MCS-P1372-0906, Argonne National Laboratory, Mathematics and Computer Science Division.

Floudas, C. A. & Pardalos, P. M. (1990). *A Collection of Test Problems for Constrained Global Optimization Algorithms.* New York: Springer-Verlag.

Fowler, K. R., Reese, J. P., Kees, C. E., Dennis Jr., J. E., Kelley, C. T., Miller, C. T., Audet, C., Booker, A. J., Couture, G., Darwin, R. W., Farthing, M. W., Finkel, D. E., Gablonsky, G., Gray, G., & Kolda, T. G. (2008). A comparison of derivative-free optimization methods for water supply and hydraulic capture community problem. *Advances in Water Resources* **31**, 743–757.

Fricker, T. E., Oakley, J. E. & Urban, N. M. (2013). Multivariate gaussian process emulators with nonseparable covariance stuctures. *Technometrics* **55**, 47–56.

Gaspari, G. & Cohn, S. E. (1999). Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Metereological Society* **125**, 723–757.

Gelfand, A. E., Schmidt, A. M., Banerjee, S. & Sirmans, C. F. (2004). Nonstationary multivariate process modeling through spatially varying coregionalization. *Test* **13**, 263–312.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A. & Rubin, D. (2013). *Bayesian Data Analysis, Third Edition.* Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

Goulard, M. & Voltz, M. (1992). Linear coregionalization model: Tools for estimation and choice of cross-covariogram matrix. *Mathematical Geology* **21**, 269–286.

Gould, N. I. M., Leyffer, S. & Toint, P. L. (2005a). A multidimensional filter algorithm for nonlinear equations and nonlinear least squares. *SIAM Journal on Optimization* **21**, 17–38.

Gould, N. I. M., Sainvitu, C. & Toint, P. L. (2005b). A filter trust region method for unconstrained optimization. *SIAM Journal on Optimization* **16**, 341–357.

Gramacy, R. B. (2012). *plgp: Particle Learning of Gaussian Processes*. R package version 1.1-5.

Gramacy, R. B., Gray, G. A., Le Digabel, S., Lee, H. K. H., Ranjan, P., Wells, G. & Wild, S. M. (2015). Modeling an augmented lagrangian for improved blackbox constrained optmization. *To appear in Technometrics* .

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

Gramacy, R. B. & Lee, H. K. H. (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing* **22**, 713–722.

Gramacy, R. B. & Polson, N. G. (2011). Particle learning of Gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics* **20**, 102–118.

Gupta, A. K. & Nagar, D. K. (2000). *Matrix Variate Distributions*. Boca Raton, Florida: Chapman & Hall/CRC Press.

Hayashi, K., Takenouchi, T., Tomioka, R. & Kashima, H. (2012). Self-measuring similarity for multi-task gaussian process. In *ICML Unsupervised and Transfer Learning*, Eds. I. Guyon, G. Dror, V. Lemaire, G. W. Taylor & D. L. Silver, volume 27 of *JMLR Proceedings*, pp. 145–154. JMLR.org.

Hedar, A. (2004). *Studies on Metaheuristics for Continuous Global Optimization Problems.* Ph.D. thesis, Kyoto University, Japan.

Higdon, D. (2002). Space and space–time modeling using process convolutions. In *Quantitative Methods for Current Environmental Issues*, Eds. C. Anderson, V. Barnett, P. C. Chatwin & A. H. El-Shaarawi, pp. 37–56. London: Springer-Verlag.

Higdon, D., Swall, J. & Kern, J. (1999). Non-stationary spatial modeling. In *Bayesian Statistics 6*, Eds. J. M. Bernardo, J. O. Berger, A. P. Dawid & A. F. M. Smith, pp. 761–768. Oxford University Press.

Hjort, N. L. & Omre, H. (1994). Topics in spatial statistics. *Scandinavian Journal of Statistics* **21**, 289–357.

Jones, D., Schonlau, M. & Welch, W. J. (1998). Efficient global optimization of expensive black box functions. *Journal of Global Optimization* **13**, 455–492.

Journel, A. G. & Huijbregts, C. J. (1978). *Mining Geostatistics.* Academic Press.

Kraft, D. (1988). A software package for sequential quadratic programming. Technical report, DFVLR-FB 88-28, Institut fuer Dynamik der Flugsysteme, Oberpfaffenhofen.

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

Lee, H. K. H., Gramacy, R. B., Linkletter, C. & Gray, G. A. (2011). Optimization

subject to hidden constraints via statistical emulation. *Pacific Journal of Optimization* **7**, 467–478.

Liang, W. W. J. & Lee, H. K. H. (2014). Sequential process convolution gaussian process models via particle learning. *Statistics and Its Interface* **7**, 465–475.

Lindberg, D. & Lee, H. K. H. (2015a). Optimization under constraints by applying an asymmetric entropy measure. Technical report, University of California, Santa Cruz.

Lindberg, D. & Lee, H. K. H. (2015b). Optimization under constraints by applying an asymmetric entropy measure. *Journal of Computational and Graphical Statistics* **24**, 379–393.

Liu, X., Chen, F., Lu, Y. & Lu, C. (2013). Spatial prediction of large multivariate non-gaussian data. Technical report, Department of Computer Science, Virginia Tech.

Loeppky, J. L., Sacks, J. & Welch, W. J. (2009). Choosing the sample size of a computer experiment: A practical guide. *Technometrics* **4**, 366–376.

MacEachern, S. M., Clyde, M. & Liu, J. S. (1999). Sequential importance sampling for nonparametric bayes models: The next generation. *Canadian Journal of Statistics* **27**, 251–267.

Majumdar, A. & Gelfand, A. E. (2007). Multivariate spatial modeling for geostatistical data using convolved covariance functions. *Mathematical Geology* **39**, 229–245.

Mardia, K. V. & Goodall, C. R. (1993). Spatial–temporal analysis of multivariate environmental monitoring data. *Multivariate Environmental Statistics* **6**, 76.

Matheron, G. (1962). *Traité de géostatistique appliquée*, volume 14. Paris: Editions Technip.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology* **58**, 1246–1266.

Matott, L. S., Leung, K. & Sim, J. (2011). Application of matlab and python optimizers to two case studies involving groundwater flow and contaminant transport modeling. *Computers and Geosciences* **37**.

Mayer, A. S., Kelley, C. T. & Miller, C. T. (2002). Optimal design for problems involving flow and transport phenomena in saturated subsurface systems. *Advances in Water Resources* **25**.

McDonald, M. G. & Harbaugh, A. W. (1996). Programmers documentation for modflow-96, an update to the u. s. geological survey modular finite difference groundwater flow model. Technical report, Open-File Report 96-486, U. S. Geological Survey.

McKay, M. D., Conover, W. J. & Beckman, R. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21**, 239–245.

Montagna, S. & Tokdar, S. T. (2013). Computer emulation with non-stationary Gaussian processes. Technical report, Duke University.

Moustaki, I. & Knott, M. (2000). Generalized latent trait models. *Psychometrika* **65**, 391–411.

Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian

regression and classification. Technical Report 9702, Deptartment of Statistics, University of Toronto.

Neal, R. M. (1999). Regression and classification using gaussian process priors (with discussion). In *Bayesian Statistics 6*, Ed. e. a. J. M. Bernardo, pp. 476–501. Oxford University Press.

Omre, H. (1987). Bayesian kriging – merging observations and qualified guesses in kriging. *Mathematical Geology* **19**, 25–38.

Omre, H. & Halvorsen, K. (1989). The bayesian bridge between simple and universal kriging. *Mathematical Geology* **21**, 767–786.

Omre, H., Halvorsen, K. B. & Berteig, V. (1989). A bayesian approach to kriging. In *Geostatistics*, Ed. M. Armstrong, volume 4 of *Quantitative Geology and Geostatistics*, pp. 109–126. Springer Netherlands.

Parr, J. M., J., K. A., Forrester, A. I. J. & Holden, C. M. E. (2012). Infill sampling criteria for surrogate-based optimization with constraint handling. *Engineering Optimization* **44**, 1147–1166.

Pourmohamad, T. & Lee, H. K. H. (2015). Multivariate stochastic process models for correlated responses of mixed type. *Bayesian Analysis* doi:10.1214/15-BA976.

Powell, M. J. D. (1994). A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis: proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, pp. 51–67. Kluwer Academic Pub.

Prado, R. & West, M. (2010). *Time Series: Modelling, Computation & Inference*. Boca Raton, FL: Chapman & Hall/CRC Press.

Rasmussen, C. E. & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press.

Ribiero, A. A., Karas, E. W. & Gonzaga, C. C. (2008). Global convergence of filter methods for nonlinear programming. *SIAM Journal on Optimization* **19**, 1231–1249.

Rowe, D. B. (2003). *Multivariate Bayesian Statistics: Models for Source Separation and Signal Unmixing*. Boca Raton, FL: Chapman & Hall/CRC.

Sacks, J., Welch, W. J., Mitchell, T. J. & Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science* **4**, 409–435.

Sammel, M. D., Ryan, L. M. & Legler, J. M. (1997). Latent variable models for mixed discrete and continuous outcomes. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)* **59**, 667–678.

Santner, T. J., Williams, B. J. & Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. New York, NY: Springer-Verlag.

Sasensa, M. J., Papalambros, P. & Goovaerts, P. (2002). Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization* **4**, 263–278.

Schmidt, A. M. & Gelfand, A. E. (2003). Multivariate emulators with nonseparable covariance stuctures. *Journal of Geophysical Research: Atmospheres* **108**.

Shen, C., Xue, W. & Pu, D. (2009). Global convergence of a tri-dimensional filter sqp algorithm based on the line search method. *Applied Numerical Mathematics* **59**, 235–250.

Stein, M. L. (1999). *Interpolation of Spatial Data*. New York, NY: Springer.

Storvik, G. (2002). Particle filters in state space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing* **50**, 281–289.

Svanberg, K. (2002). A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization* **12**, 555–573.

Svenson, J. D. & Santner, T. J. (2012). Multiobjective optimization of expensive black-box functions via expected maximin improvement. Technical report, Department of Statistics, Ohio State University.

Taddy, M., Lee, H. K. H., Gray, G. A. & Griffin, J. D. (2009). Bayesian guided pattern search for robust local optimization. *Technometrics* **51**, 389–401.

Urban, N. M. & Fricker, T. E. (2010). A comparison of latin hypercube and grid ensemble designs for the multivariate emulation of an earth system model. *Computers and Geosciences* **36**, 746–755.

Van Der Vaart, A. W. & Van Zanten, J. H. (2009). Adaptive bayesian estimation using a gaussian random field with inverse gamma bandwidth. *The Annals of Statistics* **37**, 2655–2675.

Wackernagel, H. (2003). *Multivariate Geostatistics: An Introduction with Applications*. New York, NY: Springer.

Wild, S. M., Regis, R. G. & Shoemaker, C. A. (2008). Orbit: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal of Scientific Computing* **30**, 3197–3219.

Wilson, B., Capoelleri, D., Simpson, T. W. & Frecker, M. (2001). Efficient pareto frontier exploration using surrogate approximations. *Optimization and Engineering* **2**, 31–50.

Xu, Z., Yan, F. & Qi, Y. (2012). Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. In *Proceedings of the 29th International Conference on Machine Learning*, ICML '12.

Ypma, J. (2014). *nloptr: R Interface to NLopt*. R package version 1.0.4.

Yu, K., Tresp, V. & Schwaighofer, A. (2005). Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 1012–1019. New York, NY, USA: ACM. doi:10.1145/1102351.1102479.

Zhe, S., Qi, Y., Youngja, P., Molloy, I. & Chari, S. (2013). Dintucker: Scaling up gaussian process models on multidimensional arrays with billions of elements. Technical report, arXiv preprint arXiv:1311.2663.

Zhe, S., Xu, Z., Chu, X., Qi, Y. & Park, Y. (2015). Scalable nonparametric multiway data analysis. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics* pp. 1125–1134.