Constructing large feasible suboptimal intervals for constrained nonlinear optimization*

Tibor Csendes†, Zelda B. Zabinsky‡, Birna P. Kristinsdottir‡

May 3, 2005

Abstract

An algorithm for finding a large feasible n-dimensional interval for constrained global optimization is presented. The n-dimensional interval is iteratively enlarged about a seed point while maintaining feasibility. An interval subdivision method may be used to check feasibility of the growing box. The resultant feasible interval is constrained to lie within a given level set, thus ensuring it is close to the optimum.

The ability to determine such a feasible interval is useful for exploring the neighbourhood of the optimum, and can be practically used in manufacturing considerations. The numerical properties of the algorithm are tested and demonstrated by an example problem.

Key words: Constrained Nonlinear Optimization, Sensitivity Analysis, Inclusion Function, Interval Arithmetic.

Running head: Constructing feasible suboptimal intervals.

AMS Subject Classifications: 90C30, 65K05.

*This work was supported by the Grants OTKA 2879/1991 and OTKA 2675/1991, and in part by NSF Grant DDM-9211001, the Boeing Commercial Airplane Company and the NASA-Langley Research Center.

†Kalmár Laboratory, József Attila University, Szeged, Hungary

‡Industrial Engineering Program, University of Washington, Seattle WA, USA

1 Introduction

1.1 Problem setting

Consider the nonlinear optimization problem (P)

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_j(x) \leq 0 \qquad j = 1, 2, \ldots, m, \end{array}$$

where $f(x) : I\!R^n \to I\!R$ and the constraint functions $g_j(x) : I\!R^n \to I\!R$ are continuous nonlinear functions, and n is the dimension of the problem. Let us denote the set of feasible points by A, that is $A := \{x \in I\!R^n : g_j(x) \leq 0$ for each $j = 1, 2, \ldots, m\}$. Also let x^* be an optimal solution for problem (P).

It may happen that the optimal solution x^*, or an approximation of it, is known, yet this result is not suitable for practical use. For example, consider an engineering design problem which is formulated as a constrained global optimization problem, see for example [8], [12] and [17]. It is possible that the optimal design cannot be reproduced exactly with current manufacturing processes, thus each variable has a specific manufacturing tolerance, $\delta > 0$. Since the optimal solution x^* may be on one or more active constraints, then the n-dimensional interval $[x_i^* - \delta, x_i^* + \delta]$ for $i = 1, 2, \ldots, n$ is not feasible. From a practical point of view, it is preferable to find a feasible suboptimal box instead of a single optimal point. Thus we seek a feasible n-dimensional interval X^* for which $g_j(x) \leq 0$, $j = 1, 2, \ldots, m$, for all $x \in X^*$. It is also desirable to have this feasible box as close to the optimum as possible. Thus we also impose the constraint $f(x) \leq f(x^*) + \varepsilon$ for some $\varepsilon > 0$ for all $x \in X^*$. Such an interval would also reflect the sensitivity of the objective function [3, 7, 14] because the size of the feasible box may vary as ε varies.

We restate our problem: find an n-dimensional interval X^* such that for all $x \in X^*$

$$f(x) \leq f_\varepsilon \equiv f(x^*) + \varepsilon, \quad \text{and} \tag{1}$$

$$g_j(x) \leq 0 \qquad \text{for } j = 1, 2 \ldots, m. \tag{2}$$

Methods discussed in earlier papers [4, 5, 6] study similar problems. In [4, 5] an interval method was introduced to find a bounding interval of the level set of an unconstrained nonlinear optimization problem. The algorithm converges to the smallest n-dimensional interval containing the specified level

set. Another technique [6] locates the boundary of a level set in a given direction. Kearfott discussed an interval branch and bound method for bound constrained global optimization [10]. In a recent study [16] the solution of a difficult nonlinear optimization problem (in an alternate form) was reported by using a customized interval subdivision scheme. These methods provide a guaranteed reliable solution, although they are computationally tractable. They are based on inclusion functions and interval arithmetic, which will be reviewed next.

1.2 Inclusion functions and interval arithmetic

Inclusion functions [1, 15] offer a theoretically reliable and computationally tractable means of locating a feasible suboptimal interval. An inclusion function provides more information over an interval than could be conveyed with independent real function evaluations. The inclusion function gives upper and lower bounds on the objective function over the specified interval. We next discuss background and notation for inclusion functions and interval arithmetic.

Let $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous function. The set of compact real intervals (pairs of reals a and b such that $a \leq b$) is denoted by I and the set of n-dimensional compact intervals is denoted as I^n. An n-dimensional interval $X \in I^n$ is a parallelepiped with sides parallel to the coordinate axes. In this text X will usually be referred to as a *box*. The *range* of $f(x)$ over an n-dimensional interval X is denoted as $\overline{f}(X)$ and defined as

$$\overline{f}(X) = \{f(x) : \ x \in X\}.$$

The function $F(X) : I^n \rightarrow I$ is called an *inclusion function* of $f(x)$, if $F(X) \supseteq \overline{f}(X)$ for every $X \in I^n$. The *width* of a one-dimensional interval $[a, b]$ is denoted as

$$w([a, b]) = b - a.$$

For an n-dimensional interval $X \in I^n$ the *width* is defined as the maximum length of the edge of the box, that is

$$w(X) = \max\{w(X_i) : \ i = 1, ..., n\}$$

where X_i is the i-th coordinate interval of X. We assume that all inclusion functions mentioned are *isotone*, which means, if $F : I^n \rightarrow I$ is an inclusion

function then

$$X \subseteq Y \text{ implies } F(X) \subseteq F(Y)$$

for all $X, Y \in I^n$. An inclusion function $F(X)$ is said to be of *order* $\alpha > 0$ if there is a real constant c such that

$$w(F(X)) - w(\bar{f}(X)) \leq cw(X)^\alpha$$

for every $X \in I^n$.

Finding the range for a function over an n-dimensional interval has in general the same complexity as the original optimization problem, because we have to find the extreme values of the function over the interval. By using interval arithmetic it is possible to find bounds on the function values more efficiently. The interval operations can be carried out using only real operations. For example, interval addition is defined as: $[a, b] + [c, d] = [a+c, b+d]$. If outside rounding is also applied, then the calculated resulting interval contains every real number that can be a result of the given function on real numbers inside the original intervals. For example, the function $f(x_1, x_2) = x_1 + x_2$ is continuous and $f(x) : I\!R^2 \to I\!R$. Now consider the 2-dimensional box $X = \{(x_1, x_2) : \ x_1 \in [a, b], x_2 \in [c, d]\}$. Then the inclusion function defined by interval arithmetic is $F(X) = [a, b] + [c, d] = [a+c, b+d]$ which gives precisely the range of $f(x)$. The technique of producing an inclusion function by substituting the real variables and operations by their interval equivalent is called *natural interval extension* [1, 15].

Many programming languages are now available that support interval datatypes with the corresponding operations and intrinsic functions [2, 9, 11, 13]. These programming environments provide a convenient access to inclusion functions (with automatic outside rounding), but one can also simulate interval operations and functions by subroutines in any algorithmic language. We used the natural interval extension to calculate the inclusion functions. For more information about inclusion functions and interval arithmetic, see [1, 15].

2 The algorithm and its convergence

2.1 The algorithm

The algorithm iteratively grows a box about a given seed point. A seed point x^{seed}, which lies interior to the region of feasibility and in the ε-level set must be provided to start the algorithm. Thus the seed point, x^{seed}, must satisfy the following conditions:

$$f(x^{\text{seed}}) < f_\varepsilon \text{ and} \tag{3}$$
$$g_j(x^{\text{seed}}) < 0 \text{ for each } j = 1, 2, \ldots, m. \tag{4}$$

This will imply that there exists a feasible box with a positive volume containing the seed point. It is possible to construct nonlinear optimization problems for which no proper seed point exists which satisfies (3) and (4). In such cases the search for feasible suboptimal boxes also makes no sense.

Seed points can be obtained in several ways. One way is to find an approximation of x^*, and if it is interior to the feasible region, use it as x^{seed}. If it lies on an active constraint, search along the normal of the active constraint to generate x^{seed}. Another way is to sample randomly (a normal distribution may be appropriate) around the optimal point until a feasible interior point is found. This may be used as x^{seed}. A slight variation would be to sample according to a uniform distribution in the interval hull of the feasible region intersected with the ε-level set. The first feasible point with objective function value less than f_ε would be used as x^{seed}.

We will call an interval X a *feasible interval around* x^{seed}, if $x^{\text{seed}} \in X$ and equations (1) and (2) are satisfied for all $x \in X$. An interval will be called *maximal regarding* x^{seed}, if it is a feasible interval around x^{seed}, and there is no other feasible interval that contains it. Note that there may exist many maximal feasible boxes around a seed point. Also note that two different seed points may be contained in the same maximal feasible box.

It may appear disturbing that there is not a unique relationship between x^{seed} and a maximal feasible box around x^{seed}. For an example, suppose the set of feasible points with objective function values less than or equal to f_ε is an ellipse. Suppose x^{seed} is interior to the ellipse. Then there is an infinite number of feasible boxes around x^{seed} which do not contain one another (see Figure 1). At some point, it may be interesting to find the maximal

feasible box around x^{seed} that has the largest volume. However at this point the algorithm does not find the largest volume box around a seed point, but simply a maximal feasible box. In our application to manufacturing tolerances this is not a disadvantage. It is more desirable to compare trade-offs between coordinate lengths of various maximal boxes, than it is to know the box with largest volume, as this has no meaning for the application.

We call an interval Y *strongly feasible*, if $f(y) < f_\varepsilon$, and $g_j(y) < 0$ for all $y \in Y$ $(j = 1, 2, \ldots, m)$.

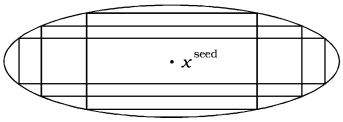

Figure 1. Illustration of multiple boxes within the ε-level set.

The main algorithm discussed in this paper is presented below. The algorithm uses parameters $d(i, 1)$ and $d(i, 2)$ for $i = 1, 2, \ldots, n$ and η, which are set at the beginning to positive reals. To start, $d(i, 1)$ and $d(i, 2)$ must be larger than η. The stopping criterion indicates that the algorithm should stop increasing the size of the actual box X, when the change along each coordinate is less than the threshold η in all directions.

Main algorithm

Step 0 Initialize interval vector $X_i = [x_i^{\text{seed}}, x_i^{\text{seed}}]$, and $d(i, j) \geq \eta > 0$ for all $i = 1, 2, \ldots, n$ and $j = 1, 2$.

Step 1 For $i = 1$ to n do:

Step 2 Set $Y_j = X_j$ for $j = 1, 2, \ldots, n$; $j \neq i$, and

$$Y_i = [\min(X_i) - d(i, 1), \min(X_i)].$$

Step 3 Use the checking routine to check whether $f(y) < f_\varepsilon$ and $g_j(y) < 0$ $(j = 1, 2, \ldots, m)$ for each $y \in Y$. If the answer is yes, then set $X = X \cup Y$. Otherwise $d(i, 1) = (\min(X_i) - \max(Z_i))/2$, where Z is the interval passed back by the checking routine as not strongly feasible.

Step 4 Set $Y_j = X_j$ for $j = 1, 2, \ldots, n$; $j \neq i$, and

$$Y_i = [\max(X_i), \max(X_i) + d(i, 2)].$$

Step 5 Use the checking routine to check whether $f(y) < f_\varepsilon$ and $g_j(y) < 0$ $(j = 1, 2, \ldots, m)$ for each $y \in Y$. If the answer is yes, then set $X = X \cup Y$. Otherwise $d(i, 2) = (\min(Z_i) - \max(X_i))/2$, where Z is the interval passed back by the checking routine as not strongly feasible.

Step 6 End of i-loop

Step 7 Stopping criterion: if the number of inclusion function calls is less than 100,000, and there is an $i = 1, 2, \ldots, n$ such that either $d(i, 1) \geq \eta$ or $d(i, 2) \geq \eta$ then go to **Step 1**.

Step 8 Print X, and STOP.

The core of the algorithm is the checking procedure called in Steps 3 and 5. This is a version of the interval subdivision method modified to check whether the actual box Y lies entirely in the region satisfying equations (1) and (2). The parameter θ is set to a small positive real value. The relation $d(i, j) \geq \theta$ must hold, or else the interval Y will always be rejected in Step 1. The checking procedure is in detail:

Checking routine

Step 0 Initialize the list L to be empty.

Step 1 If the width of Y is less than θ, then go to **Step 7**.

Step 2 Evaluate the inclusion functions $F(Y)$ and $G_j(Y)$ for each $j = 1, 2, \ldots, m$.

Step 3 If $\max F(Y) \geq f_\varepsilon$ or $\max G_j(Y) \geq 0$ for any $j = 1, 2, \ldots, m$, then go to **Step 5**.

Step 4 If the list L is empty, then go to **Step 6**, else put the last item of the list L into Y, delete this item from the list, and go to **Step 1**.

Step 5 Subdivide Y into subintervals U and V, set $Y = U$, put V into the list L as the last member, and go to **Step 1**. The subdivision should be made, such that the largest side of Y is halved.

Step 6 RETURN that the checked interval was strongly feasible.

Step 7 RETURN $Z = Y$, and that the checked interval was not strongly feasible.

If the checking routine indicates, that the checked interval is not strongly feasible, this means more precisely that a very small not strongly feasible subinterval was found. By properly setting θ, the place where the strong feasibility is violated can be located.

2.2 Convergence results

In this section the convergence properties of the algorithm introduced are investigated to provide theoretical background for the numerical implementations. First the checking routine is studied.

Lemma 1 *1. If the checking routine accepts an interval Y as strongly feasible, then $f(x) < f_\varepsilon$, and $g_j(x) < 0$ for each $x \in Y$, and $j = 1, 2, \ldots, m$.*

2. If the checking routine rejects an interval Y as not strongly feasible, then there exists a nested set of intervals, $Y = Y^1 \supset Y^2 \supset \ldots \supset Y^p$ generated by the routine, with the smallest interval having width less than θ, such that one of the conditions $\max F(Y^i) < f_\varepsilon$ or $\max G_j(Y^i) < 0$ was violated for each Y^i in the nested set of intervals, where $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, m$.

Proof. 1. If an interval Y is returned by the checking routine as a strongly feasible one, then Y must have been covered by subintervals $Y^1, Y^2, \ldots, Y^p$ in such a way that $w(Y^i) \geq \theta$, $\max F(Y^i) < f_\varepsilon$, and $\max G_j(Y^i) < 0$ for each $j = 1, 2, \ldots, m$, and $i = 1, 2, \ldots, p$.

If one of the conditions $\max F(X) < f_\varepsilon$ or $\max G_j(X) < 0$ was violated for an actual interval X, then X had to be divided. In this way only subintervals

of X can appear in the final set of covering intervals $Y^1, Y^2, \ldots, Y^p$, and not X itself. No point of X can be lost however during the subdivision procedure, since the union of the actual interval and the intervals on the list is decreased only in **Step 4** after finding the actual interval strongly feasible.

Consequently, for every $x \in Y$ there exists a Y^i $(i = 1, 2, \ldots, p)$ such that $x \in Y^i$,

$$f(x) \leq \max F(Y^i) < f_\varepsilon, \text{ and}$$

$$g_j(x) \leq \max G_j(Y^i) < 0$$

for each $j = 1, 2, \ldots m$.

2. The proof follows immediately from the subdivision procedure in the checking routine. □

Notice that only the inclusion property of F and G_j was utilised in the proof of Lemma 1, and no further requirement (like isotonicity or convergence order of the inclusion functions involved) was necessary.

Lemma 2 *Assume that*

$$w(F(X)) \to 0 \text{ as } w(X) \to 0, \text{ and} \tag{5}$$

$$w(G_j(X)) \to 0 \text{ as } w(X) \to 0 \tag{6}$$

for all $j = 1, 2, \ldots, m$.

1. If $f(x) < f_\varepsilon$ *and* $g_j(x) < 0$ *for every* $x \in Y$ *and* $j = 1, 2, \ldots, m$, *then there exists a threshold value* $\theta^{\mathrm{T}} > 0$ *such that for all* θ: $0 < \theta < \theta^{\mathrm{T}}$ *the checking routine stops after a finite number of iteration steps and it states that* Y *is strongly feasible.*

2. If $\theta > 0$ *and there is a point* $x \in Y$ *such that* $f(x) \geq f_\varepsilon$ *or* $g_j(x) \geq 0$ *for any* $j = 1, 2, \ldots, m$, *then the checking routine will stop after a finite number of iteration steps and it states that* Y *is not strongly feasible.*

Proof. 1. Assume that the checking routine does not stop in a finite number of steps. Then there must be an infinite embedded subsequence X^k, $k = 1, 2, \ldots$ of generated intervals inside Y. Denote the condensation point of the subsequence by x. The assumed properties (5) and (6) imply that $\lim_{k=1}^{\infty} \max F(X^k)$ is equal to $f(x)$, which is less than f_ε. Similarly $\lim_{k=1}^{\infty} \max G_j(X^k)$ is equal to $g_j(x) < 0$ for each $j = 1, 2, \ldots, m$. These

facts, however, contradict the infinity of the sequence, since each member of it should have been generated in **Step 5** of the checking routine after finding them not strongly feasible in **Step 3**. Consequently, there exists a finite set of strongly feasible boxes covering Y. With θ^{T} equal to the smallest width of the boxes, the statement is proven.

2. We prove only the case when $f(x) \geq f_\varepsilon$ for an $x \in Y$. The proof is similar when $g_j(x) \geq 0$ for any $j = 1, 2, \ldots, m$. The inclusion property ensures that $\max F(X) \geq f_\varepsilon$ for all intervals X for which $x \in X$. The conditions in **Step 3** of the checking routine are then fulfilled for all intervals containing x, and those intervals are then subdivided in **Step 5**. The interval Y cannot be covered by a finite number of strongly feasible subintervals (cf. Lemma 1), and hence the width of the actual subinterval would converge to zero. This procedure is terminated when the width of the actual interval becomes less than θ, and the interval Y is announced to be not strongly feasible. □

Notice that Lemma 2 ensures that a not strongly feasible interval Y will always be detected in a finite number of steps. However, it is possible that a strongly feasible interval will be mistaken as not strongly feasible if θ is too large. Thus it is important that θ be chosen with care.

Consider, again, a fixed constrained nonlinear optimization problem as given in section 1.1. Denote the result box calculated with the algorithm parameters θ and η by $X^*_{\theta,\eta}$, and the level set belonging to the function value f_ε by S_{f_ε}. The following theorems characterize the convergence properties of our algorithm:

Theorem 1 *If the set $S_{f_\varepsilon} \cap A$ is bounded, the seed point x^{seed} fulfils the conditions (3) and (4), $d(i,j) > 0$, and the properties (5) and (6) hold for the inclusion functions $F(X)$ and $G(X)$, then there exist threshold values $\theta^{\mathrm{T}} > 0$ and $\eta^{\mathrm{T}} > 0$ such that for all θ: $0 < \theta < \theta^{\mathrm{T}}$ and η: $0 < \eta < \eta^{\mathrm{T}}$*

1. *the algorithm stops after a finite number of steps,*
2. *the result box $X^*_{\theta,\eta}$ has a positive measure, and*
3. *the result interval $X^*_{\theta,\eta}$ is strongly feasible, $X^*_{\theta,\eta} \subset S_{f_\varepsilon} \cap A$.*

Proof. 1. The number of checking routine iteration steps is determined by Lemma 2 stating that with sufficiently small positive θ and η parameters, a finite number of iteration steps are needed to decide the strong feasibility of a given interval. The question remains then whether the checking

routine is called only a finite number of times to fulfil the stopping conditions of the main algorithm. The actual interval X of the main algorithm grows monotonically, the step sizes $d(i,1)$, $d(i,2)$ $(i = 1, 2, \ldots, n)$ decrease monotonically, and at each iteration at least one step size has to be greater then η to continue the procedure. On the other hand, there exists a bounding box $X^{\max}$ for the bounded set $S_{f_\varepsilon} \cap A$. This interval contains all the actual boxes of our algorithm: $X \subseteq X^{\max}$. Hence the number of iterations of the main algorithm in which the actual box X grows must be finite (at most $2n \max_i w(|X_i^{\max} - x_i^{\text{seed}}|)/\eta$). This fact implies that the number of checking routine calls in which the subinterval is found to be strongly feasible (accepting calls) have to be finite.

Now the number of rejecting calls (which indicated the given subinterval was not strongly feasible) of the checking routine must also be shown to be finite. After each rejecting call, the stepsize is decreased to at most the half of its previous value. The stepsize can only be decreased in a finite number of times, because it must remain greater than the algorithm parameter η. Hence the number of accepting and rejecting calls of the checking routine, and thus also the number of objective and constraint function evaluations, must be finite.

2. Assume that the result box $\overline{X}$ of our algorithm is of zero measure. Then, at least in the direction of one coordinate axis, the width of $\overline{X}$ must be zero. Let this direction be the one parallel to the x_k coordinate axis. Now the result box $\overline{X}$ is also strongly feasible, because the seed point x^{seed} and all intermediate intervals added to form $\overline{X}$, are strongly feasible. Due to the continuity of the functions $f(x)$ and $g_j(x)$ $(j = 1, 2, \ldots, m)$ there exists a positive number c such that the intervals $X' = (\overline{X}_1, \ldots, [\min \overline{X}_k - c, \min \overline{X}_k], \ldots, \overline{X}_n)$ and $X'' = (\overline{X}_1, \ldots, [\max \overline{X}_k, \max \overline{X}_k + c], \ldots, \overline{X}_n)$ are both strongly feasible. According to Lemma 2, both intervals X' and X'' are accepted as strongly feasible by the checking routine with a sufficiently small positive parameter θ. Hence, with parameter $0 < \eta < c$, the result interval has a positive width in the direction of the x_k coordinate axis. This conveys the proof of the positive measure of the result interval.

3. The result interval $X^*_{\theta,\eta}$ contains a finite number of subintervals accepted by the checking routine. Thus, according to Lemma 1, $X^*_{\theta,\eta}$ must be strongly feasible. □

Notice that the strong feasibility of the accepted intervals was utilised only in proving the positive volume of the result intervals. With the exception of this, the convergence results remain valid if the checking routine accepts feasible intervals.

Theorem 2 describes the limit of the result boxes when the algorithm parameters θ and η are equal and converge together to zero.

Theorem 2 *If the conditions of Theorem 1 are fulfilled, then the limiting interval* $X^* = \lim_{\theta \to 0} X^*_{\theta,\theta}$ *exists, and* X^* *is maximal in the sense that for every box* X' *the relations* $X^* \subseteq X'$ *and* $X' \subseteq S_{f_\varepsilon} \cap A$ *imply* $X' = X^*$.

Proof. The boundedness of the intervals $X^*_{\theta,\eta}$ and the monotonicity $X^*_{\theta',\eta'} \subseteq X^*_{\theta'',\eta''}$ if $\theta' > \theta'' > 0$ and $\eta' > \eta'' > 0$ imply the existence of the limiting interval X^*.

Assume that there exists an interval X' such that $X^* \subset X'$ and $X' \subseteq S_{f_\varepsilon} \cap A$. Then $\min X'_i < \min X^*_i$ or $\max X^*_i < \max X'_i$ must be true for an $i = 1, 2, \ldots, n$. For both cases, there exists an interval Y such that $X^* \cap Y$ is a side of X^*, $Y \subset X'$, and Y is of positive measure. Lemma 2 ensures that there exists a positive θ^{T} such that for every positive $\theta < \theta^{\mathrm{T}}$ the box Y and all of its subintervals are accepted as strongly feasible. Thus with $\eta^{\mathrm{T}} = \min_i w(Y_i)$, and a θ that approaches zero, the algorithm accepts Y, which is contradictory. □

The limiting interval X^* is not necessarily strongly feasible. For example, if $S_{f_\varepsilon} \cap A$ is an n-dimensional interval, then this may be a limiting interval of a sequence of strongly feasible result intervals.

Theorem 1 suggests that for a problem satisfying its conditions, sufficiently small positive θ and η values ensure a positive measure result interval in a finite number of iteration steps, i.e. after a finite number of objective and constraint function calls. Theorem 2 gives the basis that with θ and η values close to the machine precision one may obtain a closely maximal result box. It has to be stressed that beyond the given algorithm many others can be given for the same problem, and that it is a very difficult problem to find a maximal volume feasible interval (equivalent to a global optimization problem cf. [5]). In general, the location of a maximal volume feasible interval can only be solved with a certain kind of backtracking.

3 Numerical testing and examples

Consider the following simple constrained quadratic problem to illustrate how the algorithm discussed above proceeds. Let

$$f(x) = x_1^2 + x_2^2,$$

$$g_1(x) = (3 - x_1)^2 + (3 - x_2)^2 - 18, \text{ and}$$
$$g_2(x) = 1 - (2 - x_1)^2 - (2 - x_2)^2.$$

The set of feasible points A is now the circle C_1 with center at $(3, 3)$ and with a radius of $3\sqrt{2}$ with the exceptions of the points of the circle C_2 with center $(2, 2)$, and radius 1. The only global optimal point is at the origin, and the optimal function value is $f^* = f(0, 0) = 0$. The level sets S_{f_ε} are circles around the origin with radii of $\sqrt{f_\varepsilon}$, respectively. The constraint $g_1(x) \leq 0$ is active at the global minimum, and its normal is parallel to the line $x_1 = x_2$.

The inclusion functions are generated by natural interval extension:

$$F(X) = X_1^2 + X_2^2,$$

$$G_1(X) = (3 - X_1)^2 + (3 - X_2)^2 - 18, \text{ and}$$
$$G_2(X) = 1 - (2 - X_1)^2 - (2 - X_2)^2.$$

The capital letters denote again intervals with the subscript indicating coordinate direction. These inclusion functions are exact in the sense that the so-called excess width (defined by $w(F(X)) - w(\overline{f}(X))$) is zero for every argument interval. It is unfortunately not typical for interval calculations, yet it makes the demonstration of the working of the algorithm more transparent.

3.1 Assuming exact arithmetic

Set the seed point to $x^{\text{seed}} = (0.5, 0.5)^T$. The conditions (3) and (4) are now fulfilled for each $f_\varepsilon > 0.5$:

$$g_1(x^{\text{seed}}) = -5.5 < 0,$$

$$g_2(x^{\text{seed}}) = -3.5 < 0,$$
$$f(x^{\text{seed}}) = 0.5 < f_\varepsilon.$$

Choose the algorithm parameters $d(i,1) = d(i,2) = 0.1$ for $i = 1, 2$ and $\eta = 0.01$. For $f_\varepsilon = 2.0$ the intersection set of feasible points and the level set S_{f_ε} is the intersection of the circles with centres (0, 0) and (3, 3), and with radii $\sqrt{2}$ and $3\sqrt{2}$, respectively. It is a convex set, and the maximal volume inscribed box is $X_1^* = [0.0, 1.0]$, $X_2^* = [0.0, 1.0]$.

The starting interval is set to $X_1 = [0.5, 0.5]$ and $X_2 = [0.5, 0.5]$. The first check is made on the interval $Y_1 = [0.4, 0.5]$, $Y_2 = [0.5, 0.5]$. The corresponding inclusion function values are

$$F(Y) = [0.4, 0.5]^2 + [0.5, 0.5]^2 = [0.16, 0.25] + [0.25, 0.25] = [0.41, 0.5],$$

$$G_1(Y) = [6.25, 6.76] + [6.25, 6.25] - 18 = [-5.5, -4.99]$$

and

$$G_2(Y) = 1 - [2.25, 2.56] - [2.25, 2.25] = [-3.81, -3.5].$$

The checking routine returns thus that Y is strongly feasible, and X is set in **Step 3** of the main algorithm to $([0.4, 0.5], [0.5, 0.5])^T$.

The next check is then made in **Step 5** on the interval $Y_1 = [0.5, 0.6]$, $Y_2 = [0.5, 0.5]$. The corresponding inclusion function values are $F(Y) = [0.5, 0.61]$, $G_1(Y) = [-5.99, -5.5]$ and $G_2(Y) = [-3.5, -3.21]$. The actual interval is then updated to $X_1 = [0.4, 0.6]$, $X_2 = [0.5, 0.5]$.

The actual interval is modified for $i = 2$ to $([0.4, 0.6], [0.4, 0.5])^T$, and then to $([0.4, 0.6], [0.4, 0.6])^T$. The sequence of actual intervals is as follows:

$$X = ([0.3, 0.7], [0.3, 0.7])^T,$$

$$X = ([0.2, 0.8], [0.2, 0.8])^T,$$

$$X = ([0.1, 0.9], [0.1, 0.9])^T.$$

The interval X was obtained after 48 inclusion function evaluations. The calculation of inclusion functions involves on the average two times more computation than the corresponding real functions do. Until this point was reached, the checking routine accepted all the extension intervals Y immediately, without subdivision. Thus the value of the algorithm parameter θ had no effect on this part of the result. In the next iteration the checked intervals and the inclusion function values $F(Y)$ are as follows:

$$Y = ([0.9, 1.0], [0.1, 0.9])^T, \text{ and } F(Y) = [0.82, 1.81],$$

$$Y = ([0.0, 0.1], [0.1, 0.9])^T, \text{ and } F(Y) = [0.01, 0.82],$$
$$Y = ([0.0, 1.0], [0.9, 1.0])^T, \text{ and } F(Y) = [0.81, 2.00].$$

The last interval is not strongly feasible, and a new $d(2,1) < 0.05$ is determined (depending on the value of θ). Then

$$Y = ([0.0, 1.0], [0.0, 0.1])^T, \text{ and } F(Y) = [0.00, 1.01],$$

and in this way $X = ([0.0, 1.0], [0.0, 0.9])^T$.

With further calculations this actual interval may be refined to obtain a maximal box X^*. We have a computational proof that each point x of the actual boxes and the result interval is feasible, and $f(x) < f_\varepsilon$.

3.2 Computer implementation with outside rounding

The main difference between the results of section 3.1 and those obtained by the computer program is that the latter is produced by operations with outside rounding. For example, $Y = ([0.0, 1.0], [0.9, 1.0])^T$ would be found feasible (but not strongly feasible) calculating with exact arithmetic (since $\max F(Y) = 2.0$), while $\max F(Y) > 2.0$ if it is evaluated with outside rounding. This is the reason why the results in the first line of Table 1 may be slightly different from those discussed in 3.1. It is worth mentioning, that if the stopping condition would be based on the difference $(\min(X_i) - d(i,1)) - \min(X_i)$ then this value could also attain zero because of the computer representation of floating point numbers.

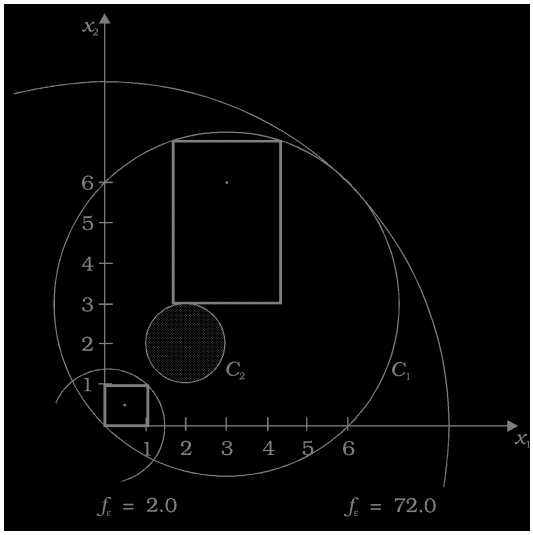

Figure 2. Test problem and result intervals of the first and last lines of Table 1.

Table 1. The role of the seed point in locating maximal feasible boxes.

x^{seed}	X^{res}	$\text{vol}(X^{\text{res}})$	NFE
$(0.50, 0.50)^T$	$([-0.000028, 1.046680], [0.000098, 0.951000])^T$	0.99532	1822
$(0.10, 0.10)^T$	$([-0.000056, 1.000066], [0.000069, 0.999902])^T$	0.99996	1945
$(0.01, 0.01)^T$	$([-0.003131, 1.034968], [0.003153, 0.963763])^T$	0.99721	2065
$(0.90, 0.90)^T$	$([-0.000028, 1.000160], [0.000098, 0.999805])^T$	0.99989	2118
$(0.10, 0.90)^T$	$([-0.400226, 0.556738], [0.462630, 1.300000])^T$	0.80133	1610
$(0.00, 1.00)^T$	$([-0.425529, 0.522607], [0.496856, 1.314084])^T$	0.77484	1669
$(-0.01, 0.10)^T$	$([-0.072507, 0.990052], [0.074328, 1.009826])^T$	0.99402	1996
$(4.0, 4.0)^T$	$([2.662546, 6.048340], [2.749031, 5.9508304)^T$	10.841	3015
$(5.0, 5.0)^T$	$([2.600000, 6.048340], [2.800000, 5.9508246)^T$	10.865	2677
$(3.0, 6.0)^T$	$([1.732192, 4.267773], [3.000000, 7.0487793)^T$	10.266	2801

Table 1 contains details of the results on the numerical test that examines how the place of the seed point affects the result box constructed by the program. In the following, we use only one initial value for all $d(i, j)$ step sizes ($i = 1, 2$; $j = 1, 2$). All of the problem and algorithm parameters were constant during this test ($d(i, j) = 0.1$ for $i = 1, 2$ and $j = 1, 2$, $\eta = 0.0001$ and $\theta = 0.0001$), only x^{seed} and f_ε was changed. The latter was 2.0 for the first 7 lines and 72.0 for the last three. For the problem specified by $f_\varepsilon = 2.0$, the maximal volume feasible box is $X^* = [0.0, 1.0]^2$. The result interval calculated by the program is denoted by X^{res}, and $\text{vol}(X^{\text{res}})$ is its volume. The latter is found close to one (it is, of course, not greater than $\text{vol}(X^*) = 1$). NFE stands for the number of function ($F(X)$ and $G_j(X)$ for $j = 1, 2, \ldots, m$) evaluations.

The test results presented in Table 1 suggest that the seed point may be chosen close to the normal of the active constraint at x^*, even if it is outside of X^*. It is interesting that the center of the maximal volume inscribed box is not an optimal seed point. It also worth mentioning that in the first 7 lines (where $S_{f_\varepsilon} \cap A$ is symmetric for the $x_1 = x_2$ line) the first component of the result interval is always wider than the second one. The explanation for it is that the actual interval is always enlarged first along the first coordinate direction. Two result boxes are shown on Figure 2 together with the constraint functions and the corresponding levels.

The algorithm stops if each subinterval to be checked is thinner than η in the actual direction. A small η may ensure that the growing of the

Table 2. The effects of algorithm parameters η and θ on the volume of the result box and on the number of function evaluations.

η	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
θ	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
$\mathrm{vol}(X^{\mathrm{res}})$	0.90000	0.99542	0.99542	0.99542	0.99542	0.99542
NFE	480	1550	2546	3171	3921	4182

η	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
θ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$\mathrm{vol}(X^{\mathrm{res}})$	0.97134	0.98906	0.99529	0.99541	0.99542	0.99542
NFE	376	762	1009	1303	1550	1836

η	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
θ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$\mathrm{vol}(X^{\mathrm{res}})$	0.97134	0.99430	0.99532	0.99541	0.99542	0.99542
NFE	376	956	1882	3081	4182	6332

actual box is not stopped too early when further increase is still possible. The procedure parameter θ gives the width of the interval that should be regarded as hopeless for further subdivision. A smaller θ means more function evaluations, but also the recognition of a not strongly feasible subinterval will be more reliable. Thus both η and θ help determine when the algorithm stops. Table 2 shows the combined effect of these parameters on the volume of the result box and the number of function evaluations required. The problem setting was the same as in Table 1 ($f_\varepsilon = 2.0$), except the seed point was always $x^{\mathrm{seed}} = (0.5, 0.5)^T$.

As it can be seen in Table 2, the total number of function evaluations NFE is growing only slightly with the decrease of θ. This phenomenon is in a nice accordance with the findings of [6]. No $\theta \geq 0.1$ value was studied, since then no increase of the initial box can be achieved (cf. the comments after the checking routine). Similarly, a setting $\eta > 0.1$ does not make sense, since then the algorithm stops after the first iteration regardless of the given problem. According to the test results comprised in Table 2, the algorithm parameters should be set to small values ensuring relatively large result boxes at moderate computational costs.

Table 3. The effect of the initial step size $d(i,j)$ on the volume of the result box and on the number of function evaluations.

$d(i,j)$	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}
$\mathrm{vol}(X^{\mathrm{res}})$	0.91723	0.99666	0.99542	0.99995	0.9999977	1.00000
NFE	5290	4949	4128	3525	8245	61618

Table 3 shows the results of the numerical test that examines how the algorithm parameter $d(i,j)$ affects the result box constructed by the program and the number of function evaluations. All other problem and algorithm parameters ($f_\varepsilon = 2.0$, $x^{\mathrm{seed}} = (0.5, 0.5)^T$, $\eta = 10^{-6}$ and $\theta = 10^{-6}$) were the same for the runs of this test.

According to the test results presented in Table 3, the choosing of the initial step size has a major impact on the volume of the result box, and — at least in this particular case — the maximal volume feasible box can well be approximated with a suitable $d(i,j)$. In a practical problem, however, it may be difficult to set the initial step size to obtain a large result box efficiently.

For practical problems, our method is used with given approximations of the global minimum and the global minimum point. Table 4 shows some sample runs demonstrating the results of the algorithm when multiple levels are used to highlight the relation between f_ε and the tolerances. The seed points were simply chosen as feasible points along the normal to the active constraint. The initial step sizes were set to about two orders of magnitude smaller than the anticipated width of $S_{f_\varepsilon} \cap A$ in every direction.

For $f_\varepsilon = 100.0$ and $f_\varepsilon = 10.0$ the respective feasibility sets ($S_{f_\varepsilon} \cap A$) are not convex. The growing of the actual box is stopped only by the constraints $G_j(X) \leq 0$ $(j = 1, 2)$ and not by $F(X) \leq f_\varepsilon$. This is why the first two rows in Table 4 give the same results. Otherwise the result set seemingly approximates the global minimum point x^* as f_ε approximates the global minimum f^*. There is an obvious trade-off between the degree of suboptimality characterized by the level f_ε and the parameter tolerances (the width of the result intervals). The algorithm is also capable of giving the lower and upper bounds for the function values of points inside the result set X^{res}, which bounds may be much tighter than $[f^*, f_\varepsilon]$.

The presented algorithm was also applied to a real life engineering design

Table 4. Sample runs with different levels approximating the optimal value.

f_ε	$d(i,j)$	x^{seed}	X^{res}	vol(X^{res})	NFE
100.0	1.0	1.0	[-0.0000003, 1.9999981]	1.999996	5787
		1.0	[0.0000010, 1.0000000]		
10.0	1.0	1.0	[-0.0000003, 1.9999981]	1.999996	5787
		1.0	[0.0000010, 1.0000000]		
2.0	10^{-2}	10^{-1}	[-0.0000005, 1.0049500]	0.999950	3939
		10^{-1}	[0.0000006, 0.9950249]		
1.0	10^{-2}	10^{-1}	[-0.0000005, 0.7113538]	0.499963	4152
		10^{-1}	[0.0000006, 0.7028336]		
10^{-1}	10^{-3}	10^{-2}	[-0.0004998, 0.2240527]	0.049999	4569
		10^{-2}	[0.0004999, 0.2231596]		
10^{-2}	10^{-4}	10^{-3}	[-0.0000496, 0.0707105]	0.0049999	9569
		10^{-3}	[0.0000504, 0.0707104]		
10^{-3}	10^{-5}	10^{-4}	[-0.0000047, 0.0223600]	0.00049996	27258
		10^{-4}	[0.0000531, 0.0223600]		

problem to construct manufacturing tolerances for an optimal design of composite materials [17] that motivated our study. The numerical experiences on this composite laminate design problem will be reported in a forthcoming paper.

References

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Calculations* (Academic Press, New York, 1983).

[2] J.H. Bleher, S.M. Rump, U. Kulisch, M. Metzger, Ch. Ullrich and W. Walter, FORTRAN-SC. A study of a FORTRAN extension for engineering/scientific computation with access to ACRITH, Computing 39(1987) 93-110.

[3] J.D. Buys and F. Gonin, The use of Augmented Lagrangian functions for sensitivity analysis in nonlinear programming, Mathematical Programming 12(1977) 281-284.

[4] T. Csendes, An interval method for bounding level sets of parameter estimation problems, Computing 41(1989) 75-86.

[5] T. Csendes, Interval method for bounding level sets: revisited and tested with global optimization problems, BIT 30(1990) 650-657.

[6] T. Csendes and J. Pintér, A new interval method for locating the boundary of level sets, Int. J. of Computer Mathematics 46(1993) in press.

[7] A.V. Fiacco, Sensitivity analysis for nonlinear programming using penalty methods, Mathematical Programming 12(1976) 287-311.

[8] C.A. Floudas and P.M. Pardalos, *A collection of test problems for constrained global optimization algorithms* (Lecture Notes in Computer Science Vol. 455, Springer-Verlag, New York, 1990).

[9] H.-P. Jüllig, BIBINS/2.0 — C++ Bibliotheken für Vektoren und Matrizen über beliebigem skalaren Datentyp unter Berücksichtigung spezieller spärlicher Strukturen sowie Intervalldatentypen, Bericht 92.6, Technical University of Hamburg-Harburg (1992).

[10] R.B. Kearfott, An interval branch and bound algorithm for bound constrained optimization problems, J. Global Optimization 2(1992) 259-280.

[11] R. Klatte, U. Kulisch, M. Neaga, D. Ratz and Ch. Ullrich, *PASCAL-XSC — Language Reference with Examples* (Springer-Verlag, Berlin, 1992).

[12] Z. Kovács, F. Friedler and L.T. Fan, Recycling in a separation process structure, AIChE J. (1993) accepted for publication.

[13] Ch. Lawo, C-XSC — A Programming Environment for Verified Scientific Computing and Numerical Data Processing, Manuscript, University of Karlsruhe (1991).

[14] K.R. Lutchen and A.C. Jackson, Confidence bounds on respiratory mechanical properties estimated from transfer versus input impedance in humans versus dogs, IEEE T. on Biomedical Engineering 39(1992) 644-651.

[15] H. Ratschek and J. Rokne, *Computer Methods for the Range of Functions* (Ellis Horwood, Chichester, 1984).

[16] H. Ratschek and J. Rokne, A circuit design problem, Paper presented at the II. Workshop on Global Optimization, Sopron, Hungary, 1990. (to be published in the J. Global Optimization)

[17] Z.B. Zabinsky, D.L. Graesser, M.E. Tuttle and G.I. Kim, Global optimization of composite laminates using Improving Hit-and-Run, in: Recent Advances in Global Optimization, Princeton University Press, Princeton (1992) p. 343-368.