

# Sequential Experiment Design for Contour Estimation from Complex Computer Codes

Pritam Ranjan and Derek Bingham  
Department of Statistics and Actuarial Science  
Simon Fraser University  
Burnaby, BC, CANADA V5A 1S6  
(pritamr@cs.sfu.ca, dbingham@cs.sfu.ca)

George Michailidis  
Department of Statistics  
University of Michigan  
Ann Arbor, MI 48109  
(gmichail@umich.edu)

## Abstract

Computer simulation is often used to study complex physical and engineering processes. While a computer simulator can often be viewed as an inexpensive way to gain insight into a system, it can still be computationally costly. Much of the recent work on the design and analysis of computer experiments has focused on scenarios where the goal is to fit a response surface or process optimization. In this article, we develop sequential methodology for estimating a contour from a complex computer code. The approach uses a stochastic process model as a surrogate for the computer simulator. The surrogate model and associated uncertainty are key components in a new criterion used to identify the computer trials aimed specifically at improving the contour estimate. The proposed approach is applied to exploration of a contour for a network queuing system. Issues related to practical implementation of the proposed approach are also addressed.

KEY WORDS: Computer experiment; Gaussian process; Inverse problem.

## 1 Introduction

Computer simulation is frequently used as a cost-effective means to study complex physical and engineering processes. While a computer code (simulator) can often be viewed as an inexpensive way to gain insight into a system, it can still be computationally costly. Consequently, it is desirable to perform only a limited number of simulation trials. Investigators must therefore select the runs carefully and perform a *computer experiment*.

A computer experiment frequently involves the modeling of complex systems using a deterministic computer code. As such, replicate runs of the same inputs will yield identical responses. To deal with the lack of randomness in the response, Sacks et al. (1989) proposed modeling the response as a realization from a Gaussian stochastic process (GASP). Most recent work on the design of computer experiments has focused on experiments where the goal is to fit a response surface or to optimize a process (for an overview, see Santner et al. 2003). Experiment plans such as Latin hypercube designs (Mckay et al. 1979), orthogonal array based Latin hypercube designs

(Owen 1992; Tang 1993), space filling designs (Johnson et al. 1990), uniform designs (Fang et al. 2000) and sequential approaches (Jones and Jin 1994) are commonly used in such investigations. Response surface estimation and optimization are not the only practical objectives in computer experiments. For example, for the problem that motivated this work the objective is to determine the system configurations which produce a specific output from a network queuing simulator. In contrast to the global objectives of fitting a response surface or optimization mentioned above, this is a more localized objective which requires an alternate design strategy to most efficiently utilize the available computing resources.

In this article, we develop a sequential design methodology for estimating a contour (also called a level set or iso-surface) of a non-degenerate complex computer code. In the network queuing application which motivated this work, the problem is to estimate a contour of a globally or locally monotone response surface, where the contour identifies the boundary which distinguishes “good” from “bad” performance. Other applications include exploring when the response surface is a function of many input factors that have different cost structures, and identifying the contour allows in subsequent stages to minimize the production cost of equally desirable system configurations.

The problem of contour estimation bears some resemblance to problems where wombling techniques proved useful (Lu and Carlin 2005; Barbujani et al. 1989; Banerjee and Gelfand 2005). These approaches have been used to identify regions of rapid or abrupt changes known as difference boundaries (e.g., Banerjee and Gelfand 2005). Effort is spent on detecting such regions and estimating the boundary where the change occurs. For the problem addressed here, interest lies in estimating a *pre-specified* contour from an expensive computer simulator. The presence of such rapid changes may or may not exist in our setting.

The article is outlined as follows. The network queuing problem is introduced in the next section. In Section 3, a new design and analysis methodology is proposed for contour estimation in computer experiments. Next, we apply the methodology to several examples in Section 4 to illustrate the performance, and return to the network queuing problem in Section 5. Final comments and recommendations are given in Section 6.

## 2 Motivating Example

In this section we discuss an example stemming from a computer networking application that motivated the present work. Consider a queueing system comprised of  $K$  first-in-first-out (FIFO) queues and a single server. There is a stochastic flow of jobs arriving at rate  $\lambda_k$ ,  $k = 1, \dots, K$  to the queues (Warland, 1988). The queues have infinite capacity buffers, where jobs are placed while waiting to be served. At any given time the server performs the jobs of the queues at a certain

rate. A server allocation/scheduling policy is used to decide which queues to serve. The policy that provably achieves maximum *throughput* (i.e. the average amount of work processed by the system per unit of time) was studied by Bambos and Michailidis (2004). This canonical queuing system captures the essential dynamics of many practical systems, such as data/voice transmissions in a wireless network, where the modulated service rates are a consequence of the fluctuations due to interference and/or changing environmental conditions of the transmitting channel, or in a multi-product manufacturing system, where the modulated service rates are due to the availability of the necessary resources for production purposes.

In this paper, we consider for illustration purposes a 2-queue system ( $K = 2$ ), operating in discrete time. The performance measure of interest is the average delay for jobs (customers) into the system as a function of the input vector  $\lambda = (\lambda_1, \lambda_2)$ . Unfortunately the average delay is not available in closed form and must be found numerically via a computer simulator. Therefore, the computer model of interest is the one that simulates this queuing process. For this system, the computer code simulates 100,000 departures from the system with fixed  $\lambda$ . The first 10,000 are discarded from every simulation run as they correspond to the warm-up period (burn-in) from a lightly to a heavily loaded system. The average delay of the remaining 90,000 departures is then computed. The large number of departures is chosen for two reasons: (i) to ensure that the system has warmed up to a steady state for any  $\lambda$ ; and (ii) to observe simulation outputs that are (essentially) noiseless.

An important problem for the system's operator is to identify the set of input rate configurations,  $\lambda$ , such that the aggregate queue length/delay process does not exceed a certain threshold that is deemed acceptable to the customers served by such a system. Customers will likely look elsewhere if the service time is too long. This problem is equivalent to estimating the corresponding contour of the response surface (queue length/delay) as a function of  $\lambda$ . Even in this 2-queue system, computer simulation when the input rates are high can be quite costly, taking up several hours per run. Consequently, one must judiciously select the simulation trials to efficiently estimate the contour of interest. That is, an experiment design strategy for contour estimation is needed.

### 3 Methodology and Algorithm

A new methodology for performing sequential computer trials to explore and estimate a contour is now developed. The proposed approach consists of two main components. Firstly, a stochastic process model is used as a surrogate for the computer model which helps provide an estimate of the contour, and also uncertainty, given the current set of computer trials. Secondly, a new criterion is proposed which is used to identify the computer trials aimed specifically at improving the contour

estimate.

### 3.1 Model Preliminaries and Notation

GASP models, are frequently used to model the output from complex computer codes (e.g., Sacks et al. 1989; Jones et al. 1998). The lack of fit in the case of modeling a deterministic code is due entirely to modeling error and not because of noise, thereby necessitating a departure from the usual approaches. Perhaps the most compelling reason for using this modeling strategy is that GASP models fit a broader class of potential response surfaces than, say, polynomial regression models, and easily adapt to the presence of non-linearity without adding complexity to the model. In our setting, the GASP model is used as an inexpensive surrogate for the computer code which can be more cheaply explored. In the following, we present the formulation of the GASP model which we have found to be successful in our applications. The properties of this model are well known (e.g., see Jones et al. 1998), and we illustrate its important features for purposes of readability.

Let the  $i^{\text{th}}$  input and output for the computer experiment be denoted by a  $d$ -dimensional vector,  $x_i = (x_{i1}, \dots, x_{id})$ , and the univariate response,  $y_i = y(x_i)$ , respectively. Without loss of generality, assume that the design space (or input space) is the unit hypercube,  $\chi = [0, 1]^d$ . The  $n \times d$  experiment design matrix,  $X$ , is the matrix of the input trials. The outputs for the simulation trials are held in the  $n$ -dimensional vector  $y = y(X) = (y_1, y_2, \dots, y_n)'$ . The output of the simulator,  $y(x_i)$ , is modeled as:

$$y(x_i) = \mu + z(x_i); \quad i = 1, \dots, n, \quad (1)$$

where,  $\mu$  is the overall mean,  $z(x_i)$  is a spatial process with  $E(z(x_i)) = 0$ ,  $Var(z(x_i)) = \sigma_z^2$ ,  $cov(z(x_i), z(x_j)) = \sigma_z^2 R_{ij}$  and,

$$R_{ij} = \text{corr}(z(x_i), z(x_j)) = \prod_{k=1}^d \exp\{-\theta_k(x_{ik} - x_{jk})^{p_k}\} \quad \forall i, j. \quad (2)$$

More generally,  $y(X)$  has a multivariate normal distribution  $N_n(\mathbf{1}_n\mu, \Sigma)$ , where  $\Sigma = \sigma_z^2 R$  and  $R = [R_{ij}]$ .

Some aspects of this formulation are worth noting. Firstly, rather than modeling the output through the mean, the correlation among the responses is modeled as a stochastic process (e.g., Sacks et al. 1989; O'Connell et al. 1997). Secondly, the exponent,  $p_k$ , relates to the smoothness of the surface in the direction of factor  $k$ . In the application in Section 2, and the examples considered in the next section, specifying  $p_k = 2$ , which stipulates a smooth response surface, is a reasonable simplifying assumption. Lastly, the chosen covariance function is separable in the sense that it is the product of component wise covariances. This is convenient in so far as it enables us to handle a large number of inputs, since each dimension requires only one parameter ( $\theta_k$ ).

The likelihood for the model is,

$$L(\theta, \mu, \sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2\sigma_z^2} (y - \mathbf{1}_n \mu)' R^{-1} (y - \mathbf{1}_n \mu)\right). \quad (3)$$

If the covariance parameters,  $\theta = (\theta_1, \dots, \theta_d)$ , is known, estimates of  $\mu$  and  $\sigma_z^2$  can be written in a closed form (see Jones et al. 1998 for details),

$$\hat{\mu} = (\mathbf{1}_n' R^{-1} \mathbf{1}_n)^{-1} (\mathbf{1}_n' R^{-1} y) \quad \text{and} \quad \hat{\sigma}_z^2 = \frac{(y - \mathbf{1}_n \hat{\mu})' R^{-1} (y - \mathbf{1}_n \hat{\mu})}{n}. \quad (4)$$

Substitution of these estimates in the actual likelihood (3) gives the ‘‘profile likelihood’’, which can be used to compute the likelihood estimates of the  $\theta_i$ ’s (Rao 1972; Wolfinger et al. 1994).

The model can now be used to estimate responses at any non-sampled point  $x^*$ . The best linear unbiased predictor (BLUP) for  $y(x^*)$  is (see Henderson 1975 for details)

$$\hat{y}(x^*) = \hat{\mu} + r' R^{-1} (y - \mathbf{1}_n \hat{\mu}), \quad (5)$$

with mean squared error

$$s^2(x^*) = \sigma_z^2 \left(1 - r' R^{-1} r + \frac{(1 - \mathbf{1}_n' R^{-1} r)^2}{\mathbf{1}_n' R^{-1} \mathbf{1}_n}\right), \quad (6)$$

where  $r = (r_1(x^*), \dots, r_n(x^*))'$ , and  $r_i(x^*) = \text{corr}(z(x^*), z(x_i))$  is defined in equation (2). In practice the parameters in (4) are replaced with the estimates obtained from MLE. Consequently,  $R$  and  $r$  are replaced with  $\hat{R}$  and  $\hat{r}$ , respectively, in (5) and (6).

A quick glance at equation (2) shows that when the distance between two points is small, the correlation among the responses is relatively high (e.g., close to 1). On the other hand, when the design points are relatively far away, the responses have low correlation. In terms of prediction at un-sampled points in equation (5), this implies that nearby sampled points will have more impact on the interpolation of predicted values than far away points. A further consequence of the model is that the predicted response at sampled points will be exactly the observed response, with the prediction error of zero (which is an appealing feature for modeling a deterministic computer code).

### 3.2 The Improvement Function

Much of the literature on computer experiments has dealt with the estimation of a global response surface and the related problem of finding the minimum (or maximum) of the response surface. The goal here is to estimate a contour. More formally, let  $a$  be the value of the response surface for which we would like to estimate the contour  $S(a) = \{x : y(x) = a\}$ . A naive approach for estimating  $S(a)$  is to simply perform an  $n$ -trial experiment design, estimate the response surface in (1) and then extract an estimated contour from the resulting surface. Such an approach is likely to be inefficient

since it may concentrate most of the sampling effort in regions of the input space which provide little help in estimating the contour of interest. The aim here is to use the available resources as effectively as possible. Furthermore, in a high-dimensional space, the  $n$ -trial experiment design will likely have to be quite large in order to give a sufficiently accurate global response surface from which the contour can be efficiently estimated.

The approach taken here is to sequentially explore the input space, attempting to make effective use of the available resources. Ideally, one could select trials which lie exactly on the contour, but the true contour is unknown. One strategy is to perform a relatively small experimental design and sequentially choose design points that are on or near the estimate of the contour. That is, choose new trials to perform where  $\hat{y}(x)$  belongs to a neighborhood,  $(a - \epsilon, a + \epsilon)$ , of the current contour estimate.

To efficiently use the available resources, one would like to maximize the information provided by new computer trials in the neighborhood of the contour. In similar spirit to Jones et al. (1998), we propose an *improvement function* to help identify optimal computer trials. Define the improvement function as,

$$I(x) = \epsilon^2(x) - \min \{ (y(x) - a)^2, \epsilon^2(x) \}, \quad (7)$$

where  $\epsilon(x) = \alpha s(x)$  for some positive constant  $\alpha$  and  $y(x) \sim N(\hat{y}(x), s^2(x))$ . The term  $\epsilon(x)$  defines a neighborhood around the contour which is a function of  $s(x)$ . This allows the radius of band to be zero for the design points already in the sample of input trials. Furthermore, the criterion will tend to be large if we sample from the set  $\{x : y(x) = a\}$  where the predicted variance is largest. This feature has nice intuitive appeal, since the predicted variance tends to be big in areas of sparse sampling.

With these features of the improvement function in mind, one may be tempted to select design points which maximize  $I(x)$  over the design space  $\chi$ . However, for any un-sampled design point  $x \in \chi$  we are uncertain about the true value of  $y(x)$ . Thus, there may be regions of the design space that have not been sufficiently explored and the standard error of  $\hat{y}(x)$  is relatively high. This would imply that even though estimates of the response are not within the  $\epsilon$ -band of the contour, the contour may lie in, say, a 95% confidence interval in this region. Therefore, the improvement is instead averaged over the uncertainty in the response surface. That is,

$$\begin{aligned} E[I(x)] &= [\alpha^2 s^2(x) - (\hat{y}(x) - a)^2] \left[ \Phi \left( \frac{a - \hat{y}(x)}{s(x)} + \alpha \right) - \Phi \left( \frac{a - \hat{y}(x)}{s(x)} - \alpha \right) \right] \\ &+ 2(\hat{y}(x) - a)s^2(x) \left[ \phi \left( \frac{a - \hat{y}(x)}{s(x)} + \alpha \right) - \phi \left( \frac{a - \hat{y}(x)}{s(x)} - \alpha \right) \right] \\ &- \int_{a - \alpha s(x)}^{a + \alpha s(x)} (y - \hat{y}(x))^2 \phi \left( \frac{y - \hat{y}(x)}{s(x)} \right) dy, \end{aligned} \quad (8)$$

where,  $E[I(x)]$  is the expected improvement at  $x \in \chi$ ,  $\phi(\cdot)$  and  $\Phi(\cdot)$  are the standard normal probability density function (pdf) and cumulative distribution function (cdf), respectively (the derivation of equation (8) can be found in Result 1 of Appendix B). Note that in practice,  $\hat{R}$  and  $\hat{r}$  are substituted in  $s(x)$  as described in (6).

The expression for  $E(I(x))$  contains three terms which are easily interpretable. When  $\hat{y}(x)$  is close to  $a$ , the first term tends to dominate this expression. These points are essentially the points that are near the  $\epsilon$ -band specified by  $I(x)$ . If  $\hat{y}(x)$  is further away from  $a$ , then the second term tends to dominate. This term supports sampling in regions of the input space where the estimated response surface is outside the  $\epsilon$ -band in  $I(x)$ , but the uncertainty of the prediction is quite high. The final term in the expression is related to the variability of the predicted response surface in the  $\epsilon$ -neighborhood of  $a$ . This term encourages sampling in regions near the estimated contour, but where the predicted variance is quite high. In essence, this term identifies points near the predicted contour in more sparsely sampled regions of the input space.

While acting simultaneously, the first and last term encourage sampling near the contour, but in regions where we have little information. The second term allows the sampling mechanism to occasionally jump away from the estimated contour to more sparsely sampled regions of the input space where the contour may plausibly exist. It is the latter scenario which justifies the use of  $E(I(x))$  over  $I(x)$ . In this situation, if we have  $(\hat{y}(x) - a)^2 > \epsilon^2(x)$ , then  $I(x) = \epsilon^2(x) - \min\{(\hat{y}(x) - a)^2, \epsilon^2(x)\}$  is zero, whereas the second term in the expression of  $E(I(x))$  supports exploration of regions of the inputs space that are further away from  $a$  and where the uncertainty of the predictor is high.

### 3.3 Contour Extraction

Exact iso-surface (contour) extraction from any surface is a crucial problem, particularly in higher dimensions (Sethian 1999; Uhlir 2003; Uhlir and Skala 2003). An attractive feature of the GASP model is that an implicit function can be easily constructed thereby allowing for easy extraction of the contour. For the GASP model outlined in Section 3.1, the  $d$ -dimensional implicit function which defines the contour estimate  $S(a)$  is

$$a = \hat{\mu} + \hat{r}'\hat{R}^{-1}(y - \mathbf{1}_n\hat{\mu}), \quad (9)$$

which gives a compact and convenient representation of the contour. To use (9), to extract the contour in two dimensions, packaged software can provide the points that lie on the contour (e.g., in Examples 1 and 2, we use the set of points obtained from the *Matlab* function *contourc* for representation of the estimated contours and the true contour). For higher dimensions, the level set  $S(a) = \{x \in R^d : y(x) = a\}$  is obtained from a fine grid on the design space.

### 3.4 Implementation

The proposed approach consists of first performing a relatively small design to obtain an initial estimate of the response surface, followed by alternately selecting the new trials and refitting the surface. Finally, when the experiment trials have been exhausted the contour is extracted from the estimated surface.

To begin using the criterion in (8), an initial estimate of the response surface model outlined in equation (5) is needed. This is done by first performing an experiment design using only a fraction of the allotted experiment budget. As discussed in the next section, we have found empirically that roughly 25% – 35% of the experimental budget tends to be a good rule of thumb. There are a number of choices for initial designs, as mentioned in Section 1. Experience shows that most of these designs perform equally well for our purposes. For the examples examined in the next section, random Latin hypercube designs are employed, due to their ease of implementation and hence attractiveness to practitioners.

After performing the initial design, the GASP model in (1) is estimated using the maximum likelihood procedure outlined in Section 3.1. Next, the model parameter estimates are used to evaluate  $E(I(x))$  on the sample space with the goal of finding the design point which maximizes the expected improvement (see next section for details). A new computer experiment trial is performed at the location of the optimum of (8) and the response surface is re-estimated. The approach continues in this fashion until the budget of runs is exhausted. Lastly, the final contour estimate is extracted by the implicit function in (9). The procedure is summarized below.

1. Perform an initial experiment design of sample size  $n = n_0$ .
2. Fit the response surface in (5) to the available data.
3. Identify the design point which maximizes  $E(I(x))$  (shown in 8) and perform a computer trial at this design point.
4. Update the data (i.e.,  $X = [X' \ x_{max}]'$ ,  $Y = [Y' \ y(x_{max})]'$  and  $n = n + 1$ ).
5. Iterate between Step 2 and Step 4 until the experiment budget has been exhausted or some stopping criterion has been achieved.
6. Extract the desired contour  $\hat{S} = \{x : \hat{y}(x) = a\}$  from this final surface.

### 3.5 Optimization Issues

The proposed approach is sequential in nature and involves optimizing (8) at each step. There are two obvious ways in which this task can be accomplished. The first approach, analogous to Jones et al. (1998) is to develop a specific branch-and-bound algorithm targeted to the function under consideration. Note that such an approach requires deriving lower and upper bounds that satisfy some specific convergence criteria, namely, R1 and R2 in Balakrishnan et al. (1991). Further, it can be shown that (8) is not monotonic on the design space and therefore the bounds involve tuning according to the nature of the partial derivatives of the function. Consequently, the implementation of the algorithm becomes a rather involved exercise.

As an alternative, one could maximize (8) using readily available optimizers in common software packages (e.g., Matlab, R), which do not require any function specific information. Two such algorithms are used here: (i) the global branch-and-bound algorithm (Murty 1995); and (ii) the Matlab Optimization Toolbox function *fmincon*.

The branch-and-bound algorithm (Murty 1995) is general and does not require information about lower or upper bounds. The algorithm partitions the design space and stores the optimum value of the function for each subset instead of storing the information about lower bounds and upper bounds. Pruning of the subsets is based on the local optimum value instead of the function specific bounds. The optimum in each subset is obtained using a constrained optimization. The Matlab Optimization Toolbox function *fmincon*, on the other hand, attempts to find a constrained optimum using a sequential quadratic programming method (Schittkowski 1985, Fletcher 1987).

Like many optimization approaches, the identification of the global optimum depends on the good starting values. To identify the starting points for the optimization routines, a genetic algorithm is employed (e.g., Holland 1975; Mandal et al. 2006) with the following specifications.

- *Initial Population*: A population of  $500d$  candidate solutions is randomly generated from the design space, where  $d$  denotes the number of factors involved.
- *Selection*: Candidate solutions are selected in order to optimize the objective function.
- *Crossover*: A pair of candidate solutions is crossed at  $[d/2]$  randomly chosen locations (i.e., values of the factors at the chosen locations are swapped for the two solutions).
- *Mutation*: Every candidate solution is subjected to perturbations to avoid the search being trapped into local optima.

After the crossover and mutation steps, the best  $500d$  solutions from the three populations are kept. The above steps are repeated for  $5d$  generations and the best solution is retained from the final

population. Subsequently, this final solution is used as a starting value for the optimizers. To avoid obtaining a solution that corresponds to a local optimum, a set of  $2d$  starting values (i.e. optimal solution from  $2d$  restarts of the genetic algorithm) is used as input to the optimizers and then the solution that achieves the maximum value of (8) amongst this set is selected for a new trial. It turns out that both the genetic algorithm and the packaged optimizers are computationally fast (we comment on timing issues at the end of Section 4).

### 3.6 Convergence of $E[I(x)]$

The convergence of the proposed approach is now discussed. We begin with a useful lemma, and use it in the proof of the section's main result which establishes the rate of convergence. All proofs are given in Appendix A.

**Lemma-1:** *If  $f_n(\cdot|x), \forall n \geq 1$  are non-negative functions, then*

$$\sup_{x \in \mathcal{X}} f_n(\cdot|x) \rightarrow 0 \quad \Rightarrow \quad \lim_{n \rightarrow \infty} \sup_{x \in \mathcal{X}} E[f_n(\cdot|x)] = 0.$$

The implications of the lemma are straightforward. Setting  $f_n(y|x) = I(x)$ , the following are equivalent:

$$\begin{aligned} \lim_{n \rightarrow \infty} \sup_{x \in \mathcal{X}} E(I(x)) &= 0, \text{ and} \\ \lim_{n \rightarrow \infty} \sup_{x \in \mathcal{X}} E[f_n(y|x)] &= 0. \end{aligned}$$

So, if it can be shown that  $\sup_{x \in \mathcal{X}} I(x) \rightarrow 0$ , then it implies that  $\lim_{n \rightarrow \infty} \sup_{x \in \mathcal{X}} E(I(x)) = 0$ . That is, as  $n \rightarrow \infty$  the contour is known perfectly and we can not improve our knowledge.

**Theorem-1:** *Under the correlation structure defined in equation (2) and the expected improvement function defined in equation (8),*

$$\lim_{n \rightarrow \infty} \sup_{x \in \mathcal{X}} E(I(x)) = 0.$$

*More precisely,  $\sup_{x \in \mathcal{X}} E(I(x)) = O\left(\frac{1}{\log(n)}\right)$  and therefore converges to zero in limit.*

Recall that  $I(x) = \alpha^2 s^2(x) - \min\{(y(x) - a)^2, \alpha^2 s^2(x)\}$ , and also  $0 \leq I(x) \leq \alpha^2 s^2(x)$ , which satisfies the conditions of Lemma 1 and Theorem 1. Since  $\sup_{x \in \mathcal{X}} I(x) \leq \sup_{x \in \mathcal{X}} \alpha^2 s^2(x)$ , then proving  $\sup_{x \in \mathcal{X}} s^2(x) \rightarrow 0$  as  $n \rightarrow \infty$  is sufficient for the convergence of the algorithm.

A lurking mathematical problem is to decide on the sufficient number of trials to get a good approximation of the underlying contour. The theorem implies that, as we add more trials, the supremum of the expected improvement goes to zero with a rate faster than  $\frac{1}{\log(n)}$ . In the case of

unlimited computer trials, or more realistically the ability to run several more trials, this points to a strategy for a stopping rule for the procedure. That is, one could pre-specify a value of the expected improvement and would terminate the algorithm when the supremum of the expected improvement is consistently lower than the pre-specified value.

## 4 Examples

To illustrate the approach, several examples are now presented. Before proceeding directly, we note that there is still the choice of the initial run-size of the experiment,  $n_0$ , facing the experimenter. In each of the examples, several choices of  $n_0$  are considered and the performance is observed. We also present several plots which aid in evaluation of the approach. To provide a basis of comparison, we compare the proposed methodology to the simpler approach of laying out all design points according to a single random Latin hypercube design (McKay et al. 1979) and extracting a contour from the estimated response surface.

**Example 1.** Consider a simple first order polynomial model,  $y(x_1, x_2) = x_1 + x_2 + x_1x_2$  with  $x_1, x_2 \in [0, 1]$  and the contour of interest be  $S(1.5)$ . In spite of the relatively simple nature of the underlying response surface, there are still a few choices to be made. Firstly, the experimenter must choose the total run-size,  $n$ , of the experiment. In practice, the run-size budget may be fixed beforehand because of the cost of running the code. Next, one must choose the fraction of the experimental budget, and thus the initial run-size,  $n_0$ , that will be assigned to an initial experiment design. It has been recommended that roughly 10-20 points per dimension is a reasonable choice for the response surface estimation (Jones et al. 1998). In the same spirit, we consider run-sizes of  $n = 20$ , and  $n = 25$  for comparison under this relatively simple situation. To investigate the performance with different sizes of the initial designs, Latin hypercube designs (McKay et al. 1979) with run-size of  $n_0 = 5, 10$ , and  $15$  are used to begin the procedure.

To visualize the methodology, consider a single application of the procedure where  $n = 20$  and  $n_0 = 10$ . Figure 1(a) displays the location of the  $n_0 = 10$  random Latin hypercube design points, as well as the true (solid line) and estimated contours (dotted line). The void circle on the bottom left corner of the Figure 1(a) is the location of the new trial, obtained by maximizing the expected improvement function (8). Similarly, Figures 1(b) and (c) show the true contour, along with the contour extracted from the surface based on 1 and 5 additional trials, respectively, and the next trial to be performed. The final estimate of the contour, based on 10 additional design points, is shown in Figure 1(d).

Looking at these figures, we see that the true contour is well approximated after adding only 1 additional design point. Furthermore, note that only one design point (the tenth additional trial)

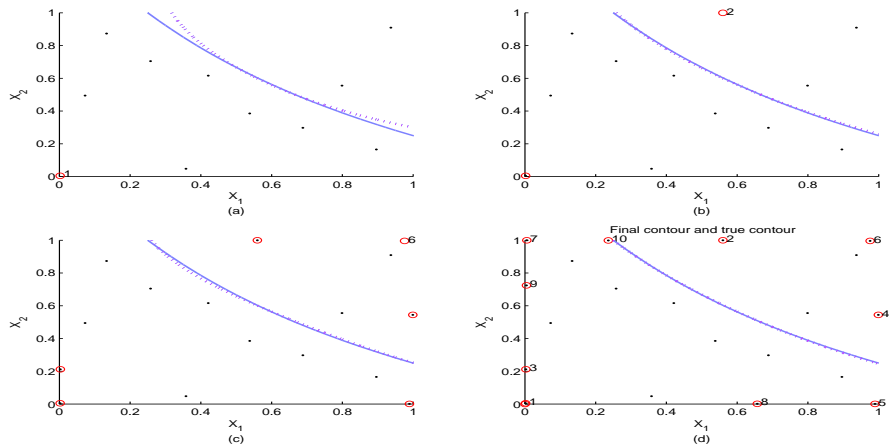


Figure 1: Illustration of the sequential approach for contour estimation ( $n = 20$  and  $n_0 = 10$ ). The solid curve refers to the true contour, the dotted curve to the estimated contour and the dotted points refer to the initial design.

is near the predicted contour. This demonstrates the importance of the second component of (8), which stipulates that sampling should not only be done near the contour, but also in regions where the prediction variance is high and the contour is plausible.

Since the true contour in this example is known, we can attempt to evaluate the “goodness” of the estimated contour after adding each new trial. One can envision many ways to measure the closeness of the true and estimated contours. We consider three discrepancy measures (Veltkamp 2000; Brault and Mounier 2001) to evaluate the closeness of the estimated contour and true contour.

Let  $C_{n_0,k}$  be the estimate of the contour  $S(a)$  from the surface based on  $n_0$  many starting design points and  $k$  additional trials (i.e.,  $C_{n_0,k} = \{x : \hat{y}(x) = a\}$ ). Here  $\hat{y}(x)$  is the predicted surface after adding  $k$  additional trials. Hence,  $C_{n_0,0}$  is the contour estimate extracted from the predicted surface based on  $n$  design points using Latin Hypercube design. Also, let  $C_t$  be the true contour. A sample of  $m$  points is taken from each contour and the discrepancy measures evaluated on the sample. Assuming that the true underlying function and the estimated surfaces are known, sampling from the estimated contour and the true contour is inexpensive. Therefore, if needed, one can take large values of  $m$  to obtain a good approximation of the contours. As mentioned in Section 3.3, one can try different grid sizes, leading to different values of  $m$ , to get a stable representation of the contours. Note that the value of  $m$  may vary from function to function because of the complexity of the contour of interest. After obtaining a representative sample from the contours, denote the discrete sample from  $C_{n_0,k}$  as  $\tilde{C}_{n_0,k} = \{x_1^k, \dots, x_m^k\}$ , where  $x_i^k = (x_i^{1k}, \dots, x_i^{dk})$ . Similarly, denote the sample from  $C_t$  as  $\tilde{C}_t = \{x_1^t, \dots, x_m^t\}$ .

The first discrepancy measure calculates the lack in correlation between the estimated and true

contours, and is denoted by

$$M_1 = \sum_{l=1}^d (1 - \text{corr}(x^{lk}, x^{lt})). \quad (10)$$

Here,  $x^{lk}$  and  $x^{lt}$  are the  $l$ -th coordinate of  $C_{n_0,k}$  and  $C_t$  respectively. So,  $\text{corr}(x^{lk}, x^{lt})$  is the correlation between  $x^{lk}$  and  $x^{lt}$ , where,  $x_i^k$  is homologous to  $x_i^t$  for each  $i = 1, \dots, m$  and for each  $k$ .

The second discrepancy measure is the average  $L_2$  (Euclidean) distance between  $C_{n_0,k}$  and  $C_t$

$$M_2 = \frac{1}{|\tilde{C}_{n_0,k}|} \sum_{x \in \tilde{C}_{n_0,k}} d(x, \tilde{C}_t), \quad (11)$$

where  $d(x, \tilde{C}_t) = \min\{\|x - y\|_2 : y \in \tilde{C}_t\}$ .  $M_2$  measures the average distance between two contours.

Lastly, the third discrepancy measure is the maximum  $L_2$  distance between  $C_{n_0,k}$  and  $C_t$

$$M_3 = \max\{d(x, \tilde{C}_t) : x \in \tilde{C}_{n_0,k}\}. \quad (12)$$

This discrepancy measure aims to measure maximum separation between two contours. This metric guards against the worst case in terms of closeness, and tends to be more sensitive to large departures than  $M_2$ .

We have found both the numeric and visual displays of these discrepancy measures to be helpful in evaluating the methodology. The diagnostic plots based on these discrepancy measures, for this single application of the procedure, are shown in Figure 2. Note that all the discrepancy measures have a decreasing trend as  $n$  gets larger. As we are adding more trials (i.e., as  $k$  increases)  $C_{n_0,k} \rightarrow C_t$ . In fact, one can show that each of the discrepancy measures converges to zero as  $k \rightarrow \infty$ . Proofs are given in Appendix B.

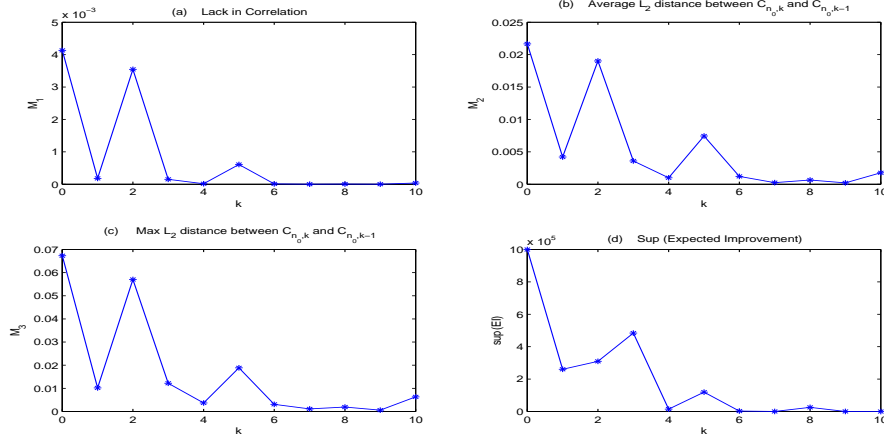


Figure 2: Lack of fit plots illustrating the improvement in the information of the contour as a result of sequential addition of trials. (a) refers to the average  $L_2$  distance, (b) to the lack in correlation, (c) to the maximum  $L_2$  distance between the estimated and true contours and the maximum expected improvement is shown in (d).

In practice, one does not have infinite resources available. Nonetheless, as more trials are performed, one can plot the sequence of values for these metrics to gain some insight into the closeness of the estimated contour to the true contour. Further, one can formulate a reasonable stopping rule based on the discrepancy measures (i.e., when they are close to zero). When sufficient trials have been chosen to get a good approximation of the contour, further additional trials will show an insignificant change in these discrepancy measures. However, this feature alone does not always indicate a good approximation of the underlying contour. So, in practice one would want to observe a sequence of several small changes in the discrepancy measures before concluding that the contour has been adequately approximated.

Finally, for the representation of the contour from the final surface, the implicit function is obtained from the GASP model as described in equation (9). Figure 3 displays the estimated contour (solid line) and the true contour (dotted line). For this example, the estimated and true contours are indistinguishable.

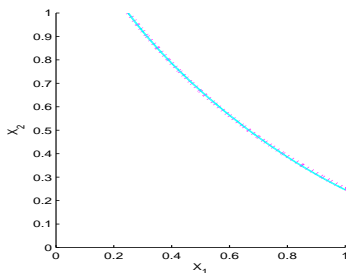


Figure 3: Final estimate of the contour plotted using implicit function from GASP. Solid curve refers to the estimated contour and the dotted curve to the true contour.

To compare the proposed sequential approach with a single stage random Latin hypercube design, a simulation study is conducted. The first stage of the proposed methodology uses a random Latin hypercube design with  $n_0$  trials and proceeds to sequentially add  $k$  additional points. This approach is compared to using a single random Latin hypercube design with  $n$  points ( $n = n_0 + k$ ). The procedure is repeated 500 times.

Table 1 summarizes the results of the simulation study for this response function. Entries in the table correspond to the average discrepancies over the 500 simulation runs for the single stage Latin hypercube design (denoted by LHC) and the new sequential approach (denoted by Sequential) with  $n = 20$  and 25, respectively. Also note that the table is divided into different sizes of initial starting designs ( $n_0 = 5, 10$  and 15). The third column gives the relative efficiency (R.E.) of the two approaches and is the ratio of entries in the first column with the corresponding entry in the second column.

Table 1: Results of the simulation study for comparing the performance of the sequential approach and the single stage Latin Hypercube sampling, separated by  $n = 20, 25$  and different choices of initial run size  $n_0$ .

		$n_0 = 5, k = 15 (n = 20)$			$n_0 = 5, k = 20 (n = 25)$		
Measure		LHC	Sequential	R.E.	LHC	Sequential	R.E.
	$M_1$	3.54e-04	1.96e-04	1.80	1.05e-04	5.20e-05	2.02
	$M_2$	0.0025	0.0022	1.13	0.0015	0.0012	1.25
	$M_3$	0.0103	0.0056	1.84	0.0066	0.0031	2.13
		$n_0 = 10, k = 10 (n = 20)$			$n_0 = 10, k = 15 (n = 25)$		
Measure		LHC	Sequential	R.E.	LHC	Sequential	R.E.
	$M_1$	3.54e-04	2.82e-04	1.25	1.05e-04	6.54e-05	1.61
	$M_2$	0.0025	0.0020	1.25	0.0015	0.0014	1.07
	$M_3$	0.0103	0.0057	1.80	0.0066	0.0034	1.94
		$n_0 = 15, k = 5 (n = 20)$			$n_0 = 15, k = 10 (n = 25)$		
Measure		LHC	Sequential	R.E.	LHC	Sequential	R.E.
	$M_1$	3.54e-04	3.03e-04	1.16	1.05e-04	8.68e-05	1.21
	$M_2$	0.0025	0.0024	1.04	0.0015	0.0013	1.15
	$M_3$	0.0103	0.0073	1.41	0.0066	0.0035	1.88

See that for both  $n = 20$  and  $n = 25$ , regardless of the initial sample size  $n_0$ , the relative efficiencies are larger than 1. This indicates that the proposed methodology is performing better than naive Latin hypercube design. Also note that as we go down the table, for given  $n$ , relative efficiencies are decreasing to 1 as  $(n - n_0)$  is decreasing to 0. This is expected since, as  $n_0$  gets larger the proposed approach becoming more like the naive Latin hypercube design approach. The best choice of initial run size considered for this example is  $n_0 = 5$ .

**Example 2.** Let  $x_1, x_2 \in [0, 1]$ , and the underlying function  $f$  be the Goldprice function (Andre, Siarry and Dognon 2000),

$$f(x_1, x_2) = \left[ 1 + \left( \frac{x_1}{4} + 2 + \frac{x_2}{4} \right)^2 \left\{ 5 - \frac{7x_1}{2} + 3 \left( \frac{x_1}{4} + \frac{1}{2} \right)^2 - \frac{7x_2}{2} + \left( \frac{3x_1}{2} + 3 \right) \left( \frac{x_2}{4} + \frac{1}{2} \right) + 3 \left( \frac{x_2}{4} + \frac{1}{2} \right)^2 \right\} \right]^* \\ \left[ 30 + \left( \frac{x_1}{2} - \frac{1}{2} - \frac{3x_2}{4} \right)^2 \left\{ 26 - 8x_1 + 12 \left( \frac{x_1}{4} + \frac{1}{2} \right)^2 + 12x_2 - (9x_1 + 18) \left( \frac{x_2}{4} + \frac{1}{2} \right) + 27 \left( \frac{x_2}{4} + \frac{1}{2} \right)^2 \right\} \right].$$

The Goldprice function is a significantly more complicated function than the one considered in the first example. Let the contour of interest be  $S(1.5 * 10^5)$ . The contour at this height is not a contiguous curve, but the implicit function method we have used can easily extract the corresponding iso-surface.

Because the underlying response surface is known to be fairly complicated, we investigate the methodology with more design points ( $n = 35$  and  $n = 40$ ) than in the previous example. In addition, we consider a variety of choices for  $n_0$ .

Consider a single application of the sequential approach with  $n_0 = 12$  and  $n = 40$ . Figure 4(a) shows the estimated contour obtained starting with a random Latin hypercube design. The first new computer trial is selected in a region of high variability near the estimated contour segment. Acting simultaneously on the sparsely sampled regions and in the neighborhood of the contour, the criterion forces the algorithm to make a trade-off between sampling locations. As a result, some of the new trials are sampled in the neighborhood of the contour, while others are in sparsely sampled regions. This avoids both the under-detection of the contour (bottom right of Figure 4(b)) and the inclusion of a false contour segment (bottom left of Figure 4(c)). The final estimate of the contour with 28 additional trials is shown in Figure 4(d).

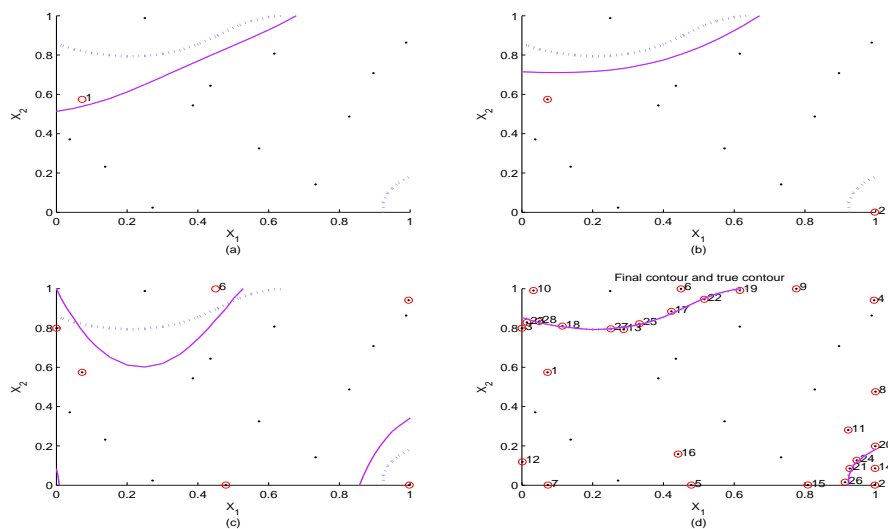


Figure 4: Illustration of sequential approach for contour estimation ( $n = 40$  and  $n_0 = 12$ ). The dotted curve refers to the true contour, the solid curve to the estimated contour and the dotted points refer to the initial design points.

**Remark:** The illustration in Example 1 may have given the impression that most of the new trials are forced near boundaries of the design region. This is not necessarily the case. Note that the initial Latin hypercube design does a relatively good job of sampling in the interior of the design space. So, for this example (Example 2), some points are chosen near the boundaries where there is very little sampling and some of the additional trials are in the neighborhood of the contour. In general, most space filling designs force points into the interior of the design region, and relatively few near the boundary. Therefore, it is not surprising that some points are sampled near the boundary in our setting.

Similar to Example 1, a simulation study, with 500 different starting designs, is conducted to compare the new methodology with the simpler approach. The simulation results are summarized in Table 2 for  $n = 35$  and 40, with  $n_0 = 5, 10, \dots, 30$ .

Table 2: Results of the simulation study for comparing the performance of the sequential approach and single stage Latin Hypercube sampling, separated by  $n = 35, 40$  and different choices of initial run size  $n_0$ .

		$n_0 = 5, k = 30 (n = 35)$			$n_0 = 5, k = 35 (n = 40)$		
Measure	LHC	Sequential	R.E.	LHC	Sequential	R.E.	
$M_1$	0.4573	0.1984	2.30	0.1968	4.14e-04	475.36	
$M_2$	0.1048	0.0560	1.87	0.0514	0.0056	9.17	
$M_3$	0.4468	0.2336	1.91	0.2303	0.0271	8.49	
		$n_0 = 10, k = 25 (n = 35)$			$n_0 = 10, k = 30 (n = 40)$		
Measure	LHC	Sequential	R.E.	LHC	Sequential	R.E.	
$M_1$	0.4573	0.0024	190.54	0.1968	2.38e-04	826.89	
$M_2$	0.1048	0.0157	6.67	0.0514	0.0044	11.68	
$M_3$	0.4468	0.0572	7.81	0.2303	0.0211	10.91	
		$n_0 = 15, k = 20 (n = 35)$			$n_0 = 15, k = 25 (n = 40)$		
Measure	LHC	Sequential	R.E.	LHC	Sequential	R.E.	
$M_1$	0.4573	0.0020	228.65	0.1968	2.26e-04	870.79	
$M_2$	0.1048	0.0149	7.03	0.0514	0.0043	11.95	
$M_3$	0.4468	0.0574	7.78	0.2303	0.0206	11.18	
		$n_0 = 20, k = 15 (n = 35)$			$n_0 = 20, k = 20 (n = 40)$		
Measure	LHC	Sequential	R.E.	LHC	Sequential	R.E.	
$M_1$	0.4573	0.0022	207.86	0.1968	2.31e-04	851.94	
$M_2$	0.1048	0.0162	6.47	0.0514	0.0044	11.68	
$M_3$	0.4468	0.0607	7.36	0.2303	0.0210	10.96	
		$n_0 = 25, k = 10 (n = 35)$			$n_0 = 25, k = 15 (n = 40)$		
Measure	LHC	Sequential	R.E.	LHC	Sequential	R.E.	
$M_1$	0.4573	0.0209	21.88	0.1968	2.77e-04	710.47	
$M_2$	0.1048	0.0196	5.34	0.0514	0.0044	11.68	
$M_3$	0.4468	0.0785	5.69	0.2303	0.0201	11.45	
		$n_0 = 30, k = 5 (n = 35)$			$n_0 = 30, k = 10 (n = 40)$		
Measure	LHC	Sequential	R.E.	LHC	Sequential	R.E.	
$M_1$	0.4573	0.1108	4.13	0.1968	3.14e-04	625.51	
$M_2$	0.1048	0.0371	2.82	0.0514	0.0053	9.70	
$M_3$	0.4468	0.1540	2.90	0.2303	0.0246	9.36	

A quick glance at Table 2 reveals that all the R.E.'s are larger than 1, and in all the cases, the

proposed approach performs better than using a single random Latin hypercube design. Indeed, for most cases, the R.E.'s are significantly larger than 1. In Example 1, the function is fairly simple and both approaches are bound to do reasonably well. However, for this more complicated function, a more judicious choice of design points is required. In particular, for the  $n = 40$ , the proposed approach is performing about 10 times better than the Latin hypercube design approach in two metrics ( $M_2$  and  $M_3$ ). For the  $n = 35$  case, the proposed approach is also performing much better than the naive approach. In general, we have found that the proposed approach provides substantial improvement over single stage designs when the underlying contour is relatively complex. Interestingly, the dimensionality of the problem does not seem to impact the relative efficiency nearly as much.

As mentioned before,  $M_1$  is less sensitive to large departures, and, in this case, converges much faster than the other two measures ( $M_2$  and  $M_3$ ). For both  $n = 35$  and  $n = 40$ , a close look at the R.E.'s in Table 2 shows that they have an increasing trend as  $n_0$  is increasing to 15, and then a decreasing trend as  $n_0$  further increases to 30 runs. This simulation study suggests that  $n_0 \approx 15$  will be a good choice.

In general, we recommend that choosing  $n_0$  to be around 25% – 33% of the experimental budget should be a good choice in practice. Starting with very few trials may not be able to capture the overall nature of the surface. On the other hand, if  $n_0$  is too large, much of the resources will have been used in locations which do not help in contour estimation and the benefits of the sequential approach are not realized.

**Example 3:** Consider the **Levy function** (Levy and Montalvo 1985) written as,

$$f(x_1, \dots, x_6) = \sin^2(3\pi x_1) + \sum_{i=1}^5 (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1})) + (x_6 - 1)(1 + \sin^2(2\pi x_6)),$$

with  $x_i \in [-5, 5]$  for  $i = 1, \dots, 6$ . This function is frequently used as a test function in the computer experiment literature (e.g., Santner et al. 2003). Further, suppose for illustration purposes that the contour of interest is  $S(140)$ .

Table 3 shows the performance of the proposed methodology compared to a single stage random Latin hypercube design. Again, the sequential strategy outperforms the random Latin Hypercube design. We should add that as discussed earlier, if the underlying contour is relatively simple (which is the case here), the gains using the proposed algorithm will be less substantial regardless of dimensionality.

For this larger size example, we take a slight detour to discuss some computational performance issues. To optimize the expected improvement function in our examples, we use a genetic algorithm

Table 3: Results of the simulation study for comparing the performance of the sequential approach and single stage Latin Hypercube sampling, separated by  $n = 60, 80$  and different choices of initial run size  $n_0$ .

		$n_0 = 15, k = 45 (n = 60)$			$n_0 = 15, k = 65 (n = 80)$		
Measure		LHC	Sequential	R.E.	LHC	Sequential	R.E.
$M_1$		1.0424	0.7313	1.42	0.9848	0.5485	1.80
$M_2$		0.4980	0.3958	1.26	0.4804	0.3352	1.43
$M_3$		0.9722	0.8712	1.12	0.9602	0.8397	1.14
		$n_0 = 30, k = 30 (n = 60)$			$n_0 = 30, k = 50 (n = 80)$		
Measure		LHC	Sequential	R.E.	LHC	Sequential	R.E.
$M_1$		1.0424	0.7669	1.36	0.9848	0.5826	1.69
$M_2$		0.4980	0.3942	1.26	0.4804	0.3539	1.36
$M_3$		0.9722	0.8819	1.10	0.9602	0.8544	1.12
		$n_0 = 45, k = 15 (n = 60)$			$n_0 = 45, k = 35 (n = 80)$		
Measure		LHC	Sequential	R.E.	LHC	Sequential	R.E.
$M_1$		1.0424	0.9405	1.11	0.9848	0.6909	1.43
$M_2$		0.4980	0.4508	1.10	0.4804	0.3872	1.24
$M_3$		0.9722	0.9405	1.05	0.9602	0.8620	1.11

to obtain starting values, which are then used as inputs to the packaged optimizers (global branch-and-bound algorithm; Matlab Optimization Toolbox function *fmincon*). For the 6-dimensional Levy function with  $n_0 = 30$  and  $k = 50$ , the average time taken by the genetic algorithm to get a starting value is approximately 56 seconds and both packaged optimizers take approximately 0.08 seconds to converge to the optimum. For the 2-dimensional Goldprice function with  $n_0 = 12$  and  $k = 28$ , the genetic algorithm takes approximately 6 seconds on average and the average time taken by both the packaged optimizers are less than 0.1 seconds. The averaging of time is based on 100 simulations and the computation is done on a Pentium(R) 4 processor machine running Windows XP. So, the optimization of the expected improvement is relatively fast especially considering the amount of time needed for expensive computer models.

## 5 Queueing Example - Revisited

A fundamental issue in any queueing system is that of stability. Roughly speaking, the system is said to be stable if the expected delay is finite. We restrict our attention to the input region,  $\chi$ , where the system is stable (henceforth called the *stability region*).

When the input rates are high, simulation of the system can be quite costly. Fortunately, we have at our disposal the results of a large computer experiment (Bambos and Michailidis 2004) conducted on a grid of points in the stability region. We evaluate the proposed methodology on

this grid of points. Figure 5 shows the available design points in  $\chi$ .

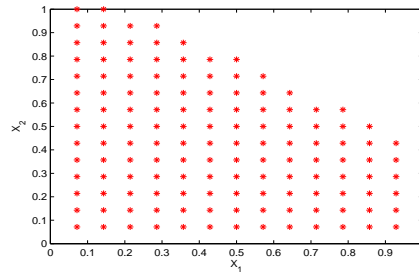


Figure 5: Available design points in the stability region for the queuing example.

Since the design region is not rectangular, and we have finitely many grid points to choose the trials from, we use a minimax design (Johnson et al. 1990; John et al. 1995) rather than a Latin hypercube design as our starting design. Also, only the trials lying on the lattice points in Figure 5 are used. We consider estimating the contour at delay of  $a = 0.75$  (i.e.  $S(0.75)$ ). For this illustration, the starting design is a minimax one with  $n_0 = 5$ .

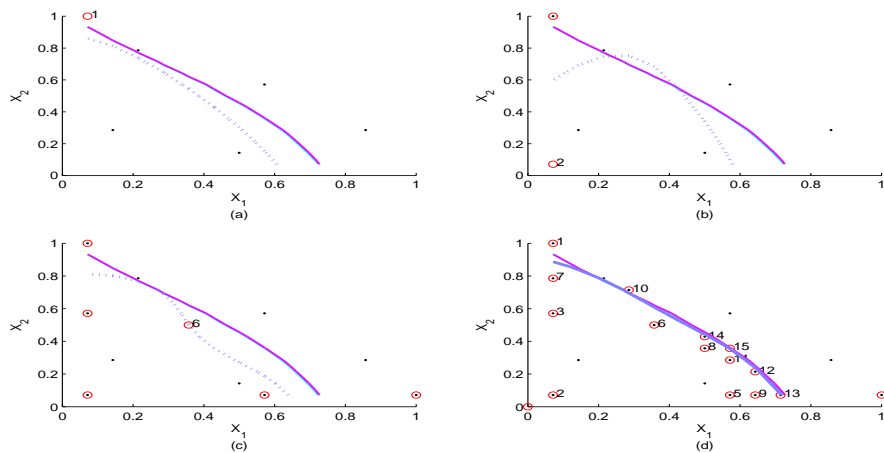


Figure 6: Illustration of the sequential approach for the contour estimation ( $n = 20$  and  $n_0 = 5$ ) for the queuing example. The solid curve refers to the true contour, the dotted curve to the estimated contour and the dotted points refer to the initial design points.

Figure 6(a) shows the initial 5-point minimax design as well as the true (solid curve) and the estimated contours (broken curve). Figures 6(a) and 6(b) show that the new trials are off the estimated contour to minimize the overall variability of the surface and then almost all of the additional trials are in the neighborhood of the estimated contour. Figure 6(d) displays the final contour estimate after adding 15 new trials.

Since the true function is unknown in practice, one requires different *lack of fit* measures for study-

ing the behavior of convergence for the estimated contours. Replacing  $C_t$  with  $C_{n_0,k-1}$  in equations (10), (11) and (12), we get a similar set of discrepancy measures which can be used for diagnostic purposes. That is, the current estimated contour is compared to the previous estimate. The interpretation of these measures will be slightly different from the discrepancy measures defined earlier. Now, when  $M_1$  is close to zero, this implies that the two contours  $C_{n_0,k}$  and  $C_{n_0,k-1}$  are almost the same. Therefore, in the following lack of fit plots, it is preferable to see the values of these discrepancy measures taking values consistently close to zero. If the discrepancies stay in the neighborhood of zero with further additional trials, we take this as evidence that the contour estimate has stabilized.

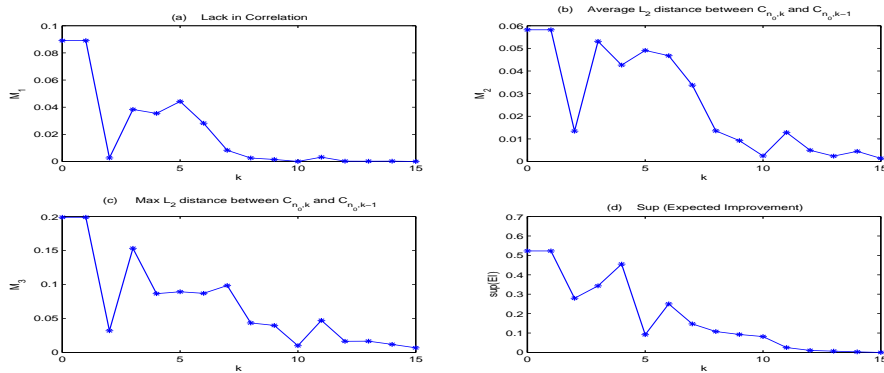


Figure 7: Lack of fit plots illustrating the improvement in the information of the contour as a result of the sequential addition of trials for the queuing example. These plots refer to discrepancy measures between  $C_{n_0,k}$  and  $C_{n_0,k-1}$ .

Figure 7 shows the discrepancy measures' plots for this example. As one would expect, the plots have a decreasing trend. Figures 7(b) and 7(d) show clearly that the last few additional trials are not causing large changes in the contour estimates. Because  $M_2$  and  $M_3$  tend to be more sensitive to large departures, it takes more trials for these discrepancies to stabilize near zero.

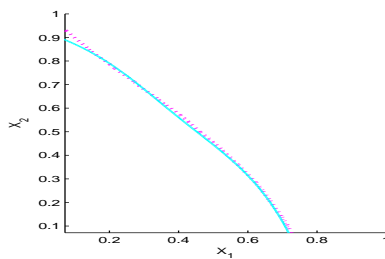


Figure 8: Final estimate of the contour plotted using implicit function obtained from the GASP model. The solid curve refers to the estimated contour and the dotted curve to the true contour.

Finally, the contour extracted from the final surface is shown in the Figure 8. The estimated contour closely matches the contour extracted by using a model, based on the responses at all of the lattice points.

## 6 Concluding Remarks

In this article, we have developed sequential methodology for estimating a contour from a complex computer code. The approach uses a stochastic process model as a surrogate for the computer simulator. The stochastic process model and associated uncertainty are integral components in the new criterion used to identify the computer trials aimed specifically at improving the contour estimate. Our examples demonstrate that the proposed approach can significantly outperform the single stage designs, particularly for more complex response surfaces.

Future work includes simultaneous investigation of multiple contours from complex computer models. In this setting, one is interested in a variety of system states and the associated boundary between the states.

## Acknowledgments

The authors would like to thank the AE and two anonymous referees for many useful comments and suggestions.

## Appendix A (Proof of main results)

**Lemma 1:** *If  $f_n(\cdot|x)$ 's are non-negative functions  $\forall n \geq 1$ , and  $\forall x \in \chi$  then*

$$\sup_{x \in \chi} f_n(\cdot|x) \rightarrow 0 \quad \Rightarrow \quad \lim_{n \rightarrow \infty} \sup_{x \in \chi} E[f_n(y|x)] = 0.$$

**Proof:** Since  $f_n(\cdot|x) \geq 0$ ,  $\forall n \geq 1$ , and  $\forall x \in \chi$ , then  $0 \leq f_n(y|x) \leq \sup_{x \in \chi} f_n(y|x)$ . Taking the expectation over the inequalities, we get  $0 \leq E[f_n(y|x)] \leq E[\sup_{x \in \chi} f_n(y|x)] \forall n \geq 1$  and  $\forall x \in \chi$ . Since  $E[f_n(y|x)]$  is less than and equal to  $E[\sup_{x \in \chi} f_n(y|x)]$  for all  $x$ ,  $\sup_{x \in \chi} E[f_n(y|x)]$  is also less than and equal to  $E[\sup_{x \in \chi} f_n(y|x)]$  (for details of these steps see Rudin 1987). Suppose  $\sup_{x \in \chi} f_n(\cdot|x) \rightarrow 0$  as  $n \rightarrow \infty$ . Since  $E[\sup_{x \in \chi} f_n(y|x)] \rightarrow 0$ , then  $\sup_{x \in \chi} E[f_n(y|x)] \rightarrow 0$ . Consequently, one can see that  $\lim_{n \rightarrow \infty} \sup_{x \in \chi} E[f_n(y|x)] = 0$ .  $\square$

**Theorem 1:** *Under the correlation structure defined in equation (2) and the expected improvement function defined in equation (8),*

$$\lim_{n \rightarrow \infty} \sup_{x \in \chi} E(I(x)) = 0.$$

More precisely,  $\sup_{x \in \mathcal{X}} E(I(x)) = O\left(\frac{1}{\log(n)}\right)$  and therefore converges to zero in limit.

**Outline of the Proof:** Recall that  $I(x) = \alpha^2 s^2(x) - \min\{(y(x) - a)^2, \alpha^2 s^2(x)\}$ , and also  $0 \leq I(x) \leq \alpha^2 s^2(x)$ . Let  $I_n(x)$  and  $s_n^2(x)$  denote  $I(x)$  and  $s^2(x)$ , respectively, based on a sample of size  $n$ . Let  $I_n(x)$  be  $f_n(y|x)$  in the lemma. Then,  $\sup_{x \in \mathcal{X}} f_n(y|x) = \sup_{x \in \mathcal{X}} I_n(x) \leq \sup_{x \in \mathcal{X}} \alpha^2 s_n^2(x)$ . From Lemma 1 it is sufficient to show that  $\sup_{x \in \mathcal{X}} s_n^2(x) \rightarrow 0$  in order to prove the theorem.

Denote  $s_n^2(x) = \sigma_z^2 \left(1 - r' R_n^{-1} r + \frac{(1 - 1_n' R_n^{-1} r)^2}{1_n' R_n^{-1} 1_n}\right)$ , where  $R_n$  is the correlation matrix  $R$  based on  $n$  trials and  $r$  is  $\text{corr}(x, (x_1, \dots, x_n)')$ . We show that  $\sup_{x \in \mathcal{X}} s_n^2(x) \rightarrow 0$ , using three results. These results can be easily verified by mathematical induction on  $n$  by noting that (i) as  $n \rightarrow \infty$ ,  $r \rightarrow R_i$  for some  $i \in \{1, \dots, n\}$ ; and (ii) as a consequence of (i) as  $n \rightarrow \infty$ ,  $R^{-1} r \rightarrow R^{-1} R_i = e_i$  (here,  $e_i$  is the unit vector with 1 at the  $i^{\text{th}}$  place and 0 elsewhere). The three useful results are:

1.  $1_n' R_n^{-1} r \rightarrow 1$ , and therefore  $(1 - 1_n' R_n^{-1} r)^2 \rightarrow 0$ .
2.  $r' R_n^{-1} r \rightarrow 1$ , which implies  $1 - r' R_n^{-1} r \rightarrow 0$ .
3.  $1_n' R_n^{-1} 1_n \geq \log(n)$ , and therefore  $\frac{1}{1_n' R_n^{-1} 1_n} \leq \frac{1}{\log(n)}$ .

The three results imply that  $s_n^2(x) \leq C \frac{1}{\log(n)}$  (for an appropriate positive constant  $C$ ), which further implies that  $s_n^2(x) = O\left(\frac{1}{\log(n)}\right)$ . Therefore  $s_n^2(x) \rightarrow 0$  as  $n \rightarrow \infty$ . Furthermore then implies that the  $\sup_{x \in \mathcal{X}} s_n^2(x) \rightarrow 0$ . So, from Lemma 1, we have  $\lim_{n \rightarrow \infty} \sup_{x \in \mathcal{X}} E(I_n(x)) = 0$ .  $\square$

## Appendix B (Derivation of the Expected Improvement)

Let  $I(x) = \epsilon^2(x) - \min\{(y(x) - a)^2, \epsilon^2(x)\}$  as defined in equation (7) and  $y(x) \sim N(\hat{y}(x), s^2(x))$ , then for some positive constant  $\alpha$  and  $\epsilon(x) = \alpha s(x)$ :

$$\begin{aligned} E[I(x)] &= [\alpha^2 s^2(x) - (\hat{y}(x) - a)^2] \left[ \Phi\left(\frac{a - \hat{y}(x)}{s(x)} + \alpha\right) - \Phi\left(\frac{a - \hat{y}(x)}{s(x)} - \alpha\right) \right] \\ &+ 2(\hat{y}(x) - a)s^2(x) \left[ \phi\left(\frac{a - \hat{y}(x)}{s(x)} + \alpha\right) - \phi\left(\frac{a - \hat{y}(x)}{s(x)} - \alpha\right) \right] \\ &- \int_{a - \alpha s(x)}^{a + \alpha s(x)} (y - \hat{y}(x))^2 \phi\left(\frac{y - \hat{y}(x)}{s(x)}\right) dy. \end{aligned}$$

**Proof:** Given that,  $I(x) = \epsilon^2(x) - \min\{(y(x) - a)^2, \epsilon^2(x)\}$ , and thus,

$$\begin{aligned} \min\{(y(x) - a)^2, \epsilon^2(x)\} &= (y(x) - a)^2 \quad \text{for } y(x) \in (a - \epsilon(x), a + \epsilon(x)) \\ &= \epsilon(x)^2 \quad \text{for } y(x) \in (-\infty, a - \epsilon(x)] \cup [a + \epsilon(x), \infty) \end{aligned}$$

Therefore,

$$\begin{aligned} E[I(x)] &= - \int_{a - \epsilon(x)}^{a + \epsilon(x)} (y - a)^2 \phi\left(\frac{y - \hat{y}(x)}{s(x)}\right) dy \\ &+ \epsilon(x)^2 \left[ \Phi\left(\frac{a + \epsilon(x) - \hat{y}(x)}{s(x)}\right) - \Phi\left(\frac{a - \epsilon(x) - \hat{y}(x)}{s(x)}\right) \right]. \end{aligned} \quad (13)$$

Considering the first term only,

$$\begin{aligned}
& \int_{a-\epsilon(x)}^{a+\epsilon(x)} (y-a)^2 \phi\left(\frac{y-\hat{y}(x)}{s(x)}\right) dy \\
&= \int_{a-\epsilon(x)}^{a+\epsilon(x)} (y-\hat{y}(x))^2 \phi\left(\frac{y-\hat{y}(x)}{s(x)}\right) dy + (\hat{y}(x)-a)^2 \left[ \Phi\left(\frac{a+\epsilon(x)-\hat{y}(x)}{s(x)}\right) - \Phi\left(\frac{a-\epsilon(x)-\hat{y}(x)}{s(x)}\right) \right] \\
&\quad + 2(a-\hat{y}(x))s^2(x) \left[ \phi\left(\frac{a+\epsilon(x)-\hat{y}(x)}{s(x)}\right) - \phi\left(\frac{a-\epsilon(x)-\hat{y}(x)}{s(x)}\right) \right] \tag{14}
\end{aligned}$$

Therefore, after substituting  $\epsilon(x) = \alpha s(x)$  and equation (14) into (13),  $E[I(x)]$  becomes

$$\begin{aligned}
E[I(x)] &= [\alpha^2 s^2(x) - (\hat{y}(x) - a)^2] \left[ \Phi\left(\frac{a-\hat{y}(x)}{s(x)} + \alpha\right) - \Phi\left(\frac{a-\hat{y}(x)}{s(x)} - \alpha\right) \right] \\
&\quad + 2(\hat{y}(x) - a)s^2(x) \left[ \phi\left(\frac{a-\hat{y}(x)}{s(x)} + \alpha\right) - \phi\left(\frac{a-\hat{y}(x)}{s(x)} - \alpha\right) \right] \\
&\quad - \int_{a-\alpha s(x)}^{a+\alpha s(x)} (y-\hat{y}(x))^2 \phi\left(\frac{y-\hat{y}(x)}{s(x)}\right) dy \quad \square
\end{aligned}$$

## Appendix C (Convergence of Discrepancy Measures)

**Result 1 (Convergence of  $M_3$ ):** Fixing  $n_0$ , let  $C_{n_0,k}$  be the contour estimate extracted from the surface estimated after adding  $k$  new design points and let  $M_3 = \sup\{d(x, C_{n_0,k-1}) : x \in C_{n_0,k}\}$ , where  $d$  is any standard metric (we use the  $L_2$  metric), then  $M_3 \rightarrow 0$ , as  $k \rightarrow \infty$ .

**Proof:** Recall that  $d(x, C_{n_0,k-1}) = \inf\{d(x, y) : y \in C_{n_0,k-1}\}$ . It is sufficient to show that, for every fixed  $x \in C_{n_0,k}$ ,  $d(x, C_{n_0,k-1}) \rightarrow 0$ . First discretize  $C_{n_0,k}$  into  $m$  points (i.e., draw  $m$  points from the contour  $C_{n_0,k}$  so that it represents the contour). Let

$$C_{n_0,k} = \{x_i^k : i = 1, \dots, m\} \quad \text{and} \quad C_{n_0,k+1} = \{x_i^{k+1} : i = 1, \dots, m\}.$$

See that  $x_j^k \in \chi (= [0, 1]^d)$ . Discretization of  $C_{n_0,k+1}$  is based on  $C_{n_0,k}$ . That is,  $x_i^{k+1}$  is chosen such that  $d(x_i^k, x_i^{k+1}) < d(x_i^k, x_j^{k+1}) \forall j \neq i$ , and therefore,  $d(x_i^k, C_{n_0,k+1}) = d(x_i^k, x_i^{k+1})$ . Thus, we have to show that, for every  $i \in \{1, \dots, m\}$ ,  $d(x_i^k, x_i^{k+1}) \rightarrow 0$ , as  $k \rightarrow \infty$ . Suppose  $C_{n_0,k} \rightarrow C_t$ , as  $k \rightarrow \infty$  (here,  $C_t$  will be the underlying true contour which we are trying to estimate), and  $x_i^t$  is the homologous point on  $C_t$  corresponding to  $x_i^k$  on  $C_{n_0,k}$ . Then using triangle inequality,  $0 \leq d(x_i^k, x_i^{k+1}) \leq d(x_i^k, x_i^t) + d(x_i^t, x_i^{k+1})$ . Note that,  $d(x_i^k, x_i^t) \rightarrow 0$  as  $k \rightarrow \infty$ , and also  $d(x_i^t, x_i^{k+1}) \rightarrow 0$  as  $k \rightarrow \infty$ . Therefore for every fixed  $i \in \{1, \dots, m\}$ ,

$$\begin{aligned}
& d(x_i^k, x_i^{k+1}) \rightarrow 0 \quad \text{as } k \rightarrow \infty \\
& \Rightarrow d(x_i^k, C_{n_0,k+1}) \rightarrow 0 \quad \text{as } k \rightarrow \infty
\end{aligned}$$

$$\begin{aligned} &\Rightarrow \max\{d(x_i^k, C_{n_0, k+1}) : i = 1, \dots, m\} \rightarrow 0 \quad \text{as } k \rightarrow \infty \\ &\Rightarrow \sup\{d(x, C_{n_0, k+1}) : x \in C_{n_0, k}\} \rightarrow 0 \quad \text{as } k \rightarrow \infty. \end{aligned}$$

Consequently, it has been shown that  $M_3 \rightarrow 0$ .  $\square$

**Result 2 (Convergence of  $M_2$ ):** Let  $C_{n_0, k}$  and  $C_{n_0, k+1}$  be the sets as defined in Result 1, then as  $k \rightarrow \infty$

$$M_2 = \frac{1}{|C_{n_0, k}|} \sum_{x \in C_{n_0, k}} d(x, C_{n_0, k+1}) \rightarrow 0.$$

**Proof:** Since, in Result 1, it has been shown that  $\sup\{d(x, C_{n_0, k+1}) : x \in C_{n_0, k}\} \rightarrow 0$  as  $k \rightarrow \infty$ , it is simple to see that  $M_2 \rightarrow 0$  as  $k \rightarrow \infty$ .  $\square$

**Result 3 (Convergence of  $M_1$ ):** Define  $M_1$  as in (10). Then as  $k \rightarrow \infty$ ,  $M_1 \rightarrow 0$ .

**Proof:** To prove the result, it is enough to show that  $\text{corr}(x^{lk}, x^{lt}) \rightarrow 1$ , as  $k \rightarrow \infty$ . Using the same notation as used in the previous result, see that

$$\text{corr}(x^{lk}, x^{lt}) = \frac{\sum_{i=1}^m (x_i^{lk} - \bar{x}^{lk})(x_i^{lt} - \bar{x}^{lt})}{\sqrt{\sum_{i=1}^m (x_i^{lk} - \bar{x}^{lk})^2} \sqrt{\sum_{i=1}^m (x_i^{lt} - \bar{x}^{lt})^2}}.$$

Since,  $C_{n_0, k} \rightarrow C_t$  as  $k \rightarrow \infty$ , then  $x_i^{lk} \rightarrow x_i^{lt}$  as  $k \rightarrow \infty$  for all  $i$  and for all  $l$ . Hence,  $\text{corr}(x^{lk}, x^{lt}) \rightarrow 1$ , as  $k \rightarrow \infty$ , and, therefore,  $M_1 \rightarrow 0$  as  $k \rightarrow \infty$ .  $\square$

Note that the convergence of  $M_i$ 's relies on the distance between  $x_i^k$  and  $x_i^t$ . Supposing the contour estimates are same, if  $m$  is very small, the location of  $x_i^k$  and  $x_i^t$  may change which can cause variation in  $M_i$ . To avoid this unnecessary variation in  $M_i$ 's, the choice of  $m$  should be significantly large to efficiently approximate the contours and the associated  $M_i$ 's.

## References

- Andre, J., Siarry, P., and Dognon, T. (2000), "An Improvement of the Standard Genetic Algorithm Fighting Premature Convergence," *Advances in Engineering Software*.
- Balakrishnan, V., Boyd, S. and Balemi, S. (1991), "Branch and Bound Algorithm for Computing the Minimum Stability Degree of Parameter-dependent Linear Systems." *Int. J. of Robust and Nonlinear Control*, 1(4):295-317.
- Bambos, N. and Michailidis, G. (2004), "Queueing and Scheduling in Random Environments," *Advances in Applied Probability*, 36, 293-317

- Banerjee, S. and Gelfand, A.E. (2005), "Boundary Analysis: Significance and Construction of Curvilinear Boundaries", *Journal of the American Statistical Association* (to appear).
- Barbujani G., Oden N.L., Sokal R.R. (1989), "Detecting Regions of Abrupt Change in Maps of Biological Variables", *Systematic Zoology*, 38,376-389.
- Brault, P., and Mounier, H. (2001), "Automated, Transformation Invariant, Shape Recognition Through Wavelet Multiresolution," *SPIE01 Proceedings of International Society for Optical Engineering, San Diego*.
- Dixon, L.C.W., and Szego, G. P. (1978), "The Global Optimization Problem: An Introduction," *In Towards Global Optimization 2 (Edited by L.C.W. Dixon and G.P. Szego), 1-15. North Holland, Amsterdam*.
- Fang, K. T., Lin, D. K. J., Winker, P., and Zhang, Y. (2000), "Uniform Design: Theory and Applications," *Technometrics*, 42, 237-248.
- Fletcher, R. (1987), "Practical Methods of Optimization," *John Wiley and Sons*.
- Henderson, C. R. (1975), "Best Linear Unbiased Estimation and Prediction Under a Selection Model," *Biometrics*, 31, 423-447.
- Holland, J. H. (1975), "Adaptation in Natural and Artificial Systems," *Ann Arbor: The University of Michigan Press*.
- John, P. W. M., Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1995), "Minimax Distance Designs in Two-Level Factorial Experiments," *Journal of Statistical Planning and Inference*, 44, 249-263.
- Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990), "Minimax and Maximin Distance Designs," *Journal of Statistical Planning and Inference*, 26, 131-148.
- Jones, D. H., and Jin, Z. (1994), "Optimal Sequential Designs for On-Line Item Estimation," *Psychometrika*, 59, 57-75.
- Jones, D., Schonlau, M., and Welch, W. (1998), "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, 13, 455-492.
- Levy, A. V., and Montalvo, A. (1985), "The Tunnelling Algorithm for the Global Minimization of Functions", *SIAM Journal of Scientific and Statistical Computing*, 6, 15-29.
- Lu, H. and Carlin, B.P. (2005), "Bayesian Areal Wombling for Geographical Boundary Analysis," *Geographical Analysis*, 37, 265-285.
- Mandal, A., Wu, C.F.J., and Johnson, K. (2006), "SELC: Sequential Elimination of Level Combinations by means of Modified Genetic Algorithms", *Technometrics*, 48(2), 273-283.

- 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(2), 239-245.
- Murty, K. G., (1995), "Operations Research: Deterministic Optimization Models", *Prentice Hall*.
- O'Connell, M. and Wolfinger, R. (1997), "Spatial Regression Models, Response Surfaces and Process Optimization," *Journal of Computational and Graphical Statistics*, 6, 224-241.
- Owen, A. B. (1992), "Orthogonal arrays for computer experiments, integration and visualization," *Statistica Sinica*, 2, 439-452.
- Rao, C. R. (1972), "Estimation of Variance and Covariance Components in Linear Models," *Journal of the American Statistical Association*, 67, 112 -115.
- Rudin, W. (1987), "Real and Complex Analysis, Third Edition," *McGraw-Hill, Inc.*.
- Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409-423.
- Santner, T. J., Williams, B. J., and Notz, W. I. (2003), "The Design and Analysis of Computer Experiments," *Springer Verlag, New York*.
- Schittkowski, K. (1985), "NLQPL: A FORTRAN-Subroutine Solving Constrained Nonlinear Programming Problems," *Annals of Operations Research*, 5, 485-500.
- Sethian, J. A. (1999), "Level Set Methods and Fast Marching Methods," *Cambridge University Press*.
- Tang, B. (1993), "Orthogonal Array-Based Latin Hypercubes," *Journal of American Statistical Association*, 88, 1392-1397.
- Uhlir, K., and Skala, V. (2003), "Implicit Function Modelling System Comparison of C++ and C# Solutions," *proceedings of C# and .NET Technologies 2003 Int. Workshop, ISBN 80-90301-3-6, UNION Agency - Science Press, Plozen, Czech Republic*, 87-92.
- Uhlir, Karel. (2003), Doctoral Thesis on "Modelling Methods with Implicitly Defined Objects," *Technical Report No. DCSE/TR-2003-04*.
- Veltkamp, R. C. (2001), "Shape Matching: Similarity Measures and Algorithms," *Proc. Shape Modelling International Conference, Los Alamitos, IEEE*, 188-197.
- Warland, J. (1988), "Introduction to Queuing Networks," *Prentice Hall, Englewood Cliffs*.
- Wolfinger, R. D., Tobias, R. D., and Sall, J. (1994), "Computing Gaussian Likelihoods and Their Derivatives for General Linear Mixed Models," *SIAM Journal on Scientific Computing*, 15(6), 1294 -1310.