AIAA 2002-0538

USE OF DATA SAMPLING, SURROGATE MODELS, AND NUMERICAL OPTIMIZATION IN ENGINEERING DESIGN

Anthony A. Giunta[*]
Sandia National Laboratories[†]
Optimization and Uncertainty Estimation Department
Albuquerque, NM, USA

Abstract

An engineering design study was performed using computational simulation software coupled with the DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) software package. This study made use of the data sampling methods in DAKOTA, which included grid-based parameter studies and Latin hypercube sampling. Multidimensional surface fitting methods such as quadratic polynomial regression were used to smooth out numerical noise generated by the computational simulation software. This enabled the application of a surrogate-based optimization algorithm to solve the design problem. These results serve as a case study that demonstrates the utility of employing a combination of statistical methods and optimization methods in engineering design.

Keywords: *Latin hypercube sampling, response surface approximation, surrogate-based optimization, metamodels, numerical noise, nonsmooth, parallel computing*

1. Introduction

The objective of this study is to demonstrate the use of statistical data sampling methods, surrogate modeling methods, and numerical optimization techniques, to perform design optimization on problems that are difficult or intractable using conventional gradient-based methods. Typical computational simulations used in engineering analyses exhibit nonsmooth trends or "numerical noise" in response quantities of interest (e.g., stress values, viscous drag estimates, heat transfer rates) due to factors such as limited grid resolution, fixed iteration counts in iterative solvers, and discrete values in tabular databases. This noise inhibits the use of traditional gradient-based optimization methods since gradients estimated using finite differences can be inaccurate or may not exist. While this noise is not stochastic, data sampling and statistical analysis methods can be applied to quantify the magnitude of the noise. If the magnitude of the noise does not obscure the overall trends in the response quantities of interest, then a common approach is to use surrogate modeling methods, also known as response surface approximation methods or metamodels, to smooth out the numerical noise and to permit gradient-based optimization.[1,2,3]

The DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) toolkit[4] has been under development at Sandia National Laboratories since 1994. Originally developed as collection of gradient-based and nongradient-based optimization software, DAKOTA now includes methods for statistical data sampling, surrogate modeling methods, and surrogate-based optimization strategies (SBO), plus a variety

[*] Senior Member of Technical Staff, Senior Member, AIAA
[†] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

of other statistical and mathematical software tools useful to the design engineer.

This report describes of application of various methods in the DAKOTA toolkit to solve an engineering design problem that could not be solved using traditional gradient-based optimization. Section 2 of this report provides an overview of the DAKOTA software, while Sections 3 and 4 cover data sampling methods and surrogate modeling methods, respectively. Section 5 provides information on the surrogate-based optimization algorithm in DAKOTA. The application of SBO to an engineering design optimization study is given in Section 6, and concluding remarks are presented in Section 7.

2. DAKOTA Software

The DAKOTA toolkit is a software framework for systems analysis that includes methods for optimization, nonlinear parameter estimation, uncertainty quantification, design of experiments, statistical sampling methods, parameter studies, sensitivity analysis, and surrogate models (i.e., metamodels, response surface approximations).

DAKOTA employs a generic interface capability based on UNIX commands that permit the linking of DAKOTA to a variety of commercial and custom-developed engineering and physics simulation codes. Figure 1 depicts this generic interface, or "black-box," coupling approach between DAKOTA and a simulation code. That is, DAKOTA and the simulation code remain entirely independent, with data being transferred between DAKOTA and the simulation code through writing and reading text files. During the execution of DAKOTA, the UNIX `system` command is used to run the simulation code along with any pre- and post-processing steps that are needed to exchange data between the two codes. A combination of text output and graphical output allow the user to monitor DAKOTA's progress during execution.

The DAKOTA toolkit is designed to exploit massively parallel computing platforms through a multi-level parallelism approach[5] which takes advantage of opportunities for concurrent function evaluations that are afforded by the different optimization algorithms. For example, a two-level parallelism strategy is often used when many function evaluations (i.e., simulation code runs) are needed to estimate gradients via finite difference approximation. On a parallel computer, the function evaluations are performed concurrently, where each individual function evaluation uses a domain-decomposition approach to run on multiple processors. Depending on the optimization algorithm in use and the nature of the engineering design problem, up to four levels of parallel computing can be utilized by DAKOTA. This multi-level parallelism approach enables the user to achieve near-linear scaling on massively parallel computers.

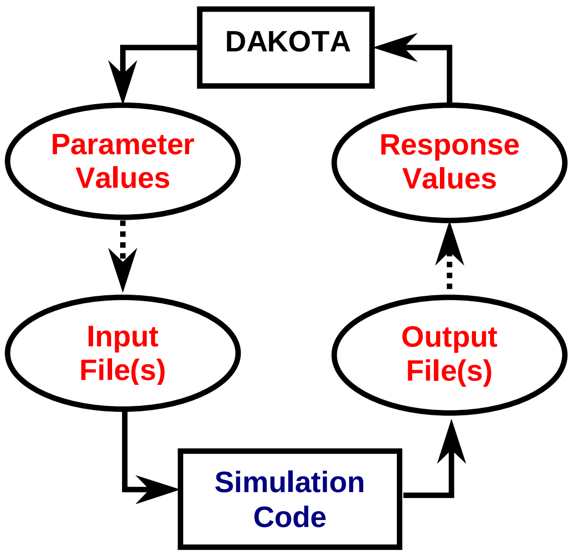

Figure 1. This flowchart demonstrates the "black-box" coupling between DAKOTA and computational simulation codes. The dashed lines represent file input/output.

Numerous optimization algorithms are available in DAKOTA. These include gradient-based nonlinear programming methods, nongradient-based pattern search and genetic algorithm methods, and mixed integer-continuous variable nonlinear programming methods. The flexibility, and extensibility, of the C++ object-oriented design approach used in creating DAKOTA permits the rapid development of more sophisticated optimization strategies such as surrogate-based optimization, hybrid optimization (e.g., a mix of nongradient- and gradient-based methods), and optimization under uncertainty. Additional capabilities in DAKOTA that pertain to this study are described below. For a more complete description of DAKOTA's capabilities, consult the DAKOTA online manuals and web site.[6, 7]

3. Data Sampling Methods

3.1 Background

Many sampling methods are available for choosing sample sites in a parameter space. Classical design of experiments (DOE) methods were originally developed for use in planning physical experiments where there are random variations that cannot be eliminated. Sampling methods developed in classical DOE include full factorial designs, central composite designs, Box-Behnken designs, and many variants.[8]

As computer simulations (i.e., computational experiments) became more widely used in scientific research, new data sampling approaches were developed. Known as design and analysis of computer experiments (DACE), these approaches include Monte Carlo sampling, Latin hypercube sampling,[9] orthogonal array sampling,[10] and other approaches. DACE differs from classical DOE methods in that it is based on the notion of deterministic computer simulations rather than on stochastic physical experiments. Because of this assumption, DACE sampling methods tend to be more "space-filling" than classical DOE methods. That is, DACE methods tend to spread the samples throughout the parameter space, whereas DOE methods tend to concentrate samples on the boundaries of the parameter space.

Latin hypercube sampling (LHS) is one popular DACE method that has found wide application in computational applications. Reasons for its popularity include its computational simplicity and its theoretical underpinnings that show it can have less variance error than Monte Carlo (random) sampling. In addition, LHS gives the user the ability to tailor the number of sample sites to the available computational budget. For example, with an expensive computer simulation with n design parameters, one may only be able to afford $O(n)$ samples. In contrast, many classical DOE methods have a rigid requirement for the number of samples, and typically the number of samples scales as $O(n^2)$.

3.2 Latin Hypercube Sampling

The Latin hypercube sampling method was developed by McKay, et al.,[9] as an alternative to random sampling. Under certain monotonicity conditions associated with the function to be sampled, Latin hypercube sampling provides a more accurate estimate of the mean value than does random sampling. That is, given an equal number of samples, the LHS estimate of the mean will have less variance than the mean value obtained through random sampling.

Figure 2 demonstrates Latin hypercube sampling on a two-variable parameter space. Here, the range of both parameters, X_1 and X_2, is [0,1]. Also, for this example both X_1 and X_2 have uniform statistical distributions. For Latin hypercube sampling, the range of each parameter is divided into p "bins" of equal probability. For parameters with uniform distributions, this corresponds to partitions of equal size. For n design parameters, this partitioning yields a total of p^n bins in the parameter space. Next, p samples are randomly selected in the parameter space, with the following restrictions: (a) each sample is randomly placed inside a bin, and (b) for all one-dimensional projections of the p samples and bins, there will be one and only one sample in each bin.

In a two-dimensional example such as that shown in Figure 2, these LHS rules guarantee that only one bin can be selected in each row and column. For $p=4$, there are four partitions in both X_1 and X_2. This gives a total of 16 bins, of which four will be chosen according to the criteria described above. Note that there is more than one possible arrangement of bins that meet the LHS criteria. The stars in Figure 2 represent the four sample sites in this example, where each sample is randomly located in its bin. There is no restriction on the number of bins in the range of each parameter, however, all parameters must have the same number of bins.

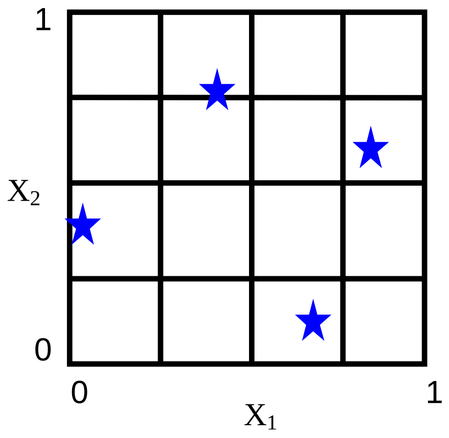

Figure 2. Latin hypercube sampling with four bins in each of the parameters X_1 and X_2. The stars are sample sites randomly selected inside each bin.

3.3 Data Sampling Software

The DAKOTA toolkit contains two software packages that can be used to perform Latin hypercube sampling, as well as a variety of other DACE and classical DOE sampling methods.

The LHS software package[11] provides both Monte Carlo (random) sampling and Latin hypercube sampling methods, which can be used with probabilistic variables in DAKOTA that have the following distributions: Gaussian (normal), lognormal, uniform, loguniform, Weibull, and user-supplied histograms. In addition, the user can supply a correlation matrix for the variables to account for correlations among the variables.

The DDACE (Distributed Design and Analysis of Computer Experiments) software package[12] includes both stochastic sampling methods and classical design of experiments methods. The stochastic methods are random sampling, Latin hypercube sampling, and orthogonal array sampling. The DDACE package currently supports variables that have either normal or uniform distributions. However, only the uniform distribution is available in the DAKOTA interface to DDACE. The classical design of experiments methods in DDACE are central composite design, Box-Behnken sampling, and full factorial sampling.

4. Surrogate Modeling Methods

4.1 Background

Surrogate model construction, also known as surface fitting, metamodeling or response surface approximation, is often used in engineering design applications to provide insight on the trends exhibited by response quantities (objective function, constraints, etc.) with respect to changes in design parameter values. Once surrogate models are constructed, they provide an inexpensive substitute for the physical experiment or computational simulation from which the original response data were generated. For this reason, surrogate models have been extensively used in numerically intensive studies such as optimization and uncertainty quantification. Of course, the challenge of using surrogate models for such purposes is that they must be of sufficient accuracy to the original experiment or simulation.

DAKOTA contains several types of surface fitting methods that can be used in SBO. These are: quadratic polynomial models, first-order Taylor series expansion, kriging spatial interpolation, stochastic layered perceptron artificial neural networks, and multivariate adaptive regression splines. All of these surface fitting methods can be applied to problems having an arbitrary number of design parameters. However, surface fitting methods usually are practical only for problems where the number of parameters is relatively small (e.g., typically 2-10 parameters, but potentially ranging up to 50 parameters, or more).

4.2 Polynomial Regression

The functional form of a quadratic polynomial is

$$y(\boldsymbol{x}) = c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n}\sum_{j=i}^{n} c_{i,j} x_i x_j + \varepsilon, \tag{1}$$

where n is the number of design parameters, $y(\boldsymbol{x})$ is the response value, the x_i, x_j terms are the design parameters, the c_0, c_i, $c_{i,j}$ terms are unknown coefficients, and ε is the residual error term. Given a set of response values and associated design parameters, Equation (1) can be used to form a system of linear equations, $\boldsymbol{y=Xc}$, where $\boldsymbol{y}$ is the vector of responses, $\boldsymbol{c}$ is the vector of coefficients and $\boldsymbol{X}$ is matrix of terms from Equation (1). If $\boldsymbol{X}$ has full rank, the method of linear least squares can be applied to determine the values of the polynomial coefficients.

Quadratic polynomials are often used in surrogate-based optimization due to their computational simplicity and their convenient algebraic form. In addition, polynomial regression is useful in smoothing out numerical noise that can appear in response function data. However, polynomials are not well-suited for all surface fitting applications, particularly when fitting response functions that have multiple, well-defined local extrema.[13] In such cases, more accurate surrogate models may be obtained using methods such as cubic or higher-order polynomial regression, kriging interpolation,[14] or artificial neural networks.

5. Surrogate-Based Optimization

The surrogate-based optimization (SBO) algorithm used in this study is based on the work of Giunta and Eldred,[15] Alexandrov, et al.,[16,17] and Rodriguez, et al.[18] The SBO algorithm solves a sequence of optimization problems, each of which is an approximation to the original optimization problem. For example, consider a general nonlinear programming problem of the form

$$\begin{aligned} \text{minimize: } & f(\boldsymbol{x}) \\ \text{subject to: } & g_i(\boldsymbol{x}) \le 0, \text{ for } i=1,\dots,m \\ & \boldsymbol{x}_L \le \boldsymbol{x} \le \boldsymbol{x}_U \end{aligned} \tag{2}$$

where $f(\boldsymbol{x})$ is a scalar-valued objective function, $g_i(\boldsymbol{x})$ are the m scalar-valued inequality constraints, $\boldsymbol{x}$ is a vector of design parameters, and the subscripts "L" and "U" denote lower and upper bounds, respectively, on $\boldsymbol{x}$. The SBO algorithm reformulates Equation (2) as

$$\begin{aligned} \text{minimize: } & \hat{f}(\boldsymbol{x}) \\ \text{subject to: } & \hat{g}_i(\boldsymbol{x}) \le 0, \text{ for } i=1,\dots,m \\ & \boldsymbol{x}_L \le \boldsymbol{x} \le \boldsymbol{x}_U \\ & \left\| \boldsymbol{x} - \boldsymbol{x}_c^k \right\|_\infty \le \Delta^k \\ & k=0,1,2,\dots,k_{max} \end{aligned} \tag{3}$$

where $\hat{f}(\boldsymbol{x})$ and $\hat{g}_i(\boldsymbol{x})$ are surrogate functions for the original objective and constraint functions. The added constraint $\left\| \boldsymbol{x} - \boldsymbol{x}_c^k \right\|_\infty \le \Delta^k$ in Equation (3) confines the k^{th} optimization subproblem to the trust region, Δ^k, where $\boldsymbol{x}^k{}_c$ is the center of the k^{th} trust region. Thus, optimization is performed on the surrogate model(s) rather than on the original, possibly expensive, simulation code(s).

One of the primary challenges in SBO is in finding, or constructing at an acceptable computational cost, surrogate models that accurately capture the trends in the original functions. At the end of each of the k iterations in the SBO algorithm there is a verification step that compares the value of the surrogate models with the true value of the response functions. This step, and the associated logic in the SBO algorithm, ensures that the trust region size is adjusted to maintain the accuracy of the surrogate models.

The use of SBO permits the application of gradient-based optimization methods to problems where there are clear trends in the parameter study data, but where small-scale nonsmooth behavior would lead to inaccurate finite-difference gradient estimates. The SBO algorithm in DAKOTA can be implemented using heuristic rules (less expensive) or provably-convergent rules (more expensive). The SBO algorithm is particularly effective on real-world engineering design problems that contain nonsmooth features (e.g., slope discontinuities, multiple local minima) where gradient-based optimization methods often have trouble, and where the computational expense of the simulation precludes the use of nongradient-based methods.

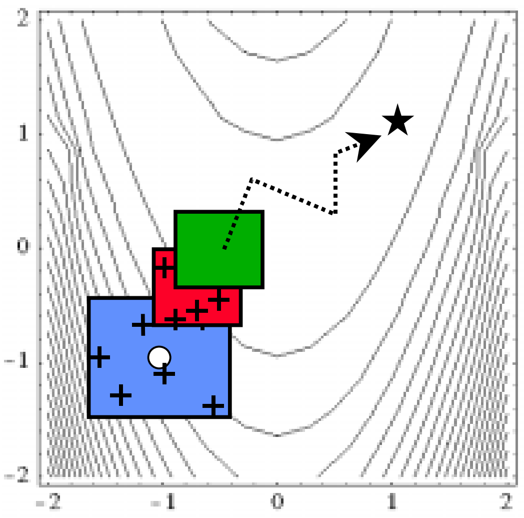

Figure 3. An illustration of the surrogate-based optimization algorithm used to minimize the Rosenbrock function.

Figure 3 depicts a typical application of the SBO algorithm to an optimization problem, where the contours correspond to those of the Rosenbrock function.[19] The squares denote the trust region of each optimization problem that is solved during SBO. The cross marks denote data samples (i.e., function evaluations) taken inside each trust region. The data samples are used to generate local surrogate models, and then the optimization algorithm operates on the surrogate models in lieu of the original functions.

6. Engineering Design Application

This engineering design case study demonstrates the use of the statistical sampling methods, surface fitting methods, and surrogate-based optimization capabilities of DAKOTA.

6.1 Simulation Code and Model

The computational simulation software employed in this study is typical of the methods used to solve systems of partial differential equations such as those encountered in computational structural mechanics, computational fluid dynamics, and computational electromagnetics. The simulation software employs a domain decomposition approach so that the computational mesh or grid can be partitioned to allow parallel computing. For this report, the acronym “CM” will be used to designate both the computational simulation software and the associated computational model.

6.2 Computational Environment

The Computational Plant (Cplant™) at Sandia National Laboratories is a massively parallel computing resource constructed from commodity-based computer parts and running the LINUX operating system.[20] Currently, the Cplant™ cluster is comprised of over 1000 compute nodes, each of which is a 500 MHz Compaq Alpha 21164 processor with 2 MB of level-3 cache memory and 192 MB of main memory.

The Cplant™ computer system is comprised of a service partition and a compute partition. Users log on to one of the nodes in the service partition to compile software and to submit jobs that execute on the nodes in the compute partition. For this study, the DAKOTA software was configured to execute on the service partition nodes with the capability to run multiple concurrent CM jobs in the compute partition.

Figure 4 shows the results of a parallel performance study running the CM code on Cplant™, where the number of processors per CM job was varied from one to 40, and the size of the computational mesh was held constant. Thus, as the number of processors increased, inter-processor communication began to dominate the total wall-clock time. Based on the trend shown in Figure 4, it was decided to use 16 processors per CM job as a compromise between fast execution and parallel efficiency.

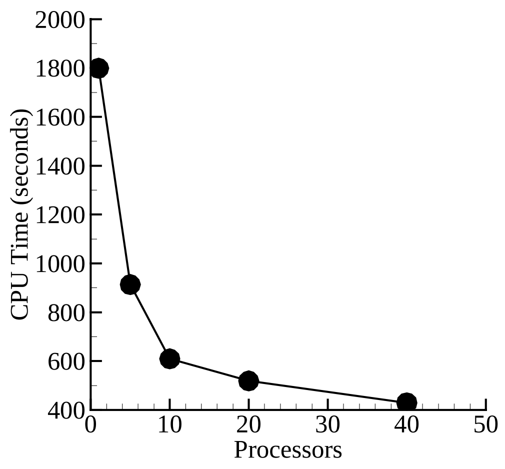

Figure 4. Parallel execution time of the computational simulation code "CM" for a varying number of processors on Cplant™.

6.3 Design Problem Formulation

Several optimization problems were solved during this study, with the number of design parameters ranging from two to seven. For ease of discussion, a two-parameter, unconstrained optimization problem will be presented here. The formulation of this optimization problem is

$$\begin{aligned} &\text{maximize:} \quad f(\boldsymbol{x}) = f(\, f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\,) \\ &\text{subject to:} \quad \boldsymbol{x_L} \le \boldsymbol{x} \le \boldsymbol{x_U} \end{aligned} \tag{4}$$

where $\boldsymbol{x} = (x_1, x_2)$. This two-parameter problem formulation permits the viewing of contours of the objective function and will be used to show the optimization results of the SBO algorithm.

6.4 Parameter Studies

Prior to performing optimization on the CM, single-variable parameter studies were performed to assess the level of nonsmooth trends in objective function due to perturbations in the design parameters. Figure 5 shows the results of one of these parameter studies for variable x_1. In this case, two response quantities, f_1 and f_2, were computed. Both response functions show nonsmooth trends over the variations in x_1. Similar nonsmooth trends were observed for x_2. These results indicate that there are likely to be multiple local minima in the parameter space.

Fine-scale parameter studies were performed to examine the nonsmooth trends at the scale of finite-difference step sizes. The results from one of these studies are shown in Figure 6. Because of both the large-scale and small-scale nonsmoothness in the response functions, this optimization problem is well suited for surrogate-based optimization.

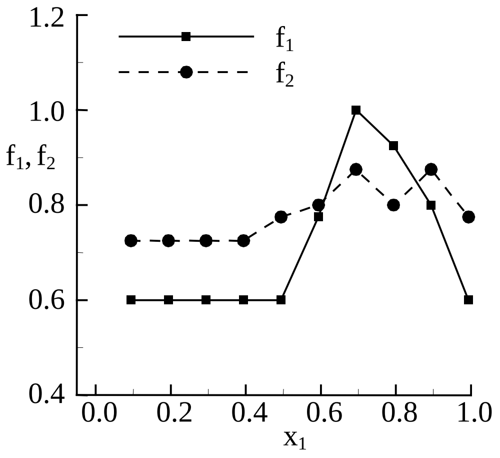

Figure 5. Nonsmooth output of the computational simulation code due to variations in variable x_1.

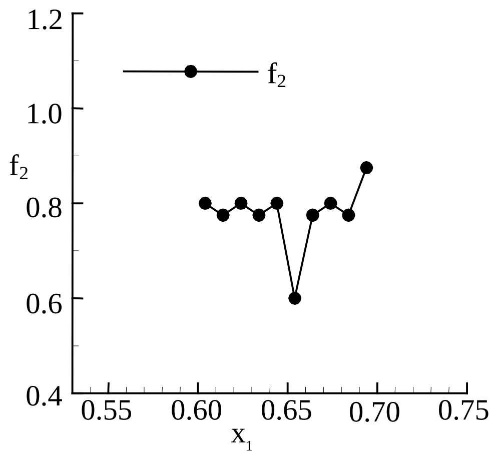

Figure 6. A fine-scale parameter study in x_1 showing numerical noise at the scale of the finite difference step size.

6.5 DAKOTA and CM Code Setup

For this study, DAKOTA and the CM code were coupled using the "black-box" approach shown in Figure 1, with a collection of pre- and post-processing software and UNIX scripts to pass data between the two codes. The pre-processing step consisted of transferring the values of x_1 and x_2 from DAKOTA's parameters file to the input file for the CM code, plus the execution of the grid generation code that creates the computational model for the CM code. The post-processing step consisted of a Fortran77 code that extracted values of f_1 and f_2 from the output files of the CM code, computed the value of the objective function, and wrote the data to DAKOTA's results file.

All of the pre- and post-processing commands, plus the command to execute the CM code, were bundled into one UNIX script that was executed by DAKOTA using the UNIX `system` command. Thus, the entire loop depicted in Figure 1 was automated.

6.6 Surrogate-Based Optimization

This application of SBO employed the Latin hypercube sampling method provided in the DDACE package. Six Latin hypercube samples were generated inside each trust region created by the SBO algorithm. A different random number seed was used for generating each set of samples. This ensured that the samples would have different spatial locations in each trust region.

A quadratic polynomial was used as the surrogate model type during the SBO iterations. For a quadratic polynomial with two variables, there are six unknown polynomial coefficients. Thus, the six Latin hypercube samples are the minimum needed to fit the surrogate model.

DAKOTA has the capability to reuse some or all of the previously generated samples when constructing the surrogate models during iterations of the SBO algorithm. Past experience has shown that this can lead to premature convergence of the SBO iterations, and it can cause ill-conditioning in the linear algebra used to build some of the surrogate model types in DAKOTA. In general, the option to reuse some or all of the samples is exercised only when the cost of generating the samples is expensive and when only a few iterations of the SBO algorithm will be attempted.

In SBO, the user controls the location and size of the first trust region, and then relies on the SBO algorithm to move and resize the trust region during the optimization process. On test problems that are smooth, differentiable, and have a unique minimum, the size of the initial trust region has little impact on the optimization result. In contrast, for real-world optimization problems that are nonsmooth and have multiple local minima, the choice of the trust region greatly affects the final outcome; too small of a starting trust region and SBO converges to the nearest local minimum. Based on previous experience with SBO,[15] and on the nonsmooth response trends shown in the parameter studies of Figure 5 and Figure 6, the initial trust region size was selected to be 100% of the size of the global bounds on x_1 and x_2.

The user also controls the scaling factors that are applied to resize the trust region during the SBO iterations. After a successful SBO iteration, the trust region bounds can either be retained or expanded, depending on the results of logic in the SBO algorithm. An unsuccessful SBO iteration results in a shrinking of the trust region bounds. For this study, the expansion and contraction factors on the trust region size were 2.0 and 0.25, respectively. Note that the trust region bounds are not allowed to grow beyond the original global bounds on the design parameters.

Each iteration of the SBO algorithm requires the solution to an optimization problem inside the current trust region. The unconstrained quasi-Newton BFGS update algorithm in the DOT commercial optimization software package[21] was used in this study, where DOT is one of the numerous optimization packages that have been coupled into DAKOTA.

Table 1. The initial design point and three optimal design points found using surrogate-based optimization.

	Initial	**Opt. 1**	**Opt. 2**	**Opt. 3**
x_1	0.61	0.54	0.58	0.76
x_2	0.40	0.74	0.75	0.75
$f(\mathbf{x})$	0.80	1.00	1.00	1.00

6.7 Optimization Results

Due to the random nature of the Latin hypercube sampling method coupled with the known existence of multiple local maxima in this design problem, each execution of the SBO algorithm was expected to generate a different optimal design. For this reason, three runs of the SBO algorithm were conducted on this design problem, all of which were started from the same initial design point. The results of these three SBO runs are shown in Table 1. All three optima have an objective function of 1.0, but are in different locations in the parameter space.

Optimization trial #1 required six iterations of the SBO algorithm to converge. The iteration history of trial #1 is listed in Table 2 and provides the size of the trust region bounds, along with the SBO algorithm's decision to accept or reject the candidate optimum. The SBO algorithm terminated after six iterations due to lack of progress in maximizing the objective function. In this case, two design points were found with identical objective function values.

Trial #1 required a total of 48 runs of the CM code, four of which were duplicate design points. DAKOTA detected the duplicate points and did not re-run the corresponding CM cases. Instead, data were reused from DAKOTA's internal database. Six CM code runs were used at each step of the SBO algorithm to provide data to build the surrogate model of the objective function. The remaining CM code runs were needed by the SBO algorithm to test the candidate optimum design points generated after each SBO iteration. Optimization trials #2 and #3 required approximately the same number of SBO iterations and CM code runs as did trial #1.

The optimal design points in optimization trials #1 and #2 are close to each other, and trial #3 has a similar value for x_2, but a different value for x_1. This prompted a post-optimality parameter study to further investigate the characteristics of the objective function.

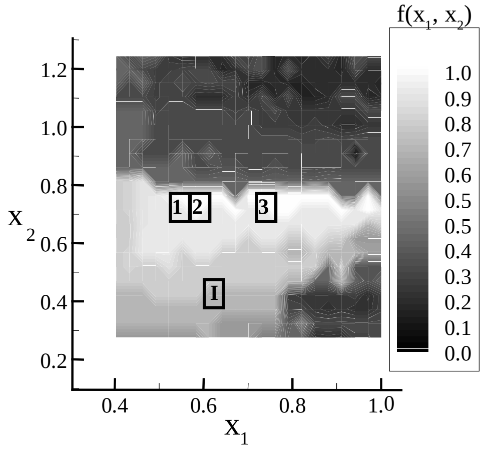

Figure 7. A contour plot of the objective function over a portion of the parameter space in x_1 and x_2. The location of the initial design point "I" and the three optima are shown.

6.8 Post-optimization Parameter Study

Following the optimization runs, a parameter study in x_1 and x_2 was performed. This study used a 21 x 21 grid in a portion of the parameter space to generate a detailed contour map of the objective function. This parameter study was only practical due to the relatively low computational expense of running the CM code, plus the low dimension of the parameter space. A two-dimensional view of the objective function contours is shown in Figure 7. The initial design point for all three SBO runs is indicated by the point labeled "I". The optimal design points from the three SBO runs are also shown in the figure. Note the large change in the objective function that occurs near the line x_2=0.8. The three optima lie on this steeply-sided ridge. Thus, while all three optimization trials produced local maxima, these optimal design points are not robust with respect to variations in x_2.

The contours of the objective function illustrate the nonsmooth features of this optimization problem that has multiple local maxima and slope discontinunities. Figure 8 provides a three-dimensional view of the objective function contours, and clearly shows the plateau regions in the parameter space and the local maxima that create difficulties when applying traditional gradient-based optimization to this problem. In fact, when gradient-based optimization was attempted on this problem, no progress was made from the initial design point. This occurred because the initial point is located on one of the plateau regions where the gradient of the objective function with respect to both design parameters is zero.

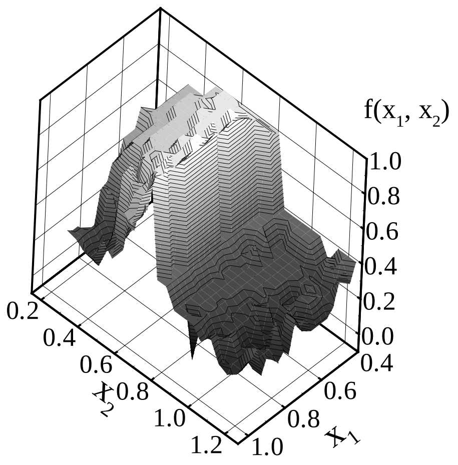

Figure 8. A three-dimensional view of the objective function contours. Note the plateau regions on the upper left side of the plot.

6.9 Computational Expense and Parallel Computing

When running the optimization trials on Cplant™, each CM job was allocated 16 processors (fine-grained parallelism via domain decomposition). A maximum number of six concurrent CM jobs could have been executed during the SBO iterations, where the six jobs correspond to the six Latin hypercube samples that were needed to build the surrogate model of the objective function. At best, the optimization trials could have used 96 processors (six CM jobs, each running on 16 processors). The best possible wall-clock timing for optimization trial #1 would have been approximately 2.3 hours. In contrast, complete serial execution of trial #1 (serial CM, no concurrent CM job executions) would have required approximately 22 wall-clock hours.

During this study it was not always possible to gain access to 96 processors on Cplant™ due to the heavy use of the computer. Thus, optimization trials #1-3 were run with a varying number of processors per CM job and as few as three concurrent CM jobs. A typical optimization trial used 48 processors (three concurrent 16-processors CM jobs) and required about 3.3 wall-clock hours. A similar pragmatic scheduling approach was used to reduce the wall-clock time of the 441 CM jobs needed for the parameter study.

7. Summary

This study demonstrates the use of Latin hypercube data sampling methods, response surface approximation methods, and numerical optimization capabilities to solve an engineering design problem. A surrogate-based optimization algorithm has been developed which combines the above-mentioned methods, and has been implemented in the DAKOTA toolkit. The engineering design problem presented here illustrates the effectiveness of the surrogate-based optimization algorithm on design problems that have nonsmooth trends and multiple local optima; features that are common in engineering practice, but which pose difficulties for traditional gradient-based optimization methods.

Parallel computing was an integral aspect of this study, which exploited fine-grained parallelization in the computational simulation code and coarse-grained parallelization in the data sampling portion of the SBO algorithm. The use of parallel computing reduced the completion time of an optimization study from approximately 22 hours to less than four hours.

Table 2. Iteration history of surrogate-based optimization trial #1.

SBO Iter. #	**x_1***	**x_2***	***f(x*)***	**x_1 Trust Region Bounds**	**x_2 Trust Region Bounds**	**Accept *x** *(Yes/No)***
0	0.61	0.40	0.8			
1	0.54	0.73	1.0	[0,1]	[0,1]	Yes
2	0.00	1.00	0.3	[0,1]	[0,1]	No
3	0.80	0.70	0.9	[0.30, 0.80]	[0.48, 0.98]	No
4	0.60	0.80	0.4	[0.48, 0.60]	[0.67, 0.80]	No
5	0.56	0.74	0.9	[0.53, 0.56]	[0.72, 0.75]	No
6	0.54	0.74	1.0	[0.54, 0.55]	[0.73, 0.74]	Yes

References

[1] Wang, L., Grandhi, R. V., and Canfield, R. A., "Multivariate Hermite Approximation for Design Optimization," *International Journal for Numerical Methods in Engineering*, Vol. 39, 1996, pp. 787-803.
[2] Torczon, V. and Trosset, M. W., "Using Approximations to Accelerate Engineering Design Optimization," in *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, AIAA Paper 98-4800, Sept. 2-4, 1998.
[3] Madsen, J. I., Shyy, W., and Haftka, R. T., "Response Surface Techniques for Diffuser Shape Optimization," *AIAA Journal*, Vol. 38, No. 9, 2000, pp. 1512-1518.
[4] Eldred, M. S., Outka, D. E., Bohnoff, W. J., Witkowski, W. R., Romero, V. J., Ponslet, E. R., and Chen, K. S., "Optimization of Complex Mechanics Simulations with Object-Oriented Design Software," *Computer Modeling and Simulation in Engineering*, Vol. 1, No. 3, 1996, pp. 323-352.
[5] Eldred, M. S., Hart, W. E., Schimel, B. D., and van Bloemen Waanders, B. G., "Multilevel Parallelism for Optimization on MP Computers: Theory and Experiment," in *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, AIAA Paper 2000-4818, Sept. 6-8, 2000.
[6] Eldred, M. S., Giunta, A. A., van Bloemen Waanders, B. G., Wojtkiewicz, S. F., Jr., Hart, W. E. and Alleva, M. P., *DAKOTA Users Manual: Version 3.0*, Technical Report, Sandia National Laboratories, Albuquerque, NM, 2001 (online version available: http://endo.sandia.gov/DAKOTA).
[7] Eldred, M. S., Giunta, A. A., van Bloemen Waanders, B. G., Wojtkiewicz, S. F., Jr., Hart, W. E. and Alleva, M. P., *DAKOTA Reference Manual: Version 3.0*, Technical Report, Sandia National Laboratories, Albuquerque, NM, 2001 (online version available: http://endo.sandia.gov/DAKOTA).
[8] Myers, R. H., and Montgomery, D. C., *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley and Sons, Inc., New York, NY, 1995, pp. 1-141, 279-401, 462-480.
[9] McKay, M. D., Beckman, R. J., and Conover, W. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, 1979, pp. 239-245.
[10] Koehler, J. R., and Owen, A. B., "Computer Experiments," Vol. 13 *of Handbook of Statistics*, Elsevier-Science, New York, NY, eds. S. Ghost and C. R. Rao, 1996, pp. 239-245.
[11] Iman, R. L. and Shortencarier, M. J., ''A Fortran 77 Program and User's Guide for the Generation of Latin Hypercube Samples for Use with Computer Models,'' NUREG/CR-3624, Technical Report SAND83-2365, Sandia National Laboratories, Albuquerque, NM, 1984.
[12] Tong, C. H., and Meza, J. C., ''DDACE: A Distributed Object-Oriented Software with Multiple Samplings for the Design and Analysis of Computer Experiments,'' Technical Report (in preparation), Sandia National Laboratories, Livermore, CA (see http://csmr.ca.sandia.gov).
[13] Giunta, A. A. and Watson, L. T., "A Comparison of Approximation Modeling Techniques: Polynomial Versus Interpolating Models," in *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, AIAA Paper 98-4758, Sept. 1998, pp. 392-404.
[14] Cressie, N., *Statistics for Spatial Data*, John Wiley and Sons, Inc., New York, NY, 1991, pp. 1-26.

[15] Giunta, A. A., and Eldred, M. S., "Implementation of a Trust Region Model Management Strategy in the DAKOTA Optimization Toolkit," in *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, AIAA Paper 2000-4935, Sept. 6-8, 2000.
[16] Alexandrov, N. M., Dennis, Jr., J. E., Lewis, R. M., and Torczon, V., "A Trust-region Framework for Managing the Use of Approximation Models in Optimization," *Structural Optimization*, Vol. 15, 1998, pp. 16-23.
[17] Alexandrov, N. M., Lewis, R. M., Gumbert, C. R., Green, L. L., and Newman, P. A., "Optimization with Variable-Fidelity Models Applied to Wing Design," AIAA Paper 2000-0841, 38th Aerospace Sciences Meeting and Exhibit, Reno, NV, 2000.
[18] Rodriguez, J. F., Renaud, J. E., and Watson, L. T., "Convergence of Trust Region Augmented Lagrangian Methods Using Variable Fidelity Approximation Data," *Structural Optimization*, Vol. 15, pp. 1-7, 1998.
[19] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, New York, NY, 1981, pp. 95-96.
[20] Brightwell, R., Fisk, L. A., Greenberg, D. S., Hudson, T., Levenhagen, M., Maccabe, A. B., and Riesen, R., "Massively Parallel Computing using Commodity Components," *Parallel Computing*, Vol. 26, No. 2-3, 2000, pp. 243-266.
[21] Vanderplaats Research & Development, Inc., *DOT User's Manual*, Version 4.20, Colorado Springs, CO, 1995.