A Surrogate-Model-Based Method For Constrained Optimization

Article · September 2000

DOI: 10.2514/6.2000-4891

5 authors, including:

Charles Audet
Polytechnique Montréal
177 PUBLICATIONS **8,058** CITATIONS

SEE PROFILE

J. E. Dennis
Rice University
173 PUBLICATIONS **30,082** CITATIONS

SEE PROFILE

AIAA-2000-4891

A SURROGATE-MODEL-BASED METHOD FOR CONSTRAINED OPTIMIZATION

Charles Audet, J. E. Dennis, Jr., Douglas W. Moore

Department of Computational & Applied Mathematics,

Rice University, 6100 Main St., Houston, TX 77005.

Andrew Booker, Paul D. Frank

Mathematics & Engineering Analysis,

Boeing Phantom Works,

Mathematics and Computing Technology,

Box 3707, M/S 7L-68, Seattle, WA 98124.

Abstract

This paper describes an algorithm and provides test results for surrogate-model-based optimization. In this type of optimization, the objective and constraint functions are represented by global "surrogates", i.e. response models, of the "true" problem responses. In general, guarantees of global optimality are not possible. However, a robust surrogate-model-based optimization method is presented here that has good global search properties, and proven local convergence results. This paper describes methods for handling three key issues in surrogate-model-based optimization. These issues are maintaining a balance of effort between global design space exploration and local optimizer region refinement, maintaining good surrogate model conditioning as points "pile up" in local regions, and providing a provably convergent method for ensuring local optimality.

Acknowledgments: Work of the first author was supported by NSERC (Natural Sciences and Engineering Research Council) fellowship PDF-207432-1998, and the first three authors was supported by DOE DE-FG03-95ER25257, AFOSR F49620-98-1-0267, The Boeing Company, Sandia LG-4253, ExxonMobil and CRPC CCR-9120008.

1 Introduction

This paper describes an optimization method for the case where the objective and constraint functions are represented by inexpensive-to-evaluate "surrogate" models of a "true" simulation code. The use of response models is particularly attractive when the true responses come from compute-intensive computer simulations. Examples include 3D fluid dynamics analysis and structural analysis via the finite element method. The use of inexpensive-to-evaluate surrogate models has obvious potential for speeding the optimization process. The surrogate models also have the benefits of allowing optimization of problems with nonsmooth or noisy responses, and providing insight into the nature of the design space [4][5][10].

Because response models are inexpensive-to-evaluate surrogates for expensive analysis codes, they are often used to represent disciplines in multidisciplinary design optimization (MDO) problems (see e.g. [7][12][2][26]) and competing objectives in multiobjective optimization.

There are many types of surrogate models to choose from. Among these are neural nets, well-conditioned low-degree polynomials [11], and Kriging models [9][8][23][24]. As will be discussed later, the method used here for selecting points for model refinement requires estimates of uncertainty in model value

predictions at arbitrary points in design space. These estimates can readily be obtained for Kriging models. This is one reason that Kriging models are used in the implementation described in this paper. However, it should be noted that except for the details of model updating and selection of points for model refinement, the overall optimization method is independent of the type of surrogate model used.

A mathematical statement of the optimization problem is:

$$\begin{aligned} \min_{x \in \Re^n} \quad & f(x) \\ \text{s.t.} \quad & C(x) \leq 0 \\ & x_{\ell b} \leq x \leq x_{ub} \end{aligned} \tag{1}$$

where $f : \Re^n \rightarrow \Re$ and $C : \Re^n \rightarrow \Re^m$ are continuously differentiable functions with $C = (c_1, \ldots, c_m)^T$. The vectors $x_{\ell b}$ and x_{ub} in $\Re^n$ are the lower and upper bounds on the variables. The inequalities in (1) are completely general because a bound other than zero can be handled by redefining the appropriate component function $c_i(x)$. Lower-bounded inequalities can be handled by negating the corresponding $c_i(x)$. Equality constraints can be specified as a pair consisting of an upper-bounded and lower-bounded inequality with the same constraint functions and bounds. However, since this paper deals with modeling of expensive simulations, it is assumed that there will be a fairly loose feasibility tolerance for equality constraints. Thus, for ease of exposition and for discussion of convergence, equalities will be considered to be inequality constraints with upper and lower bounds that happen to be "somewhat" close to each other.

For ease of exposition, it is assumed in (1) that $f(x)$ and the constraint functions $c_i(x)$ are responses from a "true" simulation code, each of which is represented by a single surrogate model. In actuality, the optimization method presented here is valid if $f(x)$ or component functions of $C(x)$ are represented by arbitrary functions of surrogate model values. For example, the surrogate objective can be a weighted sum of several response models. In addition, the method can handle the case where $f(x)$ or some components of $C(x)$ are "truth" values, i.e. are not modeled. For example, this is the case when some constraints are simple analytic functions of the variables, such as constraints on geometry.

Figure 1 is a high-level view of the optimization algorithm. The method shown in Figure 1 is a particular example of the very general filter-based, derivative-free optimization method described in Section 2. Much of this optimization method is encompassed in the Figure 1 box labeled "CONTROLLER".

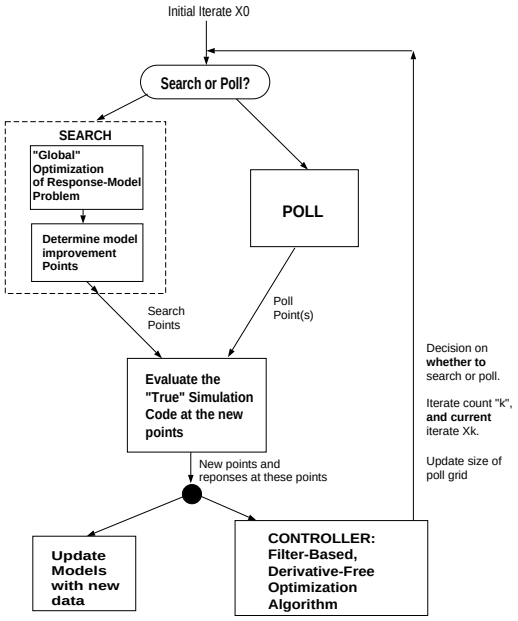

Figure 1: The optimization algorithm

The basic optimization method is an iterative process. At each iteration, the method identifies points at which more "true" simulation data is needed. The data are obtained and surrogate models are updated using the new data. The optimization controller uses the new data to decide how the optimization process will proceed. One such decision is whether to be in the local, lattice-based, POLL mode or the more global SEARCH mode. The modes are discussed next.

The SEARCH mode selects new points for running the simulation code that are either "global optimizers" for the surrogate model problem or are points for *model improvement*. The search for "global optimizers" may consist of simply running a local optimization code from several (say 10) randomly selected start points in the design space. The model-improvement points are not global optimizers of the response-model problem, but lie in regions of possibly large model error, where a global optimum is not unlikely. Thus, it is worth obtaining new "true" data at these points. Since these points tend to be far from existing data, they represent the most global aspect of the optimization method. The constrained, balanced, local-global search (CBLGS) method presented in Section 4 describes selection of these model-improvement points.

The POLL mode is a local, lattice-based, search. The simplest poll pattern is the collection of unit steps in the $2n$ positive and negative coordinate directions on

an n–dimensional grid. However, it has been shown [20] that polling can be done along any set of vectors that form a *positive spanning set* for n–dimensional space. A *positive spanning set* is any set of vectors that spans $n-$dimensional space when summed with nonnegative coefficients.

A key item in Figure 1 is the "Update Models with new data" process. A common problem in surrogate-model-based optimization is the degradation in conditioning as the models are updated to incorporate new data. This is because the optimization process tends to "cluster" points in design space. Section 3 discusses Kriging models and presents a new method for maintaining well-conditioned models during the optimization process.

A summary of the content of this paper is as follows. The "filter-based" optimization method, and convergence results for this method, are given in Section 2. Section 3 provides a description of Kriging models and presents a new method for maintaining model conditioning. The CBLGS algorithm for selecting *model-improvement* points is given in Section 4. Testing of the optimization method is summarized in Section 5. Conclusions and future work are discussed in Section 6.

2 Filter-Based Derivative-Free Optimization Algorithms

2.1 Introduction

Torczon [27] presents a flexible class of generalized pattern search (GPS) algorithms for derivative-free unconstrained optimization. Here, our method (thoroughly presented in [1]) extends the GPS algorithm to general nonlinear programming problems by combining it with filter algorithms (described below) in a way that reduces transparently to GPS for unconstrained problems. We do this without penalty constants or Lagrange multiplier estimates (as required in [21]) by formulating a step acceptance rule based on filter methods.

Filter algorithms were introduced by Fletcher and Leyffer [13] as a way to globalize SQP and SLP without using a merit function that would require a troublesome penalty or barrier parameter to be estimated. A step is accepted if it either reduces the objective function or the constraint violation. Fletcher, Leyffer and Toint [14] show convergence of the method that uses sequential linear programming to suggest steps. Fletcher, Gould, Leyffer and Toint [15] show convergence of the method that uses sequential quadratic programming to suggest steps. Previous filter algorithms thus require explicit use of the derivatives of both the objective and the constraints. Our method does not.

We present a pattern search version of the filter approach that converges using any user-defined finite SEARCH procedure, here based on surrogates, to suggest steps, and which falls back on a special POLL step when the user defined procedure stalls, i.e., when the SEARCH is unsuccessful. Elsewhere [1] we show that our filter-pattern search class of algorithms guarantees subsequence convergence to points that satisfy appropriate optimality conditions depending on the local smoothness of the problem functions. Convergence is assured without assuming even finite values or continuity.

The nonnegative constraint violation function satisfies $h(x) = 0$ if and only if x is feasible (and thus $h(x) > 0$ if and only if x is infeasible). We assume that h satisfies a weak monotonicity property that if $C(x)_+$ is the vector whose j^{th} component is $\max(0, c_j(x))$, then the component-wise inequality $C(x)_+ \leq C(y)_+$ implies $h(x) \leq h(y)$. For example, we could set $h(x) = \|C(x)_+\|$ where $\|\cdot\|$ is a vector norm. Our best theoretical results are for $h(x) = \|C(x)_+\|_2^2$.

A filter $\mathcal{F}$ is a finite set containing the feasible point with best known objective, and infeasible points in $\Re^n$ such for no pair of infeasible points (x, x') in the filter does one point have both a better objective and constraint violation function value than the other. An infeasible point x' is said to be filtered if either $h_{max} \leq h(x')$ for some user-supplied upper bound h_{max}, or if for some infeasible x ($x' \neq x$) belonging to the filter, $h(x') \geq h(x)$ and $f(x') \geq f(x)$. A feasible point is filtered if its objective function value is greater than or equal to the feasible incumbent value f^F (i.e., the least function value found so far at a feasible solution). It is unfiltered otherwise. Of course, the first feasible point encountered (if any) is automatically the initial feasible incumbent solution.

The set of filtered points $\overline{\mathcal{F}}$ is denoted in standard notation as:

$$\begin{aligned}\overline{\mathcal{F}} \quad = \quad & \bigcup_{x \in \mathcal{F}} \left\{ x' \in \Re^n \setminus \mathcal{F} : \begin{array}{l} h(x') \geq h(x), \\ f(x') \geq f(x) \end{array} \right\} \\ & \bigcup \quad \{x' \in \Re^n : h(x') \geq h_{max}\} \\ & \bigcup \quad \{x' \in \Re^n : h(x') = 0, f(x') > f^F\}.\end{aligned}$$

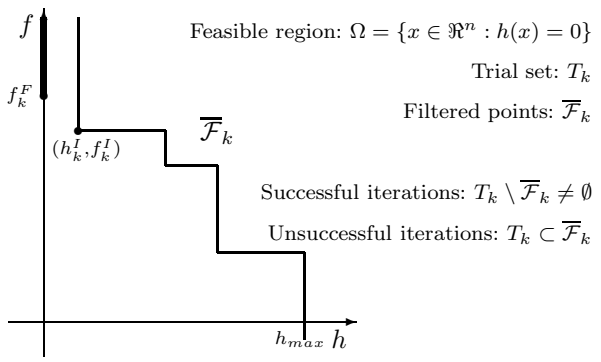

Figure 2: The three possible types of iterations.

2.2 Algorithm

Unsuccessful iterations are those where all trial points are filtered. *Successful* iterations are those where an unfiltered trial point is found. The SEARCH or POLL step may be terminated without any more function or constraint evaluations if such a point is found. The mesh size parameter is either increased or kept constant in successful iterations, and it is decreased in unsuccessful ones.

We define two types of incumbent solutions: the feasible ones, and the most nearly feasible ones (see Figure 2). Let f_k^F represent the feasible incumbent value, i.e., the smallest objective function value (for feasible points) found by the algorithm up to iteration k. Let $h_k^I > 0$ be the least positive constraint violation value found up to iteration k, and let f_k^I denote the smallest objective function value of the points found whose constraint violation function value are equal to h_k^I. The superscript F stands for *feasible* and I for *infeasible*.

The poll set for any iteration is the set of mesh neighbors of p_k. The poll center p_k is either chosen in the set of feasible incumbent solutions, or it must be one of the most nearly feasible incumbent solutions. Generally, there will be a single value in each of the sets of incumbents.

The poll center p_k either satisfies $(h(p_k), f(p_k)) = (0, f_k^F)$, or, $(h(p_k), f(p_k)) = (h_k^I, f_k^I)$. Our class of algorithms and their analyses are completely flexible about the choice between these two alternative poll centers p_k at a given iteration. It is up to the user to select the strategy defining the poll center. For example, it might be a good strategy to alternate choosing the poll centers in the set of feasible and infeasible incumbents until an unsuccessful iteration occurs.

Even if we already have a feasible incumbent solution, we may wish to poll around one of the nearly feasible points, which might have a lower objective function value and thus lead us to explore a different part of the feasible region Ω. This option allows the algorithm to avoid stalling (when infinitely many consecutive unsuccessful iterations are generated) in the Lewis and Torczon [19] example.

The most useful successful iterations are those that produce a feasible iterate x_{k+1} which improves the feasible incumbent value to $f_{k+1}^F = f(x_{k+1})$. Next, are the successful iterations that do not produce a feasible iterate, but improve the most nearly feasible incumbent solution: $h_{k+1}^I = h(x_{k+1})$ and $f_{k+1}^I = f(x_{k+1})$. Finally, there are the other successful iterations. They leave the incumbents unchanged $f_{k+1}^F = f_k^F, h_{k+1}^I = h_k^I$ and $f_{k+1}^I = f_k^I$, but add some elements to the filter.

The unsuccessful iterations are such that all points in the trial set are filtered. Regardless of the feasibility of x_k, if the iteration is unsuccessful, the next iterate x_{k+1} is set to x_k, and the mesh size parameter is decreased: $\Delta_{k+1} < \Delta_k$.

Our algorithm for constrained optimization is stated formally below. We allow for the fact that in some applications, a set of initial solutions may be used to seed the filter. Without any loss of generality we assume that any such points are on the initial mesh and that they have been "filtered" to be consistent with our initialization step in the sense that x_0 will not be filtered by the other seed points. An easy way to assure this would be to take x_0 to be the most feasible seed point, breaking ties by taking one with the smallest objective function value.

A standard trick in engineering design, when seed designs are available, is to go further and make linear combinations of the seed points be the new design space. This certainly makes sense, can lead to a big reduction in the dimension of the design space, and makes the initial mesh be defined by the seed points in a simple way.

It is implicit in all we have said that our initial solution must have finite values for the objective and constraint violations, and this will be true of all the iterates generated by the algorithm. In [1], we show how to incorporate a finite number of linear constraints by setting the objective value to infinity points that violate them while using the filter for more general constraints.

Algorithm

- INITIALIZATION:
  Let x_0 be one of a set of initial points in the filter

$\mathcal{F}_0$. Fix $h_{max} > h(x_0)$ and $\Delta_0 > 0$ and set the iteration counter k to 0.

- DEFINITION OF INCUMBENT SOLUTIONS:
  Define (if possible)
  f_k^F: the smallest objective function value for all feasible solutions found so far;
  $h_k^I > 0$: the least positive constraint violation function value found so far;
  f_k^I: the smallest objective function value of the points found so far whose constraint violation function value are equal to h_k^I.

- SEARCH AND POLL STEPS:
  Perform the SEARCH and possibly the POLL steps (or only part of the steps) until an unfiltered trial point x_{k+1} is found, or when it is shown that all trial points are filtered.

  - SEARCH STEP:
    Evaluate the functions h and f on a set of trial points on the current mesh M_k (the strategy that gives the set of points is usually provided by the user).
  - POLL STEP:
    Evaluate the functions h and f on the poll set around p_k, where p_k satisfies either $(h(p_k), f(p_k)) = (0, f_k^F)$ or $(h(p_k), f(p_k)) = (h_k^I, f_k^I)$.

- PARAMETER UPDATE:
  If the SEARCH or the POLL step produced an unfiltered iterate $x_{k+1} \in \mathcal{F}_{k+1}$, then declare the iteration *successful* and update $\Delta_{k+1} \geq \Delta_k$.
  Otherwise, set $x_{k+1} = x_k$, declare the iteration *unsuccessful* and update $\Delta_{k+1} < \Delta_k$.
  Increase $k \leftarrow k + 1$ and go back the definition of the incumbents. ∎

2.3 Convergence behavior

We give the flavor of our results here; details of our rather technical analysis can be found in [1]. We say that a subsequence of the sequence of convergent unsuccessful poll centers is a refining sequence, if the corresponding subsequence of mesh size parameters $\{\Delta_k\}$ goes to zero.

Theorem 2.1 *Let $\hat{x}$ be an accumulation point of a refining sequence. If h is strictly differentiable at $\hat{x}$, then $\nabla h(\hat{x}) = 0$.*

This means that using $h(x) = \|C(x)_+\|_2^2$ gives nice results, but the scaling of the constraint violations is then more sensitive. Thus, the question arises as to the differentiability of the more common choice $h_1(x) = \|C(x)_+\|_1$. The answer is that this choice is rarely strictly differentiable at $\hat{x}$, and a proof requires a very technical lemma.

We now summarize some results for the objective function at a limit point.

Theorem 2.2 *Let $\hat{x}$ be an accumulation point of any refining subsequence. If $\hat{x}$ is strictly feasible and if f is strictly differentiable at $\hat{x}$, then $\nabla f(\hat{x}) = 0$.*

This says that if we converge to a point in the interior of the feasible region, then the KKT first order necessary conditions hold. Wherever it is located, $\hat{x}$ is a KKT point for a related problem. We spare the reader the technical result (rigorously shown in [1]) that for general constraints, we can construct a cone that contains $N_\Omega(\hat{x})$ as well as $-\nabla f(\hat{x})$. In the case that the constraints are sufficiently simple (for example, a finite number of linear constraints), then one can arrange the polling directions so that this related problem is the actual problem. Of course, in theory it is always possible to include the right polling directions so that this happens.

By using a filter-based step acceptance criterion, we have overcome a difficulty in applying pattern search algorithms to constrained optimization; specifically, the problem that the objective function descent directions in the positive spanning directions may be infeasible. Lewis and Torczon [19] give an example where a nonfilter-based version of the pattern search algorithm stalls (all subsequent iterations are unsuccessful) when the positive spanning directions are poorly chosen at a feasible point where the gradient of f is nonzero.

The following result shows that our algorithm will not stall at any point x for which the $\nabla f(x) \neq 0$, regardless of the choice of positive spanning set. The implication is that there eventually will be a successful iteration that will move the iterate away from the current point. Thus, we can hope for a more global solution even if we begin at a constrained local solution.

Proposition 2.3 *Suppose that f is strictly differentiable at the poll center p_k, where k is the iteration number. If $\nabla f(p_k) \neq 0$, then there exists an index $\ell \geq k$ such that $x_{\ell+1} \neq x_k$.*

The remainder of the paper can be viewed as providing sophisticated tools and procedures for the SEARCH phase of the algorithm. Our earlier work [3] showed that kriging surrogates can be extraordinarily effective SEARCH tools.

3 Well-Conditioned Kriging Models

This section describes the interpolating models used as approximate models and an approach to well–conditioned model updating. See, *e.g.*, [24] for a more detailed description of Kriging models in computer output. The model update approach and tests on standard (unconstrained) optimization test problems are described in [6].

Kriging models are models of the output Y of a deterministic simulation code as a function of d–dimensional inputs x via a stationary Gaussian process Z plus a constant

$$Y(x) = \beta + Z(x)$$

where for each x, $Z(x)$ has mean 0, variance σ^2 and correlations between $Y(x)$ and $Y(w)$ are given by the positive definite correlation function

$$Cor(Y(x), Y(w)) \equiv R(x, w).$$

We note that the correlation function usually depends on d "scale" parameters $\theta_1, \ldots, \theta_d$ [24].

When Y is observed at N points (called *sites* in statistical literature)

$$S = (s_1, ..., s_N),$$

denoted by

$$Y_S \equiv (Y(s_1), ..., Y(s_N))^T,$$

then the interpolating approximate model is given by

$$\hat{Y}(x) = E(Y(x) \mid Y_S).$$

which can be shown to be

$$\hat{Y}(x) = \hat{\beta} + r(x, S)^T R^{-1}(Y_S - \vec{1} \cdot \hat{\beta}) \qquad (2)$$

where $(R_{i,j})$ = correlation between observations $Y(s_i)$ and $Y(s_j)$, $r(x, S)$ = vector of correlations between $Y(x)$ and $Y(s_i), i = 1, \ldots, N$ and

$$\hat{\beta} \equiv \frac{\sum_j Y(s_j) \sum_i R^{-1}_{i,j}}{\sum_{i,j} R^{-1}_{i,j}}.$$

In optimization with approximate models, points are continually added to an initial model to update the model. These points tend to "pile up" as we (hopefully) approach an optimum. The result is that R becomes ill-conditioned so that one can neither solve for

$$R^{-1}(Y_S - \vec{1} \cdot \hat{\beta})$$

nor compute $\hat{\beta}$.

We propose the following solution to the problem of ill-conditioning: Model the output as the sum of two stochastically independent Gaussian processes, one with the original correlation parameters estimated from the first set of points, and one with a "finer" correlation structure. Thus

$$Y(x) = \beta + Z_1(x) + Z_2(x).$$

where Z_1 and Z_2 are independent of each other. If, for Z_i, we denote the correlation function by R_i and the variance by σ_i, then the resulting correlation function for Y is

$$R(x, w) = \lambda R_1(x, w) + (1 - \lambda) R_2(x, w)$$

where

$$\lambda = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}.$$

Second Process Parameter Estimation

Updating a Kriging model with a second Gaussian process requires estimation of correlation parameters and of the weight λ. For correlations in the first process Z_1 we use the Gaussian product correlation function ([24]). The initial set of sites and model fit provide the correlation parameters (computed via Maximum Likelihood Estimation [24]) for Z_1. These parameters are retained at each model update. For the second process Z_2 we choose the positive cubic spline correlation structure. The scale correlation parameters for Z_2 are assumed to be all unknown, but equal, *i.e.*, $\theta_j = \theta$ for all j. The fit to the model is then obtained by estimating the parameter θ for Z_2 and the weight λ, via maximum likelihood. The likelihood is then maximized over λ in the range [0, 1] and θ in the range $[0, 1/dist]$, where $dist$ is the minimum over pairs of distinct sites of the maximum difference in coordinates between pairs. For the positive cubic spline correlation, θ greater $1/dist$ implies all sites are uncorrelated with respect to R_2. Note that finding the maximum likelihood over (λ, θ) is a two variable optimization problem. An approximate maximum likelihood is found by evaluating the likelihood on a weighted "λ by θ" grid.

4 Determining Points for Model Improvement

This section introduces the constrained, balanced, local-global search (CBLGS) method for determining points for model improvement. The goal is to locate promising points in design space that are far

from points where data from the "true" analysis code have already been obtained. Points are considered "promising" if the combination of model-predicted values and estimated model errors indicate a "reasonable" chance for feasibility and good objective values. Since the method tends to select points far from existing data points, it truly emphasizes "global" search. This global emphasis exceeds that of finding global optimizers for the model problem, which is based on existing information. The most local of all the processes in Figure 1 is the POLL. As the name implies, the CBLGS algorithm attempts to balance the considerations of local and global search of the design space.

For best utility, CBLGS must run quickly. Thus, it is undesirable to have to solve a difficult optimization problem to find model-improvement points. Instead, CBLGS relies on model evaluations and error bound estimates over a large set (*dense cloud*) of points, numbering say 10,000 to 100,000, that are well-spread-out in design space. In doing this, CBGLS takes advantage of the facts that large sets of well-spread-out points can be obtained quickly, and model evaluations and model error estimates can be computed very cheaply. The dense clouds of points used in CBLGS are orthogonal array-based Latin hypercubes (see e.g., Owen [22]). For "reasonably" shaped feasible regions, the feasible subset of the dense cloud will retain the property of being well-spread out in design space. Before presenting the CBLGS algorithm, it is necessary to define the concepts of *expected improvement* and *expected violation*.

For simplicity, *expected improvement* will be defined in terms of one Gaussian process. Suppose the Gaussian process with variance σ^2 has been evaluated using equation (2) with $\hat{y} = \hat{Y}(x)$, and f_{min} is the best objective function value obtained thus far. The key concepts here come from Schonlau [25], who defines the *expected improvement* $E(I)$ at x as the integral of the distribution $Y(x)$ over all values less than or equal to f_{min}. This yields,

$$E(I) = \begin{cases} (f_{min} - \hat{y})\Phi(\frac{f_{min}-\hat{y}}{\sigma}) + \sigma\phi(\frac{f_{min}-\hat{y}}{\sigma}) \\ \qquad\qquad\qquad\qquad \text{if } \sigma > 0 \\ 0 \qquad\qquad\qquad\qquad \text{if } \sigma = 0 \end{cases} \tag{3}$$

where $\phi()$ and Φ denote the probability density function and the cumulative distribution function of the standard normal distribution. Roughly speaking, the expected improvement is small unless either the model predicted value $\hat{y}$ is less than f_{min}, or the amount by which $\hat{y}$ exceeds f_{min} is not large compared to the variance.

Expected improvement provides a globalization tool by accounting for model error bounds, which are functions of variance. However, this only helps in terms of the objective function. The concept of expected improvement can easily be extended to the constraints by introducing the notion of *expected violation*. For example, the expected violation at x, of a lower bound for $c_i(x)$, can be obtained from (3) by replacing f_{min} by the lower bound value and setting $\hat{y}$ to the model-predicted value for $c_i(x)$. The expected violation of an upper bound can be computed in a similar way, by interchanging the roles of f_{min} and $\hat{y}$ in (3). Having defined *expected improvement* and *expected violation*, the CBLGS algorithm can now be stated.

Constrained, Balanced, Local-Global Search (CBLGS) Algorithm

The parameters provided to CBLGS are: the number of model refinement points requested $ngoal$**, the size of the initial *dense cloud* of candidate points** $ncloud$**, the maximum allowable size for the candidate set** $maxcloud$**, the tolerance on expected feasibility** $featol$**, and the best feasible objective value found thus far (or an estimate for it)** f_{min}**.**

1. Compute an orthogonal array-based Latin hypercube that has more than $ncloud$ points, but does not exceed $ncloud$ by a "large" number.
2. Compute the L-infinity norm of the expected constraint violations at all the candidate points. Let $nfeas$ be the number of candidates with expected violations less than $featol$. If $nfeas < ngoal$ and $ncloud < maxcloud$, increase the size of $ncloud$ (without exceeding $maxcloud$) and go to 1.
3. Evaluate expected improvement at the $nfeas$ points that passed the expected feasibility test. Return as the set of model improvement points the set of $\min(ngoal, nfeas)$ points yielding the highest values for expected improvement.

As indicated above, CBLGS selects the best candidates from a "bucket" of points with small enough expected violations. Some values that have been used for the CBLGS parameters are 5,000 for $ncloud$, 100,000 for $maxcloud$, 5 for $ngoal$, and 0.01 for $featol$ on a well-scaled problem.

A modification of CBLGS can be used to obtain sample points for initial modeling, that are well-spread

out and are feasible with respect to a set of simulation-code-independent constraints. For example, it may be desirable to run an expensive simulation code only at a set of points that is feasible with respect to geometric constraints that can be computed without running the simulation code. In this situation, the main changes required for CBLGS are redefining expected violation to be the computed violation, and eliminating step 3 of the algorithm.

5 Test Results

SEQOPT is an optimization code that incorporates all three major features of this paper. That is, filter-based optimization, well-conditioned Kriging models, and CBLGS. Testing is in the preliminary phase, and the initial test results are reported here.

The first test results presented are for problems number 59 (denoted HS59) and 100 (denoted HS100) from the widely-used Hock and Schittkowski [17] local optimization test set. Since these problems are differentiable and have known answers, they provide a means for validating SEQOPT, and allow us to obtain comparative data using the highly-regarded local optimization code NPSOL [16]. So that the tests reflect conditions where the problem functions come from expensive simulations, loose convergence tolerances were used. That is, a 0.001 solution tolerance was set for relative constraint violation and for the relative optimal objective value. Also, upper and lower bounds were placed on all the variables. For HS59, the vector of lower bounds is (0.0,0.0) and the vector of upper bounds is (65.0,75.0). For HS100, the vector of lower bounds is (-10.0,-5.0,-5.0,-10.0,-3.0,-10.0,-5.0) and the vector of upper bounds is (10.0,5.0,5.0,10.0,3.0,10.0,5.0).

To obtain comparative data, NPSOL was started from 20 randomly-generated points in the design space. Table 1 summarizes the comparative results for SEQOPT and NPSOL. The function value counts for SEQOPT, given in the third column of Table 1, are the sum of those used in obtaining the initial response models and those computed during the SEQOPT run. For HS59, 16 function evaluations were used to obtain the initial models. For HS100, 64 function evaluations were used to obtain the initial models. The function value counts for NPSOL, given in the fourth column of Table 1, are the sum of the function evaluations plus n times the number of gradients required, based on the cost of one-sided finite difference derivatives. The NPSOL results are averaged over those runs for which NPSOL obtained the globally optimal function values, to within 0.001 relative tolerances. The final column of Table 1 indicates the number of runs, out of 20, that NPSOL converged to a local optimizer that is not the global optimizer. (NPSOL identified four different local optimizers for each of the two problems).

The message to be gotten from Table 1 is not that SEQOPT is a better optimizer than NPSOL. Clearly, the two codes are designed for different purposes. Instead, the message is that while SEQOPT is a global search code, it is reasonably efficient in obtaining its solutions. We next discuss results for a nondifferentiable set of problems, for which we have no competing solver to SEQOPT.

Problem	n	# fevals SEQOPT	# fevals NPSOL	# non-global local opts NPSOL
HS59	2	56	41	9/20
HS100	2	162	178	4/20

Table 1: Comparative results on two "standard" test problems

An earlier version of SEQOPT was tested on Boeing wing planform design problems. The earlier version differs from the current version only in that it did not allow the option of a poll from the least-infeasible point, once a feasible point had been found.

Figure 3 illustrates the wing planform design problem. The wing planform is the two-dimensional, downward projection of the wing. The design variables are the line segment end points for the wing leading edges, trailing edges, and spars. These are the dots in Figure 3. In addition to the above, there are variables related to wing thickness and aerodynamic loading. A typical design problem is to minimize direct operating cost subject to several constraints. The constraints include required range, maximum approach velocity, maximum required runway length, and several others. The analysis code is a sophisticated combination of preliminary design tools from many disciplines. The disciplines include structures, aerodynamics, weights, costing, and configuration management. SEQOPT results for two wing planform design problems are summarized in Table 2. Table 2 indicates the size of the problems and the number of function evaluations required by SEQOPT to solve the problem to approximately 0.001 relative accuracy. (This assumes that the solution found was, in fact, the true global solution. We believe this to be true, but it's impractical to verify this result.) The number of function evaluations given is the sum of those used in obtaining the initial response models and those computed dur-

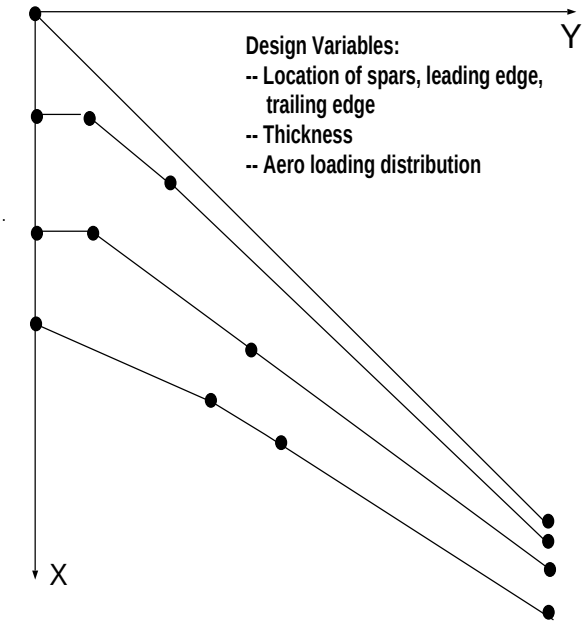

Figure 3: Wing planform design problem

ing the SEQOPT run. The initial models are based on 123 runs for Airplane A, and 126 runs for Airplane B.

The only comparative results available are for the optimal SEQOPT results versus the "baseline" design. For both airplanes, the baseline designs violated three of the constraints by considerable margins. SEQOPT found feasible airplane designs, that had lower operating costs than the baseline designs.

Design Problem	n	# of constrs	# fevals SEQOPT
Airplane A	15	11	304
Airplane B	15	11	292

Table 2: Summary for SEQOPT on planform design problems

6 Conclusions

A provably convergent method for surrogate-model-based optimization has been presented. The method features filter-based optimization, robust surrogate models, and a balance of local and global search. The method not only shows promise on standard test problems, but has been validated on problems of industrial significance. There is considerable opportunity for further work in many directions. There is an obvious need for more testing and refinement. This is particularly true due to the great algorithmic flexibility allowed within the filter-based framework. We are currently studying the relative merits of several different types of surrogate models. In the longer term, there is the opportunity to expand into methods and codes for discrete problems.

References

[1] Audet C. and Dennis J.E.Jr.(2000), "A Pattern Search Filter Method for Nonlinear Programming without Derivatives," *TR00-09* Department of Computational & Applied Mathematics, Rice University, Houston TX.

[2] Balabanov V., Giunta A.A., Grossman B., Haim D., Mason W.H., and Watson L.T.(1996), "Wing design for a high-speed civil transport using a design of experiments methodology", 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, AIAA, 1, September 4-6, 1996 AIAA-96-4001-CP, pp. 168-183.

[3] Booker A.J., Dennis J.E.Jr, Frank P.D., Serafini D.B., Torczon V. and Trosset M.W.(1999), "A rigorous framework for optimization of expensive functions by surrogates," *Structural Optimization* Vol.17 No.1, 1-13.

[4] Booker A.J.(1998), "Design and Analysis of Computer Experiments", Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA-98-4757.

[5] Booker A. J. (1998), "Examples of Surrogate Modeling of Computer Simulations", Presented at ISSMO/NASA/AIAA First Internet Conference on Approximations and Fast Reanalysis in Engineering Optimization.

[6] Booker A. J. (2000), "Well–Conditioned Kriging Models for Optimization of Computer Models, Boeing Phantom Works, Mathematics and Computing Technology, Report M&CT-TECH-002, February.

[7] Burgee S.L., Giunta A.A., Balabanov V., Grossman B., Mason W.H., Narducci R., Haftka R.T., and Watson L.T.(1996), "A coarse grained parallel variable–complexity multidisciplinary optimization paradigm", *Intl. J.*

Supercomputing Applications and High Performance Computing, 10, 269–299.

[8] Currin C., Mitchell T., Morris M., and Ylvisaker D.(1988), "A Bayesian approach to the design and analysis of computer experiments", Technical Report ORNL–6498, Oak Ridge National Laboratory.

[9] Currin C., Mitchell T., Morris M., and Ylvisaker D.(1991), "Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments", *Journal of the American Statistical Association*, 86(416), 953–963.

[10] Booker A.J., Dennis J.E.Jr., Frank P.D., Moore D.W. and Serafini D.B.(1998), "Managing Surrogate Objectives to Optimize a Helicopter Rotor Design Example", Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA-98-4717.

[11] deBoor, C. and Ron, A.(1992), "Computational aspects of polynomial interpolation in several variables", *Mathematics of Computation*, 58(198), 705–727.

[12] Giunta A.A.(1997), *Aircraft Multidisciplinary Optimization using Design of Experiments Theory and Response Surface Modeling Methods*, PhD thesis, Virginia Tech, 1997. Available as MAD 97-05-01, May 1997, Department of Aerospace and Ocean Engineering, Virginia Tech, 215 Randolph Hall, Blacksburg, Va 24061.

[13] Fletcher R. and Leyffer S.(1997), "Nonlinear Programming without a penalty function," Dundee University, Dept. of Mathematics, Report NA/171.

[14] Fletcher R, Leyffer S. and Toint Ph.L.(1998), "On the global convergence of an SLP-Filter algorithm," Dundee University, Dept. of Mathematics, Report NA/183.

[15] Fletcher R, Gould N.I.M., Leyffer S. and Toint Ph.L.(1999), "On the global convergence of trust-region SQP-Filter algorithms for general nonlinear programming," Department of Mathematics, FUNDP, Namur (B), Report 99/03.

[16] Gill P.E., Murray W., Suanders M.A. and Wright M.H.(1986) "User's Guide for NPSOL (Version 4.0): a Fortran Package for Nonlinear Programming," Department of Operations Research, Stanford University, Report SOL 86-2.

[17] Hock W. and Schittkowski K.(1981) "Test Examples for Nonlinear Programming Codes," Springer-Verlag, New York.

[18] Lewis R.M. and Torczon V.(1998), "Pattern search methods for linearly constrained minimization," *ICASE NASA Langley Research Center* TR 98-3. To appear in *SIAM Journal on Optimization*.

[19] Lewis R.M. and Torczon V.(1996), "Pattern search algorithms for bound constrained minimization," *SIAM Journal on Optimization*, Vol.9 No.4, 1082-1099.

[20] Lewis R.M. and Torczon V.(1996), "Rank ordering and positive basis in pattern search algorithms," *ICASE NASA Langley Research Center* TR 96-71.

[21] Lewis R.M. and Torczon V.(1998), "A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds," *ICASE NASA Langley Research Center* TR 98-31.

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

[23] Sacks J, Schiller S.B., and Welch W.J.(1989), "Designs for computer experiments", *Technometrics*, 31(1), 41–47.

[24] Sacks J., Welch W.J., Mitchell T.J., and Wynn H.P.(1989), "Design and analysis of computer experiments", *Statistical Science*, 4(4), 409–435.

[25] Schonlau M.(1997), *Computer experiments and global optimization*, PhD thesis, Statistics Department, University of Waterloo, Ontario, Canada.

[26] Simpson T.W. (1998), "Comparison of response surface and Kriging models in the multidisciplinary design of an aerospace nozzle", ICASE Report No. 98-16, February.

[27] Torczon V.(1997), "On the Convergence of Pattern Search Algorithms," *SIAM Journal on Optimization* Vol.7 No.1, 1–25.