NASA/CR-1998-208460
ICASE Report No. 98-33

Using Approximations to Accelerate Engineering Design Optimization

Virginia Torczon and Michael W. Trosset
The College of William & Mary, Williamsburg, Virginia

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA

Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-97046

August 1998

USING APPROXIMATIONS TO ACCELERATE ENGINEERING DESIGN OPTIMIZATION*

VIRGINIA TORCZON† AND MICHAEL W. TROSSET‡

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 often are computationally intensive. Within a standard nonlinear optimization algorithm, the computational expense of evaluating the functions that define the problem would necessarily be incurred for *each* iteration of the optimization algorithm.

Faced with such prohibitive computational costs, an attractive alternative is to make use of surrogates within an optimization context since surrogates can be chosen or constructed so that they are typically much less expensive to compute. For the purposes of this paper, we will focus on the use of algebraic approximations as surrogates for the objective.

In this paper we introduce the use of so-called *merit functions* that explicitly recognize the desirability of improving the current approximation to the objective during the course of the optimization. We define and experiment with the use of merit functions chosen to simultaneously improve both the solution to the optimization problem (the objective) and the quality of the approximation. Our goal is to further improve the effectiveness of our general approach without sacrificing any of its rigor.

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

Subject classification. Applied & Numerical Mathematics

1. Introduction. The presentation that follows is intended to illustrate that global approximations can be helpful in facilitating optimization. We present several simple examples, selected to illustrate two fundamental perspectives that have guided our recent research:

- When one uses algebraic approximations within an iterative optimization framework, the initial stages of the optimization should concentrate on the *predictive* ability of the approximation. The *accuracy* of the approximation should be a concern only when in the vicinity of a minimizer or when it becomes clear that the approximation is not doing a good job of identifying trends in the objective.
- It is desirable to compromise between the single-minded pursuit of a minimizer and the construction of an approximation that gives a reasonable "picture" of the behavior of the objective—particularly when a problem is known to have many local minimizers.

Neither of these observations is unique to us; however, they motivate us to introduce *merit functions* that explicitly recognize the desirability of improving the current approximation to the expensive simulation

*This research was supported in part by the National Science Foundation under Grant CCR–9734044 and by the National Aeronautics and Space Administration under NASA Contract No. NAS1–97046 while the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, Virginia 23681–2199.

†Assistant Professor, Department of Computer Science, College of William & Mary, P.O. Box 8795, Williamsburg, Virginia 23187–8795, va@cs.wm.edu

‡Associate Professor, Department of Mathematics, College of William & Mary, P.O. Box 8795, Williamsburg, Virginia 23187–8795, trosset@math.wm.edu

in certain regions. In other words, we introduce a criterion that balances the expenditure of expensive evaluations of the objective function between the search for a single minimizer and the desire to learn more about the the behavior of the objective.

1.1. Problem. For the sake of the points we wish to make, we concentrate on the following general expression for the design optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in [\ell, u], \end{array}$$

where $x \in [\ell, u]$ denotes that every component of the p-dimensional vector x is greater than or equal to the corresponding component in ℓ and less than or equal to the corresponding component in u (i.e., there are bounds on all the design or decision variables x).

We cannot guarantee a global minimizer for the objective—the computational cost associated with such a guarantee would be too prohibitive. What we intend, instead, is an approach that is less prone to identifying the "nearest" local minimizer with the hope that this will often lead us to a global minimizer.

1.2. Operating Assumptions. We make two fundamental assumptions about the design problem:

- Evaluation of the objective depends on the output of complicated computer simulations involving large numbers of state or system variables; it is *not* a simple algebraic expression exclusively in terms of the design variables x.
- The objective is expensive to compute because the simulations upon which its value is based are expensive to compute. (Though the strategy we outline is amenable to settings where the evaluation of the objective may be expensive for other reasons).

1.3. Consequences. The consequences of these assumptions are:

- Simulations of complex physical processes can lead to
  - nonstochastic errors in the evaluation of the objective due to the approximation, rounding, and truncation introduced by the simulations; and
  - high-frequency, low-amplitude distortions of the "true" value of the objective,

  features which can cause difficulties for higher-order optimization methods.
- We must be frugal in our evaluation of the objective.

1.4. Goals. Within this setting, we set forth the following goals. Given a modest number of evaluations of the objective (a "budget") produce:

1. A better "picture" of the behavior of the objective in the region(s) of interest.
2. Do so in a way that is guaranteed—in theory—to converge to at least a local constrained stationary point of the objective.

With a limited number of objective evaluations, it may not be possible to ensure convergence to a constrained stationary point of the objective, but satisfying the conditions required by the theory provides some assurance that this is a sensible approach to the design optimization problem.

We start with the following simple, but fundamental idea: use a *sequence* of surrogate objectives to *predict* optimal design candidates.

We do so because we wish to concentrate on the *predictive* ability of the surrogate and concern ourselves with *accuracy* only in the vicinity of a minimizer.

2. Optimization Using Approximations. We begin by noting that our basic strategy should work with any type of surrogate for the objective. We distinguish between *models* and *approximations*.

Surrogate models are auxiliary simulations that are less physically faithful, but also less computationally expensive, than the expensive simulations that are 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 [4]. 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) [14] and the kriging estimates employed in the design and analysis of computer experiments (DACE) [23, 1]. Once the approximation has been constructed, it is typically inexpensive to evaluate.

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 [18, 17] to handle the optimization, though other approaches are possible. We choose pattern search techniques because they can be easily amended to exploit surrogates, 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 [9].

Our approach [21] 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.

2.1. Basic Strategy. We now outline the basic strategy we will use:

1. Choose:
   (a) an initial grid over the feasible region $[\ell, u]$ that signifies the degree of resolution desired (this can be refined later, as deemed appropriate) and
   (b) an initial baseline design $x_c \in [\ell, u]$ at which f is known.
2. Perform an initial computer experiment:
   (a) select N initial design sites,
   (b) evaluate the true objective f at the initial design sites, and
   (c) construct an initial approximation a of f based on the objective values obtained at the design sites.
3. Do until a minimizer of f has been confirmed (for the current resolution of the grid) or until the "budget" V of evaluations has been exhausted:
   (a) find a candidate x_t that minimizes a on the grid and treat x_t as a site at which a *predicts* a minimizer for f on the grid.
   Evaluate $f(x_t)$.
   (b) Update the approximation a to include the objective value $f(x_t)$.

(c) If $f(x_t) < f(x_c)$, then let $x_c = x_t$. Else leave x_c unchanged.

(d) Repeat Step 3.

2.2. Remarks on the Basic Strategy. There are numerous remarks to be made regarding this basic strategy.

First, we use the grid to ensure the procedure is *robust*; with additional refinements of the grid, we are assured that asymptotically the sequence of designs x_c converges to a constrained stationary point of f under certain mild assumptions [10, 11, 16]. Furthermore, ours is a feasible point method since it has been our experience that simulations often will fail at infeasible points—and still may fail at feasible points [2, 3]. Any feasible x_c may be chosen to start the search and only feasible candidates x_t will be considered.

We can use any of a wide variety of approximation techniques. We favor those that are amenable to updates; in particular, we use the kriging techniques favored in the design and analysis of computer experiments (DACE) literature [23] because these are the approximation techniques with which we have some experience. For simplicity, we use cubic splines for the illustrative examples that follow. (We note that spline interpolation is mathematically equivalent to kriging. See, for example, [22].) The interpolatory approximations we have favored to date allow us to incorporate the new objective values $f(x_t)$ obtained during the course of the optimization in a straightforward fashion, though we will have more to say about potential difficulties.

We also can use any of a wide variety of optimization methods to produce a candidate x_t. We have used both quasi-Newton methods [21] and pattern search methods [3] to find a candidate x_t. For simplicity we use the "eyeball" method for the examples that follow. Again we stress that we insist on a candidate x_t that is on the grid to ensure convergence to a stationary point of f.

The convergence theory requires strict improvement of f at x_c to prevent "cycling." We seek to confirm x_c is a local stationary point of f at the current resolution of the grid by evaluating f at the $2p$ adjacent grid points defined by the positive and negative coordinate directions. Again, this point should be made clearer in the examples that follow.

3. Simple Examples. We show some simple one-dimensional examples for illustrative purposes. In the sequence of graphs that display our results, the objective f is indicated using a solid line while the approximation a is indicated using a dashed line. The grid upon which we are operating is shown in "hatch" marks along the x-axis. The points at which $f(x)$ is known are denoted with open circles.

When we have only two known points (which is what we use in the initial experimental design) we build a simple linear interpolant. We distinguish these two initial design sites by including a + mark in the open circle. When we have three known points (after the first optimization step), we build a quadratic interpolant. For subsequent iterations, we use a cubic spline interpolant [12], which for the one-dimensional examples we show requires a minimum of four data points.

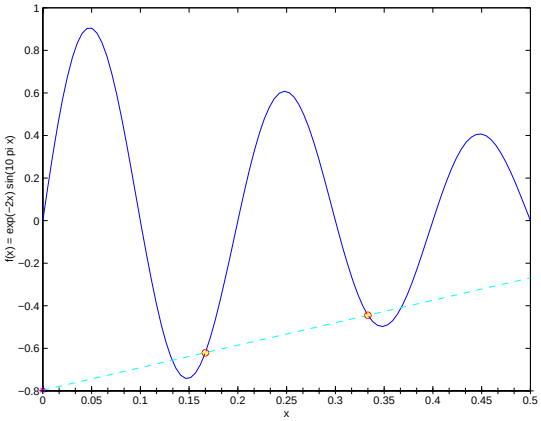
The candidate x_t for which the approximation a predicts decrease in f—and at which we will evaluate f before proceeding—is denoted with a star. When the approximation does not predict any further decrease in f, the star indicates the point at which we will evaluate f (if unknown) to confirm that x_c is a local stationary point of f on the grid.

One word of warning: the graphs that follow were generated using Matlab. The sequence of graphs presented have been scaled by Matlab to best portray the range of values specific to that particular graph; thus, different scales for the y-axis are used for different graphs to capture the changing features of the interpolant as additional points are added. (The scale of the x-axis remains constant for each example.)

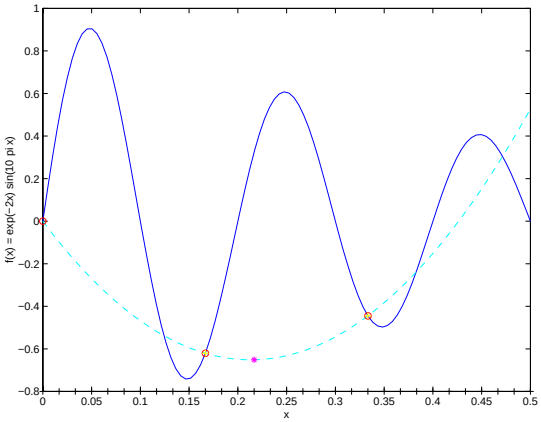
3.1. First Example. We start with the following simple example:

$$\begin{aligned} \text{minimize} \quad & f(x) = e^{-2x} sin(10\pi x) \\ \text{subject to} \quad & x \in [0, 0.5], \end{aligned}$$

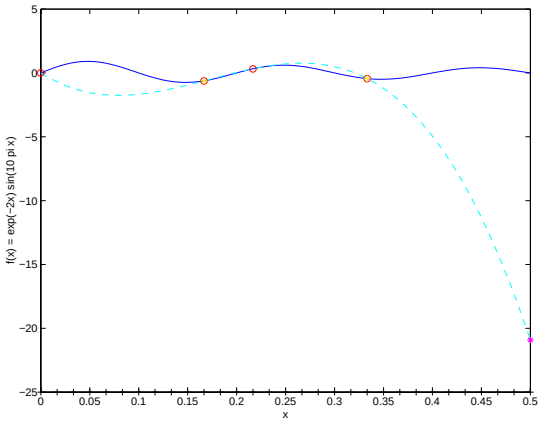
Applying the basic strategy outlined above, we obtain the following sequence of iterations:



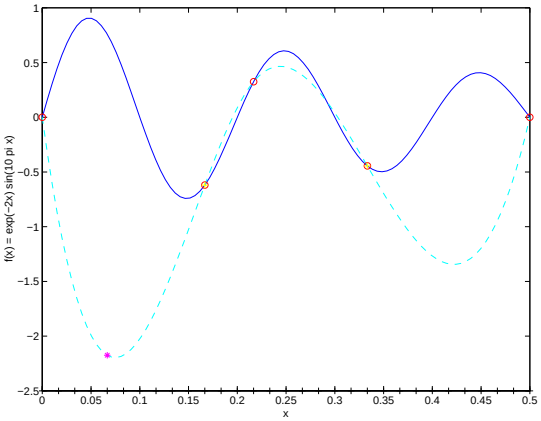
ITERATION 1: *Interpolation with 2 base points*



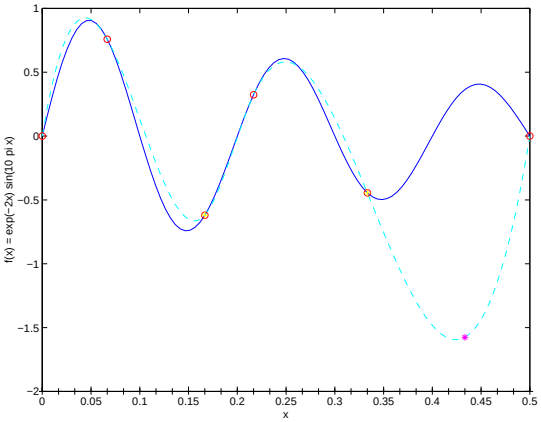
ITERATION 2: *Interpolation with 2 base points + 1 opt point*



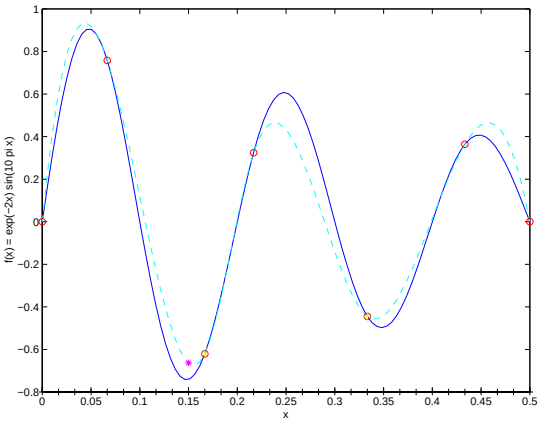
ITERATION 3: *Cubic spline interpolation with 2 base points + 2 opt points*



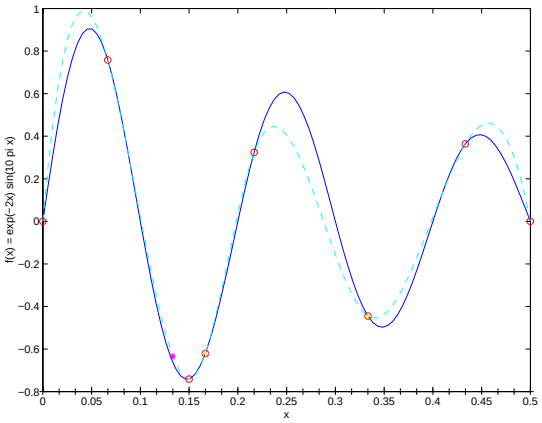
ITERATION 4: *Cubic spline interpolation with 2 base points + 3 opt points*



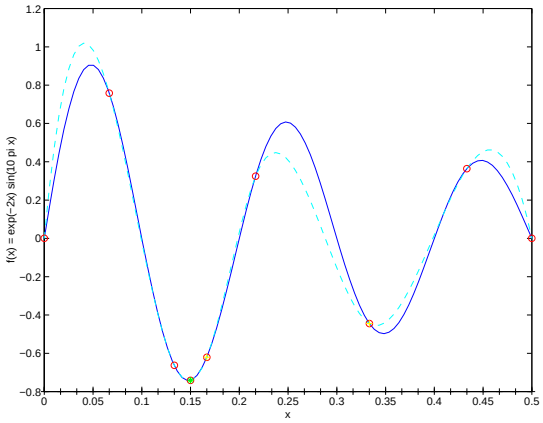
ITERATION 5: *Cubic spline interpolation with 2 base points + 4 opt points*



ITERATION 6: *Cubic spline interpolation with 2 base points + 5 opt points*



ITERATION 7: *Cubic spline interpolation with 2 base points + 6 opt points*



ITERATION 8: *Cubic spline interpolation with 2 base points + 6 opt points + 1 confirmation point*

Consider the sequence of proposed solutions using the interpolation coupled with the optimization, as shown in Table 3.1. We denote by x_c the current best solution for the "true" problem $f(x)$ and by x_t the "trial" solution proposed by the approximation a. Alternatively (as we see at the end of the process), x_t may be a "confirmation point" if the approximation suggests that a minimizer has already been identified. So, for instance, at Iteration 7, we know that the function increases from the value at $x_c = 0.150$ if we move immediately to the right (to 0.167), but we have not yet ascertained if further decrease is possible if we move immediately to the left (to 0.133). Thus, we evaluate f at 0.133 to "confirm" that 0.150 is a (local) minimizer.

Iteration	x_c	x_t
1	0.167	0.000
2	0.167	0.217
3	0.167	0.500
4	0.167	0.067
5	0.167	0.433
6	0.167	0.150
7	0.150	0.133
8	0.150	—

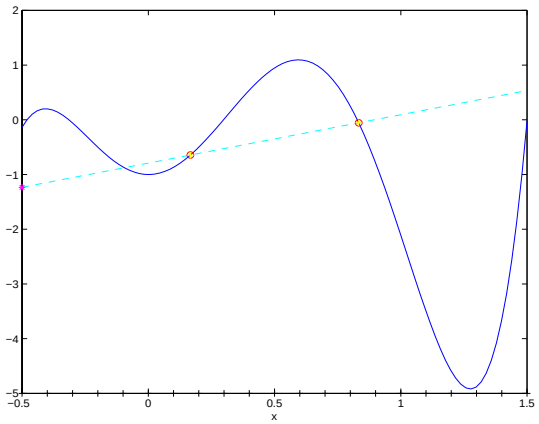
TABLE 3.1
Sequence of iterates and trial points for the first example

Note that while the approximation a is poor, we do not realize any improvement on f; the best solution to the problem of minimizing f is the one identified by one of the two initial sites for sampling f to construct a linear interpolant. But by Iteration 6, the approximation has identified the minimizer. Iteration 7 confirms that the candidate improves upon the current solution for f and, in fact, gives a global solution for the approximation a. Thus, we are left to confirm that this candidate is indeed a minimizer for f. To confirm this, we need the value of the true objective f at the two adjacent grid points. We already have computed the value of f at $x = 0.167$, so what remains is to compute the value of f at $x = 0.133$.

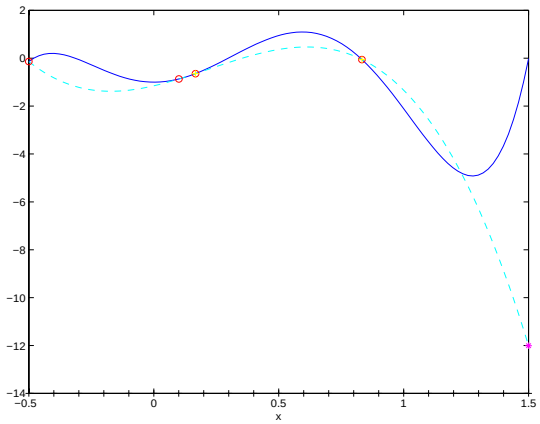
At Iteration 8 we stop with a *confirmed* (local) minimizer of f (at least to the resolution of the grid) at $x_* = 0.150$ and a decent approximation a to our objective f on the interval $[0, 0.5]$.

Before moving on to our next example, we note the following. It is often said that if a *model* of the objective or a constraint function has a weakness, the optimization procedure will find it. Here we see the same phenomenon using *approximations* of the objective. In the early stages of the optimization process, the approximation predicts a minimizer in precisely those region(s) where we have the least information about the objective: consider Iterations 3, 4, and 5. But by using the iterative process, we can improve the approximation with the eventual goal of using it to successfully *predict* a minimizer of f.

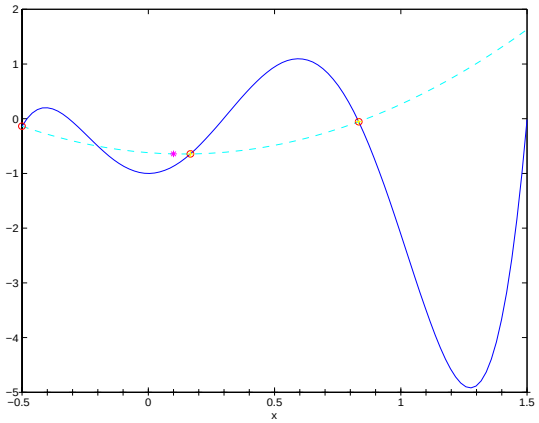
3.2. Second Example. Now to illustrate the main contribution of our proposal, we will reconsider the process with an objective function constructed to illustrate that our basic strategy, while *provably* robust enough to guarantee asymptotic convergence to a *local* minimizer, may not always produce a global minimizer—which is often the preferred solution. Consider the following sequence of iterations:



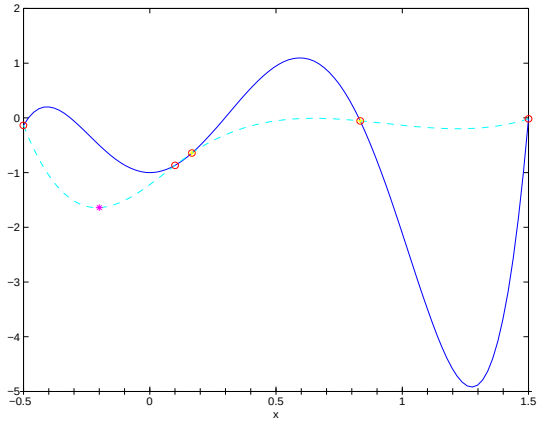
ITERATION 1: *Interpolation with 2 base points*



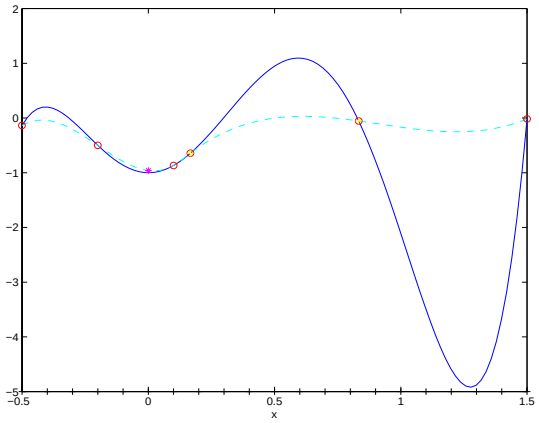
ITERATION 3: *Cubic spline interpolation with 2 base points + 1 opt point + 1 confirmation point*



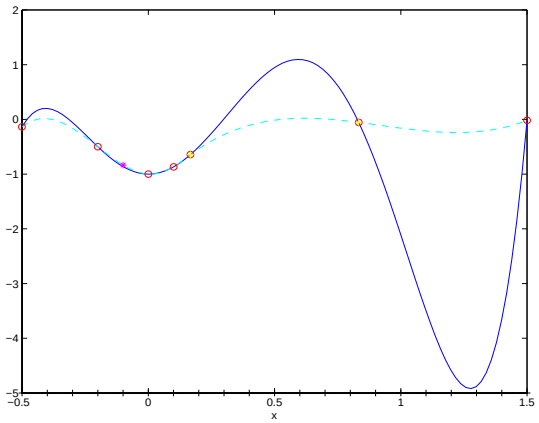
ITERATION 2: *Interpolation with 2 base points + 1 opt point*



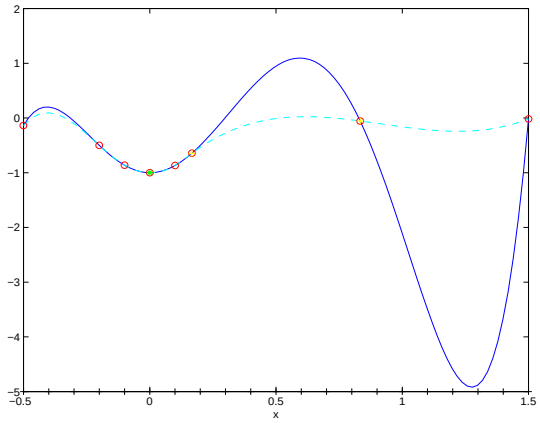
ITERATION 4: *Cubic spline interpolation with 2 base points + 2 opt points + 1 confirmation point*



ITERATION 5: *Cubic spline interpolation with 2 base points + 3 opt points + 1 confirmation point*



ITERATION 6: *Cubic spline interpolation with 2 base points + 4 opt points + 1 confirmation point*



ITERATION 7: *Cubic spline interpolation with 2 base points + 4 opt points + 2 confirmation points*

Notice that when we are done we have a confirmed minimizer at 0.0 and our cubic spline approximation is quite accurate in the neighborhood of the minimizer $[-0.2, 0.167]$. But the approximation is not accurate in the region $[0.167, 1.5]$; it suggests that at best there is a local minimizer at either 1.2 or 1.3, when in fact 1.3 is the global minimizer for this function.

The reason we found the local minimizer (fairly quickly, as Table 3.2 shows) is a result of our initial design for constructing an approximation. One of our two data points lies in the basin of attraction for the local minimizer, while we have no data in the basin of attraction for the global minimizer.

The question then becomes, how to balance our desire to find a confirmed minimizer with our desire to "know" enough about the objective to avoid missing a potentially better solution simply because the approximation may not be sufficiently good to predict a better solution?

4. An Alternate Strategy. In the second example illustrated above, we reach a point in the optimization process (Iteration 6) where the approximation suggests we have found a local minimizer of the objective—which happens to be a global minimizer of the current approximation. Using the simple strategy with which we started, we then proceed to evaluate nearby points to confirm that we have a local minimizer of the objective. We ignore indications that there may be another (local) minimizer of the objective in the region [0.833,1.5]—which happens to be a region where we know very little about the objective.

Instead of being so single-minded in our pursuit of a single minimizer, what if as an alternate search

Iteration	x_c	x_t
1	0.167	−0.500
2	0.167	0.100
3	0.100	1.500
4	0.100	−0.200
5	0.100	0.000
6	0.000	−0.100
7	0.000	—

TABLE 3.2
Sequence of iterates and trial points for the second example

strategy we search both in regions where the approximation indicates there might be a minimizer of the objective *and* where we realize that we "know" very little about the objective? The question then becomes: how do we formalize this notion?

What we propose is to combine these two goals into a single objective, which we refer to as a *merit function*. The idea is to balance these two goals much as one does for a multiobjective optimization problem or (possibly) as a way to handle constraint violations.

4.1. A Family of Merit Functions. One possibility is to use a merit function of the form:

$$m_c(x) = a_c(x) - \rho_c d_c(x),$$

where $\rho_c \geq 0$ and

$$d_c(x) = \min \|x - x_i\|_2,$$

where the minimum in d_c is computed over all points x_i at which we know the value of the true objective. Thus, $d_c(x)$ is the distance from x to the nearest previous design site.

The merit function m_c comprises two components, a_c and d_c. The approximation function a_c plays the same role as before: we want to minimize (or at least decrease) the value of a_c at x_c as a way of predicting decrease on $f(x_c)$. The second function is an experimental design criterion that is intended to inhibit clustering and thereby to ensure that the trial point is placed where information obtained from evaluating the objective will be helpful. Many such criteria are possible; ours is adapted from the *maximin distance design* criterion proposed in [8].

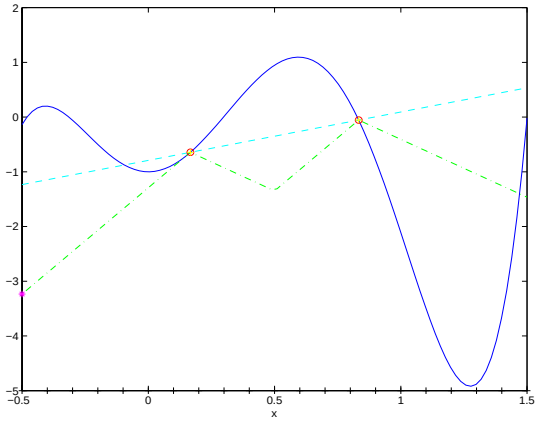
A useful analogy for explaining the maximin design criterion is the problem of building a chain of convenience stores. Regardless of how many stores one wishes to build, it is desirable to spread them around the city in question. One natural possibility is to locate stores so as to maximize the shortest distance between two stores. This is what is known as a maximin distance design. Although maximin designs are easily computed in one dimension, they may be quite difficult to compute in many dimensions. Fortunately, one can compute various approximate maximin designs using conventional nonlinear programming algorithms, subject to the usual caveats about the possibility that such algorithms will find nonglobal solutions [19].

The maximin design criterion is an example of a sensible, *space-filling* design criterion. Of course, the design space rarely is full in an absolute sense because typically only a relatively small number of design sites are used. Rather the term space-filling is meant to indicate that the design sites are spread out and do not cluster in one portion of the experimental region.

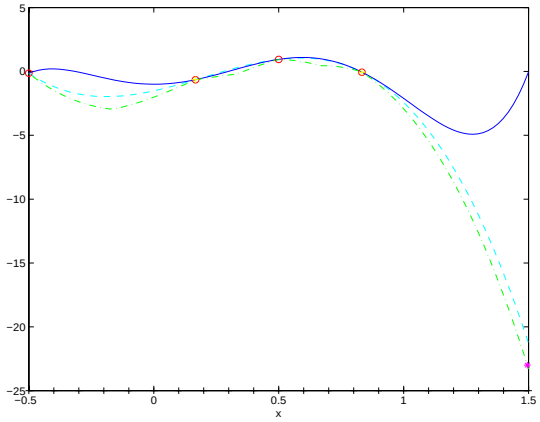
So how does the maximin design criterion figure into our merit function? By subtracting some positive multiple of our maximin design criterion d_c, we are giving merit to designs that *maximize* the shortest distance between two observations because a large value for $d_c(x_t)$ will further decrease the value of $m_c(x_t)$.

We still use the approximation a to predict candidate minimizers of the "true" objective f, but we now weight our predictions based on how close we are to known information about the objective. Our choice of ρ_c dictates how much emphasis we will place on learning more about the objective in regions for which we have no samples versus emphasizing the rapid identification of a (local) minimizer. For the example that follows, we have kept ρ_c constant, but this is a quantity that can be varied at each iteration and, in fact, should eventually tend to zero in order to ensure convergence to a local solution.

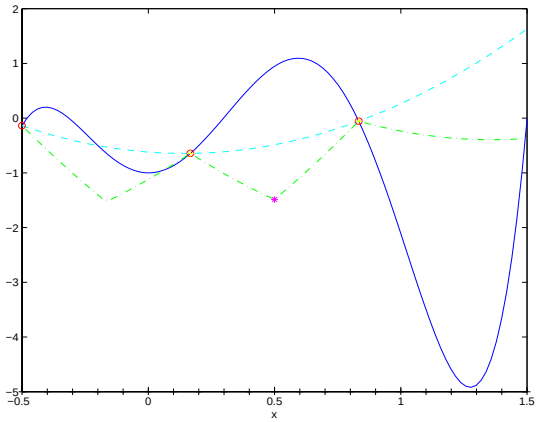
4.2. Second Example Revisited. So how well does this work? The following sequence repeats our second example but now uses $m_c(x)$ (shown using a "dot-dash" line) to choose candidates to minimize the objective. For this example, we used $\rho_c = 3$. (There is no real motivation for this choice of ρ_c; it was the first value tried and it happened to work particularly well for this example.)



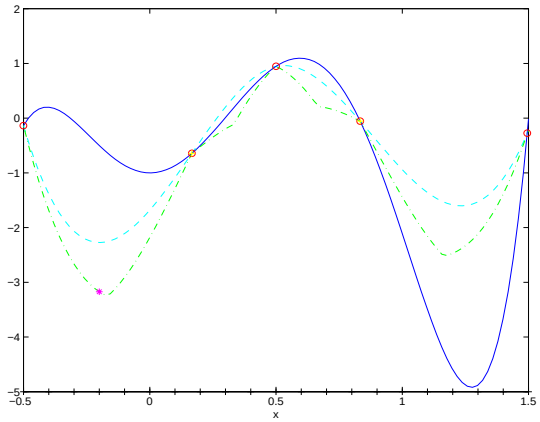
ITERATION 1: *Merit function for interpolation with 2 base points*



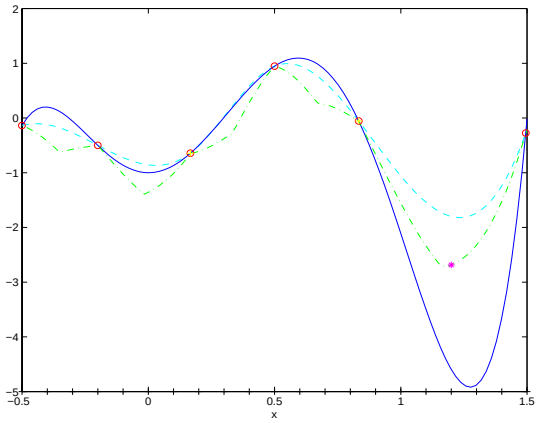
ITERATION 3: *Merit function for cubic spline interpolation with 2 base points + 2 opt points*



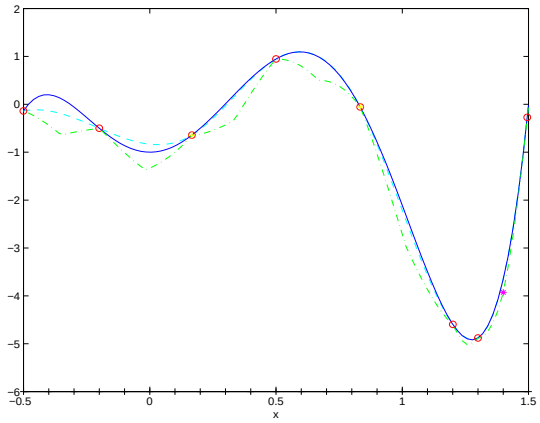
ITERATION 2: *Merit function for interpolation with 2 base points + 1 opt point*



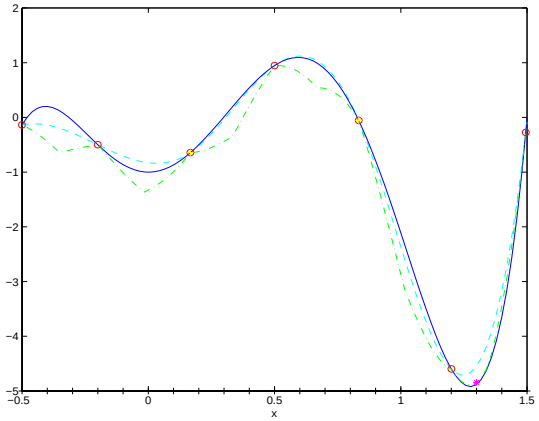
ITERATION 4: *Merit function for cubic spline interpolation with 2 base points + 3 opt points*



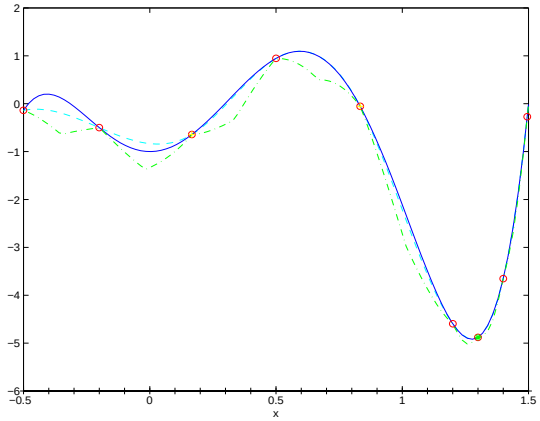
ITERATION 5: *Merit function for cubic spline interpolation with 2 base points + 4 opt points*



ITERATION 7: *Merit function for cubic spline interpolation with 2 base points + 6 opt points*



ITERATION 6: *Merit function for cubic spline interpolation with 2 base points + 5 opt points*



ITERATION 8: *Merit function for cubic spline interpolation with 2 base points + 6 opt points + 1 confirmation point*

Again we consider the sequence of proposed solutions using our merit function coupled with the optimization, shown in Table 4.1. As before, we denote by x_c the current best solution for the "true" problem defined by $f(x)$. But now x_t is the solution proposed by the merit function rather than by the approximation.

Iteration	x_c	x_t
1	0.167	−0.500
2	0.167	0.500
3	0.167	1.500
4	0.167	−0.200
5	0.167	1.200
6	1.200	1.300
7	1.300	1.400
8	1.300	—

TABLE 4.1

Sequence of iterates and trial points for the second example, using the merit function

Note how different the outcome now is. When we stop this time, we have a confirmed (to the scale of res-

olution) minimizer of the objective—which now happens to be the global minimizer—and an approximation that is qualitatively much better than the one we realized previously for this same objective function.

It is the case that because we now assign "merit" to sampling in regions for which we have no information about the objective, we are slower to improve the best known solution to the optimization problem. For this example, x_c is not replaced until Iteration 6 whereas, earlier, working only with the approximation a, we replace x_c by Iteration 3.

On the other hand, using the merit function—rather than just the approximation—enforces a better "spread" of the sample points throughout the design region. The beneficial effects of this are apparent by Iteration 5. Earlier, when we did not make use of the merit function, with the exception of the two end points, all our sampling occurred in the basin with a local minimizer at 0.0; thus, the behavior of the objective in the rest of the design space is largely unknown. However, when we use the merit function to select candidate designs, by Iteration 5 even the *approximation* indicates that a global minimizer should be in the region near 1.2; the merit function only further emphasizes this prediction.

Note that the approximation a still predicts the local minimizer at 0.0—and the merit function emphasizes this prediction too. If we wished to search for other minimizers, we certainly have adequate predictive ability to do so. But now the global minimizer dominates. Having identified it, we are content to stop.

4.3. Summary. Our illustrations are necessarily simple and the results that we obtained are better than should be expected for more complicated problems. Nonetheless, we believe that two conclusions can be drawn:

1. Global approximations can be helpful in facilitating optimization.
2. There is merit in balancing the goal of finding minimizers of the current approximation against the goal of constructing better approximations for future iterations.

Again, we stress that the latter observation is not unique to us. Recent work from the nonlinear programming community [15, 5] has emphasized the need to monitor the "geometry" of the sample points even when using them to build a local quadratic approximation to the objective—in particular, to avoid the numerical ill-conditioning that can arise if such considerations are ignored. Furthermore, a standard theme in global optimization [13, 7, 6] is the need to sample in regions where relatively little is known about the objective in an effort to devise heuristics that are more likely to produce global minimizers.

In higher dimensions, we are not sanguine that good approximations of the objective can be constructed over the entire design region. The curse of dimensionality is likely to hold sway for all but the simplest of objective functions. Nonetheless, we wish to avoid the tendency of most derivative-based optimization techniques to find the local minimizer nearest the initial trial point used to start the optimization process. Furthermore, by attempting to construct a global rather than a local approximation to the objective, we can use such tools as analysis of variance (ANOVA) to discern other information about the objective either before or after the optimization process.

4.4. Outstanding Issues. Of course, there is no such thing as a free lunch. In essence, we have turned our original single objective problem (find a minimizer to the approximation a to predict a minimizer for the objective f) into a biobjective problem (find a point that both improves on the value of the approximation a *and* improves the quality of the approximation of the regions in which we have not yet sampled to predict a minimizer for the objective f). Once we do so, we raise the usual issues one finds in multiobjective optimization: to wit, how to balance the two objectives in a sensible fashion. For now, we introduce the parameter ρ_c and leave it to the user's control—which would be considered a feature by some users and burden by others.

There is also the issue of how to do the optimization on either the approximation a or the merit function m to predict a candidate minimizer for the objective f. Obviously, the approximation is constructed so that it is inexpensive to compute; otherwise, we would work directly with the objective. If the objective is particularly expensive to computer, then we can afford to spend some time solving the "easier" optimization problem posed by the approximation. Nonetheless, if we are interested in a global minimizer of the approximation, finding such a minimizer becomes more problematic for all but the simplest problems in higher dimensions. When we move to the family of merit functions we have proposed, the problem is necessarily compounded. As the example should make clear, we necessarily introduce multiple local minimizers. Fortunately, the special structure of the family of merit functions suggests specialized optimization techniques for identifying potential minimizers—but this is a subject for ongoing research.

We are encouraged by our preliminary results, but there remains much work to be done to answer the research questions raised and to help ascertain practical choices for the many options that face us and guidelines to suggest to users as to when these approaches are likely to be of most use.

5. Conclusions. There are now several encouraging reports of numerical experiments that commend the sequential use of approximations when optimizing expensive objective functions [21, 16, 3]. We have outlined 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 paper 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 [20].

Acknowledgments. We first presented this work at the ISSMO/NASA First Internet Conference on Approximations and Fast Reanalysis in Engineering Optimization, June 14–27, 1998. We would like to thank both the organizers of the conference for the invitation to participate and the other participants for the questions and comments made during the course of the conference.

This research evolved from an ongoing collaboration with John Dennis (Rice University); David Serafini (Lawrence Berkeley National Lab); Andrew Booker, Paul Frank, and Greg Shubin (Boeing); and Andrew Conn and Katya Scheinberg (IBM). Although we have been profoundly influenced by the ideas of our collaborators, some of the opinions expressed in this paper may not be shared by all members of the collaboration.

REFERENCES

[1] A. J. Booker, *DOE for computer output*, Tech. Report BCSTECH–94–052, Boeing Computer Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124, December 1994.

[2] A. J. Booker, A. R. Conn, J. E. Dennis, P. D. Frank, D. Serafini, V. Torczon, and M. Trosset, *Multi-level design optimization: A Boeing/IBM/Rice collaborative project. 1996 final report*, Tech. Report ISSTECH–96–031, Boeing Information & Support Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124, December 1996.

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

[4] K. J. Chang, R. T. Haftka, G. L. Giles, and P.-J. Kao, *Sensitivity-based scaling for approximating structural response*, Journal of Aircraft, 30 (1993), pp. 283–288.

[5] A. R. Conn and P. L. Toint, *An algorithm using quadratic interpolation for unconstrained derivative free optimization*, in Nonlinear Optimization and Applications, G. Di Pillo and F. Giannessi, eds., Plenum Publishing, New York, 1996, pp. 27–47. Proceedings of the International School of Mathematics "G. Stampacchia" 21st Workshop, Erice, Italy , June 13–21, 1995.

[6] D. D. Cox and S. John, *SDO: A statistical method for global optimization*, in Multidisciplinary Design Optimization: State-of-the-Art, N. M. Alexandrov and M. Y. Hussaini, eds., SIAM, Philadelphia, 1997, pp. 315–329.

[7] P. D. Frank, *Global modeling for optimization*, SIAG/OPT Views-and-News: A Forum for the SIAM Activity Group on Optimization, 7 (1995), pp. 9–12.

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

[9] R. M. Lewis and V. J. Torczon, *A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds*, Tech. Report 98-31, Institute for Computer Applications in Science and Engineering (ICASE), Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681–2199, 1998.

[10] ——, *Pattern search algorithms for bound constrained minimization*, 1998. To appear in *SIAM J. on Optimization*.

[11] ——, *Pattern search methods for linearly constrained minimization*, Tech. Report 98-3, Institute for Computer Applications in Science and Engineering (ICASE), Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681–2199, January 1998.

[12] C. F. V. Loan, *Introduction to Scientific Computing: A Matrix-Vector Approach Using Matlab*, Prentice-Hall, Upper Saddle River, NJ, 1997.

[13] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*, Kluwer Academic Publishers, Dordrecht, 1989.

[14] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments*, Wiley-Interscience, New York, 1995.

[15] M. J. D. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*, in Advances in Optimization and Numerical Analysis, Proceedings of the 6th Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico, vol. 275, Dordrecht, 1994, Kluwer Academic Publishers, pp. 51–67.

[16] D. B. Serafini, *A Framework for Managing Models in Optimization for Computationally Expensive Functions*, PhD thesis, Department of Computational and Applied Mathematics, Rice University, P.O. Box 1892, Houston, Texas 77005–1892, 1998.

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

[18] V. Torczon and M. W. Trosset, *From evolutionary operation to parallel direct search: Pattern search algorithms for numerical optimization*, Computing Science and Statistics, 29 (1998), pp. 396–401.

[19] M. W. Trosset, *Approximate maximin designs*. In progress.

[20] ——, *Optimization on a limited budget*, in 1998 Proceedings of the Section on Physical and Engineering Sciences, American Statistical Association. To appear, 1999.

[21] M. W. Trosset and V. Torczon, *Numerical optimization using computer experiments*, Tech. Report 97–38, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley

Research Center, Hampton, Virginia 23681–2199, 1997.

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

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