Computer Experiments and Global Optimization

by

Matthias Schonlau

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosphy
in
Statistics

Waterloo, Ontario, Canada, 1997

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

A complex mathematical model that produces output values from input values is now commonly called a computer model. This thesis considers the problem of finding the global optimum of the response with few function evaluations. A small number of function evaluations is desirable since the computer model is often expensive (time consuming) to evaluate.

The function to be optimized is modeled as a stochastic process from initial function evaluations. Points are sampled sequentially according to a criterion that combines promising prediction values with prediction uncertainty. Some graphical tools are given that allow early assessment about whether the modeling strategy will work well. The approach is generalized by introducing a parameter that controls how global versus local the search strategy is. Strategies to conduct the optimization in stages and for optimization subject to constraints on additional response variables are presented.

Special consideration is given to the stopping criterion of the global optimization algorithm. The problem of achieving a tolerance on the global minimum can be represented by determining whether the first order statistic of N dependent variables is greater than a certain value. An algorithm is developed that quickly determines bounds on the probability of this event.

A strategy to explore high-dimensional data informally through effect plots is presented. The interpretation of the plots is guided by pointwise standard errors of the effects which are developed. When used in the context of global optimization, the graphical analysis sheds light on the number and location of local optima.

Acknowledgements

My thesis would have been impossible without the invaluable guidance of my supervisor William J. Welch. His overflowing wealth of ideas, his remarkable ability to teach scientific writing, and his insight into numerical aspects of programming have greatly influenced both me and my thesis.

I thank my thesis committee, Angela Dean (external examiner), Jerald F. Lawless, Lara J. Wolfson, and Henry Wolkowicz for their careful reading and helpful suggestions. Markus Abt gave valuable comments on an earlier draft of Chapter 3.

I am very grateful to Michael Hamada, University of Michigan at Ann Arbor, for involving me in the solar collector project which now forms Chapter 3 of my thesis. His excitement for research is exceptionally contagious.

I am very grateful to Donald R. Jones, General Motors R&D Center, Warren, Michigan, for his initial involvement of what now forms Chapter 4, in particular for pointing out the expected improvement algorithm. Donald Jones also went out of his way to provide us with the Piston Data set for Section 5.4.1.

I also thank S. Cao (now at the National Research Council of Canada) and T. Hollands, Solar Thermal Lab, Department of Mechanical Engineering, University of Waterloo, who were involved on the engineering side in the solar collector application.

A big thanks goes also to Robert Zvan, Department of Computer Science, University of Waterloo, and the MFCF computer consultants who patiently put up with all my programming questions. A late night pub discussion with Edward Carr, Department of Computer Science, University of Waterloo, led to the formulation and proof of Theorem 3.

My parents still think I should have done my Ph.D. in Germany but supported me unconditionally nonetheless. I find that truly remarkable and I am most grateful for their support.

Rekha Agrawal was a fantastic friend throughout all these years. The German saying "someone to steal horses with" alludes to the trust and confidence bestowed upon someone you would be willing to commit a crime with. Rekha, I would steal horses with you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis is about global optimization of expensive-to-compute computer models. The approach that we take is closely linked to computer experiments in that we repeatedly use methodology developed for computer experiments for modeling the unknown function to be optimized. The thesis is organized as follows:

Chapter 2 gives a review of the analysis of computer experiments and comments briefly on the connection with Kriging, a stochastic method of spatial prediction.

As well as showing how to identify key features of computer models, Chapter 3 presents an illustrative example for modeling a computer experiment. Even though Chapter 3 is not about optimization, some of its aspects (particularly visualization) are inherently useful for the data-analytic approach to optimization that we adopt later on. Along the way we introduce novel methodology for attaching standard errors for the estimates of main effects and interactions. We also present a method for finding a suitable nonlinear regression model when the functional relationship between response and explanatory variables is unknown.

Chapter 4 presents a data-analytic approach for global optimization. Novel aspects include the use of diagnostics before optimizing an expensive-to-compute-

function, using methodology for the analysis of computer experiments in the context of optimization, and the availability of a reliable stopping rule.

Chapter 5 takes this approach to global optimization a step further adding three novel aspects: (i) a parameter that controls the balance between local and global components of the optimization; (ii) methodology for optimization in stages rather than one-point-at-a-time; and (iii) optimization subject to constraints on additional response variables.

Chapter 6 gives an algorithm for the evaluation of the CDF of the minimum of N dependent variables. The algorithm is particularly fast when it is sufficient to specify whether certain bounds on the CDF are met. The algorithm incorporates three ideas that make it fast and therein lies the novelty. While this is a topic in its own right, we show how it can also be used as an alternative stopping rule for the global optimization algorithm introduced in Chapter 4.

This thesis is computationally intensive. The examples shown throughout were generated with the following software: ACED (Algorithms for the Construction of Experimental Designs), software developed by William J. Welch, was used throughout the thesis for all design aspects for computer experiments. GASP (GAussian Stochastic Processes), also software by William J. Welch, was used in Chapter 3 for the analysis of computer experiments and for Figures 4.8 and 4.9. SPACE (Stochastic Processes Analysis of Computer Experiments), software by Matthias Schonlau, was used for the analysis and optimization throughout except for Chapter 3 and Figures 4.8 and 4.9. Appendix A contains a brief overview of the major components that a computer program for the design, analysis and optimization of computer models must contain.

Chapter 2

Review

2.1 The Analysis of Computer Experiments

A complex mathematical model that, given a set of input values, produces a set of output values is now commonly referred to as a computer model. The name stems from the necessity to have computers do the extensive computations, as almost always the model cannot be written in closed form and/or it requires an iterative solution. Computer models are distinct from models of data from physical experiments in that they are often not subject to random error. A computer (the same computer architecture) fed with the same input will always produce the same output. Due to the lack of random error, traditional modeling approaches are not useful. For example, one of the principles of design of experiments, replication, leads to redundant information in computer experiments.

This section gives a overview of the analysis of computer experiments. Relevant references include Currin et al. (1991), Mitchell and Morris (1994), Morris, Mitchell, and Ylvisaker (1993), Sacks, Schiller, and Welch (1989), Sacks et al. (1989), and Welch et al. (1992).

The Deterministic Model

The data from a computer experiment consist of n vectors of covariate values (or inputs) denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_n$ for d covariates and the corresponding response values (or outputs) $\mathbf{y} = (y_1, \ldots, y_n)^t$. Then the response is modeled by a linear model plus departures from the linear model:

$$\text{Response} = \text{Linear model} + \text{Systematic Departure.}$$

One convenient way of expressing the systematic departure function is to view it as a sample path from a suitably chosen stochastic process. This point of view, namely the resemblance of the systematic departure to a realization of a random function, respects the deterministic nature of a computer code, since a realization of a stochastic process is deterministic, but provides a stochastic framework for assessing uncertainty. The model can be written formally as:

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

where $[f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]^t$ are k known regression functions, $(\beta_1, \beta_2, \ldots, \beta_k)$ are the corresponding parameters, and $Z(\mathbf{x})$ is a stochastic process. As a notational convention we write vectors and matrices in bold letters. The covariance between the Z's at two inputs $\mathbf{x} = (x_1, \ldots, x_d)$, and $\mathbf{x}' = (x_1', \ldots, x_d')$ is

$$\text{Cov}(Z(\mathbf{x}), Z(\mathbf{x}')) = \sigma_z^2 R(\mathbf{x}, \mathbf{x}'), \tag{2.2}$$

where $R(\cdot, \cdot)$ is a correlation function that can be tuned to the data and σ_z^2 is a scale factor, also called the process variance.

We require the stochastic process to be stationary, which implies that $E(Z(\mathbf{x})) =$

0, and that the covariance between the Z's at points $\mathbf{x}$ and $\mathbf{x}'$ depends only on $\mathbf{x} - \mathbf{x}'$, that is on their relative location, not on $\mathbf{x}$ and $\mathbf{x}'$, that is on the locations themselves. For computational reasons it is convenient to choose a correlation function that adopts the so called *product correlation rule*:

$$R(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} R_j(x_j - x_j') \quad .$$

While there are many choices, a sufficiently flexible and commonly used correlation family is the following:

$$R(\mathbf{x}, \mathbf{x}') = \prod_{j=1}^{d} \exp(-\theta_j \, |x_j - x_j'|^{p_j}), \tag{2.3}$$

where $\theta_j \geq 0$ and $0 < p_j \leq 2$. The p_j's can be interpreted as smoothness parameters. The response surface is smoother with respect to x_j as p_j increases. In fact, the correlation function and hence the sample path Z is infinitely differentiable for $p = 2$ in a mean square sense. The θ's indicate how local the estimate is. If the θ's are large, only data points in the immediate vicinity of a given point are highly correlated with that point and are thus influential on the prediction of that point. If the θ's are small, points further away are still highly correlated and still influence the prediction of that point.

The deterministic nature of the problem is kept because $R(\mathbf{x}, \mathbf{x}) = 1$. For this reason, the predictor is an interpolator.

We use the correlation family (2.3) throughout the thesis. With two parameters for each dimension this family is very flexible yet it is not too costly for parameter estimation (see also the discussion in Sacks, Welch, Mitchell, and Wynn, 1989).

Measurement Error

In the presence of measurement error equation (2.1) can be easily modified to

$$Y(\mathbf{x}) = \sum_{j=1}^{k} \beta_j f_j(\mathbf{x}) + Z(\mathbf{x}) + \epsilon(\mathbf{x}), \tag{2.4}$$

where Z is the systematic departure and ϵ is the measurement error. In that case, the variance must reflect this change, Var $(Y(\mathbf{x})) = \sigma_z^2 + \sigma_\epsilon^2$, and the covariance becomes

$$\begin{aligned} \mathrm{Cov}(Y(\mathbf{x}), Y(\mathbf{x}')) &= \sigma_z^2 R(\mathbf{x}, \mathbf{x}') && (2.5) \\ \mathrm{Cov}(Y(\mathbf{x}), Y(\mathbf{x})) &= \sigma_z^2 && (2.6) \\ \mathrm{Var}(Y(\mathbf{x})) &= \sigma_z^2 + \sigma_\epsilon^2. && (2.7) \end{aligned}$$

where the covariances (2.5) and (2.6) strictly refer to two distinct observations (which for (2.6) are replicates), the variance (2.7) refers to only one observation. σ_z^2 is the process variance as defined in (2.2), σ_ϵ^2 is the error variance, and R is the correlation function as defined in (2.3). Equivalently, the correlation in the presence of measurement error is

$$\begin{aligned} \mathrm{Cor}(Y(\mathbf{x}), Y(\mathbf{x}')) &= \frac{\sigma_z^2}{\sigma^2} R(\mathbf{x}, \mathbf{x}') && (2.8) \\ \mathrm{Cor}(Y(\mathbf{x}), Y(\mathbf{x})) &= \frac{\sigma_z^2}{\sigma^2} && (2.9) \end{aligned}$$

where the correlations in (2.8) and (2.9) strictly refer to two distinct observations (which in (2.9) are replicates), and $\sigma^2 = \sigma_z^2 + \sigma_\epsilon^2$.

The fraction σ_z^2/σ^2 constitutes an additional correlation parameter; its value has to be optimized as well. Except for this redefinition the random error model is treated just like the deterministic model.

Best Linear Unbiased Predictor

We now introduce some more notation, before we derive the best linear predictor of Y at an untried input $\mathbf{x}$. Let

$$\mathbf{F} = \begin{pmatrix} \mathbf{f}^t(\mathbf{x}_1) \\ \cdots \\ \mathbf{f}^t(\mathbf{x}_n) \end{pmatrix}$$

be the $n \times k$ expanded design matrix, let

$$\mathbf{r}_x = [R(\mathbf{x}_1, \mathbf{x}), \ldots, R(\mathbf{x}_n, \mathbf{x})]^t$$

be the vector of correlations between the Z's at the design sites $\mathbf{x}_1, \ldots, \mathbf{x}_n$, let

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]^t$$

be the vector of k known regression functions, let $\mathbf{R}$ be the $n \times n$ correlation matrix with element i, j defined by $R(\mathbf{x}_i, \mathbf{x}_j)$ in (2.3) and let the untried input be $\mathbf{x}$. For data $\mathbf{y} = (Y_1, \ldots, Y_n)^t$, the model in (2.1) is written as

$$\mathbf{y} = \mathbf{F}\boldsymbol{\beta} + \mathbf{z} \tag{2.10}$$

where $\mathbf{F}$ is the above defined expanded design matrix, $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_k)$ the corresponding parameters, and $\mathbf{z} = (Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_n))^t$ the stochastic process.

For any linear predictor $\mathbf{c_x^t y}$ of $Y(\mathbf{x})$ the mean squared error of prediction is :

$$\begin{aligned} E(\mathbf{c_x^t y} - Y(\mathbf{x}))^2 &= E\left[\mathbf{c_x^t y y^t c_x} + Y^2(\mathbf{x}) - 2\mathbf{c_x^t y} Y(\mathbf{x})\right] \\ &= E\left[\mathbf{c_x^t}(\mathbf{F}\boldsymbol{\beta} + \mathbf{z})(\mathbf{F}\boldsymbol{\beta} + \mathbf{z})^t \mathbf{c_x} + (\mathbf{f}_x^t\,\boldsymbol{\beta} + Z(x))^2\right. \\ &\quad \left. -2\mathbf{c_x^t}(\mathbf{F}\boldsymbol{\beta} + \mathbf{z})(\mathbf{f}_x^t\boldsymbol{\beta} + Z(x))\right] \\ &= (\mathbf{c_x^t F}\boldsymbol{\beta} - \mathbf{f}_x^t\boldsymbol{\beta})^2 + \mathbf{c_x^t}\sigma_z^2\mathbf{R c_x} + \sigma_z^2 - 2\mathbf{c_x^t}\sigma_z^2\mathbf{r}_x \\ &= \mathbf{c_x^t}\sigma_z^2\mathbf{R c_x} + \sigma_z^2 - 2\mathbf{c_x^t}\sigma_z^2\mathbf{r}_x \quad . \end{aligned} \tag{2.11}$$

The last equation follows if we impose the unbiasedness constraint $\mathbf{F}^t\,\mathbf{c_x} = \mathbf{f}_x$. This constraint follows from equating

$$E(\mathbf{c_x^t y}) = \mathbf{c_x^t F}\boldsymbol{\beta}$$

and

$$E(\hat{Y}(\mathbf{x})) = \mathbf{f}_x^t\boldsymbol{\beta}$$

for all β.

Introducing k Lagrange multipliers $\boldsymbol{\lambda}$ for the k equations $\mathbf{F}^t\mathbf{c_x} = \mathbf{f}_x$ and taking the derivative with respect to $\mathbf{c_x}$ in (2.11) yields

$$\sigma_z^2\mathbf{R c_x} - \sigma_z^2\mathbf{r}_x - \mathbf{F}\boldsymbol{\lambda} = 0 \quad .$$

Together with the unbiasedness constraint we have a system of two sets of equations in the two unknown vectors $\mathbf{c_x}$ and $\boldsymbol{\lambda}$:

$$\begin{aligned} \sigma_z^2\mathbf{R c_x} - \mathbf{F}\boldsymbol{\lambda} &= \sigma_z^2\mathbf{r}_x \\ \mathbf{F}^t\mathbf{c_x} &= \mathbf{f}_x \quad . \end{aligned}$$

We rewrite this system in matrix form :

$$\begin{pmatrix} 0 & \mathbf{F}^t \\ \mathbf{F} & \sigma_z^2\mathbf{R} \end{pmatrix} \begin{pmatrix} -\boldsymbol{\lambda} \\ \mathbf{c_x} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_x \\ \sigma_z^2\mathbf{r}_x \end{pmatrix} \quad . \tag{2.12}$$

The best linear unbiased predictor is then

$$\hat{y}(\mathbf{x}) = \mathbf{c_x^t y} = (\mathbf{f}_x^t, \mathbf{r}_x^t) \begin{pmatrix} 0 & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \mathbf{y} \end{pmatrix} \quad . \tag{2.13}$$

This can also be written as

$$\hat{y}(\mathbf{x}) = \mathbf{f}_x^t\hat{\boldsymbol{\beta}} + \mathbf{r}^t(\mathbf{x})\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}}), \tag{2.14}$$

where $\hat{\boldsymbol{\beta}} = (\mathbf{F}^t\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^t\mathbf{R}^{-1}\mathbf{y}$ is the generalized least squares estimator of $\boldsymbol{\beta}$. It turns out it is also the MLE, as will be derived later.

The MSE of the estimate can be derived by substituting (2.12) in (2.11):

$$\begin{aligned} \mathrm{MSE}[\hat{y}(\mathbf{x})] &= E[\mathbf{c_x^t y} - Y(\mathbf{x})]^2 \\ &= \sigma_z^2[1 + \mathbf{c_x^t R c_x} - 2\mathbf{c_x^t r}_x] \\ &= \sigma_z^2\left[1 - (\mathbf{f}_x^t, \mathbf{r}_x^t) \begin{pmatrix} 0 & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{f}_x \\ \mathbf{r}_x \end{pmatrix}\right]. \end{aligned} \tag{2.15}$$

Another way of looking at this is to consider $\mathbf{y}$ and $Y(x)$ together, assuming they are jointly normally distributed:

$$\begin{pmatrix} \mathbf{Y} \\ Y(x) \end{pmatrix} \sim \mathrm{MVN}\left(\begin{pmatrix} \mathbf{F} \\ \mathbf{f}_x^t \end{pmatrix}\boldsymbol{\beta}, \sigma_z^2 \begin{pmatrix} \mathbf{R} & \mathbf{r}_x \\ \mathbf{r}_x^t & 1 \end{pmatrix}\right).$$

Then, we can understand $\hat{Y}(x)$ as the conditional expectation of $\hat{Y}(x) = Y(x)|\mathbf{y}$. More precisely,

$$\hat{Y}(x) = Y(x)|\mathbf{y} \sim \mathrm{N}\left(\mu(x), \sigma^2(x)\right).$$

where

$$\mu(x) = \mathbf{f}_x^t\boldsymbol{\beta} + \mathbf{r}_x^t\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\boldsymbol{\beta}) \tag{2.16}$$

$$\sigma^2(x) = \sigma_z^2(1 - \mathbf{r}_x^t\mathbf{R}^{-1}\mathbf{r}_x). \tag{2.17}$$

Equations (2.16) and (2.14) are equivalent. Equation (2.15) differs from (2.17). because the estimation of $\boldsymbol{\beta}$ is ignored in the latter case.

Maximum Likelihood Estimation

We consider now the problem of finding maximum likelihood estimates of the unknown parameters: $\boldsymbol{\beta}$ in (2.1), σ_z in (2.2), $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_d)$ in (2.3), $\mathbf{p} = (p_1, p_2, \ldots, p_d)$ in (2.3), and in the case of random error $\frac{\sigma_z^2}{\sigma^2}$ in (2.9). Assuming the stochastic process is Gaussian, the log-likelihood up to an additive constant is

$$-\frac{1}{2}\left[n \ln \sigma_z^2 + \ln \det\mathbf{R} + (\mathbf{y} - \mathbf{F}\boldsymbol{\beta})^t\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\boldsymbol{\beta})/\sigma_z^2\right] \tag{2.18}$$

Given the correlation parameters $\boldsymbol{\theta}$ and $\mathbf{p}$, by differentiation with respect to $\boldsymbol{\beta}$. the MLE of $\boldsymbol{\beta}$ is the generalized least squares estimator

$$\hat{\boldsymbol{\beta}} = (\mathbf{F}^t\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^t\mathbf{R}^{-1}\mathbf{y} \quad . \tag{2.19}$$

The MLE of σ_z^2 is

$$\hat{\sigma}_z^2 = \frac{1}{n}(\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})^t\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}}). \tag{2.20}$$

If we substitute $\hat{\sigma}_z^2$ and $\hat{\beta}$ back into (2.18) we obtain

$$-\frac{1}{2}(n \ln \sigma_z^2 + \ln \det \mathbf{R}).$$

This function of the data and the correlation parameters θ and $\mathbf{p}$ has to be numerically maximized. Direct maximum likelihood estimation is very expensive to compute. Hence, an algorithm which introduces the parameters sequentially is often introduced to cut down on computing time. For example, see the algorithm described in Welch et al. (1992).

Bayesian Approach

Alternatively, instead of viewing $\mathbf{y}$ as a *realization* of a stochastic process, one can take a Bayesian point of view and predict $y(\mathbf{x})$ by the posterior mean

$$\hat{y}(\mathbf{x}) = E[Y(\mathbf{x})|\mathbf{y}_s]$$

where $\mathbf{y}_s$ denotes data at the design points. Currin et al. (1991) take this approach, representing prior uncertainty by a Gaussian stochastic process with fixed mean and variance, thus without a prior to represent the uncertainty in the mean, θ and $\mathbf{p}$. Because of this, they come to the same result as displayed in equations (2.16) and (2.17). Estimation of the parameters is then also performed by the Maximum Likelihood method. See also Morris, Mitchell, and Ylvisaker (1993) in the context of the Bayesian point of view.

2.2 Stochastic Methods of Spatial Prediction (Kriging)

The stochastic process model presented in the previous section has traditionally been used in geostatistics under the name of Kriging for the exploration of gold mines, oil fields, etc.. Since this stochastic process model is used frequently in this thesis, a brief overview of Kriging is given here. For more extensive reviews the reader is referred to Cressie (1993), Journel and Huijbregts (1978), and, for nonlinear Kriging, also Rivoirard (1994).

The word "Kriging" is synonymous with optimal spatial prediction. It has been termed after a South-African mining engineer with the name Krige, who first popularized stochastic methods for spatial prediction.

When the underlying stochastic process is Gaussian and a quadratic loss function is chosen, then an optimal predictor is given by $E(Y(\mathbf{x})|\mathbf{y}_s)$. Because of the Gaussian assumption, the predictor is a linear function of $\mathbf{x}$.

Usually the following additional model assumption is made:

$$Y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}) \; .$$

where $Z(.)$ is a random process with mean 0, and $\mu(.)$ is a parametric model specifying the mean structure.

Simple Kriging

The simplest Kriging models are ones where the mean structure $\mu(\mathbf{x})$ and the covariance structure $\mathrm{Cov}(Z(\mathbf{x}), Z(\mathbf{x}'))$ are assumed known. Furthermore, the predictor is assumed to be a linear function of the data.

The optimal predictor can then be derived as

$$\hat{y}(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{r}'\mathbf{R}^{-1}(\mathbf{z} - \mu).$$

Often the mean and the covariance structure are not known. The ordinary Kriging method therefore relaxes the assumption of full knowledge of mean and the covariance structure.

Ordinary Kriging

The mean $\mu(\mathbf{x})$ is unknown, but assumed constant. The random function Z is stationary. The predictor is a linear function of the data and is uniformly unbiased. i.e. $E(\hat{y}(\mathbf{x})) = \mu$.

This method no longer requires full knowledge of the mean; however, it only allows for stationary models. In the following method, a class of non-stationary models is introduced through a nonstationary mean structure.

Universal Kriging

The mean structure is given by

$$\mu(\mathbf{x}) = \sum_{i=0}^{k} \beta_j f_j(\mathbf{x}).$$

The random function Z is stationary. Furthermore, the predictor is linear in the data and uniformly unbiased.

The analysis of computer experiments uses the Universal Kriging approach. Unlike Kriging models in geostatistics, however, computer experiments are considered to be deterministic. This difference is reflected in the covariance structure. Another difference is that correlations for Kriging are usually estimated by variograms (e.g., Cressie, 1993) whereas computer experiments typically use maximum likelihood estimation.

Chapter 3

Understanding Key Features of Computer Codes via Graphical Analyses

3.1 Introduction

Computer models or codes are now frequently used in engineering design, and in many other areas of physical science. For instance, the main example discussed here concerns the engineering design of a solar collector. This code computes an increase in heat transfer effectiveness, y, resulting from an engineering innovation. The design is characterized by six factors (engineering parameters), $x_1, \ldots, x_6$. Further details will be given in Section 3. As is often the case, the code is expensive to compute and the engineers wanted to understand key features of the complex functional relationships embodied in their computer code. In particular, they were interested in possible nonlinearities and interactions.

Figure 3.1 shows scatter plots of the response against each x variable in turn for

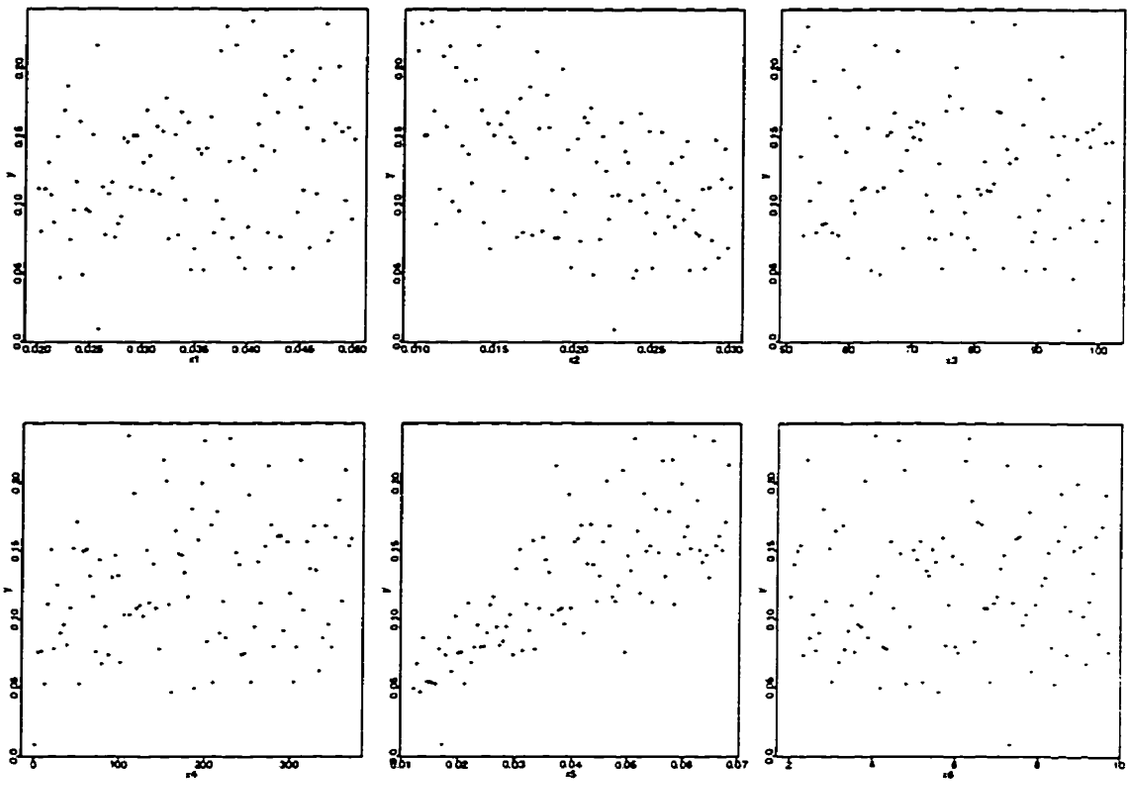


Figure 3.1: Scatter Plots of y versus x_i, i=1 ... 6.

data from an experiment on the solar collector code. They indicate some trend in the relationship between y and x_2 and x_5. However, the scatter plots do not show, for example, the strong relationship in x_4, because it is masked by the effects of the other covariates. This would not matter if the effects were all linear and additive, but, as we shall see in Section 3, the effect of x_4 is highly nonlinear. With nonlinear effects, we need to know the form of the model to be fitted, and simple plotting of the data does not suggest a class of nonlinear parametric models here. Moreover,

nonlinearities are common in computer experiments, because the inputs often cover wide ranges.

There is already some work on the design and analysis of computer experiments. See, for example, Currin, Mitchell, Morris, and Ylvisaker (1991), Sacks, Schiller, and Welch (1989), Sacks, Welch, Mitchell, and Wynn (1989), and Welch, Buck, Sacks, Wynn, Mitchell, and Morris (1992). The methods proposed in these references take into account the deterministic nature of a code like the solar collector computer model. Given the same inputs, it always reproduces the same output(s). Typically, the code will be expensive to run, e.g., it solves a large number of differential equations which may require several hours or more of computer time.

So far work on the design and analysis of computer experiments has focused on finding a good cheap-to-compute nonparametric surrogate (i.e., predictor) for the computer model. In the solar collector example, however, *explanation* rather than *prediction* is the overriding objective. The class of nonparametric predictors suggested in the above references and (2.14) is unsuitable for this task: They are computationally cheap approximations, but they are nonetheless mathematically complex.

In this chapter we propose to explore key features of computer codes such as nonlinearities and interactions by testing specific hypotheses about their functional form. To facilitate hypotheses generation, that is identifying key features, we introduce some new methodology for attaching standard errors to the nonparametric estimates of the effects. We then construct parametric models that embody the hypothesized key features. The parametric framework allows us to test key features, and thus to confirm them. As will be shown, the visualization of effects is fairly automatic.

An overview of the chapter is as follows. Section 2 first outlines the nonparamet-

ric method we use for analyzing data from a computer experiment. It has several advantages, but it is by no means the only method that might accomplish this task. Section 2 then explains how key features of the nonparametric model can be identified graphically and confirmed by building parametric models. Section 3 states two theorems, special cases of which explain how to estimate effects and their standard errors. Section 4 demonstrates these ideas using the solar collector code. Section 5 concludes with some discussion, including comments on the choice of experimental design and alternative modeling approaches.

3.2 Identifying Key Features of Computer Codes

Identifying key features of the relationship between input and output variables is easy if there is only one covariate. A simple scatter plot reveals the functional relationship, which for a computer model is exact since the relationship is deterministic. Then the data analyst often chooses to fit a parametric model to the data, where a class of (possibly nonlinear) models might be suggested by the scatter plot. This approach was used in a case study presented in Bates and Watts (1988, Section 3.13) for physical experimental data which contained random error. While the data from a computer experiment contain no random error, the objective here remains the same, i.e., to summarize the relationship between input and output variables in a concise way.

Scatter plots are not very useful for the identification of functional relationships where there is more than one covariate, however. The relationship between the response and each covariate can be masked by the relationships between the response and the other covariates (e.g., Montgomery and Peck, 1982, Section 4.2.5). To overcome the masking problem, a plot of a function involving only the covariate

of interest is needed. In other words, the effects of the other covariates need to be eliminated. Such plots will be considered shortly, after some preliminaries.

First, a brief overview of the nonparametric predictor used in this chapter is given because it plays a key role in the method proposed shortly. The data from a computer experiment consist of n vectors of covariate values (or inputs) denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_n$ for the d - dimensional covariates $x_1, \ldots, x_d$ as specified by a particular experimental design. The corresponding response values (for a particular output variable) are denoted $\mathbf{y} = (y_1, \ldots, y_n)^t$. Then, following the approach of, e.g., Welch et al. (1992), the response y is treated as a realization of a stochastic process:

$$Y(\mathbf{x}) = \beta + Z(\mathbf{x}), \tag{3.1}$$

where $E(Z(\mathbf{x})) = 0$ and $\mathrm{Cov}(Z(\mathbf{w}), Z(\mathbf{x})) = \sigma_z^2 R(\mathbf{w}, \mathbf{x})$ for two input vectors $\mathbf{w}$ and $\mathbf{x}$. The correlation function $R(\cdot, \cdot)$ can be tuned to the data. and is assumed here to have the form:

$$R(\mathbf{w}, \mathbf{x}) = \prod_{j=1}^{d} \exp(-\theta_j |w_j - x_j|^{p_j}), \tag{3.2}$$

where $\theta_j \geq 0$ and $0 < p_j \leq 2$. The p_j's can be interpreted as smoothness parameters—the response surface is smoother with respect to x_j as p_j increases—and the θ_j's indicate how local the estimate is. If the θ_j's are large, only data at points in the immediate vicinity of a given point are highly correlated with Y at that point and are thus influential in the prediction at that point. If the θ_j's are small, data at points further away are still highly correlated and still influence the prediction at that point. Correlation functions other than (3.2) could be chosen, for example Matérn (Yaglom, 1987, p.139). While Matérn's correlation function does

give more control over smoothness, it is also more expensive and not clear that it is a better choice in practice (see also the discussion in Sacks, Welch, Mitchell, and Wynn, 1989).

The best linear unbiased predictor of Y at an untried $\mathbf{x}$ can be shown to be (see (2.14) with $\mathbf{F} = \mathbf{1}$ and $\mathbf{f}_x = 1$):

$$\hat{Y}(\mathbf{x}) = \hat{\beta} + \mathbf{r}^t(\mathbf{x})\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\beta}), \tag{3.3}$$

where $\mathbf{r}(\mathbf{x})$ is the $n \times 1$ vector of the correlations between $Y(\mathbf{x})$ and $\mathbf{y}$, $\hat{\beta}$ is the generalized least squares estimator of β, $\mathbf{R}$ is the $n \times n$ correlation matrix with element i, j defined by $R(\mathbf{x}_i, \mathbf{x}_j)$ in (3.2) and $\mathbf{1}$ is an $n \times 1$ vector of 1's. Except for very large n this predictor is cheap to compute. The cost of one evaluation of the likelihood is of order n^3, but the evaluation of the predictor is only of order n. While this predictor has proven to be accurate for numerous applications, it does not reveal the relationship between y and $x_1, \ldots, x_d$ in a readily interpretable way. Consequently, this predictor is unsuitable for *explaining* the functional relationship between the covariates and the response.

Recall that in order to identify the functional relationship between a group of covariates and the response, the effect of these covariates needs to be isolated from the others. When we want to isolate the effect of a single covariate, the true *main* effect of the covariate can be defined in the following two ways:

1. *Integrating out the other factors.* The main effects are defined as:

$$\mu_i(x_i) = \frac{1}{V} \int Y(\mathbf{x}) \prod_{h \neq i} dx_h$$

(Sacks, Welch, Mitchell, and Wynn, 1989).

For simplicity we assume a hyper-rectangular integration region in $\mathbf{x}$. Estimates of $\mu_i(x_i)$ and their standard errors are discussed further in the next section.

2. *Keeping the other variables fixed.* For example, the other variables might be fixed at their respective midranges. Standard errors for the estimated effects using this method are available directly from $\text{MSE}(\hat{Y}_{\mathbf{x}})$ as given for example in (Sacks, Welch, Mitchell, and Wynn, 1989).

In both calculations, the unknown $y(\mathbf{x})$ needs to be replaced by $\hat{Y}(\mathbf{x})$ from Equation (3.3). The first approach may be preferred because it is analogous to analysis of variance in that all the other covariates are averaged out. Note also that integrating $\hat{Y}(\mathbf{x})$ is numerically easy to perform if the $\mathbf{x}$ region is cuboidal and if the correlations are in product from as in (3.2). In a similar fashion, the effect of two or more covariates can be investigated by integrating out all the other covariates or fixing the other covariates at specific values.

Main effects for each x_i and effects of, say, two covariates for each pair (x_i, x_j) can then be displayed graphically. By choosing a tentative model for each of the effect plots which displays some key feature (i.e., impacts the response), an overall model can be developed by adding up all the corresponding candidate models. The standard errors for the effects are useful here in that they may guide the choice of tentative models. They are further discussed in the next section.

If there are no interactions (and hence, additivity holds) the d-dimensional problem has been reduced to d one-dimensional problems. If large interactions are present, then the interacting covariates need to be considered jointly. Covariates might then be grouped so that covariates in two different groups do not interact. Provided that the groups contain no more than two variables, candidate models

may still be identified from contour plots of the response. For larger sized groups, such plots will generally not be helpful. In this case, when faced with many interactions, transforming the response may help in reducing the apparent complexity. Experience with a number of computer models, however, suggests the complexity of computer models tends to arise from additive nonlinearities rather than through interactions.

Subsequently, the key features summarized in the parametric model can be confirmed by fitting it using standard nonlinear regression techniques. Starting values for the parameter estimates can often be estimated from the effect plots.

3.3 Estimates for Effects and their Standard Errors

Suppose we want to plot the estimated effect of some of the x variables, denoted by $\mathbf{x}_{\text{effect}}$. The remaining x variables, denoted by $\mathbf{x}_{\text{out}}$, have to be integrated out of the predictor. The effect is

$$\mu(\mathbf{x}_{\text{effect}}) = \frac{1}{V}\int Y(\mathbf{x})d\mathbf{x}_{\text{out}}, \tag{3.4}$$

where V is the volume of the $\mathbf{x}_{\text{out}}$ region over which we integrate. For example, for the main effect of x_1 in the solar collector application with six explanatory variables, $\mathbf{x}_{\text{effect}} = x_1$, and the plotting coordinates for the estimated main effect of x_1, $\hat{\mu}(x_1)$, require an integration over $\mathbf{x}_{\text{out}} = (x_2, \ldots, x_6)^t$ for each value of x_1 plotted. The integral in (3.4) is easy to approximate if the x-space is cuboidal, and if the correlation function is a product of correlation functions for each x variable.

Numerically, we approximate (3.4) by a sum over a grid of m points $\mathbf{x}_{\text{out}}^{(1)}, \ldots, \mathbf{x}_{\text{out}}^{(m)}$

representing the $\mathbf{x}_{\text{out}}$ space. Thus (3.4) becomes

$$\mu(\mathbf{x}_{\text{effect}}) = \frac{1}{m}\sum_{i=1}^{m} Y(\mathbf{x}_{\text{effect}}, \mathbf{x}_{\text{out}}^{(i)}), \tag{3.5}$$

We now show how to estimate (3.5) and the standard error associated with the estimate. In fact, we prove a more general result for estimating linear combinations of Y_i's, which we show later can also be used for estimates of interactions and their standard errors. Both theorems are new work.

Theorem 1: The best linear unbiased predictor (BLUP) of $\sum b_i Y_i$ is $\sum b_i \hat{Y}(\mathbf{x}_i)$.

Theorem 2: The mean squared error of $\sum b_i \hat{Y}_i$ is

$$\text{MSE}\left(\sum_{i=1}^{m} b_i \hat{Y}_i\right) = \sigma_z^2 \left[\mathbf{b}^t \mathbf{R} \mathbf{b} - \begin{pmatrix} \bar{\mathbf{f}} \\ \bar{\mathbf{r}} \end{pmatrix}^t \begin{pmatrix} 0 & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} \bar{\mathbf{f}} \\ \bar{\mathbf{r}} \end{pmatrix}\right], \tag{3.6}$$

where $\mathbf{b} = (b_1, b_2, \ldots, b_m)$, $\bar{\mathbf{f}} = \sum_i b_i \mathbf{f}_{x_i}$, and $\bar{\mathbf{r}} = \sum_i b_i \mathbf{r}_{x_i}$. The standard error of $\sum b_i \hat{Y}_i$ is

$$\text{SE}\left(\sum_{i=1}^{m} b_i \hat{Y}_i\right) = \sqrt{\text{MSE}\left(\sum_{i=1}^{m} b_i \hat{Y}_i\right)}.$$

We prove both theorems in Appendix B.

All effects or linear combinations of effects can be written as $\sum b_i Y_i$ with suitable coefficients b_i, $i = 1 \ldots m$. Then the estimates of the effects are given by Theorem 1 and their standard errors by Theorem 2.

For (3.5), Theorem 1 with $b_i = 1/m$, $i = 1, \ldots, m$ states that the BLUP of the effects given by (3.5) is the corresponding sum of estimated function values. Moreover, Theorem 2 gives a pointwise standard error for the estimated effect. Appendix B contains further examples showing how Theorems 1 and 2 can be used and explains why (3.6) is easy to evaluate.

3.4 Application to a Solar Collector Code

In this section, the proposed method is applied to an expensive-to-compute computer model for the heat exchange effectiveness between the air and an unglazed transpired-plate solar collector with slot-like perforations (henceforth, referred to as holes). The use of equally spaced slot-like holes replaces the unrealistic assumption of infinitesimally small and infinitesimally close holes and thus, represents an engineering novelty in the design of unglazed solar collectors. Golneshan (1994) showed that the heat exchange effectiveness for these solar collectors is a function of six covariates, (1) inverse wind velocity, (2) dimensionless slot width, (3) Reynolds number, (4) admittance, (5) dimensionless plate thickness, and (6) the radiative Nusselt number, as defined by a system of differential equations. The computer code (Cao, 1993) solves the system of differential equations for given covariate values and requires around two hours of computing time on a workstation. The response considered here is the increase in heat exchange effectiveness attributed to the heat transfer in the holes from the hole sides and is expressed as a-percentage (0-100). For further details, see Cao (1993). For notational simplicity the six covariates listed above will be referred to as $x_1, x_2, \ldots, x_6$ and the response as y.

The mechanical engineers who had developed the solar collector code were interested specifically in explaining the impact of the six covariates (which are design factors) on the response, heat exchange effectiveness; ultimately, the explanation would help to identify better solar collector designs. Note that such understanding was not apparent from inspecting the system of differential equations. The engineers were also interested in developing a surrogate parametric model because empirical models of this type existed in the literature for solar collectors based on

older technologies; they had no preconceived idea of what form the model should take because the collectors with slot-like holes represented state-of-the-art technology. Hence, the need arose for performing an experiment on the solar collector code, i.e., a computer experiment.

The experimental design used for the computer design was one that filled the six dimensional cuboidal region, a so-called space filling design. Specifically, a Latin hypercube design (McKay, Beckman, and Conover, 1979) consisting of 100 points was chosen in which the minimum distance between points (i.e., the covariate vectors) in low-dimensional projections was maximized. The design was found using ACED (Algorithms for Constructing Experimental Designs) which was developed by Welch. All the two-dimensional projections of the Latin hypercube design can be seen in Figure 3.2 which shows that the design is indeed space-filling.

Scatter plots of the data (Figure 3.1) indicate a possible linear trend in x_2 and x_5. The remaining relationships, if any, are masked by the presence of the other covariates. In the following, the proposed method for identifying key features of the computer code will be applied.

The stochastic process predictor (3.3) with the correlation function (3.2) was fit for the response using the software GaSP (Gaussian Stochastic Processes), developed by Welch. GaSP also estimates the correlation parameters θ_j and p_j. $j = 1 \ldots d$, as well as σ_z^2 via the maximum likelihood approach.

The predictor appears to be reasonably accurate. Main effect and joint effects plots, generated by integrating out the other covariates, are as shown in Figure 3.3 for covariates x_1 through x_6 and Figure 3.4 for the pairs (x_2, x_5) and (x_4, x_5), respectively. By joint effect, we mean (3.5) where $\mathbf{x}_{\text{effect}}$ includes two variables and $\mathbf{x}_{\text{out}}$ all other variables.

The main effect for covariate x_6 is very flat, and all but two two-way interactions

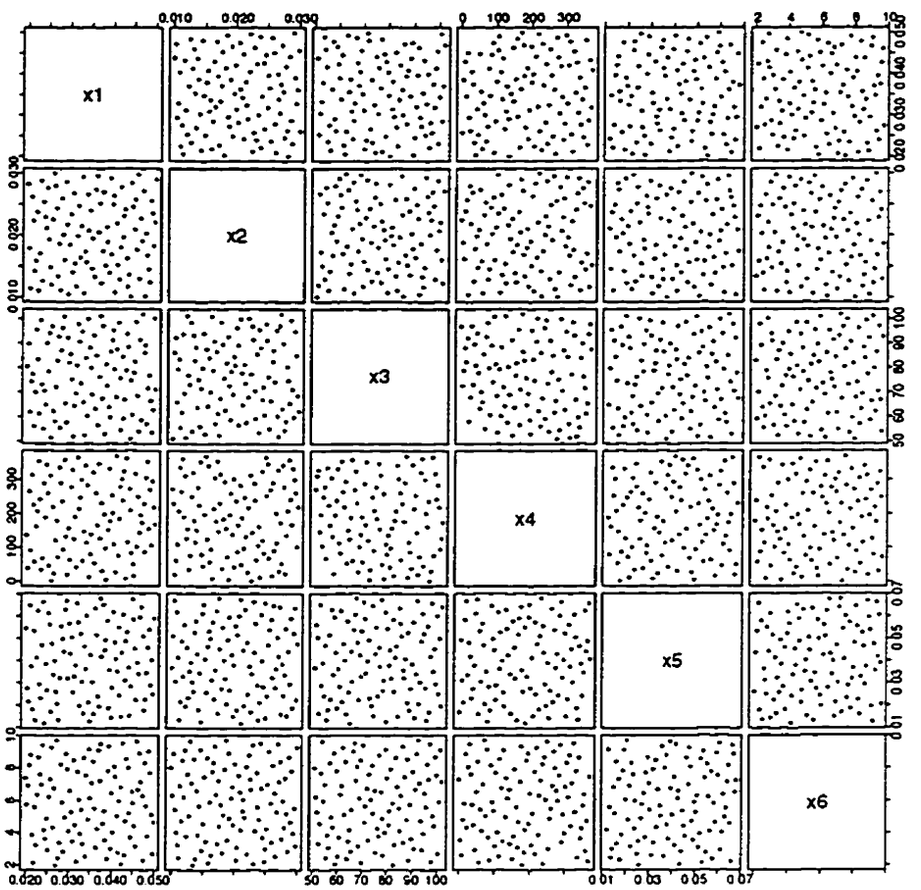


Figure 3.2: Two Dimensional Projections of the Latin Hypercube Design.

are close to zero everywhere. These effects were considered negligible by the engineers. The features displayed in the main effect plots suggest that the effects of x_1 and x_3 are approximately linear and the effects of x_2 and x_5 are approximately quadratic.

The main effect plot for x_4 is rather ragged. Although the plot gives a good indication of the apparently nonlinear x_4 effect, it is doubtful that the true x_4 relationship is that bumpy. One possible explanation is that the computer code may have some numerical convergence problems in certain regions of the $\mathbf{x}$ space.

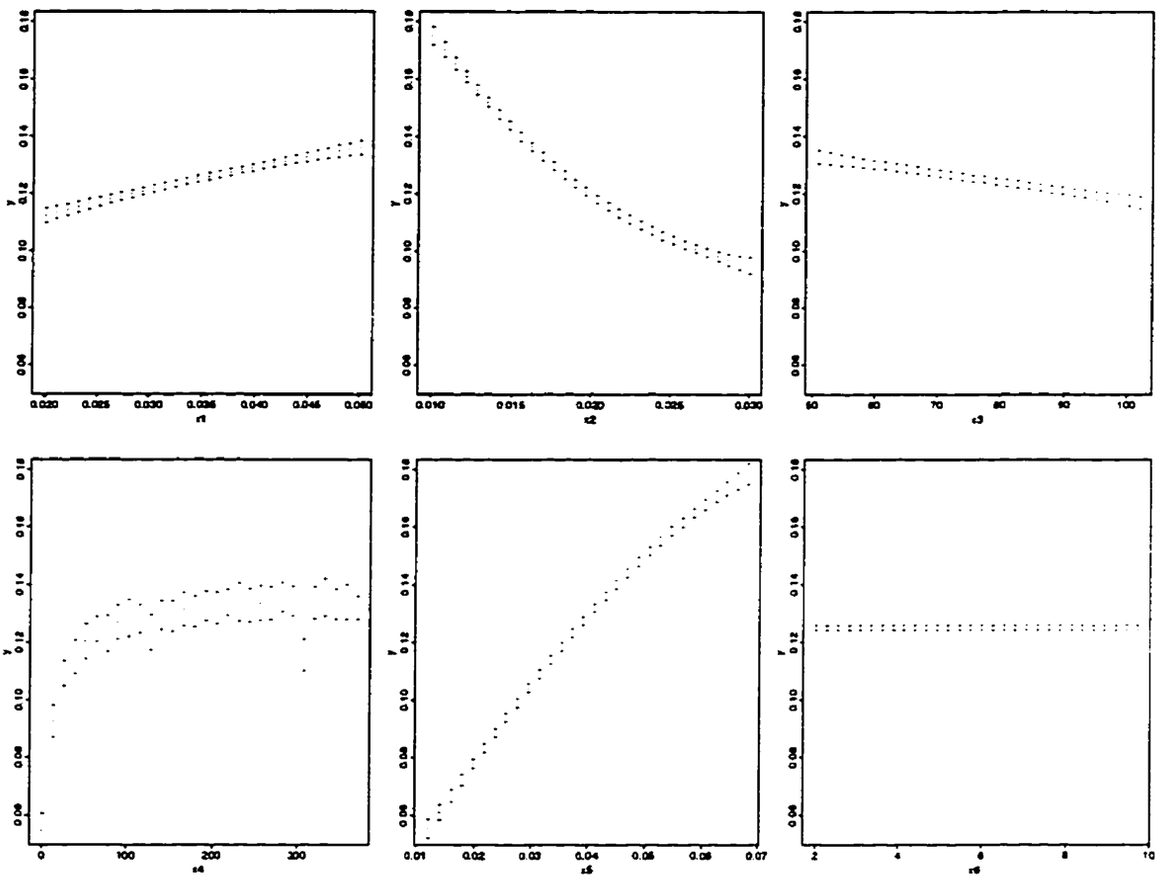


Figure 3.3: Main effect plots. The Middle Line is the Estimated Effect, the Upper and Lower Lines are Approximate 95% Pointwise Confidence Limits Based on the Standard Error Given by Theorem 2.

This possible erratic behavior may then be erroneously attributed to x_4 which clearly has the most nonlinear or complex impact on the response. Engineering knowledge suggests that the increase in heat efficiency is a monotone increasing function of the admittance rate of the plate x_4. The head engineer commented: "The slight blip in the curve is almost certainly due to some numerical problem" (Hollands, 1995, personal communication). Therefore, we do not model the little down peak at $x_4 = 300$.

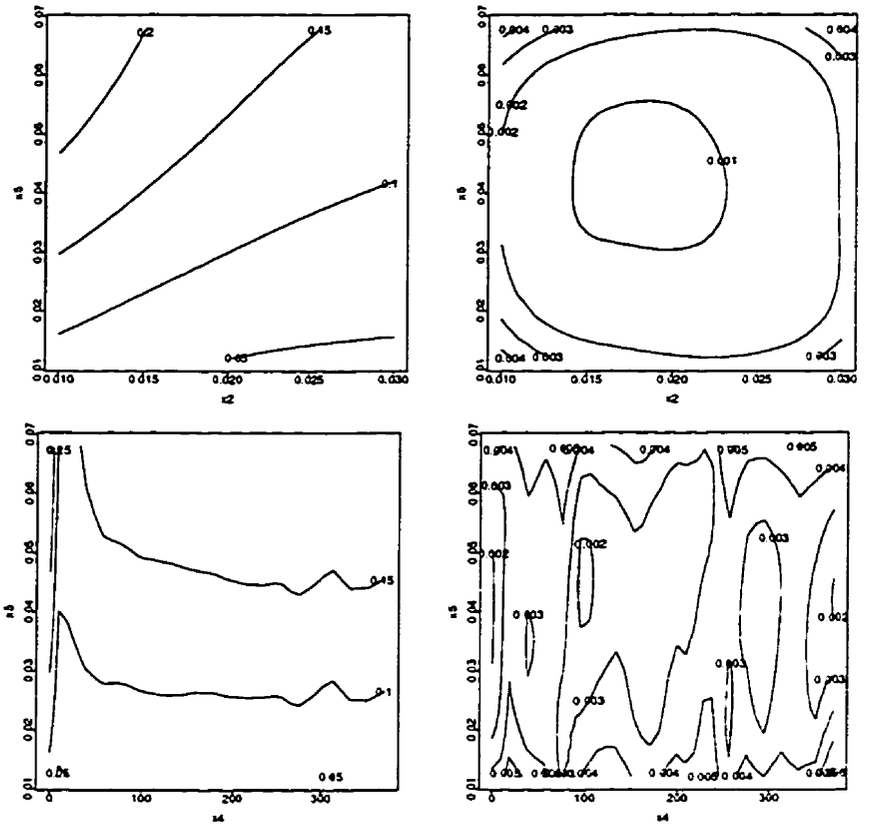


Figure 3.4: Joint Effects (left) and Standard Error Plots (right) for (x_2, x_5) and (x_4, x_5). The Standard Error is Given by Theorem 2.

Plots of the main effects using the method of fixing the other variables at their respective midranges rather than averaging them out, result in very similar graphs. For example, Figure 3.5 shows the Method 2 main effect plot for x_4.

The nonlinear shape of the x_4 main effect plot which appears to approach an asymptote can be captured by a Michaelis-Menten model (Bates and Watts, 1988, p. 329); the Michaelis-Menten model has long been used to model the behavior of a limiting chemical reaction which rises at a decreasing rate to an asymptote. It also arises in the context of a reciprocal link function in generalized linear models.

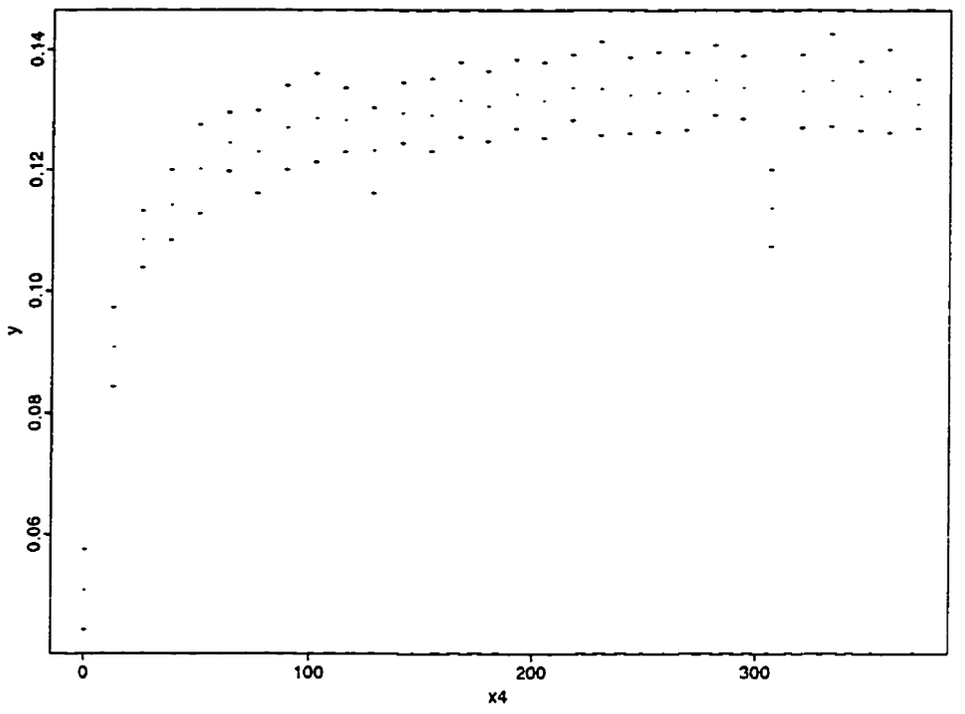


Figure 3.5: Predicted $\hat{Y}(\mathbf{x})$ versus x_4, Keeping All Other x Variables Fixed at Their Midranges (Method 2). The Middle Line is the Estimated Effect. the Upper and Lower Lines are Approximate 95% Pointwise Confidence Limits.

where an inverse linear response function is assumed (McCullagh and Nelder. 1989. p. 291). We reparameterize the Michaelis-Menten model to make the nonlinear fitting numerically easier:

$$y = \frac{1}{\beta_0 + \beta_1/x_4} = \frac{\gamma_0\, x_4}{x_4 + \gamma_1}.$$

where $\beta_0 = 1/\gamma_0$ and $\beta_1 = \gamma_1/\gamma_0$.

Both x_2 and x_5 appear to be quadratic. For the joint effect, we notice that when x_5 increases, the response rises more rapidly when x_2 is low than when x_2 is high. This points to the presence of interaction. Nonetheless, the interaction does

not seem very complex and we try the simplest form for the interaction term, x_2x_5.

For the joint effect between x_4 and x_5, we notice that for high values of x_5 the sudden rise due to x_4 seems to be more pronounced than for low values of x_5. Hence we hypothesize the existence of an interaction of the form $h_4(x_4)\,x_5$ where $h_4(.)$ is the main effect model for x_4, i.e. the Michaelis-Menten model. For added flexibility, we allow different parameters for the Michaelis-Menten term in $h_4(.)$ and in the corresponding main effect term.

To confirm the key features found, we then fit the overall model consisting of linear effects in x_1, x_2, x_3, and x_5, a quadratic effect in x_2 and x_5, the Michaelis-Menten model for x_4, the bilinear term for (x_2, x_5), and the interaction term between the main effect model for x_4 and x_5 using standard nonlinear regression software which gave

$$\hat{y} = 0.09 + 0.86x_1 - 7.60x_2 - 0.00031x_3 + 135.97x_2^2 + 2.88x_5 - 21.02x_5^2 + 0.0025\frac{x_4}{x_4 - 9.37} - 47.49x_2x_5 + 2.30\frac{x_4x_5}{x_4 + 16.30}.$$

All of the parameters were significant at the 0.0001 level, except for the multiplicative parameter for the main effect for x_4 (0.0025), which was marginally significant at the 0.05 level. Also, adding x_6 reveals that x_6 is not significant at the .10 level. Further, when replacing the interaction model $h_4(x_4)x_5$ with the bilinear term x_4x_5 the latter is not significant. Although the data contain no random error so that significance testing has no theoretical grounds here, the results of the significance tests do indicate the importance of the various effects relative to the ability of the overall model to fit the data. Alternatively taking the Bayesian point of view, one could calculate posterior model probabilities. Note that the model contains only twelve parameters but fits the 100 data points quite well as indicated by the cor-

responding cross validation plot given in Figure 3.6. The fact that the parametric nonlinear model does not fit the data quite as well as the nonparametric model is not surprising, since the parametric model is much simpler. The cross validated root MSE is defined as

$$\text{Cross Validated RootMSE} = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_{-i}(\mathbf{x}_i))^2}{n}}$$

where $\hat{y}_{-i}(\mathbf{x}_i)$ is the cross validated prediction value at $\mathbf{x}_i$ based on all but the i^{th} observation. The better the fit is, the smaller is the cross validated root MSE. Here they are .0059 for the nonparametric and .0071 for the parametric model.

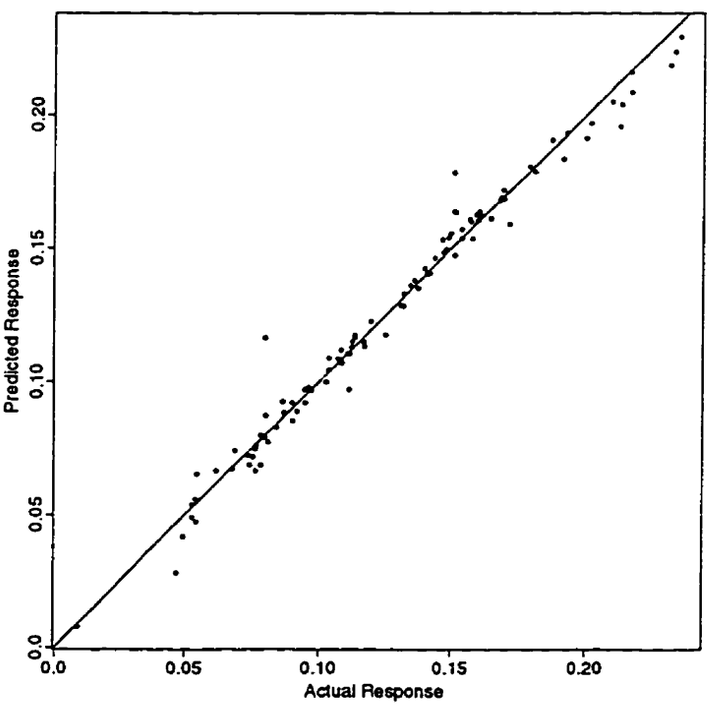


Figure 3.6: Cross Validation Predictions from the Parametric Nonlinear Model. The Line Predicted Response = Actual Response is Shown. The Cross Validation Root MSE is .0071 .

3.5 Discussion

The examples presented in nonlinear regression books typically deal with only a single covariate x, where the functional relationship between x and the response y is unknown. On the other hand, the method proposed here can be applied to an arbitrarily large number of covariates.

Throughout this chapter, we have used model (3.1) for the initial nonparametric analysis. Other nonparametric methods, like Generalized Additive Models, could be used. However, the model we use has three main theoretical advantages: first, the model is truthful to the deterministic nature of the data, second, error bounds for the effects are available, and third, interactions do not need to be modeled explicitly.

For a comparison in practice, we fit a Generalized Additive Model (GAM) to the data (see Figure 3.7). We choose Generalized Additive Models for its popularity and because the algorithm is readily available in Splus. The cross validation plot for GAM in Figure 3.8 shows a slight bias at the upper and lower range of the response. The cross validated root MSE is .0133, more than twice as large as the one for the stochastic model (Figure 3.9) and almost twice as large as the one for the parametric model (Figure 3.6). The effects for the GAM Model are the same but they are less obvious. Due to the smoothing the sudden rise for low values of x_4 is not as clear. At present GAM software does not support nonparametric interactions. Hastie and Tibshirani (1990, section 9.5) suggest among other things examining the residuals for interaction. Due to the lack of error bounds it is more difficult to assess, for example, the effect of x_6.

Breiman (1991) criticized algorithms for producing "only one picture" of the functional relationship, thus ignoring the many other "pictures" which are almost

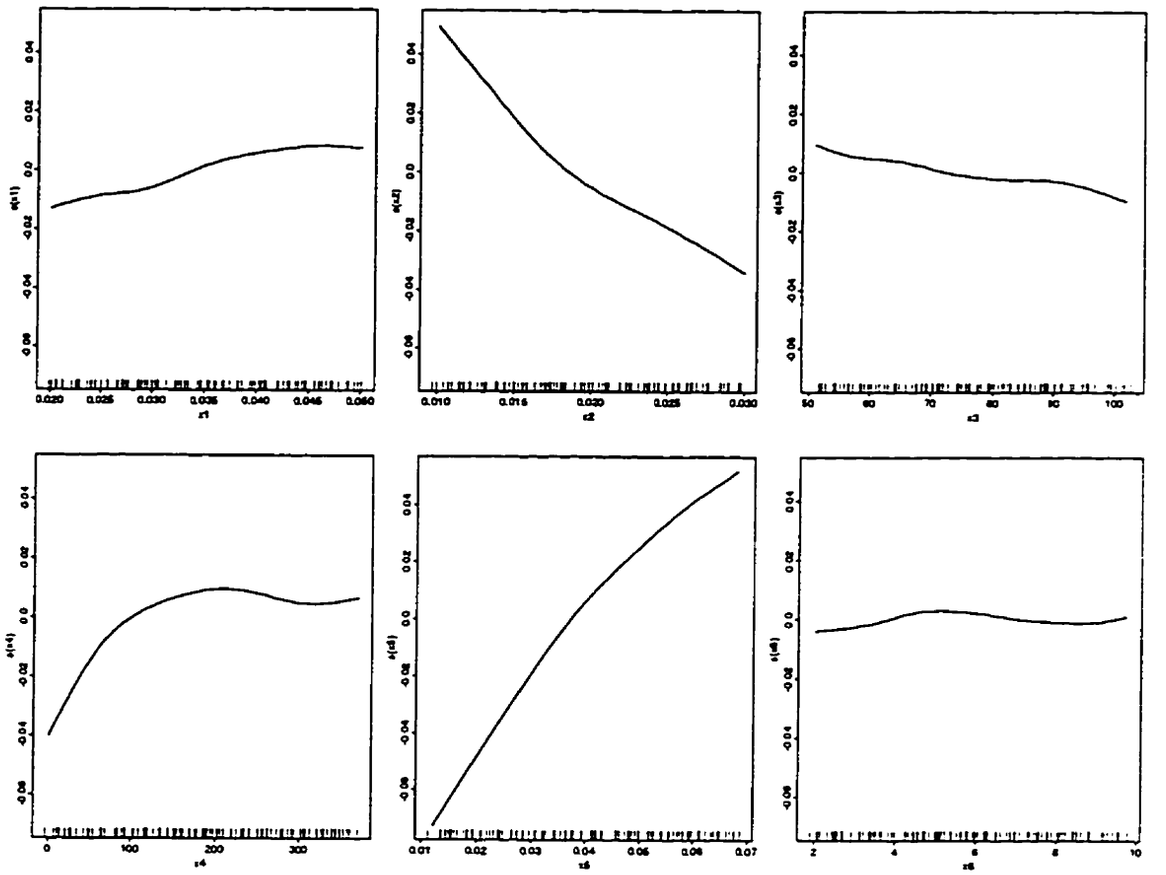


Figure 3.7: Main Effect Plots from the Generalized Additive Models (GAM) Approach. Except for Scaling of the Vertical Axes, the Plots Represent the Default Setting in Splus. The Rug at the Bottom Indicates Frequencies.

as good. The error bounds given for the effects can serve here as an assessment of the variability of the effect fit.

There are certainly other ways to identify key aspects of input-output relationships. For example, clever residual analyses in the hand of a skilled data analyst may well lead to the same results. For the solar collector experiment, an added variable (partial regression) plot for x_4 based on a linear regression model for the remaining covariates shows the effect of x_4 is nonlinear, albeit with considerable

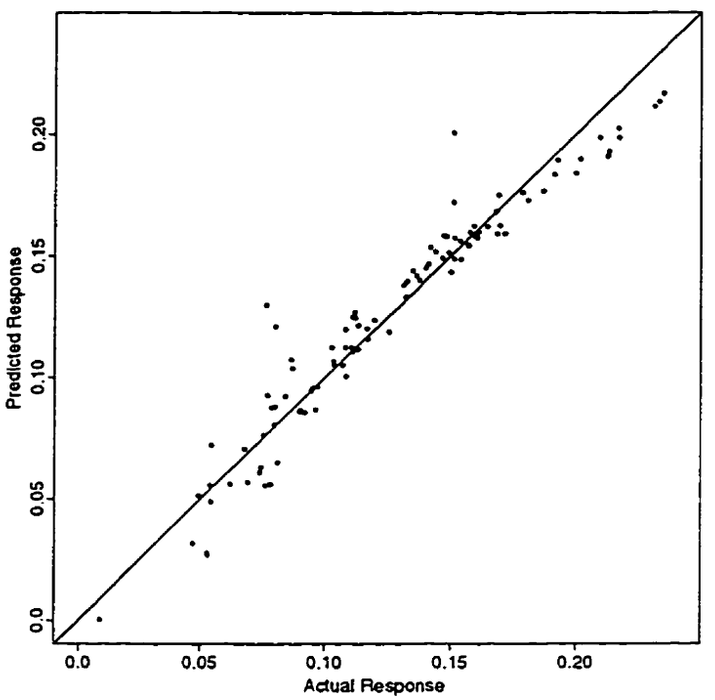


Figure 3.8: Cross Validation Predictions from the GAM Model. The Line Predicted Response = Actual Response is Shown. The Cross Validation Root MSE is .0133 .

scatter as displayed in Figure 3.10. This success is not surprising since the assumption of a linear model for the remaining variables turns out to be a good approximation. If the true model had contained several strong nonlinearities, then added variable plots on their own would not have sufficed. It might also be possible to find the interactions with residual analysis, though with considerable difficulty.

Elaborate residual analyses are often not done for three reasons: (1) They are hard to do, especially when the "true" model contains more than one nonlinear effect. (2) Data analysts, especially inexperienced ones, may not always know about them. (3) They can take a lot of time to perform. The method presented here is easy and fairly automatic for detecting nonlinear effects and interactions.

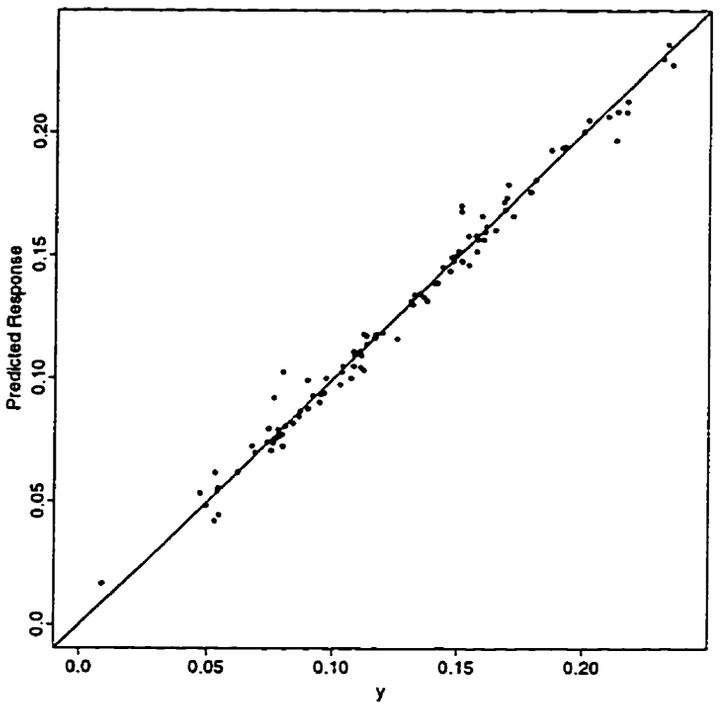


Figure 3.9: Cross Validation Predictions from the Nonparametric Stochastic Processes Model. The Line Predicted Response = Actual Response is Shown. The Cross Validation Root MSE is .0059 .

It is not a panacea for all functional relationships, however. If the relationship cannot be transformed to near additivity with few or no interaction effects, then identification of key features with several covariates will still be a challenge. For most of these cases, it is doubtful whether alternate methods will work either.

The effect plots play a key role in the proposed method and their resolution depends on the experimental design used. The Latin hypercube design is a desirable choice because the design points fill the experimental region well and produce high-resolution plots.

Originally, a 4^{6-2} fractional factorial design was considered for the solar collector computer experiment. While the choice of a fractional factorial or even full factorial

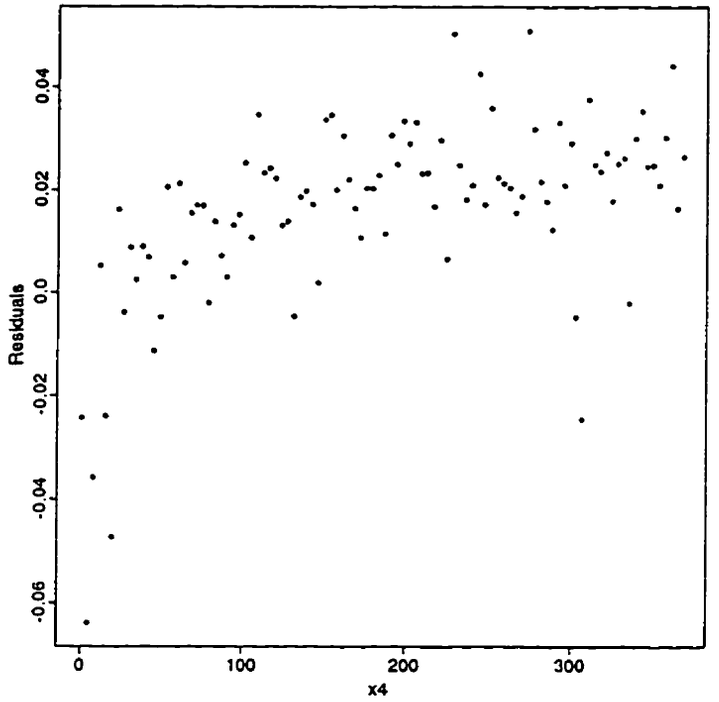


Figure 3.10: Added variable plot for x_4.

design would lead to estimates that are uncorrelated, there would have been several drawbacks, however. First, if only a few covariates (factors) had an impact, the design effectively collapses into a design in the active factors with replications. But, replications in a computer experiment are non-informative because of the deterministic nature of the computer code and therefore would have been a waste of resources. Second, it could have been easy to miss an unknown effect by only experimenting at a few different points for each factor. For example, the exact nature of the nonlinear x_4 effect would have been difficult to identify with only four levels; in fact, the dramatic nonlinear behavior of x_4 surprised the engineers. Analogous arguments apply for interactions. Third, the decision of where to place the levels becomes much more crucial for the factorial design; lower dimensional

projections of Latin hypercube design typically consist of n distinct and spread-out points so that their exact position is less important. Finally, a 4^{6-2} fractional factorial design would have required 256 runs. Contrast this with the 100-run Latin hypercube design that was used; even fewer runs might have been sufficient.

Computer experiments typically use such space filling designs so the proposed method is particularly suited to computer experiments. While physical experiments typically collect much less data than computer experiments, in principle the proposed method can be applied to physical experiments by adding a random error term to the model.

Chapter 4

A Data Analytic Approach to Bayesian Global Optimization

4.1 Introduction

Global optimization, that is the search for a global extremum, is a problem frequently encountered. Sometimes it is extremely costly to evaluate a function for an engineering design. For example, Frank (Davis, 1996) says about experiences at Boeing:

> "Designing helicopter blades to achieve low vibration is an extreme example of a problem where it is prohibitively expensive to compute responses for large numbers of design alternatives."

For such applications one is interested in minimizing the total number of function evaluations needed to find the global extremum.

When function evaluations are extremely expensive, it appears sensible to examine previous function evaluations, that is already sampled points, very carefully.

It is of particular interest to discover potential optimization problems before large amounts of sampling resources are spent.

In this chapter we introduce a set of diagnostic plots which early on assess the likely success of the global optimization method. If a problem is diagnosed, it is often possible to overcome it fully or partially by optimizing a suitable transformation of the response rather than the untransformed response.

The method proposed in this chapter deals with the unconstrained global optimization problem, minimize $f(\mathbf{x})$ where $\mathbf{x} = (x_1, \ldots, x_d)$. This includes the class of problems with simple constraints like $a_i \leq x_i \leq b_i$, $i = 1, \ldots, d$. since these problems can be transformed to unconstrained global optimization problems. Throughout we assume without loss of generality that the extremum of interest is a minimum.

The outline of this chapter is as follows. In Section 2 we review briefly the Bayesian global optimization approach and introduce a more flexible stochastic model in that framework. Also, a theorem concerning convergence of Bayesian global optimization is given. Section 3 describes the diagnostic plots. We show how they are used to assess and improve the model fit and hence the effectiveness of the global optimization method. Section 4 shows by means of several examples from the optimization literature that this approach is very efficient in terms of the number of function evaluations required. Section 5 concludes with some discussion.

4.2 Expected Improvement Algorithm

This algorithm is based on the idea that any future sampled point constitutes a potential improvement over the minimal sampled value up to the present stage. Uncertainty about the function value at a point to be sampled is dealt with by cal-

culating the expected improvement, based on some statistical model. The *expected improvement* criterion is equivalent, we show later, to one-step-ahead optimality in Bayesian Global Optimization.

The expected improvement algorithm proceeds in five steps:

1. Choose a small initial set of sampled points spread over the entire x space. Evaluate the true function at these points.

2. Model the true function using all previous function evaluations.

3. Search over $\mathbf{x}$ for the maximum expected improvement in f. The location of the maximum is the next sampled point.

4. Compute a stopping criterion based on the maximum expected improvement. If the criterion is met stop.

5. Evaluate the true function at the new sampled point. Go to Step 2.

Note that after each sampling step the predictor is updated (Step 2), and the expected improvement as a function of $\mathbf{x}$ is redefined in Step 3.

For Step 1, Latin hypercube sampling schemes (McKay et al., 1979) are particularly useful, because they have a space filling property, i.e. they uniformly cover the x domain to explore the function globally. The number of points sampled at this initial stage is somewhat arbitrary. We choose about 10 points per active variable because one needs at least that many points to obtain a reasonably good fit for moderately complex functions (Welch, personal communication).

For the modeling approach in Step 2 we use a stochastic process with a more flexible correlation structure than has been previously employed in the Bayesian global optimization literature. This is discussed further in Section 2.1.

The expected improvement criterion in Step 3 is based on the idea that any additional function evaluation constitutes a potential reduction of the minimal function evaluation found so far. This is discussed further in Section 2.2.

For Step 4, we propose to stop when the maximum of the expected improvement is smaller than a tolerance value; smaller in absolute value or relative to the current minimal function value. Step 5 consists of evaluating the next sampled point.

4.2.1 Modeling Approach

Suppose that, after an initial experimental design (set of sampled points) or at some stage of the algorithm, we have n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ at which the function f has been evaluated. Each vector $\mathbf{x}$ is d-dimensional for the d covariates (or inputs) $x_1, \ldots, x_d$. The corresponding response values (or outputs) are denoted $\mathbf{y} = (y_1, \ldots, y_n)^t$. Then, following the approach of Chapter 2 or, e.g., Welch et al. (1992), the response is treated as a random function or a realization of a stochastic process:

$$Y(\mathbf{x}) = \beta + Z(\mathbf{x}), \tag{4.1}$$

where $E(Z(\mathbf{x})) = 0$ and $\mathrm{Cov}(Z(\mathbf{w}), Z(\mathbf{x})) = \sigma^2 R(\mathbf{w}, \mathbf{x})$ for two inputs $\mathbf{w}$ and $\mathbf{x}$. The correlation function $R(\cdot, \cdot)$ can be tuned to the data. Here it is assumed to have the form:

$$R(\mathbf{w}, \mathbf{x}) = \prod_{j=1}^{d} \exp(-\theta_j |w_j - x_j|^{p_j}), \tag{4.2}$$

where $\theta_j \geq 0$ and $0 < p_j \leq 2$. The p_j's can be interpreted as parameters which indicate the smoothness of the response surface (smoother as the p's increase) and

the θ's indicate how local the predictor is (more local as the θ's increase).

The best linear unbiased predictor of y at an untried $\mathbf{x}$ can be shown to be (see (2.14) with $\mathbf{F} = 1$ and $\mathbf{f}_x = 1$):

$$\hat{y}(\mathbf{x}) = \hat{\beta} + \mathbf{r}^t(\mathbf{x})\mathbf{R}^{-1}(\mathbf{y} - 1\hat{\beta}), \tag{4.3}$$

where $\mathbf{r}(\mathbf{x})$ is the $n \times 1$ vector of correlations $R(\mathbf{x}, \mathbf{x}_i)$ in (4.2) for $i = 1, \ldots, n$ between Z at $\mathbf{x}$ and each of the n sampled points, $\mathbf{R}$ is a $n \times n$ correlation matrix with element (i, j) defined by $R(\mathbf{x}_i, \mathbf{x}_j)$ in (4.2), $\hat{\beta} = (1^t\mathbf{R}^{-1}1)^{-1}1^t\mathbf{y}$ is the generalized least squares estimator of β, and $\mathbf{1}$ is a vector of 1's.

The mean squared error (MSE) of the predictor can be derived as (see (2.15) with $\mathbf{F} = 1$ and $\mathbf{f}_x = 1$):

$$\mathrm{MSE}[\hat{y}(\mathbf{x})] \equiv s^2(\mathbf{x}) = \sigma^2 \left[1 - (1 \quad \mathbf{r}_x^t) \begin{pmatrix} 0 & 1^t \\ 1 & \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ \mathbf{r}_x \end{pmatrix} \right]. \tag{4.4}$$

The predictor based on the correlation function (4.2) in (4.3) has proven to be accurate for numerous applications, see e.g. Currin et al. (1991), Sacks et al. (1989a), Sacks et al. (1989b), Welch et al. (1992). Mockus (1989) used a Wiener field instead.

In practice, σ^2 defined after (4.1) and the correlation parameters $\theta_1, \ldots, \theta_d$ and $p_1, \ldots, p_d$ in (4.2) have to be tuned to the data. We use maximum likelihood estimation; see, for example, Welch et al. (1992) for details.

4.2.2 Expected Improvement

We will now derive the expected improvement criterion.

If the function is sampled at $\mathbf{x}$ to determine $y = f(\mathbf{x})$ then the improvement I over f^n_{min}, the minimal sampled function value after n evaluations, is defined as

$$I = \begin{cases} f^n_{min} - y & \text{if } y < f^n_{min} \\ 0 & \text{otherwise} \end{cases} = \max(0, f^n_{min} - y).$$

The expected improvement is given as

$$E(I) = \int_{-\infty}^{f^n_{min}} (f^n_{min} - y)\phi(y)dy, \tag{4.5}$$

where $\phi()$ is the probability density function representing uncertainty about y.

Mockus (1989) proposed a generalization by specifying a loss function on the sequential n-step optimization strategy S_n :

$$L(S_n, f) = f^n_{min} - \min_x f(x),$$

i.e., loss is defined as the difference between the global minimum and the best function value found after n steps. The risk, or the average loss is then given as

$$E\left(L(S_n, f)\right) = E(f^n_{min}) - E(\min_x f(x)). \tag{4.6}$$

An optimal strategy is defined as one that minimizes the risk (4.6). Computing an optimal strategy turns out to be computationally infeasible for even a moderate number of points n. The standard approach then is to relax the n-step optimality to one-step optimality. The criterion for one-step optimality is equivalent to (4.5).

To predict $Y(\mathbf{x})$ at an untried $\mathbf{x}$, we have $\hat{y}(\mathbf{x})$ from (4.3) with a mean squared error given by (4.4). For notational simplicity, we omit the dependence on $\mathbf{x}$. and denote $\hat{y}(\mathbf{x})$ by $\hat{y}$ and the mean squared error by s^2. If we further assume that

the random function $Y(\mathbf{x})$ is Gaussian, then $\hat{y}$ is also normal. Thus, we represent uncertainty about the true y by saying it is $N(\hat{y}, s^2)$. The expected improvement in (4.5) can be expressed as

$$E(I) = \begin{cases} (f_{min}^n - \hat{y})\Phi(\frac{f_{min}^n - \hat{y}}{s}) + s\phi(\frac{f_{min}^n - \hat{y}}{s}) & \text{if } s > 0 \\ 0 & \text{if } s = 0 \end{cases} \tag{4.7}$$

where $\phi()$ and $\Phi()$ denote the probability density function and the cumulative distribution function of the standard normal distribution. The first term in (4.7) is the predicted difference between the current minimum and y at $\mathbf{x}$, penalized by the probability of improvement. The second term is large when $\hat{y}(\mathbf{x})$ is close to f_{min}^n and s is large, i.e., when there is much uncertainty about whether $y(\mathbf{x})$ will beat f_{min}^n. Thus, the expected improvement will tend to be large at a point with predicted value smaller than f_{min}^n and/or where there is much uncertainty associated with the prediction.

A practical problem, though, is finding the global maximum of the expected improvement criterion over a continuous region. Expected improvement is zero at sampled points. As distance from all sampled points increases, so does s, one of the factors leading to large expected improvement. Random starting points are chosen such that in each coordinate the random point is halfway between two adjacent design points. Since the original design was space filling, it is ensured that the entire x-space is covered with local optimization tries. This does not guarantee to find the global maximum, of course. Mockus (1994) states in this context "[...] there is no need for exact minimization of the risk function", because we only determine the point of the next observation.

The following theorem holds for the expected improvement algorithm when the number of possible sampling points is finite:

Theorem 3 : Suppose we use the Gaussian model (4.1) and the covariance function (4.2) is such that the mean square error of prediction in (4.4) is positive for any unsampled point $\mathbf{x}$. Further, suppose the number of possible sampling points is finite. Then the expected improvement algorithm will visit all the sampling points and hence will always find the global minimum.

Proof: Given in Appendix C.

4.3 Diagnostics

The success of the Bayesian minimization algorithm depends on having a valid model. The better the model the more likely the algorithm will terminate quickly and with an accurate tolerance on the minimum. For this reason one would like to assess the performance of the modeling approach as soon as possible. that is after the initial function evaluations. When the model does not fit well it is often possible to improve the fit through appropriate transformations of the response. For this purpose we propose four diagnostic plots to be used after the initial function evaluations have been obtained. All of them are based on the concept of cross validation.

Cross validation is a statistical technique often used for assessing a model's predictive capability, when it is not convenient to test the model at further sampled points. It consists of setting aside and predicting a small portion of the data from a model based on the remaining larger portion of data. Most commonly only one point at a time is set aside, and cross validation is performed once for each point left out. In this chapter we always use leave-one-out cross validation.

We remove case i from (4.3) and (4.4) to obtain $\hat{y}_{-i}(\mathbf{x}_i)$ and $s_{-i}(\mathbf{x}_i)$. The notation emphasizes that case i is removed when predicting at $\mathbf{x}_i$. Cross-validated

standardized prediction errors (residuals), for example, can be written as

$$e_i = \frac{y_i - \hat{y}_{-i}(\mathbf{x}_i)}{s_{-i}(\mathbf{x}_i)}. \tag{4.8}$$

We propose the following four diagnostic plots:

1. A plot of the cross validation predictions versus the true y's, i.e. $\hat{y}_{-i}(\mathbf{x}_i)$ versus y_i, to indicate prediction accuracy.

2. A plot of the cross validated standardized errors versus the cross validated predictions, i.e., e_i in (4.8) versus $\hat{y}_{-i}(\mathbf{x}_i)$. This plot assesses whether the estimated uncertainty in prediction is realistic. The standardized errors should not lie far outside about $[-2, 2]$ or, if many points are plotted, $[-3.3]$. We are particularly concerned that estimated prediction accuracy is realistic for smaller predicted values, $\hat{y}$, as they are of most interest in minimization.

3. A quantile-quantile (Q-Q) plot of the ordered cross validated standardized errors versus quantiles from the standard normal distribution. If the normal approximation in deriving (4.7) is valid, we should see a straight line through the origin with slope 1.

4. A plot of the cross validated expected improvements versus the true function values, i.e., $E(I)$ evaluated at $\mathbf{x}_i$ based on $\hat{y}_{-i}(\mathbf{x}_i)$ and $s_{-i}(\mathbf{x}_i)$ versus y_i. Thus, we pretend that $\mathbf{x}_i$ was just introduced and compare the expected improvement with the function value actually achieved. If the expected improvement criterion is to find further points with good improvement, the lowest y's to date should be associated with the highest expected improvements.

If the plots indicate a poor fit, a transformation of the data can often improve

the fit. This is possible because the transformed data may more closely resemble a realization of a Gaussian stochastic process.

It is often useful to visualize the estimated function surface, as that may give some idea of the number of local minima or it might be possible to rule out certain regions of the x-space as a potential location for the global minimum with a high degree of confidence. In more than two dimensions visualization of the function surface is not straightforward. Instead, we estimate and plot main and joint effects. i.e., the response as a function of only one or two variables at a time. The main effect of x_i is obtained by averaging out from the predictor $\hat{y}(\mathbf{x})$ all x variables except x_i. Similarly, joint effects of two variables are obtained by averaging out all but two variables of interest (see e.g. Welch et al., 1992 or Section 3.3).

4.4 Examples

Our methodology is aimed at optimizing functions that are very expensive to compute, for example finite-element codes. It is convenient, however, to take example functions from the optimization literature. They demonstrate many qualitative features of real functions. They are often highly nonlinear and have several local optima. Moreover, using these well-known test-examples facilitates comparison with previous methods.

4.4.1 Branin Function (Br)

The Branin function (Törn and Žilinskas 1989) is

$$f(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10. \tag{4.9}$$

The x ranges are $-5 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 15$. The function has three global minima.

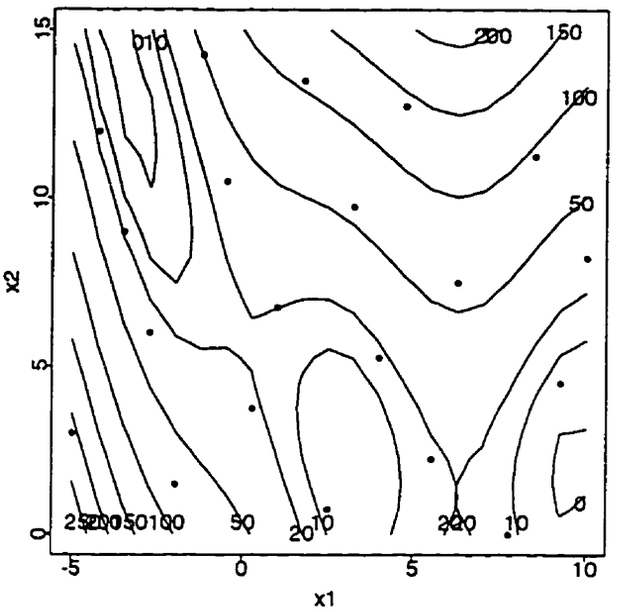
Initially, we sample the function at 21 points generated by a Latin Hypercube experimental design (Welch, work in progress). The choice of 21 is motivated by the rule of thumb "10 times the number of active variables". Choosing 21 points rather than 20 conveniently spaces points at 5% of the range.

Since the Branin function has only two x variables, in addition to looking at the proposed diagnostics we are able to visualize the function. Contour plots of the estimated function from (4.3) along with the 21 initial points can be seen in Figure 4.1a. For comparison Figure 4.1b shows the true function; it is seen that the predictor based on the correlation function (4.2) is fairly accurate here.

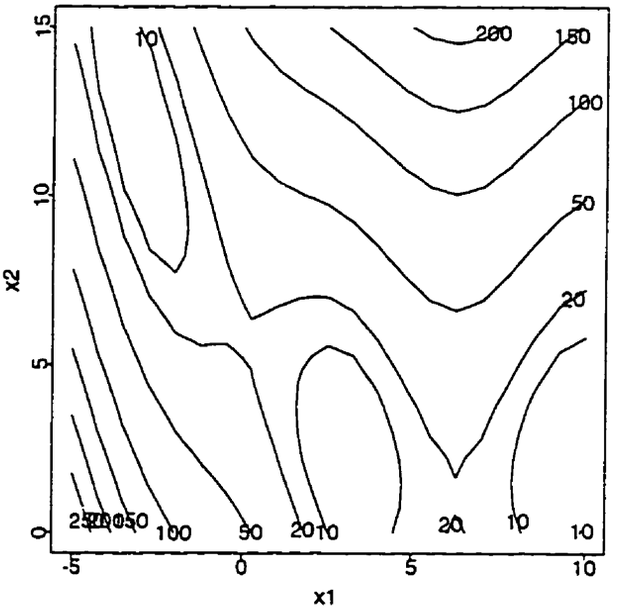
The four diagnostic plots can be seen in Figure 4.2. Figure 4.2a shows the function is extremely well fit except for the largest (and hence most uninteresting) value of y. Figure 4.2b shows that the standardized residuals are all in the range $[-2, 2]$. Even the one point with a big error in Figure 4.2a has a moderate standardized residual, i.e., its large error is in line with the measure of uncertainty provided by the standard error. Figure 4.2c indicates that the normal approximation is fairly good. Figure 4.2d clearly attaches the highest expected improvement to the lowest y. Some of the smaller y's have some expected improvement, while the expected improvement for larger y's is essentially 0. The diagnostic plots indicate that the model fits well and the expected improvement strategy is promising.

We then start the expected improvement algorithm. The points from the initial 21-point experimental design (denoted by dots) and from the sequential optimization (denoted by their respective numbers) can be seen in Figure 4.3.

We can see that the sequential points cluster around the three global optima. The minimal sampled function value after a total of 33 function evaluations is

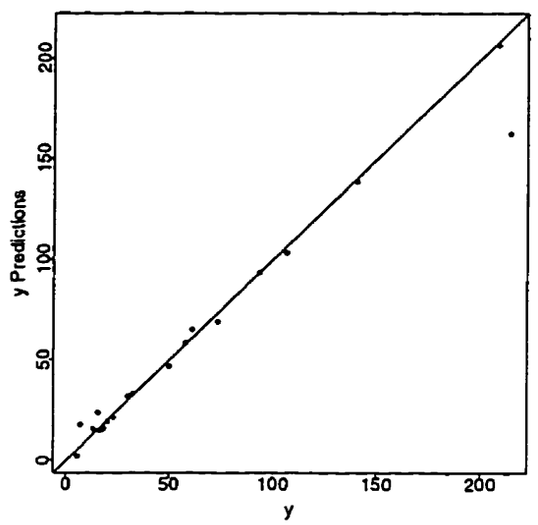


(a) Contour Plot of the Estimated Function. Dots Indicate Sampled Points.

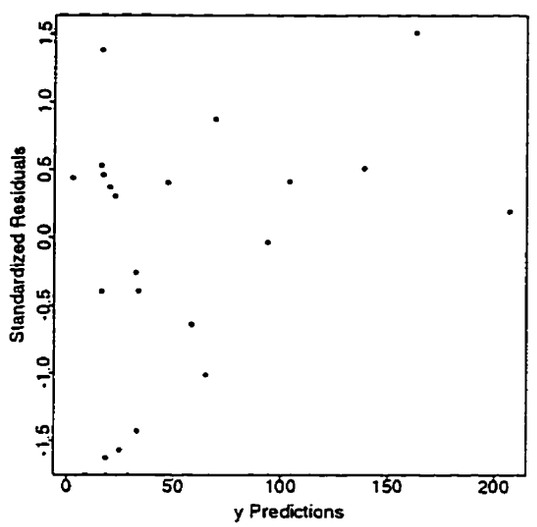


(b) Contour Plot of the True Function

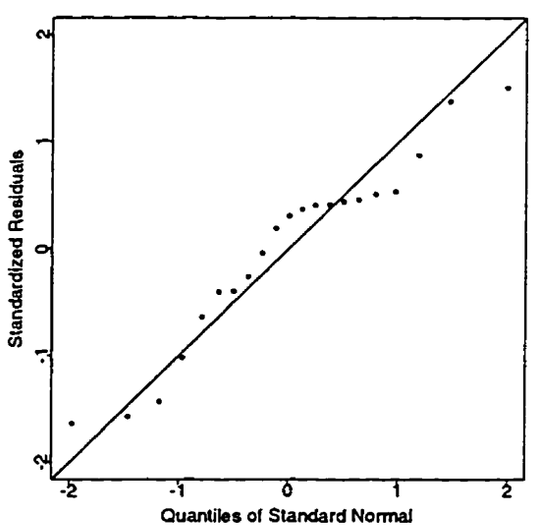
Figure 4.1: Branin Function: Contour Plots of the Estimated and True Function



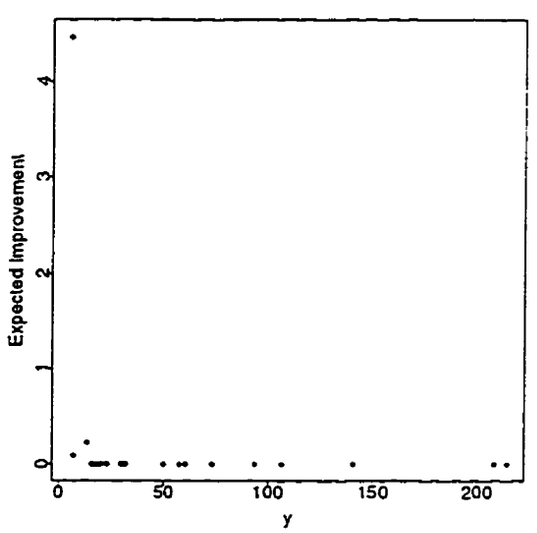
(a) Cross-validated Predictions versus True Values



(b) Standardized Cross-validated errors versus Predictions



(c) Q-Q Plot of the Cross Validated Residuals



(d) Cross-validated Expected Improvement versus True Values

Figure 4.2: Branin Function: Diagnostic Plots

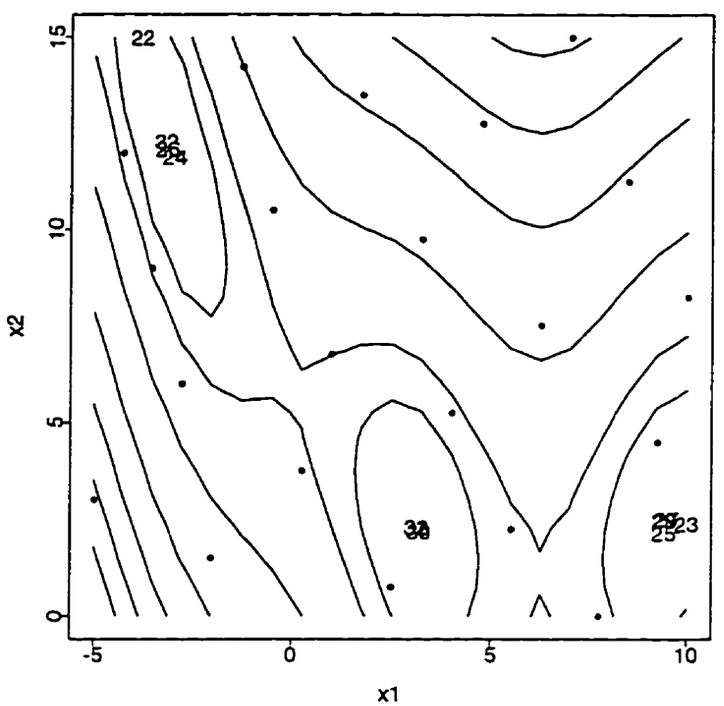


Figure 4.3: Branin Function: Initial Experimental Design (Dots) and Points Introduced by the Sequential Minimization (Case Numbers). Contours of the True Function are Shown.

0.39790, the true minimum is about 0.39788. The relative tolerance for the stopping criterion was set to .0001.

Table 4.1 gives an overview of tolerances and other performance criteria for the expected improvement algorithm applied to the Branin function and other functions. Table 4.2 compares the number of function evaluations needed by various global optimization methods. The functions are from the test suite of functions introduced by Dixon and Szegö (1978) which is often used for comparison purposes.

More extensive tables are given in Törn and Žilinskas (1989) and Jones et al. (1993). There are some difficulties associated with comparing the numbers in Table 4.2 since the stopping criteria are all different. Often stopping rules do not exist and instead the number of function evaluations is reported when the optimization first reaches a specified tolerance value of the (in practice unknown) global minimum. Mockus' (1989) Bayesian method using a Wiener field needs 189 function evaluations.

The 3-dimensional Hartman function (H3) which is also part of the test suite introduced by Dixon and Szegö (1978) is dealt with analogously to the Branin function (Br). The diagnostics look similar and no transformation is needed. Results are given in Tables 4.1 and 4.2.

4.4.2 Goldstein-Price Function (Gp)

The Goldstein-Price function (Törn and Žilinskas, 1989) also has two independent variables:

$$\begin{aligned} f(x_1, x_2) = & \\ & \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \\ & \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]. \end{aligned} \quad (4.10)$$

The variables x_1 and x_2 are both defined on the interval $[-2, 2]$. The Goldstein-Price function has one global minimum that is equal to 3 at $(0, -1)$. Not far from the global minimum, there are three local minima. The function values range over several orders of magnitudes.

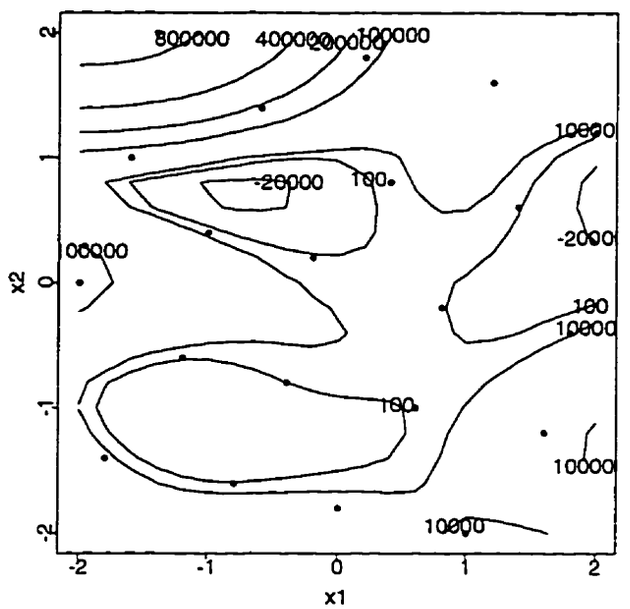
The initial experimental design is identical to the one used for the Branin func-

	Br	Gp	H3	H6	Sh10
Transformation	none	ln	none	ln	inverse
Initial Observations	21	21	30	51	40
Total Observations	33	106	38	125	131
Target Relative Tolerance	.00010	.00010	.00010	.00010	.01000
Actual Relative Tolerance	.00002	.00001	.00009	.00006	.00380
N until Target Rel. Tol. Reached	29	95	38	124	82

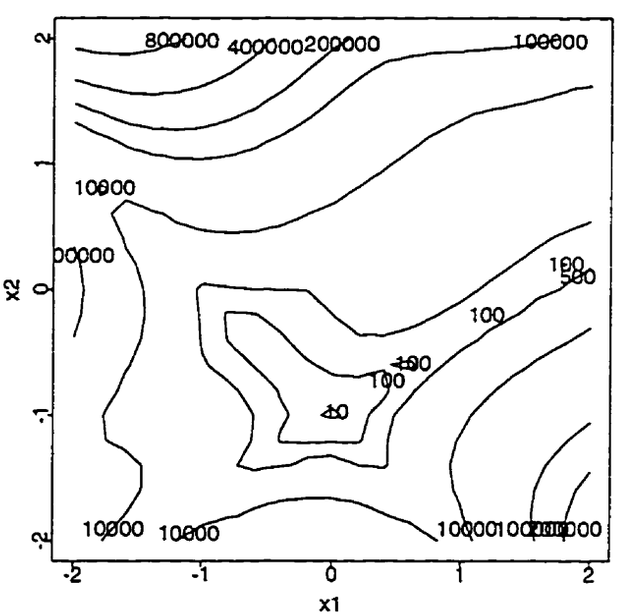
Table 4.1: Function Evaluations and Tolerances for Test Functions: Branin Function(Br), Goldstein-Price Function(Gp), 6-dimensional Hartman Function (H6), 3-dimensional Hartman Function (H3), Shekel Function with 10 Local Optima (Sh10). "N until Target Rel. Tol. reached" Refers to the Number of Points Until the Target Tolerance on the Original Scale was First Actually Reached. All Tolerances are on the Original Scale.

Authors	Br	Gp	H3	H6	Sh10
Kostrowicki and Piela (91)	*	120	200	200	12000
Perttunen (90)	97	82	264	*	250
Perttunen and Stuckman (90)	109	113	140	175	109
Mockus (78)	189	362	513	1232	1209
Žilinskas (80a)	164	165	363	627	2224
Žilinskas (86)	133	153	285	531	533
Jones, Perttunen, Stuckman (93)	195	191	199	571	145
Schonlau (97)	33	106	38	125	131

Table 4.2: Function Evaluations of Global Optimization Algorithms Based on Statistical Models of Objective Functions for Test Functions : Branin (Br). Goldstein-Price (Gp), the 3- and 6-dimensional Hartman functions (H3,H6) and the Shekel Function with 10 Local Optima (Sh10). The Symbol * Indicates that the Method was not Applied to the Test Function. This Table is Compiled from more Extensive Tables in Törn and Žilinskas (1989, Table 8.8) and Jones et al. (1993).



(a) Contour Plot of the Estimated Function. Dots Indicate Sampled Points.



(b) Contour Plot of the True Function

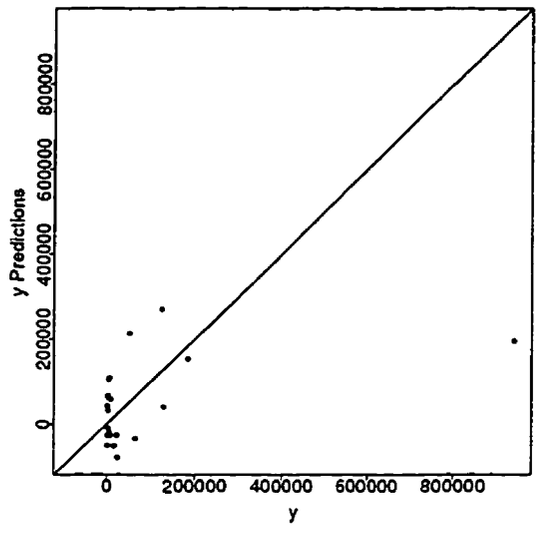
Figure 4.4: Goldstein-Price Function: Contour Plots of the Estimated and True Function

tion, scaled to suit the range of the x-variables of the Goldstein-Price function. Figure 4.4 compares the estimated function after initial modeling with the contours of the true function.

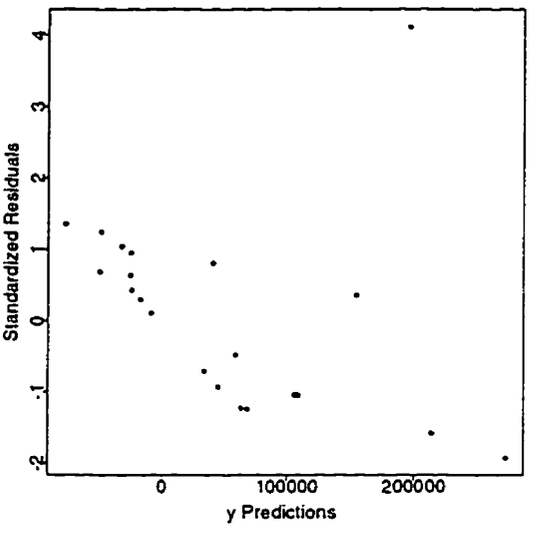
The diagnostic plots for the Goldstein-Price function can be seen in Figure 4.5. The first plot indicates that the function is predicted poorly, even if the largest function value is ignored. The second plot has one very large standardized residual of about 4. Thus, the standard error is underestimating prediction uncertainty, and the expected improvement algorithm is in danger of terminating prematurely. It appears, however, that the standardized residuals are larger for large predicted values. The Q-Q plot highlights the one very large standardized residual. The cross validated expected improvement plot indicates that there is little discriminating power between large and small y values.

The function values of the initial sample range over several orders of magnitude, and the cross validated residuals seem to be increasing with the magnitude of the response. This is suggestive of a logarithmic transformation of the response. We refit the model in $\ln(y)$ and obtain another set of diagnostic plots (Figure 4.6). The first plot now shows more relationship, though accuracy is is not as good as for the Branin function. There is no apparent trend in the second plot any more, and the standardized residuals are roughly within $[-2, 2]$. The Q-Q plot shows that the (fairly large) uncertainty of prediction is well represented by the normal approximation. Overall, we have a predictor that is fairly inaccurate given only 21 sampled points, but the amount of uncertainty is well estimated by our model. That the ln transformation promises to work reasonably well is confirmed by the last plot which shows that, with the exception of one point, low true values give the largest expected improvements.

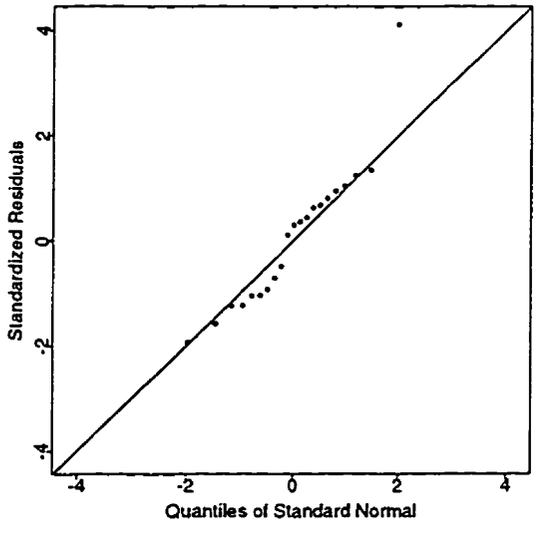
The points from the initial 21-point experimental design (denoted by dots) and



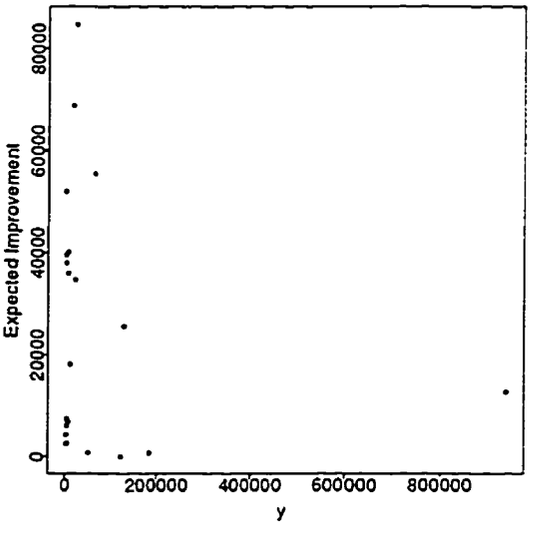
(a) Cross-validated Predictions versus True Values



(b) Standardized Cross-validated errors versus Predictions

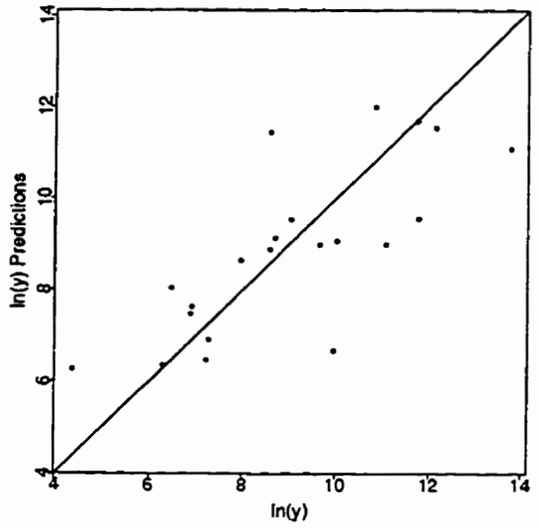


(c) Q-Q Plot of the Cross Validated Residuals

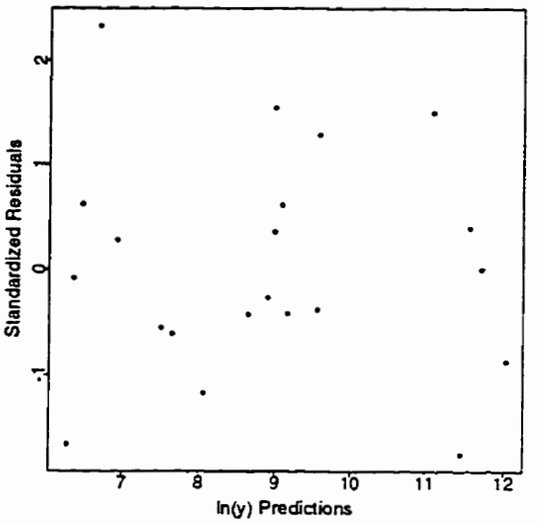


(d) Cross-validated Expected Improvement versus True Values

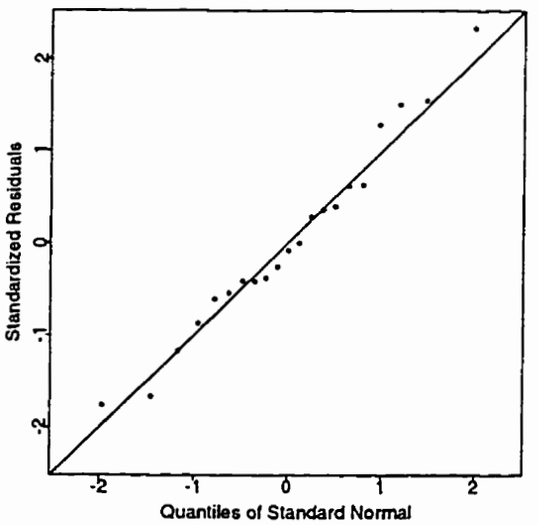
Figure 4.5: Goldstein-Price Function: Diagnostic Plots



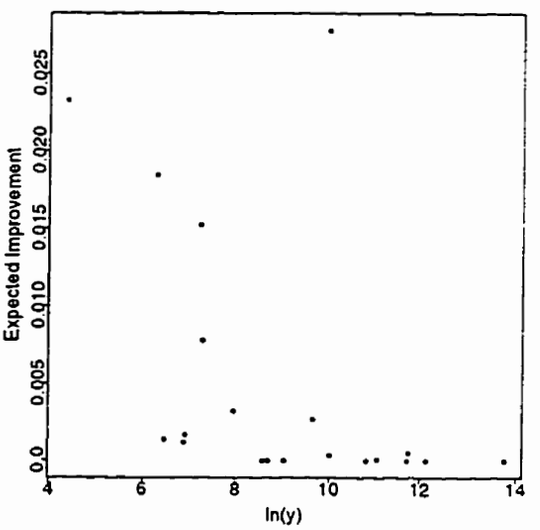
(a) Cross-validated Predictions versus True Values



(b) Standardized Cross-validated Errors versus Predictions



(c) Q-Q Plot of the Cross Validated Residuals



(d) Cross-validated Expected Improvement versus True Values

Figure 4.6: Ln Goldstein-Price Function: Diagnostic Plots

the sequential optimization (denoted by their respective numbers) can be seen in Figure 4.7. The optimization initially focuses on a local minimum close to the global minimum. After the local minimum is explored the algorithm finds the global minimum. The algorithm stops after a total of 106 observations. The global minimum on the ln scale is approximately 1.09861. The smallest function evaluation sampled is also 1.09861. The absolute tolerance for the stopping criterion was set to .0001 corresponding to a relative tolerance of .0001 on the original scale. The results for different global optimization algorithms can be seen in Table 4.2.

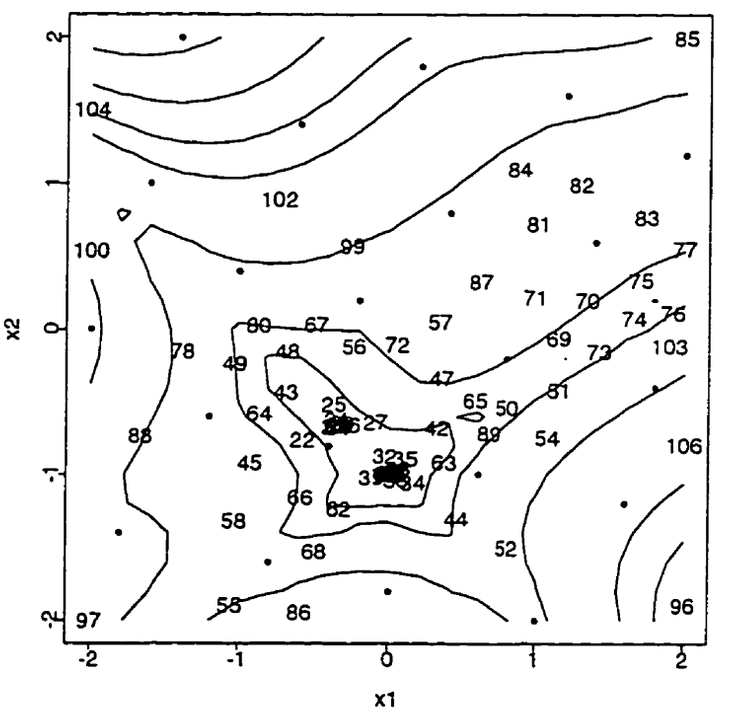


Figure 4.7: Ln Goldstein-Price Function: Initial Experimental Design (Dots) and Points Introduced by the Sequential Minimization (Case Numbers). Contours of the True Function are shown.

4.4.3 Hartman 6 Function (H6)

The Hartman 6 function (Törn and Žilinskas, 1989) is

$$f(x_1, \ldots, x_6) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} \alpha_{ij}(x_j - p_{ij})^2\right]. \tag{4.11}$$

where c_i, p_{ij}, and α_{ij} are coefficients given in Table 4.3. The x ranges are $0 \leq x_i \leq 1$ for $i = 1, \ldots, 6$. There is one minimum.

i	α_{ij} , $j = 1, \ldots, 6$						c_i
1	10	3	17	3.5	1.7	8	1
2	.05	10	17	.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	.05	10	.1	14	3.2

i	p_{ij}, $j = 1, \ldots, 6$					
1	.1312	.1696	.5569	.0124	.8283	.5886
2	.2329	.4135	.8307	.3736	.1004	.9991
3	.2348	.1451	.3522	.2883	.3047	.6650
4	.4047	.8828	.8732	.5743	.1091	.0381

Table 4.3: Hartman 6 Function: Coefficients for (4.11)

For the initial experimental design we choose 51 points (the choice of 51 results in convenient spacing at 1/50 of the range). Diagnostic plots for the Hartman 6 function are similar to those for the Goldstein-Price function. Again a ln transformation is suggested. (In fact, $-\ln(-y)$ was used as the original 51 function values are all negative.)

Since the Hartman 6 function is six-dimensional, visualization of the estimated function surface is not straightforward. Therefore, we inspect main and joint effects instead. Figure 4.8 shows main effects of x_1, x_4 and x_5, and Figure 4.9 the joint

effect between x_5 and x_6 for the transformed Hartman 6 function. The pointwise confidence intervals plotted are based on the standard error given by Theorem 2. A normal distribution for uncertainty is assumed stemming from the assumption that the process is Gaussian. The remaining main effects and joint effects are approximately constant. From these plots it appears that the function is probably

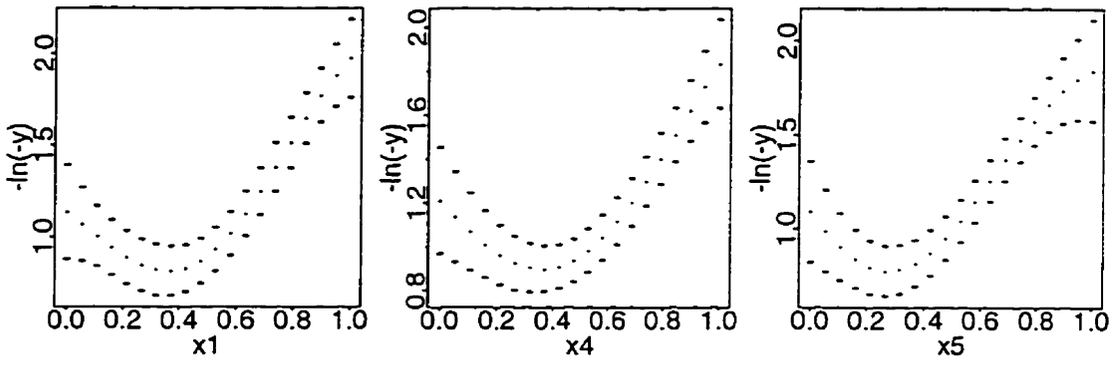


Figure 4.8: Main effects for x_1, x_4 and x_5 of the Hartman 6 Function with Transformation $-\ln(-y)$. The Middle Line is the Estimated Effect, the Upper and Lower Lines are Approximate 95% Pointwise Confidence Limits.

unimodal. One might even be tempted to proceed with a local minimization algorithm, using starting values from the graphical analysis. Furthermore, from the graphs it is clear that the global minimum occurs with x_1, x_4 and x_5 roughly in $[0.1, 0.5]$, while x_6 will be in $[0.5, 1.0]$. The remaining variables, x_2 and x_3, are relatively unimportant.

The insights from the initial graphical analysis could be used to reduce the search space, but we apply the expected improvement algorithm to $-\ln(-y)$ with all six variables on [0,1] to facilitate comparison. Two dimensional projections of the experimental design and the points resulting from the sequential optimization

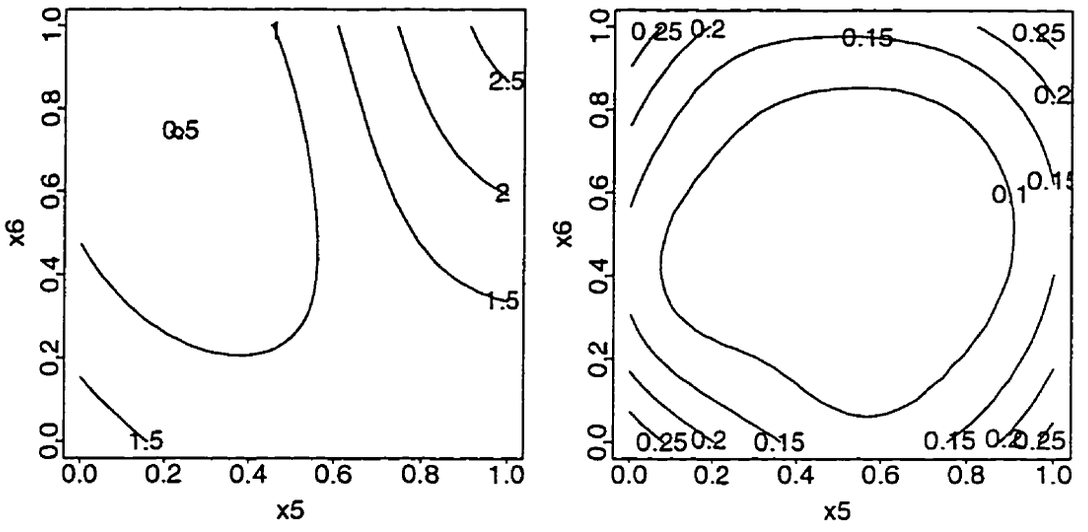


Figure 4.9: Joint Effect of x_5 and x_6 and its Pointwise Standard Error for the Hartman 6 Function with Transformation $-\ln(-y)$.

can be seen in Figure 4.10.

We can see that the algorithm explores the edges and also clusters around one single point which indeed is the minimum. During the minimization, only 74 additional points were sampled, a total of 125 points. The minimal value found is -1.20066 on the transformed scale; the true minimum equals -1.20068. The absolute tolerance was set to 0.0001 or a relative tolerance of .0001 on the original scale. The results for different global optimization algorithms can be seen in Table 4.2. Mockus' (1989) Bayesian algorithm needs 1232 observations.

4.4.4 Shekel 10 Function (Sh10)

The remaining functions in the suite introduced by Dixon and Szegö (1978) are the 4-dimensional Shekel family of which we present only the most difficult one, the

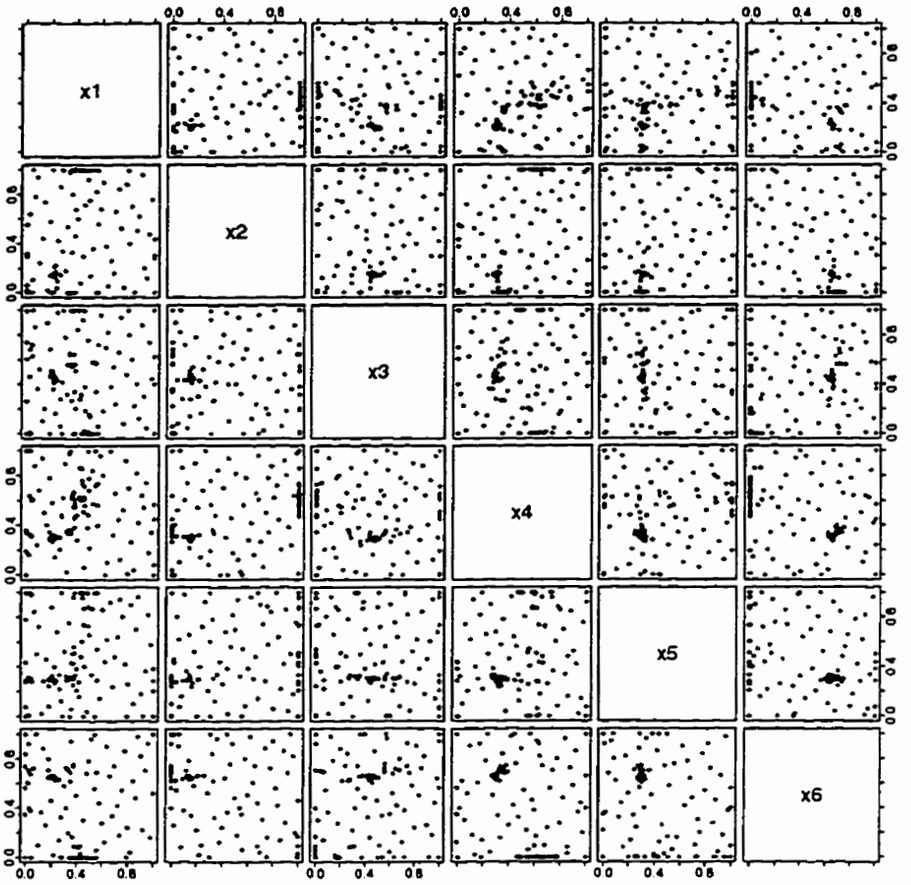


Figure 4.10: $-\ln(-y)$ Hartman 6 Function: Initial Experimental Design and Points Introduced by the Sequential Minimization.

Sh10 function with 10 local optima:

$$f(\mathbf{x}) = -\sum_{i=1}^{10} \frac{1}{(\mathbf{x} - A(i))(\mathbf{x} - A(i))^t + c_i} \tag{4.12}$$

where $\mathbf{x} = (x_1, x_2, x_3, x_4)$ and the c_i and A_i are coefficients and coefficient vectors given in Table 4.4. The x ranges are $0 \leq x_i \leq 10$ for $i = 1, \ldots, 4$.

The Shekel 10 function is relatively flat with the exception of 10 sharp wells for the local optima. Figure 4.11 shows a marginal view of the well at the global

i	A(i)				c_i
1	4	4	4	4	.1
2	1	1	1	1	.2
3	8	8	8	8	.2
4	6	6	6	6	.4
5	3	7	3	7	.4
6	2	9	2	9	.6
7	5	5	3	3	.3
8	8	1	8	1	.7
9	6	2	6	2	.5
10	7	3.6	7	3.6	.5

Table 4.4: Shekel 10 Function: Coefficients for (4.12)

minimum. For the minimization of the 4-dimensional Shekel 10 function we choose 40 starting points.

Figure 4.12 is an attempt to visualize the first 131 four-dimensional data points for the Shekel function. Each observations is represented by four points; one for each dimension. For example, suppose that the first of the four dimensional observations for the Shekel 10 Function is given by $(2, 5, 7, 0)$. For visualization, the four dimensional observation is represented by four points in two dimensional space: (0.7,1), (1.0,2), (1.2,3) and (0.0,4). The first coordinate of each point is standardized such that the range limits (0 and 10 for the Shekel Function) are represented by 0.5 and 1.5, the second is just an increasing integer valued counter. The range of 0.5 to 1.5 is chosen such that its midrange represents the observation number, here 1.0 corresponding to the first observation. (A different range length could be chosen as long as it is the same for all observations.)

The four points corresponding to an observation are then connected such that they form a piecewise linear line. An extra point at mid-range with coordinates

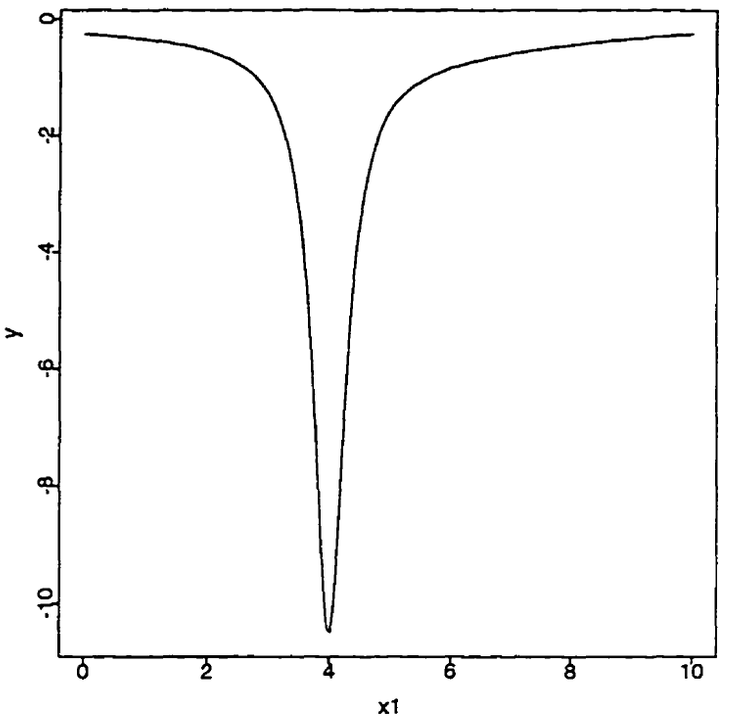


Figure 4.11: Shekel 10 Function: Plot of y versus x_1 where $x_2 = x_3 = x_4 = 4.0$. This is a Marginal Plot of the "Sharp Well" at the Global Minimum.

(1,0) is also connected for better orientation. Since the first coordinate of this extra point is at mid-range, it always corresponds to the observation number.

Each subsequent line is offset by 1 unit in the first coordinate from the previous one: the first coordinate of the i^{th} point has the range of $i - 0.5$ to $i + 0.5$, and the extra point has the coordinates $(i, 0)$.

As a result, observations with the same (similar) coordinates have the same (similar) lines except for the offset. Note that depending on the length of the range chosen, lines corresponding to adjacent observations may overlap. Details in form

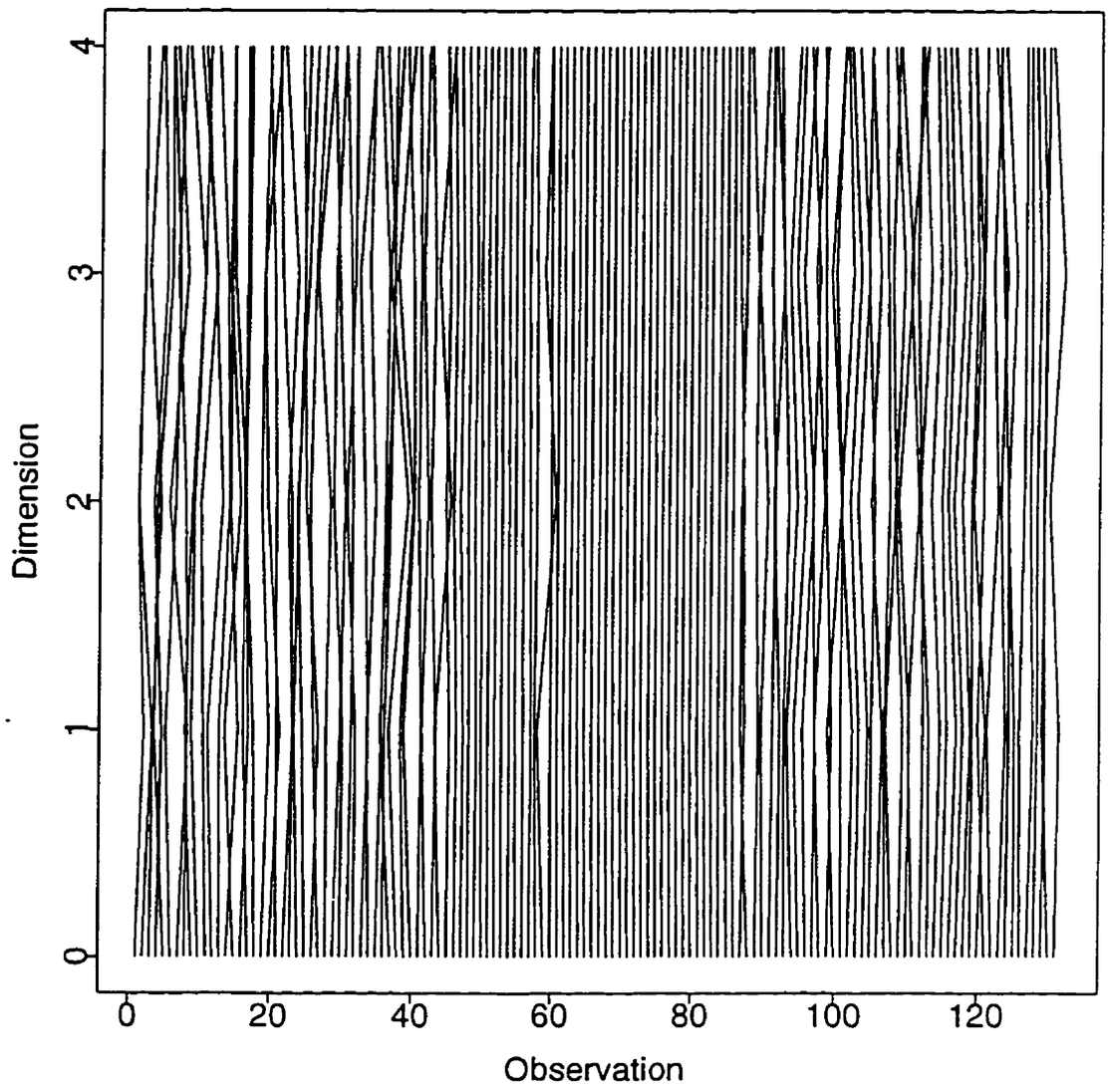


Figure 4.12: Inverse Shekel 10 Function: Visualization of the Design (40 points) and Points Sampled During Minimization (91 points)

of a Splus function are given in Appendix D.

The lines corresponding to the first 40 sampled points in Figure 4.12 are all dissimilar because they represent a space filling sample. After the initial 40 function evaluations, the algorithm chooses observations $41-44$ in different locations, which may be interpreted as a global search. The f^n_{min} decreases slightly. Observations $45-86$ (except observation 59) home in on the global minimum, representing a local search. From then on the observations are spread throughout the space, i.e. they represent a global search. During this global search f^n_{min} does not decrease any more. Figure 4.12 makes the duality between local search (similar lines) and global search (dissimilar lines) very clear.

Upon discovery of the global "sharp well", the MLE adjusts the estimate of the standard error upwards which leads to an increase in the expected improvement. In what follows, the model accounts for the possibility of undiscovered similar "sharp wells". As a result the expected improvement remains approximately constant during the global search while the algorithm starts to fill the 4-dimensional space with points to rule out that possibility. We therefore decided to set the target tolerance to 0.01 for the Shekel 10 function leading to 131 function evaluations. At about that time it becomes clear that the algorithm essentially tries to fill the space more densely. We would be suspicious of any algorithm that will terminate easily before it can rule out the possibility of further "sharp wells".

4.5 Discussion

In this chapter we have used the Bayesian approach to Global Optimization with the objective of reducing the number of function evaluations needed and still terminating with reliable error tolerances. We have achieved this goal by improving the

fit of the stochastic model in two ways: (1) by replacing the commonly used Wiener field with the more flexible generalized exponential correlation function and (2) by assessing model adequacy and if needed attempting to improve it by an appropriate transformation.

Since the correlation function for the stochastic process model adopted here is much more flexible than the Wiener process correlation function, it is no surprise that it leads to a smaller number of function evaluations. The examples given demonstrate that the difference can be quite substantial.

This difference comes at the cost of a greater computational burden which makes the method very ineffective if the target function is cheap to evaluate. Further, the evaluation of the predictor requires the inversion of a correlation matrix of size n, where n is the sample size. Realistically, this puts an upper bound on the number of function evaluations that can be analyzed at a few hundred. Since the method proposed specifically aims to reduce the number of function evaluations needed, this is not an issue in practice for many problems.

Mockus (1989) used the expected improvement algorithm for a fixed number of observations and then proceeded with a local optimization technique. The local optimizer used the minimal sampled function value as a starting value. The rationale is that locally the stochastic model is less effective, and a steepest descent model will reach the required accuracy faster. A local optimization technique could similarly follow on the algorithm that we present.

Finally, visualization of the response function provides insight into the qualitative features of the input-output relationship. In an engineering context, this insight is useful for assessing trade-offs and suggesting new engineering approaches.

Chapter 5

Extensions to Bayesian Global Optimization

5.1 Introduction

In this chapter we consider several further aspects of Bayesian Global Optimization. In Section 5.2 we generalize the expected improvement criterion by introducing an additional parameter. The additional parameter determines how global versus local the search will be. This will be illustrated with the Goldstein-Price function.

In the outline of the expected improvement algorithm we have always assumed that we were to sample one point at a time. Section 5.3 relaxes this assumption and addresses the question of how to sample several points at a time. Again, we use the Goldstein-Price function for illustration.

In Section 5.4 we consider the problem of finding the global minimum subject to constraints on additional response variables. An example from the automotive industry is given.

5.2 Generalized Expected Improvement

The expected improvement algorithm works very well especially when the unknown function is well approximated by the stochastic process model. Given the correlation parameters, the expected improvement criterion optimally chooses where to sample one point according to an average case analysis. The paradigm of the average case analysis, given the correlation parameters, thus ultimately determines the balance between the global and local components of the search. When the correlation parameters are poorly estimated, an average case analysis is not sensible, and typically the search is too local.

It is therefore desirable to introduce a version of the expected improvement algorithm that searches more globally. We achieve this goal in this section by generalizing the expected improvement criterion to include an additional integer-valued parameter, g (for global). The larger the value g takes the more globally will the algorithm tend to search.

If the function is sampled at $\mathbf{x}$ to determine $y = y(\mathbf{x})$ then the improvement to the power of g, I^g, is defined as

$$I^g = \begin{cases} (f^n_{min} - y)^g & \text{if } y < f^n_{min} \\ 0 & \text{if otherwise} \end{cases} \tag{5.1}$$

where $g = 0, 1, 2, \ldots$. For $g = 0$ taking the expectation yields the probability of improvement:

$$E(I^0) = P(z < f'^n_{min}) = \Phi(f'^n_{min})$$

where $f'^n_{min} = \frac{f^n_{min} - \hat{y}}{s}$. The probability of improvement has been used as a criterion

in an axiomatically-based algorithm by Žilinskas (see Törn and Žilinskas . 1989) [1].

For $g = 1, 2, \ldots$ it is possible to show (see Appendix E) that $E(I^g)$ is

$$E(I^g) = s^g \sum_{k=0}^{g} (-1)^k \begin{pmatrix} g \\ k \end{pmatrix} {f'^n_{min}}^{g-k} T_k \quad , \tag{5.2}$$

and the T_k are given by (see Appendix E)

$$T_k = \begin{cases} -\phi(f'^n_{min}) \left[\sum_{j=1}^{(k-1)/2} {f'^n_{min}}^{2(j-1)} \prod_{i=j}^{(k-1)/2} 2i + {f'^n_{min}}^{k-1} \right] & \text{if } k \text{ is odd} \\ \Phi(f'^n_{min}) \prod_{i=1}^{k/2} (2i-1) - \phi(f'^n_{min}) \left[\sum_{j=2}^{k/2} {f'^n_{min}}^{2j-3} \prod_{i=j}^{k/2} (2i-1) + {f'^n_{min}}^{k-1} \right] & \text{if } k \text{ is even} \end{cases} \tag{5.3}$$

Alternatively, the T_k satisfy the recursive equation (see Appendix E)

$$T_k = -\phi(f'^n_{min}) {f'^n_{min}}^{k-1} + (k-1) T_{k-2} \tag{5.4}$$

with starting points $T_0 = \Phi(f'^n_{min})$ and $T_1 = -\phi(f'^n_{min})$. This latter equation is easier to program.

For example, for the special cases $g = 1, 2, 3$ and $s > 0$ we obtain from (5.2):

$$\begin{aligned} E(I) &= s \left(f'^n_{min} \Phi(f'^n_{min}) + \phi(f'^n_{min}) \right) \\ E(I^2) &= s^2 \left(({f'^n_{min}}^2 + 1) \Phi(f'^n_{min}) + f'^n_{min} \phi(f'^n_{min}) \right) \\ E(I^3) &= s^3 \left(({f'^n_{min}}^3 + 3 f'^n_{min}) \Phi(f'^n_{min}) + (2 + {f'^n_{min}}^2) \phi(f'^n_{min}) \right) \end{aligned}$$

Of course, $g = 1$ reproduces the expected improvement derived earlier in (4.7).

[1] Table 4.2 includes comparisons with two different version of this algorithm

The case $g = 2$ is interesting as

$$E(I^2) = [E(I)]^2 + \mathrm{Var}(I) \,. \tag{5.5}$$

The criterion in (5.5) consists of (a monotone transformation of) the original criterion, the expected improvement $E(I)$, and the variation of the improvement $\mathrm{Var}(I)$. The variation tends to be larger further away from sampled points and thus represents a global component.

There is a tradeoff in choosing between small improvement with large probability (local search) versus large improvement with small probability (global search). As g increases larger improvements receive more weight and the search is more global.

In practice the question arises which value of g to choose. There are two indicators that show that the choice of $g = 1$ may be undesirable: when the diagnostic plots indicate that the unknown model is poorly approximated by the stochastic process model even after transformation and/or when the design contains points which are very close to one another. As g increases the points will spread out more. which can be used as a rough guide to what value g should take. The choice of $g = 0$, or the probability of improvement, results in a more local search and is probably not advisable unless one is reasonably certain that the approximate location of all local optima has been established.

The relative and absolute stopping criteria based on $E(I)$ no longer apply for the generalized expected improvement. Instead of $E(I)$, we use $[E(I^g)]^{1/g}$ for $g \geq 1$. Since I is nonnegative and I^g is a convex function of I for $I \geq 0$, Jensen's inequality applies and yields $[E(I^g)]^{1/g} > E(I)$. Assuming the same tolerances, stopping rules based on $[E(I^g)]^{1/g}$ will tend to sample more points. To avoid possible confusion, note that I^g is maximized over $\mathbf{x}$ (not I) and that the function is not convex in $\mathbf{x}$.

5.2.1 Example: Goldstein-Price Function

In this section we minimize the Goldstein-Price function with $g = 2$ and $g = 5$ ($g = 1$ was dealt with in Section 4.4.2). We use $E(I^g)^{1/g} < .001$ as the absolute stopping criterion on the log scale for all three minimizations. This corresponds to a relative tolerance of approximately .001 on the untransformed scale. Table 5.1 summarizes the results of both minimizations and the corresponding one for $g = 1$ that was seen earlier in Figure 4.7 (where the targeted tolerance was .0001). From

	Actual Rel Tol	n
g=1	.006	56
g=2	.000008	127
g=5	.000037	173

Table 5.1: Ln Goldstein-Price Function: Comparison of Minimizations where $g = 1, 2, 5$. The Target Tolerance for the Stopping Criterion was .001 in All Cases. "Actual Rel Tol" Refers to the Relative Tolerance on the Original Scale, "n" to the Number of Function Evaluations until the Stopping Criterion is Met.

Table 5.1, we see that the number of function evaluations increases considerably as g increases. However, we attribute this mostly to the problem that the stopping criterion $E(I^g)^{1/g} < .001$ is harder to meet for larger g values.

The final designs can be seen in Figure 4.7 (for $g = 1$, first 56 points only). Figure 5.1 (for $g = 2$) and Figure 5.2 (for $g = 5$). With increasing global parameter the points are more spread out. For $g = 5$ the optimization does not get initially stuck in the local optimum as is the case for $g = 1$ and $g = 2$. On the other hand, $g = 5$ also samples many points in areas that are uninteresting in hindsight.

Both for $g = 2$ and $g = 5$ the actual relative tolerances obtained are very low, especially when compared with that achieved for $g = 1$. The tolerance values are expected to be lower as the number of function evaluation increases and the more local the search is (i.e. small g) - provided that the global minimum is found.

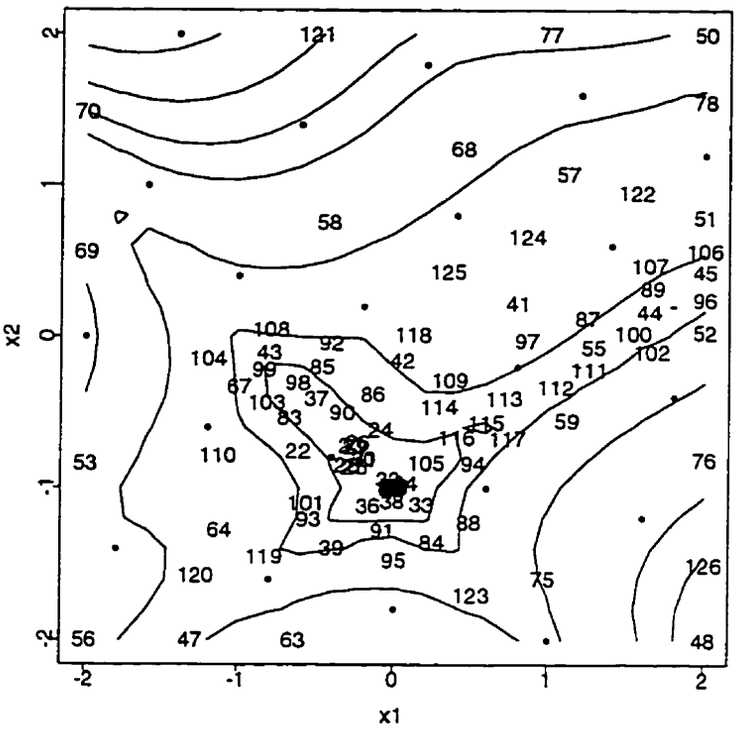


Figure 5.1: Ln Goldstein-Price Function: Initial Experimental Design (Dots) and Points Introduced by the Sequential Minimization with $g = 2$ (Case Numbers).

For the Goldstein-Price function, one would probably prefer a minimization with $g > 1$, especially since the first few points chosen with $g = 1$ are very close to one another.

Incidentally, minimization for the Branin function with $g = 2$ and $g = 5$ results in virtually identical designs except that the points are sampled in a slightly different order.

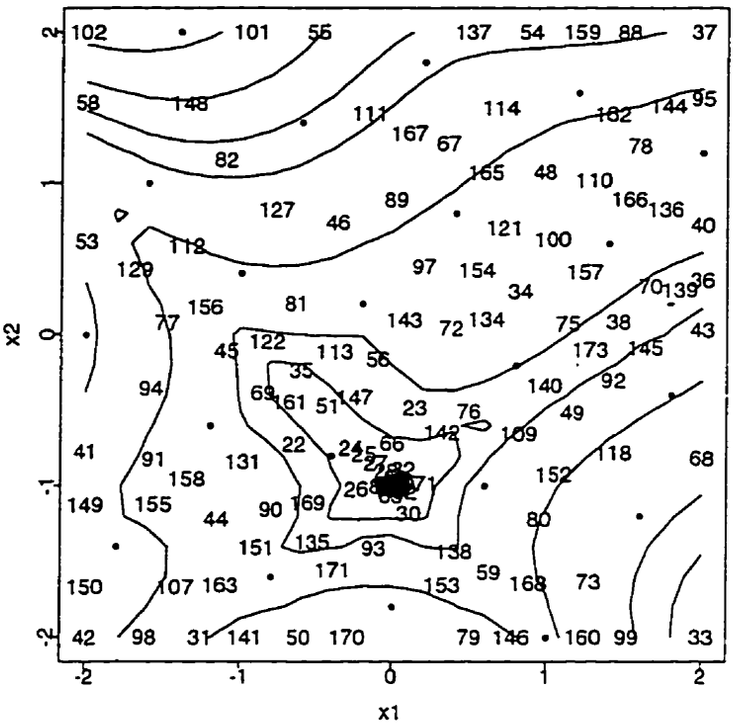


Figure 5.2: Ln Goldstein-Price Function: Initial Experimental Design (Dots) and Points Introduced by the Sequential Minimization with $g = 5$ (Case Numbers).

5.3 Sequential Design in Stages

The expected improvement algorithm is a sequential one-point-at-a-time algorithm. For many applications sampling one point at a time is unrealistic. For one, unless the sampling can be computer automated it is very time consuming. Second, it may also be more cost effective to have only a few stages where at each stage a number of points are sampled. In other words, sampling m points at a time may be a lot cheaper than sampling at m stages.

In order to select m further points, generalizing (4.5), we would ideally find design points $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+m}$ that maximize

$$E_{Y_1} E_{Y_2} \ldots E_{Y_m}(I_m) \tag{5.6}$$

where $Y_1, Y_2, \ldots Y_m$ are the random variables corresponding to the m point design and the m-step improvement I_m over f^n_{min} is defined as

$$I_m = \max(0, f^n_{min} - y_1, \ldots, f^n_{min} - y_m).$$

Unfortunately, this is a much harder problem than (4.5) due to the fact that it involves multiple integrals with normal densities. Rather than computing the integrals numerically, which would be very time consuming, we suggest an alternative strategy.

We simplify (5.6) in two ways : (a) we compute the expectations sequentially rather than jointly and (b) we update at each step the estimate of the standard error σ (but not f'^n_{min}). The two simplifications induce the following expected improvement at the $(n+i)^{th}$ step :

$$E^{(n+i)}(I) = \begin{cases} s^{(n+i)} \left(f'^n_{min} \Phi(f'^n_{min}) + \phi(f'^n_{min}) \right) & \text{if } s^{(n+i)} > 0 \\ 0 & \text{if } s^{(n+i)} = 0 \end{cases} \tag{5.7}$$

where $i = 1, \ldots, m$ and $f'^n_{min} = \frac{f^n_{min} - \hat{y}^{(n)}}{s^{(n)}}$. Note that $s^{(n+i)}$ the standard error of prediction at $\mathbf{x}_{n+i}$ depends only on $\mathbf{x}_{n+i}$ and the correlation parameters, not on the (unknown) response. We do not update $s^{(n)}$ in f'^n_{min} because this would imply that we know the difference $f^n_{min} - \hat{y}^{(n)}$ with greater certainty than we actually do and would lead to an agglomeration of points at one site.

The first simplification is similar to forward selection in linear regression, in that instead of computing the optimal subset of n variables, sequentially the best variable at each step is chosen. Unlike in forward selection, however, where the effect of the inclusion of any one variable can be fully assessed, the predictor necessary to compute the expected improvement is only known for the first step. Hence the second simplification is introduced.

One might argue that this sampling strategy should be used instead of the Latin Hypercube scheme for selecting the initial set of sampling points. This is not possible because f'^{0}_{min} cannot be computed. One could use one or two initial starting points to overcome this problem and then use the sampling strategy proposed in this section. Unfortunately, this does not work well either. because the true surface is very poorly approximated with so few points.

The sequential design in stages can also be applied to the generalized expected improvement methodology. As before with $E(I)$, the criterion $E(I^g)$ is a function of s and f'^{n}_{min}. The expected improvement to the power of g at the i^{th} step can therefore be obtained from (5.2) where only s is updated :

$$E^{(i)}(I^g) = (s^{(i)})^g \sum_{k=0}^{g} (-1)^k \begin{pmatrix} g \\ k \end{pmatrix} f'^{g-k}_{min} T_k$$

where the T_k are defined as before.

After the initial function evaluations and before starting a one-point-at-a-time minimization, it may be useful to generate one stage of the sequential design to find out whether the search is likely to proceed too locally.

5.3.1 Example: Goldstein-Price Function

Here we are demonstrating the design in stages with the *ln* Goldstein-Price Function where the global parameter is $g = 2$. After the initial 21 function evaluations we proceed in stages of 10 points each. The stopping criterion is $\sqrt{E(I^2)} < .001$.

After 13 stages or a total of 151 points the stopping criterion is met. Figure 5.3 shows the final design. After the initial stage, stage 1 chooses points close to one of

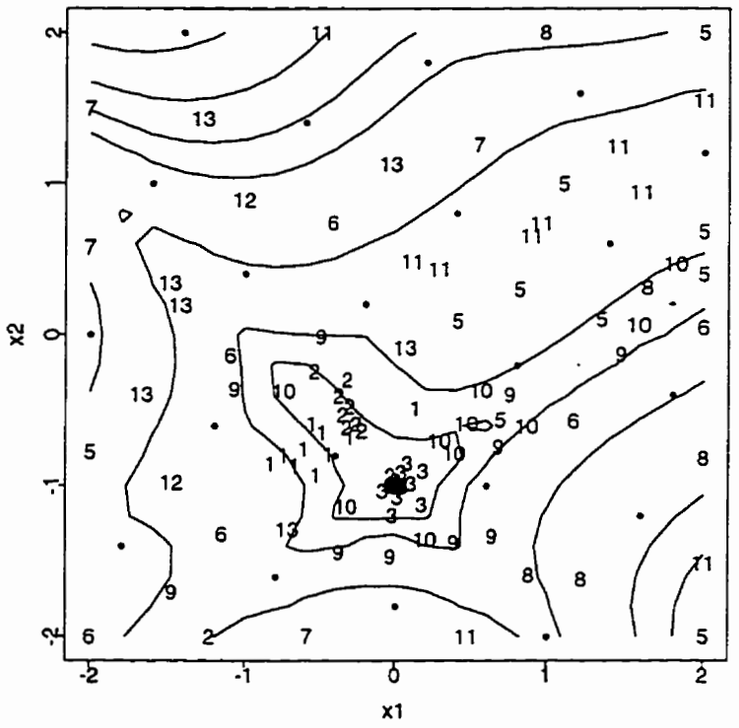


Figure 5.3: Ln Goldstein-Price Function: Initial Experimental Design (Dots) and 13 stages of 10 Points Each Introduced by the Sequential Minimization in Stages with $g = 2$ (Stage Numbers).

the local optima. One of the points in stage 2 comes relatively close to the global

optimum. From stage 3 onwards the function value of that point is known and, as a consequence, stages 3 and 4 sample exclusively around the global optimum. Actually, stage 4 points are much more closely clustered around the global optimum than stage 3 points. Due to the agglomeration of points around the global optimum this is difficult to make out in Figure 5.3. From stage 5 onwards many of the points are selected globally, through most of the stages still contain some points near the global minimum.

Compared to the one-point-at-a-time minimization of the *ln* Goldstein-Price function with $g = 2$ in Section 5.2.1, the number of function evaluations has gone up from 127 to 151. Both minimizations achieve about the same relative tolerance on the original scale which is smaller than .00001. Overall, the minimization in stages exhibits a greater spread of points.

5.4 Minimization Subject to Constraints on Additional Response Variables

In this section we consider the problem of minimizing a function subject to constraints on c additional response variables. A strategy is offered treating the predictions for the $c+1$ response variables as statistically independent. The strategy for the dependent case is outlined and still requires the specification of a certain covariance matrix (explained further below). In many practical applications it may be adequate to assume that prediction errors for several response variables are approximately independent.

Denote the c response functions acting as constraints by $g_1(\mathbf{x}), \ldots, g_c(\mathbf{x})$ and suppose we want to minimize $y(\mathbf{x})$ subject to $a_i < g_i(\mathbf{x}) < b_i$ for $i = 1, 2, \ldots, c$. For example, in the next section we have two measured outputs: one, undesirable

piston motion, is to be minimized and the other, friction of a piston, may never exceed a certain value. We define the improvement subject to constraints, $I_c(\mathbf{x})$, as

$$I_c(\mathbf{x}) = \begin{cases} f^n_{min} - y(\mathbf{x}) & \text{if } y(\mathbf{x}) < f^n_{min}\,, \qquad a_i \le g_i(\mathbf{x}) \le b_i \text{ for } i = 1, 2, \ldots, c \\ 0 & \text{otherwise} \end{cases}$$

where, like in Chapter 4, f^n_{min} is the minimum (feasible) y-value amongst the current n runs.

The expected improvement subject to constraints can then be derived as follows:

$$\begin{aligned} E_{y,g_1,g_2,\ldots,g_c}(I_c) &= E_y\left(E_{g_1,\ldots,g_c|y}(I_c)\right) \\ &= E_y\left(\int_{a_1}^{b_1}\cdots\int_{a_c}^{b_c} \max(f^n_{min} - y, 0)\; f_{g_1,\ldots,g_c|y}\; d_{g_1}\ldots d_{g_c}\right) \\ &= E_y\left(\max(f^n_{min} - y, 0)\int_{a_1}^{b_1}\cdots\int_{a_c}^{b_c} f_{g_1,\ldots,g_c|y}\; d_{g_1}\ldots d_{g_c}\right) \quad (5.8) \end{aligned}$$

Equation (5.8) could be evaluated numerically, if the multivariate normal distribution MVN$(y, g_1, \ldots, g_c)$ was completely specified. The covariance matrix is partitioned, with blocks on the diagonal corresponding to the within variable covariance structures, given by $\sigma^2_{z,i}\mathbf{R}_i$, $i = 1, \ldots, c$. Unfortunately the off diagonal blocks, corresponding to the between variable covariance structure, are unknown. It is not trivial to specify a sensible covariance structure between variables, because it is difficult to show that the resulting correlation structure is positive definite. It may be possible to borrow some ideas from the cokriging literature (e.g. Cressie, 1993, Section 3.2.3).

Treating the variables $y, g_1, \ldots, g_c$ as statistically independent, (5.8) simplifies

to

$$
\begin{aligned}
& E_{y,g_1,\ldots,g_c}(I_c) \\
&= E_y(I) \prod_{i=1,\ldots,c} (\Phi_{g_i}(b_i) - \Phi_{g_i}(a_i)) \\
&= E_y(I)\ P(a_1 < G_1 < b_1)\ P(a_2 < G_2 < b_2)\ \cdots\ P(a_c < G_c < b_c). \qquad (5.9)
\end{aligned}
$$

That is, treating the response variables as statistically independent the expected improvement is multiplied by the probability that each constraint is met. Equation (5.9) does not require the between variable covariance structure and can thus easily be computed.

5.4.1 Example: Piston Application

To illustrate this we will use an example from the automotive industry. We want to minimize undesirable piston motion ($pmax$) such that the friction of a piston (mp) does not exceed a certain value. A piston is a part of the engine in an automobile that moves up and down in an engine cylinder igniting fuel during every cycle. The two piston functions are related to two design variables (x-variables).

The piston problem originally included a second constraint as well as three additional x-variables. For illustration purposes and simplicity we have extracted the most interesting feature of this problem involving only one constraint and two x-variables.

Figure 5.4 gives contour plots of of the true objective and the true constraint, respectively. Since the contour levels for $pmax$ are irregularly spaced to accentuate features of the minimization region, we also give a perspective plot of $pmax$ (Figure 5.5). The actual constraint is $mp < 3$.

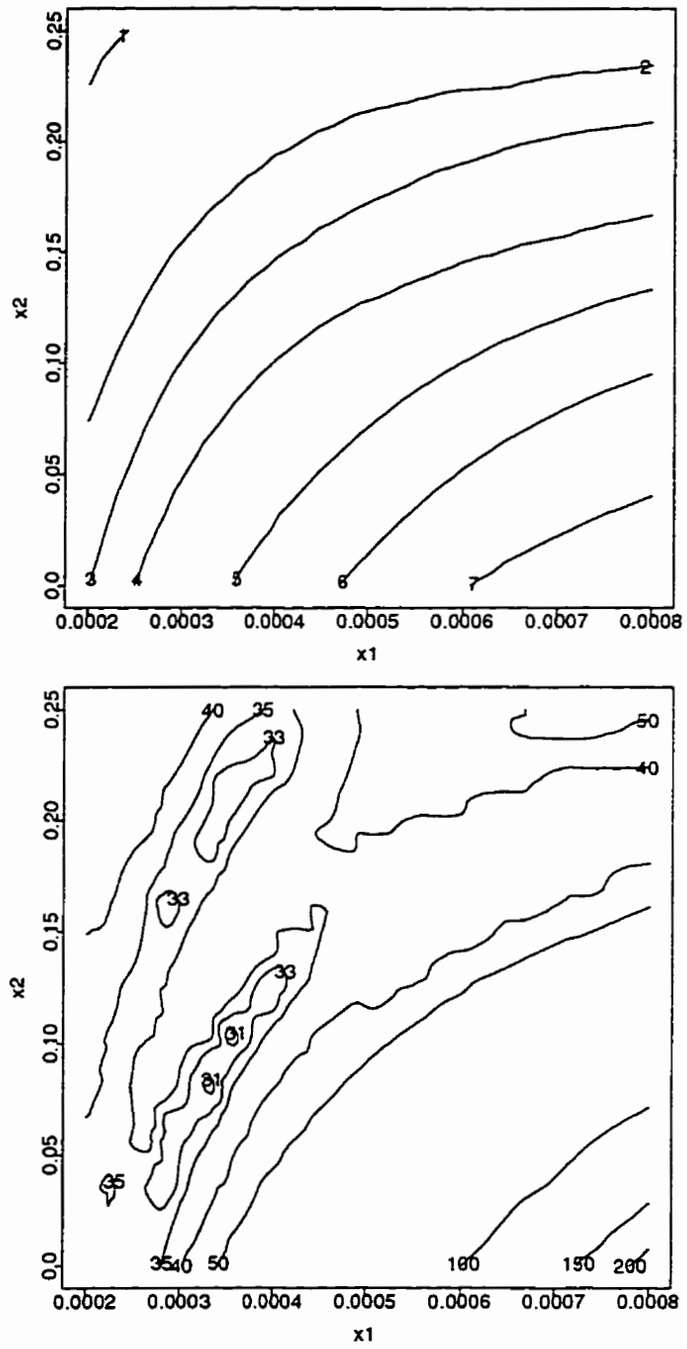


Figure 5.4: Piston Application: Contour Plot of the True Function for mp (top) and $pmax$ (Bottom)

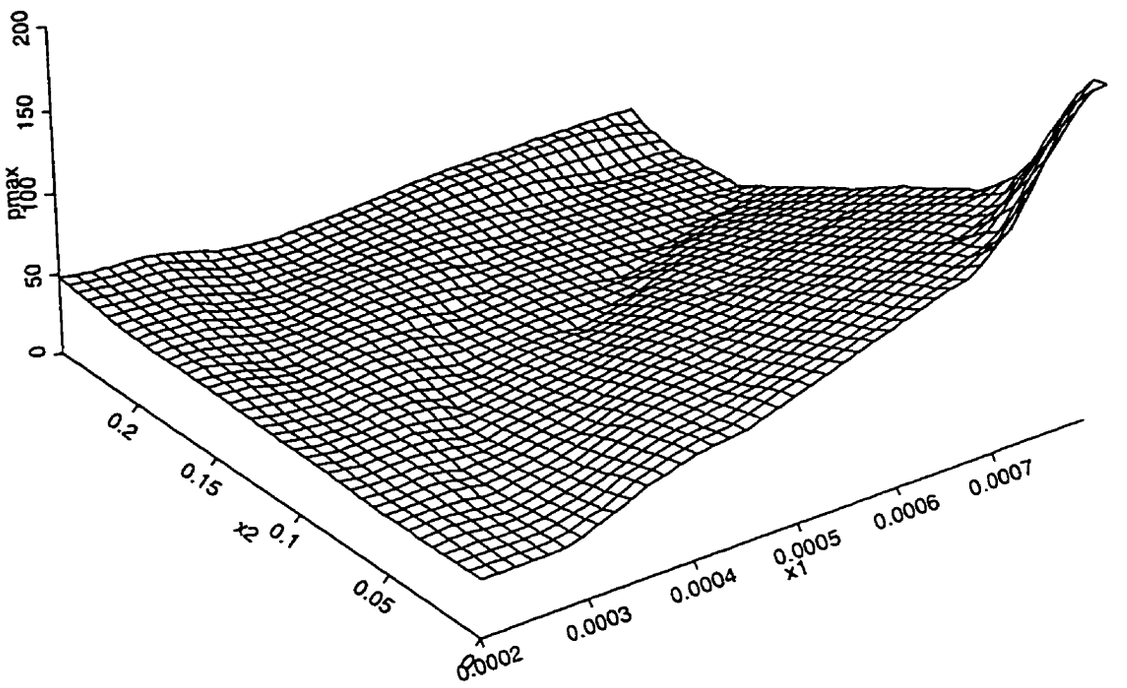
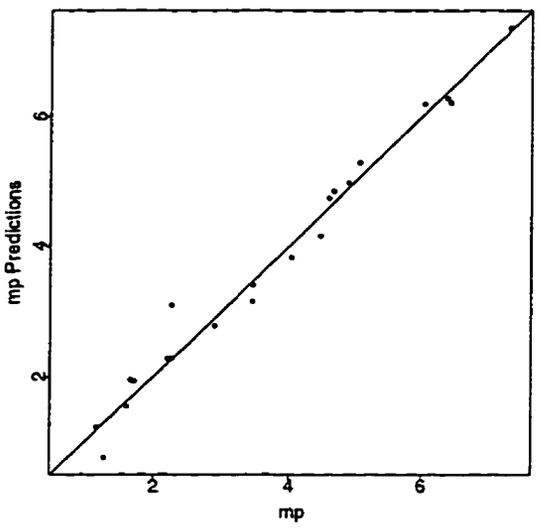


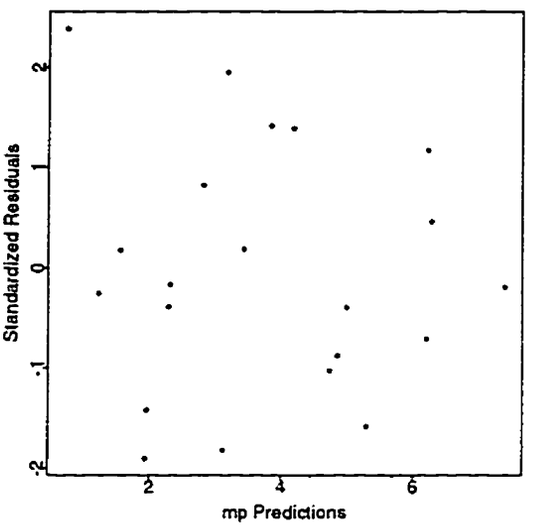
Figure 5.5: Piston Application: Perspective Plot of *pmax*

As we can see from Figures 5.4 and 5.5, the x-space that meets this constraint is fairly flat compared to the slope in the lower right corner. The global (unconstrained) minimum with $pmax = 30.117$ is located at $(0.00035, 0.103)$ outside of the constrained region. The lowest point in the constrained region is (0.00035.0.214) with $pmax = 31.323$, also a near global minimum. The function has a number of further local minima.

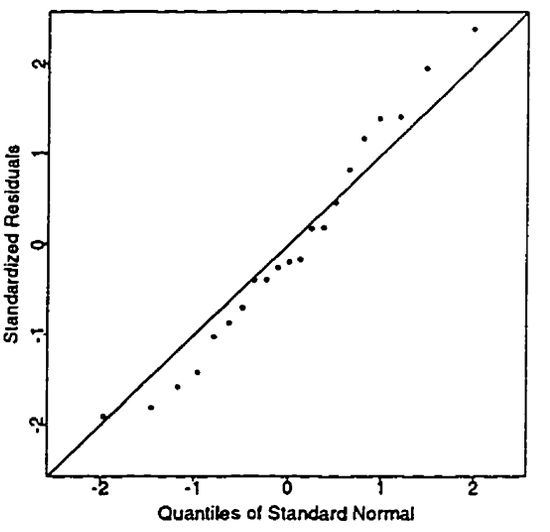
Initially, we evaluate the function again at 21 sites. Diagnostic plots for *mp* and *pmax* are shown in Figures 5.6 and 5.7, respectively. The plots indicate *mp* is well fit. For *pmax*, we can see from Figure 5.7a that a large proportion of points has relatively low function values. While the overall fit is good, the most



(a) Cross-validated Predictions versus True Values

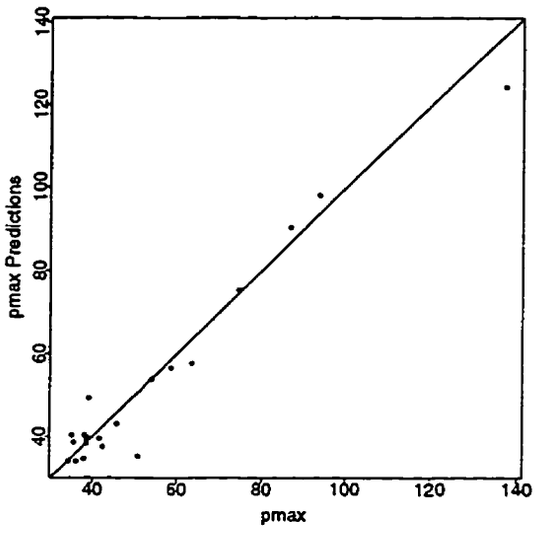


(b) Standardized Cross-validated errors versus Predictions

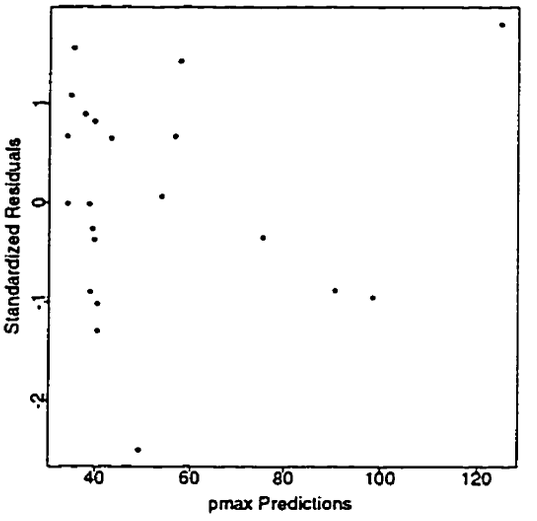


(c) Q-Q Plot of the Cross Validated Residuals

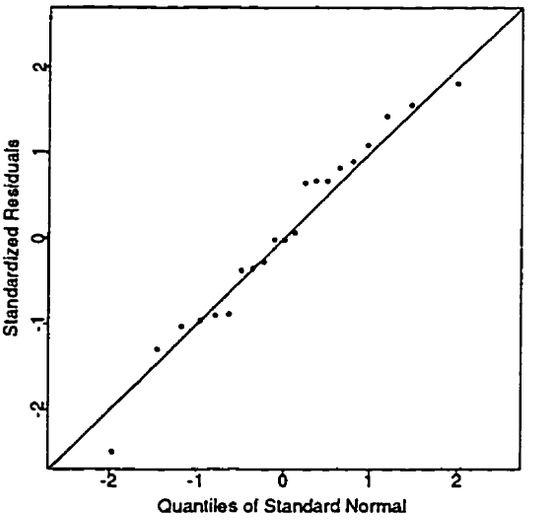
Figure 5.6: Piston Application: Diagnostic Plots for mp



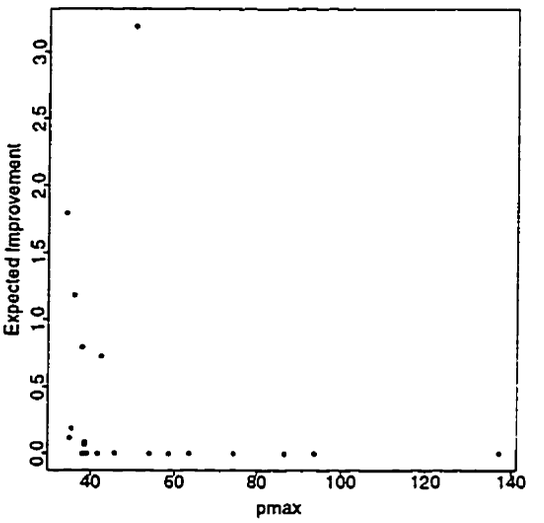
(a) Cross-validated Predictions versus True Values



(b) Standardized Cross-validated errors versus Predictions



(c) Q-Q Plot of the Cross Validated Residuals



(d) Cross-validated Expected Improvement versus True Values

Figure 5.7: Piston Application: Diagnostic Plots for *pmax*

interesting points with low function values are poorly fit. Figure 5.7d confirms this; the cross validated improvements for low function values are somewhat out of order. We tried several transformations for *pmax* none of which improved the diagnostic plots. We therefore proceed without a transformation, but because of potential modeling problems we are inclined to use $g > 1$ for minimization.

For the one-point-at-a-time minimization we use $g = 2$. The criterion for selecting the next sampling site derived from (5.9) is thus $E_{mp}(I^2)\, P(mp < 3)$. The result of the one-point-at-a-time minimization of *pmax* subject to the constraint on *mp* can be seen in Figure 5.8.

The algorithm explores two local minima intensively, and samples a number of points in the lower left corner close to the boundary of the constraint where the response values are also relatively low. The remainder of the function evaluations are fairly wide spread; they can be interpreted as a global search. Virtually no points are sampled outside of the constrained region because the constraint is a relatively simple function and modeled very well.

A total of 89 points were sampled before a stopping criterion with a relative tolerance of .0001 was met. At this point the actual relative tolerance was smaller than 10^{-7}.

The $pmax = 35$ contour line demarks two valleys, which join up in the lower left corner. One of the valleys contains the lowest function value in the constrained region and the other one contains the (unconstrained) global minimum. This second valley, which we will call the southern valley for reference, falls completely outside the constrained region. It is very unfortunate that none of the initial 21 function evaluations fall inside the southern valley. Not surprisingly, the valley is not captured by the model.

We make the minimization problem harder by shifting the constraint from $mp <$

3 to $mp < 3.5$, thus including about half of the southern valley into the constrained minimization region (lengthwise). The new constrained global minimum is located on the boundary of the constrained region near the unconstrained global minimum.

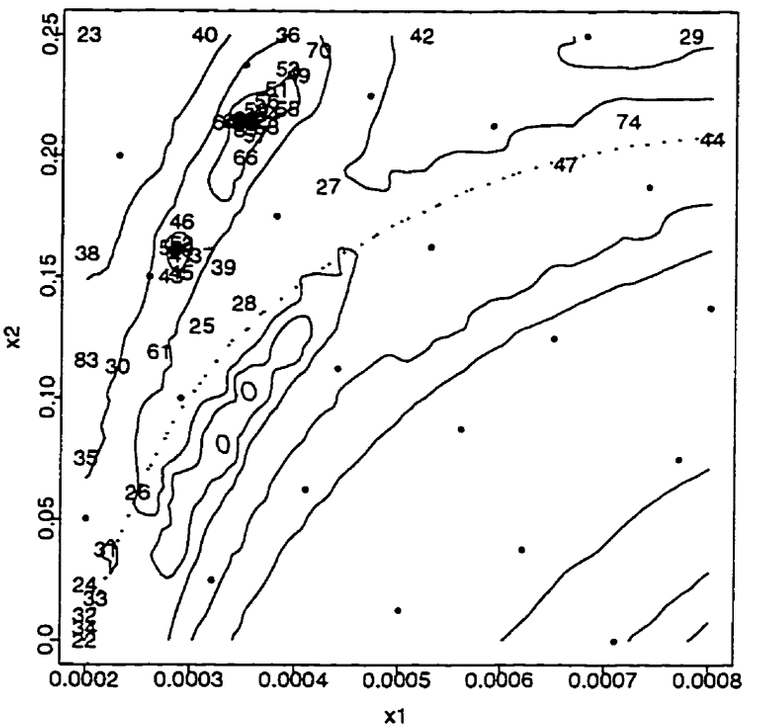


Figure 5.8: Piston Application: Initial Experimental Design (Dots) and Points Introduced by the Sequential Minimization (Case Numbers) for $g = 2$. Contours of *pmax* and $mp = 3$ (dotted). The Dotted Contour Line $mp = 3$ Represents the Boundary of the Constrained Region.

With $g = 2$ the minimization fails to discover the southern valley. A more cautious global search with $g = 5$ can be seen in Figure 5.9. This search explores

the southern valley along the boundaries of the constrained region. While the smallest local minimum in the interior of the constrained region is explored and several local minima along the boundary are established, the constrained global minimum along the boundary of the constrained region is not found.

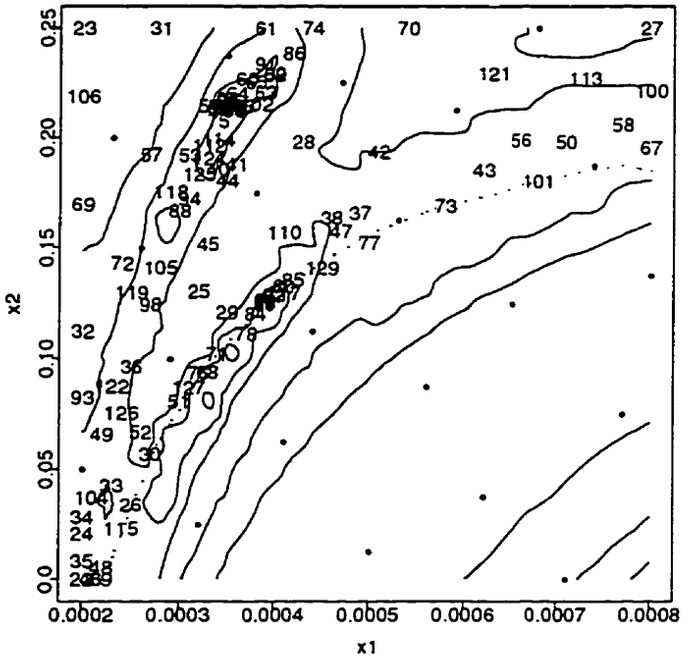


Figure 5.9: Piston Application: Initial Experimental Design (Dots) and Points Introduced by the Sequential Minimization (Case Numbers) for $g = 5$. Contours of $pmax$ and $mp = 3.5$ (dotted). The Latter one (dotted) Denotes the Boundary of the Constrained Region.

The function values of the local minimum in the northern valley ($pmax = 31.323$) and the constrained global minimum along the boundary ($pmax = 31.223$) are very close and the function has been evaluated in the vicinity of both minima. Since for higher g values we aim to search more globally, it is not surprising that

the algorithm fails to distinguish more accurately between those two minima.

The relative stopping criterion of .001 is met after 131 observations. At this point the actual relative tolerance is .003.

Chapter 6

Fast Evaluation of the CDF of the Minimum of N Dependent Variables

6.1 Introduction

In this chapter we develop an algorithm for evaluating the cumulative distribution function for the minimum of N dependent variates, when the mean, covariance, and possible higher order moments are known. Intermediate steps of the algorithm give a sequence of (not necessarily nested) lower and upper bounds on the cumulative probability. This, we show later, can be applied as a stopping criterion for the minimization algorithms in Chapters 4 and 5.

When it is sufficient to determine whether a specified bound on the cumulative probability is met, the algorithm will often be able to terminate in a very small fraction of the time it would be required to compute the exact minimum. Therein

lies the algorithm's principal strength.

For example, consider the first order statistic (or the minimum) of four variables, $Z_1, \ldots, Z_4$, denoted by $Z_{1:4} = \min(Z_1, \ldots, Z_4)$. Suppose one wants to know whether

$$P(Z_{1:4} \leq h) < \alpha \tag{6.1}$$

is true for given h (any real number) and α in $[0, 1]$. We will show that the CDF probability of ordered variables in (6.1) can be rewritten in terms of CDF probabilities of *unordered* variables generated by all possible subsets of the variables $Z_1, \ldots, Z_4$. For example, the CDF probability for the unordered variables Z_1 and Z_3 is $P(Z_1 \leq h, Z_3 \leq h)$. The four variables generate 2^4 possible subsets of variables that have to be evaluated. This chapter explains why it is usually sufficient to evaluate only a small number of these 2^4 subsets in order to determine whether (6.1) is true. In fact, in the context of our applications it appears that only univariate, bivariate and occasionally trivariate CDF probabilities need be evaluated to determine whether (6.1) is true.

This chapter is organized as follows : Section 2 outlines the basic algorithm. Section 3 applies several modifications to the basic algorithm which improve computational speed considerably. Section 4 makes some algorithmic considerations including the choice of a suitable data structure. Section 5 gives an illustrative hypothetical example. In Section 6 we show how a certain first order statistic can be used as a stopping rule for the minimization algorithm presented in Chapter 4. Examples are given.

6.2 A Basic Algorithm

The basic algorithm has three components: First, we show how the cumulative distribution function of the first order statistic (or, equivalently, the minimum) of n random variables from any joint distribution can be computed following Maurer and Margolin (1976). Secondly, we show that by performing the calculations in a certain order, we obtain successively lower and upper bounds on the cumulative probability. Thirdly, we discuss how to compute the cumulative distribution functions of unordered variables required for the implementation of Maurer and Margolin's (1976) method.

6.2.1 Computing the CDF of a first order statistic

Maurer and Margolin (1976) develop a formula for computing the cumulative distribution function of any subset of order statistics from dependent random variables. We state their result here for the special case where the subset consists of only the first order statistic.

Theorem (Maurer and Margolin, 1976) : Let $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_N)$ be a vector of N dependent random variables and let h denote a real number. Then the cumulative distribution function of the first order statistic $Z_{1:N}$ is given as

$$P(Z_{1:N} \leq h) = \sum_{a=1}^{N} (-1)^{a+1} \sum_{j_1 < j_2 < \ldots < j_a} P\left(\cap_{m=1}^{a} (Z_{j_m} \leq h)\right). \tag{6.2}$$

This result is a direct application of the inclusion-exclusion formula, in which events A_i are defined as $A_i = \{Z_i \leq h\}$. In terms of the inclusion-exclusion formula,

$P(Z_{1:N} \leq h)$ can be interpreted as the probability that at least one event of $\{A_i\}$. $i = 1 \ldots n$, is realized.

It is worth noting that this result is completely general, in particular, it does not require exchangeability of the random variables.

Equation (6.2) involves joint cumulative distribution functions of subsets of (unordered) variables. Their evaluation will be considered later.

6.2.2 Upper and Lower Bounds

For notational convenience, denote

$$S_a = \sum_{j_1 < j_2 < \ldots < j_a} P\left(\cap_{m=1}^{a} (Z_{j_m} \leq h)\right).$$

Then (6.2) can be rewritten as

$$P(Z_{1:N} \leq h) = \sum_{a=1}^{N} (-1)^{a+1} S_a. \tag{6.3}$$

Feller (1968, Chapter 4, equation 5.2) gives formula (6.3) except that he uses general events A_i rather than specific events $A_i = \{Z_i < h\}$. Then, if in (6.3) only the first t terms are retained ($1 \leq t < N$), and the remaining ones are dropped, "the error (i.e., true value minus approximation) has the sign of the first omitted term [...], and is smaller in absolute value." (Feller, 1968, Section IV.5(c)). That is,

$$\left| \sum_{i=t+1}^{N} (-1)^i S_i \right| \leq S_{t+1}. \tag{6.4}$$

where $t \in (1, 2, \ldots, N-1)$.

Hence, by omitting terms, we can generate upper and lower bounds on $Z_{1:N}$, depending on whether t is even or odd:

$$\begin{aligned} \sum_{a=1}^{t}(-1)^{a+1}S_a &\leq P(Z_{1:N} \leq h) \qquad \text{t is even} \\ \sum_{a=1}^{t}(-1)^{a+1}S_a &\geq P(Z_{1:N} \leq h) \qquad \text{t is odd} \end{aligned} \tag{6.5}$$

where $1 \leq t \leq N$.

We will illustrate this by means of an example. Suppose the total number of variables is $N = 3$. Then, according to (6.2), the CDF of the 1^{st} order statistic is given as

$$P(Z_{1:3} \leq h) = P(Z_1 \leq h) + P(Z_2 \leq h) + P(Z_3 \leq h) - P(Z_1 \leq h, Z_2 \leq h)$$
$$-P(Z_1 \leq h, Z_3 \leq h) - P(Z_2 \leq h, Z_3 \leq h) + P(Z_1 \leq h, Z_2 \leq h, Z_3 \leq h) = S_1 - S_2 + S_3$$

After the computation of the univariate CDF's we have an upper bound

$$0 \leq P(Z_{1:3}) \leq S_1.$$

The computation of S_2 leads to a lower bound

$$S_1 - S_2 \leq P(Z_{1:3}) \leq S_1.$$

and finally the computation of S_3 leads to the answer $S_1 - S_2 + S_3$ (t odd in (6.5) with equality). If there were more than three variables, the computation of S_3 would lead to a new upper bound replacing the old one, S_4 to a new lower bound, and so forth.

6.2.3 Evaluating the CDF of a Multivariate Distribution

The algorithm described in this chapter only depends on the multivariate distribution of the random variables in that a function calculating the CDF of the multivariate distribution (of unordered variables) has to be supplied.

If a direct evaluation of a multivariate CDF is not possible, it is always possible to calculate the CDF using a Monte Carlo technique. That has the advantage that the variance of the estimate does not depend on the number of dimensions but rather on the sample size used in the simulation. Depending on the distribution at hand, there may be other approaches, too.

In the special case of a multivariate normal distribution, a strategy to evaluate the multiple normal integrals does exist (Schervish, 1984) but is computationally too costly. Deák(1980) proposes a sophisticated modification of the Monte Carlo approach, which is our method of choice. Deák's (1980) approach is described in detail in Appendix F.

It turns out that the algorithm proposed below only uses the evaluation of the CDF of the bivariate and occasionally the trivariate normal distribution, since so far the algorithm has terminated in all cases before any 4-variate or higher normal distribution would have been needed. The CDF of a univariate normal distribution can be evaluated directly (e.g. Press et al., 1992, Chapter 6). A more careful implementation of the algorithm might also incorporate methods for the bivariate and trivariate cases, thus avoiding Monte Carlo techniques completely.

6.3 Reducing the Number of Terms to be Evaluated

For practical purposes, the basic algorithm is still not very useful as it requires an immense amount of computations when the number of dimensions is large. The number of CDF probabilities to be computed for N variables is $2^N - 1$. The number thus grows exponentially as a function of N.

In this section we offer two modifications of the basic algorithm, which drastically reduce the number of probabilities to be evaluated. For later reference we will call them Reduction 1 and Reduction 2.

6.3.1 Reduction 1

For the first reduction we exploit the fact that

$$P(A, B) \leq \min\left(P(A), P(B)\right) \tag{6.7}$$

where A and B are two events. An event may consist of more than one condition, e.g. $A \equiv \{Z_{s_1} < h, \ldots, Z_{s_a} < h\}$, where a is the cardinality of s.

For example, suppose for an arbitrary subset of variables s we find that

$$P(A) = P(Z_{s_1} < h, \ldots, Z_{s_a} < h) < \epsilon,$$

where ϵ is small. Then all CDF probabilities $P(A, B_i)$, where B_i is an arbitrary event, i.e. all CDF probabilities of sets of variables that contain s as a subset, are smaller than ϵ. Provided ϵ was sufficiently small to be considered negligible, none of them needs to be computed.

In the above example, only one set was used to determine whether the subset

need be computed. To fully exploit (6.7), we want to consider all possible subsets, since

$$P(\cap_{i=1}^{n} A_i) \leq \min_{s \in S} P(\cap_{j \in s} A_j) \tag{6.8}$$

where A_i are events, S is the set of all (proper) subsets of the set of variables of interest. Rather than considering all possible subsets S, it suffices to consider all subsets of size $n-1$ because

$$\min_{s \in S} P(\cap_{j \in s} A_j) = \min_{s_{n-1} \in S_{n-1}} P(\cap_{j \in s_{n-1}} A_j) \tag{6.9}$$

where S_{n-1} is the set of all subsets of size $n-1$.

We will now show that (6.9) is true. Since $S_{n-1} \subseteq S$, the left hand side of (6.9) is smaller or equal to the right hand side. For an arbitrary $s^* \in S$ there exists $s^*_{n-1} \in S_{n-1}$ such that $s^* \subseteq s^*_{n-1}$. Hence

$$P(\cap_{j \in s^*} A_j) \geq P(\cap_{i \in s^*_{n-1}} A_i)$$

Hence the left hand side of (6.9) is greater or equal to the right hand side. Therefore (6.9) holds.

6.3.2 Reduction 2

Suppose one is interested in knowing whether $P(Z_{1:N} \leq h) < \alpha$ is true, rather than evaluating $P(Z_{1:N} \leq h)$ exactly. Roughly speaking, the second reduction exploits the fact that in order to determine whether $P(Z_{1:N} \leq h) < \alpha$ is true it may suffice to compute $P(Z_{1:M} \leq h)$, where $M < N$ and $Z_1, \ldots, Z_M$ form a subset of $Z_1, \ldots, Z_N$.

For joint probabilities of a number of events we were able to obtain an inequal-

ity (6.7) by noticing that as we increase the number of events the joint probability cannot increase. The operators "Joint Probability" and "Minimum" behave in the same way: For the minimum of a number of variables as we increase the number of variables the minimum cannot increase either. Analogously to (6.7) we have

$$\min(S_M, S_{N\setminus M}) = \min\left(\min(S_M), \min(S_{N\setminus M})\right) \tag{6.10}$$

where S_M and $S_{N\setminus M}$ represent sets of variables and $S_N = S_M + S_{N\setminus M}$. Due to idiosyncrasies of the operator "Minimum" in (6.10) equality is attained when the two inequalities $\min(S_N) \leq \min(S_M)$ and $\min(S_N) \leq \min(S_{N\setminus M})$ are combined.

Moreover, the CDF probabilities for the minimum at a given h cannot *decrease*:

$$P\left(\min_{i\in S_N}(Z_i) \leq h\right) \geq \min\left[P\left(\min_{i\in S_M}(Z_i) \leq h\right), P\left(\min_{i\in S_{N\setminus M}}(Z_i) \leq h\right)\right]. \tag{6.11}$$

We rewrite (6.11) for ease of notation with the minimum replaced by the first order statistic and drop the second argument of the minimum on the right hand side:

$$P(Z_{1:N} \leq h) \geq P(Z_{1:M} \leq h). \tag{6.12}$$

Suppose we knew whether $P(Z_{1:i} \leq h) > \alpha$ is true for a subset of i variables. Then this knowledge may bear relevant information about whether $P(Z_{1:N} \leq h) > \alpha$ is true:

$$P(Z_{1:i} \leq h) > \alpha, \quad \text{hence } P(Z_{1:N} \leq h) > \alpha$$
$$P(Z_{1:i} \leq h) < \alpha, \quad \text{no relevant information},$$

where $Z_{1:i}$ is the first order statistic of a set of i variables nested in the set of N

variables. If knowledge about $P(Z_{1:i} \leq h) > \alpha$ can be obtained before $P(Z_{1:N} \leq h)$ is fully evaluated, then the remaining calculations may no longer be necessary. If it is possible to gain that knowledge at no additional cost then this is useful.

As will be shown later, one can arrange the order in which the individual components of $P(Z_{1:N} \leq h)$ are calculated such that whether $P(Z_{1:i} \leq h) < \alpha$ is true or not is known before all individual components of $P(Z_{1:N} \leq h)$ are evaluated. Moreover, it is possible to successively determine whether $P(Z_{1:i} \leq h) > \alpha$ is true for *all* variables $i = 1 \ldots N$. While this does put constraints in the order of evaluation of the individual components, it also comes at no additional computational cost except for some book keeping.

A situation in which we are interested in knowing whether $P(Z_{1:N} \leq h) < \alpha$ is true or not, might arise, for example, if we are interested in whether a critical value or significance level is met. If we perform a sequence of significance tests until the significance level is met, then $P(Z_{1:N} \leq h) > \alpha$ holds except for the very last test, when the significance level is actually met. That means that the above reduction is going to be useful every single time except the last time.

Not only do Reductions 1 and 2 reduce the number of CDF probabilities to be calculated. The probabilities avoided are those for higher dimensions, which are more computationally intensive.

6.4 Algorithmic Considerations

In this section we motivate the choice of a tree-based data structure. We explain how the data structure lends itself to implementation of the two reductions. An algorithm for the computation of the first order statistic is outlined.

N variables generate 2^N different subsets of variables, including the empty set.

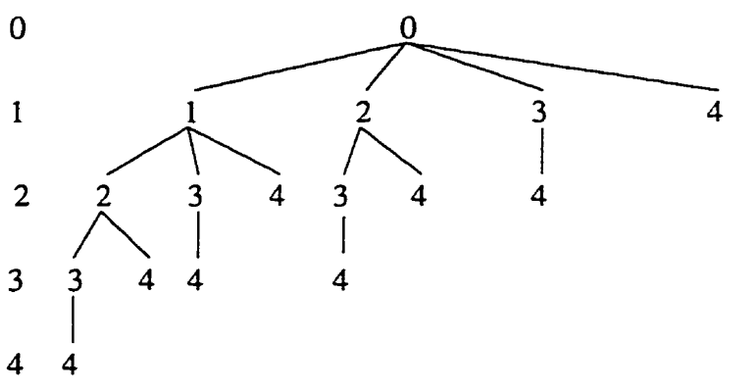


Figure 6.1: Generating All Possible Subsets of Four Variables in a Tree Structure

We arrange all subsets into a tree, whose root branches out into all N univariate CDF's, the univariate branch's generate all bivariate CDF's, these in turn all trivariate CDF's and so forth.

Figure 6.1 depicts a tree for $N = 4$ variables. The root at level 0 is denoted by a 0, level 1 contains all the univariate CDF's, level 2 below contains all the bivariate CDF's, and so forth. At any given level i, the i variables corresponding to a particular branch are given by that branch's number, and all numbers of that branch's parent, grandparent, great-grandparent etc.

There are several reasons why we choose a tree structure over, for example, a list. These reasons fall into two categories : memory requirements and cpu-time.

- (Memory) For any joint probability of i variables, only the label of the last variable has to be stored. For example, in Figure 6.1, for the joint probability for the variables 1 through 4 only the label '4' is depicted in level 4.

- (Memory) Not all joint probabilities have to be created (that is, stored). If according to reduction 1 or 2 some probabilities are not needed, their indices do not have to even be created in the tree. This is further illustrated in the

examples.

- (CPU-time) The ordering of a tree structure makes it very easy and fast to exploit part of reduction 1 (checking of *one* subset of size $n-1$). In a list structure, extensive searches have to be done to implement reduction 1.
- (CPU-time) Any particular element can be accessed fast, which is relevant for the implementation of the remainder of reduction 1 (checking of the remaining $n-1$ subsets of size $n-1$). A joint probability of i variables can be accessed in i steps. Assuming that a search mechanism is necessary for each level in a list, up to N choose i elements have to be searched in level i.

It is easy to lose the basic structure of an algorithm when too many details obstruct the view. For this reason only a bare-bones algorithm is given in Figure 6.2.

6.5 An Example Based on Hypothetical Data

We base this example on 4 variables. The tree structure for this situation is depicted in Figure 6.1. Assume that $\alpha = .05$ and we want to decide whether $P(Z_{1:4} \leq h) > \alpha$.

Furthermore, suppose that $P(Z_1 < h) < \epsilon$, $P(Z_2 < h) = .3$, $P(Z_3 < h) < \epsilon$. and $P(Z_4 < h) = .5$. Also, suppose $S_2 = .25$.

Basic Algorithm

Each level generates a new bound. All the univariate CDF's on level 1 constitute S_1 and hence generate the first upper bound. All the bivariate CDF's on level 2 generate S_2, and $S_1 - S_2$ forms the first lower bound. The trivariate CDF's generate a new upper bound, and so forth.

```
i = 0                          initialize the number of variables
DO                             construct tree with i variables
   i = i + 1                   increase the number of variables
   j = 0                       initialize the current level (of the tree)
   DO
      j = j + 1                increase the current level under investigation
      calculate S_j            compute probabilities on level j (descendants
                               from i variables)
      adjust lower/upper bounds for i variables by S_j
   UNTIL (lower bound > α OR upper bound< α
   OR full tree for i variables searched)
UNTIL (i = N OR lower bound > α )
IF (lower bound > α)        THEN 1st order statistic > α
IF (upper bound < α)        THEN 1st order statistic < α
IF (i = N AND N is odd)     THEN 1st order statistic = lower bound
IF (i = N AND N is even)    THEN 1st order statistic = upper bound
```

Figure 6.2: Algorithm for Computing the First Order Statistic

The sum S_1 can be calculated to be $S_1 \simeq .8$. Then $S_1 - S_2 \simeq 0.55$. Since $S_1 - S_2$ constitutes a lower bound, and the lower bound is greater $\alpha = .05$. no further calculations need to be done. That is the 4 trivariate and the 4-variate CDF do not need to be calculated.

Basic Algorithm with Reduction 1

If at any given node the corresponding CDF is smaller than ϵ, all descendents of that branch will have a CDF smaller than ϵ and will thus all be ignored.

All four univariate CDF's have to be evaluated as well as the bivariate CDF corresponding to variables $2, 4$. The remaining $1, 1$ CDF's don't have to be evaluated. All but one of them are either descendants of variable 1 or 3 (Reduction 1), or they are in level 3 or higher (level 2 is a bound), or both. The CDF corresponding to $2, 3$ does not have to be evaluated because it contains 3 as a subset, even though

2, 3 is not a descendant of 3.

Basic Algorithm with both Reductions

Reduction 2 states that rather than computing all CDF's for all N variables, it may suffice to look all CDF's for a smaller number of variables. Hence we first look at all CDF's corresponding to the first variable, then at the ones for the first 2 variables, and so forth.

We start out just considering variable 1. Since $P(Z_1 < h) < \epsilon$, we need to increase the set of variables. We consider variables 1 and 2. We only need to compute $P(Z_2 < h)$, since $P(Z_1 < h)$ is known already and the bivariate probability need not be calculated. Since $P(Z_{1:2} \leq h)) = .3 > \alpha$, we know that $P(Z_{1:4} \leq h) > \alpha$, too. Hence, after evaluating only the two univariate probabilities for variables 1 and 2, we are done.

Note that if any one of the univariate probabilities is greater than α we know that $P(Z_{1:4} \leq h) > \alpha$ holds. This suggests reordering the variables in order of decreasing univariate CDF probability. The ordering of N variables however would come at an additional cost proportional to $N \log(N)$. Incidentally, looking at univariate probabilities, the reordering of variables, or, more generally, the idea of splitting a set of variables into subsets in different ways is related to fully exploiting (6.11) (rather than dropping one of its arguments) and is also analogous to (6.8) in Reduction 1.

6.6 Application: A Stopping Rule Based on a First Order Statistic

In this section we argue that a certain first order statistic can be used as a stopping rule for the minimization algorithm presented in Chapter 4.

In Chapter 4 we based the stopping rule on the expected improvement. We would stop the algorithm when one or both of the two following criteria are met:

$$\max_{\mathbf{x}}(E(I)) < \text{(Absolute Tolerance)}$$
$$\frac{\max_{\mathbf{x}}(E(I))}{|f^n_{min}|} < \text{(Relative Tolerance)}$$

where $\max_{\mathbf{x}}(E(I))$ is the maximum expected improvement at any given step.

This stopping rule has some undesirable properties. First, we cannot guarantee that the maximal expected improvement has been found. Therefore we may base the stopping criterion on a smaller number than we should, which would tend to lead to a premature termination of the algorithm.

Second, the fact that we always use the *maximal* expected improvement rather than expected improvement at a given point constitutes a multiple comparison problem. On average, the observed improvement is much smaller than the (maximal) expected improvement. This would then lead to a too conservative stopping rule, that is the stopping rule would terminate too late. The tradeoff between these two issues is unclear. In either case, both premature and late termination of the algorithm are undesirable.

Third, knowing that the improvement over the current minimum f^n_{min} is *on average* smaller than a tolerance value may not be very satisfactory for a particular stopping problem. The decision to stop is made only once and may require a more conservative approach.

Rather than basing the stopping rule on the expected one-step-ahead reduction in the target function, it may be conceptually more appropriate to base a stopping rule on the difference between the current minimum f^n_{min} and the global minimum. Unfortunately the global minimum is unknown and it is not possible to obtain a

lower bound on the global minimum without making additional assumptions such as the existence of a Lipschitz condition with known Lipschitz constant (see also Betro, 1991).

Alternatively, one might ask : What is the probability that the global minimum and f^n_{min} are no farther apart than a tolerance value δ for this particular problem? Our alternate Stopping Rule proposes to stop when this probability is very small, i.e. when

$$P(\text{Global Minimum} < f^n_{min} - \delta) < p_{\text{crit}}$$

where δ is the tolerance value with $\delta \geq 0$, and p_{crit} in $[0, 1]$ is a critical value.

It is not clear how to calculate the distribution of the minimum over a continuous region. Therefore, we simplify the problem by discretizing it: we consider a large number of points N that fill the continuous space well. The distribution of the global minimum then becomes the distribution of the first order statistic of N points.

In other words, the decision to stop sampling is made when

$$F_{Y_{1:N}}(f^n_{min} - \delta) < p_{\text{crit}}$$

where $Y_{1:N}$ denotes the first order statistic of $Y_1, Y_2, \ldots, Y_N$. Typically, $p_{\text{crit}} = 0.01$.

Theoretically, it makes little sense to set $\delta = 0$. Unless f^n_{min} happens to be a local minimum in the modelled function, one is guaranteed to beat f^n_{min} in its immediate vicinity. Practically, unless some of the discrete points are extremely close to one another, the choice of $\delta = 0$ is fine.

The problem of computing the first order statistic of dependent random variables with known moments has been considered in this chapter. For this application we are interested in the special case of the first order statistic of a multivariate

normal distribution. Recall that all points $\mathbf{Y}$ are assumed to be distributed as a multivariate normal distribution

$$\mathbf{Y} \sim MVN(\hat{\mathbf{Y}}, \Sigma).$$

We calculate the first order statistic of a regular grid with 10 points in each dimension (i.e., for each explanatory variable). The 10 points are equally spaced between the lower and upper bound in each dimension, and include the bounds themselves. Depending on the number of explanatory variables the task to compute the first order statistic of the grid points can be quite formidable. When there are two explanatory variables, the grid contains 100 points, six explanatory variables already leads to a grid with 1 million points.

The time that is needed to compute first order statistics depends very much on the data, and specifically on whether higher order CDF probabilities need to be calculated or not. We have calculated first order statistics using the algorithm outlined in this chapter with up to ten thousand variables. That means that from about four dimensions onwards it becomes too time consuming to compute the first order statistic every time.

In six dimensions with 1 million points, if the stopping rule is met we still need to evaluate all 1 million univariate CDF probabilities. That in itself is too time consuming. Further reductions therefore must contain a reduction in the number of grid points. For example, it may be possible to determine a priori areas in the d dimensional space which are unlikely to lead to an improvement and then to disregard grid points that fall into these areas.

6.6.1 Examples: Branin and Goldstein-Price function

We now give examples for the use of the first order statistic as a stopping criterion. The first order statistic is calculated each time after a new point is sampled. The stopping criterion is met when $P(Z_{1:N} \leq f_{min}^n) < \alpha$. In fact, we require that the stopping criterion be met three times in sequence to be on the safe side.

For both the Branin and the Goldstein-Price function we use a 10×10 Grid, i.e. $N = 100$, $\epsilon = 10^{-10}$, $\alpha = .01$, $\delta = 0$, and the Monte Carlo sample size for bivariate and higher order CDF probabilities is 1000.

The result for the Branin function can be seen in Table 6.1. Without the reductions and bounds introduced in this chapter, the computation of each order statistic would require the evaluation of $2^{100} - 1$ CDF probabilities. It turns out that only univariate probabilities need to be evaluated. For observations 21 through 25 a single univariate probability exceed α, so that the stopping criterion could not be met. From observation 26 onwards the stopping criterion was met. Only a very small number of the univariate CDF probabilites exceeded ϵ.

The algorithm terminates after sampling 28 points, compared to 33 points with the tolerance stopping criterion in Table 4.1. It is difficult to compare the two stopping rules in terms of number of function evaluations. For the first order statistic a finer grid may increase the function evaluations. Likewise, a greater tolerance value would decrease the number of function evaluations for the tolerance criterion.

Results for the Goldstein-Price function are presented in Table 6.2. The stopping criterion is reached after a total of 99 points as opposed to 106 with the tolerance stopping rule in Table 4.1. The first few order statistics are evaluated very fast since in each case there is a single univariate CDF probability greater than α. All but one of the remaining order statistics are evaluated in under five minutes of

Sample Size	univ. count	univ.$> \epsilon$ count	univ.$> \alpha$	CPU-time (sec)	f^n_{min}
21	17	2	*	.5	5.246
22	10	2	*	.4	3.586
23	18	4	*	.5	2.337
24	52	3	*	.8	0.582
25	52	3	*	.9	0.513
26	100	2		1.3	0.419
27	100	2		1.4	0.430
28	100	1		1.5	0.400

Table 6.1: The First Order Statistic Stopping Criterion Applied to the Branin Function: "univ. count" is the Number of Univariate CDF Probabilities Evaluated; "univ. $> \epsilon$ count" is the Number of Univariate CDF Probabilities Evaluated that are Greater than 10^{-10}; "univ. $> \alpha$" is an Indicator Whether a Single Univariate CDF Probability is Greater than α. The Full Grid has $10 \times 10 = 100$ Points. α is 0.01, ϵ is Set to 10^{-10}, and $\delta = 0$.

CPU-time. One order statistic (at sample size 95) requires the evaluation of nearly 16000 trivariate CDF probabilities and takes 2 hours of CPU-time. The aim of a more careful implementation of the algorithm should be to avoid if possible the calculation of trivariate and higher order probabilities.

It is interesting to see how the number of univariate CDF probabilities required jumps from 55 at sample size 36 back to 2 at sample size 37. This is a consequence of changes in the mle-estimates of the parameters. It happens that with the new mle-estimates one of the first two points on the grid has a univariate CDF probability greater than α.

Sample Size	univ. count	biv. count	triv. count	univ.> ϵ count	univ. > α	CPU-time sec	f^n_{min}
21	33			10	*	.2	4.3653
22	43			13	*	.2	4.3653
23	43			21	*	.2	3.5031
24	43			21	*	.2	3.4610
25	43			12	*	.2	3.4610
26	43			13	*	.3	3.4610
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
34	54	105		15		15.3	3.4381
35	55	120		16		17.5	3.4287
36	55	120		16		17.3	3.4287
37	2			2	*	0.3	3.4287
38	7	21		7		3.3	2.3954
39	9	36		9		5.5	2.3954
40	9	36		9		5.6	2.3954
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
79	11	55		11		10.0	1.0989
80	12	65	165	12		75.6	1.0989
81	12	66		12		12.2	1.0989
82	17	136		17		24.2	1.0989
83	17	135	560	17		243.6	1.0989
84	16	120		16		21.1	1.0989
⋮	⋮		⋮	⋮	⋮	⋮	⋮
94	51	780		40		148.8	1.0989
95	70	1539	15950	56		7192.2	1.0989
96	70	1540		56		296.9	1.0989
97	100			15		3.8	1.0989
98	100			15		4.4	1.0989
99	100			15		4.1	1.0989

Table 6.2: The First Order Statistic Stopping Criterion Applied to the Goldstein-Price Function: "univ. count" is the Number of Univariate CDF Probabilities Evaluated; "biv. count" is the Number of Bivariate CDF Probabilities Evaluated; "triv. count" is the Number of Trivariate CDF Probabilities Evaluated; "univ. $> \epsilon$ count" is the Number of Univariate CDF Probabilities Evaluated that are Greater than 10^{-10}; "univ. $> \alpha$" is an Indicator Whether a Single Univariate CDF Probability is Greater than α. The Full Grid has $10 \times 10 = 100$ Points. α is 0.01, and ϵ i s Set to 10^{-10}.

Chapter 7

Concluding Remarks

In this thesis we have been considering the problem of finding the global optimum of expensive-to-compute computer models with few function evaluations. We have achieved this goal at the cost of a considerable computational burden. Optimization starts taking a long time when several hundred observations are needed until the stopping criterion is met. The three major time consuming factors are in decreasing order: the computation of the maximal expected improvement over the range of x, the maximum likelihood estimation of the parameters, and the computation of the first order statistic when used as stopping criterion.

Having said that, for most industrial applications a large number of runs is probably unrealistic. Also, the goal of many industrial applications is not exact optimization but rather a good design that is noticeably better than the previous one. For example, for one application in chemical engineering we are currently looking at, the engineer is interested in reviving a former experiment if, based on previous data, it looks as though we can find a design site with a response value several percent higher than the best previously known one.

The computation of the first order statistic of N dependent variables has been

greatly speeded up using the two reductions and the bounds. However, in order to use it successfully as a stopping criterion and to use it in higher dimensions, further time reductions need to follow. Given all the knowledge about the function gained through the computation of the first order statistic, one wonders whether it might be possible to use this knowledge in deciding on the next sampling site, thus replacing the expected improvement criterion.

The identification of nonlinearities and interactions in Chapter 3 is useful in a broader context than just global optimization. It would be nice to be able to give explicit rules as to how to use standard errors in the identification of key features. This is likely a difficult task.

Theorems 1 and 2, despite their inconspicuousness, are quite powerful due to their generality. They include many special cases, e.g., interactions.

Appendix A

On Programming

Here we give a brief overview over the major components that a computer program must contain for design, analysis, and minimization of computer models. The components are :

- Latin-Hypercube Design

  The initial set of points are designed by a Latin-Hypercube Design.

- Parameter Estimation

  The parameters $\mathbf{p}$, θ, β, and σ^2 are estimated via maximum likelihood.

- BLUP and MSE

  The best linear unbiased estimator and the mean squared error are computed at a new input $\mathbf{x}$.

- Cross-Validation

  The cross validated BLUP and MSE are computed for all design points.

- Visualization

  Main effects and joint effects (or interactions) are computed.

- Minimization

  This includes both minimization in stages as well as point-by-point minimization.

- (optional) First Order Statistic of a Multivariate Normal Distribution

  This is only needed when the first order statistic is used as a stopping criterion for the minimization.

SPACE (Stochastic Processes Analysis of Computer Experiments) by Matthias Schonlau has implemented these steps with the exception of the Latin Hypercube Design. The program contains more than 10000 lines of C code and in line comments.

Appendix B

Addendum to Section 3.3

Here we prove Theorems 1 and 2 from Section 3.3 and give an example of how they can be applied to estimating effects such as main effects and interactions.

Theorem 1: The best linear unbiased predictor (BLUP) of $\sum_i b_i Y_i$ is $\sum_i b_i \hat{Y}(\mathbf{x}_i)$.

Proof: Let $\bar{Y}$ denote $\sum_i b_i Y(\mathbf{x}_i)$ and $\bar{Z}$ denote $\sum_i b_i Z(\mathbf{x}_i)$. As before in Section 3.3 we define $\mathbf{b} = (b_1, b_2, \ldots, b_m)$, $\bar{\mathbf{f}} = \sum_i b_i \mathbf{f}_{x_i}$, and $\bar{\mathbf{r}} = \sum_i b_i \mathbf{r}_{x_i}$.

For any linear predictor $\mathbf{c}^t\mathbf{y}$ of $\bar{Y}$ the mean squared error of prediction is :

$$\begin{aligned} E(\mathbf{c}^t\mathbf{y} - \bar{Y})^2 &= E\left[\mathbf{c}^t\mathbf{y}\mathbf{y}^t\mathbf{c} + \bar{Y}^2 - 2\mathbf{c}^t\mathbf{y}\bar{Y}\right] \\ &= E\left[\mathbf{c}^t(\mathbf{F}\boldsymbol{\beta} + \mathbf{z})(\mathbf{F}\boldsymbol{\beta} + \mathbf{z})^t\mathbf{c} + (\bar{\mathbf{f}}^t\,\boldsymbol{\beta} + \bar{Z})^2\right. \\ &\quad \left. -2\mathbf{c}^t(\mathbf{F}\boldsymbol{\beta} + \mathbf{z})(\bar{\mathbf{f}}^t\boldsymbol{\beta} + \bar{Z})\right] \\ &= (\mathbf{c}^t\mathbf{F}\boldsymbol{\beta} - \bar{\mathbf{f}}^t\boldsymbol{\beta})^2 + \mathbf{c}^t\sigma_z^2\mathbf{R}\mathbf{c} + \sigma_z^2\mathbf{b}^t\mathbf{R}\mathbf{b} - 2\mathbf{c}^t\sigma_z^2\bar{\mathbf{r}} \\ &= \mathbf{c}^t\sigma_z^2\mathbf{R}\mathbf{c} + \sigma_z^2\mathbf{b}^t\mathbf{R}\mathbf{b} - 2\mathbf{c}^t\sigma_z^2\bar{\mathbf{r}} \quad . \end{aligned} \tag{B.1}$$

The last equation follows from the unbiasedness constraint $\mathbf{F}^t\,\mathbf{c} = \bar{\mathbf{f}}$.

Introducing k Lagrange multipliers $\boldsymbol{\lambda}$ for the k equations $\mathbf{F}^t\mathbf{c} = \bar{\mathbf{f}}$ and taking

the derivative with respect to $\mathbf{c}$ in (B.1) yields

$$\sigma_z^2 \mathbf{R}\mathbf{c} - \sigma_z^2 \bar{\mathbf{r}} - \mathbf{F}\boldsymbol{\lambda} = \mathbf{0} \quad .$$

Together with the unbiasedness constraint we have a system of two sets of equations in the two unknown vectors $\mathbf{c}$ and $\boldsymbol{\lambda}$:

$$\sigma_z^2 \mathbf{R}\mathbf{c} - \mathbf{F}\boldsymbol{\lambda} = \sigma_z^2 \bar{\mathbf{r}} \tag{B.2}$$

$$\mathbf{F}^t \mathbf{c} = \bar{\mathbf{f}} \quad . \tag{B.3}$$

This can be solved for $\mathbf{c}$ to yield the best linear unbiased predictor:

$$\hat{y}(\mathbf{x}) = \mathbf{c}^t \mathbf{y} = (\bar{\mathbf{f}}^t, \bar{\mathbf{r}}^t) \begin{pmatrix} \mathbf{0} & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{y} \end{pmatrix} \quad . \tag{B.4}$$

This can also be written as

$$\widehat{\sum_i b_i Y_i} = \bar{\mathbf{f}}^t \hat{\boldsymbol{\beta}} + \bar{\mathbf{r}}^t \mathbf{R}^{-1} (\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}}) = \sum_i b_i \hat{Y}(\mathbf{x}) \, , \tag{B.5}$$

where $\hat{\boldsymbol{\beta}} = (\mathbf{F}^t \mathbf{R}^{-1} \mathbf{F})^{-1} \mathbf{F}^t \mathbf{R}^{-1} \mathbf{y}$ is the generalized least squares estimator of $\boldsymbol{\beta}$. □

Theorem 2: The Mean Squared Error of $\sum b_i \hat{Y}_i$ is

$$\text{MSE}(\sum b_i \hat{Y}_i) = \sigma_z^2 \left[\mathbf{b}^t \mathbf{R} \mathbf{b} - \begin{pmatrix} \bar{\mathbf{f}} \\ \bar{\mathbf{r}} \end{pmatrix}^t \begin{pmatrix} \mathbf{0} & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} \bar{\mathbf{f}} \\ \bar{\mathbf{r}} \end{pmatrix} \right]. \tag{B.6}$$

Proof: Using (B.2) first and then (B.3), equation (B.1) can be written as :

$$\begin{aligned} E(\mathbf{c}^t \mathbf{y} - \bar{Y})^2 &= \mathbf{c}^t \sigma_z^2 \mathbf{R} \mathbf{c} + \sigma_z^2 \mathbf{b}^t \mathbf{R} \mathbf{b} - 2\mathbf{c}^t \sigma_z^2 \bar{\mathbf{r}} \\ &= \sigma_z^2 \left(\mathbf{c}^t (\bar{\mathbf{r}} - \mathbf{F}\boldsymbol{\lambda}') + \mathbf{b}^t \mathbf{R} \mathbf{b} - 2\mathbf{c}^t \bar{\mathbf{r}} \right) \\ &= \sigma_z^2 \left(\bar{\mathbf{f}}^t \boldsymbol{\lambda}' + \mathbf{b}^t \mathbf{R} \mathbf{b} - \mathbf{c}^t \bar{\mathbf{r}} \right) \end{aligned} \tag{B.7}$$

Working backwards and using (B.2) and (B.3) simultaneously we have:

$$\begin{aligned} &\sigma_z^2 \left[\mathbf{b}^t \mathbf{R} \mathbf{b} - \begin{pmatrix} \bar{\mathbf{f}} \\ \bar{\mathbf{r}} \end{pmatrix}^t \begin{pmatrix} \mathbf{0} & \mathbf{F}^t \\ \mathbf{F} & \mathbf{R} \end{pmatrix}^{-1} \begin{pmatrix} \bar{\mathbf{f}} \\ \bar{\mathbf{r}} \end{pmatrix} \right] \\ &= \sigma_z^2 \left[\mathbf{b}^t \mathbf{R} \mathbf{b} - \begin{pmatrix} \bar{\mathbf{f}} \\ \bar{\mathbf{r}} \end{pmatrix}^t \begin{pmatrix} \boldsymbol{\lambda}' \\ \mathbf{c} \end{pmatrix}^t \right] = \sigma_z^2 \left[\mathbf{b}^t \mathbf{R} \mathbf{b} - \bar{\mathbf{f}}^t \boldsymbol{\lambda}' - \mathbf{c}^t \bar{\mathbf{r}} \right] \end{aligned} \tag{B.8}$$

which is the same as (B.7). □

The expressions in equation (B.6) are easy to evaluate if

1. the correlation function $R(\mathbf{x}_1, \mathbf{x}_2)$ is a product of correlations in each x variable,

$$R(\mathbf{x}_1, \mathbf{x}_2) = \prod_j R_j(x_j^{(1)}, x_j^{(2)}),$$

where $x_j^{(1)}$ denotes the value of the j^{th} x-variable for point $\mathbf{x}_1$, and

2. the points representing the space of the variables integrated out, $\mathbf{x}_{\text{out}}^{(1)}, \ldots, \mathbf{x}_{\text{out}}^{(m)}$, are a grid. Without loss of generality, suppose $\mathbf{x}_{\text{out}}$ is the first q of the x variables, $x_1, \ldots, x_q$, and the grid of $\mathbf{x}_{\text{out}}$ values is

$$\{x_1^{(1)}, \ldots, x_1^{(m_1)}\} \otimes \ldots \otimes \{x_q^{(1)}, \ldots, x_q^{(m_q)}\},$$

where $x_j^{(i)}$ is the i^{th} grid value for variable x_j, and $\prod_{j=1}^{d} m_j = m$.

For example, if these conditions hold, $\mathbf{b}^t\mathbf{R}\mathbf{b}$ in (B.6) becomes

$$\begin{aligned} \mathbf{b}^t\mathbf{R}\mathbf{b} &= \sum_{i,i'} b_i b_{i'} R\left((\mathbf{x}_{\text{effect}}, \mathbf{x}_{\text{out}}^{(i)}), (\mathbf{x}_{\text{effect}}, \mathbf{x}_{\text{out}}^{(i')})\right) \\ &= R(\mathbf{x}_{\text{effect}}, \mathbf{x}_{\text{effect}}) \sum_{i,i'} b_i b_{i'} R(\mathbf{x}_{\text{out}}^{(i)}, \mathbf{x}_{\text{out}}^{(i')}) . \end{aligned}$$

Since $R(\mathbf{x}_{\text{effect}}, \mathbf{x}_{\text{effect}}) = 1$ this gives

$$\mathbf{b}^t\mathbf{R}\mathbf{b} = \sum_{i,i'} b_i b_{i'} R(x_1^{(i)}, x_1^{(i')}) \cdots R(x_q^{(i)}, x_q^{(i')}) . \tag{B.9}$$

In the case of main effects and overall effects of two or more variables, $b_i = 1/m$, i.e., constant, and the sum of products in (B.9) can be written as a product of sums.

The computation of $\bar{\mathbf{r}}$ is similar to (B.9). Because we restrict the terms in $f(x)$ to be polynomials $x_1^{a_1} \ldots x_d^{a_d}$, i.e., a product in each of the x variables, the weighted averages $\bar{\mathbf{f}}$ are also simple to compute.

For a simple example, Table B.1 gives the coefficients b_i, $i = 1 \ldots m$, needed for Theorems 1 and 2 for various effects. The example is based on three variables with 2 levels each, i.e. $m = 8$. Each row contains the coefficients for one level of an

x_1	x_2	x_3	ME x_1		ME x_2		JE x_1, x_2				IE x_1, x_2			
			0	1	0	1	00	01	10	11	00	01	10	11
0	0	0	.25		.25		.5				.125	-.125	-.125	.125
0	0	1	.25		.25		.5				.125	-.125	-.125	.125
0	1	0	.25			.25		.5			-.125	.125	.125	-.125
0	1	1	.25			.25		.5			-.125	.125	.125	-.125
1	0	0		.25	.25				.5		-.125	.125	.125	-.125
1	0	1		.25	.25				.5		-.125	.125	.125	-.125
1	1	0		.25		.25				.5	.125	-.125	-.125	.125
1	1	1		.25		.25				.5	.125	-.125	-.125	.125

Table B.1: Coefficients for Main Effects, Joint Effects and Interactions for the Use of Theorems 1 and 2 Based on a $2 \times 2 \times 2$ Grid on three Variables. "ME" Denotes Main Effect, "JE" the Joint Effect or Overall Effect of Two Variables, "IE" the 2-factor Interaction. The Two Levels of Each Variable are Denoted by 0 and 1. Coefficients that are Zero have been Omitted.

effect. For example, the joint effect for x_1, x_2 at $x_1 = 1$ and $x_2 = 0$ is given in the third column of the four joint effect columns.

Appendix C

Proof of Theorem 3

We prove Theorem 3 from Section 4.2.

Theorem 3 : Suppose we use the Gaussian model (4.1) and the covariance function (4.2) is such that the mean square error of prediction in (4.4) is positive for any unsampled point $\mathbf{x}$. Further, suppose the number of possible sampling points is finite. Then the expected improvement algorithm will visit all the sampling points and hence will always find the global minimum.

Proof: At already sampled points s equals zero, and therefore the expected improvement at points already sampled is zero (see equation 4.7). At points not previously sampled s is strictly greater than zero, and hence so is the expected improvement. This can be seen from (4.5): in the range of $y \in [-\infty, f^n_{min})$ the argument $f^n_{min} - y$ is always positive, and since $\phi(.)$ is non-degenerate $\phi(y)$ is positive for $y \in [-\infty, f^n_{min})$. Note we have used the fact that the tails of the normal distribution do not drop to zero.

Then, as long as there are unsampled points, the algorithm will never sample a

previously sampled point. Since the number of points is finite, the algorithm will sample all points and the global minimum will be found. □

It is interesting to note that the proof does not require that the maximal expected improvement is found at each step (as long as the expected improvement found is positive) nor does it require that the unknown function is approximated well by the Gaussian stochastic process.

Appendix D

A Splus Function for the Visualization of High Dimensional Data

In this Appendix we give a Splus function that produces graphs for the visualization of high dimensional data like the one in Figure 4.12.

The function takes one parameter, α, that determines the amount of overlap among the lines. Individual lines on graphs with more overlap are harder to distinguish. On the other hand more overlap makes it possible to detect finer similarities for groups of lines. The value $\alpha = 1$ translates to no overlap with other lines, $\alpha = 3$ means overlap with the two neighboring lines, and so forth. Figure 4.12 was produced with $\alpha = 3$.

```
visualize <- function(x, upperrange = 0, priorobs = 0, alpha = 3)
{
#upperrange: upper limit of desired x-range
#priorobs:   number of obs before this plot (for label of x axis)
#alpha:      stretch factor of differences to the left and right
```

```
#            (alpha=1 corresponds to no overlap of lines)
#            50 obs works well with alpha=1
#            100 obs needs alpha= 2 or 3 to see differences in lines
     n <- nrow(x)
     d <- ncol(x)
     y <- matrix(rep(0, n * (d + 1) * 2), n * (d + 1), 2)
     for(i in 1:n) {
          y[(d + 1) * (i - 1) + 1, 1] <- priorobs + i
          y[(d + 1) * (i - 1) + 1, 2] <- 0
          for(j in 1:d) {
               temp <- diff(range(x[, j]))
               y[(d + 1) * (i - 1) + j + 1, 1] <- priorobs + i +
                       (alpha * (x[i, j] - 0.5 * temp))/temp
               y[(d + 1) * (i - 1) + j + 1, 2] <- j
          }
     }
     i <- 1  # set range up for plot
     y1.range <- range(y[, 1])
     if(upperrange != 0) {
          y1.range[2] <- max(upperrange, y1.range[2])
     }
     y2.range <- range(y[, 2])
     plot(y[((i - 1) * (d + 1) + 1):(i * (d + 1)), 1],
          y[((i - 1) * (d + 1) + 1):(i * (d + 1)), 2],
          type = "l", xlim = y1.range, ylim =
          y2.range, xlab = "Observation", ylab = "Dimension")
     for(i in 2:n) {
          lines(y[((i - 1) * (d + 1) + 1):(i * (d + 1)), 1],
          y[((i - 1) * (d + 1) + 1):(i * (d + 1)), 2])
     }
}
```

Appendix E

Derivation of the Generalized Expected Improvement

In this Appendix we derive the equations (5.2), (5.3) and (5.4).

We can rewrite the improvement given in (5.1) as

$$I^g = \begin{cases} s^g(f'^n_{min} - z)^g & \text{if } z < f'^n_{min} \text{ and } s > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $z = \frac{y-\hat{y}}{s}$ and $f'^n_{min} = \frac{f^n_{min}-\hat{y}}{s}$.

For $s > 0$, taking the expectation yields

$$\begin{aligned} E(I^g) &= s^g \int_{-\infty}^{f'^n_{min}} \sum_{k=0}^{g} (-1)^k \binom{g}{k} {f'^n_{min}}^{g-k} z^k \phi(z) dz \\ &= s^g \sum_{k=0}^{g} (-1)^k \binom{g}{k} {f'^n_{min}}^{g-k} T_k \quad , \end{aligned}$$

where

$$T_k = \int_{-\infty}^{f'^n_{min}} z^k \phi(z) dz.$$

We now calculate T_k using the partial integration technique, splitting the integrand up into z^{k-1} and $z\phi(z) = -\phi'(z)$:

$$\begin{aligned} T_k &= -\left[z^{k-1}\phi(z)\right]_{-\infty}^{f'^n_{min}} + (k-1)\int_{-\infty}^{f'^n_{min}} z^{k-2}\phi(z)dz \\ &= -f'^n_{min}{}^{k-1}\phi(f'^n_{min}) + (k-1)T_{k-2} \quad . \end{aligned}$$

This establishes the recursion formula. Since T_k is a function of T_{k-2}, two starting values, $k = 0$ and $k = 1$, are needed:

$$\begin{aligned} T_0 &= \int_{-\infty}^{f'^n_{min}} \phi(z)dz = \Phi(f'^n_{min}) \\ T_1 &= \int_{-\infty}^{f'^n_{min}} z\phi(z)dz = -\left[\frac{exp(-z^2/2)}{\sqrt{2\pi}}\right]_{-\infty}^{f'^n_{min}} = -\phi(f'^n_{min}) \, . \end{aligned}$$

We now prove by induction that (5.3) solves the recursive formula (5.4). The proof is split up into two cases : k is odd, and k is even.

Case 1 : k is odd.

The initial step for $k = 1$ can be easily verified. For the induction step:

$$\begin{aligned} & -\phi(f'^n_{min})f'^n_{min}{}^{k-1} + (k-1)T_{k-2} \\ = \; & -\phi(f'^n_{min})f'^n_{min}{}^{k-1} \\ & -\phi(f'^n_{min})(k-1)\left[\left(\sum_{j=1}^{(k-3)/2} f'^n_{min}{}^{2(j-1)} \prod_{i=j}^{(k-3)/2} 2i\right) + f'^n_{min}{}^{k-3}\right] \\ = \; & -\phi(f'^n_{min})\left[f'^n_{min}{}^{k-1} + (k-1)f'^n_{min}{}^{k-3} + \left(\sum_{j=1}^{(k-3)/2} f'^n_{min}{}^{2(j-1)} \prod_{i=j}^{(k-1)/2} 2i\right)\right] \end{aligned}$$

$$= -\phi(f'^n_{min}) \left[f'^n_{min}{}^{k-1} + \left(\sum_{j=1}^{(k-1)/2} f'^n_{min}{}^{2(j-1)} \prod_{i=j}^{(k-1)/2} 2i \right) \right] = T_k \, .$$

Case 2 : k is even.

The initial step for $k = 0$ can be easily verified. For the induction step:

$$\begin{aligned}
& -\phi(f'^n_{min}) f'^n_{min}{}^{k-1} + (k-1) T_{k-2} \\
= \; & -\phi(f'^n_{min}) f'^n_{min}{}^{k-1} + (k-1) \prod_{i=1}^{(k/2)-1} (2i-1) \Phi(f'^n_{min}) \\
& -\phi(f'^n_{min}) \left[(k-1) f'^n_{min}{}^{k-3} + \sum_{j=2}^{(k/2)-1} (k-1) f'^n_{min}{}^{2j-3} \prod_{i=j}^{(k/2-1)} (2i-1) \right] \\
= \; & \Phi(f'^n_{min}) \prod_{i=1}^{k/2} (2i-1) - \phi(f'^n_{min}) \left[f'^n_{min}{}^{k-1} + \sum_{j=2}^{k/2} f'^n_{min}{}^{2j-3} \prod_{i=j}^{k/2} (2i-1) \right] = T_k \, .
\end{aligned}$$

Appendix F

Deák's algorithm

Deák (1980, 1986, 1990) presents a modification of the Monte Carlo Method to evaluate multivariate normal integrals. Specifically, he reduces the dimensionality from n to $n-1$ through a clever transformation which allows for easy exact evaluation of one of the integrals. Furthermore, he makes use of the concept of antithetic variables twice, once in the context of orthonormalized estimators. We will now present the method in more detail.

Denote by f the indicator function

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \leq \mathbf{h} \\ 0, & \text{otherwise} \end{cases}$$

where $\mathbf{x}$ is distributed as a multivariate normal vector, and $\mathbf{h}$ is the vector of bounderies.

We are interested in calculating p

$$p = P\{\mathbf{x} < \mathbf{h}\} = \int \dots \int_{R^n} f(\mathbf{x}) d\Phi(\mathbf{x}) \qquad \text{(F.1)}$$

where Φ is the CDF of the multivariate normal distribution. A simple unbiased Monte Carlo estimator of p is then given by

$$\theta_1 = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i),$$

where $\mathbf{x}_i$ is a realization of $\mathbf{x}$. A realization can be computed easily with $\mathbf{Tq} = \mathbf{x}$, where $\mathbf{T}$ is the Choleski decomposition of the covariance matrix $\mathbf{R}$ such that $\mathbf{TT}' = \mathbf{R}$, and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ whose components are independent $N(0,1)$ random variables.

Dimensional reduction and antithetic variables

Further, $\mathbf{q}$ can be written as $\mathbf{q} = k\mathbf{z}$, where k is a χ distributed random variable with n degrees of freedom and distribution function F_n. The vector $\mathbf{z}$ is uniformly distributed on the surface S of the n-dimensional unit sphere (see also Rubinstein, 1981, p.89). Thus

$$p = P(k\mathbf{Tz} < \mathbf{h}) = \int_S \int_0^{+\infty} f(k\mathbf{Tz}) dF_n(k) dV(\mathbf{Tz})$$

where V is the cumulative distribution function of $\mathbf{Tz}$.

We rewrite the inner integral as follows:

$$e_1(\mathbf{y}) := \int_{r\mathbf{y}\in D, r\geq 0} f(k\mathbf{y}) dF_n(k)$$

where $\mathbf{y} = \mathbf{Tz}$ and D denotes the integration domain $(-\infty, \mathbf{h}]$.

We now make use of the method of antithetic variables. That is, we use negatively correlated variables in the estimator with the aim to reduce the variance of

their average. Here, instead of only using $\mathbf{y}$, we also use $-\mathbf{y}$:

$$e_2(\mathbf{y}) := \frac{1}{2}\int_{r\mathbf{y}\in D} f(k\mathbf{y})dF_n(k) = .5[e_1(\mathbf{y}) + e_1(-\mathbf{y})]$$

Our unbiased estimate of p is now

$$\theta_2 = \frac{1}{N}\sum_{i=1}^{N} e_2(\mathbf{y}_i).$$

where $\mathbf{y}_i$ is a realization of $\mathbf{y}$.

Evaluation of the integral $e_2(\mathbf{y})$

We now describe how to evaluate the integral $e_2(\mathbf{y})$. The argument of the integration $f(.)$ indicates whether the ray $r\mathbf{y}$ is in the domain D of interest or not. In order to evaluate the integral we determine when the ray $r\mathbf{y}$ "enters" and "exits" the domain D. There are at most one "entry" and one "exit", because $(-\infty, \mathbf{h}]$ forms a hyper-rectangle. Thus we need to find the constants c_1 and c_2 for which

$$r\mathbf{y} \in D \qquad \text{if} \qquad c_1 \leq r \leq c_2$$

We first determine constants c_{1i} and c_{2i} for each dimension seperately. We have

$$\begin{array}{llll}
c_{1i} = -\infty, & c_{2i} = h_i/y_i & \text{if} & y_i > 0 \\
c_{2i} = h_i/y_i, & c_{2i} = +\infty & \text{if} & y_i < 0 \\
c_{1i} = -\infty, & c_{2i} = +\infty & \text{if} & y_i = 0, h \geq 0 \\
c_{1i} = a, & c_{2i} = a & \text{if} & y_i = 0, h < 0
\end{array}$$

where a is any real number. Since $D = \cap_{i=1}^{n} D_i$, the D_i representing one-dimensional

half space, the constants c_1 and c_2 are determined by

$$c_1 = \max_i c_{1i}, \qquad c_2 = \min_i c_{2i}.$$

Knowing the "entry" and "exit" constants c_1 and c_2, the integral $e_2(y)$ can be easily determined as [1]

$$2e_2(y) = \begin{cases} F_n(c_2) - F_n(c_1) & \text{if} \quad c_1 < c_2, c_1, c_2 > 0 \\ F_n(-c_1) + F_n(c_2) & \text{if} \quad c_1 < c_2, c_1 < 0, c_2 > 0 \\ F_n(-c_1) - F_n(-c_2) & \text{if} \quad c_1 < c_2, c_1, c_2 < 0 \\ 0 & \text{if} \quad c_1 \geq c_2 \end{cases}$$

Orthonormalized estimators

Rather than just generating one other point $-\mathbf{y}$ from $\mathbf{y}$, orthonormalized estimators generate many other points from locations regularly scattered over the surface of the hyper-ball S. The extra points can be generated at much lower cost compared to random points, and their regular location ensures a variance-reducing effect (antithetic variables).

The idea is to form a basis in n-space generated by a initial (random) point $\mathbf{y}$. Including the negatives of all basis vectors, $2n$ points are generated in this way. The corresponding estimator - averaging over the $2n$ points - is called O_1.

For the estimator O_k, $k = 1, 2, \ldots$, we generate in addition to the previous ones all possible combinations of k basis vectors. The effect is that the points are scattered more densely on the surface. As k increases the lower cost advantage diminishes. In our implementation we use the estimator corresponding to O_2, which is recommended by Deák (1980).

---

[1] Deák (1986, 1990) both contain a mistake in that he gives the third line of the equation as $F_n(-c_2) - F_n(-c_1)$ rather than $F_n(-c_1) - F_n(-c_2)$.

Bibliography

[1] Bates, D.M., and Watts, D.G. (1988), *Nonlinear Regression Analysis and its Applications*, New York: John Wiley & Sons.

[2] Betro, B. (1991), "Bayesian Methods in Global Optimization", *Journal of Global Optimization*, 1, 1-14.

[3] Breiman, L. (1991), Discussion of "Multivariate Adaptive Splines" by J.H. Friedman, *The Annals of Statistics*, 19, 1-141.

[4] Cao, S. (1993), *Numerical Investigation on Unglazed Transpired Plate Solar Collector*, unpublished M.Eng. thesis, Department of Mechanical Engineering, University of Waterloo.

[5] Cressie, N. (1993), *Statistics for Spatial Data*, Revised Edition, Wiley & Sons, New York.

[6] 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", *JASA*, 86, 953-963.

[7] Davis, P. (1996), "Industrial Strength Optimization At Boeing", *Siam News*, Jan/Feb 1996.

[8] Deák, I. (1980), "Three digit accurate multiple normal probabilities", *Numerische Mathematik*, 35, 369-380.

[9] Deák, I. (1986), "Computing Probabilities of Rectangles in Case of Multinormal Distribution", *J. Statist. Comput. Simul.*, 26, 101-114.

[10] Deák, I. (1990). *Random Number Generators and Simulation*, Akadémiai Kiadó, Budapest, Hungary.

[11] Dixon, L.C.W. and Szego (eds.) (1978), *Towards Global Optimisation 2*, North Holland, Amsterdam.

[12] Feller, W. (1968), *An Introduction to Probability Theory and its Applications*. Volume 1, Third Edition, New York, Wiley.

[13] Golneshan, A. (1994), *Forced Convection Heat Transfer from Low Porosity Slotted Transpired Plates*, unpublished Ph.D. thesis, Department of Mechanical Engineering, University of Waterloo.

[14] Hastie, T. and Tibshirani, R. (1990), *Generalized Additive Models*, Chapman and Hall, London.

[15] Jones, D.R., Perttunen, C.D., and Stuckman, B.E. (1993), "Lipschitzian Optimization Without the Lipschitz Constant", *Journal of Optimization Theory and Application*, 79, 157-181.

[16] Journel, A.G., and Huijbregts, Ch. J. (1978), *Mining Geostatistics*, Academic Press, New York.

[17] Maurer, W., and Margolin, B.H. (1976), "The Multivariate Inclusion-Exclusion Formula and Order Statistics from Dependent Variables", *The Annals of Statistics*, 1190-1199.

[18] McCullagh, P. and Nelder, J.A. (1989), *Generalized Linear Models* (2nd ed.), London: Chapman and Hall.

[19] McKay, M.D., Beckman, R.J., and Conover, W.J. (1979), "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code", *Technometrics*, 21, 239–245.

[20] Mitchell, T.J., and Morris, M.D. (1994), "Asymptotically Optimum Experimental Designs for Prediction of Deterministic Functions Given Derivative Information", *Journal of Statistical Planning and Inference*, 41, 377–389.

[21] Mockus, J. (1989), *Bayesian Approach to Global Optimization* Kluwer Academic Publishers, Dordrecht, The Netherlands.

[22] Mockus, J. (1994), Application of Bayesian Approach to Numerical Methods of Global and Stochastic Optimization, *Journal of Global Optimization*, 4.347-365.

[23] Montgomery, D.C., and Peck, E.A. (1982), *Introduction to Linear Regression Analysis*, New York, John Wiley & Sons.

[24] 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, 243-255.

[25] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P (1992), *Numerical Recipes in C*, 2^{nd} edition, Cambridge University Press.

[26] Rivoirard, J. (1994), *Introduction to disjunctive kriging and non-linear geostatistics*, Clarendon Press, Oxford.

[27] Rubinstein, R. (1981), *Simulation and the Monte Carlo method*, J. Wiley, New York.

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

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

[30] Schervish, M. J. (1984), "Multivariate Normal Probabilities with Error Bound", *Applied Statistics*, 33, 81-94.

[31] Törn, A. and Žilinskas, A. (1989), *Global Optimization*. Springer Verlag, Berlin.

[32] 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, 15–25.

[33] Yaglom, A.M. (1987), *Correlation Theory of Stationary and Related Random Functions*, Vol. 1, Springer, New York, 1987.