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*)

Computer simulation often is used to study complex physical and engineering processes. Although a computer simulator often can be viewed as an inexpensive way to gain insight into a system, it still can 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 a 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 also are addressed.

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

1. INTRODUCTION

Computer simulation frequently is used as a cost-effective means to study complex physical and engineering processes. Although a computer code (simulator) often can be viewed as an inexpensive way to gain insight into a system, it still can be computationally costly. Consequently, it is desirable to perform only a limited number of simulation trials. Thus investigators must 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, Welch, Mitchell, and Wynn (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 in which the goal is to fit a response surface or to optimize a process (see Santner, Willims, and Notz 2003 for an overview). Experiment plans, such as Latin hypercube designs (McKay, Beckman, and Conover 1979), orthogonal array–based Latin hypercube designs (Owen 1992; Tang 1993), space-filling designs (Johnson, Moore, and Ylvisaker 1990), uniform designs (Fang, Lin, Winker, and Zhang 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 the present work, the objective is to determine the system configurations that produce a specific output from a network queuing simulator. In contrast to the global objectives of fitting a response surface or optimization mentioned earlier, this is a more localized objective that requires an alternate design strategy to use the available computing resources most efficiently.

In this article we develop a sequential design methodology for estimating a contour (also called a level set or iso-surface) of a nondegenerate complex computer code. In the network queuing application that motivated this work, the problem is to estimate a contour of a globally or locally monotone response surface, where the contour identifies the boundary that distinguishes "good" performance from "bad" performance. Another application is the minimization of production costs subject to requirements on product quality. An important first step might be estimating a contour in the input space where the quality requirement is satisfied. Subsequent analysis then can take advantage of input cost information to minimize costs along the contour.

The problem of contour estimation bears some resemblance to problems in which wombling techniques have proven useful (Barbujani, Oden, and Sokal 1989; Banerjee and Gelfand 2005; Lu and Carlin 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 at which the change occurs. For the problem addressed here, interest lies in estimating a *prespecified* contour from an expensive computer simulator. 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. The methodology is applied to several examples in Section 4 to illustrate the performance, and the network queuing problem is revisited 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

© 2008 American Statistical Association and
the American Society for Quality
TECHNOMETRICS, NOVEMBER 2008, VOL. 50, NO. 4
DOI 10.1198/004017008000000541

work. Consider a queueing system comprising K first-in-first-out (FIFO) queues and a single server. There is a stochastic flow of jobs arriving at rate λ_k, $k = 1, \ldots, K$, to the queues (Warland 1988). The queues have infinite capacity buffers in which 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 multiproduct manufacturing system, where the modulated service rates are due to the availability of the necessary resources for production purposes.

In this article, we consider, for illustration purposes, a two-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 through a computer simulator; therefore, the computer model of interest is the one that simulates this queueing process. For this system, the computer code simulates 100,000 departures from the system with fixed λ. The first 10,000 are discarded from every simulation run, because they correspond to the warm-up period (burn-in) from a lightly loaded 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: (a) to ensure that the system has warmed up to a steady state for any λ, and (b) 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, λ, 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 likely will 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 λ. Even in this two-queue system, computer simulation can be quite costly when the input rates are high, taking up several hours per run. Consequently, one must select the simulation trials judiciously to efficiently estimate the contour of interest; that is, an experiment design strategy for contour estimation is needed.

3. METHODOLOGY AND ALGORITHM

We have developed new methodology for performing sequential computer trials to explore and estimate a contour. The proposed approach comprises two main components. First, 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. Second, a new criterion is proposed 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, Schonlau, and Welch 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 nonlinearity without adding complexity to the model. In our setting the GASP model is used as an inexpensive surrogate for the computer code that can be explored more cheaply. Next we present the formulation of the GASP model that has been found to be successful in our applications. The properties of this model are well known (see, e.g., Jones et al. 1998), and we illustrate its important features for the purpose of readability.

Let the ith input and output for the computer model be denoted by a d-dimensional vector, $x_i = (x_{i1}, \ldots, x_{id})$, and the univariate response, $y_i = y(x_i)$. 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 $y = y(X) = (y_1, y_2, \ldots, y_n)'$. The output of the simulator, $y(x_i)$, is modeled as

$$y(x_i) = \mu + z(x_i), \qquad i = 1, \ldots, n, \tag{1}$$

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

$$\begin{aligned} R_{ij} &= \mathrm{corr}(z(x_i), z(x_j)) \\ &= \prod_{k=1}^{d} \exp\{-\theta_k (x_{ik} - x_{jk})^{p_k}\} \qquad \forall i, j. \end{aligned} \tag{2}$$

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

Some aspects of this formulation are worth noting. First, 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 and Wolfinger 1997). Second, 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. Finally, the chosen covariance function is separable in the sense that it is the product of componentwise covariances. This is convenient insofar as it enables us to handle a large number of inputs, because each dimension requires only one parameter, θ_k.

The likelihood for the model is

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

If the covariance parameters, $\theta = (\theta_1, \ldots, \theta_d)$, are known, then estimates of μ and σ_z^2 can be written in a closed form,

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

(see Jones et al. 1998 for details). Substitution of these estimates in the actual likelihood (3) gives the "profile likelihood," which can be used to compute the likelihood estimates of the θ_i's (Rao 1972; Wolfinger, Tobias, and Sall 1994).

The model can now be used to estimate responses at any nonsampled point x^*. The best linear unbiased predictor (BLUP) for $y(x^*)$ is

$$\hat{y}(x^*) = \hat{\mu} + r' R^{-1} (y - \mathbf{1}_\mathbf{n}\hat{\mu}) \tag{5}$$

(see Henderson 1975 for details), with mean squared error

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

where $r = (r_1(x^*), \ldots, r_n(x^*))'$ and $r_i(x^*) = \operatorname{corr}(z(x^*), z(x_i))$ is as defined in (2). In practice, the parameters in (4) are replaced with the maximum likelihood (ML) estimates. Consequently, R and r are replaced with $\hat{R}$ and $\hat{r}$ in (5) and (6).

A quick glance at (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 unsampled points in (5), this implies that nearby sampled points will have more impact on the interpolation of predicted values than points farther away. Another consequence of this model is that the predicted response at sampled points will be exactly the observed response, with a 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, because it may concentrate most of the sampling effort in regions of the input space that 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 likely will have to be quite large 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, trials that lie exactly on the contour could be selected, 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, we would like to maximize the information provided by new computer trials in the neighborhood of the contour. In a spirit similar to the work of Jones et al. (1998), we propose an *improvement function* to help identify optimal computer trials. Define the improvement function as

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

where $\epsilon(x) = \alpha s(x)$ for some positive constant α and $y(x) \sim \mathrm{N}(\hat{y}(x), s^2(x))$. The term $\epsilon(x)$ defines a neighborhood around the contour that is a function of $s(x)$. This allows the radius of the 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, because the predicted variance tends to be large in areas of sparse sampling.

With these features of the improvement function in mind, we may be tempted to select design points that maximize $I(x)$ over the design space χ. But, for any unsampled 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 ϵ-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,

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

where $E[I(x)]$ is the expected improvement at $x \in \chi$ and $\phi(\cdot)$ and $\Phi(\cdot)$ are the standard normal probability density function (pdf) and cumulative distribution function (cdf). The derivation of (8) is given in Result C.1 of Appendix C. 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 that 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 ϵ-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 ϵ-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 ϵ-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 terms 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 plausibly may exist. The latter scenario 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 0, whereas the second term in the expression of $E[I(x)]$ supports exploration of regions of the input space that are further away from a and where the uncertainty of the predictor is high.

3.3 Contour Extraction

Exact isosurface (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 defining the contour estimate $S(a)$ is

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

which gives a compact, 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; for example, 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 $S(a) = \{x \in [0, 1]^d : y(x) = a\}$ is obtained from a fine grid on the design space.

3.4 Implementation

Our proposed approach involves first performing a relatively small design to obtain an initial estimate of the response surface, then 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 (5) is needed. This is done by first performing an experiment 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. As mentioned in Section 1, there are a number of choices for initial designs. 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 used, because of their ease of implementation and thus attractiveness to practitioners.

After performing the initial design, the GASP model in (1) is estimated using the ML 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 that maximizes the expected improvement (see the next section for details). A new computer experiment trial is performed at the location of the optimum of (8) and the response surface is reestimated. The approach continues in this fashion until the budget of runs is exhausted. Finally, the final contour estimate is extracted by the implicit function in (9). The procedure is summarized as follows:

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 that maximizes $E[I(x)]$ [shown in (8)], and perform a computer trial at this design point.
4. Update the data, that is, $X = [X'\ x_{max}]'$, $Y = [Y'\ \mathrm{y}(x_{max})]'$, and $n = n + 1$.
5. Iterate between steps 2 and 4 until the experimental 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 that of 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 of Balakrishnan, Boyd, and Balemi (1991). Furthermore, it can be shown that (8) is not monotonic on the design space, and thus 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, (8) could be maximized 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: the global branch-and-bound algorithm (Murty 1995) and 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 on 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 used (e.g., Holland 1975; Mandal, Wu, and Johnson 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 to optimize the objective function.
- Crossover. A pair of candidate solutions is crossed at $[d/2]$ randomly chosen locations; that is, 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 becoming trapped into local optima.

After the crossover and mutation steps, the best $500d$ solutions from the three populations are kept. The foregoing 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) among 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 $E[I(x)]$

We now discuss the convergence of the proposed approach. We begin with a useful lemma, which we use 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 nonnegative functions, then

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

The implications of the lemma are straightforward. Setting $f_n(y|x) = I(x)$, where $y|x \sim \mathrm{N}(\hat{y}(x), s^2(x))$, the following are equivalent:

$$\lim_{n\to\infty} \sup_{x\in\chi} E[I(x)] = 0 \qquad \text{and} \qquad \lim_{n\to\infty} \sup_{x\in\chi} E[f_n(y|x)] = 0.$$

Thus if it can be shown that $\sup_{x\in\chi} I(x) \to 0$, then this implies that $\lim_{n\to\infty} \sup_{x\in\chi} E[I(x)] = 0$; that is, as $n \to \infty$, the contour is known perfectly, and we cannot improve our knowledge.

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

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

More precisely, $\sup_{x\in\chi} E(I(x)) = O(\frac{1}{\log(n)})$ and thus converges to 0 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. Because $\sup_{x\in\chi} I(x) \leq \sup_{x\in\chi} \alpha^2 s^2(x)$, proving that $\sup_{x\in\chi} s^2(x) \to 0$ as $n \to \infty$ is sufficient for the convergence of the algorithm.

A lurking mathematical problem is determining the sufficient number of trials to get a good approximation of the underlying contour. The theorem implies that as more trials are added, the supremum of the expected improvement goes to 0 at 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: prespecify a value of the expected improvement and terminate the algorithm when the supremum of the expected improvement is consistently lower than this prespecified value.

4. EXAMPLES

To illustrate the approach we present several examples. Before proceeding, we note that the experimenters faces the choice of the initial runsize of the experiment, n_0. In each of the examples, several choices of n_0 are considered, and the performance is evaluated. We also present several plots that 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 second-order polynomial model, $y(x_1, x_2) = x_1 + x_2 + x_1 x_2$, with $x_1, x_2 \in [0, 1]$ and contour of interest $S(1.5)$. Despite the relatively simple nature of the underlying response surface, a few choices remain. First, the experimenter must choose the total runsize, n, of the experiment. In practice, the runsize budget may be fixed beforehand because of the cost of running the code. Next, the experimenter must choose the fraction of the experimental budget, and thus the initial runsize, n_0, that will be assigned to an initial experiment design. Roughly 10–20 points per dimension is recommended as a reasonable choice for the response surface estimation (Jones et al. 1998). In the same spirit, we consider runsizes of $n = 20$ and 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 runsizes 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 one and five additional trials 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 only one additional design point is added. Furthermore, note that only one design point (the tenth additional trial) is near the predicted contour. This demonstrates the importance of the second component of (8), which stipulates that sampling should be done not only near the contour, but also in regions where the prediction variance is high and the contour is plausible.

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

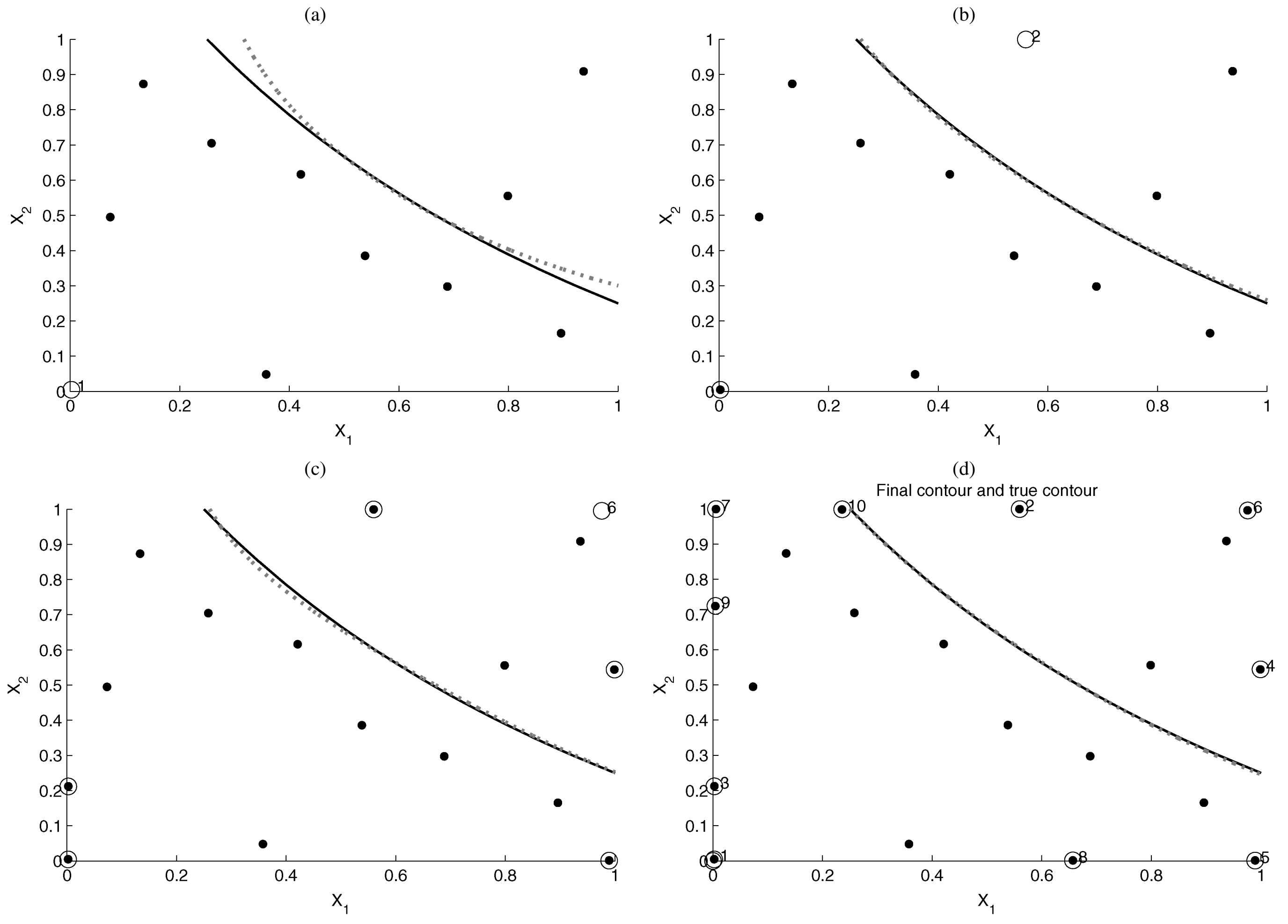

Figure 1. Illustration of the sequential approach for contour estimation ($n = 20$ and $n_0 = 10$). The solid curve represents the true contour, the dotted curve represents the estimated contour, and the dotted points represent to the initial design.

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 the addition of k additional trials. Thus, $C_{n,0}$ 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 obtained 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 necessary large values of m can be used to obtain a good approximation of the contours. As mentioned in Section 3.3, we 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 $\tilde{C}_{n_0,k} = \{x_1^k, \ldots, x_m^k\}$, where $x_i^k = (x_i^{1k}, \ldots, x_i^{dk})$. Similarly, denote the sample from C_t as $\tilde{C}_t = \{x_1^t, \ldots, x_m^t\}$.

The first discrepancy measure calculates the lack of correlation between the estimated and true contours, denoted by

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

Here x^{lk} and x^{lt} are the lth coordinate of $C_{n_0,k}$ and C_t. Thus $\mathrm{corr}(x^{lk}, x^{lt})$ is the correlation between x^{lk} and x^{lt}, where x_i^k is homologous to x_i^t for each $i = 1, \ldots, m$ and for each k. The second discrepancy measure is the average L_2 (Euclidean) distance between $C_{n_0,k}$ and C_t,

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

where $d(x, \tilde{C}_t) = \min\{\|x - y\|_2 : y \in \tilde{C}_t\}$. M_2 measures the average distance between two contours. Finally, the third discrepancy measure is the maximum L_2 distance between $C_{n_0,k}$ and C_t,

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

This discrepancy measure aims to measure maximum separation between two contours. This metric guards against the worst

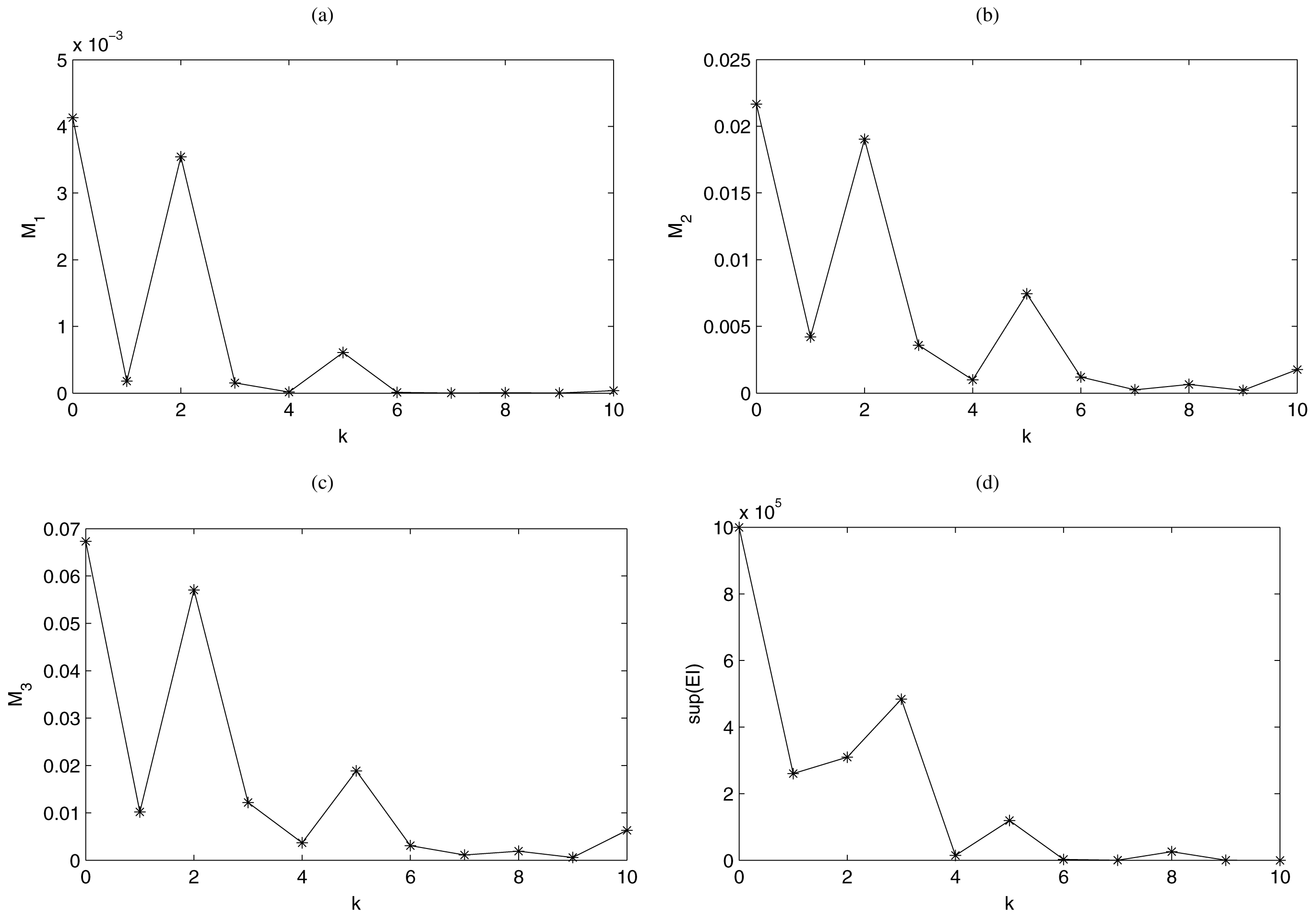

Figure 2. Lack-of-fit plots illustrating the improvement in the information of the contour as a result of sequential addition of trials. (a) The lack of correlation. (b) The average L_2 distance between $C_{n_0,k}$ and C_t. (c) The maximum L_2 distance between the estimated and true contours ($C_{n_0,k}$ and C_t). (d) The maximum expected improvement.

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. Figure 2 shows the diagnostic plots based on these discrepancy measures for this single application of the procedure. Note that all of the discrepancy measures show a decreasing trend as n gets larger. As we add more trials (i.e., as k increases), $C_{n_0,k} \to C_t$. In fact, we can show that each of the discrepancy measures converges to 0 as $k \to \infty$. Proofs are given in Appendix B.

In practice, resources are not infinite. Nonetheless, as more trials are performed, the sequence of values for these discrepancy measures can be plotted to gain some insight into the closeness of the estimated contour to the true contour. Furthermore, we can formulate a reasonable stopping rule based on these measures (i.e., when they are close to 0). When sufficient trials have been chosen to get a good approximation of the contour, additional trials will show an insignificant change in these discrepancy measures. But this feature alone does not always indicate a good approximation of the underlying contour; thus in practice, we would want to see 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 (9). Figure 3 shows the estimated contour (solid line) and the true contour (dotted line). For this example, the estimated and true contours are indistinguishable.

To compare the proposed sequential approach with a single-stage random Latin hypercube design, we conducted a simulation study. 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. We compared this approach with that using a single random Latin hypercube design with n points ($n = n_0 + k$). The procedure was 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 and the new sequential approach with $n = 20$ and 25. 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 of the two approaches, the ratio of the entries in the first column with the corresponding entry in the second column.

Note that for both $n = 20$ and $n = 25$, regardless of the initial sample size n_0, the relative efficiencies are >1. This indicates

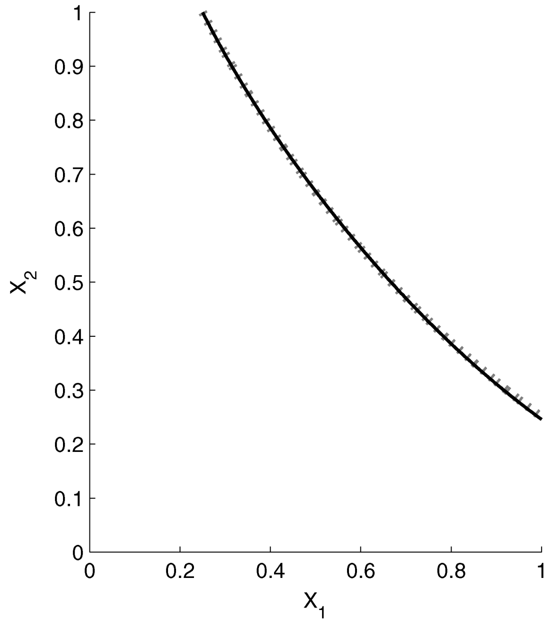

Figure 3. Final estimate of the contour plotted using implicit function from GASP. The solid curve represents the estimated contour, and the dotted curve represents the true contour.

that the proposed methodology performs better than the naive Latin hypercube design. Also note that as we go down the table, for given n, the relative efficiencies decrease to 1 as $(n-n_0)$ decreases to 0. This is expected because as n_0 grows larger, the proposed approach becomes 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 let the underlying function f be the Goldprice function (Andre, Siarry, and Dognon 2000),

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

The Goldprice function is significantly more complicated than the function considered in Example 1. Let the contour of interest be $S(1.5 * 10^5)$. The contour at this height is not a contiguous curve, but our implicit function method 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 various 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, a few of the new trials are sampled in the neighborhood of the contour, whereas others are in sparsely sampled regions. This avoids both the underdetection of the contour [bottom right of Fig. 4(b)] and the inclusion of a false contour segment [bottom left of Fig. 4(c)]. The final estimate of the contour with 28 additional trials is shown in Fig. 4(d).

Remark 1. 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, however. Note that the initial Latin hypercube design does a relatively good job sampling in the interior of the design space. Thus for 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 do so near the boundary; therefore, it is not surprising that some points are sampled near the boundary in our setting.

Table 1. Results of the simulation study for comparing the performance of the sequential approach and the single-stage Latin hypercube (LHC) design, separated by $n = 20, 25$ and different choices of initial run size n_0

Measure	LHC	Sequential	Relative efficiency	LHC	Sequential	Relative efficiency
		$n_0 = 5,\ k = 15\ (n = 20)$			$n_0 = 5,\ k = 20\ (n = 25)$	
M_1	3.54e−04	1.96e−04	1.80	1.05e−04	5.20e−05	2.02
M_2	.0025	.0022	1.13	.0015	.0012	1.25
M_3	.0103	.0056	1.84	.0066	.0031	2.13
		$n_0 = 10,\ k = 10\ (n = 20)$			$n_0 = 10,\ k = 15\ (n = 25)$	
M_1	3.54e−04	2.82e−04	1.25	1.05e−04	6.54e−05	1.61
M_2	.0025	.0020	1.25	.0015	.0014	1.07
M_3	.0103	.0057	1.80	.0066	.0034	1.94
		$n_0 = 15,\ k = 5\ (n = 20)$			$n_0 = 15,\ k = 10\ (n = 25)$	
M_1	3.54e−04	3.03e−04	1.16	1.05e−04	8.68e−05	1.21
M_2	.0025	.0024	1.04	.0015	.0013	1.15
M_3	.0103	.0073	1.41	.0066	.0035	1.88

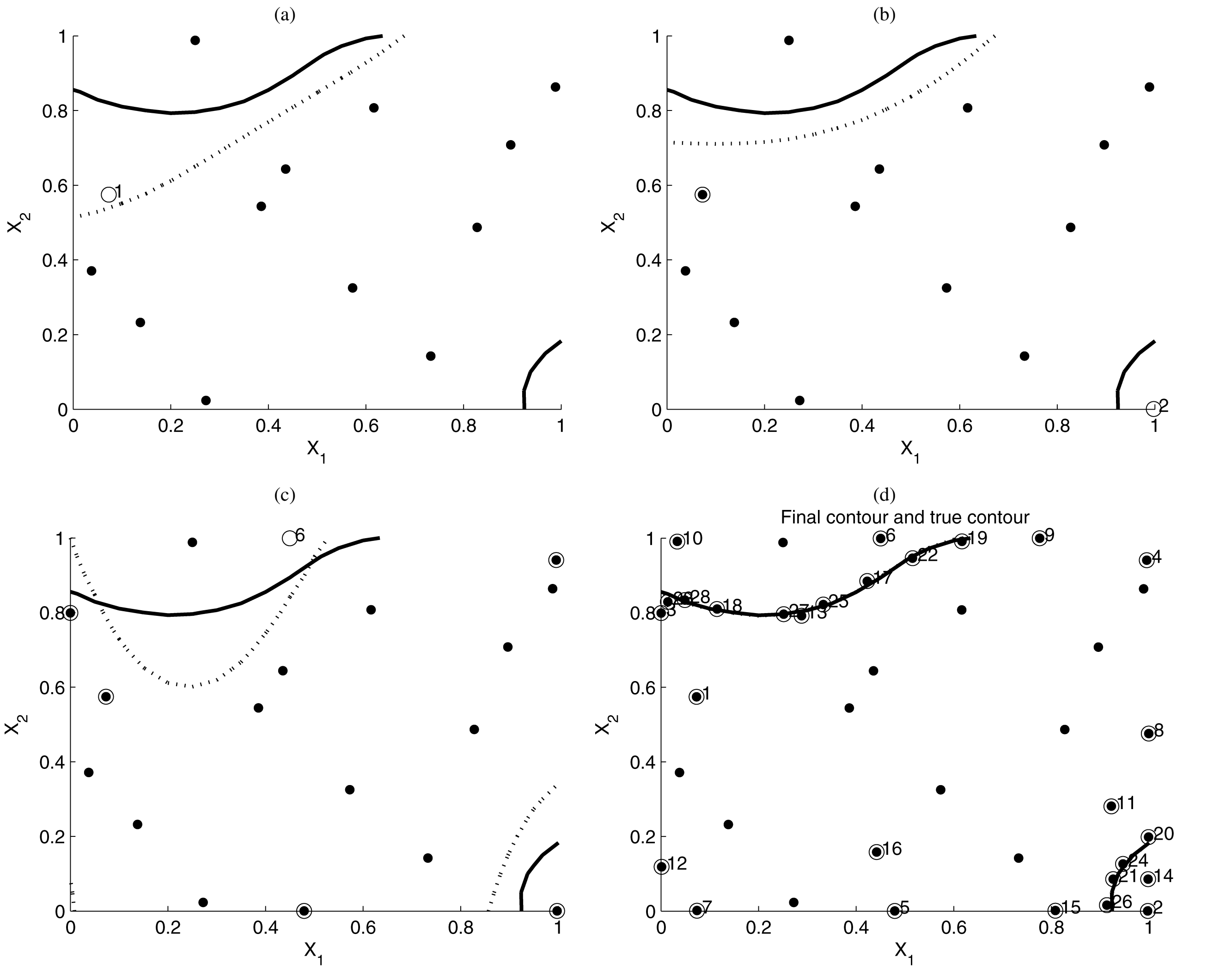

Figure 4. Illustration of the sequential approach for contour estimation ($n = 40$ and $n_0 = 12$). The solid curve represents to the true contour, the dotted curve represents the estimated contour, and the dotted points represent to the initial design points.

Similar to in Example 1, here 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 $n_0 = 5, 10, \ldots, 30$.

A quick glance at Table 2 reveals that all of the relative efficiencies are >1, and in all cases, the proposed approach performs better than the approach using a single random Latin hypercube design. Indeed, for most cases, the relative efficiencies are significantly >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 case where $n = 40$, the proposed approach performs about 10 times better than the Latin hypercube design approach in two metrics (M_2 and M_3). For the case where $n = 35$, the proposed approach also performs 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, 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 relative efficiencies in Table 2 shows an increasing trend as n_0 increases to 15, followed by a decreasing trend as n_0 increases further to 30 runs. This simulation study suggests that $n_0 \approx 15$ will be a good choice.

In general, we recommend choosing n_0 to be around 25–35% of the experimental budget in practice. Starting with very few trials may not capture the overall nature of the surface. On the other hand, if n_0 is too large, then much of the resources will have been used in locations that do not aid contour estimation, and the benefits of the sequential approach will not be realized.

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

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

Table 2. Results of the simulation study for comparing the performance of the sequential approach and single-stage Latin hypercube design (LHC), separated by $n = 35, 40$ and different choices of initial run size n_0

Measure	LHC	Sequential	Relative efficiency	LHC	Sequential	Relative efficiency
		$n_0 = 5,\ k = 30\ (n = 35)$			$n_0 = 5,\ k = 35\ (n = 40)$	
M_1	.4573	.1984	2.30	.1968	4.14e−04	475.36
M_2	.1048	.0560	1.87	.0514	.0056	9.17
M_3	.4468	.2336	1.91	.2303	.0271	8.49
		$n_0 = 10,\ k = 25\ (n = 35)$			$n_0 = 10,\ k = 30\ (n = 40)$	
M_1	.4573	.0024	190.54	.1968	2.38e−04	826.89
M_2	.1048	.0157	6.67	.0514	.0044	11.68
M_3	.4468	.0572	7.81	.2303	.0211	10.91
		$n_0 = 15,\ k = 20\ (n = 35)$			$n_0 = 15,\ k = 25\ (n = 40)$	
M_1	.4573	.0020	228.65	.1968	2.26e−04	870.79
M_2	.1048	.0149	7.03	.0514	.0043	11.95
M_3	.4468	.0574	7.78	.2303	.0206	11.18
		$n_0 = 20,\ k = 15\ (n = 35)$			$n_0 = 20,\ k = 20\ (n = 40)$	
M_1	.4573	.0022	207.86	.1968	2.31e−04	851.94
M_2	.1048	.0162	6.47	.0514	.0044	11.68
M_3	.4468	.0607	7.36	.2303	.0210	10.96
		$n_0 = 25,\ k = 10\ (n = 35)$			$n_0 = 25,\ k = 15\ (n = 40)$	
M_1	.4573	.0209	21.88	.1968	2.77e−04	710.47
M_2	.1048	.0196	5.34	.0514	.0044	11.68
M_3	.4468	.0785	5.69	.2303	.0201	11.45
		$n_0 = 30,\ k = 5\ (n = 35)$			$n_0 = 30,\ k = 10\ (n = 40)$	
M_1	.4573	.1108	4.13	.1968	3.14e−04	625.51
M_2	.1048	.0371	2.82	.0514	.0053	9.70
M_3	.4468	.1540	2.90	.2303	.0246	9.36

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

Table 3 shows the performance of the proposed methodology compared with that of 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). Then the gains from using the proposed algorithm will be less substantial regardless of dimensionality.

For this larger-sized 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 to obtain starting values, which we then use as inputs to the packaged optimizers (global branch-and-bound algorithm; Matlab Optimization Toolbox function *fmincon*). For the six-dimensional Levy function with $n_0 = 30$ and $k = 50$, the average time for the genetic algorithm to obtain a starting value is approximately 56 seconds, and both packaged optimizers take approximately .08 seconds to converge to the optimum. For the two-dimensional Goldprice function with $n_0 = 12$ and $k = 28$, the genetic algorithm takes approximately 6 seconds on

Table 3. Results of the simulation study for comparing the performance of the sequential approach and single-stage Latin hypercube (LHC) design, separated by $n = 60,\ 80$ and different choices of initial run size n_0

Measure	LHC	Sequential	Relative efficiency	LHC	Sequential	Relative efficiency
		$n_0 = 15,\ k = 45\ (n = 60)$			$n_0 = 15,\ k = 65\ (n = 80)$	
M_1	1.0424	.7313	1.42	.9848	.5485	1.80
M_2	.4980	.3958	1.26	.4804	.3352	1.43
M_3	.9722	.8712	1.12	.9602	.8397	1.14
		$n_0 = 30,\ k = 30\ (n = 60)$			$n_0 = 30,\ k = 50\ (n = 80)$	
M_1	1.0424	.7669	1.36	.9848	.5826	1.69
M_2	.4980	.3942	1.26	.4804	.3539	1.36
M_3	.9722	.8819	1.10	.9602	.8544	1.12
		$n_0 = 45,\ k = 15\ (n = 60)$			$n_0 = 45,\ k = 35\ (n = 80)$	
M_1	1.0424	.9405	1.11	.9848	.6909	1.43
M_2	.4980	.4508	1.10	.4804	.3872	1.24
M_3	.9722	.9405	1.05	.9602	.8620	1.11

average, and the average time taken by both the packaged optimizers is <.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. Thus 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 stability. Roughly speaking, the system is said to be stable if the expected delay is finite. We restrict our attention to the input region, χ, 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 χ.

Because the design region is not rectangular, and we have finitely many grid points from which to choose the trials, we use a minimax design (Johnson et al. 1990; John, Johnson, Moore, and Ylvisaker 1995) rather than a Latin hypercube design as our starting design. We use only the trials lying on the lattice points in Figure 5. We consider estimating the contour at the delay of $a = .75$, that is, $S(.75)$. For this illustration, the starting design is minimax with $n_0 = 5$.

Figure 6(a) shows the initial five-point minimax design as well as the true (solid curve) and 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 15 new trials are added.

Because the true function is unknown in practice, we need different *lack-of-fit* measures to study the behavior of convergence for the estimated contours. Replacing C_t with $C_{n_0,k-1}$ in

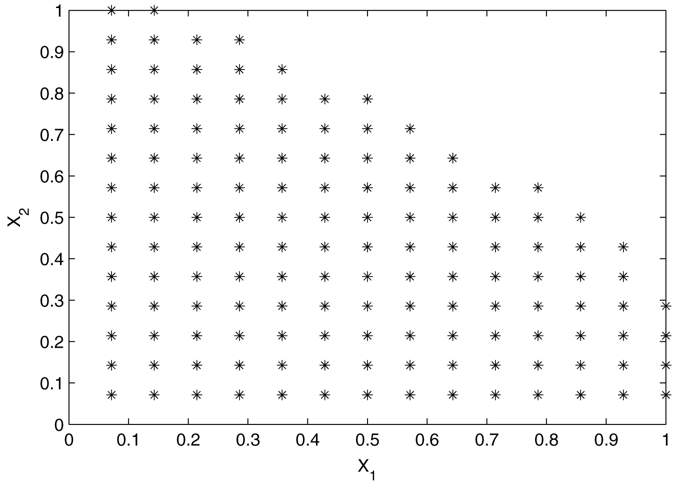

Figure 5. Available design points in the stability region for the queuing example.

(10), (11), and (12), we get a similar set of discrepancy measures, which we can use for diagnostic purposes; that is, we compare the current estimated contour with the previous estimate. The interpretation of these measures differs slightly from the discrepancy measures defined earlier. Now when M_1 is close to 0, 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 0. If the discrepancies stay in the neighborhood of 0 with further additional trials, then we take this as evidence that the contour estimate has stabilized.

Figure 7 shows the plots of the discrepancy measures for this example. As would be expected, the plots show a decreasing trend. Figures 7(b) and (d) clearly show that the last few additional trials do not cause 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 0.

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

6. CONCLUSION

In this article we have developed a 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 of 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 complex response surfaces.

Future work includes simultaneous investigation of multiple contours from complex computer models. In that setting, we are interested in various system states and the associated boundaries between these states.

ACKNOWLEDGMENTS

The authors to thank the associate editor and two anonymous referees for many useful comments and suggestions. The work of George Michailidis was supported in part by National Science Foundation grants DMS-0204247 and DMS-0505535. Derek Bingham's research was supported by grants from the Natural Sciences and Engineering Research Council of Canada and the National Program on Complex Data Structures.

APPENDIX A: PROOF OF THE MAIN RESULTS

Proof of Lemma 1

Because $f_n(\cdot|x) \geq 0$, $\forall n \geq 1$, and $\forall x \in \chi$, $0 \leq f_n(y|x) \leq \sup_{x\in\chi} f_n(y|x)$ for any fixed y in the domain of f_n. Taking the expectation over the distribution of $y|x$, we get $0 \leq E[f_n(y|x)] \leq E[\sup_{x\in\chi} f_n(y|x)]$, $\forall n \geq 1$ and $\forall x \in \chi$. Because $E[f_n(y|x)]$ is $\leq E[\sup_{x\in\chi} f_n(y|x)]$ for all x, $\sup_{x\in\chi} E[f_n(y|x)]$ is also $\leq E[\sup_{x\in\chi} f_n(y|x)]$. (For details of these steps, see Rudin 1987.) Suppose that $\sup_{x\in\chi} f_n(\cdot|x) \to 0$ as $n \to \infty$. Then $E[\sup_{x\in\chi} f_n(y|x)] \to 0$ implies that $\sup_{x\in\chi} E[f_n(y|x)] \to 0$. Consequently, we can see that $\lim_{n\to\infty} \sup_{x\in\chi} E[f_n(y|x)] = 0$.

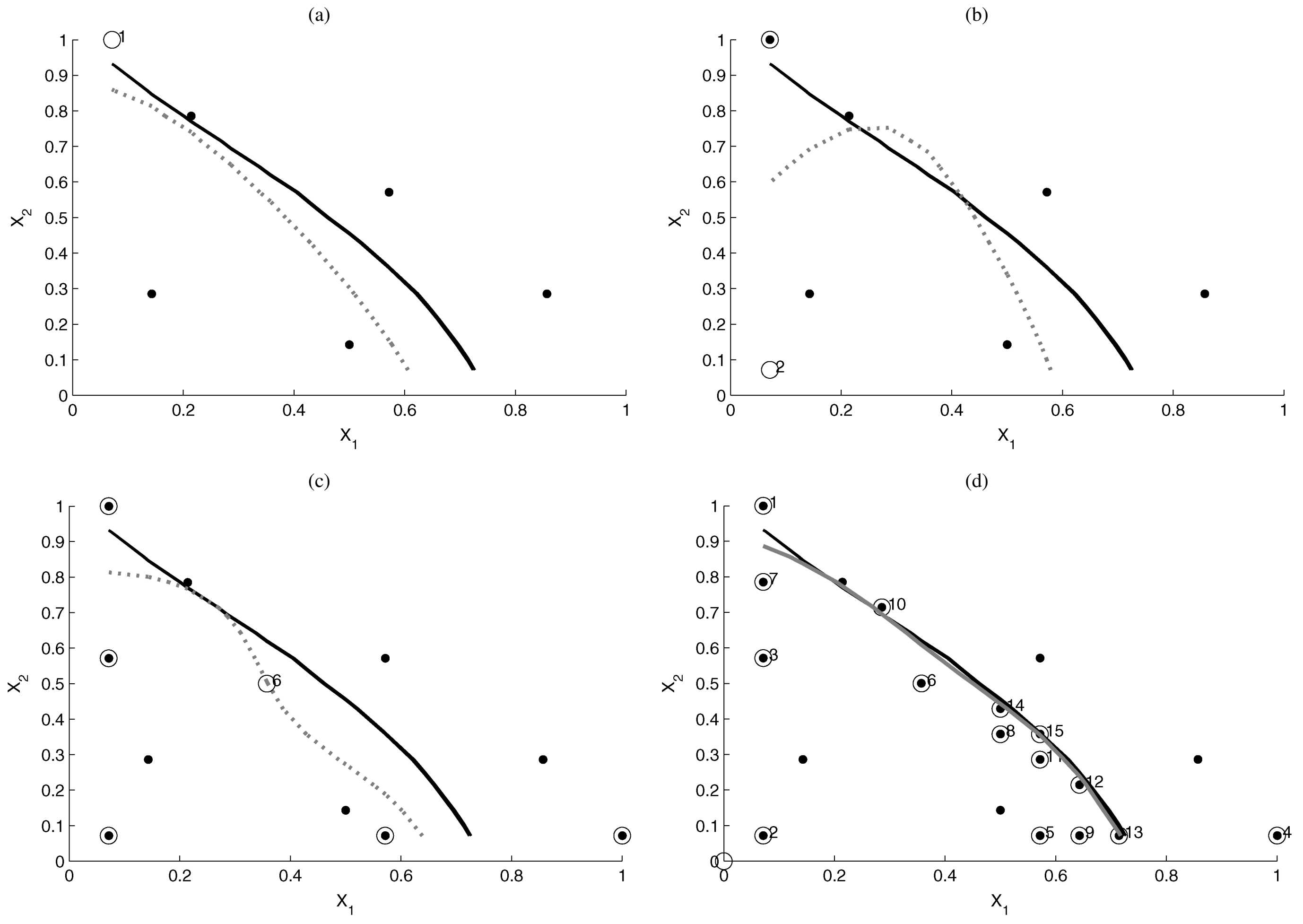

Figure 6. Illustration of the sequential approach for the contour estimation ($n = 20$ and $n_0 = 5$) for the queueing example. The solid curve represents the true contour, the dotted curve represents the estimated contour, and the dotted points represent the initial design points.

Proof of Theorem 1

Recall that $I(x) = \alpha^2 s^2(x) - \min\{(y(x) - a)^2, \alpha^2 s^2(x)\}$ and $0 \le I(x) \le \alpha^2 s^2(x)$. Let $I_n(x)$ and $s_n^2(x)$ denote $I(x)$ and $s^2(x)$, based on a sample of size n. Let $I_n(x)$ be $f_n(y|x)$ in the lemma. Then $\sup_{x\in\chi} f_n(y|x) = \sup_{x\in\chi} I_n(x) \le \sup_{x\in\chi} \alpha^2 s_n^2(x)$. From Lemma 1, it is sufficient to show that $\sup_{x\in\chi} s_n^2(x) \to 0$ to prove the theorem.

Denote $s_n^2(x) = \sigma_z^2(1 - r'R_n^{-1}r + \frac{(1-1_n'R_n^{-1}r)^2}{1_n'R_n^{-1}1_n})$, where R_n is the correlation matrix R based on n trials and r is $\mathrm{corr}(x, (x_1, \ldots, x_n)')$. We show that $\sup_{x\in\chi} s_n^2(x) \to 0$, using three results. These results can be easily verified by mathematical induction on n by noting that (a) as $n \to \infty$, $r \to R_i$ for some $i \in \{1, \ldots, n\}$ and (b) as a consequence of (a) as $n \to \infty$, $R^{-1}r \to R^{-1}R_i = e_i$. (Here e_i is the unit vector with 1 at the ith place and 0 elsewhere.) The three useful results are as follows:

1. $1_n'R_n^{-1}r \to 1$ and thus $(1 - 1_n'R_n^{-1}r)^2 \to 0$.
2. $r'R_n^{-1}r \to 1$, which implies that $1 - r'R_n^{-1}r \to 0$.
3. $1_n'R_n^{-1}1_n \ge \log(n)$, and thus $\frac{1}{1_n'R_n^{-1}1_n} \le \frac{1}{\log(n)}$.

These three results imply that $s_n^2(x) \le C\frac{1}{\log(n)}$ (for an appropriate positive constant C), which further implies that $s_n^2(x) = O(\frac{1}{\log(n)})$. Therefore, $s_n^2(x) \to 0$ as $n \to \infty$. Furthermore, this implies that $\sup_{x\in\chi} s_n^2(x) \to 0$. Thus from Lemma 1, we have that $\lim_{n\to\infty} \sup_{x\in\chi} E(I_n(x)) = 0$.

APPENDIX B: DERIVATION OF THE EXPECTED IMPROVEMENT

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

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

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

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

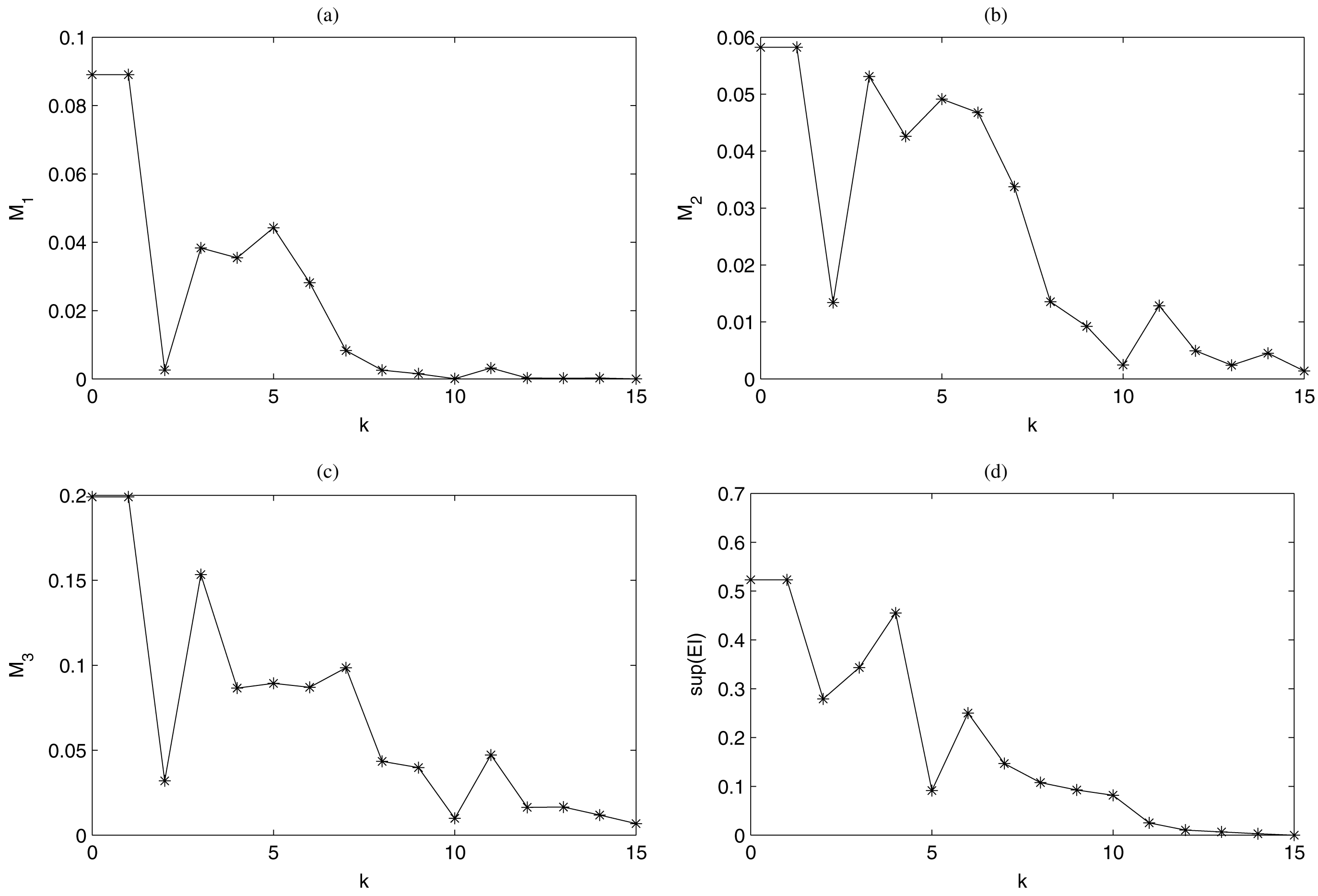

Figure 7. Lack-of-fit plots illustrating the improvement in the information of the contour as a result of the sequential addition of trials for the queueing example. (a) Lack in Correlation; (b) Average L_2 distance between $C_{n_0,k}$ and $C_{n_0,k-1}$; (c) Max L_2 distance between $C_{n_0,k}$ and $C_{n_0,k-1}$; (d) Sup (Expected improvement). These plots refer to discrepancy measures between $C_{n_0,k}$ and $C_{n_0,k-1}$.

therefore,

$$\begin{aligned}E[I(x)] &= -\int_{a-\epsilon(x)}^{a+\epsilon(x)} (y-a)^2 \phi\left(\frac{y-\hat{y}(x)}{s(x)}\right) dy \\ &\quad + \epsilon(x)^2 \left[\Phi\left(\frac{a+\epsilon(x)-\hat{y}(x)}{s(x)}\right) - \Phi\left(\frac{a-\epsilon(x)-\hat{y}(x)}{s(x)}\right)\right].\end{aligned} \tag{B.1}$$

Considering the first term only,

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

Therefore, after substituting $\epsilon(x) = \alpha s(x)$ and (B.1) into (B.2), $E[I(x)]$ becomes

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

APPENDIX C: CONVERGENCE OF DISCREPANCY MEASURES

Result C.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 $M_3 \to 0$ as $k \to \infty$.

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 $x \in C_{n_0,k}$, $d(x, C_{n_0,k-1}) \to 0$. First, discretize $C_{n_0,k}$ into m points; that is, draw m points from the

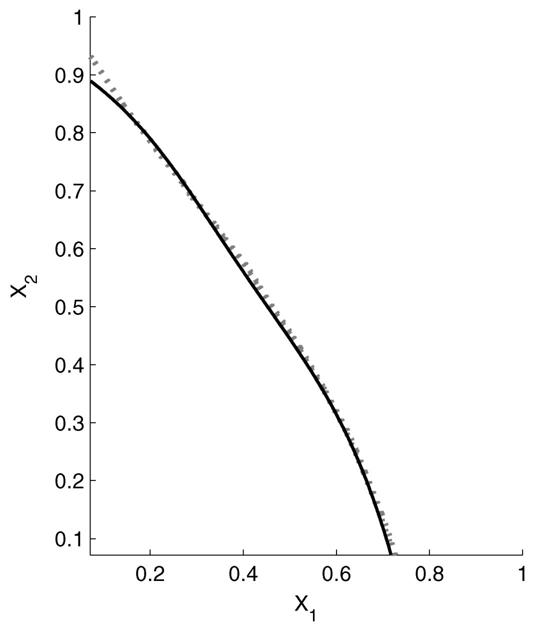

Figure 8. Final estimate of the contour plotted using the implicit function obtained from the GASP model. The solid curve represents the estimated contour; the dotted curve, the true contour.

contour $C_{n_0,k}$ so that it represents the contour. Let

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

Note 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 thus $d(x_i^k, C_{n_0,k+1}) = d(x_i^k, x_i^{k+1})$. Therefore, we need to show that for every $i \in \{1, \ldots, m\}$, $d(x_i^k, x_i^{k+1}) \to 0$ as $k \to \infty$. Suppose that $C_{n_0,k} \to C_t$ as $k \to \infty$ (here C_t is the underlying true contour that we are trying to estimate), and that x_i^t is the homologous point on C_t corresponding to x_i^k on $C_{n_0,k}$. Then, using the 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 $d(x_i^k, x_i^t) \to 0$ as $k \to \infty$ and $d(x_i^t, x_i^{k+1}) \to 0$ as $k \to \infty$. Therefore, for every fixed $i \in \{1, \ldots, m\}$,

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

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

Result C.2: Convergence of M_2. Let $C_{n_0,k}$ and $C_{n_0,k+1}$ be the discretized representation of contours as defined in Result C.1. Then, as $k \to \infty$,

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

Proof. Because, in Result C.1, it was shown that $\sup\{d(x, C_{n_0,k+1}) : x \epsilon C_{n_0,k}\} \to 0$ as $k \to \infty$, it is simple to see that $M_2 \to 0$ as $k \to \infty$.

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

Proof. To prove this result, it is sufficient to show that $\mathrm{corr}(x^{lk}, x^{lt}) \to 1$ as $k \to \infty$. Using the same notation as in the previous result, we have

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

Because $C_{n_0,k} \to C_t$ as $k \to \infty$, $x_i^{lk} \to x_i^{lt}$ as $k \to \infty$ for all i and all l. Thus $\mathrm{corr}(x^{lk}, x^{lt}) \to 1$ as $k \to \infty$, and therefore, $M_1 \to 0$ as $k \to \infty$.

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

[Received October 2005. Revised January 2007.]

REFERENCES

Andre, J., Siarry, P., and Dognon, T. (2000), "An Improvement of the Standard Genetic Algorithm Fighting Premature Convergence," *Advances in Engineering Software*, 32, 49–60.

Balakrishnan, V., Boyd, S., and Balemi, S. (1991), "Branch and Bound Algorithm for Computing the Minimum Stability Degree of Parameter-Dependent Linear Systems," *International Journal of Robust and Nonlinear Control*, 1, 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), "Bayesian Wombling: Curvilinear Gradients Assessment Under Spatial Process Models," *Journal of the American Statistical Association*, 101, 1487–1501.

Barbujani, G., Oden, N. L., and 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," in *SPIE01, Proceedings of the International Society for Optical Engineering*, San Diego, CA.

Dixon, L. C. W., and Szego, G. P. (1978), "The Global Optimization Problem: An Introduction," in *Towards Global Optimization* 2, eds. L. C. W. Dixon and G. P. Szego, Amsterdam: North-Holland, pp. 1–15.

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*, New York: Wiley.

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, MI: 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," *Jornal 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, 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, 239–245.

Murty, K. G. (1995), *Operations Research: Deterministic Optimization Models*, Englewood Cliffs, NI: 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* (3rd ed.), New York: McGraw-Hill.

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*, New York: Springer-Verlag.

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, U.K.: Cambridge University Press.

Tang, B. (1993), "Orthogonal Array–Based Latin Hypercubes," *Journal of the American Statistical Association*, 88, 1392–1397.

Uhlir, K. (2003), "Modelling Methods With Implicitly Defined Objects," Technical Report DCSE/TR-2003-04, J. E. Purkinje University, Dept. of Urology.

Uhlir, K., and Skala, V. (2003), "Implicit Function Modelling System Comparison of C++ and C# Solutions," in *Proceedings of C# and .NET Technologies* 2003 *Int. Workshop*, Plozen, Czech Republic: UNION Agency–Science Press, pp. 87–92.

Veltkamp, R. C. (2001), "Shape Matching: Similarity Measures and Algorithms," in *Proceedings of the Shape Modelling International Conference*, Genova, Italy: IEEE, pp. 188–197.

Warland, J. (1988), *Introduction to Queuing Networks*, Englewood Cliffs, NI: Prentice-Hall.

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, 1294–1310.