Adapting Response Surface Methods for the Optimization of Black-Box Systems

Jacob J. Zielinski

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Statistics

G. Geoffrey Vining, Chair
Jeffrey B. Birch
Leanna L. House
Angela N. Patterson

August 16, 2010
Blacksburg, Virginia

Keywords: DACE, Response Surface, Computer Experiments, Gaussian Stochastic Process, Optimization, Bayesian, Kriging

Adapting Response Surface Methods for the Optimization of Black-Box Systems

Jacob J. Zielinski

(ABSTRACT)

Complex mathematical models are often built to describe a physical process that would otherwise be extremely difficult, too costly or sometimes impossible to analyze. Generally, these models require solutions to many partial differential equations. As a result, the computer codes may take a considerable amount of time to complete a single evaluation. A time tested method of analysis for such models is Monte Carlo simulation. These simulations, however, often require many model evaluations, making this approach too computationally expensive. To limit the number of experimental runs, it is common practice to model the departure as a Gaussian stochastic process (GaSP) to develop an emulator of the computer model. One advantage for using an emulator is that once a GaSP is fit to realized outcomes, the computer model is easy to predict in unsampled regions of the input space. This is an attempt to 'characterize' the overall model of the computer code. Most of the historical work on design and analysis of computer experiments focus on the characterization of the computer model over a large region of interest. However, many practitioners seek other objectives, such as input screening (Welch *et al.*, 1992), mapping a response surface, or optimization (Jones *et al.*, 1998). Only recently have researchers begun to consider these topics in the design and analysis of computer experiments. In this dissertation, we explore a more traditional response surface approach (Myers, Montgomery and Anderson-Cook, 2009) in conjunction with traditional computer experiment methods to search for the optimum response of a process. For global optimization, Jones, Schonlau, and Welch's (1998) Efficient Global Optimization (EGO) algorithm remains a benchmark for subsequent research of computer experiments. We compare the proposed method in this paper to this leading benchmark. Our goal is to show that response surface methods can be effective means towards estimating an optimum response in the computer experiment framework.

Dedication

To my beloved family and friends, especially those willing to supply a couch to sleep on.

Acknowledgements

I would like to express my sincere gratitude to my advisor Geoff Vining. I benefited greatly from his experience in teaching, research, writing, and multi-tasking. I consider him a great mentor and an even greater friend.

I would also like to thank the remainder of my committee: Jeff Birch, Angie Patterson, and Leanna House. Their comments, questions, and suggestions dictated much of my research. As a result, the final product is even better than what I had envisioned years ago.

I would especially like to thank my parents, Cara, Emmah, the McAlister family, the Bertolami family, and all of my friends for supporting my decisions the last twenty-seven years. From those who passed on to the new arrivals soon to come, you have given me a reason to smile. For that, I am forever grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A black-box system is one that uses complex mathematical operations to convert inputs into outputs. Computer experiments use such black-box systems as the basis for generating data. Typically, computer experiments have either one of two possible goals: characterization of the black-box system over a large region of interest (a region of operability), or global optimization of a particular output of interest. Some computer experiments are computationally expensive; they can take hours, days, and sometimes even months to compute. Therefore, a low number of experimental runs is desirable. One way to limit these runs is to build an emulator for the computer code and use prediction techniques of a Gaussian stochastic process to investigate the process without having to sample directly from the computer code.

Using complex mathematical models to describe a physical process that would otherwise be extremely difficult, too costly or sometimes impossible to analyze is not a new concept. Due to advances of computing power, the application of computer codes as a surrogate for reality has increased in recent years. For example, Aslett *et al.* (1998) and Bernardo *et al.* (1992) describe examples in electrical engineering, where they optimize circuit designs through the statistical inference of a computer experiment. Finite element analysis, a common practice of mechanical and aerospace engineers, consists of a computer model for a material or design

that an engineer analyzes for specific results. Chang *et al.* (2001) use finite-element methods to model the flexing and stress shielding of hip prosthesis.

In most applications, complex computer codes, or models, are deterministic. In some cases, simulation experiments introduce random error into the model, but the models themselves are deterministic. Simulations begin with a deterministic model and then add error to the inputs to produce noise on the response. The addition of the random seed as an input transforms a deterministic experiment into a simulation experiment. Our efforts only consider deterministic models. Deterministic models yield identical output(s) given the same inputs, which means the three principles of experimental design for controlling noise and bias; namely replication, randomization, and local control of error (blocking), no longer apply. As a result, we must analyze the data in a different way.

Researchers began analyzing deterministic computer models in the 1970's (McKay, Beckman, and Conover, 1979). The very first approach analyzes the probability distribution of the output $Y(\mathbf{x})$ by treating the inputs $\mathbf{x}$ as random variables, a la Monte Carlo simulation; it proves to be a sufficient method for a low dimension example. However, Monte Carlo simulations often require many model evaluations, making this approach too computationally expensive for complex computer models. A second approach relies on a statistical approximation of the computer model, or emulator, to ease the computational burden of analyzing these complex models. We begin by modeling the deterministic response from the computer model as

$$Response = LinearModel + Departure. \tag{1.1}$$

The departure in Equation 1.1 is often treated as white noise or random experimental error

in traditional response surface methods. Outcomes from a deterministic computer model however, have zero random error. The same outcome will result from the same input, regardless of replication. Thus, the departure is considered systematic and often modeled by a Gaussian stochastic process (GaSP). One advantage for using an emulator is that once a GaSP is fit to realized outcomes, the computer model is easy to predict in unsampled regions of the input space.

A non-Bayesian prediction technique analyzes the computer simulation through investigation of the Best Linear Unbiased Predictor (BLUP) of the response at an untried input. To use this method, one assumes that the model parameters are known; however, in reality they usually are not and must be estimated. A Bayesian approach incorporates this uncertainty from estimation of parameters and allows the investigation of model inaccuracies via a discrepancy error. Both methods, Bayesian and non-Bayesian, treat the analysis of computer experiments through interpolation via some prediction method.

A prediction technique is used to sample over the entire region of operability, which we call 'characterization'. It is an attempt to 'characterize' the overall model. Most of the historical work on design and analysis of computer experiments focus on the characterization of the computer model over a large region of interest. However, many practitioners seek other objectives, such as input screening (Welch *et al.*, 1992), mapping a response surface, or optimization (Jones *et al.*, 1998). Only recently have researchers begun to consider these topics in the design and analysis of computer experiments. The goal for global optimization research is to achieve a good estimate of the optimum response of these computer models, while limiting the number of experimental runs (computer evaluations).

For global optimization, Jones, Schonlau, and Welch's (1998) Efficient Global Optimization (EGO) algorithm remains a benchmark for subsequent research of computer experiments.

Huang *et al.* (2006) present an augmented expected improvement function that extends the EGO scheme to stochastic computer functions. Villemonteix *et al.* (2009) provides a preferable method when confronted with irregular or "bumpy" functions. These recent papers use the EGO algorithm as their barometers of overall performance.

In this dissertation, we explore a more traditional response surface approach (Myers, Montgomery and Anderson-Cook, 2009) in conjunction with traditional computer experiment methods to search for the optimum response of a process, using the core ideas from the key papers in Chapter 2. Vining (2008) provides an overview of adapting RSM techniques to computer experiments. Table 1.1 discusses the contrast between the two methods: RSM and computer experiments. Clearly, the two approaches are quite different from each other. It should surprise very few people that under the traditional computer experiment environment the traditional RSM methods can be ineffective. Thus, we need to be careful about how we apply response surface methods to computer experiments because assumptions and approximations may be invalidated in a new environment.

Table 1.1: Contrast of RSM and Computer Experiments

RSM	Computer Experiment
Series of Experiments	One-shot Experiment
Few Active Factors	Many Factors
Small Design Space	Large Design Space
Low Dimension	High Dimension
Subset of Region of Operability	Entire Region of Operability
Assumes Relatively Smooth Surface	Surface Often Relatively Bumpy
Well Defined Models	Often No Real Model
True Background Noise	No or Artificial Background Noise
Describes Behavior Near Optimum	Often Cannot Describe Behavior Near Optimum

RSM assumes a relatively small number of active factors and a reasonably smooth surface. For this environment, RSM proves to be an effective tool for process improvement. The

consensus in the literature is that response surface methods have little place in the design and analysis of computer experiments. However, the past applications of RSM methods to computer models were not appropriate.

Several studies, including Iman and Helton (1988), show that the performance of response surface techniques over the entire region of operability is poor. The region of operability represents all feasible settings for the inputs of our computer model. Typically, response surfaces over the entire region of operability are extremely bumpy. In such a situation, the Taylor series approximation of RSM is not appropriate. Therefore, response surface methods generally cannot properly characterize the entire region of operability. However, if we only need to characterize in the neighborhood of the optimum, we can concentrate resources better.

By adapting the environment of traditional computer experiments to better accommodate Taylor-series approximations, we propose that response surface methods can be an effective means towards optimization inside the traditional computer experiment domain. More specifically, if we change the philosophy of operating within the entire region of operability to a subset in a neighborhood of the optimum, response surface methods can be used more effectively.

The remainder of the dissertation is organized as follows. Chapter 2 reviews previous work in design and analysis of computer experiments and the research in response surface designs. For computer experiments, we highlight key papers by Jones *et al.* (1998), Kennedy and OHagan (2001), and Sacks, Schiller, and Welch (1989). In the response surface field, we look specifically at the research by Box and Wilson (1951) and Box and Hunter (1957).

Chapter 3 outlines the algorithm used in our methods to adapt response surface methods

in the presence of computer experiments. This chapter introduces two examples from the literature that are closed-form functions where we measure the performance of the algorithm by comparing its optimum estimates to previous methods in the literature. One criticism of these closed form functions is that the results are not directly interpretable. Thus, it is hard to make tractable conclusions for new methods. For this reason, Chapter 3 also includes an analysis on a real computer experiment.

Chapter 4 compares two versions of the algorithm presented in Chapter 3. This chapter compares two different factor selection procedures, the method of steepest ascent and the Box and Meyer method, for the unreplicated fractional factorial designs inside the RSO algorithm. This chapter addresses many of the issues seen in the results from Chapter 3.

Chapter 5 attempts to mimic traditional RSM testing on a physical system. We present a pseudo lack of fit test to examine model transition and the exploration of the white noise of the system. This chapter presents our final recommendation on the structure of the proposed algorithm. Finally, we end with concluding remarks and possible extensions to this research in Chapter 6.

Chapter 2

Review of Literature

The lack of random error produces a contrast between computer and physical experiments. As a result, researchers gravitate towards ad hoc methods to handle this difference. In the first application of computer experiments, McKay, Conover, and Beckman (1979) consider experimental designs for deterministic computer models by applying Monte Carlo techniques. Ten years later the focus of the literature shifted to the characterization techniques of Sacks, Schiller, Welch (1989); Sacks, Welch, Mitchell, and Wynn (1989); Jones *et al.* (1998); Kennedy and O'Hagan (2001); Villemonteix, Vazquez, and Walter (2009).

2.1 Design and Analysis of Computer Experiments

Bayarri *et al.* (2007) provide a systematic validation process governed by the inferential ideas of Kennedy and OHagan (2001) and the Gaussian stochastic process work of Sacks, Welch, Mitchell, and Wynn (1989). Their six step process is as follows:

1. Specify model inputs and uncertain model parameters with associated uncertainty and ranges

2. Determine evaluation criteria

3. Design experiments and collect data (field and simulation)

4. If computer code is slow, develop a fast approximation

5. Analyze and compare computer model output and field data

6. Feedback information into the current validation exercise and feed-forward into future validations

This process is an iterative learning procedure that depends on the underlying needs of the practitioner, i.e. optimization of response, input screening, etc. Steps one and two are usually determined by the problem at hand. There has been extensive research on the development of the remaining steps, which have been outlined in the remainder of this section.

2.1.1 The Gaussian Stochastic Process

The use of Gaussian stochastic processes to model unknown regression functions in a nonparametric way dates back to Kimeldorf and Wahba (1970), Blight and Ott (1975), and O'Hagan (1978). The problem with using many regression techniques is that we must assume that the errors are independent, which does not hold for deterministic computer codes. It should be apparent, that if $\mathbf{x}$ and $\mathbf{x}^*$ are two points close to each other, their errors, $\varepsilon(\mathbf{x})$ and $\varepsilon(\mathbf{x}^*)$, will be close as well. Jones *et al.* (1998) give a thorough explanation of how these errors are correlated.

Sacks, Schiller, Welch (1989) and Sacks, Welch, Mitchell, and Wynn (1989) suggest using a nonparametric regression technique that applies a Gaussian stochastic processes to computer experiments to achieve this relationship. Their primary goal, like most of the earliest work

in the field of computer experiments, was concerned with interpolation of the simulator, i.e., producing efficient predictions. In the absence of true background noise, one would expect some issues when modeling the response. Therefore, we model the response from Equation 1.1 as a realization of a Gaussian stochastic process (a random function $Y(\boldsymbol{x})$),

$$Y(\boldsymbol{x}) = \sum_{j=1}^{k} \beta_j f_j(\boldsymbol{x}) + Z(\boldsymbol{x}), \tag{2.1}$$

where for any finite set of locations $x_1, \ldots, x_k$, a Gaussian stochastic process $Z(\cdot)$ is one in which the vector $Z(x_1), \ldots, Z(x_k)$ has a multivariate normal distribution. In this equation, each f_j is a linear or nonlinear function and the β_j's are unknown coefficients to be estimated. The model in Equation 2.1 consists of two components. The first is a linear regression model to account for the drift in the response (often just a constant). The second component represents the departure (modeling error) from the first component. It is modeled by a Gaussian stochastic process Z$(\cdot)$ that is assumed to have mean zero and covariance:

$$cov(Z(t), Z(w)) = \sigma_Z^2 V(t, w) \tag{2.2}$$

between two Z's at k-dimensional inputs $t = (t_1, \ldots, t_k)$ and $w = (w_1, \ldots, w_k)$ where σ_Z^2 is the process variance and V$(\cdot, \cdot)$ is a correlation function that satisfies the property $V(t, w) = 1$ when $t = w$. Thus, points closer together have a higher correlation; the correlations between errors is related to the distance between the corresponding inputs.

Characterizing a computer code involves predicting the value of $y(x)$ where x is a new input site and $y(x)$ is a realization of random variable $Y(x)$. Thus, the process must satisfy an

ergodicity property that allows valid statistical inference about a process based on a single point, see Cressie (1993) and Santner, Williams, and Notz (2003). Ergodicity requires that the GaSP's are stationary (homogeneous) and cov(h) $\rightarrow$ 0 as h $\rightarrow \infty$, which implies that E(Z($\mathbf{x}$))=0 and its covariance must satisfy:

$$V(\mathbf{t}, \mathbf{w}) = V(\mathbf{t} - \mathbf{w})$$

i.e. all pairs of locations $\mathbf{t}$ and $\mathbf{w}$ with common orientation and common inter-point distance will have the same correlation. There are many possible formulations of this correlation structure, and often the choice depends on the situation. Kennedy and OHagan (2001) discuss the importance of modeling the covariance function for computer experiments and give the following generic form for this structure:

$$V(t, w) = e^{-\sum_{j=0}^{k} \theta_j |t_j - w_j|^{p_j}}. \tag{2.3}$$

The $p_j's$ are smoothing parameters, which give the stochastic process method its major computational advantage over Monte Carlo methods (Helton, 1993). The $\theta_j's$ are roughness parameters which indicate the importance of variable j. Large values of θ_j indicate that the j^{th} input is 'active'. Many different covariance structures exist; however, we use the formulation in Equation 2.3 in our methods.

The work of Sacks, Schiller and Welch (1989) suggest that a linear model component(i.e., independent of x) from Equation 2.1 consisting of only a constant produces results comparable to the linear and quadratic trends. Therefore, Equation 2.1 is written as:

$$Y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}). \tag{2.4}$$

where μ is the mean of the stochastic process and Z($\mathbf{x}$) is Normal(0,σ^2V(t,w)). The parameters μ and σ^2 may not have a direct physical interpretation, but are needed to characterize the computer model. From this model, we can derive the distribution of an n-vector of observed responses $\mathbf{y} = (y_1, \ldots, y_n)$,

$$\mathbf{y} \sim N(\mathbf{1}\mu, \sigma^{\mathbf{2}}\mathbf{R}). \tag{2.5}$$

where $\mathbf{R}$ is an n x n matrix whose (i , j) entry is the covariance shown in Equation 2.2. The expression in 2.5 is used in two main approaches for getting predictions to characterize the simulator. The first main approach, Kriging, uses 2.5 in the development of the best linear unbiased predictor, see Section 2.1.2. This method is seen in Sacks *et al.* (1989), Sacks, Schiller, and Welch (1989), Welch *et al.* (1992), and Jones *et al.* (1998). The second approach, given by Kennedy and O'Hagan (2001), is a Bayesian approach that uses 2.5 for the derivation of the likelihood of $Y(\mathbf{x})$; see Section 2.1.3.

2.1.2 Characterization via the Kriging Method

The Kriging method is a non-Bayesian technique adopted from spatial statistics literature, see Cressie (1993). Sacks *et al.* (1989) give a detailed explanation of the Kriging technique (a linear predictor) for the expression in Equation 2.1. Since the linear regression component in this expression is usually a constant, we will use the formulation in Equation 2.4. To derive the BLUP, consider the linear predictor

$$p = \mathbf{c}^T \mathbf{y}$$

for response,y, at a new point $\boldsymbol{x}^*$. We get the best linear unbiased predictor (BLUP) by taking the $n \times 1$ vector $\mathbf{c}$ to minimize the mean square error (MSE) of $y(\hat{\boldsymbol{x}^*})$

$$MSE(p) = E(p - Y(\boldsymbol{x}^*))^2$$

subject to the unbiased constraint

$$E(\mathbf{c}^T \boldsymbol{y}) = E(Y(\boldsymbol{x})).$$

Assuming known correlation parameters (smoothing and roughness), we can calculate the BLUP and its variance as

$$\hat{y}(\boldsymbol{x}^*) = \hat{\mu} + \mathbf{r^T R^{-1}}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{2.6}$$

and

$$var(\hat{y}(\boldsymbol{x}^*)) = \hat{\sigma}^2[1 - \mathbf{r^T R^{-1} r} + \frac{\mathbf{(1 - 1^T R^{-1} r)^2}}{\mathbf{1^T R^{-1} 1}}]. \tag{2.7}$$

where $\hat{\mu}$ and $\hat{\sigma^2}$ in Equations 2.6 and 2.7 are the maximum likelihood estimates given by

$$\hat{\mu} = \frac{\mathbf{1^T R^{-1} y}}{\mathbf{1^T R^{-1} 1}}$$

and

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^{\mathbf{T}}\mathbf{R}^{-\mathbf{1}}(\mathbf{y} - \mathbf{1}\hat{\mu})}{n}.$$

Sacks *et al.* (1989) give a full derivation of this method. The BLUP and its variance provided in Equations 2.6 and 2.7 give a prediction for $\hat{y}(\boldsymbol{x}^*)$, and thus serves as an interpolator of the computer model. We use these predictions as a cheap approximation of the computer simulation; we call this method characterization. A model with a strong characterization has BLUP's that adequately approximate the response surface of the computer simulation. The equations listed above use maximum likelihood to estimate the models n+2 parameters (μ, σ^2, and the n parameters in the correlation structure) which could be expensive to compute. To reduce computing time, Welch *et al.* (1992) propose an algorithm to introduce parameters of inputs sequentially.

2.1.3 Characterization via the Bayesian Method

The biggest advantage of using a Bayesian technique over the Kriging approach is that it allows some of the uncertainty attributed to estimation. This is addressed in Chapter 4. Kennedy and O'Hagan (2001) present a Bayesian technique that also attempts to correct for any inadequacy of the model via the discrepancy between observed data (reality) and the model predictions (emulator). Due to computation limitations, their method uses maximum likelihood estimates for parameters of the correlation functions. Although this method is not fully Bayesian, they found that the results were comparable to a full Bayesian analysis.

Before understanding the emulator they are building, it is easier to take a look at the physical system that they are trying to understand,

$$D(\mathbf{x}) = \zeta(\mathbf{x}) + \mathbf{e}(\mathbf{x}) \tag{2.8}$$

where $\zeta(\mathbf{x_i})$ represents the response of the actual physical process and $e(\mathbf{x_i})$ represents the observational error. Since the model we are using to approximate the actual process is deterministic, we usually do not have any observational error (unless there is field data). When field data exists we can decompose $\zeta(\mathbf{x_i})$ into two parts, the code output and the inadequacy of the model (mostly due to the uncertainty stemming from a collection of inputs left out of $\mathbf{x}$). This model is:

$$\zeta(\mathbf{x}) = \rho\eta(\mathbf{x}, \theta) + \delta(\mathbf{x}), \tag{2.9}$$

ρ is an unknown regression parameter, $\eta(\mathbf{x_i}, \theta)$ is the simulator output represented by a zero-mean stationary GaSP with unknown variance σ_Z^2 and correlation function $R_Z(\cdot)$, and $\delta(\mathbf{x_i})$ expresses the inadequacy of the model via a discrepancy error that is represented by another zero-mean stationary GaSP with unknown variance σ_δ^2 and correlation function $R_\delta(\cdot)$. Using the same simplification in expression 2.4, the model in 2.9 can be simplified to

$$Y(\mathbf{x}) = \mu + \mathbf{Z}(\mathbf{x}) + \delta(\mathbf{x}) + \mathbf{e}(\mathbf{x}). \tag{2.10}$$

Santner, Williams, and Notz (2003) show that when variances and correlation functions of the Gaussian stochastic processes are known, the likelihood of our response $Y(\cdot)$ is normally distributed. Higdon *et al.* (2008) provide detailed instruction on the derivation of the posterior distribution assuming all parameters unknown for their fully Bayesian approach. Their formulation for the likelihood of the response follows directly from the Model 2.10,

$$L(\mathcal{D}|\text{parameters}) \propto |\Sigma_{\mathcal{D}}{}^{-\frac{1}{2}}| exp\{-\frac{1}{2}(\mathcal{D}-\mu\mathbf{1})'\boldsymbol{\Sigma}_{\mathcal{D}}{}^{-\mathbf{1}}(\mathcal{D}-\mu\mathbf{1})\}. \tag{2.11}$$

where $\mathcal{D} = (\mathbf{y}, \eta)$ is a (n + m) vector of field observations (n) and simulation runs (m),

$$\boldsymbol{\Sigma}_{\mathcal{D}} = \boldsymbol{\Sigma}_Z + \begin{pmatrix} \boldsymbol{\Sigma}_y + \boldsymbol{\Sigma}_\delta & 0 \\ 0 & 0 \end{pmatrix} \tag{2.12}$$

where $\boldsymbol{\Sigma}_y$ is an $n \times n$ observation covariance matrix, $\boldsymbol{\Sigma}_\delta$ is an $n \times n$ discrepancy covariance matrix and $\boldsymbol{\Sigma}_Z$ is an $(n+m) \times (n+m)$ covariance matrix from the GaSP $Z(\cdot)$. By using conjugate or non-informative priors on unknown parameters we can derive, in closed form, the posterior distribution from

$$\pi(\text{parameters}|\mathcal{D}) \propto L(\mathcal{D}|\text{parameters})\pi(\text{parameters}) \tag{2.13}$$

where π(parameters) is the joint prior distribution of all of the unknown parameters. In some cases, we cannot express Equation 2.13 in closed form, but can explore it through Markov chain Monte Carlo (MCMC). Once a posterior distribution is revealed, whether it be analytically or assessed stochastically, we can perform predictive inference on reality.

The Bayesian approach to prediction is no different from the ideas in the Kriging and Monte Carlo approaches. For these approaches, an estimate for the emulator's mean function and its variance are calculated by the BLUP and Monte Carlo sample mean respectfully. To make predictive inferences with the Bayesian approach, the posterior predictive distribution is needed,

$$\begin{aligned} \pi(\mathbf{y}^*|\mathbf{y}) &= \int \pi(\mathbf{y}^*, \theta|\mathbf{y})\mathbf{d}\theta \\ &= \int \pi(\mathbf{y}^*|\theta)\pi(\theta|\mathbf{y})\mathbf{d}\theta \end{aligned} \tag{2.14}$$

where $(\mathbf{y}^*)$ is an unknown observation and $\pi(\theta|\mathbf{y})$ is the posterior distribution of all model parameters. From Equation 2.14 we can calculate the posterior predictive mean and variance using common formulas for means and variance of conditional distributions,

$$E(\mathbf{y}^*|\mathbf{y}) = E(E(\mathbf{y}^*|\mathbf{y}, \theta)) \tag{2.15}$$

and

$$var(\mathbf{y}^*|\mathbf{y}) = E(var(\mathbf{y}^*|\mathbf{y}, \theta)) + var(E(\mathbf{y}^*|\mathbf{y}, \theta)). \tag{2.16}$$

The kriging and Bayesian characterization methods, discussed thus far, often lead to different results, except when the stochastic process, $Z(\cdot)$, is Gaussian and improper priors are placed on the parameters in Equation 2.4. This special case presents an old result that shows the BLUP is the limit of the Bayesian predictor as the prior variances tend to infinity (Sacks *et al.* (1989)).

We mentioned earlier that when model runs are expensive to complete, the model evaluations that are needed for Monte Carlo techniques tend to be infeasible. This is no different for MCMC techniques. Bayarri *et al.* (2007) recommend using the method provided by Kennedy and O'Hagan (2001) or what they call the 'modular-MLE' approach. This analysis fixes the

GaSP hyper-parameters at their MLEs, leaving only the variances and calibration parameters as random. In terms of prediction, the 'modular-MLE' approach produces similar results as a full Bayesian analysis and significantly cuts down computation time. Higdon *et al.* (2008) use basis representations (principal components) to reduce dimensionality inside the fully Bayesian approach, which can be extended to the "modular-MLE" framework for further reduction of the computational burden.

2.1.4 Characterization via Bayes Linear Method

At times, it is very difficult to specify full probability models for Bayesian analysis. A useful alternative is the Bayes linear method that relies on expectation, rather than probability. This means that we specify the moments of random variables, not distributions, to asses data and we do not use Bayes rule to calculate posterior or 'adjusted' moments. Based on Bayes linear theory, the adjusted expectation and variance of y^* given y are as follows (Craig *et al.*, 2001):

$$E_y(y^*) = E(y^*) + \text{cov}(y^*, y)\text{var}(y)^{-1}(y - E(y)) \tag{2.17}$$

and

$$\text{var}_y(y^*) = \text{var}(y^*) + \text{cov}(y^*, y)\text{var}(y)^{-1}\text{cov}(y^*, y). \tag{2.18}$$

We may apply the same methodology to learn about the simular, given simular evaluations. In such situations, however, the Bayes linear adjusted expectation and variance are equivalent to Equations 2.6 and 2.7 produced by the Kriging prediction technique.

2.1.5 Designs for Computer Experiments

A common goal of experimental design is to collect the most information with the smallest number of design points to accurately estimate the model parameters, i.e. the most "bang" for your "buck". For computer experiments, we are concerned with the choice of points at which the computer code is run. The easiest solution to the design problem is to perform designs already established in RSM, like factorial or central composite designs. Iman and Helton (1988) produced response surfaces from fractional factorial designs and found that for a number of examples the RSM approach did not adequately represent the computer code output but was useful in ranking the importance of the inputs.

Most computer codes are deterministic, therefore we need designs that do not take more than one observation for any set of inputs. Because we are wary to make prior judgements about true relationship between inputs and response, we need designs that allow one to fit a variety of models and designs that provide the most information possible about the experimental region.

The last argument suggests the use of a 'space filling' experimental design, which agrees with the works of Sacks, Schiller and Welch (1989), Morris *et al.* (1993), Morris and Mitchell (1995), and Bates *et al.* (1996). The reason for the wide use of 'space filling' designs for computer experiments is that they ensure the representation of an active input when only a few inputs are important (effect sparsity).

Arguably, the most common 'space filling' design seen in literature is the latin hypercube design (LHD). Latin hypercube sampling (LHS) is a form of stratified sampling that can be applied to multiple variables. The key is that the design when projected to one dimension has no overlapping points. Figure 2.1 shows an example of an LHD that samples five design

point of two variables x_1 and x_2.

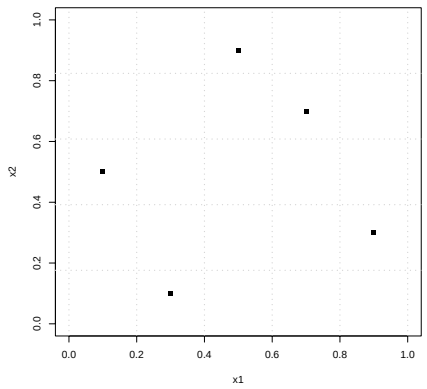

Figure 2.1: A space filling latin hypercube design

McKay, Beckman, and Conover (1979) compare simple random sampling to two other types of sampling plans: stratified sampling and LHS. Their results show that the LHS designs are the more favorable designs for the prediction of a response. Santner, Williams, and Notz (2003) give a comprehensive look at many of the designs discussed in the literature, including space filling and criterion based designs.

One complaint of the latin hypercube design is that it does not have strong sequential properties. For instance, if an LHD was performed and more points were desired, then a new LHD is required to perform the analysis. We desire an experimental design that can add points sequentially via an iterative procedure. The sobol sequence achieves this by generating a uniform distribution in probability space and 'fills in' previously unsampled regions of the probability function, see Santner, Williams, and Notz (2003).

2.1.6 Global Optimization Methods

Optimization using Kriging or Bayesian methods is an iterative process that evaluates the function at a point that optimizes a prespecified criterion which is based on previous model evaluations. Among the many possible criteria, we feel that two stand out from the rest: conditional minimizer entropy and expected improvement.

Jones, Schonlau, and Welch (1998) balance the search for a local and global optimum by utilizing the expected improvement 'figure of merit'. The expected improvement criterion is calculated by taking the expectation of:

$$E[I(\mathbf{x})] = E[max(f_{min} - Y, 0)], \tag{2.19}$$

where $f_{min} = min(y_1, ..., y_k)$ is the current best function value and $Y \sim N(\hat{y}, s^2)$. Jones *et al.* (1998) use Kriging techniques to introduce the prediction $\hat{y}$ (BLUP) and its variance s^2. After applying calculus to the right hand side of Equation 2.19, a closed form solution for the expected improvement is found:

$$E[I(\mathbf{x})] = (f_{min} - \hat{y})\Phi\left(\frac{f_{min} - \hat{y}}{s}\right) + s\phi\left(\frac{f_{min} - \hat{y}}{s}\right),$$

where ϕ and Φ represent the standard normal density and distribution function. Once in closed form, Jones *et al.* (1998) use a branch-and-bound algorithm called EGO (for Efficient Global Optimization) to maximize the expected improvement to find a global optimum.

To calculate the conditional minimizer entropy criterion, we first define the entropy of a

discrete random variable, U,

$$H(U) = -\sum_{u} P(U = u) \log P(U = u)$$

which measures the spread of the distribution of U. Theory allows us to expand this formula to produce the conditional entropy of U given another random variable, V,

$$H(U|V) = -\sum_{v} P(V = v) H(U|V = v)$$

and the conditional entropy of U given V and β,

$$H(U|V, \beta) = -\sum_{v} P(V = v|\beta) H(U|V = v, \beta).$$

For optimization problems, we calculate the conditional entropy of Z given the vector of evaluation results, f_n, and the potential result of an evaluation y_i, F_c,

$$H(c) = H(Z|F_n, F_c)$$

where

$$H(Z|F_n, F_c) = \sum_{i=1}^{M} P(F_c = y_i|F_n = f_n) H(Z|F_n = f_n, F_c = y_i)$$

and

$$H(Z|F_n = f_n, F_c = y_i) = -\sum_{z} P(Z = z|F_c = y_i, F_n = f_n) \log P(Z = z|F_c = y_i, F_n = f_n).$$

A full derivation of the conditional entropy minimizer criterion is seen in Villemonteix, Vazquez, and Walter (2009). The authors use this criterion for the IAGO (for Informational Approach to Global Optimization) algorithm that measures the anticipated uncertainty remaining in Z given the candidate evaluation point $\mathbf{x}$ and the result f_n of the previous evaluations.

Over the last ten years, the expected improvement criterion has been the subject of many publications (e.g., Jones *et al.* (1998), Williams *et al.* (2000), Haung *et al.* (2006), and Villemonteix *et al.* (2009)). The Efficient Global Optimization (EGO) algorithm by Jones, Schonlau, and Welch (1998) uses this criterion and stands as the benchmark for comparison in the field. Williams, Santner, and Notz (2000) introduce an extension to the expected improvement algorithm of Jones, Schonlau, and Welch (1998). They follow a Bayesian framework while incorporating the GaSP from Sacks, Welch, Mitchell, and Wynn (1989) with a Matern class correlation structure for finding the optimum of $E_y(\mathbf{x}_c, X_e)$ (the objective function). Their method computes the posterior expected improvement over the current optimum for each untested site, however, the computation of the expectation is relatively expensive. The method presented by Williams, Santner, and Notz (2000) is one of few that apply a full Bayesian approach to optimization of computer experiments, however, Huang *et al.* (2006) point out its numerous deficiencies and present an augmented expected improvement function that extends the EGO scheme to stochastic computer functions.

Lastly, Villemonteix, Vazquez, and Walter (2009) introduce the Informational Approach to Global Optimization (IAGO) algorithm, which is a competitive alternative to EGO using the conditional minimizer entropy criterion. The IAGO and EGO algorithms differ only by the sampling criterion for choosing the next evaluation. It is important to note that IAGO characterizes the computer experiment over the entire region of operability significantly

better than EGO, especially for highly irregular test functions. However, there is no evidence in a drop off in performance for estimating the global optimum, which prompts us to use the less complex expected improvement criterion as our basis for comparison.

2.2 Response Surface Methodology

Box and Wilson (1951) laid the foundations for the entire field of RSM. Their paper introduced a sequential strategy based on low-order Taylor series approximation that seeks the 'optimal' processing conditions through the path of steepest ascent and has become the core of industrial experimentation. Box and Lin (1999) illustrate the application of RSM in a training example of paper helicopters. They state that industrial innovation is a process of dynamic learning, not static. In experimentation, each subset of observations supply information for the next group and sometimes even the objective itself may change. Box (1999) outlines a discussion on sequential learning and its effects on teaching and learning. RSM and its sequential learning strategy helps us implement a great deal of practical experience, which includes an understanding of the process.

RSM begins with an initial experiment in the neighborhood of the recommended set of operating conditions. The experimenter must determine the set of factors and their levels for this experiment. At this point, we require a strict first-order model for the response to minimize design size. It is crucial to use as small an experimental design as possible at this stage in order to save resources for the latter stages of experimentation. The purpose of the initial experiment is to screen out the inactive factors and to find the direction for better results. The full and fractional factorial designs are usually the designs of choice for the initial experiment.

When the experimental error is small for a k-factor experiment, we may explore a small sub region of the entire experimental region with only a few experiments. The results of this experiment allow you to move in the direction of potential improvement, which is essentially a 'path' to find the region or neighborhood of the optimum (path of steepest ascent). Consider the fitted first-order response surface model,

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_k x_k, \tag{2.20}$$

where $\hat{y}$ is the predicted response, x_i represents the i^{th} independent factor, $\hat{\beta}_0$ is the estimated intercept, and the individual $\hat{\beta}_i$ are the estimated coefficients for the i^{th} independent factor. The method of steepest ascent seeks the points $x_1, x_2, \ldots, x_k$ that produce the maximum estimated response over all points that are a fixed distance r from the center of the design. As a result, the optimization problem involves the use of a Lagrange multiplier (λ) subject to the constraint, $\sum_{i=1}^{k} x_i^2 = r^2$. It is assumed that one wishes to move from the origin of your design r units on the surface of a hyper-sphere such that a maximum increase in the response is achieved. By taking partial derivatives with respect to x_j of

$$L = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_k x_k - \lambda(\sum_{i=1}^{k} x_i^2 - r^2)$$

and setting it equal to zero

$$\frac{\partial L}{\partial x_j} = \hat{\beta}_j - 2\lambda x_j = 0$$

we get the coordinate x_j of the path of steepest ascent $x_j = \frac{\hat{\beta}_j}{2\lambda}$. By selecting the step size in one of the variables, a 'key' factor (the input with the largest estimated coefficient), we

determine the increment of the other process variables. Define a key factor as x_i. With a one unit increase in the key factor, the change in variable x_j is

$$\Delta x_j = \frac{\hat{\beta}_j}{\hat{\beta}_i}\Delta x_i. \quad j = 1, 2, \ldots, k \quad i \neq j \tag{2.21}$$

Equation 2.21 shows that increasing the key factor unit by unit produces the path of steepest ascent. Typically, steepest ascent works best when the initial experiment is far from the region with optimum conditions, because here the first order approximations are most reasonable. As we near the optimum, interactions and pure quadratics begin to dominate, the steepest ascent method becomes less effective. When the effectiveness diminishes, the use of a fitted second order response surface model is recommended for finding optimum conditions in the response. The fitted second-order model is

$$\hat{y} = \hat{\beta}_0 + \sum_{i=1}^{k} \hat{\beta}_i x_i + \sum_{i=1}^{k} \hat{\beta_i} i {x_i}^2 + \sum \sum_{i<j=2}^{k} \hat{\beta_{ij}} x_i x_j. \tag{2.22}$$

Figure 2.2 displays the geometric nature of the second-order function in Equation 2.22. The model in Equation 2.22 and its form in Figure 2.2 will become vital in Section 2.2.1 as we discuss the location of optimum conditions, which occur at the center of the system or stationary point. In Figure 2.2, the stationary point is a maximum response.

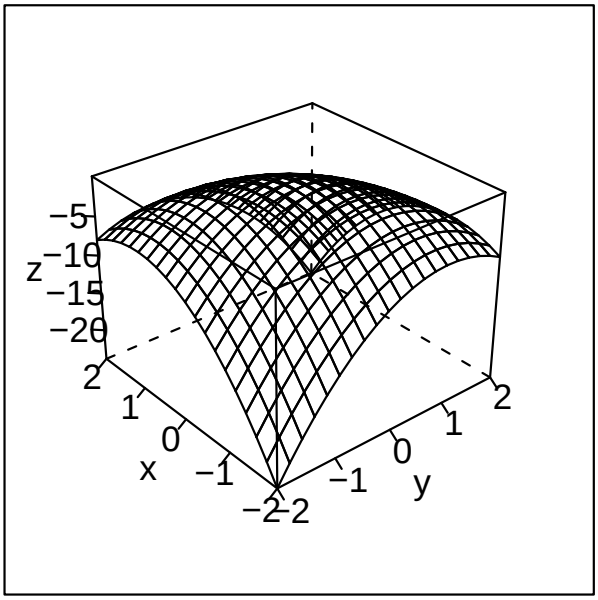

Figure 2.2: Second order system displaying a maximum

Myers, Montgomery, and Anderson-Cook (2009) discuss many second-order designs, but the most common design used in industry is the central composite design (CCD). The CCD consists of a two level factorial or fraction (resolution V) combined with 2k axial runs and n_c center runs. The design matrix of a CCD is constructed as

Table 2.1: Central Composite Design Matirx

x_1	x_2	$\cdots$	x_k
$\pm\mathbf{1}$	$\pm\mathbf{1}$	$\cdots$	$\pm\mathbf{1}$
α	0	$\cdots$	0
$-\alpha$	0	$\cdots$	0
0	α	$\cdots$	0
0	$-\alpha$	$\cdots$	0
$\vdots$	$\vdots$		$\vdots$
0	0	$\cdots$	α
0	0	$\cdots$	$-\alpha$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$

where $\pm\mathbf{1}$ is a vector of factorial runs, α is the axial run value, and $\mathbf{0}$ is a vector of n_c center runs. The relationship between sequential experimentation and the CCD should be apparent. The factorial points represent a variance optimal first order design. The axial runs yield information about the pure quadratic terms and allows for efficient estimation of both the pure quadratics and the main effects. The center runs provide information on the curvature or lack of fit in the model. The CCD can be built up sequentially through these components. One question about central composite designs is the value associated with α. It can take on any value but is usually selected with some design property in mind. More specifically, α is chosen to make designs spherical, cuboidal or rotatable.

A spherical design is one where all points, except for center runs, lay on the surface of a sphere. The choice for α in these designs is $\sqrt{k}$. An $\alpha = 1$, usually chosen due to constraints on the design levels of factors, yields a cuboidal design region called the face-centered cube. The last property, rotatability, was derived by Box and Hunter (1957).

Rotatability is an important property for second order designs as it requires that the scaled prediction variance is equal for all points $\mathbf{x_0}$ that are the same distance from the center of the design. Box and Hunter (1957) found that first order orthogonal designs, like factorial designs, are rotatable. For second order designs, two conditions are required for rotatability; they are:

1. All odd moments through order four are zero
2. The ratio of moments $[iiii]/[iijj] = 3$ $(i \neq j)$

Therefore, a rotatable central composite design requires that

$$\frac{[iiii]}{[iijj]} = \frac{F + 2\alpha^4}{F} = 3 \quad or \quad \alpha = \sqrt[4]{F}$$

where F indicates the number of factorial points in the design. Box and Hunter (1957) provide a table that includes α values for rotatable CCD's. Some of these values may be impractical to use in practice, i.e. a rotatable design with three factors has an $\alpha = 1.682$, which may be hard to produce. Values close to the designated α are considered near-rotatable.

Box and Hunter (1957) also develop a strategy for orthogonal blocking where design points are assigned to blocks so that the block effects are orthogonal to all coefficients of the model to control extraneous sources of variance. A method often used when experimental runs cannot be performed under homogenous conditions, but due to the deterministic nature of computer simulations this principal is no longer needed.

Along with the increased use of computer experiments, as computing power increases the more people gravitate towards computer generated designs as an alternative to the standard response surface designs (factorial, CCD, Box-Behnkin, etc.). These computer-generated designs tend to be primarily based on the alphabetic optimality criterion introduced by Keifer (1959). Alphabetic optimal criteria often choose experimental designs to achieve certain properties in the moment matrix,

$$\mathbf{M} = \frac{\mathbf{X'X}}{N},$$

where N is the number of observations. The best-known and most often used criterion is D-optimality. For D-optimality, we aim to minimize the volume of the confidence ellipsoid

around the regression coefficients. This is proportional to the determinant of the moment matrix, thus a D-optimal design is one that minimizes $N^p|(\mathbf{X}'\mathbf{X})^{-1}|$. The first order orthogonal designs used in screening experiments are D-optimal designs. For central composite designs a spherical α yields the most preferable D-optimal design. The literature shows parallels between alphabetic optimal designs and Bayesian optimal designs for computer experiments. In some cases, Bayesian and non-Bayesian criteria yield equivalent results (Chaloner and Verdinelli (1995)). Myers, Montgomery, and Anderson-Cook (2009) discuss all alphabetic criteria for constructing, evaluating, and comparing experimental designs.

Box and Draper (1975) provide a valuable list of 14 criteria for what constitutes a 'good' response surface design. They argue for the use of design robustness (compromising among several criteria) rather than one single optimality measure. Ultimately, they argue that "classical" designs such as the 2^{k-p} fractional factorial and the CCD are robust and should be preferred in practice to designs optimal in terms of a single variance property.

2.2.1 The Analysis of Second-Order Response Surfaces

In the previous section, we indicated that steepest ascent becomes less effective as we near the optimum response. The interactions and pure quadratics tend to dominate and the first order models used in the beginning stages of RSM are inadequate. As a result, a fitted second-order response surface model is a reasonable choice for finding optimum conditions in the response.

There exists many ways of determining the response surface and its stationary point (optimum response) and the nature of the system is often best described through the combination of analytical and graphical techniques. We discuss two analytical methods for determining a stationary point for second-order models: canonical analysis and desirability functions.

Canonical Analysis

For canonical analysis, let us consider the fitted second order model again in Equation 2.22, this time in matrix notation,

$$\hat{y} = \beta_0 + \mathbf{x}'\mathbf{b} + \mathbf{x}'\hat{\mathbf{B}}\mathbf{x} \tag{2.23}$$

where β_0, $\mathbf{b}$, and $\hat{\mathbf{B}}$ are the estimates of the intercept, linear, and second-order coefficients, respectively. When we differentiate $\hat{y}$ in Equation 2.23 with respect to $\mathbf{x}$,

$$\frac{\partial \hat{y}}{\partial \mathbf{x}} = \mathbf{b} + \mathbf{2}\hat{\mathbf{B}}\mathbf{x}$$

,

and set the derivative equal to zero, we can solve for the stationary point,

$$\mathbf{x_s} = -\frac{\mathbf{1}}{\mathbf{2}}\hat{\mathbf{B}}^{-\mathbf{1}}\mathbf{b}. \tag{2.24}$$

Canonical analysis uses the eigenvalue decomposition of $\widehat{\mathbf{B}}$ to describe the nature of the stationary point. This behavior is determined from the signs of the eigenvalues of the matrix $\widehat{\mathbf{B}}$. Let the $k \times k$ matrix $\mathbf{C}$ be the matrix whose columns are normalized eigenvectors associated with the eigenvalues of $\widehat{\mathbf{B}}$. If $\mathbf{\Lambda}$ is a symmetric matrix, we know that

$$\mathbf{\Lambda} = \mathbf{C}'\widehat{\mathbf{B}}\mathbf{C}$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\widehat{\mathbf{B}}$. As in factor analysis, rotation of canonical weights can improve interpretability of the stationary point solution. If we transform the second-order model in Equation 2.23 to a new center, i.e. the stationary point, $\mathbf{z} = \mathbf{x} - \mathbf{x_s}$, and rotate the axis with respect to $\mathbf{C}$, $\mathbf{w} = \mathbf{C}'\mathbf{z}$, we get

$$\begin{aligned} \hat{y} &= \beta_0 + (\mathbf{z} + \mathbf{x_s})'\mathbf{b} + (\mathbf{z} + \mathbf{x_s})'\widehat{\mathbf{B}}(\mathbf{z} + \mathbf{x_s}) \\ &= \hat{y_s} + \mathbf{z}'\widehat{\mathbf{B}}\mathbf{z}. \end{aligned}$$

From equation 2.24, the rotation gives

$$\begin{aligned} \hat{y} &= \hat{y_s} + \mathbf{w}'\mathbf{P}'\widehat{\mathbf{B}}\mathbf{P}\mathbf{w} \\ &= \hat{y_s} + \mathbf{w}'\mathbf{\Lambda}\mathbf{w}. \\ &= \hat{y_s} + \sum_{i=1}^{k} \lambda_i w_i^2 \end{aligned} \tag{2.25}$$

where $\hat{y_s}$ is the estimated response at the stationary point and $\lambda_1, \lambda_2, \ldots, \lambda_k$ are the eigenvalues of $\widehat{\mathbf{B}}$. Equation 2.25 describes the nature of the stationary point and the behavior around it. Examining the signs of the eigenvalues give a better understanding of the response system:

1. If all eigenvalues are negative, the stationary point yields a maximum response.
2. If all eigenvalues are positive, the stationary point yields a minimum response.
3. If the eigenvalues are mixed in sign, the stationary point yields a saddle point.

Ridge Analysis

Canonical analysis is generally inadequate for multidimensional surfaces. It can be performed for $k > 2$, but lacks any graphical capabilities. Ridge analysis is an extension to canonical analysis that also portrays the behavior of surfaces and optimal regions graphically without holding variables fixed. A ridge displays a path of steepest ascent in the direction of the optimum response (or descent if a minimum response is desired). It is a constrained optimization procedure that fixes $x'x = R^2$ and maximizes Equation 2.23 subject to this constraint.

Mathematically, the points of a ridge are determined by differentiating Equation 2.23 with respect to x, equating to zero, and solving for x. The resulting equation is

$$x = -(\widehat{\mathbf{B}} - \lambda \mathbf{I})^{-\mathbf{1}}\mathbf{b}, \tag{2.26}$$

where λ is the Lagrangian multiplier and $\mathbf{b}$, and $\hat{\mathbf{B}}$ are the estimates of the linear, and second-order coefficients, respectively. If $\lambda_1, \lambda_2, \ldots, \lambda_k$ are the ranked eigenvalues of $\widehat{\mathbf{B}}$ then we can describe the nature of ridge and the surface around it. Examining the signs of the eigenvalues give a better understanding of the response system:

1. The maximum ridge is defined by $\lambda \geq \lambda_1$.
2. The minimum ridge is defined by $\lambda < \lambda_k$.
3. Secondary ridges, i.e. local optima, are defined by $\lambda_j < \lambda < \lambda_{k+1}$.

Desirability Function

The desirability function method is the most popular technique for expressing multiple objectives or responses. This approach is to first convert each response, Y_i, into an individual desirability function d_i. This desirability function varies over the range of

$$0 \leq d_i \leq 1$$

where $d_i = 1$ indicates a response at its goal or target and $d_i = 0$ shows that the response is outside an acceptable region. Design variables are then chosen to maximize the overall desirability, D,

$$D = \left(\prod_{i=1}^{m} d_i \right)^{\frac{1}{m}}, \tag{2.27}$$

where m denotes the number of responses (only one in our case). A solution with the highest overall desirability provides an estimate of the optimum response. Depending on whether a particular response is to be maximized, minimized, or assigned a target value, different desirability functions d_i can be used. Derringer and Suich (1980) discuss in detail the different classes of desirability functions. First, let L_i, U_i and T_i be the lower, upper, and target values, respectively, that are desired for response y_i. We then define three useful classes of desirability functions:

Targeted Response

If a targeted response is desired, the individual desirability is defined as

$$d_i = \begin{cases} 0 & \text{if } \hat{y}_i < L_i, \\ \left(\frac{y_i - L_i}{T_i - L_i}\right)^{r_1} & \text{if } L_i \leq y_i \leq T_i, \\ \left(\frac{U_i - y_i}{U_i - T_i}\right)^{r_2} & \text{if } T_i \leq y_i \leq U_i, \\ 0 & \text{if } y_i > U_i. \end{cases}$$

where r_i determines the importance of hitting the target. Choosing $r_i > 1$ places more emphasis on being close to the target value, and choosing $0 < r_i < 1$ makes this less important.

Maximize Response

If a response is to be maximized, the individual desirability is defined as

$$d_i = \begin{cases} 0 & \text{if } y_i < L_i, \\ \left(\frac{y_i - L_i}{T_i - L_i}\right)^{r} & \text{if } L_i \leq y_i \leq T_i, \\ 0 & \text{if } y_i > T_i. \end{cases}$$

where T_i is interpreted as a large enough value for the response.

Minimize Response

If a response is to be minimized, the individual desirability is defined as

$$d_i = \begin{cases} 0 & \text{if } y_i < T_i, \\ \left(\frac{U_i - y_i}{U_i - T_i}\right)^{s} & \text{if } T_i \leq y_i \leq U_i, \\ 0 & \text{if } y_i > U_i. \end{cases}$$

where T_i is interpreted as a small enough value for the response. The desirability approach can be summarized by the following steps:

1. Conduct experiment and fit response models for all m responses.

2. Determine individual desirability functions, as defined above, for all m responses.

3. Maximize the overall desirability, D.

The nature of the system around the stationary point with a maximum overall desirability yields an estimate of the optimum response.

Chapter 3

Optimization of Black-Box Systems via a Hybrid Response Surface Methodology

3.1 Introduction

Response surface methodology (RSM) has been a useful statistical technique for building, improving, and optimizing processes. RSM is a sequential strategy based on low-order Taylor series approximation that seeks the 'optimal' processing conditions. It is comprised of three distinct phases that are usually performed inside a region of operability. A region of operability is a designated region that a certain process allows. This region is usually expansive and we would generally not want to perform a single experiment over this entire region.

The first phase of RSM builds an experiment in some subset of the region of operability that we call the the region of experimentation. We usually refer to this first phase as the screening experiment, which is an experiment designed to eliminate unimportant factors. The next phase uses the first-order model estimated by the screening experiment, and the method of steepest ascent to move in the direction of a local optimum. Steepest ascent conducts a

series of runs along a path until no improvement is found, which results in a neighborhood around better operating conditions. This sequential experimentation strategy is iterated until we reach a reasonable neighborhood of the optimum. The sequential experimental process usually culminates in a second-order model when the experimenter believes that he/she is in the neighborhood of 'optimum' process conditions, which is the final phase. By 'optimum' we mean much better operating conditions than initially and not necessarily the best. Figure 3.1 gives an overview of the sequential learning strategy.

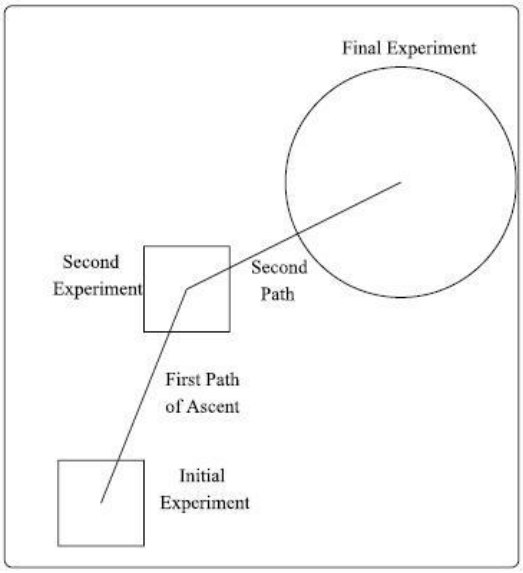

Figure 3.1: Sequential Learning Strategy of RSM

Our methods merge traditional computer experiment and response surface methods through an algorithm. The current focus in the design and analysis of computer experiments requires searching the entire region of operability, whereas, the response surface world focuses on a subset of the entire region. Our algorithm uses knowledge about the process, previous data, and the characterization to help pinpoint a subset of the entire region of operability for the use of RSM practices. We then construct our initial experiment of RSM in this subset region

and take advantage of the sequential learning strategy of RSM to search for an optimum response.

In this chapter, we will discuss a hybrid response surface method for optimizing computer experiments, which is introduced in Section 3.2. Section 3.3 introduces two examples from the literature that are closed-form functions where we measure the performance of the algorithm by comparing its optimum estimates to previous methods in the literature. This section also includes an analysis on a real computer experiment that simulates an ethanol-water distillation column commonly seen in chemical engineering. Finally, we conclude with a few comments in Section 3.5.

3.2 Response Surface Optimization Algorithm

1. **Initial Characterization** : To characterize a computer model, we start with a set of simulated outcomes from a computer experiment based on a space-filling design of the model input space. The reason for the wide use of 'space filling' designs for computer experiments is that they ensure the representation of the active inputs in the emulator itself. We use a sobol sequence space filling design because of its desirable sequential properties. The general rule of thumb for design size is 10k (where k is the number of 'active' variables). Our method uses a space-filling design on the order of 4k experimental runs.

2. **Validation** A leave-one-out cross validation technique is used to determine how the characterization in Step 1 will generalize to an independent data set, i.e. how accurate the predictions would perform in practice. As the name suggests, one observation (validation data) is extracted from the original data set and the model is characterized

with $n-1$ observations (training data). This is repeated such that each observation is used once in a validation data set. As a diagnostic, confidence intervals are built around the the predictions from the training data. We then use the validation data to summarize the performance of the GaSP model; data within prediction limits indicates good performance.

Let $y(\hat{\boldsymbol{x}}_0)$ be the predicted response of the computer model at location $\boldsymbol{x}_0$. The variance of $y(\hat{\boldsymbol{x}}_0)$ from the model in Equation 2.5 is,

$$\sigma^2 \boldsymbol{x}_0'(\boldsymbol{X}'\mathbf{R}^{-1}\boldsymbol{X})^{-1}\boldsymbol{x}_0.$$

Consider a cross-validation data set. Let y_i be the actual observed response in the data set from the computer code. Let $\hat{y}_i$ be the prediction from the GaSP model. An estimate of σ^2 is

$$\hat{\sigma^2} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

where n is the number of observations in the cross validation set. The basis for the confidence interval for any $y(\hat{x}_0)$ is

$$\hat{\sigma^2} \boldsymbol{x}_0'(\boldsymbol{X}'\mathbf{R}^{-1}\boldsymbol{X})^{-1}\boldsymbol{x}_0. \tag{3.1}$$

3. **Re − characterization** The GaSP model is updated or 're-characterized' by adding new observations to the data set through a sobol sequence design. We use an iterative process of validation and re-characterization, adding an additional 1k observations each iteration, until the GaSP model sufficiently predicts new observations.

4. **Locate Sweet Spot** : As previously noted, response surface experiments generally begin around a starting point within a reasonable neighborhood of an empirical optimum. We call this neighborhood a 'sweet spot'. The sweet spot(s) provide a smaller experimental region closer to the optimum that better facilitates RSM practices. From the data provided in Steps 1 through 3, a sweet spot is built around the 'best' observation(s) in terms of the desired response, i.e. the sample minimum, maximum, or target.

5. **Emulation** : The Gaussian stochastic process emulator is used to explore the sweet spot prior to Steps 6 and 7. We consider these emulator runs as free, since they are instantaneous. A large space filling design is built around the sweet spot(s) found in Step 4 and predictions of these new observations are made via the GaSP emulator.

6. **Probability Plot** : Normal probability plots is a graphical technique used to asses whether a data set is approximately normal. The data are plotted against theoretical normal distribution in such a way that if the data are normal, the graph will depict a straight line. By looking at probability plots, usually normal, on the response of the top 5 % of the emulator from Step 5 we get a better understanding of the range of values each factor has in our sweet spots. We use these plots to develop levels for factors of the initial fractional factorial design used in the next step. To derive the levels for each factor we take the 25th and 75th percentiles of these probability plots and set them as the low and high levels of our initial fractional factorial design.

7. **Fractional Factorial** : Our initial design is a Resolution III fractional factorial design with levels derived in Step 6. The resolution of a design refers to the degree at which the effects are aliased; a Resolution III design is one where the main effects are confounded with two factor interactions. Recall that the intention of an initial experiment is to screen for active factors, thus, eliminating the need to estimate higher order terms.

There are several possible follow-up experiments. The choice of follow-up experiment depends upon what is learned from the first experiment and expert knowledge of the subject area. The goal of this step is to provide a design with minimal size that will allow the response to head in the direction of the optimum, which is achieved by fitting a first order model from Equation 2.20 and applying the method of steepest ascent seen in the next step.

8. **Steepest Ascent** : We use the method of steepest ascent to head in the direction of the optimum response. In the absence of true background noise, this step provides us with a means for model transition. Using the relationship in Equation 2.21, we can find the step size of steepest ascent in all of the active variables and move towards a neighborhood with better operating conditions than with what we started.

   Once steepest ascent fails, i.e. we are no longer heading in the direction of an improved response, we construct another fractional factorial. Due to the complexity of these computer models, the new fractional factorial includes all of the original factors, since historically 'active' factors tend to fluctuate depending on the region. This makes the method of steepest ascent an iterative process that continues until we reach a plateau in the response, i.e. a region where there is no improvement in operating conditions. Once we reach this plateau we augment the design to a second order optimization experiment.

9. **Optimization Experiment** : Our optimization experiment is built by augmenting the fractional factorial design to a second order design on the active factors from previous region. An active factor is one that is misspecified in the initial design but very important to the response of interest. This design is usually a central composite design (CCD) or a small composite design to limit the number of experimental runs.

As we move into a region with curvature present, interactions and pure quadratics begin to dominate and the first order approximation seen in previous steps becomes less effective.

The purpose of the CCD design is that it allows us to fit a second order model in Equation 2.22, where the factorial runs work at estimating the first order terms, the center runs help estimate the quadratic terms, and the axial runs estimate both first order and the quadratic terms. Equation 2.22 will typically prove useful in a region near the optimum response, since it allows us to model such curvature and to find analytical and graphical solutions for the optimum conditions in Step 11.

10. **Stationary Point** : Once we have performed our optimization experiment, curvature has been detected and we are in a neighborhood of an improved response. The purpose of this step is to determine the location and nature (minimum, maximum, saddle point) of the local optimum. We determine the stationary point of our local optimum analytically via canonical analysis, ridge analysis, or a desirability function in Equations 2.25, 2.26 and 2.27 respectfully.

11. **Confirmatory Experiment/Runs** : A confirmatory experiment is then performed on the optimum estimate found in Step 10. This experiment depends on the previous knowledge of the current experimental region and the subject area expertise.

Historical methods tend to resort to a 'pick the winner' strategy for optimization where we iteratively evaluate the function at a point that optimizes a criterion based on the model obtained using previous evaluation results. After a stopping criterion is met, we then pick the optimum response found in the iteration. Our RSM approach searches for optimum conditions strategically via the sequential learning strategy. Table 1.1 shows the importance

of experimenting in the appropriate environment for RSM in computer experiments. The response surface optimization algorithm allows us to begin in a smaller subset of the entire region of operability through the advent of the 'sweet spot', which produces an environment where the sequential learning strategy of RSM can work.

3.3 Examples

To demonstrate the Response Surface Optimization (RSO) algorithm, we have applied it to two mathematical functions seen in the literature: the Branin function and the Hatrman 6 function. The majority of the mathematical functions used in the literature are fairly rough or 'bumpy' with lower dimensions, including the two used here. We believe, however, that real computer simulations generally behave like the physical systems they are approximating: smooth, nonlinear, and multimodal. Therefore, we illustrate our algorithm with a third example that demonstrates these desired features. We compare the total number of experimental runs (computer evaluations) and the optimum response estimates with the EGO algorithm from Jones *et al.* (1998), which is run until the expected improvement criterion is less than 1% of the current best function value, a stopping criterion suggested by the authors. All space-filling designs of the competing method use the general rule $n = 10k$ where k is the number of factors.

Standard parametric tests, such as t tests, may be poor choices for data exhibiting unique features. The data expressed by the following examples are highly skewed and certainly not normal. Therefore, we exercise distribution free or nonparametric methods to analyze the data. Common nonparametric tests for testing equality of population medians among groups are the Mann-Whitney-Wilcoxon test or the Kruskal-Wallis test. These tests, however, require two assumptions: the groups have the same spreads; and the data distributions have

the same shape. These are clearly violated when examining the data; see Figures 3.2, 3.3, and 3.4. Therefore, we explore another nonparametric approach called bootstrapping.

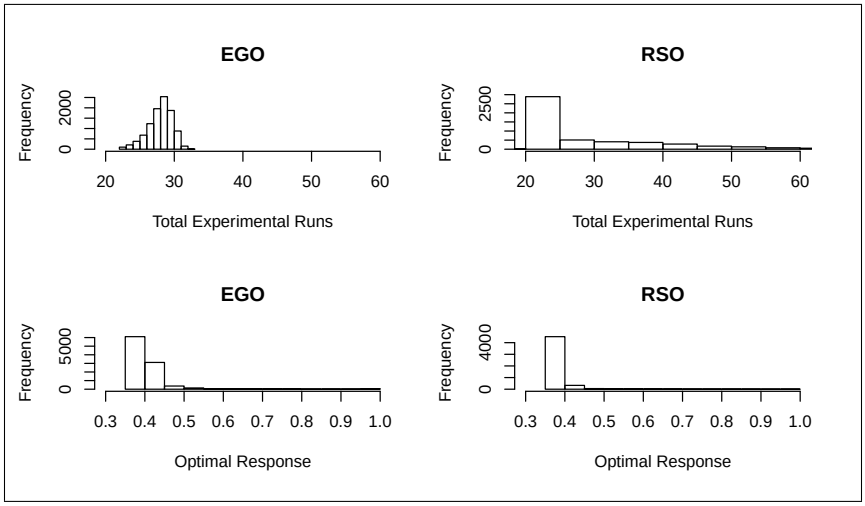

Figure 3.2: Branin Function Data

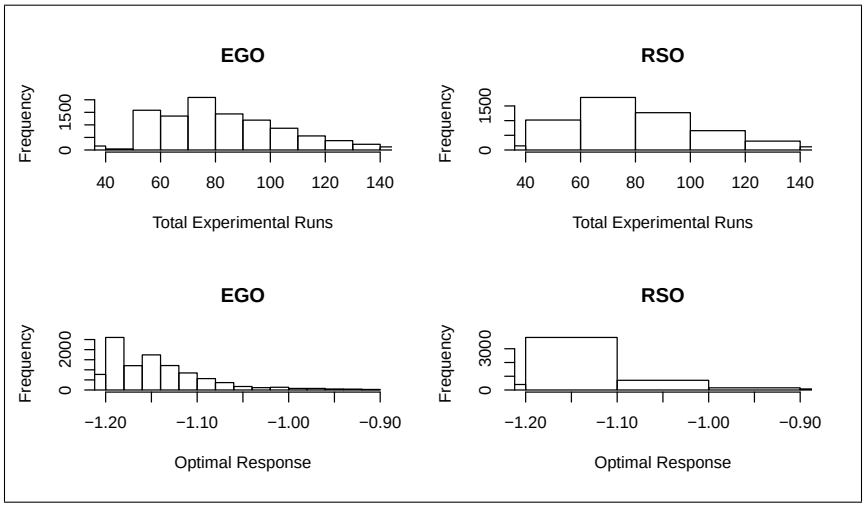

Figure 3.3: Hartman 6 Function Data

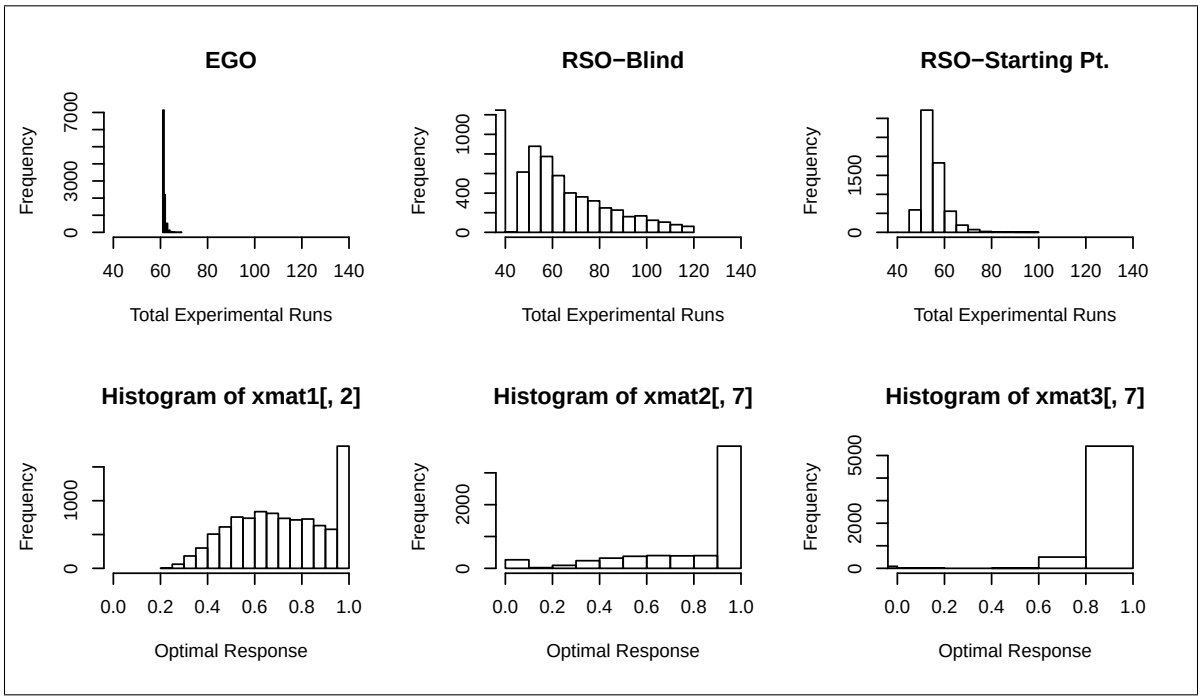

Figure 3.4: Distillation Column Data

Bootstrapping is a statistical inference approach that estimates the sampling distribution of a complex estimator by re-sampling with replacement from the original sample. We use a percentile bootstrapping strategy to construct confidence interval of the difference of the EGO and RSO algorithms. A 25 % trimmed mean for each method was used as the measure for the center of their distributions (a more robust measure to the presence of outliers and skewed data).

3.3.1 Branin Function

The Branin function is

$$f(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}{x_1}^2 + \frac{5}{pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10, \tag{3.2}$$

where the x ranges are $-5 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 15$. The function has three global minima located at $x^* = (-\pi, 12.275); (\pi, 2.275); (9.42478, 2.475)$, where the true global minimum is $f(x^*) = 0.397887$. Optimum estimates and experimental runs were calculated for ten thousand simulations of the RSO and EGO algorithms. Table 4.3 shows the point estimates and a 95% bootstrap confidence interval for the difference between RSO and EGO (margin of error is presented in parentheses).

Table 3.1: Branin function results

Method	Optimum Estimate	Experimental Runs
EGO Algorithm	0.3996	28.65
RSO Algorithm	0.3978	25.83
Difference	0.0017(0.0001)	2.81(0.22)

True Optimum: 0.397887

Both methods prove effective in estimating the true global optimum, with percent errors of less than one percent. RSO provides a better optimum estimate with two less experimental runs on average. The Branin function, however, has low dimensionality which generally is not the case in practice. We use the Hartman 6 function, our next example, to look at our method when higher dimensionality is present.

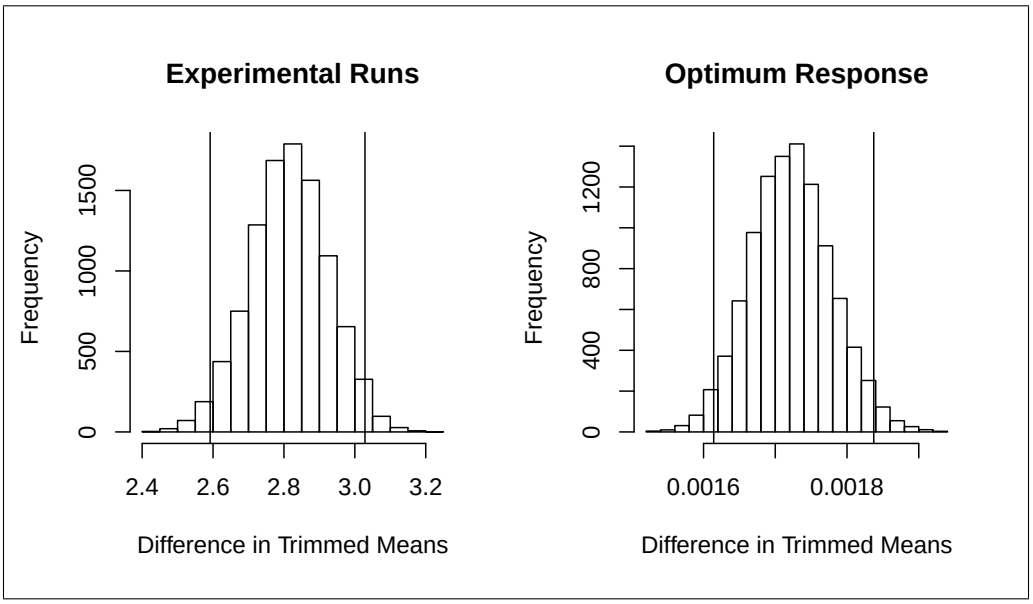

Figure 3.5: Difference between EGO and RSO: Branin Function

3.3.2 Hartman 6 Function

The Hartman 6 function is

$$f(x_1, \ldots, x_6) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} \alpha_{ij}(x_j - p_{ij})^2\right), 0 \leq x_j \leq 1 \tag{3.3}$$

where c_i, α_{ij}, and p_{ij} are coefficients listed below:

$$c_i = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}, \alpha_{ij} = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ .05 & 10 & 17 & .1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & .05 & 10 & .1 & 14 \end{pmatrix},$$

$$p_{ij} = \begin{pmatrix} .1312 & .1696 & .5569 & .0124 & .8283 & .5886 \\ .2329 & .4135 & .9307 & .3736 & .1004 & .9991 \\ .2348 & .1451 & .3522 & .2883 & .3047 & .6650 \\ .4047 & .8828 & .8732 & .5743 & .1091 & .0381 \end{pmatrix}$$

The function has six local minima and one global minima located at $x^* = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573)$, where the true global minima is $f(x^*) = -1.20068$. It should be noted that a $-\log(-y)$ transformation was used, as suggested in Jones, Schonlau, Welch (1998). Optimum estimates and experimental runs were calculated for ten thousand simulations of the RSO and EGO algorithms. Table 4.4 shows the point estimates and a 95% bootstrap confidence interval for the difference between RSO and EGO (margin of error is presented in parentheses).

Table 3.2: Hartman 6 function results

Method	Optimum Estimate	Experimental Runs
EGO Algorithm	-1.1768	105.28
RSO Algorithm	-1.1507	78.70
Difference	$-0.0260(0.0027)$	26.57(1.22)

True Optimum: -1.20068

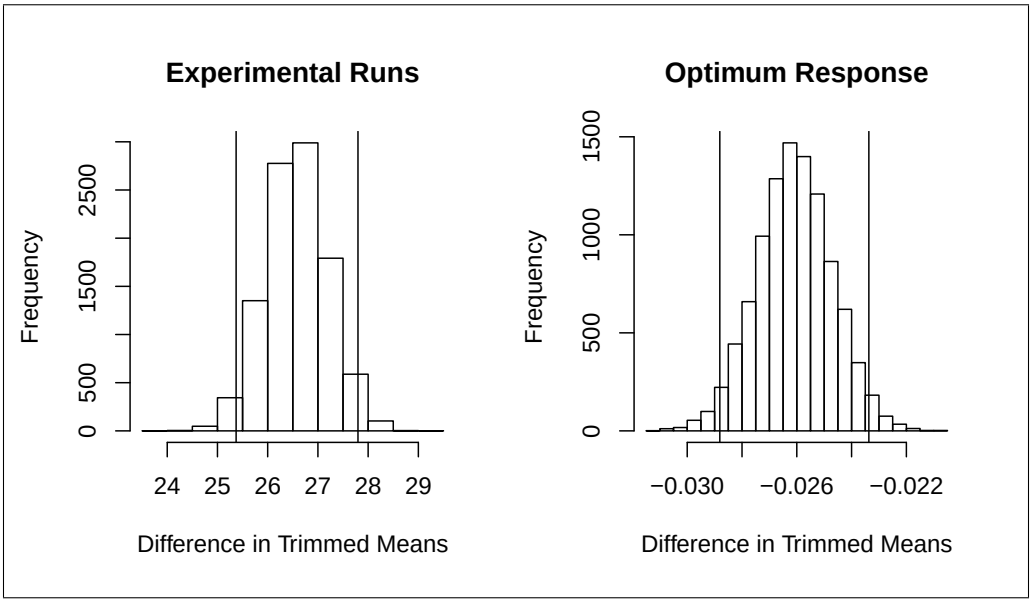

Figure 3.6: Difference between EGO and RSO: Hartman6 Function

Clearly, the EGO algorithm yields a better estimate of the optimum response, but at the expense of roughly twenty-six experimental runs. We see here that there is a tradeoff between

the number of experimental runs and the estimate of the optimum response. Unfortunately, this closed form mathematical function provides us with a tradeoff with no direct interpretation. Hypothetically, if it were a real computer experiment, we could measure the tradeoff by determining if the twenty-six extra runs are worth the additional accuracy on the optimal response.

3.3.3 Etanol-Water Distillation Column

The previous two examples are convenient, closed-form mathematical formulas that are used in place of a real computer experiment. As previously stated, these examples display traits that we believe poorly represent real computer simulations and is our motivation for including a third example illustrating a real computer experiment.

The "distillation column" is a 2000 line Fortran program designed to mimic the actual performance of an ethanol-water column. One of the authors routinely uses this column to teach experimental design techniques. Vining and Kowalski (2006, page 567) suggest its use for engineering statistics courses (information on using this column as a teaching tool can be found at http://www.stat.vt.edu/ vining). The pedagogical version of the column makes the ethanol concentration in the feed stream random. This paper takes the random error off the inlet feed concentration. As a result, the column is a deterministic computer model of an ethanol-water separation.

The program implements the methodology outlined in Perry and Chilton (1973, pages 13-17 - 13-26, 13-32 - 13-35, and 18-1 - 18-19). This methodology represents the state of the art in the mid 1970s for designing a distillation column. The basic idea is to build a computer model for the column using actual physical property and efficiency data.

This specific column consists of 40 bubble-cap trays or plates, a total condenser, and a partial reboiler. The column assumes a saturated liquid feed at tray 18. Saturated liquid feed means that the feed solution is in a liquid state ready to boil. The computer model must solve simultaneously all of the heat and material balances for each plate, the condenser, and the reboiler. The program uses actual efficiency data for an ethanol-water system. The program does take one liberty, however. Ethanol and water are a non-ideal system, forming an azeotrope at 95% ethanol. As a result, in the "real world," the highest possible concentration that the column can achieve is 95% ethanol. This program treats ethanol and water as an ideal system, which greatly simplifies the code. As a result, this code does allow the column to produce concentrations of ethanol in the final product higher than 95%.

The response of interest is concentration of ethanol, expressed as a mole fraction, in the distillate stream. Traditional computer experimentation suggests that we know nothing about the process or that we are blind as to where to start in the region of operability. Table 3.3 shows the point estimates and a 95% bootstrap confidence interval for the difference between RSO and EGO.

Table 3.3: Distillation Column Results

Method	Yield	Experimental Runs
EGO Algorithm	0.7176	61.07
RSO Algorithm	0.9464	60.86
Difference	−0.2287(0.0084)	0.22(0.79)

Table 3.3 provides favorable results for the RSO algorithm, yielding significantly better optimum estimates in roughly the same number of experimental runs (confidence interval includes zero). Although the blind approach shows a significant improvement, we can still take advantage of process knowledge by starting in a smaller experimental region that is

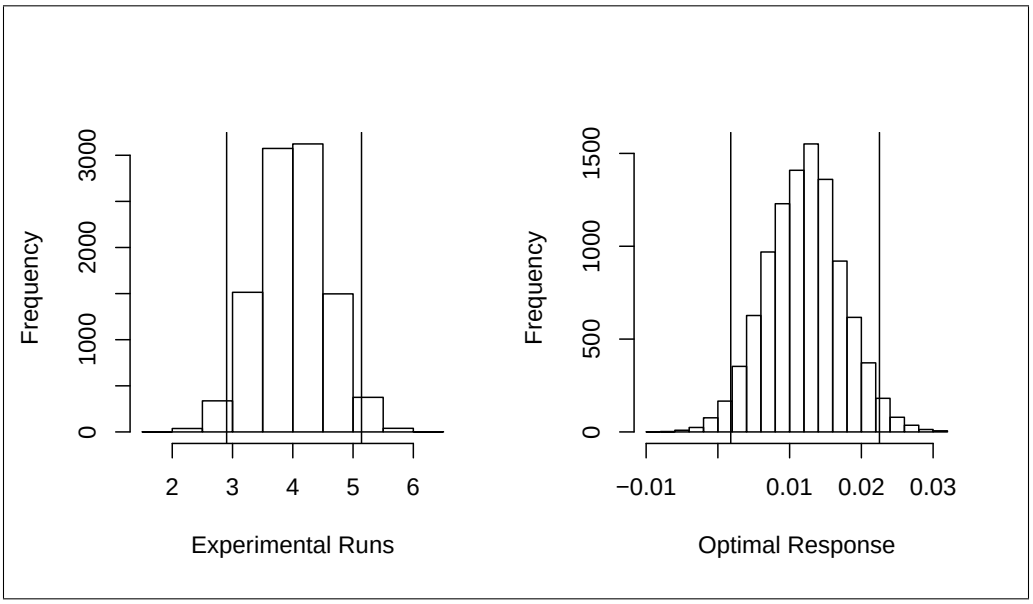

Figure 3.7: Difference between EGO and RSO: Distillation Column

known to have good properties. We compare this method with the blind approach in Table 3.4.

Table 3.4: Effect of Prior Knowledge

Method	Yield	Experimental Runs
RSO Blind	0.9464	60.86
RSO w/ starting pt.	0.9905	54.82
Difference	−0.0443(0.0067)	6.05(0.59)

Table 3.4 shows a significant improvement in performance when we use knowledge of a smaller experimental region as our starting position.

3.4 A Comparison of Characterization Techniques

Section 3.3 illustrates the effectiveness of using RSM inside the computer experiment system. We see that, for smooth functions, the RSO algorithm outperforms the benchmark for

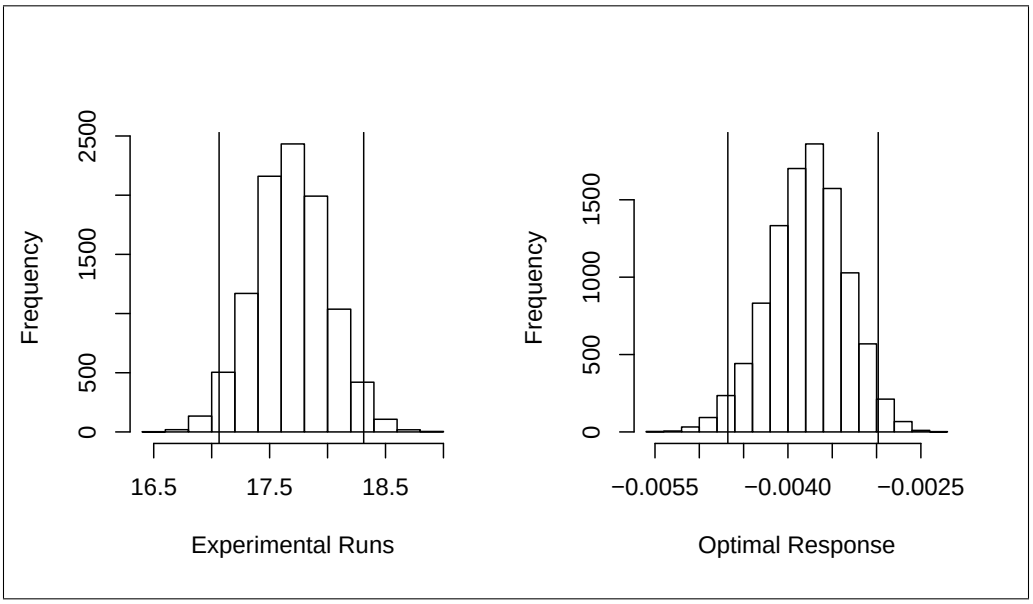

Figure 3.8: Effect of Prior Knowledge on RSO Algorithm

success in optimization of black-box functions. When confronted with irregular functions, however, the Kriging and RSM methods implemented in the RSO algorithm may lead to poor approximations, influencing its accuracy and efficiency. We feel that this situation is the exception rather than the rule and incorporating prior knowledge in the presence of such functions is crucial for performance. However, extending the RSO algorithm to become more flexible is still desired. We begin by introducing two new characterization techniques to the RSO algorithm.

Kennedy and O'Hagan (2001) discuss a Bayesian characterization technique that incorporates more variation into modeling than the Kriging method. Although Bayesian research for global optimization of computer simulations is lacking, it's believed that its predictive capabilities will help better approximate irregular functions inside the RSO algorithm. Traditional Bayesian analysis is very difficult to do at times, especially analysis based on fully specified distributions. Bayes Linear analysis, introduced by Goldstein (1999), uses partially

specified models to circumvent this difficulty. All three methods, Kriging, Bayesian and Bayes Linear, treat the analysis of computer experiments through interpolation via some prediction method. In this section, we examine the effect of three types of characterization techniques (Kriging, Bayesian, Bayes Linear) on the performance of the response surface optimization algorithm introduced in Section 3.4.1.

3.4.1 Alternative Characterization Techniques

A major limitation of the kriging method is that it ignores effects of uncertainty in the estimated parameters. The two Bayesian methods addressed as alternatives in this section, however, observes the uncertainty of estimation and can also account for model inadequacy, observation error, and prior knowledge about parameters.

In Bayesian statistics, the posterior distribution contains all relevant information on the unknown parameters given the observed data. We make statistical inferences typically by evaluating integrals of the posterior distribution, e.g. the posterior predictive distribution in Equation 2.14. The problem is that these integrals are usually very hard to evaluate analytically and numerical programs may fail when presented with high dimensional data sets, which are commonplace in computer experimentation.

The Markov chain Monte Carlo (MCMC) simulation technique solves this by essentially simulating direct draws from the complex distribution of interest.. MCMC methods use the previous sample values to randomly generate the next sample value, generating a Markov chain, i.e. the probability distribution for the system at the next step only depends on the current state of the system. The general problem lies with evaluating

$$E_\pi(h(Y)) = \int h(y)\pi(\theta|y)dx \tag{3.4}$$

where $\pi(\theta|y)$ is the posterior distribution and $h(\cdot)$ is a function like those seen in Equations 2.15 and 2.16. However, we can approximate Equation 3.4 by using simulations $Y_1, Y_2, \ldots, Y_M$ of a Markov chain to calculate the sample average

$$E_\pi(h(Y)) \approx \bar{h}_M = \frac{1}{M}\sum_{t=1}^{M} h(Y_t) \tag{3.5}$$

One problem with applying Monte Carlo integration is in obtaining samples from the complex probability distribution $\pi(\theta|y)$. Attempts to solve this problem via algorithms are the roots of MCMC methods and are seen in Metropolis *et al.* (1953) and Hastings (1970). We adopt the Metropolis-Hastings algorithm described in Kennedy and O'Hagan (2001) to asses the posterior predictive distribution of Y.

The Bayes Linear method is a convenient alternative to the full Bayes for situations where Equation 2.14 is difficult to evaluate. We derive the adjusted expectation of y^* given y and the joint expectation of (y^*, y). This leads to Equations 2.17 and 2.18. Often, we are unable to provide observations of these physical systems due to cost, time, or ethical constraints. Instead we use observed simulation runs to update the expectation and variance in Equations 2.17 and 2.18. In such situations, like those seen in the examples to follow, Equations 2.17 and 2.18 are equivalent to the Kriging technique in Section 3.2.

3.4.2 Comparison of Alternative Characterization Techniques

Let us return to the three examples introduced in Section 3.3: the Branin function, the Hatrman 6 function, and a real computer experiment that simulates a water-ethanol distillation

column. In the following examples, we use a percentile bootstrapping strategy to construct confidence intervals of the difference between the EGO and RSO algorithms. Like Section 3.3, a 25 % trimmed mean for each method was used as the measure for the center of their distributions.

The results for each example are comprised in tables 3.5, 3.6, 3.7, and 3.8. They show 95% bootstrap confidence intervals for each method (margin of error is presented in parentheses).

Table 3.5: Branin Function Results: Comparing Characterization

Method	Optimum Estimate	Experimental Runs
EGO Algorithm	0.39962(0.00011)	28.65(0.05)
Kriging / Bayes Linear	0.39789(0.000001)	25.84(0.22)
Kennedy & O'Hagan	0.39887(0.000056)	34.09(0.18)

True Optimum: 0.397887

We see results similar to Section 3.3 when comparing the RSO algorithm to a benchmark in global optimization. When applying the Bayesian method to the RSO algorithm, however, a decrease in performance is seen for the number of experimental runs needed to investigate the process. As a matter of fact, it is producing roughly six and nine runs more then the leading benchmark and the RSO algorithm with Kriging characterization respectively. The Branin function, however, has low dimensionality which generally is not the case in practice. We use the Hartman 6 function, our next example, to look at our method when higher dimensionality is present.

Again, similar to the findings in Section 3.3, there is a tradeoff between the number of experimental runs and the estimate of the optimum response in the presence of an irregular response surface. Clearly, the EGO algorithm provides the best estimate of the global optimum, but at the expense of many experimental runs. Likewise, the RSO algorithm with

Table 3.6: Hartman 6 Function Results: Comparing Characterization

Method	Optimum Estimate	Experimental Runs
EGO Algorithm	−1.1767(0.0021)	105.29(0.99)
Kriging / Bayes Linear	−1.1506(0.0017)	78.69(0.71)
Kennedy & O'Hagan	−1.1421(0.0015)	61.49(0.38)

True Optimum: -1.20068

the Bayesian method provides the least number of experimental runs, but at the cost of a less accurate estimate of the optimum response. Unfortunately, this closed form mathematical function provides us with a tradeoff with no direct interpretation, so we shift our focus to the real computer experiment described in Section 3.3.

Table 3.7: Distillation Column Results: Comparing Characterization

Method	Yield	Experimental Runs
EGO Algorithm	0.7175(0.0054)	61.07(0.02)
Kriging / Bayes Linear	0.9462(0.0067)	60.87(0.57)
Kennedy & O'Hagan	0.8528(0.0059)	57.92(0.16)

Even though the RSO algorithm (applying either characterization method) beats the benchmark pretty handily, the RSO algorithm still sees the same tradeoff between optimum estimate accuracy and the number of experimental runs. We can, however, still take advantage of process knowledge by starting in a smaller experimental region that is known to have good properties. We prove the results of applying process knowledge at the beginning of testing in Table 3.8.

Table 3.8: Effect of Prior Knowledge: Comparing Characterization

Method	Yield	Experimental Runs
EGO Algorithm	0.7175(0.0054)	61.07(0.02)
Kriging / Bayes Linear	0.9904(0.0014)	54.82(0.11)
Kennedy & O'Hagan	0.8289(0.0064)	57.37(0.16)

Comparing the EGO and RSO algorithms, table 4.6 shows a significant improvement in performance when we use knowledge of a smaller experimental region as our starting position. This conclusion remains consistent throughout the last two chapters. Making adjustments to the characterization method inside the RSO algorithm, however, yields some improvement with the Kriging method but shows less improvement for the Bayesian prediction technique. Like the results from Table 3.7, we also see that the Kriging method applied to the RSO algorithm is producing far better results than when the Bayesian method is applied. One possible reason for the large discrepancy in performance between the characterization methods is the difference between each methods suggested covariance structure in the literature.

Equation 2.3 shows the generic form of a GaSP covariance function used in our methods. The difference in the literature targets how the parameters of this function are estimated. The kriging method seen in Jones *et al.* (1998) estimate the smoothing parameters ($p_j's$) and the roughness parameters ($\theta_j's$), whereas the Bayesian literature tends to estimate only the roughness parameters and fix the smoothing parameter equal to two. Thus, the previous results may not be a fair comparison. The results in Table 3.9 show performance of each example when using the generic covariance structure in Equation 2.3 and estimating all parameters.

Now that the characterization techniques are on common ground, we can compare their impacts on the performance of the RSO algorithm. We see that the two methods achieve similar performance when estimating the optimum and the number of experimental runs. Our recommendation is to use the Kriging method, since it has dominated the field of global optimization for computer experiments. Note that this suggestion is in the presence of no field data (actual data from the physical process). The presence of field data may change

Table 3.9: Comparing Characterization

	Optimum Response	
Function	Kriging	Bayesian
Branin Function	0.39789(0.000001)	0.397898(0.0000019)
Hartman 6 Function	-1.1506(0.0017)	-1.15068(0.0017)
Distillation Column (Blind)	0.9462(0.0067)	0.9463(0.0066)
Distillation Column (Knowledge)	0.9904(0.0014)	0.9903(0.0015)
	Experimental Runs	
Function	Kriging	Bayesian
Branin Function	25.84(0.22)	26.56(0.26)
Hartman 6 Function	78.69(0.71)	78.72(0.71)
Distillation Column (Blind)	60.87(0.57)	60.87(0.57)
Distillation Column (Knowledge)	54.82(0.11)	54.81(0.11)

the recommendation in favor of one of the two Bayesian methods discussed in this section.

3.5 Conclusions

In this chapter, we propose the RSO algorithm as an extension of the EGO algorithm of Jones, Schonlau, Welch (1998) for optimization of deterministic black-box functions. Instead of a 'pick the winner' strategy via an iterative search, our strategy actively searches for an optimum estimate via a hybrid response surface method. To use this method, a Kriging prediction of a Gaussian random model of the function evaluations is used to indicate an experimental region to perform RSM's sequential learning strategy. This method allows us to start in an area where the Taylor series approximations of response surface are appropriate, an issue we see with past applications of RSM inside the computer experiment environment.

We compare RSO with EGO algorithms using two test functions seen in literature and a real computer experiment that simulates an ethanol-water distillation column. The analysis of simulations run on each test function shows that RSO compared favorably to EGO in terms of optimal response and experimental runs. We see a stronger performance of the RSO

algorithm for the Branin function, a smoother function with lower dimensions. However, in the presence of a highly irregular surface (Hartman 6), the Kriging method used in this paper may lead to poor approximations, influencing the accuracy and efficiency of the RSO method.

This is most likely the cause of poorer performance when estimating the optimum of the Hartman 6 function. The analysis of experimental runs of this test function, however, did depict the usefulness of the RSO algorithm when we encounter highly dimensional data. Like physical experiments, we feel that computer experiments are generally smooth, at least in the region of experimentation. Therefore, the documented performance of the RSO algorithm is reassuring, however, ongoing efforts address an adjustment to the RSO algorithm that will help approximate irregular functions.

For the distillation column simulation, we see significant results in favor of the RSO algorithm. The core of traditional response surface methods center on its sequential learning strategy, each experiment uses information from previous experiments, which includes knowledge gained before experiments are conducted. In industry an engineer, physicist, or mathematician generally has an idea how their product behaves and what governs it even before they begin testing the physical system, making a 'blind' approach over the entire region of operability undesirable. This type of thinking has been applied to physical experiments for over fifty years.

The literature suggest the use of many alternative characterization methods, which directs our extension to investigate the affect of different characterization methods in the RSO algorithm. We see no measurable improvement in performance when comparing the alternative characterization methods. The Bayesian method, as a matter of fact, performed similar to the Kriging prediction technique.

It should be noted that the results display a special case of the Bayes linear approach, where it is theoretically equivalent to the Kriging method. For situations where direct field data exist, the Bayes linear characterization method accounts for more uncertainty than the Kriging method and thus, may benefit from the added uncertainty. Otherwise, using the Kriging prediction technique to characterize inside the response optimization algorithm is the preferred method.

The distillation column example shows the benefit of applying process knowledge at the onset of computer experiments. In this example, we see that the RSO algorithm provides better optimum estimates and a smaller number of experimental runs on the average. These illustrations show that, in contrast to popular belief, response surface methodology is an effective application for the optimization of black-box functions.

Chapter 4

A Comparison of Factor Selection Procedures for Unreplicated Fractional Factorial Designs

4.1 Introduction

In Chapter 2, we discuss the importance characterization has played on the computer experiment field in the past decade. The benefits of building an emulator as a cheap approximation of the computer code is well documented, see Sacks *et al.* (1989); Goldstein (2003) ; Jones *et al.* (1998); Kennedy and O'Hagan (2001); Williams, Santner and Notz (2000). Chapter 3 offers an alternative method when shifting the focus from characterization to optimization. By using a Kriging characterization method, the RSO algorithm moves the focus of experimentation from the entire region of operability to a smaller subset of that region (sweet spot). One concern of the RSO algorithm is its use of probability plots to analyze unreplicated fractional factorial designs. This technique is structured so that data are plotted against a theoretical normal distribution in such a way that if the data are normal, the graph will depict a straight line. Significant or active effects generally appear as extreme points away from the straight line, whereas the inert effects tend to fall roughly along it. Although this

graphical analysis is very powerful and effective, its nature is subjective and reduces the framework for automation of the algorithm.

Examples in Figure 4.1, taken from Box and Meyer (1986), display this subjectivity. Example 1 shows the division between 'active' and 'inert' effects clearly as a straight line through the origin shows factors 14 and 15 fall off of this line. Thus, Factors 14 and 15 are the active factors in Example 1. The conclusions for Example 2, on the other hand, are more subjective and are sensitive to the assumptions of an individual. Box and Meyer (1986) present a more formal approach for screening active factors in an unreplicated fractional factorial design and suggest using this analysis as a supplement to normal probability plots. We investigate the impact of using this method and compare it to the results from Chapter 3.

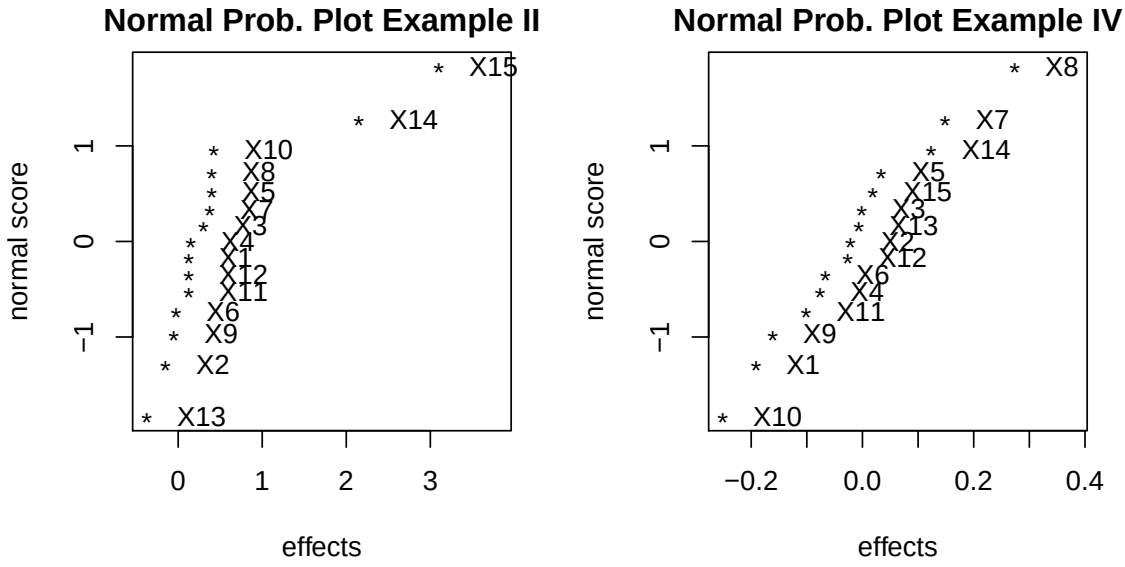

Figure 4.1: Left: Example 1 Right:Example 2

Box and Meyer (1986) present a formal approach for screening active factors in an unreplicated fractional factorial design and suggest using this analysis as a supplement to normal

probability plots. Box and Meyer's method is particularly effective for discovering the active factors that have a major influence on the response through their main effects and/or interactions. Figure 4.2 shows results from applying the Box and Meyer method to an experiment run on six variables. The last variable looks like the only variable that impacts the response significantly, which can be see by the length of the bar in the posterior probability plot.

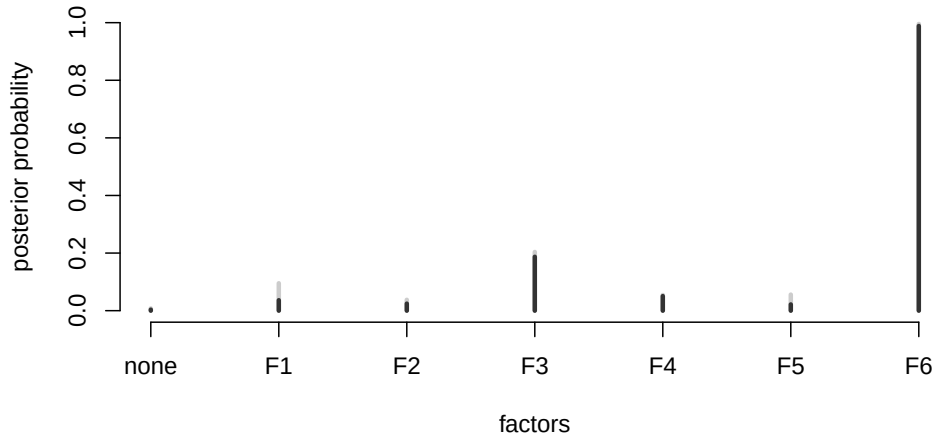

Figure 4.2: Box and Meyer Posterior Probability Plot

The Box and Meyer method presents theoretical models that uililize the factor sparsity principle explicitly. The normal probability plotting method used in the current formulation of the RSO algorithm does this only implicitly. Thus, the Box and Meyer method provides a proven factor screening procedure that has additional rigor over the graphical methods commonly used in practice.

The remainder of this chapter is organized as follows. In Section 4.2, we provide an overview of the RSO algorithm. Section 4.3 introduces the Box and Meyer method. Section 4.4 reintroduces the two closed form mathematical function from Section 3.3. These examples compare

the optimum estimates when applying the Box and Meyers method to the probability plot method used in RSO. We also investigate the robustness of choosing prior probabilities inside the Box and Meyer method. We also reintroduce the real computer experiment that simulates an ethanol-water distillation column based on first principles of chemical engineering. Section 4.5 contains our conclusions and discussion on the topic.

4.2 The Response Surface Optimization Algorithm

Below is an overview of the response surface optimization algorithm from Chapter 3. This overview compartmentalizes the eleven steps of RSO into three main components: Gaussian stochastic process modeling, Kriging prediction emulation, and response surface methodology.

1. **Gaussian Stochastic Process Modeling** : For characterization, we model the deterministic response through Equation 2.1. Once the response is modeled, an iterative process of validation and re-characterization techniques help move the experiment from the entire region of operability to a smaller subset, a 'sweet spot'. This allows us to make better use of the sequential learning strategy of response surface methodology.

2. **Kriging Prediction Emulation** : This component uses data, process knowledge, and prediction techniques to fine tune the region of experimentation for RSM testing. By moving from the entire region of operability, we enter a smaller subregion more appropriate for the low-order Taylor series approximations in the last component. Like the previous component, this component uses a Kriging prediction method that calculates the best linear unbiased predictor (BLUP) and its variance, from Equations 2.6 and 2.7 respectively.

3. **Response Surface Methodology** : The last component consists of the sequential learning strategy of RSM that is used to search for an optimum response. This method is based on low-order Taylor series approximation that seeks the 'optimal' processing conditions through the path of steepest ascent. The experimental designs used in this component are unreplicated fractional factorials, since replication on a deterministic model is worthless. The literature suggests the use of many different techniques to identify active effects of unreplicated experiments.

   Contrasts associated with insignificant higher order interactions are often used to estimate error variance. These inert contrasts, however, are sometimes difficult or impossible to identify. Another insufficient method, estimates the experimental error by pooling insignificant factors to generate statistical tests for main effects. The RSO algorithm utilizes a graphical technique provided by Daniel (1959), which is the method mainly used in practice. We introduce an alternative method to plotting in Section 4.3.

These three components are the backbone of adapting response surface methods for computer experiments. In Chapter 3, we see that performance is average at best when encountering irregular functions. This chapters focus is to adjust the RSO algorithm to become more flexible in the face of these functions. One attempt previously, adjusted Components 1 and 2 via alternative characterization techniques (see Chapter 3). This adjustment, however, had no influence on the overall performance of the algorithm. Thus, we focus on the third component's capability to select active factors better within the system.

4.3 Box and Meyer Method

The primary objective of the Box and Meyer method is to determine a subset of factors which are involved in the active main effects and interactions. Based on the principle of factor sparsity, a vector of estimated effects or model is stated. For each model, we develop posterior probabilities to determine which of the k factors in the experiment are active. By replacing normal probability plots with the Box and Meyer method in the RSO algorithm, we introduce a more objective and automated technique.

The Bayesian framework of the Box and Meyer method involves computing a posterior probability that each effect is active, where the prior information is summarized in two parameters: α (the prior probability that one factor is active) and k ($k^2 = \frac{\sigma^2_{overall}+\sigma^2_{effect}}{\sigma^2_{overall}}$). If we let $\mathbf{T} = (T_1, \ldots, T_k)$ be the vector of k estimated effects we can assume, given the prior information above, that $(T_1, \ldots, T_k)$ are iid from (1-α)N(0,σ^2) + αN(0,$k^2\sigma^2$); see Box and Meyer (1986). Under this assumption the conditional posterior probability, given σ, that an effect is active is

$$P_{i|\sigma} = \frac{\alpha\frac{1}{k}\exp\left[\frac{-T_i^2}{2k^2\sigma^2}\right]}{\alpha\frac{1}{k}\exp\left[\frac{-T_i^2}{2k^2\sigma^2}\right] + (1-\alpha)\exp\left[\frac{-T_i^2}{2\sigma^2}\right]}. \tag{4.1}$$

The parameter σ is then integrated out of Equation 4.1 over its posterior distribution $p(\sigma|\mathbf{T})$, via numerical integration, to achieve the probability p_i that an effect T_i is active; that is,

$$p_i = \int_0^\infty P_{i|\sigma} p(\sigma|\mathbf{T})\mathbf{d}\sigma, \tag{4.2}$$

where,

$$p(\sigma|T) \propto \sigma^{-n} \prod_{j=1}^{k} \left[(1-\alpha) \exp\left[\frac{-T_j^2}{2\sigma^2}\right] + \alpha \frac{1}{k} \exp\left[\frac{-T_i^2}{2k^2\sigma^2}\right] \right].$$

One concern with using the Box and Meyer method is the amount of computing time needed to examine the posterior probabilities for the numerous models. This concern greatly increases as the number of possible active factors increases. For example a full exploration of the model space with twenty factors requires us to examine $2^{20} = 1,048,576$ models. In practice, computer experiments generally have a large input space. Under the factor sparsity principle, it seems reasonable to first examine models with a low number of factors. Use of such a systematic approach will significantly reduce the total number of models to examine. While this is a concern in practice, the examples in Section 4.4 have reasonable dimensions and results are received instantaneously.

4.4 Examples

We begin the analysis of the Box and Meyer method by returning to the two mathematical functions from Equations 3.2 and 3.3, the Branin function and the Hartman 6 function respectfully. Box and Meyer (1986) conclude that results are generally robust to the prior assessment about the proportion of factors expected to be found active (α).

We investigate the robustness of the performance of the RSO algorithm with respect to the choice of α. We then compare these results with the results from Chapter 3, where a probability plotting method was used to select active factors of experimental designs. We also return to the real computer experiment that simulates a ethanol-water distillation column. We apply our suggestion for choice of α to this computer experiment and compare it with results from Chapter 3.

4.4.1 Choice of α and k

In practice, a small number of factors are generally found to be active. Thus, small values of α are usually appropriate. Box and Meyer (1986) present a working range for the selection of α and k with averages of 0.20 and 10 respectfully. They show that conclusions drawn from changes in these parameters are generally insensitive to moderate changes. We use two mathematical functions, Branin and Hartman 6, to demonstrate the choice of parameters of the Box and Meyer method. Optimum estimates and experimental runs were calculated for five thousand simulations of the RSO algorithm for five levels of α (0.10, 0.20, 0.40, 0.60, 0.80) and optimized over levels of k. Tables 4.1 and 4.2 show 95% bootstrap confidence intervals for the Box and Meyer method (margin of error is presented in parentheses).

Table 4.1: Branin Function Results: Choice of Alpha

α	Optimum Estimate	Experimental Runs
0.10	0.397907(0.0000042)	31.01(0.21)
0.20	0.397908(0.0000045)	31.02(0.21))
0.40	0.397909(0.0000044)	31.01(0.22))
0.60	0.397999(0.0000230)	28.29(0.13))
0.80	0.397907(0.0000044)	31.21(0.21))

Table 4.2: Hartman 6 Function Results: Choice of Alpha

α	Optimum Estimate	Experimental Runs
0.10	-1.0066(0.0049)	59.72(0.31)
0.20	-1.1702(0.0014)	79.14(0.66))
0.40	-1.1687(0.0015)	79.54(0.68))
0.60	-1.1683(0.0017)	79.70(0.79))
0.80	-1.1681(0.0015)	79.35(0.71))

Consistent with results from Box and Meyer (1986), we see a fairly robust method as we change the choice of α. Figure 4.3 shows a graphical display of the performance affect when

changing α in the Box and Meyer method. A simple smoothing spline is displayed along with the actual data extracted from the tables. We use this spline to get a rough idea of how the choice of α behaves for levels we did not directly examine. The main objective for both test functions is to minimize both the optimum response and the number of experimental runs.

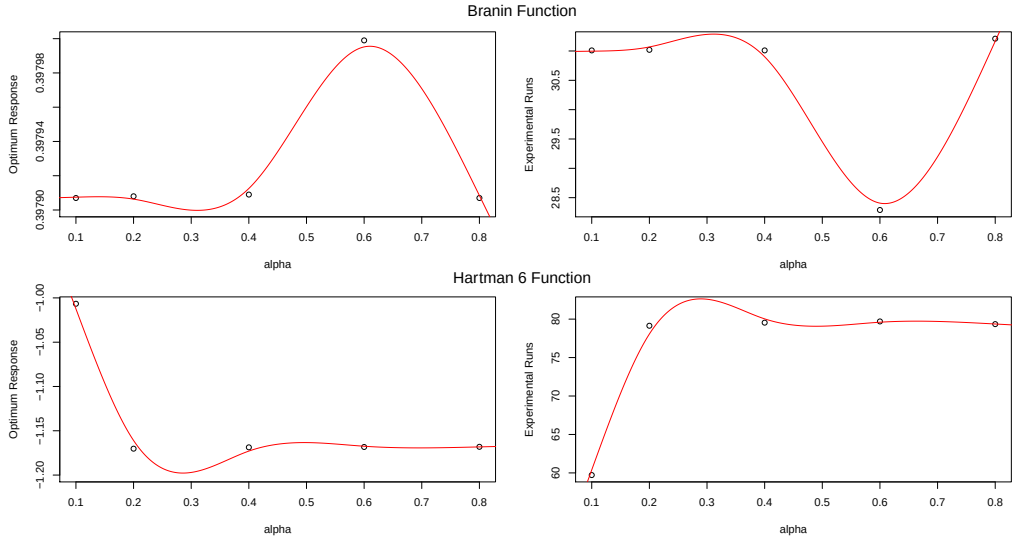

Figure 4.3: Effect of Changing Alpha in Box and Meyer Method

We recommend the choice of α to be inside range between $0.2 \leq \alpha \leq .4$, where we see that the choice is reasonably robust and performance is not sacrificed on either of the two performance measures. This recommendation is consistent with the conclusions by Box and Meyer (1986). We use an α level of 0.20 to compare the Box and Meyer method with the EGO algorithm and Normal probability plotting method in Tables 4.3 and 4.4.

4.4.2 Method Comparison

Now that we have established a rule for the choice of the prior probability about the proportion of factors expected to be found active (α), we examine the differences between the Box and Meyer ($\alpha = 0.20$) discussed in Section 4.3 and the two algorithms examined in Chapter 3. Tables 4.3 and 4.4 summarize the 95% bootstrap confidence intervals for each method (margin of error is presented in parentheses).

Table 4.3: Branin Function Comparison

Method	Optimum Estimate	Experimental Runs
EGO Algorithm	0.39962(0.00011)	28.65(0.05)
RSO w/ Probability Plot	0.39789(0.000001)	25.84(0.22))
RSO w/ Box and Meyer	0.397908(0.0000045)	31.02(0.21))
True Optimum: 0.397887		

Table 4.4: Hartman 6 Function Comparison

Method	Optimum Estimate	Experimental Runs
EGO Algorithm	-1.1767(0.0021)	105.29(0.99)
RSO w/ Probability Plot	-1.1506(0.0017)	78.69(0.71)
RSO w/ Box and Meyer	-1.1702(0.0014)	79.14(0.66))
True Optimum: -1.20068		

Clearly, the probability plot method provides the best results for the Branin function. The Branin function, however, has low dimensionality where the benefit of a more rigorous factor selection procedure is less. As our data dimension increases the benefit of the RSO algorithm increases. The Hartman 6 example shows the added effect of the Box and Meyer method for experiments with higher dimensions. We see that the more objective procedure achieves better operating conditions than when probability plots are applied to the RSO algorithm. The Box and Meyer method's optimum estimate also compares much better to the leading benchmark in global optimization for computer experiments.

The previous two examples are convenient, closed-form mathematical formulas that are used in place of a real computer experiment. As previously stated, these examples display traits that we believe poorly represent real computer simulations and is our motivation for including a third example illustrating a real computer experiment.

We once again look at two different situations for the real computer experiment. The first, assumes we know nothing about the process or that we are blind as to where to start in the region of operability. The second, uses process knowledge to begin in a region of good processing conditions, a technique that is often used in practice to assist in the sequential testing of RSM. We look at the same output as we did in the previous chapter. Table 4.5 shows 95% bootstrap confidence intervals comparing the EGO and RSO algorithms assuming with begin with no information about the process.

Table 4.5: Distillation Column Results

Method	Yield	Experimental Runs
EGO Algorithm	0.7175(0.0054)	61.07(0.02)
RSO w/ Probability Plot	0.9462(0.0067)	60.87(0.57)
RSO w/ Box and Meyer	0.9296(0.0075)	58.55(0.73)

Similar to the results from Chapter 3, Table 4.5 provides favorable results for the RSO algorithm, yielding significantly better optimum estimates in roughly the same number of experimental runs as the EGO algorithm. Applying the RSO algorithm with the Box and Meyer method shows an improvement in the number of experimental runs with roughly the same performance in estimating the optimum response when comparing it to the more common probability plotting method.

The blind approach shows positive results but we see a small tradeoff in performance when using different factor screening procedures, albeit minimal. In practice, however, we gen-

erally use process knowledge at the beginning of experimentation to our advantage. Table 4.6 summarizes 95% bootstrap confidence intervals when we use process knowledge to our advantage in the RSO algorithm.

Table 4.6: Effect of Prior Knowledge

Method	Yield	Experimental Runs
EGO Algorithm	0.7175(0.0054)	61.07(0.02)
RSO w/ Probability Plot	0.9904(0.0014)	54.82(0.11)
RSO w/ Box and Meyer	0.9989(0.00004)	48.55(0.44)

Table 4.6 and Figure 4.4 show a strong performance when using prior knowledge within the RSO algorithm, which is consistent with conclusions made in Chapter 3. Additional improvement is made when we supplant the probability plot method with the Box and Meyer factor screening method.

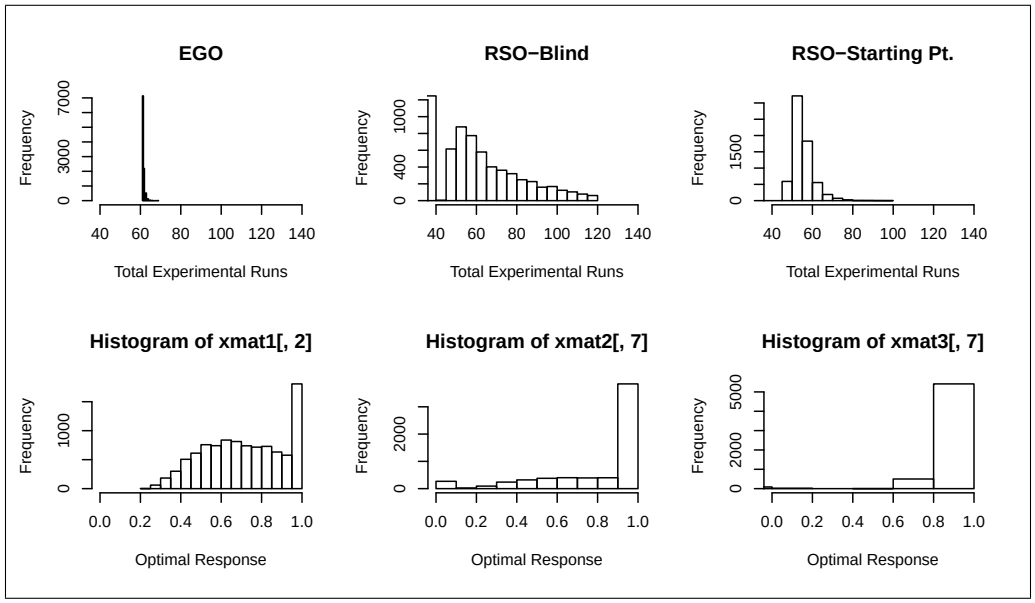

Figure 4.4: Blind RSO versus RSO with Process Knowledge

4.5 Conclusions

The response surface optimization algorithm, introduced in Chapter 3, actively searches for an optimum estimate via a hybrid response surface method. Chapter 3 incorporates a Kriging prediction method to indicate an experimental region that has better operating conditions than the entire region of operability, a better environment for RSM's success.

One concern is RSO's use of subjective plots to select active factors of the RSM experimental designs. The adjustment provided in this chapter examines this concern by presenting a more objective measure for selecting active factors. In addition to comparing the two factor selection procedures, we compare these methods to a leading benchmark in the global optimization literature, the Efficient Global Optimization algorithm by Jones *et al.* (1998).

Simulations on two test functions seen in literature and a real computer experiment that simulates an ethanol-water distillation column show similar results to those seen in Chapter 3 when comparing between the RSO algorithm and the EGO algorithm. Overall performance favors the use of the RSO algorithm over the EGO algorithm. In other words, it favors the sequential learning strategy of RSM over the 'pick the winner' of an iterative global search for the optimum.

In Chapter 3, we conclude that in the presence of smooth functions (Branin function and distillation column), we see a stronger performance of the RSO algorithm in opposition to the EGO algorithm. In the presence of a highly irregular surface (Hartman 6), however, a tradeoff exists between performance of optimum estimates and the number of computer evaluations. By replacing the probability plotting method with the Box and Meyer method for selecting active factors, we can achieve better performance for these irregular functions. We see that the tradeoff in the Hartman 6 function is eliminated through this adjustment of

the RSO algorithm.

Again, we see RSO's improvement over the benchmark. The Box and Meyer adjustment to the RSO algorithm produces good performance for the distillation column computer experiment as well. In a blind search, the performance of the Box and Meyer method is comparable to the probability plotting method. After applying process knowledge, a common practice in industry, the RSO algorithm with the Box and Meyer method is the clear winner in performance, providing a better optimum estimate in fewer experimental runs.

In Chapter 3, we show that the RSO algorithm can be an effective means for searching for the optimum of a response surface. Even though it presents promising results, an adjustment was needed to handle highly irregular functions. Given the results in this chapter, the RSO algorithm with the Box and Meyer application seems to solve this issue as the algorithm's functionality works well for both smooth and irregular response surfaces.

Recall, that the probability plots factor selection procedure interferes with the automation of the RSO algorithm. One advantage of the Box and Meyer method was its properties for allowing a more automated RSO algorithm. With this additional advantage, we not only provide better estimation for 'bumpy' functions but provide computer programmers a more viable option to write an all inclusive computer algorithm.

Chapter 5

Exploring White Noise and Model Transition Performance via a Pseudo Lack of Fit Test

5.1 Introduction

The response surface methods component of the RSO algorithm is very similar to the guideline of traditional methods. Recall that response surface methodology is a sequential strategy based on low-order Taylor series approximation that seeks the 'optimal' processing conditions through three distinct phases. The first two phases of RSM utilize a first order model and a technique called steepest ascent to search for an optimum. The sequential tests usually culminates in second order model to examine the neighborhood of the optimum.

One difference between traditional RSM and the RSO algorithm is how we handle the transition between first order and second order models. In the absence of experimental error, the RSO algorithm uses steepest ascent to transition between first and second order models. As the experimental region moves closer to the optimum, however, curvature becomes more dominant making the method of steepest ascent, a linear approximation method, less effective. For this reason, traditional methods generally apply a lack of fit test to determine

when curvature is present inside a system.

For example, let us say the true relationship between an input, x, and an output, y, is quadratic. This relationship is expressed in Figure 5.1. We use the lack of fit test to tell if a model fits the data; for a linear fit, this tests demonstrates whether or not curvature is present within the system. The example in Figure 5.1 shows some indication that a straight line fit to the data is not satisfactory. Thus, it would be helpful to have a test to determine if curvature is present.

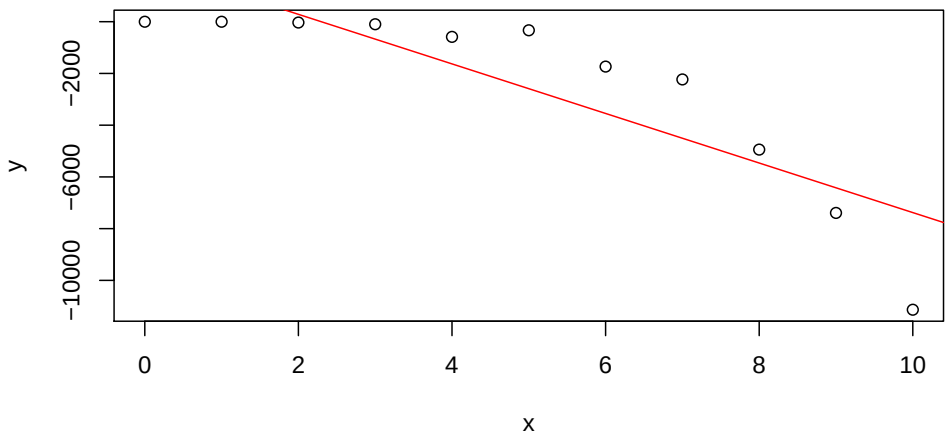

Figure 5.1: Lack of Fit of a linear model

Lack of fit tests require that we have true replicates on the response for at least one set of levels on the regressors, which we are unable to acquire when dealing with deterministic functions. Thus, we propose developing 'pseudo' error by allowing inert factors to roam freely (within the original range) to generate error for the lack of fit type test. The use of a lack of fit test may make the transition to a second order experiment more effective.

However, one concern is that results could be skewed when curvature exists in the inactive factor(s). The hope is that this effect is negligible. The idea is to use this test as an attempt to mimic traditional RSM testing on physical systems.

For physical experiments, once we find an inert factor, we build the subsequent experiments in the active factors only. In these experiments we specify the inert factor to a nominal value and then allow it to roam freely within its natural variation for the remainder of testing. For certain inert factors, the range of normal variation may be quite tight due to active engineering process control. In other cases, such as raw material factors, the range may be rather large.

In physical experimentation, a factor may be considered inert for a first-order model when, in fact, it has significant curvature that masks the first-order effect. A typical assumption in response surface methodology is effect hierarchy, where first-order effects dominate second-order. Once we drop the inert factor from our experiment, its curvature component becomes part of the estimated error, leading to bias. However, if we seek a maximum response and the nominal value is the maximum for the presumed inert factor, then we still are making a good decision about the location of the optimal response. The problem, of course, is when the nominal value for the inert factor is a minimum and we seek a maximum. This problem of curvature in a presumed inert factor is inherent to the practical application of RSM. The RSO algorithm takes the same risk. It is inherent to the approach.

There is a strong possibility that this method will add more experimental runs to the process. One advantage of developing the pseudo error for the lack of fit type test, however, is that we gain some insight on the white noise in the neighborhood of active factors. Therefore, we may be able to use the pseudo error as a post confirmation phase (a 12th step in the algorithm). This post confirmation step would serve as an intermediate step when no field

data exist. It will serve as a stepping stone as we progress from the computer experiment to the implementation of the actual process or product. In this respect, it is helpful to have some idea about the white noise in the region before making the product as we make this transition.

The remainder of this chapter is organized as follows. Section 5.2 presents the current method used for model transition inside the RSO algorithm. In Section 5.3, we introduce the linear model theory of the lack of fit test when pure error is present in a system. This section also presents the development of the "pseudo" error needed to compute the lack of fit type test. Section 5.4 presents two of the three examples seen in the previous chapters to demonstrate the pseudo lack of fit test. It also includes a detailed example of the RSO algorithm from beginning to end. Section 5.5 contains our conclusions and discussion on the topic.

5.2 Model Transition via Steepest Ascent

In the absence of true background noise, we use the method of steepest ascent to transition between models in RSM experiments, i.e. movement between first and second order models. Recall that RSM begins with an initial experiment in the neighborhood of the recommended set of operating conditions. The experimenter must determine the set of factors and their levels for this experiment. At this point, we require a strict first-order (Equation 2.20) for the response to minimize design size.

We use this initial experiment to screen out the inactive factors and to find the direction for better results. We then head in this direction by building a 'path' to find the region or neighborhood of the optimum (path of steepest ascent) via the relationship seen in Equation 2.21. Equation 2.21 shows that increasing the key factor unit by unit produces the path of steepest ascent.

Typically, steepest ascent works best when the initial experiment is far from the region with optimum conditions, because here the first order approximations are most reasonable. As we near the optimum interactions and pure quadratics begin to dominate, the steepest ascent method becomes less effective. When the effectiveness diminishes, the use of a fitted second order response surface model (Equation 2.22) is recommended for finding optimum conditions in the response.

This is where we transition between the first and second order response model. Currently, a transition decision is based on the method of steepest ascent. For example, lets say we go through two iterations of steepest ascent to head in the direction of the optimum. After the second iteration, we enter a region where there is no improvement in operating conditions, i.e a plateau in the response. This is our indicator to transition to a second order response surface model. When curvature exists, however, using this method may be self defeating. Myers, Montgomery, and Anderson-Cook (2009) present Figure 5.2 which shows how the method of steepest ascent is affected by the presence of curvature.

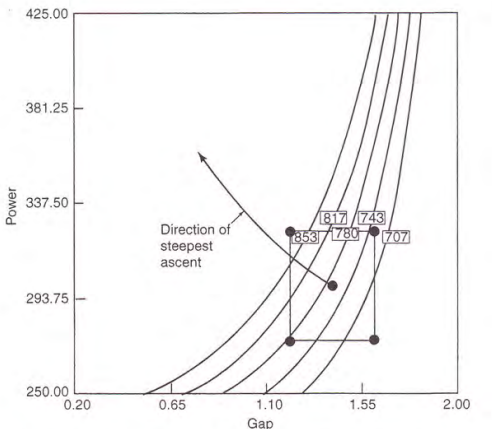

Figure 5.2: Curvatures Effect on Steepest Ascent

Since computer experiments are highly nonlinear, this method may be inappropriate for

model transitions. One alternative to the steepest ascent method is the lack of fit test. These tests can point out the presence of significant curvature and indicate the neighborhood of a local optimum. Therefore, the development of a formal lack-of-fit test may be helpful in the adaptation of response surface methods for computer experiments.

5.3 Lack of Fit

Section 5.2 discusses the importance of the development of a statistical test for model transitions and follow-up experiments, i.e. lack of fit tests. In the computer experiment environment, where we do not have random error, we cannot use the traditional lack-of-fit framework. Thus, we develop a framework for developing 'pseudo error' to construct the pseudo lack of fit test. We begin first by looking at an example of an unreplicated fractional factorial design from a system with three active factors and three inactive factors:

Table 5.1: Unreplicated Fractional Factorial

x_1	x_2	x_3
-1	-1	-1
$+1$	-1	-1
-1	$+1$	-1
$+1$	$+1$	-1
-1	-1	$+1$
$+1$	-1	$+1$
-1	$+1$	$+1$
$+1$	$+1$	$+1$
0	0	0

Table 5.1 displays a fractional factorial design (factor levels are coded between $\pm$ 1) commonly used in the current framework of the RSO algorithm. Clearly, there is no replication in this table, each combination of factors occur only once. To produce error for the lack of fit test, we generate it through the inactive factors of the system by allowing the inactive

factors to roam freely within the original range. In our example, we generate pseudo error for the active factors by adding the experimental runs in Table 5.2 to Table 5.1.

Table 5.2: Generating Pseudo Error

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0.004761952	0.566804482	0.749745762
0	0	0	-0.560650374	-0.713535209	-0.081889970
0	0	0	-0.551911723	-0.503022706	0.019056173
0	0	0	0.586754873	0.977899793	-0.856359321

This allows us to construct a pseudo lack of fit test. Under the usual environment the lack-of-fit test construction involves us partitioning the residual sum of squares into two components,

$$\begin{aligned} SS_{Error} &= \sum_{i=1}^{m}\sum_{j=1}^{n_i}(y_{ij}-\hat{y}_i)^2 \\ &= SS_{PE} + SS_{LOF} \end{aligned} \tag{5.1}$$

where SS_{PE} is the sum of squares attributed to the pure error (i.e. the sum of squares attributed to replication) and SS_{LOF} is the sums of square due to lack of fit. Through simple algebraic manipulation of the residual sum of squares we get the following formulas:

$$SS_{PE} = \sum_{i=1}^{m}\sum_{j=1}^{n_i}(y_{ij}-\bar{y}_i)^2$$

and,

$$SS_{LOF} = \sum_{i=1}^{m} n_i(\bar{y}_i - \hat{y}_i)^2,$$

where m is the number of distinct experimental design runs and n_i are the number of replicates. In our case, each n_i for the unreplicated factorial runs is equal to one and the n_i for the center runs with pseudo error is equal to the number of center runs. The sums of squares are used to develop the test statistic for lack of fit; that is,

$$F = \frac{SS_{LOF}/(m-p)}{SS_{PE}/(n-m)} \tag{5.2}$$

where p is the number of parameters that are estimated for your model, and n is the total number of observations. The F statistic has an F-distribution with the corresponding number of degrees of freedom in the numerator and the denominator under the null hypothesis that the first order model is correct. If the model is wrong, i.e. curvature is present, the F-statistic will be large and we reject the null hypothesis that linear model is correct.

5.4 Examples

Two of the three examples seen in Chapters 3 and 4 are used again to demonstrate the pseudo lack of fit test. The Branin test function has low dimensions (two factors) and in the event that both factors are significant, we are unable to compute the pseudo lack of fit test for this function. Therefore, we have omitted this example from analyses in this section. In Section 5.4.1, we look at how the pseudo lack of fit impacts performance when used as a model transition technique.

There is a strong possibility that this method will add more experimental runs to the process. One advantage of developing the pseudo error for the lack of fit type test, however, is that we gain some insight on the white noise in the neighborhood of active factors. Therefore, we look at this effect by applying the pseudo lack of fit test solely as a post confirmation strategy. In practice, this allows us to test if the model approximations are reasonable and gives us knowledge on the white noise for production. Section 5.4.2 introduces this post confirmation strategy as a twelfth step to the RSO algorithm.

5.4.1 Model Transition

Model transitions inside the RSO algorithm occur when we shift from a first order model to a second order model, which occur in steps seven through nine of the RSO algorithm. It is here were we apply the experimental runs necessary for pseudo error. This provides a path to transition to the second order models used in the optimization experiments in Step 9.

In the two following examples, the Hartman 6 function and the distillation column experiment, we use a percentile bootstrapping strategy to construct confidence intervals of the difference between the steepest ascent and pseudo lack of fit model transition techniques. A 25 % trimmed mean for each method was used as the measure for the center of their distributions. Table 5.3 and Figure 5.3 show model transition results for the Hartman 6 test function.

Table 5.3: Hartman 6 Function Results: Lack of Fit

Method	Optimum Estimate	Experimental Runs
RSO Lack of Fit	-1.0902	86.68
RSO Steepest Ascent	-1.1717	79.69
Difference	0.08140(0.0057)	6.98(1.34)

True Optimum: -1.20068

There is a rather large discrepancy in performance between the two model transition techniques. Applying the pseudo lack of fit test to the RSO algorithm in Steps 7 and 8 provides a poor estimate of the optimum response at the expense of more experimental runs than the steepest ascent method. Recall that the Hartman 6 function is highly irregular, i.e bumpy. Thus, it should not be surprising that a test, structured to point out the presence of curvature inside a system, yields a local optimum where curvature exists.

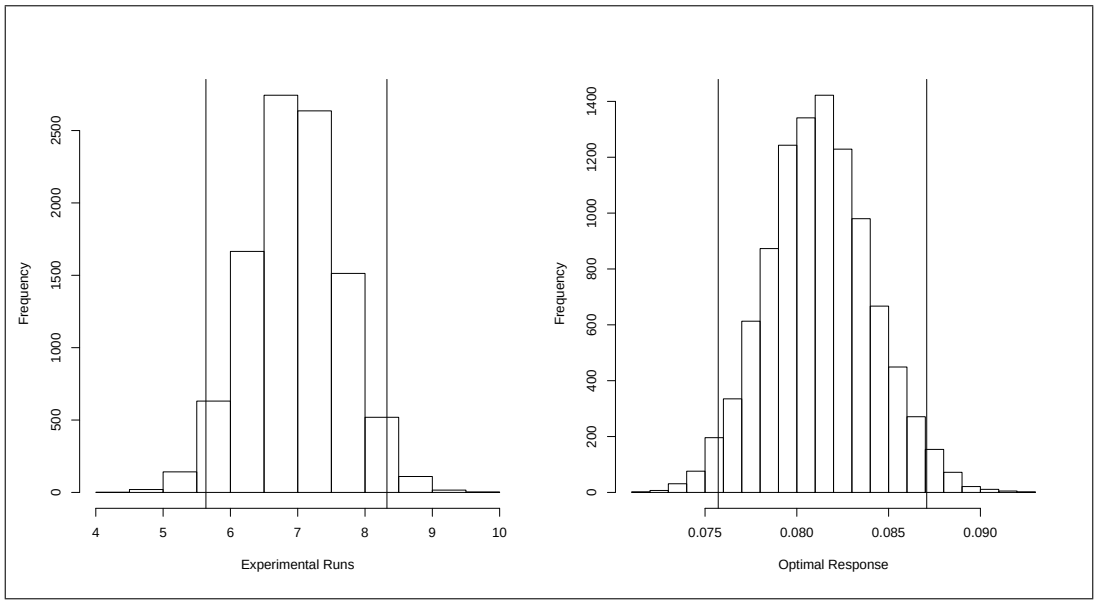

Figure 5.3: 95% bootstrap confidence intervals for Hartman 6 Function: Lack of Fit

We expect that performance of the pseudo lack of fit model transition test is more reasonable for smoother functions. A class of functions we feel are more representative of true computer experiments. Therefore, we direct our efforts towards the distillation column computer experiment, which exhibits a smoother response surface.

Like the previous two chapters, we look at two different situations for the real computer experiment. The first, assumes we know nothing about the process or that we are blind as

to where to start in the region of operability. The second, uses process knowledge to begin in a region of good processing conditions. Table 5.4 and Figure 5.4 show 95% bootstrap confidence intervals comparing the model transition techniques, assuming with begin with no information about the process.

Table 5.4: Distillation Column Results: Lack of Fit

Method	Yield	Experimental Runs
RSO Lack of Fit	0.9399	61.97
RSO Steepest Ascent	0.9279	57.95
Difference	0.0120(0.0105)	4.02(1.10)

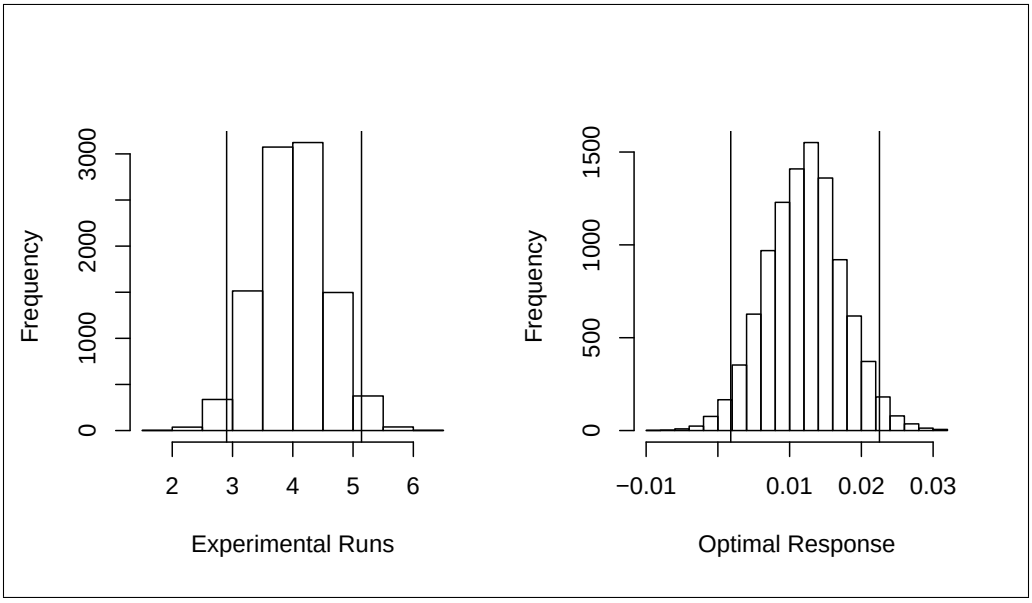

Figure 5.4: 95% bootstrap confidence intervals for Distillation Column: Lack of Fit

Much like the Hartman 6 test function, applying the pseudo lack of fit test as a model transition strategy yields poor performance in the number of experimental runs with little or no improvement estimating the optimal yield. This holds true even when we apply prior knowledge about the system at the beginning of experimentation, see Table 4.6.

Table 5.5: Effect of Prior Knowledge: Lack of Fit

Method	Yield	Experimental Runs
RSO Lack of Fit	0.9952	66.12
RSO Steepest Ascent	0.9989	48.42
Difference	−0.0038(0.00084)	17.69(0.63)

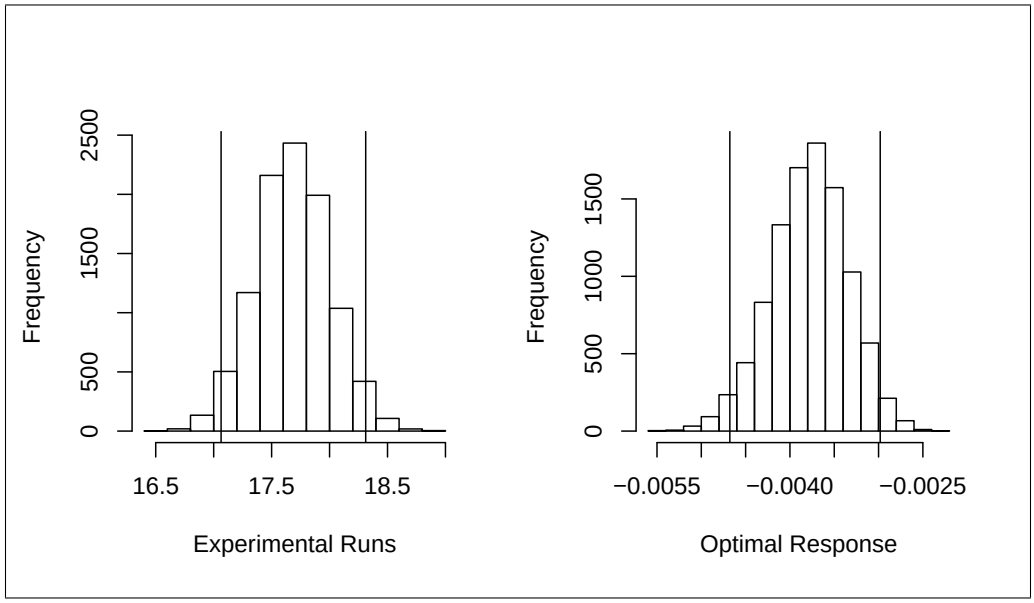

Figure 5.5: Effect of Prior Knowledge on RSO Algorithm: Lack of Fit

Although the pseudo lack of fit test is not a suitable technique for model transitions within the RSO algorithm, it did demonstrate the capacity for examining the white noise inside the process. Section 5.4.2 looks at applying the pseudo lack of fit test as a post confirmation strategy to examine the white noise near the optimum response.

5.4.2 Post Confirmation

The results in the previous section indicate that the steepest ascent model transition method is favored over the pseudo lack of fit technique presented in this chapter. In lieu of its poor performance, the lack of fit test still provided us with some insight on the white noise inside the process. Thus, we propose adding a twelfth step (post confirmation) to the RSO

algorithm to help provide us with a better estimate of the optimum response. Since there is little room for improvement on the estimates from the distillation column and Branin function examples, we use the Hartman 6 function to demonstrate the twelve step algorithm.

Twelve Step RSO Algorithm

We sample from the Hartman 6 function via a sobol sequence space filling design. The rule of thumb discussed in Chapter 3 uses a design on the order of $4k$. Thus, the initial experimental design has 24 experimental runs ($k = 6$ factors). The values for the generated design are comprised in Table 5.6.

Table 5.6: Space Filling Design: Distillation Column

x_1	x_2	x_3	x_4	x_5	x_6	y
0.943604	0.620621	0.425840	0.124702	0.266042	0.377939	2.14236
0.163548	0.898121	0.839421	0.495842	0.780408	0.117312	0.01760
0.575996	0.226170	0.075725	0.952720	0.228226	0.511123	4.11725
0.400355	0.683124	0.164693	0.312534	0.424647	0.809396	0.46568
0.862665	0.448916	0.936275	0.863415	0.566658	0.327703	5.38633
0.017972	0.038673	0.329226	0.734547	0.073302	0.176467	3.66596
0.742979	0.835624	0.569563	0.191433	0.935152	0.695255	5.19482
0.921784	0.078003	0.550180	0.384648	0.489225	0.442631	1.46948
0.317631	0.812423	0.286597	0.935344	0.502322	0.927744	5.51493
0.535621	0.706798	0.948150	0.556416	0.012874	0.575320	2.23279
0.201969	0.410121	0.215805	0.013607	0.995826	0.060928	3.83035
0.776330	0.999920	0.056472	0.627395	0.657086	0.259591	2.06620
0.484736	0.140500	0.796662	0.178071	0.334276	0.869695	-0.36236
0.641997	0.347618	0.437829	0.299163	0.836923	0.626662	2.95587
0.120908	0.519295	0.709281	0.756333	0.171467	0.237247	1.86555
0.147232	0.212603	0.620085	0.255036	0.555893	0.424312	-0.07943
0.621503	0.915530	0.356371	0.797866	0.452807	0.938712	4.89085
0.278634	0.568030	0.878368	0.676804	0.938364	0.556761	3.22382
0.992168	0.302359	0.145909	0.126128	0.053184	0.071641	4.91238
0.035025	0.853029	0.109645	0.514385	0.356534	0.373954	1.71309
0.696737	0.025102	0.849965	0.057194	0.651856	0.763362	2.43846
0.414354	0.489858	0.384535	0.436153	0.145318	0.740770	0.06335
0.815326	0.630529	0.656100	0.885456	0.846044	0.130673	3.35546
0.521252	0.404157	0.729987	0.694653	0.292200	0.882320	0.21869

Step 1: Initial Characterization

Using the initial design from Table 5.6, we use the methods from Jones *et al.* (1998) to estimate the parameters for the Gaussian stochastic process emulator and its covariance. Equation 2.4 provides the formula for the GASP emulator. For this example, the formula becomes,

$$Y(\boldsymbol{x}) = 3.838341 + Z(\boldsymbol{x}).$$

Recall that $Z(\boldsymbol{x})$ is the Gaussian stochastic process with mean of 0 and a covariance function equal to

$$cov(t, w) = \sigma_z^2 e^{-\sum_{j=0}^{k} \theta_j |t_j - w_j|^{p_j}}.$$

The remaining $2k+1$ parameters are estimated by choosing them to maximize the likelihood of the sample. The values of μ and σ_Z^2 that maximize the likelihood function can be solved in closed form. These values are then used to get a likelihood function that depends solely on θ and $\mathbf{p}$. It is this function that is maximized to get the estimates of $\hat{\theta}$ and $\hat{\mathbf{p}}$, see Jones *et al.* (1998). The estimated parameters for the example are $\hat{\sigma_z^2} = 0.058835$, $\hat{\theta} =$ (0.14151, 0.00000, 0.17654, 2.94703, 11.34867, 0.47965), and $\hat{\mathbf{p}} =$ (0.60971, 0.92913, 0.02991, 0.00013, 0.00046, 0.00251).

Step 2: Validation

Chapter 3 presents the basis for the confidence interval for any $y(\hat{x}_0)$, see Equation 3.1. The lower and upper limits for the confidence interval of each validation data point is presented in Table 5.7. Data within the confidence limits indicate good performance. In this example, the GASP model yields sufficient performance. Thus, we do not have to re-characterize the

GASP model (Step 3) and we skip directly to Step 4, where we investigate the response surface for a sweet spot.

Table 5.7: Cross Validation Results

Lower Limit	Upper Limit	Validation Data	Lower Limit	Upper Limit	Validation Data
1.52122	1.82680	2.14236	-0.13749	0.13509	-0.36236
0.92575	1.34045	0.01760	3.04849	3.24359	2.95587
2.47255	2.88925	4.11725	2.27590	2.52746	1.86555
0.52649	0.74263	0.46568	0.51321	0.64141	-0.07943
4.23735	4.52233	5.38633	3.99107	4.21955	4.89085
2.83749	3.06919	3.66596	3.01756	3.26272	3.22382
4.24065	4.49981	5.19482	3.81457	4.18809	4.91238
1.18470	1.42596	1.46948	0.87859	1.10297	1.71309
6.06290	6.36948	5.51493	1.44007	1.78665	2.43846
1.83621	2.28295	2.23279	0.53959	0.67263	0.06335
4.66970	5.01844	3.83035	2.60797	2.94401	3.35546
2.52481	2.87349	2.06620	1.37988	1.55942	0.21869

Step 4: Locate Sweet Spot

We are interested in maximizing the response, so we use the sample maximum to locate the initial sweet spot located at $\boldsymbol{x}$ = (0.4847357, 0.1405000, 0.7966616, 0.1780709, 0.3342757, 0.8696955).

Step 5: Emulation

The GASP model provided in Step 2 is used to explore the sweet spot further. Currently, the sweet spot is a point estimate, i.e. the sample maximum. Once a GaSP model is fit to realized outcomes, the computer model is easy to predict in unsampled regions of the input space via Kriging, Bayesian, or Bayes Linear prediction techniques. We use the GASP model to fine tune the sweet spot. Since we consider the runs from the emulator as free (runs are extracted instantaneously), we use a large design size to get a good understanding of the neighborhood around a sweet spot.

We generate a space filling experimental design 100 times the size of the original design (2400 experimental runs) around the sweet spot to get predictions in those unsampled regions. Since the design size is extremely large we will not display these results. We do, however, provide an example of how this data is used to make decisions in the next step.

Step 6: Probability Plot

This step is where we begin to merge into the RSM component of the RSO algorithm. Obviously, not all of the predictions from the previous step are going to be of value. Depending on the regularity of the processes response surface, some of these predictions may take us farther from the sweet spot or optimum response. Therefore, we look at the top 5% of the data in Step 5 in terms of the optimization objective, i.e. minimum, maximum, target.

Figure 5.6 shows an example of how we use the data from Step 5 to focus in on a more appropriate sweet spot. This figure displays a Normal quantile plot of one of the six factors. These plots provide a way to construct the initial levels for the fractional factorial. The initial levels are selected by the 25^{th} and 75^{th} percentiles of the quantile plot. In Figure 5.6 the 25^{th} and 75^{th} percentiles 0.2965997 are 0.3166980 respectively. The remaining levels for the example are shown in Table 5.8.

Table 5.8: Initial Fractional Factorial Levels

Factor	25^{th} Percentile	75^{th} Percentile
x_1	0.455231	0.490688
x_2	0.118510	0.157860
x_3	0.787577	0.823960
x_4	0.216241	0.224297
x_5	0.296600	0.316698
x_6	0.870205	0.904022

Step 7. Fractional Factorial

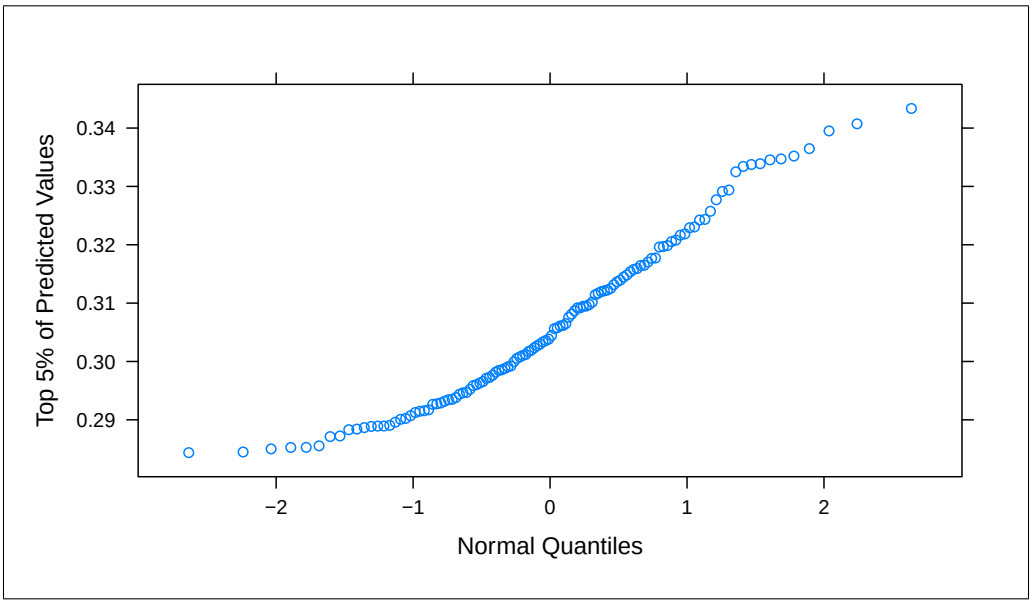

Figure 5.6: Fractional Factorial Level Selection

Using the levels from the previous step, we can set up the initial experimentation for the response surface component of the RSO algorithm. This is shown in Table 5.9.

Table 5.9: Initial Fractional Factorial Experiment

x_1	x_2	x_3	x_4	x_5	x_6	y
0.455231	0.118510	0.787577	0.224297	0.316698	0.904022	-0.385695
0.490688	0.118510	0.787577	0.216241	0.296600	0.904022	-0.347992
0.455231	0.157860	0.787577	0.216241	0.316698	0.870205	-0.498839
0.490688	0.157860	0.787577	0.224297	0.296600	0.870205	-0.476103
0.455231	0.118510	0.823960	0.224297	0.296600	0.870205	-0.431418
0.490688	0.118510	0.823960	0.216241	0.316698	0.870205	-0.365797
0.455231	0.157860	0.823960	0.216241	0.296600	0.904022	-0.405039
0.490688	0.157860	0.823960	0.224297	0.316698	0.904022	-0.353909
0.472959	0.138185	0.805768	0.220269	0.306649	0.887113	-0.414106

Step 8: Steepest Ascent

Using the Box and Meyer method to select significance we see that x_1, x_2, x_3, and x_6 are significant. We begin the method of steepest ascent by selecting the input with the largest estimated coefficient ($x_6 = 2.066$) and then determining the increment of the other active

process variables. These increments are listed in Table 5.10.

Table 5.10: Steepest Ascent Increment

Factor	β	Increment (coded)	Increment (uncoded)
x_1	1.249	-0.6045927	-0.01071854
x_2	-1.289	0.6241049	0.01227906
x_3	1.047	-0.5069926	-0.009222943
x_6	2.066	-1.00	-0.01690884

Using these increments we go through one iteration of the method of steepest ascent, shown in Table 5.11. Clearly, we have not reached a plateau in the response and another iteration of steepest ascent is needed. After five more iterations of the method of steepest ascent, we reach a plateau in the response. This process yields the center point for the optimization experiment of the next step, $x^* = (0.1897757, 0.1496875, 0.4826979, 0.2686022, 0.3166980, 0.6626320)$.

Table 5.11: Steepest Ascent Example

Run	x_1	x_2	x_3	x_4	x_5	x_6	Yield
1	0.462241	0.150464	0.796545	0.220269	0.306649	0.870205	-0.485609
2	0.451522	0.162743	0.787322	0.220269	0.306649	0.853296	-0.550794
3	0.440804	0.175022	0.778100	0.220269	0.306649	0.836387	-0.609947
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
14	0.322900	0.310092	0.676647	0.220269	0.306649	0.650390	-0.914975
15	0.312181	0.322371	0.667424	0.220269	0.306649	0.633481	-0.912562

Step 9: Optimization Experiment

We construct the optimization experiment by augmenting the previous fractional factorial into a second order design (CCD). Using the central composite design in Table 2.1 as a template, we add axial runs (α) to the fractional factorial from the last iteration of the method of steepest ascent, where x_1, x_5, and x_6 are significant. The axial runs for factors three and six are $\alpha_1 = (0.16, 0.22)$, $\alpha_5 = (0.30, 0.33)$ and $\alpha_6 = (0.64, 0.69)$ respectively.

Step 10: Stationary Point

Using a desirability function we can determine a maximum stationary point for this example, $x^* = (0.1897757, 0.1496875, 0.4826979, 0.2686022, 0.3166980, 0.6626320)$ where the response at this point is $y = -1.2003$.

Step 11: Confirmatory Experiment / Runs

Building a confirmatory experiment is an important step, it reassures us that we are operating close to a optimum response. If you do not have enough resources for an entire experiment, a couple of runs are sufficient. In this example we develop a central composite design as the confirmatory experiment, see Table 5.12.

Table 5.12: Confirmatory Experiment

x_1	x_5	x_6
0.183210	0.301814	0.640247
0.218667	0.301814	0.640247
0.183210	0.321912	0.640247
0.218667	0.321912	0.640247
0.183210	0.301814	0.674065
0.218667	0.301814	0.674065
0.183210	0.321912	0.674065
0.218667	0.321912	0.674065
0.200939	0.311863	0.657156
0.227532	0.311863	0.657156
0.227532	0.311863	0.657156
0.200939	0.326937	0.657156
0.200939	0.326937	0.657156
0.200939	0.311863	0.682519
0.200939	0.311863	0.682519

Step 12: Post Confirmation

The post confirmation step begins by augmenting the confirmatory experiment with the four runs that generate pseudo error. Table 5.13 show these four runs.

Table 5.13: Pseudo Error Experimental Runs

x_1	x_2	x_3	x_4	x_5	x_6
0.200939	0.140022	0.488065	0.269157	0.311863	0.657156
0.200939	0.133654	0.492239	0.264718	0.311863	0.657156
0.200939	0.160678	0.465051	0.268076	0.311863	0.657156
0.200939	0.148506	0.498940	0.266904	0.311863	0.657156

These runs allow us to construct the pure error and lack of fit sums of squares that are needed to perform the pseudo lack of fit test. In our example we have $SS_{PE} = 0.00000198$ and $SS_{LOF} = 0.00003666$, which allows us to construct the test statistic for the pseudo lack of fit test,

$$F = \frac{SS_{LOF}/(m-p)}{SS_{PE}/(n-m)} = 18.47316$$

This test statistic has a p-value of 0.007640466. Therefore, reject the null hypothesis and conclude that the second order model approximation does not adequately fit to the data. In order to estimate the optimum in this area, while keeping the number of experimental runs, we look at additional higher order interactions in our model and estimate the optimum using a desirability function. This method provides an optimum estimate of $y = -1.200679$. Clearly, we have improved our estimate slightly by this twelfth step. We investigate the impact on performance for five thousand samples in Section 5.4.3.

5.4.3 Post Confirmation Simulation

The detailed example in the previous section introduces the benefit of applying a post confirmation step to the RSO algorithm. We see that by applying a twelfth step the the RSO algorithm, we can improve our estimate of the optimum response. This, however, is only a single sample. Table 5.14 shows simulation results for five thousand samples of the application of the twelve step RSO algorithm.

Table 5.14: Hartman 6 Function Results: Post Confirmation

Method	Optimum Estimate	Experimental Runs
EGO algorithm	−1.1767(0.0021)	105.29(0.99)
11 Step RSO	−1.1702(0.0014)	79.14(0.66)
12 Step	−1.1741(0.0014)	95.52(0.79)

True Optimum: -1.20068

Consistent with the detailed example, Table 5.14 shows that the twelve step RSO algorithm yields a significantly better optimum estimate than the eleven step algorithm. This comes at a cost, however, of roughly sixteen experimental runs. This is not surprising, since we need the extra experimental runs to construct the optimum experiment and the runs required to generate pseudo error. Therefore, we encounter a tradeoff between estimating the optimum response and the total number of computer evaluations. In practical problems, this tradeoff stands as a decision tool for which RSO algorithm is preferred. In either case, results show that performance of the RSO excels that of the EGO algorithm, a leading benchmark in the literature.

5.5 Conclusions

One of the biggest differences between traditional response surface methods and the RSM component of the RSO algorithm is its model traditional method. In Chapters 3 and 4, the RSO algorithm uses the method of steepest ascent to transition between first and second order models. This chapter presents an alternative method that attempts to mimic the lack of fit model transition test often seen in traditional methods. The 'pseudo' error for the lack of fit test is generated by allowing inert factors to roam freely, which mimics the overall thought process of physical experimentations.

We compare the two model transition techniques using two out of the three examples from

Chapters 3 and 4. The Branin function was not used because we encountered situations where both factors in the test function were active. In this case, the pseudo lack of fit test cannot be constructed. Clearly, this is a special case for experiments with low dimensions; under the factor sparsity principle, we would expect this not to be an issue for high dimension.

Using the "pseudo" lack of fit test as a model transition technique has a negative effect on performance overall. For the Hartman 6 function, we see that the method of steepest ascent provides a better optimum estimate in less experimental runs than the pseudo lack of fit test. In addition, the results of the pseudo lack of fit test are beat by the EGO algorithm, the benchmark in performance from Chapters 3 and 4. We see the same results for the water-ethanol distillation column. For this real computer experiment, we achieve similar or marginally better estimate of the optimum but at the cost of many experimental runs.

Clearly, substituting the method of steepest ascent with the pseudo lack of fit test is undesirable. In lieu of its poor performance, the lack of fit test still provided us with some insight on the white noise inside the process, which is of value when applying the RSO algorithm to practical problems. Thus, we benefit from the pseudo lack of fit as a post confirmation step in the RSO algorithm, providing a path for improved performance when estimating the optimum. This, however, comes at the cost of a handful of experimental runs. This step also serves as an intermediate step when no field data exist as we begin to progress from the computer experiment to the implementation of the physical process, product, or system. The implementation of pseudo error gives us some idea about the white noise in the region before entering the physical world.

Chapter 6

Concluding Remarks

The primary goal of this research is to improve the current optimization methods for black-box systems. Historically, these methods tend to involve an iterative search of the function until a specific stopping criterion is met. After stopping, we check to see which evaluation resulted in a optimal point. We present a twelve step process, the response surface optimization algorithm (RSO), that moves away from this 'pick the winner' strategy and actively searches for an optimum estimate through the sequential learning strategy of response surface methodology.

A leading benchmark in the computer experiment optimization literature is the efficient global optimization (EGO) by Jones *et al.* (1998). In Chapter 3, we compare the RSO and EGO algorithms by applying the algorithms to two test functions commonly seen in the literature and a real computer experiment that simulates a water-ethanol distillation column. The analysis of simulations run on each test function shows that RSO compared favorably to EGO in terms of optimal response and experimental runs. We see a stronger performance of the RSO algorithm for the Branin function, a relatively smooth function with lower dimensions. However, in the presence of a highly irregular surface (Hartman 6), we see a tradeoff in performance of the optimum estimate and the number of experimental runs

to achieve it.

Like physical experiments, we feel computer experiments with irregular or 'bumpy' surfaces are the exception rather than the rule, however, we apply an adjustment to the RSO algorithm in Chapter 4 that will help approximate these irregular functions. The main concern of the algorithm in Chapter 3 is its use of subjective plots to select active factors within the process. Adjusting the RSO algorithm to incorporate the Box and Meyer method provides a more objective measure for selecting these active factors.

In the presence of smooth functions the performance remains fairly stable. When applying the Box and Meyer method to the Hartman 6 function, on the other hand, we no longer encounter a tradeoff as the RSO algorithm compares favorably to the leading benchmark. Applying the RSO algorithm to the real computer experiment is where we see its clear advantage over the EGO algorithm.

This advantage is heightened further when we incorporate process knowledge throughout experimentation, a common practice in industry. These results show that for a single response the RSO algorithm can be an attractive option for optimizing computer experiments. In practice, we often have multiple responses of interest that interact with each other. For example, in the water-ethanol computer experiment we could be concerned with the profit of development as well as the percent yield. The analysis of multiple responses may take us into an area with different operating conditions. One logical expansion to this research is to investigate the impact of the RSO algorithm in the presence of multiple responses.

In practice, it is helpful to have some idea about the white noise in the process. A difficult task in the face of deterministic models. In Chapter 5, we present a pseudo lack of fit test that generates pseudo error by allowing inert factors to roam freely with a certain range.

Although the test proves inefficient for transitions between first and second order models, it allows us to understand the experimental error as we make this transition from computer experiment to the physical process or product.

For complex computer experiments, it is desirable to keep the number of experimental runs low since it can take hours, days, and sometimes months to compute a single data point. The biggest advantage of the RSO algorithm is its performance in keeping these runs relatively low. This begins with the recommended $4k$ runs for the initial space filling design. A research question we have not addressed is the impact of the size of this initial design. An ideal situation would optimize the RSO algorithm in response to this initial design size.

A vast majority of the computer experiment field is split into two areas: optimization and characterization. The research presented here is primarily concerned with optimization. Characterization is only handled in the first three steps of the RSO algorithm. One interesting research topic would be investigating the impact of characterizing throughout the entire algorithm. Thus, in the process of searching for an optimum estimate, we develop a sufficient characterization of the computer experiment as well.

Another area of future research is the potential of other subjects in response surface methodology. Below is a list of possible subjects of RSM that were not addressed in this research:

- Experiments with Mixtures: A mixture experiment is a special case of RSM experiments, where the factors are components of a mixture and the response is a function of the proportions of each factor.
- Split Plot Experiments: In industry, some situations prohibit complete randomization of the order of experimental runs. As we transition from computer experiment to

physical experimentation where split plot experiments are necessary we can use pseudo error to understand the experimental error of the whole plots and split plots.

- Robust Parameter Design: Often, we divide the factors in the system into two categories: control factors and noise factors. Control factors are ones that are set by the experimenter. For example, we easily set the reboil temperature of a water-ethanol distillation column. Noise factors are factors that are difficult or impossible to control. For example, room temperature may also have an effect on the response but is difficult to control. The goal of robust parameter design is to make products more robust to these noise factors.

- Computer Generated Designs: A common group of experimental designs are those designed to optimize some sort of criterion, i.e. D-optimal designs. These designs are convenient since they are generated by a computer and may be of benefit when classical designs do not apply.

- Compensate for model discrepancy to estimate the optimal response, not just the simulated optimal response

Bibliography

[1] Aslett, R. ; Buck, R. J. ; Duvall, S. G. ; Sacks, J. ; and Welch, W. J. (1998). "Circuit Optimization Via Sequential Computer Experiments: Design of an Output Buffer". *Applied Statistics*, 47, pp. 31–48.

[2] Bates, R. A. ; Buck, R. J. ; Riccomangno, E. ; and Wynn, H. P. (1996). "Experimental Design and Observation for Large Systems". *Journal of the Royal Statistical Society B*, 58, pp. 77–94.

[3] Bayarri, M. J. ; Berger, J. O. ; Daulo, R. ; Sacks, J. ; Cafeo, J. A. ; Cavendish, J. ; Lin, C. and Tu, J. (2007). "A Framework for Validation of Computer Models". *Technometrics*, 49, pp. 138–154.

[4] Bernardo, M. C. ; Buck, R. J. ; Liu, L. ; Nazaret, W. A. ; Sacks, J. ; and Welch, W. J. (1992). "Integrated circuit design optimization using a sequential strategy". *IEEE Transactions on Computer-Aided Design*, 11, pp. 361–372.

[5] Blight, B. J. N. and Ott, L. (1975). "A Bayesian Approach to model inadequacy for polynomial regression". *Biometrika*, 62, pp. 79–88.

[6] Box, G. E. P. (1999). "Statistics as a Catalyst to Learning by Scientific Method Part II - A Discussion". *Journal of Quality Technology*, 31, pp. 16–29.

[7] Box, G. E. P. and Draper, N. R. (1975). "Robust Designs". *Biometrika*, 62, pp. 347–352.

[8] Box, G. E. P. and Hunter, J. S. (1957). "Multi-Factor Experimental Designs for Exploring Response Surfaces". *The Annals of Mathematical Statistics*, 28, pp. 195–241.

[9] Box, G. E. P. and Liu, P. Y. T. (1999). "Statistics as a Catalyst to Learning by Scientific Method Part I - An Example". *Journal of Quality Technology*, 31, pp. 1–15.

[10] Box, G. E. P. and Meyer, R. D. (1986). "An Analysis for Unreplicated Fractional Factorials". *Technometrics*, 28, pp. 11–17.

[11] Box, G. E. P. and Wilson, K. B. (1951). "On the Experimental Attainment of Optimum Conditions". *Journal of the Royal Statistical Society B*, 13, pp. 1–45.

[12] Chaloner, K. and Verdinelli, I. (1995). "Bayesian Experimental Design: A Review". *Statistical Science*, 10, pp. 273–304.

[13] Craig, P. S. ; Goldstein, M. ; Rougier, J. C. ; and Seheult, A. H. (2001). "Bayesian Forecasting for Complex Systems Using Computer Simulators". *Journal of the American Statistical Association*, 96, pp. 717–729.

[14] Cressie, N. A. (1993). *Statistics for Spatial Data*. J. Wiley, New York.

[15] Currin, C. ; Mitchell, T. ; Morris, M. ; and Ylvisaker, D. (1991). "Bayesian Prediction of Deterministic Functions, with Applications to the Design and Analysis of Computer Experiments". *Journal of the American Statistical Association*, 86, pp. 953–963.

[16] Derringer, G. and Suich, R. (1980). "Simultaneous Optimization of Several Response Variables". *Journal of Quality Technology*, 12, pp. 214–219.

[17] Goldstein, M. and Rougier, J. C. (2003). "Calibrated Bayesian Forecasting using large computer simulators". technical report, University of Durham, Statistics and Probability Group.

[18] Hastings, W. (1970). "Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, pp. 97-109.

[19] Helton, J. C. (1993). "Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal". *Reliability Engineering and System Safety*, 42, pp. 327–367.

[20] Higdon, D. ; Gattiker, J. ; Williams, B. ; and Righley, M. (2008). "Computer Model Calibration Using High-Dimensional output". *Journal of the American Statistical Association*, 103, pp. 570–583.

[21] Huang, D. ; Allen, T. T. ; Notz, W. I. ; and Zeng, N. (2006). "Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models". *Journal of Global Optimizatin*,34, pp. 441–466.

[22] Iman, R. L. and Helton, J. C. (1988). "An investigation of uncertainty and sensitivity analysis techniques for computer models". *Risk Analysis*, 8, pp. 71–90.

[23] Jones, D. R. ; Schonlau, M. ; and Welch, W. J. (1998). "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization*, 13, pp. 455–492.

[24] Kiefer, J. (1959). "Optimum Experimental Designs". *Journal of the Royal Statistical Society B*, 21, pp. 272-319.

[25] Kimeldorf, G. S. and Wahba, G. (1970). "A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines". *The Annals of Mathematical Statistics,*, 41, pp. 495–502.

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

[27] Kennedy, M. C. and O'Hagan, A. (2001). "Bayesian calibration of computer models". *Journal of the Royal Statistical Society B*, 63, pp. 425–464.

[28] Mckay, M. D. ; Conover, W. J. ; and 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, pp. 239–245.

[29] Metropolis, N. Rosenbluth, A. W. ; Rosenbluth, M. N. ; Teller, A. H. ; and Teller, E. (1953). "Equations of state calculations by fast computing machines". *Journal of Chemical Physics*, 21, pp. 1087-1091.

[30] Morris, M. D. ; Mitchell, T. J. ; and Ylvisaker, D. (1993). "Bayesian design and analysis of computer experiments: use of derivatives in surface prediction". *Technometrics*, 35, pp. 243–255.

[31] Morris, M. D. and Mitchell, T. J. (1995). "Exploratory designs for computational experiments". *Journal of Statistical Planning and Inference*, 43, pp. 381–402.

[32] Myers, R. H. ; Montgomery, D. C. and Anderson-Cook, C. M. (2009). *Response Surface Methodology: Process and Product Optimization using Designed Experiments*, 3rd Ed. New York: Wiley.

[33] Oakley, M. C. and O'Hagan, A. (2004). "Probabilistic sensitivity analysis of complex models: a Bayesian approach". *Journal of the Royal Statistical Society B*, 66, pp. 751–769.

[34] O'Hagan, A. (2006). "Bayesian Analysis of Computer Code Outputs: A Tutorial". *Reliability Engineering and System Safety*, 91, pp. 1290–1300.

[35] O'Hagan, A. and Kingman, J. F. C. (1978). "Curve Fitting and Optimal Design for Prediction". *Journal of the Royal Statistical Society B*, 40, pp. 1–42.

[36] Perry, R. H. and Chilton, S. M.(2006). *Chemical Engineers' Handbook*. McGraw-Hill: New York.

[37] Sacks, J. ; Schiller, S. B. ; and Welch, W. J. (1989). "Designs for Computer Experiments". *Technometrics*, 31, pp. 41–47.

[38] Sacks, J. ; Welch, W. J. ; Mitchell, T. J. ; and Wynn, H. P. (1989). "Design and Analysis of Computer Experiments". *Statistical Science*, 4, pp. 409–435.

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

[40] Villemonteix, J. ; Vazquez, E. ; Sidorkiewicz, M. and Walter, E. (2009). "Global optimization of expensive-to-evaluate functions: an empirical comparison of two sampling criteria". *Journal of Global Optimization*, 43, pp. 373–389.

[41] Villemonteix, J. ; Vazquez, E. and Walter, E. (2009). "An informational approach to the global optimization of expensive-to-evaluate functions". *Journal of Global Optimization*, 44, pp. 509–534.

[42] Vining, G. G.(2008). "Adapting Response Surface Methodology for Computer and Simulation Experiments". *The Grammer of Technology Development*, Japan: Springer

[43] Vining, G. G. and Kowalski, S. M.(2006). *Statistical Methods for Engineers*. Brooks-Cole: Belmont, CA.

[44] Welch, W. J. ; Buck, R. J. ; Sacks, J. ; Wynn, H. P. ; Mitchell, T. J. ; and Morris, M. D. (1992). "Screening, Predicting, and Computer Experiments". *Technometrics*, 34, pp. 15–25.

[45] Williams, B. J. ; Santner, T. J. ; and Notz, W. I. (2000). "Sequential Design of Computer Experiments to Minimize Integrated Response Functions". *Statistica Sinica*, 10, pp. 1133–1152.