SNOBFIT – Stable Noisy Optimization by Branch and Fit

WALTRAUD HUYER and ARNOLD NEUMAIER
Universität Wien

The software package SNOBFIT for bound constrained noisy optimization of an expensive objective function is described. It combines global and local search by branching and local quadratic fits. The program is made robust and flexible for practical use by allowing for soft or hidden constraints, batch function evaluations, change of search regions, etc.

Categories and Subject Descriptors: G.1.6 [**Optimization**]: Global optimization; Constrained optimization

General Terms: Algorithms

Additional Key Words and Phrases: Hidden constraints, noisy function values, soft constraints

1. INTRODUCTION

SNOBFIT (*s*table *n*oisy *o*ptimization by *b*ranch and *fit*) is a MATLAB package designed for selecting continuous parameter settings for simulations or experiments, performed with the goal of optimizing some user-specified criterion. Specifically, we consider the optimization problem

$$\begin{array}{l} \min f(x) \\ \text{s.t. } x \in [u, v], \end{array} \tag{1}$$

$u, v \in \mathbb{R}^n$, $u < v$ (with componentwise inequalities), i.e., $[u, v]$ is a bounded box in $\mathbb{R}^n$ with nonempty interior, and $f : D \to \mathbb{R}$, where D is a subset of $\mathbb{R}^n$ containing $[u, v]$. We shall call the process of obtaining an approximate function value $f(x)$ a *measurement* at the point x.

While there are many software packages that can handle such problems, they usually cannot cope well with one or more of the following difficulties arising in practice:

—the function values are expensive (for example, obtained by performing complex experiments or simulations);

—instead of a function value requested at a point x, only a function value at some nearby point $\tilde{x}$ is returned;

Authors' address: Institut für Mathematik, Universität Wien, Nordbergstraße 15, A-1090 Wien, Austria.

—the function values are noisy (inaccurate due to experimental errors or to low precision calculations);

—the objective function may have several local minimizers;

—no gradients are available;

—the problem may contain hidden constraints, i.e., a requested function value may turn out not to be obtainable;

—the problem may involve additional soft constraints;

—the user wants to measure at several points simultaneously or make parallel simulations;

—function values may be obtained so infrequently or in a heterogeneous environment that, between obtaining function values, the computer hosting the software is used otherwise or even switched off;

—the objective function or the search region may change during optimization, e.g., because users inspect the data obtained so far and this suggests to them a more realistic or more promising goal.

Many different algorithms (see, e.g., the survey [Powell 1998]) have been proposed for unconstrained or bound constrained optimization when first derivatives are not available. Conn et al. [1997] distinguish two classes of derivative-free optimization algorithms. *Sampling methods* or direct search methods proceed by generating a sequence of points; pure sampling methods tend to require rather many function values in practice. *Modeling methods* try to approximate the function over a region by some model function and the much cheaper surrogate problem of minimizing the model function is solved. Not all algorithms are equally suited for the application to noisy objective functions; for example, it would not be meaningful to use algorithms that interpolate the function at points that are too close together.

A method called DACE (design and analysis of computer experiments) (see [Sacks et al. 1989; Welch et al. 1992]) deals with finding a surrogate function for a function generated by computer experiments, which consist of a number of runs of a computer code with various inputs. These codes are typically expensive to run and the output is deterministic, i.e., rerunning the code with the same input gives identical results, but the output is distorted by high-frequency, low-amplitude oscillations. The lack of random error makes computer experiments different from physical experiments. The problem of fitting a response surface model to the observed data consists of the design problem, which is the problem of the choice of points where data should be collected, and the analysis problem of how the data should be used to obtain a good fit. The output of the computer code is modeled as a realization of a stochastic process; the method of analysis for such models is known as kriging in the mathematical geostatistics literature. For the choice of the sample points, "space-filling" designs such as orthogonal array-based Latin hypercubes [Tang 1993] are important; see [McKay et al. 1979] for a comparison of random sampling, stratified sampling and Latin hypercube sampling.

The SPACE algorithm (Stochastic Process Analysis of Computer Experiments) by Schonlau [Schonlau 1997; 2001; Schonlau et al. 1998] (see also [Jones et al. 1998] for the similar algorithm EGO) uses the DACE approach resp. Bayesian global optimization in order to find the global optimum of a computer model. The function

to be optimized is modeled as a Gaussian stochastic process, where all previous function evaluations are used to fit the statistical model. The algorithm contains a parameter that controls the balance between local and global search components of the optimization. The minimization is done in stages, i.e., rather than only one point, the algorithm samples a specified number of points at a time. Moreover, a method for dealing with nonlinear inequality constraints from additional response variables is proposed.

The authors of [Booker et al. 1999; Trosset and Torczon 1997] consider traditional iterative methods and DACE as opposite ends of the spectrum of derivative-free optimization methods. They combine pattern search methods and DACE for the bound constrained optimization of an expensive objective function. Pattern search algorithms are iterative algorithms that produce a sequence of points from an initial point, where the search for the next point is restricted to a grid containing the current iterate and the grid is modified as optimization progresses. The kriging approach is used to construct a sequence of surrogate models for the objective functions, which are used to guide a grid search for a minimizer. Moreover, Booker et al. [1999] also consider the case that the routines evaluating the objective function may fail to return $f(x)$ even for feasible x, i.e., the case of hidden constraints.

Jones [2001] presents a taxonomy of existing approaches for using response surfaces for global optimization. Seven methods are compared and illustrated with numerical examples that show their advantages and disadvantages.

Elster and Neumaier [1995] develop an algorithm for the minimization of a low-dimensional, noisy function with bound constraints, where no knowledge about the statistical properties of the noise is assumed, i.e., it may be deterministic or stochastic (but must be bounded). The algorithm is based on the use of quadratic models minimized over adaptively defined trust regions together with the restriction of the evaluation points to a sequence of nested grids.

Anderson and Ferris [2001] consider the unconstrained optimization of a function subject to random noise, where it is assumed that averaging repeated observations at the same point leads to a better estimate of the objective function value. They develop a simplicial direct search method including a stochastic element and prove convergence under certain assumptions on the noise; however, the proof of convergence does not work in the absence of noise.

Carter et al. [2001] deal with algorithms for bound constrained noisy problems in gas transmission pipeline optimization. They consider the DIRECT algorithm of [Jones et al. 1993], which proceeds by repeated subdivision of the feasible region, implicit filtering, a sampling method designed for problems that are low-amplitude, high frequency perturbations of smooth problems, and a new hybrid of implicit filtering and DIRECT, which attempts to combine the best features of the two other algorithms. In addition to the bound constraints, the objective function may fail to return a value for some feasible points. The traditional approach assigns a large value to the objective function when it cannot be evaluated, which results in slow convergence if the solution lies on a constraint boundary. In [Carter et al. 2001] a modification is proposed; the function value is derived from nearby feasible points rather than assigning an arbitrary value. In [Choi and Kelley 2000], implicit filtering is coupled with the BFGS quasi-Newton update.

The goal of the present paper is to describe a new algorithm that addresses all the above points. In Section 2 the basic setup of SNOBFIT is presented; for details we refer to Section 6. In Sections 3 to 5, we describe some important ingredients of the algorithm, namely the branching algorithm, the local quadratic models and the safeguarded nearest neighbors, and the five classes of points generated by SNOBFIT, respectively. In Section 7, the convergence of the algorithm is established, and in Section 8 a penalty function is proposed to handle soft constraints. Finally, in Section 9 numerical results are presented.

2. BASIC SETUP OF SNOBFIT

The algorithm SNOBFIT described in this paper

—produces a user-specified number of suggested evaluation points in each step;
—proceeds by successive partitioning of the box (*branch*) and building local quadratic models (*fit*);
—combines local and global search and allows the user to determine which of both should be emphasized;
—handles local search from the best point with the aid of trust regions;
—allows for hidden constraints and assigns to such points a function value based on the function values of nearby feasible points.

In the following we call a *job* the attempt to solve one and the same problem with SNOBFIT. The function and some of the tuning parameters of SNOBFIT stay the same in a job. A job consists of several *calls* to SNOBFIT. We use the notion *initial call* for the first call of a job and *continuation call* for any later call of SNOBFIT on the same job.

One call to SNOBFIT roughly proceeds as follows; for a detailed description of the algorithm and its input and output parameters we refer to Section 6. The main input ingredients are a (possibly empty) list x^j, $j = 1, \ldots, J$, of points, their corresponding function values f_j and the uncertainties Δf_j of the function values. If the user has not been able to obtain a function value for x^j, f_j should be set to NaN. In a continuation call, in addition to these "new" points, a list of "old" points, corresponding function values and uncertainties and a kD tree of subboxes are loaded from a working file; in an initial call the kD tree consists of only one box. One of the input parameters that are only set in an initial call and stay the same during the whole job is a resolution vector $\Delta x \in \mathbb{R}^n$, $\Delta x > 0$, i.e., two points are considered to be different if they differ by at least Δx_i in at least one coordinate i.

The algorithm therefore only suggests evaluation points whose ith coordinate is an integral multiple of Δx_i. It first splits all subboxes containing more than one point and generates a kD tree of subboxes containing exactly one point. Splitting is done by the branching algorithm described in Section 3.

Section 4 is devoted to selecting the local quadratic models and the safeguarded nearest neighbors. In the case of an initial call or if the current best point x^{best} has changed compared to the previous call, a local quadratic model around x^{best} is computed. Then local quadratic models are estimated around all new points and all old points whose safeguarded nearest neighbors have changed. The algorithm generates five types of points, explained in detail in Section 5, where the function

should be evaluated before the next call to SNOBFIT. Two points (type 1 and 2) are generated from the best point with the aid of its local quadratic model and trust regions. The points of type 3 are generated with the aid of the local quadratic models at positions where good function values are expected, and the fourth type of points are points in unexplored regions. Points of type 5 are only produced if the algorithm does not manage to reach the desired number of points by generating points of types 1 to 4, for example, when there are not enough points available yet to build local quadratic models, which happens in particular in an initial call with an empty set of input points and function values. The points of type 5 are chosen from a set of random points such that their distances from the already sampled points are maximal; for details see Section 5.

The adjective "stable" in the name of the algorithm is to be understood in the sense of "stable with respect to both noise in the data and the input of the user". For example, it is permitted

—to evaluate the function at other points than the ones suggested by SNOBFIT;
—to use an already sampled point again as input with a different function value – in that case an averaged function value is computed (see Section 6, Step 2);
—to use an empty set of points as input for a call to SNOBFIT (which might be useful in an initial call); and
—to use points outside the box bounds $[u, v]$ as input, which results in an extension of the search region (see Section 6, Step 1).

3. THE BRANCHING ALGORITHM

We assume that $[u, v]$ is a bounded box where the function should be explored. We want to split a subbox $[\underline{x}, \overline{x}]$ containing the pairwise distinct points x^k, $k = 1, \ldots, K$, $K \geq 2$, such that each subbox contains exactly one point. If $K = 2$, we choose i with $|x_i^1 - x_i^2|/(v_i - u_i)$ maximal and split along the ith coordinate at $y_i = \lambda x_i^1 + (1-\lambda)x_i^2$, where λ is the golden section number $\rho := \frac{1}{2}(\sqrt{5} - 1) \approx 0.62$ if $f(x^1) \leq f(x^2)$ and $\lambda = 1 - \rho$ otherwise. The subbox with the lower function value gets the larger part of the original box so that it is eligible for being selected for the generation of a point of type 4 more quickly.

If $K > 2$ we apply the following procedure:

while there is a subbox containing more than one point
  choose the subbox containing the highest number of points
  choose i such that the variance of $x_i/(v_i - u_i)$ is maximal, where the variance is taken over all points x in the box
  sort the points such that $x_i^1 \leq x_i^2 \leq \ldots$
  split in the coordinate i at $y_i = \lambda x_i^j + (1 - \lambda)x_i^{j+1}$, where $j = \text{argmax}(x_i^{j+1} - x_i^j)$
  and $\lambda = \rho$ if $f(x^j) \leq f(x^{j+1})$ and $\lambda = 1 - \rho$ otherwise
end while

To each subbox $[\underline{x}, \overline{x}]$ we assign its *smallness*

$$S := -\sum_{i=1}^{n} \text{round}(^2\log((\overline{x}_i - \underline{x}_i)/(v_i - u_i))) \approx \text{const} - {}^2\log(\text{volume}), \qquad (2)$$

where round is the function rounding to nearest integers. This quantity roughly measures how many bisections are necessary to obtain this box from $[u, v]$. We have $S = 0$ for the exploration box $[u, v]$, and S is large for small boxes.

4. LOCAL QUADRATIC MODELS AND SAFEGUARDED NEAREST NEIGHBORS

For the quadratic models a Hessian fit around the best point x^{best} is computed. Assume that x^k, $k = 1, \ldots, M$, $M \geq n$, is the list of points distinct from x^{best} that have already been sampled, let f_k be the corresponding function values and $s^k := x^k - x^{\text{best}}$, and define model errors ε_k by the equations

$$f_k - f_{\text{best}} = g^T s^k + \frac{1}{2}(s^k)^T G s^k + \varepsilon_k (s^k)^T H s^k, \quad k = 1, \ldots, M, \tag{3}$$

where $H := (\sum s^l (s^l)^T)^{-1}$. The $N := n + \binom{n+1}{2}$ parameters in (3) consisting of the vector $g \in \mathbb{R}^n$ and the symmetric matrix $G \in \mathbb{R}^{n \times n}$ are determined by minimizing $\sum \varepsilon_k^2$. Actually, we make an economy size QR factorization

$$(s^1, \ldots, s^M)^T = QR$$

with an orthogonal matrix $Q \in \mathbb{R}^{M \times n}$ and a square upper triangular matrix $R \in \mathbb{R}^{n \times n}$. Then $H = (R^T R)^{-1}$ and

$$\beta_k := (s^k)^T H s^k = \|R^{-T} s^k\|^2, \quad k = 1, \ldots, M,$$

and $L := R^{-T}$ is stored for further use (cf. Section 6, Step 7). In addition, we introduce the abbreviation

$$\sigma_G^2 := \frac{1}{\max(M - N', 1)} \sum \hat{\varepsilon}_k^2$$

with the optimized errors $\hat{\varepsilon}_k$ from the Hessian fit. N' is the number of the N parameters determined above that are unequal to zero.

For each point x its $n+5$ safeguarded nearest neighbors are determined as follows. First, for $i = 1, \ldots, n$, the point closest to x among the points y not yet in the list satisfying $|x_i - y_i| \geq \Delta x_i$ is chosen. The list of nearest neighbors is filled up with the (at least 5) points closest to x not yet in the list.

For the local quadratic fit around an arbitrary point x (including x^{best}) we use as Hessian a suitable multiple of the Hessian matrix G estimated above, and define new model errors ε_k by

$$f_k = f + g^T s^k + \frac{\gamma}{2}(s^k)^T G s^k + \varepsilon_k \sqrt{\Delta f_k^2 + \sigma_G^2 \beta_k^2},$$

where f_k and Δf_k are the corresponding function values and their uncertainties as explained in Section 1, $\beta_k := \|L(x^k - x^{\text{best}})\|^2$, $s^k := x^k - x$ and x^k takes in turn the values of x and its $n + 5$ safeguarded nearest neighbors. The $n + 2$ parameters $f, \gamma \in \mathbb{R}$ and $g \in \mathbb{R}^n$ are determined by minimizing $\sum \varepsilon_k^2$. The factor after ε_k is chosen such that for points with a large Δf_k (i.e., inaccurate function value) and a large β_k (i.e., far away from x^{best}), a larger error in the fit is permitted. We use $n + 6$ points to determine $n + 2$ parameters in order to smooth down noise.

5. RECOMMENDED EVALUATION POINTS

SNOBFIT generates points belonging to five classes. Let $\Delta x \in \mathbb{R}^n$, $\Delta x > 0$, be a resolution vector as defined in Section 2, and let $[u', v'] \subseteq [u, v]$ be the box where the points are to be generated.

The (only) point of class 1 is obtained by minimizing the local quadratic model around x^{best} over $[\max(x^{\text{best}} - d, u), \min(x^{\text{best}} + d, v)]$, where d is a trust region radius and max and min are interpreted componentwise. If the resulting point is not at the boundary of the box, d is reduced such that the point is on the new boundary.

Subsequently the (only) point of class 2 is generated by minimizing the quadratic model around x^{best} over the box $[\max(x^{\text{best}} - \rho d, u), \min(x^{\text{best}} + \rho d, v)]$, where ρ is the golden section number defined in Section 3.

For each box $[\underline{x}, \overline{x}]$ with corresponding point x the quadratic model around x is minimized over $[\underline{x}', \overline{x}']$, where

$$\begin{aligned} \underline{x}'_i &:= \begin{cases} \underline{x}_i + 0.05(\overline{x}_i - \underline{x}_i) & \text{if } (\overline{x}_i - \underline{x}_i)/\Delta x_i > 0.05 \max_j (\overline{x}_j - \underline{x}_j)/\Delta x_j, \\ x_i & \text{otherwise,} \end{cases} \\ \overline{x}'_i &:= \begin{cases} \overline{x}_i - 0.05(\overline{x}_i - \underline{x}_i) & \text{if } (\overline{x}_i - \underline{x}_i)/\Delta x_i > 0.05 \max_j (\overline{x}_j - \underline{x}_j)/\Delta x_j, \\ x_i & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

This definition serves to avoid the boundary and too narrow splits. The minimization problem is a bound constrained quadratic program, generally indefinite, and is solved with the MATLAB package MINQ [Neumaier 1998]. A point y with the model function value f_y is obtained. If $|x - y| < 0.05(\overline{x} - \underline{x})$ (with componentwise inequalities), the point y is considered to be too close to y. In this case let $i := \text{argmax}|x_i - y_i|/(\overline{x}_i - \underline{x}_i)$ (where ties are broken by taking the index with the largest $(\overline{x}_i - \underline{x}_i)/(v_i - u_i)$). Then y_i is replaced by

$$y'_i := \begin{cases} x_i + 0.05(\overline{x}_i - \underline{x}_i) & \text{if } (y_i > x_i \text{ and } x_i + 0.05(\overline{x}_i - \underline{x}_i) \le \overline{x}_i) \\ & \text{or } x_i - 0.05(\overline{x}_i - \underline{x}_i) < \underline{x}_i, \\ x_i - 0.05(\overline{x}_i - \underline{x}_i) & \text{otherwise,} \end{cases}$$

and the model function value f_y for the resulting new point is computed.

The points of class 3 are points in alternative valleys. A point is called *local* if its $n + 5$ safeguarded nearest neighbors have higher function values. First the points y generated from local points by minimization of the quadratic model as described above are chosen in the order of ascending f_y and afterwards also nonlocal points are taken (again in the order of ascending f_y) if they differ from all previously generated points of class 3 by at least $0.1(v'_i - u'_i)$ in at least one coordinate i until the desired number of points of class 3 is reached. The minimal distance between two points of class 3 prevents that the points are essentially copies of the same point.

The points of class 4 are points in unexplored regions. For a box $[\underline{x}, \overline{x}]$ with corresponding point x, the point z of class 4 is defined by

$$z_i := \begin{cases} \frac{1}{2}(\underline{x}_i + x_i) & \text{if } x_i - \underline{x}_i > \overline{x}_i - x_i, \\ \frac{1}{2}(x_i + \overline{x}_i) & \text{otherwise.} \end{cases}$$

Such a box is selected in order of increasing smallness S (i.e. boxes with a large volume are chosen first) defined by (2); since S is integral, ties are frequent and are

broken by preferring for the same S boxes with a small function value $f(x)$.

The points of class 5 serve to fill up the set of recommended evaluation points. In order to generate n_5 points of class 5, $100n_5$ random uniformly distributed points in $[u', v']$ are drawn. Initialize X as the set of all points that have already been sampled (including the recommended evaluation points of this call to SNOBFIT), and let Y be the set of the $100n_5$ random points. Then the set X_5 of points of class 5 is generated as follows:

$X_5 = \emptyset$
if $X = \emptyset$
  choose $y \in Y$
  $X = \{y\}$; $Y = Y \setminus \{y\}$; $X_5 = X_5 \cup \{y\}$;
end if
while $|X_5| < n_5$
  $y = \operatorname{argmax} \min_{x \in X} \|x - y\|_2$;
  $X = X \cup \{y\}$; $Y = Y \setminus \{y\}$; $X_5 = X_5 \cup \{y\}$;
end while

The coordinates of all these points are rounded to integral multiples of Δx_i (in the case of points of class 3 or 4 into the corresponding box) and a point is only accepted if it differs from all already sampled points by at least Δx_i in at least one coordinate i. This has the effect that some calls to SNOBFIT may not return a point of class 1 and/or of class 2.

6. THE SNOBFIT ALGORITHM

Now we are ready to describe the SNOBFIT algorithm in detail. We look for a solution of the problem (1) by repeated calls to SNOBFIT. In each call to SNOBFIT, a (possibly empty) list of points x^j, $j = 1, \ldots, J$, their function values f_j, the uncertainties of the function values Δf_j, a natural number n_{req}, two n-vectors u' and v', $u' \le v'$, and a number $p \in [0, 1]$ are fed into the program. If the user has not been able to obtain a function value for x^j, f^j should be set to NaN (while x^j should be deleted if a function value has not even been tried), and p determines the fraction of points of class 4 among the set of points of class 3 and 4. The program then returns n_{req} suggested evaluation points in the box $[u', v']$, their class as defined in Section 5, their model function values, the corresponding estimated uncertainties, the current best point, the current best function value and a measure of the accuracy of the quadratic model at the best point (see Step 7). The idea of the algorithm is that these points and their function values are used as input for the next call to SNOBFIT, but the user may feed instead other points or even an old point with a newly measured function value into the program. For example, some suggested evaluations may not have been feasible or successful, or the experiment does not allow to locate the position precisely; the position obtained differs from the intended position but can be measured with a precision higher than the error made.

When a job is started (i.e., in the case of an initial call to SNOBFIT), an n-vector $\Delta x > 0$ is needed as additional input, which is a resolution vector as described in Section 2. In the continuation calls to SNOBFIT, Δx, all function values sampled previously and all parameters characterizing the state of the splitting procedure are

reloaded from a working file created after the previous call to SNOBFIT.

One call to SNOBFIT proceeds in the following 11 steps.

STEP 1. The vectors u and v are defined such that $[u, v]$ is the smallest box containing $[u', v']$, all new input points and, in the case of a continuation call to SNOBFIT, also the box $[u^{\rm old}, v^{\rm old}]$ from the previous iteration. $[u, v]$ is considered to be the box to be explored and a kD tree of subboxes of $[u, v]$ is generated; however, all suggested new evaluation points are in $[u', v']$.

STEP 2. Duplicates in the set of points consisting of the "new" points and in a continuation call also the "old" points from the previous iterations are thrown away, and the corresponding function value f and uncertainty Δf are updated. If a point has been put into the algorithm m times with function values $f_1, \ldots, f_m$ and corresponding uncertainties $\Delta f_1, \ldots, \Delta f_m$, the quantities f and Δf are defined by

$$f = \frac{1}{m}\sum_{i=1}^{m} f_i, \qquad \Delta f = \sqrt{\frac{1}{m}\sum_{i=1}^{m}((f_i - f)^2 + \Delta f_i^2)}.$$

STEP 3. In the case of an initial call to SNOBFIT, the trust region radius d for minimization from the best point is initialized as $d = \frac{1}{4}(v - u)$. In a continuation call, let f_1 and f_2 be the function values of the points closest to the points of class 1 and 2, respectively, from the previous call to SNOBFIT (these are taken to be the presumed points of class 1 or 2; whether this is indeed the case depends on the extent to which the calling agent used the recommended measurement positions), and let $f_{\rm best}$ be the best function value and $d^{\rm old}$ be the trust region radius from the previous step. If $f_1 < \min(f_2, f_{\rm best})$, i.e., if the presumed point of class 1 is better than the presumed point of class 2 and the previous best point, the trust region radius is enlarged according to $d = d^{\rm old}/\rho$, where ρ is the golden section number defined in Section 3. If $f_{\rm best} < \min(f_1, f_2)$, i.e., if neither the presumed point of type 1 nor the presumed point of type 2 has brought an improvement in function value, the trust region is reduced according to $d = \rho d^{\rm old}$. In all other cases, we keep $d = d^{\rm old}$.

STEP 4. All current boxes containing more than one point (in the case of an initial call the kD tree of boxes consists of the single box $[u, v]$) are split according to the algorithm described in Section 3 and the smallness is computed for these boxes. If $[u, v]$ is larger than $[u^{\rm old}, v^{\rm old}]$ in a continuation call, the box bounds and the smallness are updated for the boxes for which this is necessary.

STEP 5. If the number of already sampled points is less than $n + 6$, go to Step 11. Otherwise, for each new point x a vector pointing to $n + 5$ safeguarded nearest neighbors is computed. The neighbor lists of some old points are updated in the same way if necessary.

STEP 6. For any new input point x with function value NaN (which means that the function value could not be determined at that point) and all old points marked infeasible whose nearest neighbors have changed, let f_1 and f_2 be the minimal and maximal function value among the safeguarded nearest neighbors of x, excepting the neighbors where no function value could be obtained, and let f_1 and f_2 be the minimal and maximal function value among all feasible points sampled so far in the case that all safeguarded nearest neighbors of x were infeasible. Then we set $f = f_2 + 10^{-3}(f_2 - f_1)$ and $\Delta f = \Delta f_2$.

STEP 7. The current best point x^{best} and the current best function value f_{best} are determined. In an initial call or if x^{best} has changed, a Hessian fit around x^{best} is computed according to Section 4. Otherwise, for all new points x^k the quantities β_k, defined by $\beta_k := \|L(x^k - x^{\text{best}})\|^2$, are computed, where the matrix L is defined in Section 4. Moreover, local quadratic fits around all new points and all old points with changed nearest neighbors are computed and the potential points of type 3 in each box are determined. The quantity $\max |f_k - q(x^k)|$, where q denotes the local quadratic model around x^{best} and the maximum is taken over $x^k = x^{\text{best}}$ and its $n + 5$ safeguarded nearest neighbors, is computed as a measure of the accuracy of the quadratic model around x^{best}.

STEP 8. The point of type 1 and the point of type 2 are generated as described in Section 5, which gives $n_{12} \leq 2$ evaluation points, using the safeguard of Section 5.

STEP 9. To generate the remaining $m := n_{\text{req}} - n_{12}$ evaluation points, let $n_1 := \lfloor pm \rfloor$ and $n_2 := \lceil pm \rceil$. Then a random number generator sets $m_1 = n_1$ with probability $mp - n_1$ and $m_1 = n_2$ otherwise.

STEP 10. $m - m_1$ points of type 3 and m_1 points of type 4 are generated as described in Section 5. If the algorithm does not find m_1 acceptable points of type 3 according to the choice of Δy, more points of type 4 are generated in order to obtain (if possible) n_{req} suggested evaluation points. Only boxes that do not contain a just generated point of type 3 are eligible for the generation of a point of type 4.

STEP 11. If the number of suggested evaluation points is still less than n_{req}, the set of evaluation points is filled up with points of type 5 as described in Section 5. If quadratic models are already available (i.e., if the number of sampled function values is at least $n + 6$), we assign to the points of type 5 the model function values and variances obtained from the quadratic models pertaining to their boxes; otherwise, these quantities are set to NaN.

Stopping criterion. Since SNOBFIT is called explicitly before each new measurement, the search continues as long as the calling agent (an experimenter or a program) finds it reasonable to continue. A natural stopping test would be to quit exploration (or move exploration to a different "box of interest") if for a number of calls to SNOBFIT no new point of type 1 is generated. Indeed, this means that SNOBFIT thinks that, according to the current model, the best point is already known; but since the model may be inaccurate, it is sensible to have this confirmed repeatedly before actually stopping.

Changing the objective function. We assume that the objective function f is of the form $f(x) = \varphi(y(x))$, where the vector $y(x) \in \mathbb{R}^k$ is obtained by time-consuming experiments or simulations but the function φ can be computed cheaply. Suppose that the objective function f has already been evaluated at $x_1, \ldots, x_M$ and that, at some moment, the user decides that the objective function $\tilde{f}(x) = \psi(y(x))$ is more appropriate, where ψ can be computed cheaply, too. Then the already determined vectors $y(x_1),\ldots,y(x_M)$ can be used to compute $\tilde{f}(x_1),\ldots,\tilde{f}(x_M)$, and we can start a new SNOBFIT job with $x_1,\ldots,x_M$ and $\tilde{f}(x_1), \ldots,\tilde{f}(x_M)$, i.e., we use the old grid of points but make a new partition of the space before the SNOBFIT

algorithm is continued.

7. CONVERGENCE OF THE ALGORITHM

For a convergence analysis, we assume that $\Delta x = 0$ and that the exploration box $[u, v]$ is scaled to $[0, 1]^n$. In order to make the algorithm theoretically meaningful for $\Delta x = 0$, we replace (4) by

$$\begin{aligned} \underline{x}'_i &:= \begin{cases} \underline{x}_i + 0.05(\overline{x}_i - \underline{x}_i) & \text{if } \overline{x}_i - \underline{x}_i > 0.05 \max_j(\overline{x}_j - \underline{x}_j), \\ x_i & \text{otherwise,} \end{cases} \\ \overline{x}'_i &:= \begin{cases} \overline{x}_i - 0.05(\overline{x}_i - \underline{x}_i) & \text{if } \overline{x}_i - \underline{x}_i > 0.05 \max_j(\overline{x}_j - \underline{x}_j), \\ x_i & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

and do not do any rounding, which means that all points of type 3 and 4 are accepted since they differ from the old point in the box. Moreover, we assume that $[u', v'] = [u, v] = [0, 1]^n$ during the whole job, that at least one point of class 4 is sampled in each call to SNOBFIT and that the function is evaluated at the points suggested by SNOBFIT. Then SNOBFIT is guaranteed to converge to a global minimizer if the objective function is continuous – or at least continuous in the neighborhood of a global optimizer. This follows from the fact that, under the assumptions made above, the set of points sampled by SNOBFIT forms a dense subset of the search space. That is, given any point $x \in [u, v]$ and any $\delta > 0$, SNOBFIT will eventually sample a point within a distance δ from x. We now establish this property.

When a box with smallness S is split into two parts, the following two cases are possible if S_1 and S_2 denote the smallnesses of the two subboxes. When the box is split into two equal halves, we have $S_1 = S_2 = S + 1$, and otherwise we have $S_1 = S$ and $S_2 = S + 1$ (after renumbering the two subboxes if necessary). This implies that, if $S_{\min}$ is the minimal smallness occurring in the kD tree of subboxes, the number of boxes with smallness $S_{\min}$ does not increase after a box has been split (it either stays the same or decreases by one).

The definition of the point of type 3 prevents splits in too narrow variables and the same holds for the points of type 4. Indeed, consider a box $[\underline{x}, \overline{x}]$ containing the point x and let

$$\overline{x}_i - \underline{x}_i \geq \overline{x}_j - \underline{x}_j \quad \text{for } j = 1, \ldots, n \tag{6}$$

(i.e., the ith extension of the box is the largest one). In order to simplify notation, we assume that

$$\overline{x}_j - x_j \geq x_j - \underline{x}_j \tag{7}$$

and thus we have $z_j = \frac{1}{2}(\overline{x}_j + x_j)$ for $j = 1, \ldots, n$ (the other cases can be handled similarly) for the point z of type 4. According to the branching algorithm defined in Section 3, the box will be split along the coordinate k with $z_k - x_k = \frac{1}{2}(\overline{x}_k - x_k)$ maximal, i.e., $\overline{x}_k - x_k \geq \overline{x}_j - x_j$ for $j = 1, \ldots, n$. Then (6), (7) and the definition of k imply

$$\overline{x}_k - \underline{x}_k \geq \overline{x}_k - x_k \geq \overline{x}_i - x_i \geq \frac{1}{2}(\overline{x}_i - \underline{x}_i),$$

which means that only splits along coordinates k with $\overline{x}_k - \underline{x}_k \geq \frac{1}{2}\max(\overline{x}_j - \underline{x}_j)$ are possible.

The fact that the worst case of a type 4 split is a factor $\frac{1}{4}(1+\sqrt{5}) \approx 0.8090$ and that the worst factor of a type 3 split is $\frac{1}{40}(37+\sqrt{5}) \approx 0.9809$ guarantee that the boxes containing any point $x \in [u, v]$ shrink sufficiently fast after sufficiently many splits. More precisely, for each $\delta > 0$ and $i = 1, \ldots, n$, there exists an $m_i(\delta) \in \mathbb{N}$ such that the ith side length of the box containing x is less than δ if the box has been split at least $m_i(\delta)$ times along the ith coordinate. Moreover, since we assume that in each call to SNOBFIT at least one box with minimal smallness $S_{\min}$ is split and the number of boxes with smallness $S_{\min}$ does not increase, this implies that after sufficiently many calls to SNOBFIT, there are no boxes with smallness $S_{\min}$ left any more.

If we now consider the subbox containing a global minimizer x^*, this box will eventually be split at least $m_i(\delta)$ times along each coordinate i, $i = 1, \ldots, n$, since the safeguards against too narrow splits prevent that a box is split along certain coordinates all the time and rarely along others and the choice of boxes eligible for the generation of a point of type 4 guarantees that the box containing x^* will eventually be split again.

These properties give the following convergence theorem for SNOBFIT with $\Delta x = 0$ and at least one point of type 4 in every call to SNOBFIT.

THEOREM 7.1. *Suppose that the global minimization problem* (1) *has a solution* $x^* \in [u, v]$, *and that* $f : [u, v] \to \mathbb{R}$ *is continuous in a neighborhood of* x^*, *and let* $\varepsilon > 0$. *Then the algorithm will eventually find a point* x *with* $f(x) < f(x^*) + \varepsilon$, *i.e., the algorithm converges.*

8. HANDLING GENERAL CONSTRAINTS

In this section we consider the constrained optimization problem

$$\begin{array}{l} \min f(x) \\ \text{s.t. } x \in [u, v], \quad F(x) \in \mathbf{F}, \end{array} \tag{8}$$

where, in addition to the assumptions after (1), $F : [u, v] \to \mathbb{R}^m$ is a vector of m continuous contraint functions $F_1(x), \ldots, F_m(x)$, and $\mathbf{F} := [\underline{F}, \overline{F}]$ is a box in $\mathbb{R}^m$ defining the constraints on $F(x)$.

Traditionally (see [Fiacco and McCormick 1990]), constraints that cannot be handled explicitly are accounted for in the objective function, using simple l_1 or l_2 penalty terms for constraint violations, or logarithmic barrier terms penalizing the approach to the boundary. There are also so-called exact penalty functions whose optimization gives the exact solution (see, e.g., [Nocedal and Wright 1999]); however, this only holds if the penalty parameter is large enough, and what is large enough cannot be assessed without having global information.

The use of more general transformations (cf. [Dallwig et al. 1997]) gives rise to more precisely quantifiable approximation results. In particular, if it is known in advance that all constraints apart from the simple constraints are soft constraints only (so that some violation is tolerated), one may pick a transformation that incorporates prescribed tolerances into the reformulated simply constrained problem:

THEOREM 8.1. (Soft optimality theorem). *For suitable* $\Delta > 0, \underline{\sigma}_i, \overline{\sigma}_i > 0, f_0 \in$

$\mathbb{R}$, *let*

$$q(x) = \frac{f(x) - f_0}{\Delta + |f(x) - f_0|},$$

$$\delta_i(x) = \begin{cases} (F_i(x) - \underline{F}_i)/\underline{\sigma}_i & \text{if } F_i(x) \le \underline{F}_i, \\ (F_i(x) - \overline{F}_i)/\overline{\sigma}_i & \text{if } F_i(x) \ge \overline{F}_i, \\ 0 & \text{otherwise}, \end{cases}$$

$$r(x) = 2\big(\sum \delta_i^2(x)\big)\big/\big(1 + \sum \delta_i^2(x)\big).$$

Then the merit function

$$f_{merit}(x) = q(x) + r(x)$$

has its range bounded by $]-1, 3[$*, and the global minimizer* $\hat{x}$ *of* f_{merit} *in* $[u, v]$ *either satisfies*

$$F_i(\hat{x}) \in [\underline{F}_i - \underline{\sigma}_i, \overline{F}_i + \overline{\sigma}_i] \quad \text{for all } i, \tag{9}$$

$$f(\hat{x}) \le \min\{f(x) \mid F(x) \in \mathbf{F}, x \in [u, v]\}, \tag{10}$$

or one of the following two conditions holds:

$$\{x \in [u, v] \mid F(x) \in \mathbf{F}\} = \emptyset, \tag{11}$$

$$f_0 < \min\{f(x) \mid F(x) \in \mathbf{F}, x \in [u, v]\}. \tag{12}$$

PROOF. Clearly, $q(x) \in]-1, 1[$ and $r(x) \in [0, 2[$, so that $f(x) \in]-1, 3[$. If there is a feasible point x with $f(x) \le f_0$ then $q(x) \le 0$, $r(x) = 0$ at this point. Since f_{merit} is monotone increasing in $q + r$, we conclude from $f_{\text{merit}}(\hat{x}) \le f_{\text{merit}}(x)$ that

$$q(\hat{x}) \le q(\hat{x}) + r(\hat{x}) \le q(x) + r(x) = q(x),$$

$$-1 + r(\hat{x}) \le q(\hat{x}) + r(\hat{x}) \le q(x) + r(x) \le 0,$$

hence $f(\hat{x}) \le f(x)$, giving (10), and $r(\hat{x}) \le 1$,

$$\delta_i^2(\hat{x}) \le \sum \delta_i^2(\hat{x}) \le 1,$$

giving (9). □

Since the merit function is bounded by $]-1, 3[$ even if f and/or some F_i are unbounded, the formulation is able to handle so-called *hidden constraints*. There, the conditions for infeasibility are not known explicitly but are discovered only when attempting to evaluate one of the functions involved. In such a case, if the function cannot be evaluated, the merit function value can be simply set to 3.

(11) and (12) are degenerate cases that do not occur if a feasible point is already known and we choose f_0 as the function value of the best feasible point known (at the time of posing the problem). A suitable value for Δ is the median of the $|f(x) - f_0|$ for an initial set of trial points (in the context of global optimization often determined by a space-filling design [McKay et al. 1979; Owen 1992; 1994; Sacks et al. 1989; Tang 1993]). The number σ_i measures the degree to which

the constraint $F_i(x) \in \mathbf{F}_i$ may be softened; suitable values are in many practical applications available from the meaning of the constraints.

Of course there are other choices for $q(x)$, $r(x)$, $f_{\text{merit}}(x)$ with the same properties. The choices given are simple and lead to a continuously differentiable merit function with Lipschitz-continuous gradient if f and F have these properties. (The denominator of $q(x)$ is nonsmooth, but only when the numerator vanishes, so that this only affects the Hessian.)

9. NUMERICAL RESULTS

In this section we report results of SNOBFIT on the 9 test functions used in [Jones et al. 1993], where an extensive comparison of algorithms is presented; this test set was also used in [Huyer and Neumaier 1999]. Let n be the dimension of the problem, and the default box bounds $[u, v]$ from the literature were used. The algorithm was started with $n + 6$ points chosen at random from $[u, v]$. In each call to SNOBFIT, $n + 6$ points in $[u, v]$ were generated, and we set $\Delta x = 10^{-4}(v - u)$ and $p = 0.1$. Instead of a test function $f(x)$, we considered

$$\tilde{f}(x) := f(x) + \sigma N,$$

where N is a normally distributed random variable with mean 0 and variance 1, and Δf was set to $\max(3\sigma, \varepsilon)$, where $\varepsilon := 2.22 \cdot 10^{-16}$ is the machine precision. The algorithm was stopped if $(f_{\text{best}} - f^*)/|f^*| < 10^{-2}$, where f^* is the known optimal function value (which is $\neq 0$ for our test set). The number of function calls was limited to 10000.

Since a random element is contained in the initial points as well as the function, 10 jobs were computed with each function and each value of σ, and the median nf_{med} of function calls needed to find a global minimizer was computed. In Table I, the dimensions n, the standard box bounds $[u, v]$, the number n_{slow} of jobs where a global minimizer was not found within 5000 function evaluations, and the median nf_{med} are given.

			$\sigma = 0$		$\sigma = 0.01$		$\sigma = 0.1$	
	n	$[u, v]$	n_{slow}	nf_{med}	n_{slow}	nf_{med}	n_{slow}	nf_{med}
Branin	2	$[-5, 10] \times [0, 15]$	0	64	0	68	0	48
Six-hump camel	2	$[-3, 3] \times [-2, 2]$	0	48	0	48	0	44
Goldstein–Price	2	$[-2, 2]^2$	0	100	0	92	0	100
Shubert	2	$[-10, 10]^2$	0	112	0	112	0	116
Hartman 3	3	$[0, 1]^3$	0	76.5	0	63	0	63
Hartman 6	6	$[0, 1]^6$	0	276	3	282	0	366
Shekel 5	4	$[0, 10]^4$	0	580	0	295	4	1155
Shekel 7	4	$[0, 10]^4$	0	990	0	870	6	8550
Shekel 10	4	$[0, 10]^4$	0	375	0	1045	3	305

Table I. Dimensions, box bounds and results with SNOBFIT for the unperturbed and perturbed problems

In the case of the unperturbed problem ($\sigma = 0$), the results are competitive with results of noise-free global optimization algorithms (see [Jones et al. 1993] and [Huyer and Neumaier 1999]). For the perturbed problem, the algorithm sometimes

gets stuck in a nonglobal local minimizer for the Shekel m functions, which have m local minimizers and only one of them is global.

An earlier version of SNOBFIT was used successfully for a real life constrained optimization application involving the calibration of nanoscale etching equipment with expensive measured function values, in which most of the problems mentioned in the introduction were present.

The MATLAB code of the version of SNOBFIT described in this paper is available on the web at `http://www.univie.ac.at/~neum/software/snobfit/`.

ACKNOWLEDGMENT

The major part of the SNOBFIT package was developed within a project sponsored by IMS Ionen Mikrofabrikations Systeme GmbH, Wien, whose support we gratefully acknowledge. We'd also like to thank Erich Dolejsi for his help with debugging the program

REFERENCES

ANDERSON, E. AND FERRIS, M. 2001. A direct search algorithm for optimization of expensive functions by surrogates. *SIAM J. Optim 11*, 837–857.

BOOKER, A., J.E. DENNIS, J., FRANK, P., SERAFINI, D., TORCZON, V., AND TROSSET, M. 1999. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optim 17*, 1–13.

CARTER, R., GABLONSKY, J., PATRICK, A., KELLEY, C., AND ESLINGER, O. 2001. Algorithms for noisy problems in gas transmission pipeline optimization. *Optim. Eng. 2*, 139–157.

CHOI, T. AND KELLEY, C. 2000. Superlinear convergence and implicit filtering. *SIAM J. Optim. 10*, 1149–1162.

CONN, A., SCHEINBERG, K., AND TOINT, P. 1997. Recent progress in unconstrained nonlinear optimization without derivatives. *Math. Programming 79B*, 397–414.

DALLWIG, S., NEUMAIER, A., AND SCHICHL, H. 1997. GLOPT – a program for constrained global optimization. In *Developments in Global Optimization*, I. M. Bomze, T. Csendes, R. Horst, and P. M. Pardalos, Eds. Nonconves Optimization and its Applications 18. Kluwer, Dordrecht, 19–36.

ELSTER, C. AND NEUMAIER, A. 1995. A grid algorithm for bound constrained optimization of noisy functions. *IMA J. Numer. Anal. 15*, 585–608.

FIACCO, A. AND MCCORMICK, G. 1990. *Sequential Unconstrained Minimization Techniques*. Classics in Applied Mathematics 4. SIAM, Philadelphia.

HUYER, W. AND NEUMAIER, A. 1999. Global optimization by multilevel coordinate search. *J. Global Optim. 14*, 331–355.

JONES, D. 2001. A taxonomy of global optimization methods based on response surfaces. *J. Global Optim. 21*, 345–383.

JONES, D., PERTTUNEN, C., AND STUCKMAN, B. 1993. Lipschitzian optimization without the lipschitz constant. *J. Optim. Theory Appl. 79*, 157–181.

JONES, D., SCHONLAU, M., AND WELCH, W. 1998. Efficient global optimization of expensive black-box functions. *J. Global Optim. 13*, 455–492.

MCKAY, M., BECKMAN, R., AND CONOVER, W. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 239–245.

NEUMAIER, A. 1998. MINQ: General definite and bound constrained indefinite quadratic programming. `http://www.mat.univie.ac.at/~neum/software/minq/`.

NOCEDAL, J. AND WRIGHT, S. 1999. *Numerical Optimization*. Springer Series in Operations Research. Springer, Berlin.

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

Owen, A. 1994. Lattice sampling revisited: Monte Carlo variance of means over randomized orthogonal arrays. *Ann. Stat. 22*, 930–945.

Powell, M. 1998. Direct search algorithms for optimization calculations. *Acta Numerica 7*, 287–336.

Sacks, J., Welch, W., Mitchell, T., and Wynn, H. 1989. Design and analysis of computer experiments. With comments and a rejoinder by the authors. *Stastist. Sci. 4*, 409–435.

Schonlau, M. 1997. Computer experiments and global optimization. Ph.D. thesis, University of Waterloo, Waterloo, Ontario, Canada.

Schonlau, M. 1997–2001. *SPACE Stochastic Process Analysis of Computer Experiments.* `http://www.schonlau.net`.

Schonlau, M., Welch, W., and Jones, D. 1998. Global versus local search in constrained optimization of computer models. In *New Developments and Applications in Experimental Design*, N. Flournoy, W. Rosenberger, and W. Wong, Eds. Institute of Mathematical Statistics Lecture Notes – Monograph Series 34. Institute of Mathematical Statistics, Hayward, CA, 11–25.

Tang, B. 1993. Orthogonal array-based latin hypercubes. *J. Amer. Statist. Assoc. 88*, 1392–1397.

Trosset, M. and Torczon, V. 1997. Numerical optimization using computer experiments. Tech. Rep. 97-38, ICASE.

Welch, W., Buck, R., Sacks, J., Wynn, H., Mitchell, T., and Morris, M. 1992. Screening, predicting, and computer experiments. *Technometrics 34*, 15–25.