Using Approximations to Accelerate Engineering Design Optimization

Virginia Torczon* Michael W. Trosset†

June 9, 1998

Abstract

Optimization problems that arise in engineering design are often characterized by several features that hinder the use of standard nonlinear optimization techniques. Foremost among these features is that the functions used to define the engineering optimization problem usually require the solution of differential equations, a process which is itself computationally intensive. Within a standard nonlinear optimization algorithm, the solution of these differential equations is required for *each* iteration of the algorithm. To mitigate such expense, an attractive alternative is to replace the computationally intensive objective with a less expensive *surrogate*.

In conformance with engineering practice, we draw a crucial distinction between surrogate *models* and surrogate *approximations*. Surrogate models are auxiliary simulations that are less physically faithful, but also less computationally expensive, than the expensive simulation that is regarded as "truth." An instructive example is the use of an equivalent-plate analysis method in lieu of a finite element analysis, e.g. to analyze a wing-box of a high-speed civil transport. Surrogate models exist independently of the expensive simulation and can provide new information about the physical phenomenon of interest without requiring additional runs of the expensive simulation.

Surrogate approximations are algebraic summaries obtained from previous runs of the expensive simulation. Examples include the low-order polynomials favored in response surface methodology (RSM) and the kriging estimates employed in the design and analysis of computer experiments (DACE). Once the approximation has been constructed, it is typically inexpensive to evaluate.

When surrogates are available, be they models or approximations, the optimizer hopes to use them to facilitate the search for a solution to the engineering optimization problem. Our ultimate goal is to design robust optimization algorithms, but we would like to do so in ways that allow us to make effective use of the information that good surrogates can provide. Toward this end, we adopt the perspective that the surrogate can be used to accelerate the optimization technique by exploiting the trends that such surrogates tend to identify. We do not worry about accuracy in the surrogate until it becomes clear either that the optimization technique is in the neighborhood of a minimizer or that the surrogate is not doing a sufficiently good job of identifying trends in the objective.

We consider a methodology that constructs a *sequence* of approximations to the objective. We concentrate on approaches such as DACE, that krige known values of the objective, but our general strategy is also amenable to other classes of approximations. We make use of pattern search techniques to handle the optimization, though other approaches are possible. We choose pattern search techniques because they can be easily amended to exploit surrogates,

*Department of Computer Science, College of William & Mary, Williamsburg, VA 23187-8795 (e-mail: va@cs.wm.edu).

†Department of Mathematics, University of Arizona, Tucson, AZ 85721 (e-mail: trosset@u.arizona.edu).

can handle functions that are nondifferentiable or for which sensitivities are difficult or too expensive to attain, and can be easily adapted to either a parallel or distributed computing environment. Pattern search methods also are less likely to be trapped by non-global minimizers than are traditional nonlinear optimization algorithms. Furthermore, recent analysis extends their applicability to optimization problems with general nonlinear constraints.

Our approach synthesizes recent ideas from both the numerical optimization and the computer experiments literature. Given a limited budget of expensive function evaluations that are to be used to solve an engineering optimization problem, we consider how to manage the trade-off between the expense of approximation and the expense of optimization. We believe that one should invest only a modest proportion of the original budget in constructing the initial approximation and that one should use the optimization procedure to help decide when and where further sampling should occur.

Our experience shows that managing approximations, which should be updated as optimization progresses, engenders a somewhat different set of issues than managing models, each of which remains static as optimization progresses. Earlier related work did not envision the sequential updating of a surrogate approximation, yet this sequential updating has proved to be the defining characteristic of much of our current research. We now understand that the sequential updating of surrogate approximations may create difficulties that are not encountered when either conventional optimization or the construction of the approximation are employed separately. Furthermore, the introduction of general constraints (as opposed to the special case of bound constraints), complicates not only the optimization, but also the construction of approximations.

We will conclude by discussing some of the issues that remain to be addressed. Currently, we advocate the introduction of merit functions that explicitly recognize the desirability of improving the current approximation to the expensive simulation. We also include suggestions for addressing the ill-conditioning that plagues kriging approximations constructed from sequential designs generated by optimization algorithms. In addition, we propose methods for obtaining space-filling designs for nonrectangular regions and propose pattern search algorithms for such regions. Ultimately, we wish to develop techniques for handling problems in which the constraints are implicitly defined by expensive simulations, but our investigation to date has focussed on problems for which the constraints are defined by algebraic expressions.

Key words: Design optimization, computer simulation, pattern search, kriging, nonparametric response surface methodology.

Contents

1 Introduction

We begin by considering the problem of minimizing an objective $f : \Re^p \to \Re$ subject to bound constraints, i.e.

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in [a, b], \end{array} \tag{1}$$

where $a, x, b \in \Re^p$ and we write $x \in [a, b]$ to denote $a_i \leq x_i \leq b_i$ for $i = 1, \ldots, p$. We are concerned with special cases of Problem (1) for which evaluation of the objective involves performing one (or more) complicated, deterministic computer simulation(s). Many such problems arise as engineering design problems and are often distinguished by two characteristics that preclude solution by traditional algorithms for bound-constrained optimization.

First, the output of a complicated computer simulation is usually affected by a great many approximation, rounding, and truncation errors. These errors are not stochastic—repeating the simulation will reproduce them—but their accumulation introduces high-frequency, low-amplitude distortions of the idealized objective that we would have liked to optimize. In consequence, optimization algorithms that compute or approximate (by finite differencing) sensitivities of f often fail to exploit general trends in the objective and become trapped in local minimizers created by high-frequency oscillations. In order to develop effective algorithms for such applications, we restrict attention to methods that do not use sensitivities, i.e., zero-order methods for numerical optimization.

Second, complicated computer simulations are often expensive to perform. Frank (1995) suggested that one must address problems in which a typical function evaluation costs several hours of supercomputer time. We formalize the notion that the objective is expensive to evaluate by imposing an upper bound V on the number of evaluations of f that we are allowed to perform. The severity of this restriction will depend (in part) on the relation between V and p.

When attempting to minimize an objective f that is too expensive for standard numerical optimization algorithms to succeed, it has long been a standard engineering practice, described by Barthelemy and Haftka (1993), to replace f with an inexpensive surrogate $\hat{f}$ and minimize $\hat{f}$ instead. (For example, one might evaluate f at $V - 1$ carefully selected sites, construct $\hat{f}$ from the resulting information, use a standard numerical optimization algorithm to minimize $\hat{f}$, and evaluate f at the candidate minimizer thus obtained.) This practice may also have the salutary effect of smoothing high-frequency oscillations in f. The rapidly growing literature on computer experiments offers new and potentially better ways of implementing this traditional practice. The prescription that seems to be gaining recent currency was proposed by Welch and Sacks (1991); following current convention, we refer to it as DACE (Design and Analysis of Computer Experiments). Frank (1995) offered an optimizer's perspective on this methodology, suggesting that the "minimalist approach" of minimizing a single $\hat{f}$ is not likely to yield satisfactory results, and proposed several sequential modeling strategies as alternatives. Booker (1996) studied several industrial applications of DACE and two alternative approaches.

We regard DACE and traditional iterative methods for numerical optimization as occupying opposing ends of a spectrum. When V is large relative to p, say $p = 2$ and $V = 500$, then the expense of function evaluation is not an issue and we are content to rely on traditional iterative methods. When V is not large relative to p, say $p = 2$ and $V = 5$, then the expense of function evaluation is completely crippling and we are content to rely on DACE. (If $V < p$, then the methodologies that we consider are not appropriate.) In this report we are concerned with intermediate situations and we borrow ideas from both the numerical optimization and the computer experiment literatures.

We describe a sequential modeling strategy in which approximations proposed for computer experiments are used to guide a grid search for a minimizer. Our methods elaborate and extend

an important special case of the general "model management" strategy proposed by Dennis and Torczon (1997) and developed by Serafini (1998) and Booker et al. (1998). These efforts are part of a larger collaboration described by Booker et al. (1995) and Booker et al. (1996). The specific methods described herein refine several features of the "Model-Assisted Grid Search" algorithm discussed by Trosset and Torczon (1997).

2 Optimization by Pattern Search

We require a method of solving Problem (1) that does not require sensitivities. For unconstrained optimization, one popular zero-order method is the simplex method proposed by Nelder and Mead (1965). This method is sometimes adapted for constrained optimization by means of a simple *ad hoc* device, viz. setting $f(x) = \infty$ when $x \notin [a, b]$. Unfortunately, the Nelder-Mead simplex method is suspect even for unconstrained optimization. For example, McKinnon (1996) has constructed a family of strictly convex, differentiable objectives on $\Re^2$ for which there exist starting points from which Nelder-Mead will fail to converge to a stationary point. Instead, we rely on a class of methods for which a convergence theory exists, the *pattern search methods*. Torczon and Trosset (1997) have provided an elementary introduction to pattern search methods; a convergence theory was developed by Torczon (1997) for the case of unconstrained optimization and extended by Lewis and Torczon (1996a, 1998a, 1998b) to the respective cases of optimization with bound, linear, and general nonlinear constraints.

Pattern search methods are iterative algorithms for numerical optimization. Such algorithms produce a sequence of points $\{x_k\}$ from an initial point, x_0, provided by the user. To specify an algorithm, one must specify how it progresses from the current iterate x_c to the subsequent iterate, x_+. One of the distinguishing features of pattern search methods is that they restrict their search for x_+ to a grid (more formally, a lattice) that contains x_c. The grid is modified as optimization progresses, according to rules that ensure convergence to a stationary point.

1. Specify the current grid contained in $[a, b]$. Select x_0 from the current grid. Let $x_c = x_0$.
2. Do until convergence:
   (a) Let $T(+) = \emptyset$.
   (b) Do while $T(+) = \emptyset$:
      i. Search the current grid for a set of $x_t \in [a, b]$ at which f is then evaluated. Let $T(+)$ denote the set of grid points $x_t \in [a, b]$ thus obtained for which $f(x_t) < f(x_c)$.
      ii. Update the grid.
   (c) Choose $x_+ \in T(+)$.
   (d) Let $x_c = x_+$.

Figure 1: Pattern search methods for numerical optimization.

The essential logic of a pattern search is summarized in Figure 1. Note that pattern search methods for problems with bound constraints are *feasible point* methods: we only consider points that are feasible with respect to the condition $x \in [a, b]$. This is often critical for engineering design optimization problems since the simulation(s) that characterize the problem may not be defined

outside the feasible region. It has been our experience that even when we restrict our attention to points that satisfy the explicit algebraic constraints, it may be the case that some simulations still fail to converge (Booker et al., 1996; Booker et al., 1998).

The reader is advised that this description of pattern search methods differs from the presentation in Torczon (1997) and Lewis and Torczon (1996a, 1996b, 1998a, 1998b). For example, the choice of a starting point is usually not restricted and the initial grid is constructed so that it contains the starting point. More significantly, pattern search methods are usually specified by rules that prescribe where the algorithm is to search for the subsequent iterate and the notion of an underlying grid is implicit in these rules. In this report, the grid is explicit and the search for a subsequent iterate is not restricted to a specific pattern of points. What should be appreciated is that the present description preserves all of the elements of pattern search methods required by their convergence theory.

The crucial elements of a pattern search algorithm are contained in the specification of 2(b) in Figure 1. The fundamental idea is to try to find a point on the current grid that strictly decreases the current value of the objective. Any such point can be taken to be the subsequent iterate. If one fails to find such a point, then one replaces the current grid with a finer one and tries again.

Torczon (1997) described the search in 2(b)(i) as an *exploratory moves algorithm*. Here we distinguish two components of an exploratory moves algorithm: an *oracle* that produces a set of trial points on the current grid, and a *core pattern* of trial points on the grid at which the objective must be evaluated before the algorithm is permitted to refine the grid. The convergence theory requires that the core pattern satisfy certain hypotheses; no hypotheses are placed on the oracle.

Because the methods proposed in this report critically depend on the arbitrary nature of the oracle, we emphasize that *any method whatsoever* can be employed to produce points that potentially decrease the current value of the objective. We might perform an exhaustive search of the current grid or we might specify a complicated pattern of points at which to search. We might appeal to our prior knowledge of or our intuition about the objective. It does not matter—the convergence theory for pattern search methods encompasses all such possibilities.

For the sake of clarity, we describe more fully a specific pattern search algorithm. First, we construct the grids on which the searches will be conducted. For $n = 0, 1, 2, \ldots$, we define vector lattices $\Gamma(n)$ restricted to the feasible set $[a, b]$ as follows: $x \in \Gamma(n)$ if and only if for each $i = 1, \ldots, p$ there exists $j_i \in \{0, 1, \ldots, 2^n\}$ such that

$$x_i = a_i + \frac{j_i}{2^n} (b_i - a_i) .$$

Thus, $\Gamma(0)$ comprises the vertices of $[a, b]$ and $\Gamma(n+1)$ is obtained from $\Gamma(n)$ by halving the distance between adjacent grid points (see below). When we update the current grid, say $\Gamma(n)$, in 2(b)(ii), we either retain $\Gamma(n)$ or replace $\Gamma(n)$ with $\Gamma(n + 1)$.

Next we specify a core pattern. Given $x_c \in [a, b]$, we say that $x_t \in [a, b]$ is adjacent to x_c if and only if there exists $k \in \{1, \ldots, p\}$ such that

$$x_{tk} = x_{ck} \pm \frac{1}{2^n} (b_k - a_k)$$

and $x_{ti} = x_{ci}$ for $i \neq k$. We take as the core pattern the set of grid points adjacent to the current iterate. (For example, if the current grid is the integer lattice on $\Re^2$ restricted to $[a = (0, 0)', b = (8, 8)']$, then the core pattern of $(2, 0)'$ comprises $(3, 0)'$, $(2, 1)'$, and $(1, 0)'$, as can be seen in Figure 2.) We refine the grid, i.e. we replace $\Gamma(n)$ with $\Gamma(n + 1)$, if and only if we have evaluated f at each grid point x_t adjacent to x_c and failed to find $f(x_t) < f(x_c)$.

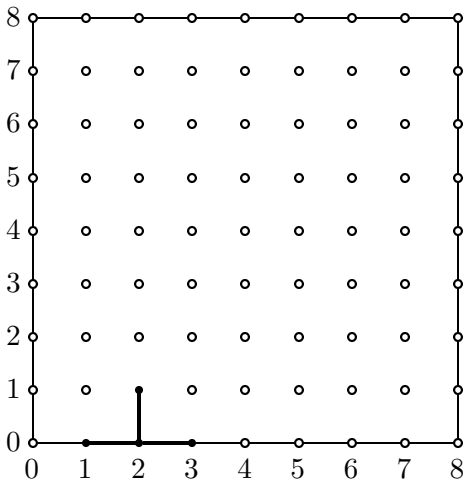

Figure 2: One possible *core pattern* centered at the current iterate $x_c = (2, 0)'$.

If f is continuously differentiable, then the theory developed by Lewis and Torczon (1996a) guarantees that the specified algorithm will produce a sequence $\{x_k\}$ that converges to a Karush-Kuhn-Tucker point of Problem (1). In practice, of course, the algorithm must terminate in a finite number of steps. Termination criteria for pattern search methods—indeed, for direct search methods in general—have not been studied extensively, but that does not concern us here. By definition, the assumption that Problem (1) is expensive means that we cannot afford enough evaluations of the objective to terminate by the traditional criteria of numerical optimization. For the problems that we consider, the relevant termination criterion is whether or not we have exhausted the permitted number (V) of function evaluations.

Zero-order methods for numerical optimization can be quite profligate with respect to the number of function evaluations that they require. Because the number of function evaluations available to us is severely limited, we want to use these evaluations as efficiently as possible. On the bright side, the convergence theory for pattern search methods allows us to replace x_c with *any* x_t for which $f(x_t) < f(x_c)$. Hence, no matter how comprehensive a search for trial points we may have envisioned, we can abort it as soon as we find a single trial point that satisfies. On the dark side, the oracle may require a great many function evaluations to produce even one x_t for which $f(x_t) < f(x_c)$. Furthermore, if the oracle is unsuccessful, then we can not refine the current grid until after f has been evaluated at each grid point adjacent to x_c—a process that may require as many as $2p$ additional function evaluations if x_c is an interior grid point and f has not yet been evaluated at any of the grid points adjacent to x_c.

Pattern search algorithms may intentionally use large numbers of function evaluations. For example, the oracle employed by Dennis's and Torczon's (1991) *parallel direct search* (PDS) intentionally casts a wide net, relying on parallel computation to defray the expense of evaluating the objective at a great many grid points. We are concerned with problems for which huge numbers of evaluations of the objective are not possible. Here, we want an oracle that proposes promising trial points using only a small number of evaluations of the objective. Our strategy for creating such an oracle will be to use previous function values to construct a current global approximation, $\hat{f}_c$, of f, then use $\hat{f}_c$ to predict trial points x_t at which $f(x_t) < f(x_c)$. Thus, we will employ the strategy described in Section 1, not once to replace Problem (1), but repeatedly to guide us to a solution of it. We elaborate further in the next section.

3 Optimization and Sequential Designs

The optimization strategies considered in this report are predicated on a simple idea, viz. that providing the oracle in Section 2 with an inexpensive approximation of the objective will allow it to more efficiently identify promising trial points and thereby reduce the cost of optimization. The approximations are constructed from known function values by kriging, as will be described in Section 4. In this section, we focus on the interplay between the pattern search algorithm and the kriging approximations.

We begin with an instructive analogy. For unconstrained minimization of smooth objective functions, the numerical algorithms of choice are the *quasi-Newton methods*. An elementary exposition of these methods was provided by Dennis and Schnabel (1996), who emphasized the following interpretation: a quasi-Newton algorithm constructs a quadratic approximation $\hat{f}_c$ of the objective f at the current iterate x_c and uses $\hat{f}_c$ to identify a trial point x_t at which it is predicted that $f(x_t) < f(x_c)$. For example, *trust region methods* obtain x_t by (approximately) minimizing $\hat{f}_c$ subject to the constraint that x_t be within a specified neighborhood of x_c. The rationale for the constraint is that the approximation $\hat{f}_c$, which is usually constructed by computing or approximating the second-order Taylor polynomial of f at x_c, can only be trusted to approximate f locally.

Trust region methods make effective use of simple local quadratic approximations of the objective. Because we are concerned with situations in which evaluation of the objective is prohibitively expensive, we are prepared to invest more resources in constructing and optimizing more complicated global approximations of the objective.

Similarly, classical response surface methodology, from Box and Wilson (1951) to Myers and Montgomery (1995), constructs local linear or quadratic regression approximations of a stochastic quadratic objective. Again, the purpose of these approximations is to guide the search for a minimizer or maximizer. Glad and Goldstein (1977) also exploited quadratic regression approximations for optimization, as did Elster and Neumaier (1995) to guide a grid search. Recently, *nonparametric response surface methods* have been proposed in which global approximations of more complicated objectives are constructed. This work, e.g. Haaland (1996), is closely related to ours.

The zero-order methods on which we base our algorithms are the pattern search methods described in Section 2. Instead of specifying a complicated pattern of trial points at which the objective is to be evaluated we attempt to conserve function evaluations by using approximations to identify promising trial points, as described in Torczon and Trosset (1997). The use of *local* linear and quadratic approximations to accelerate numerical optimization by direct search is an idea that is at least as old as Hooke and Jeeves (1961); however, our use of a *global* approximation that is sequentially updated is a more recent idea that poses new challenges.

The following outline details our general strategy for managing approximations to facilitate optimization by pattern search. The generality of this outline is intentional, for a great many specifications seem plausible. The MAGS algorithm described by Trosset and Torczon (1997) represented our first attempt to implement a specific version of this general strategy.

1. Specify an initial grid, Γ_0, that covers the feasible region.. (The methods described by Trosset and Torczon (1997) and by Booker et al. (1998) assumed a rectangular feasible region. As discussed in Sections 2 and 5, extension to nonrectangular feasible regions specified by algebraic constraints is now possible.)

2. Perform an initial computer experiment:

   (a) Select N initial design sites $x_1, \ldots, x_N$. (Trosset and Torczon (1997) employed Latin hypercube sampling for this purpose. Booker et al. (1998) preferred orthogonal arrays. Our

current preference is to modify one of these designs according to one of the approximate maximin criteria discussed further in Section 5.)

(b) Evaluate the true objective f at the initial design sites.

(c) Construct an initial approximation $\hat{f}_0$, e.g. by kriging.

3. Set the current grid, Γ_c, equal to the initial grid. Specify a current iterate, x_c, e.g. $x_c = \text{argmin}(f(x_1), \ldots, f(x_N))$. Set the current approximation, $\hat{f}_c$, equal to the initial approximation. Let Eval_c denote the current set of sites at which the objective has been evaluated. Set Eval_c equal to the set of initial design sites.

4. Do until a minimizer is identified or until the computational budget is exhausted:

   (a) Let $\text{Core}(x_c)$ denote the set of grid points that must be evaluated before the current grid can be defined. This set is specified by the convergence theory for pattern search methods. Do until $\text{Core}(x_c) \subset \text{Eval}_c$, i.e. until theory permits refining the current grid:

      i. Apply an optimization method to a merit function to obtain $x_t \in \Gamma_c \setminus \text{Eval}_c$, a grid point at which the objective has not yet been evaluated. (Trosset and Torczon (1997) applied a finite-difference quasi-Newton method to the current approximation. The need for more general merit functions that incorporate experimental design considerations is discussed in Section 6.)

      ii. Evaluate the objective at x_t. Update Eval_c and $\hat{f}_c$.

      iii. If $f(x_t) < f(x_c)$, then let $x_c = x_t$.

   (b) Refine the current grid.

In the remainder of this section we identify two inherent difficulties in the use of approximations to facilitate optimization. Both of these difficulties proceed from a common cause. When a sequence of design sites is generated by an optimization algorithm, the sites tend to cluster in regions that the optimization algorithm regards as promising. Such a sequence is rarely space-filling and is not likely to be a good experimental design for constructing better approximations. Furthermore, excessive clustering will typically lead to ill-conditioning that affects the calculations of the approximations themselves.

We address each of these difficulties, in turn, in the context of approximations constructed using kriging techniques. In Section 6 we propose the use of merit functions that force the optimization algorithm to take note of experimental design considerations; in Section 7, we comment on the problem of ill-conditioning. Both these discussions are predicated on approximation by kriging, which we discuss in the next section.

4 Approximation by Kriging

Suppose that we have observed $y_i = f(x_i)$ for $i = 1, \ldots, n$. On the basis of this information, we want to construct a global approximation $\hat{f}$ of f. Such inexpensive surrogates for f will be used by the oracle in the pattern search algorithm to identify promising trial points at which to compute additional function values.

We assume that there is no uncertainty in the $y_i = f(x_i)$, i.e. that no stochastic mechanism is involved in evaluating the objective. It is then reasonable to require the approximation to interpolate these values, i.e. to require that $\hat{f}(x_i) = f(x_i)$. Furthermore, we desire families of surrogates that are rich enough to approximate complicated objectives. Toward these ends, we consider certain

families of approximations that have been studied in the spatial statistics and computer experiment literatures. We remark, however, that other approximating families are available and we strive to develop methods that are not specific to our particular choice of approximating family.

The families of approximations that we consider are usually motivated by supposing that f is a realization of some (nice) stochastic process. For certain geostatistical applications, this supposition may be quite plausible. In the context of using computer simulations to facilitate the engineering of better product designs, its plausibility is less clear. The high-frequency, low-amplitude oscillations that we have described do resemble the realization of a stochastic process, but the general trends that are our primary concern do not. In any case, we regard the supposition of an underlying stochastic process as nothing more than a convenient fiction. The value of this fiction lies in its power to suggest plausible ways of constructing useful approximations and we will try to avoid invoking it excessively. When we do invoke it, it should be appreciated that optimality criteria such as BLUP and MLE are defined with respect to the fictional stochastic process and should not be invested with more importance than the practical utility of the approximations in which they result.

Our requirement that $\hat{f}(x_i) = f(x_i)$ will immediately suggest spline interpolation to the approximation theorist and kriging to the geostatistician. In fact, as explicated by Watson (1984) and others, these two well-known methodologies are formally equivalent. Their motivations, however, are somewhat different: whereas the goal of the former is to interpolate the $f(x_i)$ as smoothly as possible, the goal of the latter is to approximate f as accurately as possible. It is evident that the kriging perspective is more germane to our present concerns. The remainder of this section briefly summarizes some relevant facts about kriging in the context of computer experiments. See Sacks, Welch, Mitchell and Wynn (1989) and Koehler and Owen (1996) for comprehensive surveys of computer experiment methodology.

We begin by assuming that f is a realization of a stochastic process F that is indexed by the continuous parameter set $\Re^p$. We assume that this process has known mean $\mu(x) = 0$ and known covariance function $c(\cdot, \cdot)$, and that each symmetric $p \times p$ matrix $c(s, t)$ is strictly positive definite. Let

$$y = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

and for each $x \in \Re^p$ define $b(x) \in \Re^n$ to minimize $E[y'b - F(x)]^2$. Then $\hat{f}(x) = y'b(x)$ is the *best linear unbiased predictor* (BLUP) of $f(x)$ and it is well-known that

$$\hat{f}(x) = y'C^{-1}c(x), \tag{2}$$

where C is the symmetric positive definite $n \times n$ matrix $[c(x_i, x_j)]$ and

$$c(x) = \begin{bmatrix} c(x_1, x) \\ \vdots \\ c(x_n, x) \end{bmatrix}.$$

This is a simple example of kriging. Notice that kriging necessarily interpolates: since $C^{-1}C = I$ and $c(x_j)$ is column j of C,

$$\hat{f}(x_j) = y'C^{-1}c(x_j) = y'e_j = y_j = f(x_j).$$

Thus far we have assumed that the stochastic process is known. We now suppose that F is a Gaussian process with mean $\mu(x) = a(x)'\beta$ and covariance function $c(s, t) = \sigma^2 r_\theta(s, t)$. We assume

that $a : \Re^p \to \Re^q$ is a known function, that $\beta \in \Re^q$ is an unknown vector, that $\sigma^2 > 0$ is an unknown scalar, and that $r_\theta(\cdot,\cdot)$ is an unknown element of a known family of correlation functions.

Next let A denote the $n \times q$ matrix $[a_j(x_i)]$, let $R(\theta)$ denote the symmetric $n \times n$ matrix $[r_\theta(x_i, x_j)]$, and let

$$r(x;\theta) = \begin{bmatrix} r_\theta(x_1, x) \\ \vdots \\ r_\theta(x_n, x) \end{bmatrix}.$$

Then, for β and θ fixed, the BLUP of $f(x)$ is (cf. equation (2))

$$\hat{f}(x) = a(x)'\beta + (y - A\beta)'R(\theta)^{-1}r(x;\theta). \tag{3}$$

Thus, by varying β and θ, we define a family of interpolating functions from which we can select a specific $\hat{f}$ to approximate f.

Given a family of interpolating functions defined by (3), we require a sensible way of specifying $(\beta, \sigma^2, \theta)$ and thereby $\hat{f}$. For θ fixed, the *maximum likelihood estimates* (MLEs) of β and σ^2 have explicit formulas:

$$\hat{\beta}(\theta) = \left[A'R(\theta)^{-1}A\right]^{-1} A'R(\theta)^{-1}y$$

and

$$\hat{\sigma}^2(\theta) = \frac{1}{n}\left[y - A\hat{\beta}(\theta)\right]' R(\theta)^{-1}\left[y - A\hat{\beta}(\theta)\right].$$

To compute $\hat{\theta}$, the MLE of θ, it turns out that one must minimize

$$n \log \hat{\sigma}^2(\theta) + \log\det R(\theta) \tag{4}$$

as a function of θ.

It is now evident that, in specifying a family of correlation functions, there is a potential tradeoff between the richness of the family defined by (3) and the ease of maximizing (4). The richer the approximating family, the more difficult it may be to select a plausible member of it. Most papers on computer experiments have been concerned with deriving a single approximation $\hat{f}$ that will be used as a permanent surrogate for f. Understandably, the authors have used rich families with rather complicated correlation functions for which θ is a vector of dimension p or greater. This makes maximizing (4) difficult, but $\hat{\theta}$ need only be computed once. In contrast, we are concerned with deriving a sequence of approximations that will be used for the sole purpose of guiding our optimization of f. Hence, we are content to sacrifice some flexibility in (3) in order to simplify minimizing (4). In numerical experiments, we have used the 1-parameter isotropic correlation function defined by

$$r_\theta(s,t) = \exp\left(-\theta\|s - t\|^2\right), \tag{5}$$

where $\|\cdot\|$ denotes the Euclidean norm on $\Re^p$.

5 Space-Filling Initial Designs

In the language of computer experiments, points at which the objective is evaluated are called *design sites*. Evidently, one cannot construct an initial approximation until one knows the value of the objective at a set of design sites. The problem of choosing the initial design sites is a problem of experimental design.

Roughly speaking, there are two approaches to the design of computer experiments. The parametric approach is often invested with a Bayesian interpretation. As in Section 4, one assumes that f is the realization of a stochastic process and specifies a parametric family of possible processes. It is possible to extend familiar design criteria like D-optimality to this setting, then construct designs that are optimal with respect to the specified family. We are disinclined to pursue this approach because it ties the problem of choosing the design sites to the problem of specifying a family of approximations. Furthermore, the practical difficulties of actually computing such optimal designs can be formidable.

The nonparametric approach to the design of computer experiments chooses design sites in a manner that is perceived to be "space-filling". When the experimental region $E \subset \Re^p$ is a bounded rectangle, Latin hypercube sampling (McKay, Beckman, and Conover, 1979) and orthogonal array sampling (Owen, 1992, 1994; Tang, 1993) are practical, inexpensive ways of generating space-filling designs. Design criteria that are explicitly space-filling include the minimax and maximin principles proposed by Johnson, Moore, and Ylvisaker (1990); however, these designs can be difficult to compute.

Unfortunately, the utility of Latin hypercube and orthogonal array sampling diminishes when, as is often the case, the experimental region is not rectangular, i.e. when the optimization problem has a nonrectangular feasible region. If E can be inscribed in a rectangle of the same dimension, then a plausible space-filling design can often be obtained by the simple *ad hoc* device of generating a space-filling design for the circumscribing rectangle and accepting the resulting design sites that fall in E. To improve such designs—and to generate plausible space-filling designs in regions that do not readily lend themselves to this device—Trosset (1998) suggested a method of approximating maximin designs. Approximate maximin designs can be computed by conventional nonlinear programming algorithms, subject to the usual caveats about the possibility that such algorithms will find nonglobal solutions. The remainder of this section describes how this can be accomplished.

Let N denote the specified number of design sites and suppose that $x_1, \ldots, x_N \in E$, where E is a compact subset of $\Re^p$. For convenience, we place x_i' in row i of the $N \times p$ design matrix $X = (x_{ik})$. We then abuse notation and write $X \in E$.

Let

$$d_{ij}(X) = \|x_i - x_j\|_2 = \left[\sum_{k=1}^{p} (x_{ik} - x_{jk})^2\right]^{1/2}.$$

Then $X^* \in E$ is a maximin Euclidean distance design in E if

$$\min_{i>j} d_{ij}(X^*) \geq \min_{i>j} d_{ij}(X)$$

for all $X \in E$. Maximin designs are intuitively appealing because they explicitly endeavor to spread the design sites as much as possible.

Let ϕ denote any strictly decreasing function on $[0, \infty)$, e.g. $\phi(t) = \exp(-t^2)$, and let $\phi_{ij}(X) = \phi(d_{ij}(X))$. Let $v(X)$ denote the vector of length $m = n(n-1)/2$ whose kth component is $\phi_{ij}(X)$, where

$$k = (j-1)(N - j/2) + i - j$$

for $j = 1, \ldots, N-1$ and $i = j+1, \ldots, N$. Then X^* is a maximin design if and only if it is a (global) solution of the constrained optimization problem

$$\begin{array}{ll} \text{minimize} & \|v(X)\|_\infty \\ \text{subject to} & X \in E. \end{array} \tag{6}$$

The objective in Problem (6) involves the sup norm, which is not smooth. The standard way of circumventing this difficulty would lead us to reformulate Problem (6) as

$$\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & X \in E, \\ & v_1(X) \leq z, \ldots, v_m(X) \leq z, \end{array}$$

but this approach introduces a large number of nonlinear inequality constraints. Instead, we approximate $\|v(X)\|_\infty$ by the smooth objective $\|v(X)\|_\sigma$, resulting in the more tractable optimization problem

$$\begin{array}{ll} \text{minimize} & \|v(X)\|_\sigma \\ \text{subject to} & X \in E. \end{array} \tag{7}$$

Let X^σ denote a global solution of Problem (7). The following result, a proof of which appears in Trosset (1998), justifies calling X^σ an approximate maximin design, although it should be noted that the plausibility of X^σ as a space-filling design does not depend on this justification.

Theorem 1 *Let $\sigma_k \to \infty$ as $k \to \infty$ and let X^∞ be any accumulation point of $\{X^{\sigma_k}\}$. Then X^∞ is a maximin design.*

Once σ has been fixed, Problem (7) is equivalent to the following:

$$\begin{array}{ll} \text{minimize} & \sum_{j>i}[\phi_{ij}(X)]^\sigma \\ \text{subject to} & X \in E. \end{array} \tag{8}$$

How difficult it will be to solve Problem (8) will depend on how easily one can manage (a) the objective, which in turn will depend on the choice of ϕ; and (b) the constraints.

Ideally, we would like to choose ϕ in a way that facilitates global optimization by inducing as few local minimizers as possible. At present, it is difficult to see how to do this. For the present, we opt for simplicity and set $\phi(t) = \exp(-t^2)$. With this choice of ϕ, Problem (8) simplifies to

$$\begin{array}{ll} \text{minimize} & \sum_{j>i} \exp[-\sigma \sum_{k=1}^p (x_{ik} - x_{jk})^2] \\ \text{subject to} & X \in E. \end{array} \tag{9}$$

The resulting objective, for which first and second derivatives are easily computed, is reminiscent of the SSTRESS criterion for metric multidimensional scaling. The interested reader is referred to Kearsley, Tapia, and Trosset (1994) for a survey of some algorithms available for the unconstrained minimization of the SStress criterion.

We envision solving Problem (9) by employing standard nonlinear programming software. Thus, our methods are suited to situations in which the experimental region is specified by a finite number of algebraic (preferably linear) equality and inequality constraints. The reader seeking to identify algorithms and software suited to specific applications is referred to Moré and Wright (1993).

6 Merit Functions

Especially during the early stages of optimization, greater gains are likely to come from improving the current approximation to the true objective than from accurately identifying a minimizer of the current approximation. The MAGS algorithm proposed by Trosset and Torczon (1997) identified

a trial site by minimizing the current approximation without concern for the quality of the new approximation that will result after the expensive simulation has been run at the trial site.

The desirability of balancing the concerns of numerical optimization and the concerns of experimental design was recognized by Frank (1995), who proposed a dichotomous search strategy. Given an optimization criterion, e.g. the current approximation employed as a surrogate objective, and a design criterion, one obtains some fraction of new design sites using one criterion and the balance using the other. An implementation of this Balanced Local-Global Search (BLGS) was described by Booker et al. (1995) and employed by Booker (1996), Booker et al. (1996), and Booker et al. (1998).

In contrast to the dichotomous BLGS strategy, we prefer to identify new design sites by optimizing a merit function. The merit function should be specified so as to balance the potentially competing goals of finding trial sites at which the current approximation is small and choosing good design sites. This way of selecting trial points in no way affects the convergence of the underlying pattern search—it is a purely empirical matter whether or not it improves the performance of the algorithm.

To illustrate, let $\hat{f}_c(x)$ denote the current approximation at x and let $d_c(x)$ denote the distance from x to the nearest design site at which the expensive simulation has been evaluated. The latter is the maximin design criterion described in Section 5. Then a natural family of merit functions comprises those of the form

$$\Phi_c(x) = \hat{f}_c(x) - \rho_c d_c(x),$$

where $\rho_c \geq 0$.

Notice that we allow the merit function to depend on the iteration. Initially, when insufficient information has been gathered to construct a good approximation, greater weight should be placed on the design criterion. As more information is gathered and the approximation improves, progressively more weight can be placed on the optimization of the approximation. We manage the relative weighting of these criteria by adaptively reweighting them according to how well the current approximation predicts decrease in the true objective.

7 Addressing Ill-Conditioning

An approximation constructed by kriging is usually represented by an algebraic formula, e.g. (3), that requires inverting an estimated covariance matrix. In theory, this poses no difficulties because the covariance matrix is assumed to be strictly positive definite; in practice, the estimated covariance matrix may be ill-conditioned.

The covariance function is constructed so that values of the objective at points that are close together are more highly correlated than values of the objective values at points that are far apart. When the points are selected according to the principles of experimental design, the covariance matrix is usually reasonably well-conditioned. Unfortunately, optimization algorithms tend to sample the objective at points that cluster in promising basins, inevitably resulting in covariance matrices that are ill-conditioned or even singular. This difficulty has complicated our attempts to exploit kriging approximations for the purpose of facilitating optimization.

Two simple techniques address the problem of ill-conditioned estimated covariance matrices. To date, we have computed the pseudoinverse by performing a singular value decomposition and setting all singular values smaller than a specified tolerance equal to zero. Alternatively, by adding a constant to the diagonal of the covariance matrix (a "nugget effect"), we can force the matrix to be as well-conditioned as we desire. Both of these techniques sacrifice the property that the kriging

approximation interpolates the function values used to construct it. However, the error that is introduced is usually small and is rarely of consequence.

8 Conclusions

Trosset and Torczon (1997), Serafini (1998), and Booker et al. (1998) have all reported encouraging numerical experiments that commend the sequential use of approximations when optimizing expensive objectives. We have described the methodology that is common to these endeavors. More significantly, we have identified several critical issues that inevitably arise when this methodology is implemented. This report sets forth our current thinking about how to address these issues. Our central theme is that, when using approximations to facilitate optimization, the concerns of experimental design and the concerns of optimization are inextricably linked. Future progress will depend on balancing these concerns efficiently.

Acknowledgments

This research was supported in part by the Air Force Office of Scientific Research (AFOSR) under Grant No. F49620-95-1-0210; and by the National Aeronautics and Space Administration (NASA) under Contract No. NAS1-19480 when the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA. It evolved from an ongoing collaboration with John Dennis and David Serafini (Rice University); Andrew Booker, Paul Frank, and Greg Shubin (Boeing Company); and Andrew Conn (IBM Corporation). Although we have been profoundly influenced by the ideas of our collaborators, some of the opinions expressed in this report may not be shared by all members of the collaboration.

References

Barthelemy, J.-F. M. and Haftka, R. T. (1993). Approximation concepts for optimum structural design—a review. *Structural Optimization*, 5:129–144.

Booker, A. J. (1996). Case studies in design and analysis of computer experiments. In *Proceedings of the Section on Physical and Engineering Sciences*. American Statistical Association.

Booker, A. J., Conn, A. R., Dennis, J. E., Frank, P. D., , Trosset, M., and Torczon, V. (1995). Global modeling for optimization: Boeing/IBM/Rice collaborative project. 1995 final report. Technical Report ISSTECH-95-032, Boeing Information & Support Services, Research & Technology, Technology, P.O. Box 24346, M/S 7L-68, Seattle, WA 98124–0346.

Booker, A. J., Conn, A. R., Dennis, J. E., Frank, P. D., Serafini, D., Torczon, V., and Trosset, M. (1996). Multi-level design optimization: A Boeing/IBM/Rice collaborative project. 1996 final report. Technical Report ISSTECH-96-031, Boeing Information & Support Services, Research & Technology, Technology, P.O. Box 3707, M/S 7L-68, Seattle, WA 98124–2207.

Booker, A. J., Dennis, Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. (1998). A rigorous framework for optimization of expensive functions by surrogates. Accepted, subject to suitable revision, to *Structural Optimization*.

Box, G. E. P. and Wilson, K. B. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society, Series B*, XIII(1):1–45.

Dennis, J. E. and Torczon, V. (1997). Managing approximation models in optimization. In Alexandrov, N. M. and Hussaini, M. Y., editors, *Multidisciplinary Design Optimization: State-of-the-Art*, pages 330–347, Philadelphia. SIAM.

Dennis, Jr., J. E. and Schnabel, R. B. (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia.

Dennis, Jr., J. E. and Torczon, V. (1991). Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474.

Elster, C. and Neumaier, A. (1995). A grid algorithm for bound constrained optimization of noisy functions. *IMA Journal of Numerical Analysis*, 15(4):585–608.

Frank, P. D. (1995). Global modeling for optimization. *SIAG/OPT Views-and-News*, 7.

Glad, T. and Goldstein, A. (1977). Optimization of functions whose values are subject to small errors. *BIT*, 17(2):160–169.

Haaland, P. D. (1996). Nonparametric response surface methods. In *Abstracts: Summaries of Papers Presented at the 1996 Joint Statistical Meetings in Chicago, Illinois*, page 324.

Hooke, R. and Jeeves, T. A. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery (ACM)*, 8(2):212–229.

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Inference and Planning*, 26:131–148.

Kearsley, A. J., Tapia, R. A., and Trosset, M. W. (1994). The solution of the metric STRESS and STRESS problems in multidimensional scaling using Newton's method. Technical Report 94-44, Department of Computational & Applied Mathematics—MS 134, Rice University, Houston, TX 77005-1892. To appear in *Computational Statistics*.

Koehler, J. R. and Owen, A. B. (1996). Computer experiments. In Ghosh, S. and Rao, C. R., editors, *Handbook of Statistics, Volume 13*, pages 261–308. Elsevier Science, New York.

Lewis, R. M. and Torczon, V. (1996a). Pattern search algorithms for bound constrained minimization. Technical Report 96–20, ICASE, Mail Stop 403, NASA Langley Research Center, Hampton, VA 23681-0001, USA. To appear in *SIAM Journal on Optimization*.

Lewis, R. M. and Torczon, V. (1996b). Rank ordering and positive bases in pattern search algorithms. Technical Report 96–71, ICASE, Mail Stop 403, NASA Langley Research Center, Hampton, VA 23681-0001. In revision for *Mathematical Programming*.

Lewis, R. M. and Torczon, V. (1998a). A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. To appear as an ICASE Technical Report. Submitted to *SIAM Journal on Optimization.*

Lewis, R. M. and Torczon, V. (1998b). Pattern search methods for linearly constrained minimization. Technical Report 98–3, ICASE, Mail Stop 403, NASA Langley Research Center, Hampton, VA 23681-0001. Submitted to *SIAM Journal on Optimization*.

McKay, M., Beckman, R., and Conover, W. (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.

McKinnon, K. (1996). Convergence of the Nelder–Mead simplex method to a non-stationary point. Technical Report 96–006, Department of Mathematics & Statistics, The University of Edinburgh, James Clerk Maxwell Building, King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK. To appear in *SIAM Journal on Optimization*.

Moré, J. J. and Wright, S. J. (1993). *Optimization Software Guide*, volume 14 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia.

Myers, R. H. and Montgomery, D. C. (1995). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley & Sons, New York.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Owen, A. B. (1992). Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*, 2:439–452.

Owen, A. B. (1994). Lattice sampling revisited: Monte Carlo variance of means over randomized orthogonal arrays. *Annals of Statistics*, 22:930–945.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423.

Serafini, D. (1998). *A Model Management Framework for Nonlinear Optimization of Computationally Expensive Functions*. PhD thesis, Department of Computational and Applied Mathematics, Rice University, Houston, Texas.

Tang, B. (1993). Orthogonal array-based Latin hypercubes. *Journal of the American Statistical Association*, 88:1392–1397.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25.

Torczon, V. and Trosset, M. W. (1997). From evolutionary operation to parallel direct search: Pattern search algorithms for numerical optimization. *Computing Science and Statistics*, 29. To appear.

Trosset, M. W. (1998). Approximate maximin designs. In progress.

Trosset, M. W. and Torczon, V. (1997). Numerical optimization using computer experiments. Technical Report 97–02, Department of Computational and Applied Mathematics, MS 134, Rice University, Houston, TX 77005–1892. Submitted to *Technometrics*.

Watson, G. S. (1984). Smoothing and interpolation by kriging and with splines. *Mathematical Geology*, 16:601–615.

Welch, W. J. and Sacks, J. (1991). A system for quality improvement via computer experiments. *Communications in Statistics—Theory and Methods*, 20:477–495.