What Can Interval Analysis Do for Global Optimization?

H. RATSCHEK and R. L. VOLLER
Mathematisches Institut, Heinrich Heine Universität, 4000 Düsseldorf, Germany

(Received: 10 September 1990; accepted: 11 March 1991)

Abstract. An overview of interval arithmetical tools and basic techniques is presented that can be used to construct deterministic global optimization algorithms. These tools are applicable to unconstrained and constrained optimization as well as to nonsmooth optimization and to problems over unbounded domains. Since almost all interval based global optimization algorithms use branch-and-bound methods with iterated bisection of the problem domain we also embed our overview in such a setting.

AMS 1980 Subject Classification. 65K05, 65G10, 90C30.

Key words. Global optimization, interval analysis.

1. Introduction

Let I be the set of real compact intervals, R be the set of reals, $X = R^m$ or $X \in I^m$ be a box (precisely, a parallelepiped), and $f: X \to R$ be the objective function. Let f^* be the global minimum of f over X if it does exist and X^* be the set of global minimizers of f over X. The determination of f^* or X^* or a subset of X^* is called the *unconstrained optimization problem* (in the global sense). If X is of a more general shape, usually defined by constraint functions, the determination of f^*, etc. is called the *constrained optimization problem* (in the global sense). We also call f^* or X^* the *solution set* of our problem.

Solving a global optimization problem requires the comparison of a continuum of values and to pick out just the elements of the solution set. Since interval computation is a means for handling continua, it provides competitive methods for tackling global problems.

The first interval techniques for treating general global problems were established by Moore [28], [29], [31], Skelboe [42], Hansen [9], [10], Stroem [43], Dussel [7], Caprani and Madsen [4], Ichida and Fujii [16], Mancini [22], Mancini and McCormick [23], Mancini and Wilde [24], [25], Oelschlägel and Süsse [34], Süsse [44], Robinson [40], etc. Although some of these references were focusing on special problems like convex or signomial programming they were providing concepts which could give insight into more general problems and be applied to them.

Journal of Global Optimization **1**: 111–130, 1991

The overview, we provide in this paper, can only cast a quick glance at the various topics that are to be enumerated. Their thorough investigation, however, may be found in Mohd [27], Ratschek and Rokne [38], Hansen [11].

The interval techniques we want to discuss may best be demonstrated within a branch-and-bound setting, which involves an iterated bisection of the domain, X. A related prototype algorithm is presented in Section 2. In Section 3, the basic features of interval computation, that is, exact and machine interval arithmetic are explained. Inclusion functions are developed in Section 4. They are interval valued functions that enable to include real valued functions. Practically, is a real-valued function f given, inclusion functions of f the width of them is as narrow as required can be constructed without too much effort. This point is especially important for convergence properties. In Section 5, the tools for determining the solution set directly, such as midpoint test, monotonicity test, etc. are discussed. In Section 6, an interval Newton-like method is sketched which is to be applied to the gradient of f or to the Lagrangian function in order to find the stationary or Kuhn–Tucker points of the problem. In Section 7, the convergence properties of the prototype algorithm are employed. In Section 8, termination criteria are addressed. Problems over unbounded domains are found in Section 9, and in Section 10 one can see how interval tools fit into nonsmooth optimization strategies. Finally, the constrained problem is discussed in Section 11.

There remain a few interval approaches to optimization that do not fit into the prototype based frame of our paper but are, nevertheless, worth of mentioning. Let us, for example, pick out two of them: In Mancini and Wilde [24], [25], interval tools are not only connected with primal problems but also with dual ones. Another idea is taken up by Dixon and Fitzharris [6] who use interval arithmetic in order to stabilize conjugate gradient methods for minimizing quadratic forms.

2. A Prototype Algorithm

This extremely simplified prototype we present is to be seen rather as a suitable foothold for describing the various interval tools and their interaction with the optimization problem. If the reader wants to get to know really workable and ripened interval algorithm he is referred, for instance, to [10], [11], [12], [41].

The following algorithm for solving the optimization problem needs the domain for the optimization problem, $X \in I^m$, and the objective or cost function, $f: X \to R$, as input parameters. Although some epsilons are unavoidable in Step 5 (termination) they are marginal at the moment and suppressed. As X is a box, we first think of the unconstrained optimization problem, but we will use the same prototype for the constrained case in Section 11 as well as for the unbounded case in Section 9. The "solutions" that are addressed in Step 3 refer to elements of X^*, that is, to global minimizers. This prototype, however, can be used as a zero

finding algorithm for systems of nonlinear equations when "solution" means a zero of f in X, cf. Section 6.

THE PROTOTYPE ALGORITHM.

1. Put X into an ordered list $\mathscr{L}$ (which has been empty before).
2. Bisect the first box of $\mathscr{L}$ into two subboxes V_1 and V_2.
3. Delete V_i if it can be proven that V_i contains no solution or diminish V_i if it can be proven that a part of V_i contains no solution $(i = 1, 2)$.
4. Put V_i (as whole or diminished) into list $\mathscr{L}$ if V_i has not been deleted in Step 3 $(i = 1, 2)$.
5. Stop if termination criteria hold.
6. Goto Step 2.

As we shall see later, all the global minimizers will be contained in the boxes of the list, $\mathscr{L}$, at any stage of the computation. Hence, it will be intended to get the union of the boxes of $\mathscr{L}$ converging to X^*.

The steps of the Algorithm are very loose and offer a good deal of freedom. Rather different methods can be derived from the prototype depending on how the steps are completed. The most important are:

Step 2 (Bisection). In general, the box is subdivided by bisecting it at the longest edge. This procedure is a necessary condition for getting convergence to X^*, cf. Section 8. However, some authors provide a cyclic bisection where the bisection directions are exchanged cyclically ([31]). A general theory of bisection has been developed by Kearfott [18].

Step 3. How can be verified that V_1 or V_2 or parts of these boxes contain no solutions? It is one of the great strengths of interval arithmetic to be able to provided computationally executable ways of performing such tests. Their description will be one of the central topics of the overview (cf. Sections 4–6), the more as non-interval methods do not have many comparable tests, which usually depend on existence and knowledge of Lipschitz constants of the functions. It is typical for the interval based tests that they have some kind of failure rate, that decreases as the range of application of the test becomes smaller. Hence it can happen, that such a test fails for the box $V_1 \cup V_2$ but is successful for the subboxes V_1 and V_2.

Step 4. It is important at which positions and boxes V_1 and V_2 are put into the list. If they are put at the end of the list one gets an ordering of the *list by age* and has a *uniform subdivision* strategy apart from the different sizes of the boxes by step 3. Such an ordering implies convergence of the Algorithm to the solution set, see Section 7. It is also very widespread to relate with every box Y of $\mathscr{L}$ a lower bound of f over Y, which can be determined with interval arithmetic means. The boxes of $\mathscr{L}$ are then *ordered by increasing lower bounds*. In this case one gets a typical branch-and-bound mechanism (cf. [15]) which may be more effective than

the uniform subdivision but which may cause non-convergence under rare circumstances. A third strategy is the *ordering by short-livedness* where V_1 and V_2 enter the list at its head. This means that one box after the other is worked off. This procedure needs minimal storage.

Step 5. Interval arithmetic is an ideal tool for establishing various sorts of termination criteria which are mainly requirements for a prescribed accuracy of the computation of the global minimum value or the global minimizers. Therefore an error estimation is connected with the termination of the computation, cf. Section 8. Also uniqueness tests can be applied to the boxes of the final list, cf. Section 8.

3. Interval Arithmetic, Machine Interval Arithmetic

In this section, a minimum knowledge of interval arithmetic is imported. A thorough introduction to the whole area of interval arithmetic can be found in [31], [1], [3]. The development of interval tools appropriate for dealing with optimization problems is presented in [36], [38], [11].

Let R be the set of reals and I be the set of real compact intervals (these are the ones normally considered). Operations in I defined by the expression

$$A*B = \{a*b : a \in A, b \in B\} \text{ for } A, B \in I \tag{1}$$

where the symbol * stands for $+$, $-$, $\cdot$, and /, and where A/B is only defined if $0 \notin B$.

Definition (1) is motivated by the fact that the intervals A and B included some exact values, α and β, respectively, of the calculation. The values α and β are generally not known. The only information that is usually available consists of the including intervals A and B, i.e., $\alpha \in A$, $\beta \in B$. From (1), it follows that

$$\alpha * \beta \in A * B$$

which is called *inclusion principle of interval arithmetic*. Moreover, $A * B$ is the smallest set that contains the real $\alpha * \beta$.

The real and the corresponding interval operations are denoted by the same symbols. So-called *point intervals*, i.e., intervals consisting of exactly one point, $[a, a]$, are denoted by a. Expressions like Aa, $a + A$, A/a, $(-1)A$, etc., for $a \in R$, $A \in I$ are therefore defined. The expression $(-1)A$ is written as $-A$.

Definition (1) is equivalent to the following constructive rules:

$$[a, b] + [c, d] = [a + c, b + d],$$
$$[a, b] - [c, d] = [a - d, b - c],$$
$$[a, b][c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)],$$
$$[a, b]/[c, d] = [a, b][1/d, 1/c] \text{ if } 0 \notin [c, d].$$

These rules show that *subtraction and division in I are not the inverse operations* of

addition and multiplication, respectively, as is the case in R. For example, $[0,1]-[0,1]=[-1,1]$, $[1,2]/[1,2]=[1/2,2]$. This property is one of the main differences between interval arithmetic and real arithmetic. Another main difference is given by the socalled *subdistributive law*,

$$A(B+C)\subseteq AB+AC \text{ for } A,B,C\in I\,.$$

The *distributive law* is valid in some special cases, e.g.,

$$a(B+C)=aB+aC \text{ if } a\in R \text{ and } B,C\in I\,.$$

The following property follows directly from (1): Let $A, B, C, D\in I$, and $*$ be any interval operation; then

$$A\subseteq B,\ C\subseteq D \text{ implies } A*C\subseteq B*D \text{ (if } B*D \text{ is defined)}\,. \tag{2}$$

The last-mentioned property is called *inclusion isotony* of the interval operations. A summary of the algebraic behaviour of intervals is given in [35].

Machine interval arithmetic can be considered as an approximation of interval arithmetic on computer systems and is based on the inclusion isotony of the interval operations in the following manner: Let us assume again that α and β are the unknown exact values of any stage of the calculation and only including intervals are known, $\alpha\in A$, $\beta\in B$. Then A and B might not be representable on the machine. So, A and B are replaced by the smallest *machine intervals* that contain A and B, that is, $A\subseteq A_M$, $B\subseteq B_M$.

The left and right endpoints of a machine interval are machine numbers. (2) implies $A*B\subseteq A_M*B_M$.

The interval A_M*B_M need not be a machine interval and is therefore approximated by $(A_M*B_M)_M$, which is representable on the machine. This leads to the *inclusion principle of machine interval arithmetic*,

$$\alpha\in A,\ \beta\in B \text{ implies } \alpha*\beta\in(A_M*B_M)_M\,.$$

Thus, the basic principle of interval arithmetic is kept in machine interval arithmetic, i.e., the exact unknown result is contained in the corresponding known interval, and rounding errors are under control.

There are several software systems and software packages in which machine interval arithmetic is implemented, e.g., TRIPLEX-ALGOL-60, Pascal-SC, or ACRITH for some IBM computers, and ARITHMOS for some Siemens computers, FORTRAN-SC, etc. See, for instance, [21] as guide for further information.

4. Inclusion Functions and Natural Interval Extensions

Inclusion functions are a means for dealing with ranges of functions, in particular if the dependency of the range on the domain is to be investigated. Practically, inclusion functions are constructed via natural interval extensions almost automatically ([28]).

Let again $X \in I^m$ and $f: X \to R$. The set of compact intervals contained in X is denoted by $I(X)$. Let $\bar{f}(Y) = \{f(x): x \in Y\}$ for $Y \in I(X)$ be the *range* of f over Y. A function F is called an *inclusion function* for f if

$$\bar{f}(Y) \subseteq F(Y) \text{ for any } Y \in I(X) .$$

Inclusion functions for vector-valued functions are to be understood componentwise. Inclusion functions can be constructed in any programming language in which interval arithmetic is simulated or implemented, see the following paragraphs. F is called *isotone* if for $Y, Z \in I(X)$,

$$Y \subseteq Z \text{ implies } F(Y) \subseteq F(Z) .$$

Let g be any function pre-declared in some programming language (like sin, cos, exp etc.). Then the corresponding *predeclared interval function* G is defined by

$$G(Y) = \bar{g}(Y) \text{ for any } Y \in I \text{ contained in the domain of } g .$$

Since the monotonicity intervals of pre-declared functions g are well-known it is easy to realize the interval functions G on a computer. Nevertheless, the influence of rounding errors may be considered, so that $(G(Y_M))_M$ instead of $G(Y)$ will be computed on a machine.

Let $f(x)$ be any function expression in the variable $x \in R^m$. So, $f(x)$ may be an explicit formula or described by an algorithm not containing logical connectives. For simplicity, we assume that $f(x)$ is representable in a programming language. Let $Y \in I^m$ or let Y be an interval variable over I^m. Then the expression which arises if each occurrence of x in $f(x)$ is replaced by Y, if each occurrence of a pre-declared function g in $f(x)$ is replaced by G, and if the arithmetic operations in $f(x)$ are replaced by the corresponding interval arithmetic operations, is called the *natural interval extension* of $f(x)$ to Y, and it is denoted by $f(Y)$. Due to (1) we get the *inclusion principle for (programmable) functions*

$$a \in Y \text{ implies } f(a) \in f(Y) . \tag{3}$$

Therefore, $f(Y)$ seen as a function in Y is an inclusion function for the function $f(x)$. (Note: Natural interval extensions could be precisely defined only via recursion. Further, one would have to distinguish between the expressions $f(x)$ or $f(Y)$ and the functions defined by these expressions. One would also have to take care of the case that a forbidden division by 0 could be implied by the expression $f(Y)$. However, we chose this outline for simplicity.)

For example, if $f(x) = x_1 \sin x_2 - x_3$ for $x \in R^3$ then $f(Y) = Y_1 \sin Y_2 - Y_3$ is the natural interval extension of $f(x)$ to $Y \in I^3$.

Due to the algebraic properties of interval arithmetic, different expressions for a real function f can lead to interval expressions which are different as functions. For example, if $f_1(x) = x - x^2$ and $f_2(x) = x(1-x)$ for $x \in R$ then $f_1(Y) = Y - Y^2 = [-1, 1]$ and $f_2(Y) = Y(1-Y) = [0, 1]$ for $Y = [0, 1]$. For comparison,

$\bar{f}(Y) = [0, 1/4]$. In general, the problem arises as to how to find expressions of a given function that lead to natural interval extensions as good as possible. A partial solution to this problem can be found in [36].

A measure of the quality of an inclusion function F for $f: X \to R$ is the so-called *excess-width* ([28])

$$w(F(Y)) - w(\bar{f}(Y)) \text{ for all } Y \in I(X),$$

where $w([a, b]) = b - a$ is the width of an interval. F is called of *order* $\alpha > 0$ if

$$w(F(Y)) - w(\bar{f}(Y)) = O(w(Y^{\alpha})) \text{ for } Y \in I(X)$$

where the width of a box $Y = Y_1 \times \cdots \times Y_m$ is defined by $w(Y) = \max_{i=1,\ldots,m} w(Y_i)$. In order to obtain good computational results it is necessary to choose inclusion functions having as high an order α as possible, see for example, [36]. If f is programmable then, in general, the natural interval extension is of order 1 ([28], [36]). If f is differentiable and if $G^{(1)}$ is an inclusion function of ∇f satisfying $w[G^{(1)}(Y)] \leq Kw(Y)$ for $Y \in I(X)$ and some $K > 0$ then $F(Y) = f(c) + (Y - c)^T G^{(1)}(Y)$ where c is any point of Y is an inclusion function of order 2. It is called *mean value form* and obeys to the so-called class of *centered forms*, cf. [20], [36] for details.

From the definition, $F(Y)$ is a compact interval as long as $Y \in I(X)$. Hence, $F(Y)$ has two endpoints (boundaries) which will be denoted by $\min f(Y)$ and $\max F(Y)$.

5. Box-Discarding Tests

A point x^* is a global minimizer of f over $X \in I^m$ if $f(x^*) \leq f(x)$ for all $x \in X$. In general, it is not possible to replace this condition completely by local conditions as in the case with local minimizers or in the strictly convex case. Hence it is a basic feature of global methods to eliminate areas that can not contain solution points. In this section one can see how effective interval methods are for constructing such discarding processes.

The following tests are mainly due to [9], [10], but see also [31], [45], [22]. The following box-discarding tests are to be applied to the bisected subboxes $V = V_1, V_2$ of Step 3 of the Prototype Algorithm.

1. Midpoint Test. It is one of the simplest tests, if occurs in several variants and can also be found at non-interval optimization algorithms.

The test needs an inclusion function F of f and preassumes that, from the beginning of the computation, to any box Y which has entered the list $\mathcal{L}$ some function value $f(c)$ with $c \in Y$ is determined (frequently, c is the midpoint of Y). This allows to keep track of the lowest function value, say $\tilde{f}$, which has been determined up to the current iteration. Then the midpoint test reads as follows:

If $\tilde{f} < \min F(V)$ then delete V.

It is obvious that V contains no global minimizer of f over X, since $\tilde{f} < f(x)$ for any $x \in V$ due to the properties of an inclusion function.

2. Monotonicity Test. Let f be continuously differentiable and ∇f its gradient function. If f is strictly monotone in one of the variables over $V = V_1 \times \cdots \times V_m$, the box V cannot contain a global minimizer of f over $X = X_1 \times \cdots \times X_m$ except that the edge of X touches the edge of V. In this latter case it can happen that a global minimizer lies on the intersection of the two edges. Let $V(i/z)$ for $z \in R$ that box which occurs if the component V_i of V is replaced with z.

For the execution of the test we need for $i = 1, \ldots, m$ an inclusion function of the i-th component of ∇f, say $G_i^{(1)}$. Let further $V_i = [v_i, w_i]$ and $X_i = [x_i, y_i]$. Then the monotonicity test for strictly monotone increasing says:

For some $i = 1, \ldots, m$, if $0 < \min G_i^{(1)}(V)$ then

(i) if $x_i < v_i$ then discard V

(ii) if $x_i = v_i$ then replace V by the edgepiece $V(i/v_i)$.

The test for strictly monotone decreasing is analogous.

An interesting variant for the univariate case can be found in [26], where firstly derivates of f or – in more involved cases – of a Hermitean interpolation polynomial are determined recursively till a derivative has constant sign. This information is then used to gain solutions. A related simple version for polynomials is given in [47].

3. A Non-Convexity Test. Let f be twice continuously differentiable. The mathematical background of this test uses the simple fact that, if one of the main-diagonal elements of the Hessian matrix of f is always negative for points of V, no point of V exists for which f is convex. Hence, there is no global minimizer of f over X in V, except that V and X have common edge pieces.

Let $F^{(2)}$ be an inclusion function for the Hessian matrix of f, let $H = F^{(2)}(V)$ with diagonal elements $H_{ii} = [\rho_i, \sigma_i]$. Let the components of V and X as well as the notation $V(i/z)$ be as in 2., then the non-convexity test is:

For some $i = 1, \ldots, m$, if $\sigma_i < 0$ then

(i) if $x_i < v_i$ and $w_i < y_i$ then discard V,

(ii) if $x_i = v_i$ and $w_i < y_i$ then replace V by $V(i/v_i)$,

(iii) if $x_i < v_i$ and $w_i = y_i$ then replace V by $V(i/w_i)$,

(iv) if $x_i = v_i$ and $w_i = y_i$ then replace V by the two boxes $V(i/v_i)$, $V(i/w_i)$.

4. Interval Newton Methods. They are applied to ∇f in order to localize its zeros. The interval Newton methods are appropriate for generating box-discarding and box-diminishing tests as well as for proving existence and uniqueness of zeros. If V is small enough, interval Newton methods may act like classical, non-interval Newton methods, as the iterates of V shrink to the zeros. In the next section we present one sample of an interval Newton method, i.e., the one that is based upon the Krawczyk operator.

6. The Krawczyk Operator

The interval Newton method (more precisely, interval Newton-like method) which is based upon the Krawczyk operator [19] is an excellent global tool for localizing zeros of differentiable functions $g: X \to R^m$ where $X \in I^m$. (We think of g as the gradient function of f or as the Lagrangian function in Section 11.) We have chosen this operator for our discussion since it is well balanced between its computational and theoretical simplicity on the one hand, and the quality of the obtained numerical results on the other hand ([33]). A good overview of methods for solving nonlinear equations is given by [1], [33].

Let $G^{(1)}: I(X) \to I^{m \times m}$ be an isotone inclusion function of the Jacobian matrix of g. Then the *Krawczyk operator* is defined for any $Y \in I(X)$ as

$$K(Y) := c - Ag(c) + \{Id - AG^{(1)}(Y)\}(Y - c)$$

where $A \in R^{m \times m}$ is an arbitrary nonsingular matrix, $c \in Y$ any point and Id the identity matrix. [33] shows that $K(Y)$ gives best results when c is the midpoint of Y and $A = G_0^{-1}$ where G_0 is the midpoint of $G^{(1)}(Y)$ (to be understood componentwise) and, clearly, nonsingular. Hence, we shall assume this special choice of the parameters c and A in the sequel.

The Krawczyk operator has the following properties ([17], [19], [30]):

1. If x^* is a zero of g in Y then $x^* \in K(Y)$.
2. If $Y \cap K(Y) = \emptyset$, there is no zero of g in Y.
3. If $K(Y) \subseteq Y$ then Y contains a zero of g.
4. If $K(Y) \subseteq \operatorname{int} Y$ then Y contains exactly one zero of g ($\operatorname{int} Y$ denotes the interior of Y).

In order to consider the iterative aspects of K, we set $Y_0 := Y$, $Y_{n+1} := K(Y_n)$, $n = 0, 1, 2, \ldots$. Further $\| \ \|$ may be any norm on the space R^m. Then we get:

5. If $K(Y) \subseteq Y$ and $\|\mathrm{Id} - AG^{(1)}(Y)\| < 1$, then the sequence $(Y_n)_{n=0}^{\infty}$ is nested, Y contains a unique zero x^*, which lies in any Y_n, the sequence $(Y_n)_{n=0}^{\infty}$ converges to x^*, and the sequence $(w(Y_n))_{n=0}^{\infty}$ converges to 0 linearly. Modifications of the iteration steps may lead to quadratical convergence.

Note that all the assumptions that occur in the listed properties are computationally verifiable.

Supplementing the box-discarding and box-diminishing tests of the previous section we add to item 4 (interval Newton methods) the following two tests which are to be applied to the boxes $V = V_1, V_2$ of Step 3 of the Prototype Algorithm. It is assumed that f is twice continuously differentiable and we set $g := \nabla f$ when constructing K. The tests are:

(i) if $K(V) \cap V = \emptyset$, delete V,

(ii) If $K(V) \cap V \neq \emptyset$, replace V by $K(V) \cap V$.

Due to the properties of K, (i) and (ii) are obvious.

It is, however, not recommended to apply several iterations of K to V in the Prototype since there is no guarantee that the zeros of g are all global minimizers

such that the computational costs of determining them too fast may not be justified ([10]).

There exist global strategies for computing all zeros of g which are similar to the Prototype Algorithm. They combine box-deleting and box-diminishing tests with bisections and start with K-iterations as soon as the conditions which are cited under point 5 are verified. The description of such an algorithm can be found in [30].

7. Convergence Properties of the Prototype Algorithm

By specifying the bisection process and the ordering of the list $\mathscr{L}$ in the Prototype and by admitting or refusing the midpoint test we get three well-known algorithms, which show, however, varied convergence behaviour. Algorithm 1 may result from the Prototype by ordering the list by increasing lower bounds ([29]), [42]). Algorithm 2 results from Algorithm 1 by adding the midpoint test ([16]). Algorithm 3 orders the list by decreasing age of the boxes or by decreasing box widths and uses the midpoint test ([9], [10]). In all three cases, the boxes are bisected at the longest edge. The results presented in this section are proven in [32], [35], [37]).

At the n-th iteration of the Prototype, the actual state of the lists is denoted by $\mathscr{L}_n$. For instance, $\mathscr{L}_1$ consists just of X. Let X_n be a box of $\mathscr{L}_n$ that satisfies $\min F(X_n) \leq \min F(Z)$ for every box Z of $\mathscr{L}_n$. We call X_n the *leading box* of $\mathscr{L}_n$. Let $y_n := \min F(X_n)$. Hence, y_n is a lower bound of the function values $f(x)$, $x \in X_n$. The union of the boxes of $\mathscr{L}_n$ is denoted by U_n. Let again X^* be the set of global minimizers of f over X and f^* the global minimum if it exists.

Let us first consider Algorithm 1. One can show that

(a) $w(X_n) \to 0$ as $n \to \infty$.

This fact seems to be self-evident but it is not. For example, small modifications of the basic algorithm do not satisfy (a) as is the case with the cyclic bisection method ([31]). From the assumption

$$w(F(Y)) - w(f(Y)) \to 0 \text{ as } w(Y) \to 0 (Y \in I(X)) \tag{4}$$

it follows that

(b) $y_n \leq f^*$ for any n,
$y_n \to f^*$ as $n \to \infty$,
$f^* - y_n \leq w(F(X_n))$ (error estimate).

Assumption (4) is not very restrictive. It is almost always satisfied if natural interval extensions are used. However, (4) does not imply continuity, Lipschitz condition on f, etc. Let F now satisfy

$$w(F(Y)) \to 0 \text{ as } w(Y) \to 0. \tag{5}$$

Clearly, (5) implies (4) and the continuity of f. Then

(c) $w(F(X_n)) \to 0$ as $n \to \infty$

(that is, the error estimate tends to 0 and can thus be used for termination criteria),

(d) each accumulation point of the sequence (X_n) is a global minimizer.

The convergence order of the approach $y_n \to f^*$ is described by the following two results:

(e) Let any $\alpha > 0$ and any converging sequence of reals be given. Then, to any f, there exists an inclusion function of order α for which (y_n) converges slower than the given sequence.

This result indicates that the convergence can be arbitrarily slow and that no worst case exists, which is usually taken in order to establish formulas for the convergence speed or convergence order. If, however, only isotone inclusion functions are considered then the following estimate of the convergence speed is valid. Practically this estimate characterizes the proper convergence theory since it is always possible to find isotone inclusion functions.

(f) If F is isotone and of order α, then $f^* - y_n = O(n^{-\alpha/m})$.

Algorithms 2 and 3 have nearly the same behaviour as Algorithm 1 if the convergence to f^* is considered. Their properties with respect to a determination of X^* are as follows:

Let (U_n) be the sequence of unions produced by *Algorithm 2*. If (5) is assumed then

(g) the sequence (U_n) is nested and converges (with respect to the Hausdorff-metrics for compact sets) to a superset $D \supseteq X^*$. The probability, however, that $D \neq X^*$ is zero.

Let now (U_n) be the sequence of unions produced by *Algorithm 3*. If (5) is assumed then

(h) the sequence (U_n) is nested and converges to X^*.

The convergence rate which is established by property (f) is due to a worst case analysis and shows exponential cost increase with respect to the dimension. Practically, the situation is not that pessimistic and even comparable with optimization methods for local problems. This is due to the incorporation of the interval Newton method when the boxes of the lists are sufficiently small, provided f is twice continuously differentiable. Hence the worst case rate is replaced by the convergence rate of the interval Newton method from a certain stage of the computation.

An extensive number of numerical tests (up to dimension 75) is given in [46].

8. Termination Criteria

If only the determination of f^* is required, Algorithm 1 is appropriate. If we use the notation of the previous section then a reasonable criterion is

$$\text{if } w(F(X_n)) < \varepsilon \text{ then terminate} \tag{6}$$

where $\varepsilon > 0$. Since $f^* \in F(X_n)$ the number $w(F(X_n))$ is an upper bound for the absolute error when f^* is approximated by $F(X_n)$ or any value $\eta_n \in F(X_n)$, for example, $\eta_n := y_n$.

Practically one obtains better results when the criterion

$$\text{if } f_n - y_n < \varepsilon \text{ then terminate} \tag{7}$$

is used. Here f_n is the smallest function value of f which has been computed up to the n-th iteration. Since $y_n \leq f^* \leq f_n$, we again have an upper estimate of the approximation error $|y_n - f^*|$ of $|f_n - f^*|$ by $f_n - y_n$.

The function values f_n are in many cases available when the inclusion function values are computed.

If no function value is available then, in general,

$$f_n := \min_{i=1,\ldots,n} \max F(X_n)$$

will be the smallest upper bound of f^* known so far and it may be used in (7).

So far we have ignored the effect of rounding errors. If Alg. 1 is implemented on a computer then the convergence properties are disturbed or destroyed due to rounding errors. Then instead of the intended values $F(Y)$, so-called numerical values $\tilde{F}(Y)$ are delivered which approximate $F(Y)$.

If, however, machine interval arithmetic is implemented then it causes the inclusion

$$F(X_n) \subseteq \tilde{F}(X_n)$$

such that the global minimum remains included,

$$f^* \in \tilde{F}(X_n)\,.$$

The termination condition $w(F(X_n)) < \varepsilon$, cf. (6), is then numerically realized as the condition

$$w(\tilde{F}(X_n)) < \varepsilon\,. \tag{8}$$

If (8) is satisfied then it also follows that $w(F(X_n)) < \varepsilon$ such that termination of the algorithm remains correct, and

$$w(\tilde{F}(X_n)) < \varepsilon$$

is an absolute error bound of $f^* - y_n$. It is possible that in a certain phase of the computation the rounding errors overwhelm the decrease of $w(F(X_n))$, that is,

$w(\tilde{F}(x_n))$ does not tend to 0 even if $w(F(X_n)) \to 0$. Since such cases cannot be excluded, additional termination criteria are required.

We will turn now to Alg. 2 and 3. The considerations done so far for Alg. 1 are also valid for Alg. 2 and could be carried over to Alg. 3 if the right boxes would be addressed. Since Alg. 2 and 3 are intended to enclose X^*, termination criteria are expressed by the remaining boxes, that are shrinking to X^*. A termination of these algorithm via the convergence of the leading boxes is not reasonable since it does not allow an error estimation of the approximation of the global minimizers by the leading boxes.

For a unified treatment we take $B = \cap_{n=1}^{\infty} U_n \supseteq X^*$ as the solution set of both algorithms where $B = X^*$ in case of Alg. 3; see Section 7. The following two termination criteria were provided by [9], [10]. If B is at most denumerable and if B is to be included in a set with prescribed accuracy, then

$$\sum_{i=1}^{l_n} w(Z_{ni}) < \varepsilon \tag{9}$$

or

$$w(Z_{ni}) < \varepsilon \quad \text{for} \quad i = 1, \ldots, l_n \tag{10}$$

will do, where l_n is the length of $\mathscr{L}_n$. If B is nondenumerable then (9) will fail if the Lebesgue measure of B is at least equal to ε. Condition (10) works independently of the measure of B.

After having terminated, a list $\mathscr{L}_n$ of boxes $Z_{n1}, \ldots, Z_{nl_n}$ is left, and all we know is that

$$X^* \subseteq U_n = Z_{n1} \cup \cdots \cup Z_{nl_n}$$

and that (under reasonable conditions) $X^* \neq \emptyset$. There is no always working method for verifying that each of these boxes does contain a global minimizer in the strict sense of the definition. For practical purposes it does not matter and every such box can be assumed to contain global minimizers since, at this final stage of the computation, the function values of all these boxes are about f^*. Hence, existence tests, cf. Section 6, would at most cause a shrinking of these boxes.

The application of a uniqueness test, however, could lead to most insight if the boxes are connected and the box-hull of U_n is small. A positive result would then say that just one solution exists globally, which lies in one of these boxes.

Besides of the uniqueness test of Section 6 and related ones, a test of Mancini [22] is worth mentioning. This test gives reasonable conditions for positive definiteness of the Hessian matrix of f over a compact convex domain. Also, an integral Hessian matrix involved in an interval Newton-like method could be helpful ([23]).

9. Optimization over Unbounded Domains

Almost all methods for solving global optimization problems need the assumption that a bounded domain which contains the solution point is known. The boundedness is necessary for the numerical computation as well as for guaranteeing the convergence properties. If such a bounded domain is not known, linear substitutions such as $x = 1/s$ are commonly used to transform the unbounded into bounded parts. These substitutions are, however, rather troublesome to program because of the many distinctions which may arise. In this section another technique for treating unbounded domains due to [39] is described which is simple and robust and which avoids such substitutions. This technique is based upon an arithmetic of infinite intervals and is applicable to the Prototype Algorithm. A few topological considerations are necessary in order to reduce the convergence properties of the unbounded case to a bounded case.

Let $\bar{R} = R \cup \{\infty, -\infty\}$ be the two-point compactification of the real axis, R, then $\overline{R^m} := (\bar{R})^m$ is an appropriate compactification of R^m. The Prototype can be applied to $\overline{R^m}$ without any deeper modification. It is only necessary to define a width of intervals of $\overline{R}$, which should be finite and, for the bisection, the "midpoint" of such intervals. Both cause no difficulties. Further, if the principle of natural interval extensions is extended to infinite intervals it is possible to determine the lower bounds. Then most of the convergence properties of Section 7 apply to the case $\overline{R^m}$ under slight modifications of the assumptions.

It is obvious that the result gained in $\overline{R^m}$ need a careful interpretation in order to gain the results for R^m. For example, if $f^+ = -\infty$ is the global minimum gained in $\overline{R^m}$ then f is unbounded from below and no global minimizer of f exists in R^m. Or, if the only global minimizer gained in $\overline{R^m}$ is not in R^m and if $f^+ = 0$ then f has an infimum with value 0 but no global minimizer exists in R^m. The last mentioned situation occurs if, for instance, $f(x) = \exp(x)$ is considered.

The compactification is used only to simplify the discussion of the convergence properties. Thus, an arithmetic for compact intervals over $\bar{R}$ like $[0, \infty]$ need not be introduced. However, in lower to determine the lower bound of f over unbounded domains, we define an arithmetic for the set of *closed* intervals over R,

$$I_\infty = I \cup \{[a, \infty) : a \in R\} \cup \{(-\infty, a] : a \in R\} \cup \{(-\infty, \infty)\}$$

by

$$A * B = \{a * b : a \in A, b \in B\} \text{ if } * \in \{+, -, \cdot\}\ A, B \in I_\infty .$$

The quotient A/B for $A, B \in I_\infty$, $B \neq 0$ is defined as the smallest interval of I_∞ or the union of the two smallest intervals of I_∞ that contain the set

$$\{a/b : a \in A, b \in B, b \neq 0\} .$$

Hence, I_∞ is not closed with respect to division, the results like $(-\infty, a] \cup [b, \infty)$

can occur. Since such cases are rare, we split up such results into two intervals of I_∞ rather than to introduce a representation for the unions of intervals.

The arithmetic defined for I_∞ is oriented to the construction of natural interval extensions of functions over unbounded domains. This arithmetic and Kahan's well-known arithmetic are therefore different, cf. [17].

Natural interval extensions of the common pre-declared functions to unbounded intervals are defined as in Section 4: Let g be a predeclared function like sin, cos, exp, ln and D its domain. Then if $A \in I_\infty$, the natural interval extension of g to A, denoted by $g(A)$ is defined as the smallest interval $B \in I_\infty$ which contains the set $\{g(x): x \in A \cap D\}$. Therefore, natural interval extensions of function expressions to boxes $Y \in I_\infty^m$ can be defined recursively without difficulties.

The algorithms deal, as mentioned before, with compact boxes and intervals only. Hence, unbounded domains like R^m as well as unbounded inclusions like $[a, \infty)$ which have gained by natural extensions, have first to get compactified with respect to $\overline{R^m}$ or $\overline{R}$ before such data is submitted to the steps of the algorithm. This is, however, only of theoretical interest since all infinite intervals – whether compact or not – are rounded to machine representable intervals, when calculating on a computer. Let L be the largest representable machine number of the computer under consideration. If, for example, $f(x) = x^2 + \sin x$, then $f(R) = [0, \infty] + [-1, 1] = [-1, \infty)$. The representation on the computer gives $[0, L] + [-1, 1] = [-1, L]$. As one can see, machine intervals containing L or $-L$ need special attention, since L stands for $[L, \infty)$ or $[L, \infty]$, and not for the number L if it occurs as the right endpoint of an interval, etc.

There numerical results gained from the algorithms reflect and approximate the situation of $\overline{R^m}$. Thus they have to be reinterpreted in order to get the results from R^m. For example, if the numerical result says that $f^+ = 1$ and that $X^+ \subseteq [10^{90}, L]$ where X^+ is the set of global minimizers of the extended problem in $\overline{R^m}$ (with $m = 1$ for simplicity), then the logically sound conclusion is: The original problem in R^m has either a set of global minimizers, $X^* \subseteq [10^{90}, \infty)$, with global minimum, $f^* = 1$, or no global minimum exists, but $\lim_{x \to \infty} \inf f(x) = 1$.

The monotonocity test as described in Section 5 can also be extended to the unbounded case and remains a very effective means for accelerating the computation. Numerical tests can be found in [38], [39].

10. Nonsmooth Optimization

A broad spectrum of mathematical programming problems can be rather easily reduced to the minimization of nondifferentiable problems without constraints or with simple constraints. Also exact nonsmooth penalty functions in problems of nonlinear programming, maximum functions or multi-criterion optimization generate problems of nonsmooth optimization. Thus, the objective function, f, of the

optimization problem may look like

$$f(x) = \max\{f_1(x), \ldots, f_n(x)\}$$

where $f_i \in C^1$. Other objective functions arising from penalty methods are of the typical form

$$f(x) = \mu f_0(x) + \sum_{i=1}^{k} \max(0, f_i(x))$$

where f_0, $f_i \in C^1$ and $\mu > 0$ is a (reciprocal) penalty factor. It is well known that nonsmooth objective functions may cause classical methods to fail, but it is a little surprising that interval methods have no difficulties at all to handle nonsmooth problems, cf. the discussion in [38]. That is, because neither the construction of inclusion functions nor the application of monotonicity tests depend on the smoothness of the objective function. I.e., nonsmooth functions have substitutes for the nonexisting gradients such as subgradients, generalized gradients, etc. But it does not make any difference for the application of interval methods whether a mean value $f'(\xi)$ or a subgradient is included by intervals.

For an example, let us consider generalized gradients (cf. [5]). Let $X \in I^m$, $x \in X$ and $f: X \to R$ be Lipschitz near x, that shall mean, there exists an open neighbourhood of x, say u_x, in which f satisfies a Lipschitz condition. It follows by a theorem of Rademacher that f is differentiable almost everywhere in U_x. Let Ω be the set of points in U_x at which f in not differentiable, and let S be any other set of Lebesgue measure 0. Then the *generalized gradient* of f at x is defined as

$$\partial f(x) = \operatorname{conv}\{\lim_{n\to\infty} \nabla f(x_n) : x_n \to x, x_n \notin S \cup \Omega\}$$

where conv denotes the convex hull. Let $(x, y) \subseteq R^m$ denote the open line segment between x and y. A theorem of Lebourg says that, if $y \in U_x$ with $(x, y) \subseteq U_x$ is given then some $u \in (x, y)$ exists such that

$$f(y) - f(x) \in (y - x)^T \partial f(u) . \tag{11}$$

Locally, (11) can be approximated by means of the Lipschitz constant. Globally, (11) can be used to find inclusion functions of f of a meanvalue type explicitly: If $G(Y)$ is a – not necessarily bounded box – that contains $\partial f(u)$ for any $u \in Y$, then

$$F(Y) = f(c) + (Y - c)^T G(Y) \quad \text{for} \quad Y \in I(X)$$

where c denotes the midpoint of Y (also any other point of Y may be chosen), is an inclusion function of f. Further, $G(Y)$ can be used for the monotonicity test: If only one component of $G(Y)$ does not contain zero, then f is strictly monotone with respect to the corresponding direction.

Therefore, the Prototype Algorithm, as well as the monotonicity test can be applied without modification, if the objective function of f is nonsmooth. Numerical tests are given in [38], [39].

11. Constrained Optimization

The interval principles which were developed in the previous sections are also useful for constrained problems, that is,

$$\min_{x \in M} f(x) \tag{12}$$

where $M \subseteq R^m$ means the feasible set defined by some constraints

$$g_i(x) \leqslant 0,\ i = 1, \ldots, k\,,$$

$$h_i(x) = 0,\ j = 1, \ldots, s\,.$$

For simplicity, we assume that $M \subseteq X$ for some box $X \in I^m$ and that the functions f, g_i and h_j are defined on X. For a successful treatment of problem (12) we need inclusion functions F, G_i and H_j of f, g_i and h_j, respectively which satisfy (5) for $i = 1, \ldots, k$ and $j = 1, \ldots, s$.

Then a typical means of interval arithmetic is the *infeasibility test* which is applicable to any $Y \in I(X)$: If either

$$G_i(Y) > 0 \text{ for some } i \in \{1, \ldots, k\}$$

or if

$$0 \notin H_j(Y) \text{ for some } j \in \{1, \ldots, s\}$$

then all points of Y are infeasible. (The notation $[a, b] > 0$ or $[a, b] \leqslant 0$ shall indicate that $a > 0$ or $b \leqslant 0$ holds, respectively.) Hence the box Y can never contain a solution of (12) such that y can be discarded from any procedure to solve (12). Conversely, if

$$G_i(Y) \leqslant 0 \text{ for } i = 1, \ldots, k$$

and

$$H_j(Y) = 0 \text{ for } j = 1, \ldots, s$$

then all points of Y are feasible (*feasibility test*), which is due to the inclusion principle, (3). However, if equality constraints are present the direct satisfaction of conditions like $H_j(Y) = 0$ is unlikely but they come into effect at variable reduction methods.

Algorithmically, the Prototype Algorithm of Section 2 is also appropriate for the constraint problem. One has just to incorporate the infeasibility test as a box-deleting test and to adapt the tests that have already been mentioned in Section 5 to the constraint case. Also the knowledge of feasibility is important, since it is a first step before other tests can be applied, see below.

It is frequently unnecessary to check whether a box V hurts the constraints. That is, if $G_i(Y) \leqslant 0$ has been verified for some box Y and for some i, then no subbox of Y hurts the constraint $g_i(x) \leqslant 0$ so that the evaluations of $G_i(V)$ for

$V \in I(Y)$ can be spared out, cf. [38], who use a flag-system or Hansen–Jaumard–Lu [14], who use Boolean expressions for controlling this information.

When computationally verifying $H_i(Y) = 0$ some tolerance $\varepsilon > 0$ has to be conceded in oder to counterbalance the rounding errors. A way out where no tolerance is needed is suggested by [13]. Their method is based on existence tests (for instance, with an interval Newton method) and the implicit function theorem, cf. [11], [38].

The box-deleting and box-diminishing tests of Section 5 can be used again. One has to be aware however, that the function values $\tilde{f}$ that are used for the midpoint test have to be taken from feasible points only. Also the monotonicity test is only allowed for the application to feasible areas, where, similarly to the unconstraint case, one has to pay attention to binding constraints. The nonconvexity test is best applied to feasible areas, but see also [27] who investigates this test in connection with the Lagrangian function setting.

Interval Newton-like-techniques are of high importance as they are applied to the Lagrangian function in order to determine the Kuhn–Tucker points. Such a procedure was first described in [40].

Solving constraint problems with penalty methods can be found in [38]. If such methods are combined with the infeasibility test, there is no need for repeating the computations for the related unconstrained problems with an infinite number of penalty factors as is necessary, if nonexact penalty functions are used. We have to skip details.

An extensive acknowledgement of the global constraint optimization problem is given in [11], [12], [13], [27], [38], [41].

References

1. Alefeld, G. and Herzberger, J. (1983), *Introduction to Interval Computations*, Academic Press, New York.
2. Asaithambi, N. S., Shen, Z., and Moore, R. E. (1982), On Computing the Range of Values, *Computing* **28**, 225–237.
3. Bauch, H., Jahn, K. U., Oelschlägel, D., Süsse, H., and Wiebigke, V. (1987), *Intervallmathematik*, Teubner, Leipzig.
4. Caprani, O. and Madsen, K. (1979), Interval Methods for Global Optimization, Report NI 79-09, Technical University of Denmark.
5. Clarke, F. H. (1983), *Optimization and Nonsmooth Analysis*, Wiley, New York.
6. Dixon, L. C. W. and Fitzharris, A. M. (1985), Conjugate Gradients: An Interval Analysis, Technical Report Nr. 165. Numerical Optimization Center, Hatfield, Polytechnic, Hatfield.
7. Dussel, R. (1972), Einschließung des Minimalpunktes einer streng konvexen Funktion auf einem N-Dimensionalen Quader, Dissertation, Universität Karlsruhe.
8. Fang, Yuo-Kang (1982), Interval Test on Unconstrained Global Optimization (in Chinese). *Comm. Interval Anal., Math. Fac. of Liaoning Univ.* **2**, 43–59.
9. Hansen, E. (1979), Global Optimization Using Interval Analysis – The One-Dimensional Case, *J. Optim. Theory Appl.* **29**, 331–344.
10. Hansen, E. (1980), Global Optimization Using Interval Analysis – The Multi-Dimensional Case, *Numer. Math.* **34**, 247–270.

11. Hansen, E.: *Interval Tools for Global Optimization*, forthcoming.
12. Hansen, E. and Sengupta, S. (1983), Summary and Steps of a Global Nonlinear Constrained Optimization Algorithm. Lockheed Missiles & Space Co. Report No. D 889778, Sunnyvale, California.
13. Hansen, E. and Walster, G. W. (1987), Nonlinear Equations and Optimization, Preprint.
14. Hansen, P., Jaumard, B., and Lu, S.-H. (1991) An Analytical Approach to Global Optimization. *Math. Programming, Series B*, forthcoming.
15. Horst, R. and Tuy, H. (1990), *Global Optimization*, Springer-Verlag, Berlin.
16. Ichida, K. and Fujii, Y. (1979), An Interval Arithmetic Method for Global Optimization *Computing* **23**, 85–97.
17. Kahan, W. M. (1968), A More Complete Interval Arithmetic. Lectures Notes at the University of Michigan, Michigan.
18. Kearfott, R. B. (1987), Abstract Generalized Bisection and a Cost Bound, *Mathem. of Computation* **49**, 187–202.
19. Krawczyk, R. (1969), Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken, *Computing* **4**, 187–201.
20. Krawczyk, R. and Nickel, K. (1982), Die zentrische Form in der Intervallarithmetik, ihre quadratische Konvergenz und ihre Inklusions Isotonie, *Computing* **28**, 117–137.
21. Kulisch, U. and and Miranker, W. L. (1986), The Arithmetic of the Digital Computer: A New Approach, *SIAM Review*, March 1986, 1–40.
22. Mancini, L. J. (1975), Applications of Interval Arithmetic in Signomial Programming, Ph.D. Thesis. Stanford University.
23. Mancini, L. J. and McCormick, G. P. (1979), Bounding Global Minima with Interval Arithmetic, *Oper. Res.* **27**, 743–754.
24. Mancini, L. J. and Wilde, D. J. (1978), Interval Arithmetic in Unidimensional Signomial Programming, *J. Optim. Theory Appl.* **26**, 227–289.
25. Mancini, L. J. and Wilde, D. J. (1979), Signomial Dual Kuhn–Tucker Intervals, *J. Optim. Theory Appl.* **28**, 11–27.
26. McCormick, G. P. (1981), Finding the Global Minimum of a Function of one Variable Using the Method of Constant Signed Higher Order Derivatives, in: *Nonlinear Program. 4*, ed. by O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Academic Press, New York, 223–243 (1981).
27. Mohd, I. B. (1986), Global Optimization Using Interval Arithmetic Ph. D. Thesis, Univ. of St. Andrews, St. Andrews, Scotland.
28. Moore, R. E. (1966), *Interval Analysis*, Prentice-Hall, Englewood Cliffs.
29. Moore, R. E. (1976), On Computing the Range of a Rational Function of n Variables over a Bounded Region, *Computing* **16**, 1–15.
30. Moore, R. E. (1977), A Test for Existence of Solutions to Nonlinear Systems, *SIAM J. Numer. Analy.* **14**, , 611–615.
31. Moore, R. E. (1979) Methods and Applications of Interval Analysis, *SIAM*, Philadelphia.
32. Moore, R. E. and Ratschek, H. (1988), Inclusion Functions and Global Optimization II, *Math. Programming* **41**, 341–356.
33. Neumaier, A. (1991), *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, forthcoming.
34. Oelschlaegel, D. and Süsse, H. (1978), Fehlerabschätzung beim Verfahren von Wolfe zur Lösung quadratischer Optimierungsprobleme mit Hilfe der Intervallarithmetik, *Math. Operationsforsch. Statist., Ser. Optimization* **9**, 389–396.
35. Ratschek, H. (1985), Inclusion Functions and Global Optimization, *Mathematical Programming* **33**, 300–317.
36. Ratschek, H. and Rokne, J. (1984), *Computer Methods for the Range of Functions*, Horwood, Chichester.
37. Ratschek, H. and Rokne, J. (1987), Efficiency of a Global Optimization Algorithm, *SIAM J. Numer. Analysis* **24**, 1191–1201.
38. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*, Horwood, Chichester.
39. Ratschek, H. and Voller, R. L. (1990), Global Optimization over Unbounded Domains, *SIAM J. Control Optimization* **28**, 528–539.

40. Robinson, S. M. (1973), Computable Error Bounds for Nonlinear Programming, *Math. Programming* **5**, 235–242.
41. Sengupta, S. (1981), Global Nonlinear Constrained Optimization, Dissertation, Department of Pure and Applied Mathematics, Washington State University.
42. Skelboe, S. (1974), Computation of Rational Interval Functions, *BIT* **4**, 87–95.
43. Stroem, T. (1971), Strict Estimation of the Maximum of a Function of one Variable, *BIT* **11**, 199–211.
44. Süsse, H. (1977), Intervallarithmetische Behandlung von Optimierungsproblemen und damit verbundener numerischer Aufgabenstellungen, Dissertation, Technische Hochschule Leuna–Merseburg.
45. Süsse, H. (1980), Intervallanalytische Behandlung nichtlinearer Optimierungsaufgaben, Dissertation zur Promotion B. Technische Hochschule "Carl Schorlemer", Leuna–Merseburg.
46. Walster, G. W., Hansen, E. R., and Sengupta, S. (1985), Test Results for a Global Optimization Algorithm, in: *Numerical Optimization 1984*, ed. by Boggs, P. T., Byrd, R. H., and Schnabel, R. B., Siam, pp. 272–282.
47. Dussel, R. and Schmitt, B. (1970) Die Berechnung von Schranken für den Wertebereich eines Polynoms in einem Intervall, *Computing* **6**, 35–60.