Journal of Global Optimization **14:** 365–393, 1999.

A Nonsmooth Global Optimization Technique Using Slopes: The One-Dimensional Case

DIETMAR RATZ
Institut für Angewandte Mathematik, Universität Karlsruhe, D-76128 Karlsruhe, Germany
(email: Dietmar.Ratz@math.uni-karlsruhe.de)

(Received 21 April 1997; accepted in revised form 26 August 1998)

Abstract. In this paper we introduce a pruning technique based on slopes in the context of interval branch-and-bound methods for nonsmooth global optimization. We develop the theory for a slope pruning step which can be utilized as an accelerating device similar to the monotonicity test frequently used in interval methods for smooth problems. This pruning step offers the possibility to cut away a large part of the box currently investigated by the optimization algorithm. We underline the new technique's efficiency by comparing two variants of a global optimization model algorithm: one equipped with the monotonicity test and one equipped with the pruning step. For this reason, we compared the required CPU time, the number of function and derivative or slope evaluations, and the necessary storage space when solving several smooth global optimization problems with the two variants. The paper concludes on the test results for several nonsmooth examples.

Key words: Branch-and-bound, Global optimization, Interval arithmetic, Interval slopes

1. Introduction

Interval branch-and-bound methods for global optimization address the problem of finding guaranteed and reliable solutions of global optimization problems

$$\min_{x \in X} f(x), \tag{1}$$

where $f : D \to \mathbb{R}$ is the objective function and $X \subseteq D$ is the search box representing bound constraints for x. These methods usually apply several interval techniques to reject regions in which the optimum can be guaranteed not to lie. For this reason, the original box X gets subdivided, and subregions which cannot contain a global minimizer of f are discarded, while the other subregions get subdivided again until the desired accuracy (width) of the boxes is achieved.

Very often, if f is continuously differentiable, these interval methods incorporate the so called monotonicity test (see [3, 4, 6, 8, 11, 12] for example) to discard boxes. This test uses first-order information of the objective function by means of an interval evaluation of the derivative over the current box. Depending on this enclosure containing zero or not, the current box must be treated further or can be deleted, respectively. Moreover, the interval derivative evaluation together with a

centered form (see [1, 8, 10] for example) is often used to improve the enclosure of the function range.

On the other hand, it is well known that interval slopes (introduced by Krawczyk and Neumaier [9]) together with centered forms offer the possibility to achieve better enclosures for the function range, as described in [8] and [10], for example. Thus, they might improve the performance of interval branch-and-bound methods. Although, since slopes cannot replace the derivatives needed in the monotonicity test (see Section 3 for details), the necessity of alternative box-discarding techniques arises.

In this paper, we introduce a new method for computing verified enclosures for the global minimum value and all global minimum points of nonsmooth one-dimensional functions subject to bound constraints. The method incorporates a special pruning step generated by interval slopes. This pruning step offers the possibility to cut away a large part of the current box, independently of the slope interval containing zero or not. We develop the theory for this slope pruning step and we give several examples.

During the review of the original version of this paper, one of the referees pointed out strong similarities between the techniques presented in this paper and the tests introduced by Hansen et al. in [7]. We got to know this very intersting paper only afterwards. At first glance, there are such similarities. In fact, our method was developed independently, it is motivated by the nonsmoothness of the objective function, and it is based on a different approach.

The method of Hansen et al. assumes that the objective function is at least three times differentiable. The derivatives of higher order are used to improve the bounds for the function range by means of centered forms. Therefore, tests like the cut-off test can be more powerful. Moreover, the derivatives are utilized to compute so-called cord-slopes which are used by several tests to eliminate parts of the search area.

In contrast, our technique uses interval slopes computed recursively (in the sense of [10], see Section 2 for details). Therefore, no requirements concerning the smoothness of the objective functions are necessary. Additionally, in the smooth case the recursively computed slope enclosures can be tighter than cord-slopes.

Moreover, we do not need the assumption that the objective function has only finitely many local minima, which is made in [7]. This is due to the fact that our pruning technique is based on a case distinction different from the one used by the tests in [7] (see Section 3 for details). Thus, in [7] it is only proven that the function values in the deleted parts of the current interval are not smaller than the global minimum value. In contrast, we additionally prove that our pruning step does not delete any global minimizer.

Thus, in some respects the method presented in this paper is a generalization of the tests on the function value introduced in [7] to the problem of finding the set of all minimizers (even a continuum of points) of nonsmooth functions. Since the method of Hansen et al. uses higher derivatives, it is not possible to

compare the results of both methods. Therefore, we study the effect of the application of our slope pruning step when replacing the monotonicity test in a first-order model algorithm for global optimization. We underline the improvements with respect to the required CPU time, the number of function and derivative evaluations, and the necessary storage space when solving several smooth global optimization problems.

In the following, $X \subseteq D \subseteq \mathbb{R}$ and $f : D \to \mathbb{R}$. The global minimum value of f on X is denoted by f^*, and the set of global minimizer points of f on X by X^*. That is,

$$f^* = \min_{x \in X} f(x) \qquad \text{and} \qquad X^* = \{x^* \in X \mid f(x^*) = f^*\}.$$

We denote real numbers by $x, y, \ldots$ and real bounded and closed intervals by $X = [\underline{x}, \overline{x}], Y = [\underline{y}, \overline{y}], \ldots$, where $\min X = \underline{x}$, $\max X = \overline{x}$, $\min Y = \underline{y}$, $\max Y = \overline{y}$, etc.

The set of compact intervals is denoted by $\mathbb{I} := \{[a, b] \mid a \leq b,\ a, b \in \mathbb{R}\}$. The width of the interval X is defined by $w(X) = \max X - \min X$, and the midpoint of the interval X is defined by $m(X) = (\min X + \max X)/2$.

We call a function $F : \mathbb{I} \to \mathbb{I}$ an *inclusion function* of $f : \mathbb{R} \to \mathbb{R}$ in X, if $x \in X$ implies $f(x) \in F(X)$. In other words, $f(X) = f_{\text{rg}}(X) \subseteq F(X)$, where $f(X) = f_{\text{rg}}(X)$ is the range of the function f on X. The inclusion function of the derivative of f is denoted by F'. Inclusion functions can be computed via interval arithmetic [1, 4] for almost all functions specified by a finite algorithm (i.e. not only for given expressions). Moreover, applying so-called automatic differentiation or differentiation arithmetic in connection with interval arithmetic [4, 5, 8], we are also able to compute the inclusion function for the derivatives or the slopes.

Automatic differentiation combines the advantages of symbolic and numerical differentiation and handles numbers instead of symbolic formulas. The computation of the derivative (or slope) is done automatically together with the computation of the function value. The main advantage of this process is that only the algorithm or formula for the function is required. No explicit formulas for the derivative (or slope) is required.

It is assumed in the following that the inclusion functions have the *isotonicity* property, i.e. $X \subseteq Y$ implies $F(X) \subseteq F(Y)$.

2. Slopes and centered forms

Centered forms (see [1, 8–10]) are special interval expansions and serve to reduce the overestimation in computing interval enclosures of the range of a function f over some interval X. Usually, a centered form is derived from the mean-value theorem. Suppose f is differentiable on its domain D. Then $f(x) = f(c) + f'(\xi)(x - c)$ with some fixed $c \in D$ and ξ between x and c. Let $c, x \in X$, so

$\xi \in X$. Therefore

$$\begin{aligned} f(x) \;=\; f(c) + f'(\xi)(x-c) \;&\in\; f(c) + f'(X)\cdot(x-c) \\ &\subseteq\; f(c) + F'(X)\cdot(X-c), \quad c, x \in X. \end{aligned} \tag{2}$$

Here, f is expanded with respect to every $x \in X$, since $G = F'(X)$ is an interval evaluation of the derivative of f over the entire interval X.

Krawczyk and Neumaier [10] showed that if we have an interval $S \in \mathbb{I}$ such that, for all $x \in X$ we have

$$f(x) = f(c) + s\cdot(x-c) \quad \text{for some } s \in S, \tag{3}$$

then the interval $F_s(X) := f(c) + S\cdot(X-c)$ encloses the range of f over X, that is $f_{\mathrm{rg}}(X) \subseteq F_s(X)$. Such an interval S can be calculated by means of an interval slope and not only with an interval derivative. If we use a slope, then f is expanded with respect to an arbitrary but fixed $c \in X$.

DEFINITION 1. The function $s_f : D \times D \to \mathbb{R}$ with

$$f(x) = f(c) + s_f(c, x)\cdot(x-c)$$

is called a *slope* (between c and x). In the one-dimensional case ($D \subseteq \mathbb{R}$), we have

$$s_f(c,x) = \begin{cases} \dfrac{f(x)-f(c)}{x-c} & \text{if } x \neq c \\ \widetilde{s} & \text{if } x = c, \end{cases}$$

where $\widetilde{s} \in \mathbb{R}$ may be arbitrarily chosen. Assuming f to be differentiable and the slope to be continuous, we can define $\widetilde{s} := f'(c)$.

Moreover, we define the *interval slope* of f over the interval X by

$$s_f(c, X) := \{s_f(c,x) \mid x \in X, \quad x \neq c\},$$

where f needs not be differentiable.

REMARKS. (i) It is easy to see, that $S = s_f(c, X)$ satisfies (3) and

$$f(x) \in f(c) + S\cdot(x-c) \subseteq f(c) + S\cdot(X-c). \tag{4}$$

(ii) Often $c = m(X)$ is used to compute the interval slope.
(iii) If we assume f to be continuously differentiable, then we have

$$f'(X) = \bigcup_{c\in X} s_f(c, X)$$

(cf. [10]).

Slopes as well as interval slopes can be calculated by means of a process similar to automatic differentiation process ([4], [8], [10]). The main advantage of this process is that only the algorithm or formula for the function is required. No explicit formulas for the derivatives or slopes are required.

Automatic slope computation evaluates functions specified by algorithms or formulas, where all operations are executed according to the rules of a *slope arithmetic*, which is an arithmetic for ordered tripels of the form

$$\mathcal{U} = (U_x, U_c, U_s), \quad \text{with } U_x, U_c, U_s \in \mathbb{I}.$$

For a function $u : D \to \mathbb{R}$ with $D \subseteq \mathbb{R}$, we have for an interval $X \in \mathbb{I}$ ($X \subseteq D$) and for a fixed $c \in X$

$$\begin{aligned} u(x) &\in U_x, \\ u(c) &\in U_c \quad \text{and} \\ u(x) - u(c) &\in U_s \cdot (x - c) \end{aligned}$$

for all $x \in X$.

For the constant function $u(x) = \lambda$ we use the representation $\mathcal{C} = (\lambda, \lambda, 0)$. The function $u(x) = x$, representing the independent variable x, we use $\mathcal{X} = (X, c, 1)$, where again $c \in X$ is fixed.

The rules for the arithmetic are

$$\mathcal{W} = \mathcal{U} \pm \mathcal{V} \Rightarrow \begin{cases} W_x = U_x \pm V_x, \\ W_c = U_c \pm V_c, \\ W_s = U_s \pm V_s, \end{cases}$$

$$\mathcal{W} = \mathcal{U} \cdot \mathcal{V} \Rightarrow \begin{cases} W_x = U_x \cdot V_x, \\ W_c = U_c \cdot V_c, \\ W_s = U_x \cdot V_s + U_s \cdot V_c, \end{cases}$$

$$\mathcal{W} = \mathcal{U} / \mathcal{V} \Rightarrow \begin{cases} W_x = U_x / V_x, \\ W_c = U_c / V_c, \\ W_s = (U_s - W_c \cdot V_s) / V_x, \end{cases}$$

where $0 \notin V_x$ is assumed in case of division. The operations for W_x, W_c and W_s in this definition are operations for intervals.

For an elementary function φ and $\mathcal{U} = (U_x, U_c, U_s)$, we define:

$$\mathcal{W} = \varphi(\mathcal{U}) \Rightarrow \begin{cases} W_x = \varphi(U_x), \\ W_c = \varphi(U_c), \\ W_s = U_s \cdot s_\varphi(U_c, U_x). \end{cases}$$

Usually, $s_\varphi(U_c, U_x)$ must be replaced by $\varphi'(U_x)$, but for convex or concave functions φ (at least locally in X) like $(\,)^2$, $\sqrt{\ }$, exp, or ln, the slope $s_\varphi(U_c, U_x)$ can be

computed explicitly, which yields better (sharper) enclosures. For example

$$s_\varphi(U_c, U_x) = \begin{cases} U_x + U_c & \text{if } \varphi = (\ )^2, \\ 1/(W_x + W_c) & \text{if } \varphi = \sqrt{\ }, \\ [s_\varphi(\underline{u}_c, \underline{u}_x), s_\varphi(\overline{u}_c, \overline{u}_x)] & \text{if } \varphi = (\ )^4. \end{cases}$$

Details can be found in [14] or in [15].

Evaluation of $f : D \to \mathbb{R}$ over $A \in \mathbb{I}$ with fixed $c \in A$ using the slope arithmetic delivers

$$f(\mathcal{X}) = f((A, c, 1)) = (Y_x, Y_c, Y_s)$$

and we have

$$f_{\mathrm{rg}}(A) \subseteq Y_x, \quad f(c) \in Y_c \quad \text{and} \quad f(x) - f(c) \in Y_s \cdot (x - c) \quad \forall x \in A.$$

EXAMPLE 1. Let $f(x) = x^2 - 4x + 2$. Using the slope arithmetic, we compute the enclosure Y_s of $s_f(c, A)$ for $A = [1, 7]$ and $c = 4$ by

$$\begin{aligned} f(\mathcal{X}) &= f((A, c, 1)) \\ &= (A, c, 1)^2 - 4 \cdot (A, c, 1) + 2 \\ &= ([1, 7], 4, 1)^2 - (4, 4, 0) \cdot ([1, 7], 4, 1) + (2, 2, 0) \\ &= ([1, 49], 16, [5, 11]) - ([4, 28], 16, 4) + (2, 2, 0) \\ &= ([-25, 47], 2, [1, 7]) \\ &= (Y_x, Y_c, Y_s), \end{aligned}$$

and we have $Y_s = s_f(c, A) = [1, 7]$. In contrast, if we compute the interval evaluation of $f'(x) = 2x - 4$ over $A = [1, 7]$ (which might also be done by automatic differentiation [5]), we get

$$F'(A) = 2 \cdot [1, 7] - 4 = [-2, 10].$$

Now, if we compare the naive interval evaluation of f over A with the derivative and the slope expansion we have

$$\begin{aligned} F(A) &= A^2 - 4A + 2 = [-25, 47], \\ f(c) + F'(A) \cdot (A - c) &= [-28, 32], \\ f(c) + Y_s \cdot (A - c) &= [-19, 23], \end{aligned}$$

underlining that the slope expansion gives the best result.

REMARKS. (i) In [2], a method is proposed to compute and use linear lower bounds for special functions in the context of one-dimensional global optimization. With the help of a slope arithmetic and with (3) and (4) it is possible to compute such linear lower bounds automatically.

(ii) If we compare the process of automatic slope computation using the slope arithmetic with the process of computing cord-slopes as given in [7], we can see that the recursively computed interval slopes can be tighter than cord-slopes.

EXAMPLE 2. Let $f(x) = (x-2)^4$, $X = [1,7]$, and $c = 4$. With the slope arithmetic we obtain

$$s_f(c, X) = \left[\frac{1-16}{1-4}, \frac{625-16}{7-4}\right] = [5, 203].$$

To compute the cord-slope interval $[L_0, U_0]$ as defined in [7], we need the derivatives $f'(x) = 4(x-2)^3$, $f''(x) = 12(x-2)^2$, and $f'''(x) = 24(x-2)$. Using the true ranges for these derivatives, we obtain

$$\begin{aligned}
[L_0, U_0] =& f'(X) \cap (f'(c) + f''(X)(X-c)/2) \\
& \cap (f'(c) + f''(c)(X-c)/2 + f'''(X)(X-c)^2/6 \\
=& [-4, 500] \cap (32 + [0, 300][-3, 3]/2 \\
& \cap (32 + 48[-3, 3]/2 + [-24, 120][0, 9]/6 \\
=& [-4, 500] \cap [-418, 482] \cap [-76, 284] \\
=& [-4, 284],
\end{aligned}$$

which is a wider interval than $s_f(c, X)$.

3. A pruning technique using slopes

In first-order interval methods for global optimization, the monotonicity test determines whether the function f is *strictly monotone* in an entire subinterval $Y \subset X$. If this is the case, then Y cannot contain a global minimizer in its interior. Further, a global minimizer can only lie on a boundary point of Y if this point is also a boundary point of X. Therefore, if f satisfies

$$0 \notin F'(Y), \tag{5}$$

then the subbox Y can be deleted (with the exception of boundary points of X).

If we want to apply slopes instead of derivatives, we cannot use this monotonicity test, since we have $s_f(c, X) \subseteq F'(X)$, but in general it is *not* true that $f'(x) \in s_f(c, X) \quad \forall c, x \in X$. Therefore, although $x^* \in Y$ is a local (or even global) minimizer with $f'(x^*) = 0$, it might happen that $0 \notin s_f(c, Y)$, and the latter cannot be used as a criterion to discard the box Y.

EXAMPLE 3. We again consider the function f from Example 1. Since $f(x) = x^2 - 4x + 2 = (x-2)^2 - 2$, we can easily see that $x^* = 2$ is a local and global minimizer of f. With $Y = A = [1,7]$ we have $s_f(c, Y) = [1,7]$ showing that $0 \notin s_f(c, Y)$ cannot be used as a criterion to discard Y, since $x^* \in Y$.

On the other hand, it is underlined in [3], that the monotonicity test is an essential accelerating tool for an efficient interval global optimization method. Thus,

the need of a corresponding tool (applicable to nonsmooth problems) in connection with slopes arises. In the following, we develop such a tool, which we call a *pruning step using slopes*. We assume the (possibly nonsmooth) objective function f to be continuous.

The main idea for the pruning step is based on the slope extension (4) and our subsequent Theorem 1. Improvements which take into account a known upper bound for the global minimum value are based on Theorems 2, 3 and 4.

THEOREM 1. *Let $f : D \to \mathbb{R}$, $Y = [\underline{y}, \overline{y}] \in \mathbb{I}$, $c \in Y \subseteq D \subseteq \mathbb{R}$. Moreover, let $S = [\underline{s}, \overline{s}] = s_f(c, Y)$ with $\underline{s} > 0$. Then*

$$p := c + (\underline{y} - c) \cdot \underline{s}/\overline{s}$$

satisfies

$$\underline{y} \le p \le c \tag{6}$$

and

$$\min_{x \in Y} f(x) = \min_{x \in [\underline{y}, p]} f(x) < \min_{p < x \le \overline{y}} f(x). \tag{7}$$

Proof. Since $\underline{y} \le c$ and $0 < \underline{s}/\overline{s} \le 1$, we have

$$\underline{y} = (1 - \underline{s}/\overline{s}) \cdot \underline{y} + \underline{s}/\overline{s} \cdot \underline{y} \le (1 - \underline{s}/\overline{s}) \cdot c + \underline{s}/\overline{s} \cdot \underline{y} = \underbrace{c + (\underline{y} - c) \cdot \underline{s}/\overline{s}}_{= p} \le c,$$

which proves (6).

From (3) and (4) we know that for all $x \in (c, \overline{y}]$ there exists an $s_x > 0$ with $s_x \in S$ and

$$f(x) = f(c) + s_x \cdot (x - c).$$

Therefore, $f(x) > f(c),\ \forall x \in (c, \overline{y}]$, and thus we know that

$$\min_{x \in Y} f(x) = \min_{x \in [\underline{y}, c]} f(x).$$

Now let $y^* \in Y$ with

$$f(y^*) = \min_{x \in Y} f(x). \tag{8}$$

If we assume that $p < y^* \le c$, then we know that there exist $s_l \in S$ and $s_* \in S$ satisfying $f(\underline{y}) = f(c) + s_l \cdot (\underline{y} - c)$ and $f(y^*) = f(c) + s_* \cdot (y^* - c)$. Thus we have

$$\begin{aligned} f(\underline{y}) &= f(c) + s_l \cdot (\underline{y} - c) \\ &\le f(c) + \underline{s} \cdot (\underline{y} - c) \\ &= f(c) + \overline{s} \cdot (p - c) \\ &< f(c) + \overline{s} \cdot (y^* - c) \\ &\le f(c) + s_* \cdot (y^* - c) \\ &= f(y^*), \end{aligned}$$

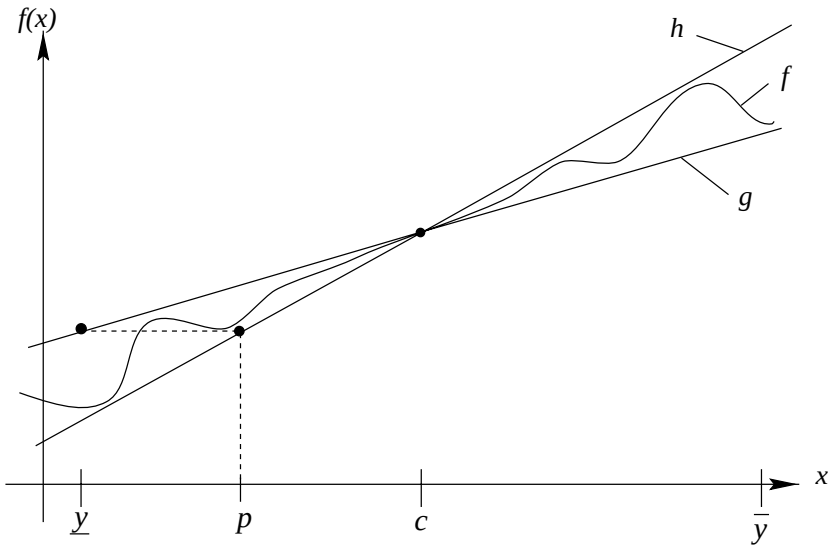


Figure 1. Generation of the pruning point p for positive interval slope.

ie. $f(\underline{y}) < f(y^*)$ which contradicts (8), and therefore $y^* \leq p$ which proves (7). □

Figure 1 illustrates the geometrical interpretation for finding the point p. First of all, we define the two lines

$$g : \mathbb{R} \to \mathbb{R} \quad g(x) := f(c) + \underline{s} \cdot (x - c) \tag{9}$$

and

$$h : \mathbb{R} \to \mathbb{R} \quad h(x) := f(c) + \overline{s} \cdot (x - c). \tag{10}$$

Then we know that $g(\underline{y})$ is an upper bound for $f(\underline{y})$ and thus for $\min_{x \in Y} f(x)$. Now we can locate p as the leftmost point in Y, for which f can not fall below $g(\underline{y})$. Since h is a lower bound for f in $[\underline{y}, c]$, we can do this very simply by computing the intersection point of h and the horizontal line z with $z(x) = g(\underline{y})$.

Using the value p of Theorem 1 within a global optimization method, we can prune a subinterval $Y \subseteq X$, if $0 < \underline{s} \leq \overline{s}$ for $S = s_f(c, Y)$ to

$$Y_P := [\underline{y}, c + (\underline{y} - c) \cdot \underline{s}/\overline{s}].$$

EXAMPLE 4. We consider $f(x) = 1/2x^2$, and we assume the current interval to be $Y = [-1, 4]$. First of all, we try to apply the monotonicity test. We evaluate the derivative $f'(x) = x$ over Y, and we get $F'(Y) = Y = [-1, 4]$. Since $0 \in F'(Y)$, we cannot discard Y from further consideration, and we must subdivide it and treat parts of Y in the same manner.

Now, we apply our new pruning step. We first evaluate the interval slope $S = s_f(c, Y) = \frac{1}{2}(c + Y)$, and with $c = 1.5$ we get $S = [0.25, 2.75]$. Since $0 \notin S$ we can prune Y to

$$\begin{aligned} Y_P &= [\underline{y}, c + (\underline{y} - c) \cdot \underline{s}/\overline{s}] = [-1, 1.5 + (-1 - 1.5) \cdot 0.25/2.75] \\ &= [-1, 1.273] \end{aligned}$$

using four significant digits and rounding outwards.

If we recall the situation in Figure 1, we see that we are able to improve the pruning of an interval Y. We can improve the point p (by moving it to the left), if we know a better (smaller) upper bound $\widetilde{f}$ for $f(x)$ on Y than $g(\underline{y})$ was. Moreover, if $\widetilde{f}$ is an upper bound for the global minimum value f^* on the whole search box X, then we can locate p as the leftmost point in Y, for which f can not fall below $\widetilde{f}$. Since h is a lower bound for f near $\underline{y}$, we can do this by computing the intersection point of h and the horizontal line z with $z(x) = \widetilde{f}$. In the context of a global optimization method using branch-and-bound techniques such as the cut-off test, an improved upper bound $\widetilde{f}$ for the global minimum value f^* is usually known. Therefore, we can state

THEOREM 2. *Let $f : D \to \mathbb{R}$, $Y = [\underline{y}, \overline{y}] \in \mathbb{I}$, $c \in Y \subseteq X \subseteq D \subseteq \mathbb{R}$. Moreover, let $S = [\underline{s}, \overline{s}] = s_f(c, Y)$ with $\underline{s} > 0$ and*

$$\widetilde{f} \geq f^* = \min_{x \in X} f(x). \tag{11}$$

Then $p := c + m/\overline{s}$ with $m = \min\{\widetilde{f} - f(c),\ (\underline{y} - c) \cdot \underline{s}\}$ satisfies

$$p \leq c, \tag{12}$$

and

$$\min_{p < x \leq \overline{y}} f(x) > f^* \quad \text{for } \underline{y} \leq p \tag{13}$$

or

$$\min_{x \in Y} f(x) > f^* \quad \text{for } p < \underline{y}, \tag{14}$$

respectively.

Proof. Since $\underline{y} \leq c$ and $0 < \underline{s} \leq \overline{s}$, we have $m \leq 0$ and

$$p = c + m/\overline{s} \leq c$$

which proves (12).

From (4) we know that for all $x \in (c, \overline{y}]$ there exists an $s_x > 0$ with $s_x \in S$ and

$$f(x) = f(c) + s_x \cdot (x - c).$$

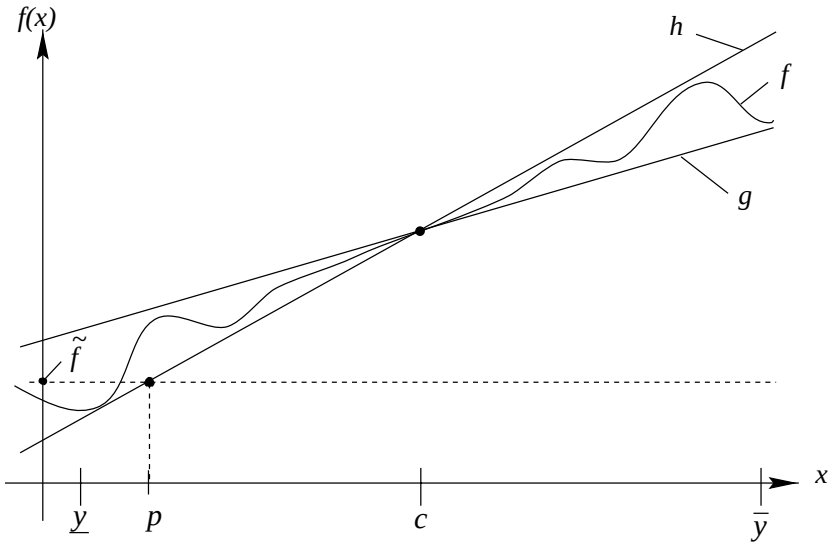


Figure 2. Generation of the pruning point p with known $\widetilde{f}$.

Therefore, $f(x) > f(c),\ \forall x \in (c, \overline{y}]$, which directly proves (13) and (14) for $p = c$.

Now, let $p < c$. For the case $m = (\underline{y} - c) \cdot \underline{s}$, Theorem 1 implies $\underline{y} \leq p$ and

$$\min_{p<x\leq\overline{y}} f(x) > \min_{x\in Y} f(x) \geq f^*.$$

For the case $m = \widetilde{f} - f(c)$, we assume that there exists an $x^* \in \mathcal{Z} := Y \cap (p, c]$ with $f(x^*) = f^*$. Then we know that there exists an $s_* \in S$ satisfying $f(x^*) = f(c) + s_* \cdot (x^* - c)$. Thus we have

$$\begin{aligned} f(x^*) &= f(c) + s_* \cdot (x^* - c) \\ &\geq f(c) + \overline{s} \cdot (x^* - c) \\ &> f(c) + \overline{s} \cdot (p - c) \\ &= f(c) + \overline{s} \cdot (m/\overline{s}) \\ &= f(c) + m \\ &= \widetilde{f}, \end{aligned}$$

which contradicts (11), and therefore $x^* \notin \mathcal{Z}$, which proves (13) and (14). □

REMARK. It is easy to see, that in the case $\widetilde{f} < f(c) + (\underline{y} - c) \cdot \underline{s}$ the value p computed in Theorem 1 is smaller than that computed in Theorem 2.

Figure 2 illustrates the geometrical interpretation for finding the point p when using the known upper bound $\widetilde{f}$ for the global minimum value. Again we use the

two lines

$$g : \mathbb{R} \to \mathbb{R} \qquad g(x) := f(c) + \underline{s} \cdot (x - c)$$

and

$$h : \mathbb{R} \to \mathbb{R} \qquad h(x) := f(c) + \overline{s} \cdot (x - c).$$

Then we know that $\widetilde{f}$ is an upper bound for $\min_{x \in Y} f(x)$. Now we can locate p as the leftmost point in Y, for which f cannot fall below $\widetilde{f}$. Since h is a lower bound for f in $[\underline{y}, c]$, we can do this very simply by computing the intersection point of h and the horizontal line z with $z(x) = \widetilde{f}$.

So, we can use Theorem 2 within a global optimization method to prune or delete a subinterval $Y \subseteq X$, if $0 < \underline{s} \leq \overline{s}$ for $S = [\underline{s}, \overline{s}] = s_f(c, Y)$. That is, we first compute

$$m = \min\{\widetilde{f} - f(c),\ (\underline{y} - c) \cdot \underline{s}\}$$

and then

$$p = c + m/\overline{s} \leq c.$$

Then, if $p \geq \underline{y}$, we replace Y by

$$Y := [\underline{y}, p],$$

otherwise we delete the whole subbox Y.

It is easy to see, that we can apply a similar procedure for pruning in the case $\overline{s} < 0$, since we immediately have

THEOREM 3. *Let $f : D \to \mathbb{R}$, $Y = [\underline{y}, \overline{y}] \in \mathbb{I}$, $c \in Y \subseteq X \subseteq D \subseteq \mathbb{R}$. Moreover, let $S = [\underline{s}, \overline{s}] = s_f(c, Y)$ with $\overline{s} < 0$ and*

$$\widetilde{f} \geq f^* = \min_{x \in X} f(x). \tag{15}$$

Then $q := c + m/\underline{s}$ with $m = \min\{\widetilde{f} - f(c),\ (\overline{y} - c) \cdot \overline{s}\}$ satisfies

$$c \leq q, \tag{16}$$

and

$$\min_{\underline{y} \leq x < q} f(x) > f^* \quad \text{for } q \leq \overline{y} \tag{17}$$

or

$$\min_{x \in Y} f(x) > f^* \quad \text{for } \overline{y} < q, \tag{18}$$

respectively.

Proof. The proof is completely analogous to the proof of Theorem 2. □

Up to now, we have treated the case $0 \notin s_f(c, Y)$, which corresponds to the (successful) case $0 \notin F'(Y)$ in the usual monotonicity test. A further advantage of our new pruning technique with slopes is, that we also can apply this technique successfully in the case $0 \in s_f(c, Y)$, which corresponds in a sense to the (unsuccessful) case $0 \in F'(Y)$ for the usual monotonicity test. For this purpose we can use the following

THEOREM 4. *Let $f : D \to \mathbb{R}$, $Y = [\underline{y}, \overline{y}] \in \mathbb{I}$, $c \in Y \subseteq X \subseteq D \subseteq \mathbb{R}$. Moreover, let $S = [\underline{s}, \overline{s}] = s_f(c, Y)$ with $0 \in S$ and*

$$f(c) > \widetilde{f} \geq f^* = \min_{x \in X} f(x). \tag{19}$$

Then

$$p := \begin{cases} c + (\widetilde{f} - f(c))/\overline{s} & \text{if } \overline{s} \neq 0, \\ -\infty & \text{otherwise}, \end{cases}$$

$$q := \begin{cases} c + (\widetilde{f} - f(c))/\underline{s} & \text{if } \underline{s} \neq 0, \\ +\infty & \text{otherwise}, \end{cases}$$

and

$$\mathcal{Z} := (p, q) \cap Y$$

satisfy

$$p < c < q, \tag{20}$$

and

$$\min_{x \in \mathcal{Z}} f(x) > f^*. \tag{21}$$

Proof. Since $\widetilde{f} - f(c) < 0$ and with $\underline{s} < 0 < \overline{s}$, we have

$$p = c + (\widetilde{f} - f(c))/\overline{s} < c < c + (\widetilde{f} - f(c))/\underline{s} = q,$$

which proves (20).

Now, we assume that there exists an $x^* \in \mathcal{Z}$ with $f(x^*) = f^*$. Then we know that there exists an $s_* \in S$ satisfying $f(x^*) = f(c) + s_* \cdot (x^* - c)$.

We also know that $x^* \neq c$, since equality would contradict (19).

If $x^* < c$, then we have $0 < s_* \leq \overline{s}$ and

$$\begin{aligned} f(x^*) &= f(c) + s_* \cdot (x^* - c) \\ &\geq f(c) + \overline{s} \cdot (x^* - c) \\ &> f(c) + \overline{s} \cdot (p - c) \\ &= f(c) + \overline{s} \cdot (\widetilde{f} - f(c))/\overline{s} \\ &= \widetilde{f}, \end{aligned}$$

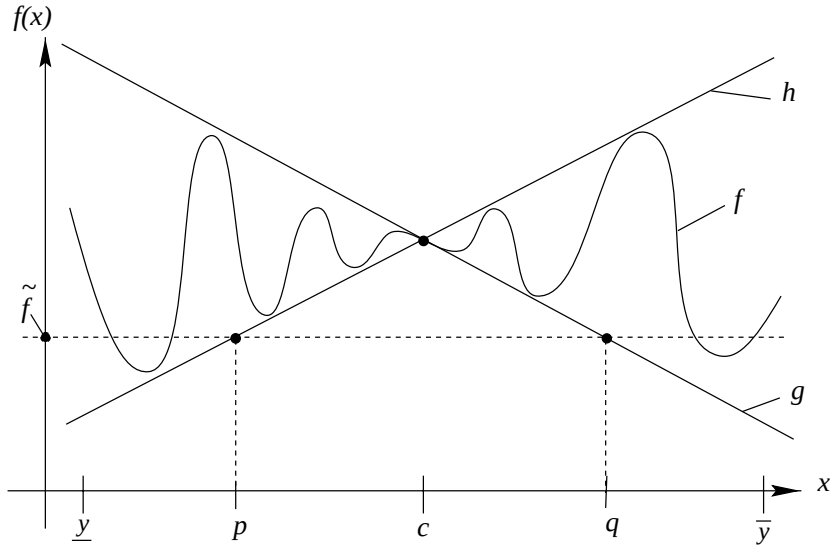


Figure 3. Generation of pruning points p and q with known $\widetilde{f} < f(c)$.

which contradicts (19).

If $x^* > c$, then in a similar way we have $\underline{s} \le s_* < 0$ and

$$\begin{aligned} f(x^*) &= f(c) + s_* \cdot (x^* - c) \\ &\ge f(c) + \underline{s} \cdot (x^* - c) \\ &> f(c) + \underline{s} \cdot (q - c) \\ &= f(c) + \underline{s} \cdot (\widetilde{f} - f(c))/\underline{s} \\ &= \widetilde{f}, \end{aligned}$$

which also contradicts (19).

Therefore $x^* \notin \mathcal{Z}$ and we proved (21). □

We illustrate the geometrical interpretation for finding the points p and q in Figure 3. Again we use the two lines

$$g : \mathbb{R} \to \mathbb{R} \quad g(x) := f(c) + \underline{s} \cdot (x - c)$$

and

$$h : \mathbb{R} \to \mathbb{R} \quad h(x) := f(c) + \overline{s} \cdot (x - c),$$

assuming $\underline{s} < 0 < \overline{s}$. Now we can locate p as the leftmost point and q as the rightmost point in Y, for which f cannot fall below $\widetilde{f}$ according to the bounding by g and h. Since h is a lower bound for f in $[\underline{y}, c]$ and since g is a lower bound for f in $[c, \overline{y}]$, we can do this very simply by computing the intersection points of h and g with the horizontal line z with $z(x) = \widetilde{f}$.

So, we can use Theorem 4 within a global optimization method to prune or delete a subinterval $Y \subseteq X$, if $\underline{s} \leq 0 \leq \overline{s}$ for $S = s_f(c, Y)$ and if $\widetilde{f} < f(c)$. That is, we first compute

$$p = c + (\widetilde{f} - f(c))/\overline{s} \quad \text{and} \quad q = c + (\widetilde{f} - f(c))/\underline{s}.$$

Then, we replace Y by

$$\begin{array}{ll} [\underline{y}, p] \cup [q, \overline{y}] & \text{if } \underline{y} \leq p \wedge q \leq \overline{y}, \\ [\underline{y}, p] & \text{if } \underline{y} \leq p \wedge q > \overline{y}, \\ [q, \overline{y}] & \text{if } \underline{y} > p \wedge q \leq \overline{y}, \end{array}$$

and otherwise we delete the whole subbox Y. If $\widetilde{f} \geq f(c)$, then we only perform a bisection of Y.

REMARK. Comparing the the proof of Test 2 in [7] with the proofs of our Theorems 2, 3 and 4, we notice that different case distinctions with respect to S are used. Moreover, in [7] it is only proven that $f(x) \geq \widetilde{f} \geq f^*$ for x in the deleted parts of the interval Y. In contrast, we additionally proved that the pruning step does not delete any global *minimizer*, i.e. $f(x) > f^*$ for x in the deleted parts of Y.

EXAMPLE 5. If we apply our pruning technique to the three times differentiable function

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x^2 & \text{if } x \geq 0 \end{cases}$$

for $X = [-3, 1]$, $c = -1$, $\widetilde{f} = 0$, and $S = [0, 1]$, it results in a bisection of X. Thus, no global optimizer is lost.

If we apply Test 2 from [7], we obtain the cord-slope interval $[L_0, U_0] = [0, 1] = S$ and

$$y = c - (f(c) - \widetilde{f})/U_0 = -1 - (0 - 0)/1 = -1.$$

Therefore, the test deletes the interval $(-1, 1]$, and the subset $(-1, 0]$ of the set of global optimizers $[-3, 0]$ is completely lost!

Nevertheless, the cord-slope technique works correctly for functions with only finitely many local minima such as polynomials, for example.

4. An algorithmic pruning step

We are now able to give an algorithmic formulation of a pruning step, which can be applied to a subinterval $Y \subseteq X$ when globally minimizing $f : D \to \mathbb{R}$ on $X \subseteq D$. The algorithm uses

$$Y = [\underline{y}, \overline{y}],\ c \in Y,\ f_c = f(c),\ S = [\underline{s}, \overline{s}] = s_f(c, Y),\ \text{and } \widetilde{f} \geq \min_{x \in X} f(x)$$

as input, and it delivers the pruned (and possibly empty) subset $U_1 \cup U_2$ of Y with $U_1, U_2 \in \mathbb{I} \cup \{\emptyset\}$ and a possibly improved $\widetilde{f}$ as output.

ALGORITHM 1. SlopePruning $(Y, c, f_c, S, \widetilde{f}, U_1, U_2)$

```
1.  U1 := ∅;    U2 := ∅;
2.  if 0 ∈ S then                                   {pruning from the center}
3.    if f̃ < fc then                                {a pruning is possible}
4.      if s̄ > 0 then                               {pruning from the center to the left}
5.        p := c + (f̃ − fc)/s̄;
6.        if p ≥ y then U1 := [y, p];               {compute remaining left part}
7.      if s < 0 then                               {pruning from the center to the right}
8.        q := c + (f̃ − fc)/s;
9.        if q ≤ ȳ then U2 := [q, ȳ];               {compute remaining right part}
10.   else                                          {a pruning is not possible}
11.     U1 := [y, c];    U2 := [c, ȳ];              {bisection}
12. else if s > 0 then                              {pruning from right}
13.   f̃ := min{f̃, (y − c) · s + fc};                {update f̃}
14.   p := c + (f̃ − fc)/s̄;
15.   if p ≥ y then U1 := [y, p];                   {compute remaining left part}
16. else {s̄ < 0}                                    {pruning from left}
17.   f̃ := min{f̃, (ȳ − c) · s̄ + fc};                {update f̃}
18.   q := c + (f̃ − fc)/s;
19.   if q ≤ ȳ then U2 := [q, ȳ];                   {compute remaining right part}
20. return U1,   U2,   f̃;
```

Summarizing the properties of this pruning step, we can state

THEOREM 5. *Let $f : D \to \mathbb{R}$, $Y \in \mathbb{I}$, $c \in Y \subseteq X \subseteq D \subseteq \mathbb{R}$. Moreover, let $f_c = f(c)$, $S = s_f(c, Y)$, and $\widetilde{f} \geq \min_{x \in X} f(x)$, then Algorithm 1 applied as* SlopePruning *$(Y, c, f_c, S, \widetilde{f}, U_1, U_2)$ has the following properties:*

1. *$U_1 \cup U_2 \subseteq Y$.*
2. *Every global optimizer x^* of f in X with $x^* \in Y$ satisfies $x^* \in U_1 \cup U_2$.*
3. *If $U_1 \cup U_2 = \emptyset$, then there exists no global (w.r.t. X) optimizer of f in Y.*

Proof. Property 1 follows from the definition of U_1 and U_2. Theorems 2 to 4 directly imply Property 2. Property 3 is a consequence of Property 2.

It is obvious that the success of Algorithm 1 in pruning Y depends on the quality of $\widetilde{f}$. Therefore, the pruning step within a global optimization method can very much benefit from a fast local search method delivering a good (small) value $\widetilde{f}$ on a very early stage of the method. For our further studies in this paper, we do not use an additional local method.

5. A global optimization algorithm using pruning steps

Subsequently, we give a simple first-order model algorithm to demonstrate the advantages of our new pruning step. Our model algorithm uses the cut-off test,

but it includes no local search procedure. Since we do not require smoothness of the objective function, we also do not use a concavity test and Newton-like steps.

ALGORITHM 2. GlobalOptimize (f, X, ε, F^*, L_{res})

```
1.  c := m(X);    f̃ := f(c);                                   {initialize upper bound}
2.  F_X := (f(c) + s_f(c, X) · (X − c)) ∩ F(X);                {centered form}
3.  L := {(X, f_X)};    L_res := { };                          {initialize working list and result list}
4.  while L ≠ { } do
5.   (Y, f_Y) := PopHead (L);    c := m(Y);                   {get first element of working list}
6.   SlopePruning (Y, c, f(c), s_f(c, Y), f̃, U_1, U_2);
7.   for i := 1 to 2 do
8.    if U_i = ∅ then next_i;
9.    c := m(U_i);
10.   if f(c) < f̃ then f̃ := f(c);
11.   F_U := (f(c) + s_f(c, U_i) · (U_i − c)) ∩ F(U_i);        {centered form}
12.   if f_U ≤ f̃ then
13.     if d_rel(F_U) ≤ ε or d_rel(U_i) ≤ ε then
14.       L_res := L_res ⊎ (U_i, f_U)                          {accept U_i for the result list}
15.     else
16.       L := L ⊎ (U_i, f_U);                                 {store U_i in the working list}
17.   end for
18.   CutOffTest (L, f̃);
19. end while
20. (Y, f_Y) := Head (L_res);    F* := [f_Y, f̃];    CutOffTest (L_res, f̃);
21. return F*,    L_res.
```

Algorithm 2 first computes an upper bound $\widetilde{f}$ for the global minimum value and initializes the working list L and the result list L_{res}. The main iteration (from Step 4 to Step 19) starts with the pruning step applied to the leading interval of the working list. Then we apply a range check using a centered form to the resulting boxes U_1 and U_2 if they are non-empty. If the current box is still a candidate for containing a global minimizer, we store it in L_{res} (if it can be accepted with respect to the tolerance ε) or in L if it must be treated further.

Note that by the operation $\uplus$ the boxes are stored as pairs $(Y, \underline{f_Y})$ in list L sorted in *nondecreasing* order with respect to the lower bounds $\underline{f_Y} \le \underline{f_{\text{rg}}(Y)}$ and in *decreasing* order with respect to the ages of the boxes in L (cf. [13]). Thus, the leading box of L is the oldest element with the smallest $\underline{f_Y}$ value. When the iteration stops because the working list L is empty, we compute a final enclosure F^* for the global minimum value and return L_{res} and F^*.

The method can be improved by incorporating an approximate local search procedure to try to decrease the value $\widetilde{f}$. For our studies in this paper, we do not apply any local method. The cut-off test is given by

ALGORITHM 3. CutOffTest $(L, \tilde{f})$

1. **for all** $(Y, \underline{f_Y}) \in L$ **do**
2. **if** $\tilde{f} < \underline{f_Y}$ **then** $L := L \uplus (Y, \underline{f_Y})$;
3. **end for**
4. **return** L;

where $L \uplus (Y, \underline{f_Y})$ removes the element $(Y, \underline{f_Y})$ from L.

For our global optimization algorithm (Algorithm 2) we can state

THEOREM 6. *Let $f : D \to \mathbb{R}$, $X \subseteq D \subseteq \mathbb{R}$, and $\varepsilon > 0$. Then Algorithm 2 has the following properties:*

1. $f^* \in F^*$.
2. $X^* \subseteq \bigcup_{(Y, \underline{f_Y}) \in L_{\text{res}}} Y$.

Proof. Since the lists are sorted in non-decreasing order with respect to the $\underline{f_Y}$ values and since $\tilde{f}$ is an upper bound of f^*, assertion 1 is proved. Assertion 2 follows from the fact that neither the cut-off test nor the slope pruning step (due to Theorem 5) deletes boxes which contain a global minimizer of f. □

EXAMPLE 6. To demonstrate the performance of our global optimization algorithm using pruning steps, we give an extract (about the first 9 steps) of the protocol of the pruning steps when applying Algorithm 2 on function

$$f(x) = \frac{(x-a)^2}{20} - \cos(x-a) + 2$$

with $a = 1.125$ and starting interval $X = [-5, 5]$.

For each current box Y in the while loop, we list its value, the value of the slope $S = s_f(c, Y)$, the chosen pruning step, and the resulting boxes U_1 and U_2. The empty set is represented by `[ / ]`.

```
    Y  = [ -5.000E+000,  5.000E+000 ]     S  = [ -1.363E+000,  1.138E+000 ]
 ==> bisection necessary
 ==> U1 = [ -5.000E+000,  0.000E+000 ]     U2 = [  0.000E+000,  5.000E+000 ]

    Y  = [  0.000E+000,  5.000E+000 ]     S  = [ -8.898E-001,  1.263E+000 ]
 ==> pruning by punching
 ==> U1 = [  0.000E+000,  2.288E+000 ]     U2 = [  2.801E+000,  5.000E+000 ]

    Y  = [  0.000E+000,  2.288E+000 ]     S  = [ -9.576E-001,  9.771E-001 ]
 ==> bisection necessary
 ==> U1 = [  0.000E+000,  1.144E+000 ]     U2 = [  1.143E+000,  2.288E+000 ]
```

```
    Y  = [  0.000E+000,  1.144E+000 ]     S  = [ -9.862E-001, -7.798E-003 ]
 ==> pruning from left
 ==> U1 = [            /             ]     U2 = [  7.384E-001,  1.144E+000 ]

    Y  = [  7.384E-001,  1.144E+000 ]     S  = [ -4.056E-001,  1.067E-002 ]
 ==> pruning by punching
 ==> U1 = [            /             ]     U2 = [  9.863E-001,  1.144E+000 ]

    Y  = [  9.863E-001,  1.144E+000 ]     S  = [ -1.481E-001,  1.687E-002 ]
 ==> pruning by punching
 ==> U1 = [            /             ]     U2 = [  1.077E+000,  1.144E+000 ]

    Y  = [  1.077E+000,  1.144E+000 ]     S  = [ -5.098E-002,  1.913E-002 ]
 ==> bisection necessary
 ==> U1 = [  1.077E+000,  1.111E+000 ]     U2 = [  1.110E+000,  1.144E+000 ]

    Y  = [  1.110E+000,  1.144E+000 ]     S  = [ -1.510E-002,  1.997E-002 ]
 ==> bisection necessary
 ==> U1 = [  1.110E+000,  1.128E+000 ]     U2 = [  1.127E+000,  1.144E+000 ]

    Y  = [  1.110E+000,  1.128E+000 ]     S  = [ -1.552E-002,  2.018E-003 ]
 ==> pruning by punching
 ==> U1 = [            /             ]     U2 = [  1.120E+000,  1.128E+000 ]
```

6. Implementation

In our implementation of Algorithms 1 and 2, we had to be aware of the fact, that the evaluation of $f(c)$ must be carried out in interval arithmetic to bound all rounding errors. Moreover, in practical computations we have $S \supset s_f(c, Y)$, i.e. S is possibly an overestimation of the true interval slope. Therefore, the formulas for computing the values p and q in the pruning step differ from the theoretical ones, and the machine versions of Theorems 1–4 need a somewhat different proof. Also, we have to take special care of correct rounding when computing p and q in the pruning step and of the correct use of interval evaluations instead of real evaluations where necessary. Details on our practical realization and implementation of the interval slope arithmetic and the algorithm itself are treated in [14].

To test our new slope pruning technique we used an implementation within the toolbox environment from [5]. We implemented Algorithm 2 together with an interval slope arithmetic to compute the interval slopes be means of an automatic differentiation process. Note that the version of the slope arithmetic used for the test does not include special techniques to make use of locally convex or concave parts of the elementary functions like $\sin x$, $\cos x$, etc. at the moment (see the definition

of $s_\varphi(U_c, U_x)$ at the end of Section 2). So we should keep in mind that the results for the slope enclosures can (in parts) be improved further.

7. Test results

7.1. SMOOTH PROBLEMS

In the following examples we demonstrate that (for smooth problems) the method using the pruning technique is superior to a similar method using first order information in form of derivative evaluations. We compare the pruning method with a corresponding method using the monotonicity test. For this alternative method we used Algorithm 2 with the modification that Steps 2, 6, and 10 were replaced by

2. $F_X := (f(c) + F'(X) \cdot (X - c)) \cap F(X)$;
6. CheckMonotonicity $(Y,\ c,\ F'(Y),\ X,\ U_1,\ U_2)$;
10. $F_U := (f(c) + F'(U_i) \cdot (U_i - c)) \cap F(U_i)$;

The used algorithm for CheckMonotonicity is given by

ALGORITHM 4. CheckMonotonicity (Y, c, D, X, U_1, U_2)

1. $U_1 := \emptyset$; $U_2 := \emptyset$;
2. **if** $0 \in D$ **then**
3. $U_1 := [\underline{y}, c]$; $U_2 := [c, \overline{y}]$; {bisection}
4. **else if** $\underline{d} > 0$ **and** $\underline{x} = \underline{y}$ **then**
5. $U_1 := [\underline{y}, \underline{y}]$; {reduce to left boundary point}
6. **else if** $\overline{d} < 0$ **and** $\overline{x} = \overline{y}$ **then**
7. $U_2 := [\overline{y}, \overline{y}]$; {reduce to right boundary point}
8. **return** U_1, U_2;

Note, that in Algorithm 4 we must take care of the boundary points of the original search region X which may be global minimizers without being stationary points.

Now, we compare the two methods for two examples.

EXAMPLE 7. We minimize $f(x) = x^2/20 - \cos(x) + 2$.

Applying our model algorithm *using derivatives and monotonicity tests*, we get

```
Search interval         : [-20,20]
Tolerance (relative)    :     1E-8
No. of function calls   :      150
No. of derivative calls :       75
No. of bisections       :       37
Necessary list length   :        2
Run-time (in sec.)      :    0.700
Global minimizer in     : [-7.6293945312500E-005, 7.6293945312500E-005 ]
Global minimum value in : [ 1.0000000000000E+000, 1.0000000000001E+000 ]
```

Applying Algorithm 2 *using slopes and the new pruning steps*, we get

```
Search interval         : [-20,20]
Tolerance (relative)    :     1E-8
No. of function calls   :       58
No. of slope calls      :       29
No. of bisections       :        1
Necessary list length   :        2
Run-time (in sec.)      :    0.270
Global minimizer in     : [-7.4615903941273E-005, 7.4615903941273E-005 ]
Global minimum value in : [ 1.0000000000000E+000, 1.0000000000001E+000 ]
```

EXAMPLE 8. We minimize $f(x) = 24x^4 - 142x^3 + 303x^2 - 276x + 93$, given by E. Hansen [6].

Applying our model algorithm *using derivatives and monotonicity tests*, we get

```
Search interval         : [0,3]
Tolerance (relative)    :  1E-8
No. of function calls   :   956
No. of derivative calls :   478
No. of bisections       :   238
Necessary list length   :    25
Run-time (in sec.)      : 0.450
Global minimizer in     : [ 1.999990940E+000, 2.000009536E+000 ]
Global minimum value in : [ 9.999999982E-001, 1.000000001E+000 ]
```

Applying Algorithm 2 *using slopes and the new pruning steps*, we get

```
Search interval         : [0,3]
Tolerance (relative)    :  1E-8
No. of function calls   :   488
No. of slope calls      :   244
No. of bisections       :    12
Necessary list length   :    15
Run-time (in sec.)      : 0.180
Global minimizer in     : [ 1.999997019E+000, 2.000009781E+000 ]
Global minimum value in : [ 9.999999996E-001, 1.000000001E+000 ]
```

We carried out further numerical tests for smooth problems on a HP 9000/730, and we used the set of 20 test functions given in the appendix. In Table 1, we give an overview of the results obtained with $\varepsilon = 10^{-8}$. For each test function we list the number of function evaluations, the number of derivative or slope evaluations, the number of bisections, the necessary list length and the required CPU time. The columns with the header M contain the values for the variant with Monotonicity test, those with the header P contain the values for the variant with slope Pruning step, and those with the header (P/M) contain the percentage of the slope method

Table 1. Results for the complete test set when using the monotonicity test (M) and the slope pruning step (P), respectively

	F eval.			D or S eval.			Bisections			List length			Execution time		
no.	M	P	(P/M)	M	P	(P/M)	M	P	(P/M)	M	P	(P/M)	M	P	(P/M)
1	142	98	(69%)	71	49	(69%)	35	13	(37%)	5	4	(80%)	0.930	0.770	(83%)
2	346	314	(91%)	173	157	(91%)	86	5	(6%)	17	18	(106%)	8.390	8.320	(99%)
3	194	154	(79%)	97	77	(79%)	48	7	(15%)	4	4	(100%)	1.030	0.840	(82%)
4	142	116	(82%)	71	58	(82%)	35	8	(23%)	3	4	(133%)	2.450	2.070	(84%)
5	526	400	(76%)	263	200	(76%)	131	9	(7%)	9	9	(100%)	3.180	2.220	(70%)
6	1234	618	(50%)	617	309	(50%)	308	6	(2%)	29	16	(55%)	0.580	0.220	(38%)
7	956	488	(51%)	478	244	(51%)	238	12	(5%)	25	15	(60%)	0.450	0.180	(40%)
8	82	82	(100%)	41	41	(100%)	20	5	(25%)	4	4	(100%)	1.100	1.080	(98%)
9	106	90	(85%)	53	45	(85%)	26	10	(38%)	4	3	(75%)	0.230	0.190	(83%)
10	106	98	(92%)	53	49	(92%)	26	9	(35%)	5	5	(100%)	0.120	0.090	(75%)
11	310	162	(52%)	155	81	(52%)	77	7	(9%)	10	7	(70%)	0.360	0.140	(39%)
12	150	58	(39%)	75	29	(39%)	37	1	(3%)	2	2	(100%)	0.700	0.270	(39%)
13	66	56	(85%)	33	28	(85%)	16	11	(69%)	2	2	(100%)	0.020	0.010	(50%)
14	166	78	(47%)	83	39	(47%)	41	1	(2%)	2	2	(100%)	0.750	0.320	(43%)
15	78	62	(79%)	39	31	(79%)	19	9	(47%)	2	3	(150%)	0.380	0.330	(87%)
16	142	54	(38%)	71	27	(38%)	35	1	(3%)	2	2	(100%)	0.260	0.100	(38%)
17	308	144	(47%)	154	72	(47%)	76	10	(13%)	11	7	(64%)	0.750	0.360	(48%)
18	582	252	(43%)	291	126	(43%)	145	3	(2%)	12	6	(50%)	0.240	0.090	(38%)
19	98	82	(84%)	49	41	(84%)	24	7	(29%)	5	4	(80%)	0.500	0.480	(96%)
20	78	70	(90%)	39	35	(90%)	19	9	(47%)	2	2	(100%)	0.030	0.020	(67%)
$\sum$	5812	3476	(60%)	2906	1738	(60%)	1442	143	(10%)	155	119	(77%)	22.45	18.10	(81%)
Ø			(69%)			(69%)			(21%)			(91%)			(65%)

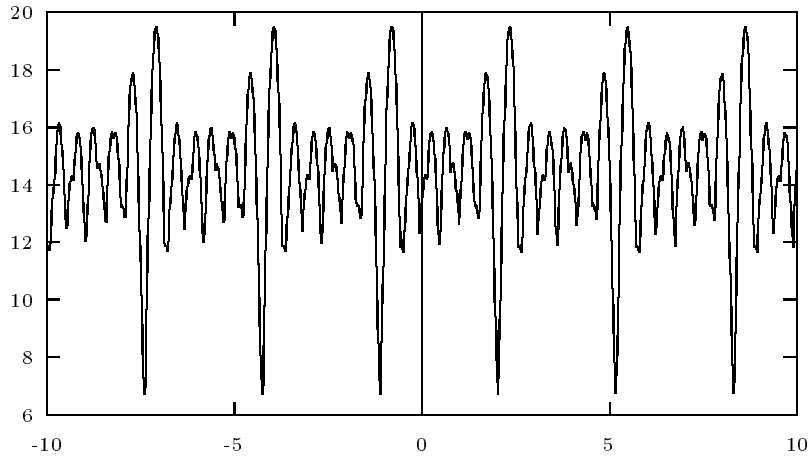
with respect to the derivative method. The last row of the table gives average values for the complete test set.

According to our numerical tests, our new method is always better than or at least as good as the traditional method with monotonicity test (with the exception of few cases where the list length was worse). On average, we have more than 30% improvement in the computation time and the number of function or derivative/slope evaluations, if we use our new pruning step. Moreover, there are many examples for which the required CPU time is reduced to around 1/3 of the time required by the old variant.

7.2. NONSMOOTH PROBLEMS

The most important fact is that the pruning technique is also applicable to nonsmooth problems, because interval slopes are computable for nondifferentiable functions, too. Therefore, we are able to use first-order information of the function within a global optimization method when applying it to nondifferentiable functions. We conclude this section by some nonsmooth examples.

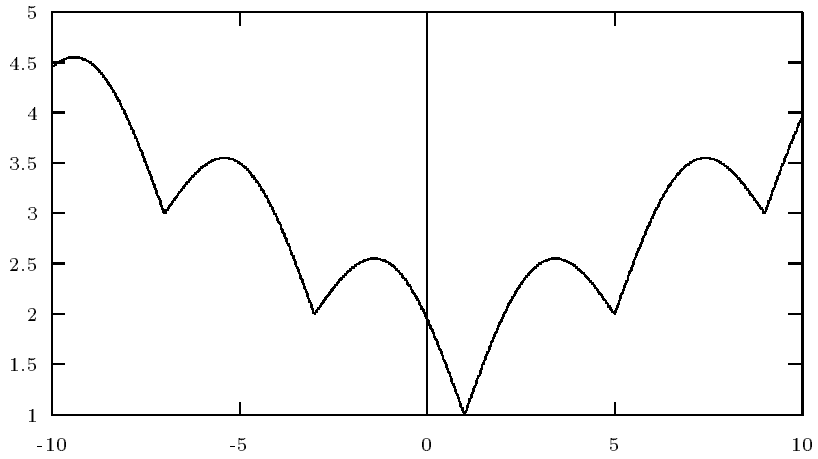
EXAMPLE 9. We minimize $f(x) = \sum_{k=1}^{5} k|\cos((k+1)x+k)| + 5$.



With Algorithm 2 we obtain:

```
Search interval         : [-10,10]
Tolerance (relative)    :     1E-8
No. of function eval.   :      412
No. of slope eval.      :      206
No. of bisections       :        6
Necessary list length   :       31
Run-time (in sec.)      :    0.430
Global minimizer in     : [ -7.397344586683E+000, -7.397344529213E+000 ]
Global minimizer in     : [ -4.255751923713E+000, -4.255751914035E+000 ]
Global minimizer in     : [ -1.114159270123E+000, -1.114159265284E+000 ]
Global minimizer in     : [  2.027433383466E+000,  2.027433388306E+000 ]
Global minimizer in     : [  5.169026037056E+000,  5.169026085146E+000 ]
Global minimizer in     : [  8.310618653412E+000,  8.310618700324E+000 ]
Global minimum value in : [  6.699793627703E+000,  6.699793776608E+000 ]
```

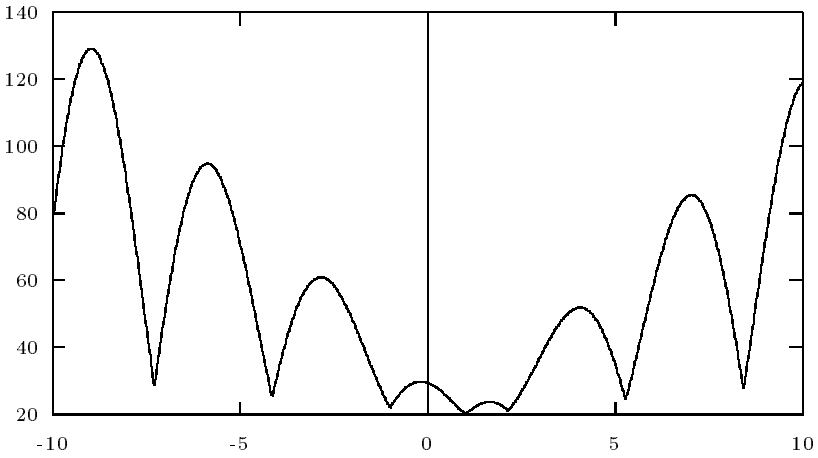
EXAMPLE 10. We minimize $f(x) = |(x-1)/4| + |\sin(\pi(1+(x-1)/4))| + 1$.



With Algorithm 2 we obtain:

```
Search interval         : [-10,10]
Tolerance (relative)    :     1E-8
No. of function eval.   :       38
No. of slope eval.      :       19
No. of bisections       :        5
Necessary list length   :        2
Run-time (in sec.)      :    0.020
Global minimizer in     : [  9.999999997437E-001,  1.000000000257E+000 ]
Global minimum value in : [  1.000000000000E+000,  1.000000000001E+000 ]
```

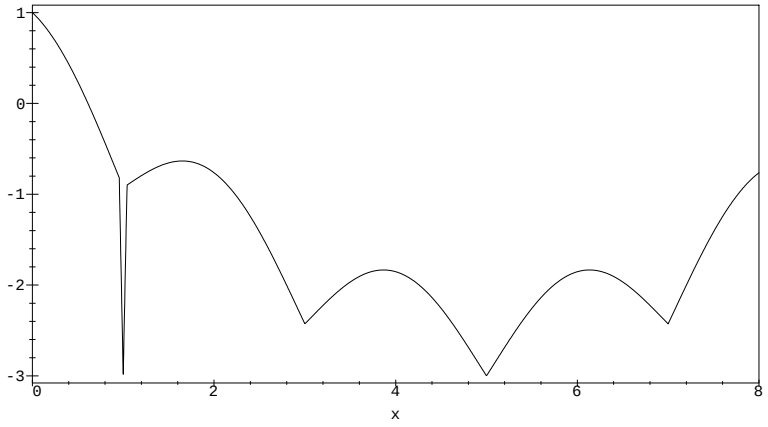
EXAMPLE 11. We minimize $f(x) = \min\{|\cos(\pi x/2)| - 3\sin(\pi x/10), 50|x - 1| - 3\}$.



With Algorithm 2 we obtain:

```
Search interval         : [0,8]
Tolerance (relative)    :  1E-8
No. of function eval.   :    58
No. of slope eval.      :    29
No. of bisections       :     5
Necessary list length   :     4
Run-time (in sec.)      : 0.040
Global minimizer in     : [  1.000000000000E+000,  1.000000000000E+000 ]
Global minimizer in     : [  4.999999986829E+000,  5.000000000008E+000 ]
Global minimum value in : [ -3.000000000000E+000, -3.000000000000E+000 ]
```

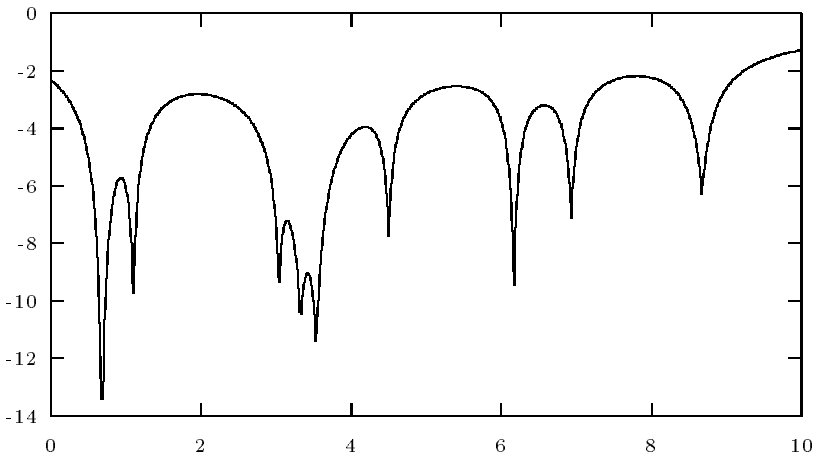
EXAMPLE 12. We minimize $f(x) = -\sum_{i=1}^{10} \frac{1}{|k_i(x - a_i)| + c_i}$ with a, c, and k as given for the smooth test function number 10 given in the appendix.



With Algorithm 2 we obtain:

```
Search interval         : [0,10]
Tolerance (relative)    :   1E-8
No. of function eval.   :    114
No. of slope eval.      :     57
No. of bisections       :     18
Necessary list length   :      6
Run-time (in sec.)      :  0.180
Global minimizer in     : [  6.739999979030E-001,  6.740000037738E-001 ]
Global minimum value in : [ -1.401836081085E+001, -1.401836055728E+001 ]
```

EXAMPLE 13. We minimize $f(x) = |x - 1|(1 + 10|\sin(x + 1)|) + 1$.



With Algorithm 2 we obtain:

```
Search interval         : [-10,10]
Tolerance (relative)    :     1E-8
No. of function eval.   :       44
No. of slope eval.      :       22
No. of bisections       :        6
Necessary list length   :        2
Run-time (in sec.)      :    0.020
Global minimizer in     : [  9.999999998471E-001,  1.000000000222E+000 ]
Global minimum value in : [  1.000000000000E+000,  1.000000000322E+000 ]
```

8. Summary

In this paper we introduced a new pruning technique based on interval slopes which can be very successfully applied in interval branch-and-bound methods for global optimization. We showed, that for smooth problems this technique can replace the often used monotonicity test and we showed the practical effect of such a replacement, ie. a considerable improvement in efficiency for our global optimization algorithm.

Moreover, since interval slopes are computable for non-differentiable functions too, the new pruning technique is also applicable to nonsmooth problems. Therefore, we are able to use first-order information of the function within a global optimization method when applying it to non-differentiable functions.

Appendix: Smooth test functions, search boxes, and solutions

The test set of smooth problems consists of one-dimensional problems collected in [16] (which "well represent practical problems"), problems presented in [4] and [6], and some new problems.

1. $f(x) = (x + \sin x) \cdot \mathrm{e}^{-x^2}$, $X = [-10, 10]$, (taken from [16])
   $X^* = \{-0.6795...\}$, $f^* = -0.8242....$
2. $f(x) = -\sum_{k=1}^{5} k \sin((k+1)x + k)$, $X = [-10, 10]$, (taken from [16])
   $X^* = \{-6.774..., -0.4913..., 5.791...\}$, $f^* = -12.03....$
3. $f(x) = \sin x$, $X = [0, 20]$,
   $X^* = \{\frac{3\pi}{2}, \frac{7\pi}{2}, \frac{11\pi}{2}\}$, $f^* = -1$.
4. $f(x) = \mathrm{e}^{-3x} - \sin^3 x$, $X = [0, 20]$, (taken from [4])
   $X^* = \{\frac{9\pi}{2}\}$, $f^* = \mathrm{e}^{\frac{27\pi}{2}} - 1$.
5. $f(x) = \sin \dfrac{1}{x}$, $X = [0.02, 1]$,
   $X^* = \{\frac{2}{(4k-1)\pi} \mid k = 1, \ldots, 6\}$, $f^* = -1$.
6. $f(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$, $X = [-10, 20]$,
   $X^* = \{\frac{5-\sqrt{5}}{2}, \frac{5+\sqrt{5}}{2}\}$, $f^* = -1$.
7. $f(x) = 24x^4 - 142x^3 + 303x^2 - 276x + 93$, $X = [0, 3]$, (taken from [6])
   $X^* = \{2\}$, $f^* = 1$.
8. $f(x) = \sin x + \sin \dfrac{10x}{3} + \ln x - 0.84x$, $X = [2.7, 7.5]$, (taken from [16])
   $X^* = \{5.1997...\}$, $f^* = -4.601....$
9. $f(x) = 2x^2 - \dfrac{3}{100}\mathrm{e}^{-(200(x-0.0675))^2}$, $X = [-10, 10]$, (taken from [4])
   $X^* = \{0.06738...\}$, $f^* = -0.02090$.
10. $f(x) = -\sum_{i=1}^{10} \dfrac{1}{(k_i(x - a_i))^2 + c_i}$, $X = [0, 10]$, (taken from [16])
    $a = (3.040, 1.098, 0.674, 3.537, 6.173, 8.679, 4.503, 3.328, 6.937, 0.700)$,
    $k = (2.983, 2.378, 2.439, 1.168, 2.406, 1.236, 2.868, 1.378, 2.348, 2.268)$,
    $c = (0.192, 0.140, 0.127, 0.132, 0.125, 0.189, 0.187, 0.171, 0.188, 0.176)$,
    $X^* = \{0.6858...\}$, $f^* = -14.59....$
11. $f(x) = -\sum_{i=1}^{10} \dfrac{1}{(k_i(x - a_i))^2 + c_i}$, $X = [0, 10]$, (taken from [16])
    $a = (4.696, 4.885, 0.800, 4.986, 3.901, 2.395, 0.945, 8.371, 6.181, 5.713)$,
    $k = (2.871, 2.328, 1.111, 1.263, 2.399, 2.629, 2.853, 2.344, 2.592, 2.929)$,
    $c = (0.149, 0.166, 0.175, 0.183, 0.128, 0.117, 0.115, 0.148, 0.188, 0.198)$,

$X^* = \{4.855...\}, \quad f^* = -13.92....$

12. $f(x) = \dfrac{x^2}{20} - \cos x + 2, \quad X = [-20, 20],$
$X^* = \{0\}, \quad f^* = 1.$
13. $f(x) = -\dfrac{1}{(x-2)^2 + 3}, \quad X = [0, 10],$
$X^* = \{2\}, \quad f^* = \frac{1}{3}.$
14. $f(x) = x^2 - \cos(18x), \quad X = [-5, 5],$
$X^* = \{0\}, \quad f^* = -1.$
15. $f(x) = (x-1)^2(1 + 10\sin^2(x+1)) + 1, \quad X = [-10, 10],$
$X^* = \{1\}, \quad f^* = 1.$
16. $f(x) = \mathrm{e}^{x^2}, \quad X = [-10, 10],$
$X^* = \{0\}, \quad f^* = 1.$
17. $f(x) = x^4 - 12x^3 + 47x^2 - 60x - 20\mathrm{e}^{-x}, \quad X = [-1, 7],$ (taken from [4])
$X^* = \{0.7136...\}, \quad f^* = -32.78....$
18. $f(x) = x^6 - 15x^4 + 27x^2 + 250, \quad X = [-4, 4],$ (taken from [6])
$X^* = \{-3, 3\}, \quad f^* = 7.$
19. $f(x) = \sin^2\left(1 + \dfrac{x-1}{4}\right) + \left(\dfrac{x-1}{4}\right)^2, \quad X = [-10, 10],$
$X^* = \{-0.7878...\}, \quad f^* = 0.4756....$
20. $f(x) = (x - x^2)^2 + (x-1)^2, \quad X = [-10, 10],$
$X^* = \{1\}, \quad f^* = 0.$

Acknowledgments

Many thanks go to an anonymous referee for drawing our attention to the related paper [7].

References

1. Alefeld, G. and Herzberger, J. (1983), *Introduction to Interval Computations*, Academic Press, New York.
2. Bromberg, M. and Chang, T.S. (1992), One-dimensional global optimization using linear lower bounds, in: C.A. Floudas and P.M. Pardalos (eds.), *Recent Advances in Global Optimization*, Princeton University Press, Princeton.
3. Csendes, T. and Pintér, J. (1993), The Impact of accelerating tools on the interval subdivision algorithm for global optimization, *European J. of Operational Research* 65: 314–320.
4. Hammer, R., Hocks, M., Kulisch, U. and Ratz, D. (1993), *Numerical Toolbox for Verified Computing I*, Springer-Verlag, Berlin.
5. Hammer, R., Hocks, M., Kulisch, U. and Ratz, D. (1995), *C++ Toolbox for Verified Computing I*, Springer-Verlag, Berlin.

6. Hansen, E. (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker, New York.
7. Hansen, P., Jaumard, B. and Xiong, J. (1994), Cord-slope form of taylor's expansion in univariate global optimization, *Journal of Optimization Theory and Applications* 80: 441–464.
8. Kearfott, R.B. (1996), *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Boston.
9. Krawczyk, R. and Neumaier, A. (1985), Interval slopes for rational functions and associated centered forms, *SIAM Journal on Numerical Analysis* 22: 604–616.
10. Neumaier, A. (1990), *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge.
11. Ratschek, H. and Rokne, J. (1988), *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester.
12. Ratz, D. (1992), *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*, Dissertation, Universität Karlsruhe.
13. Ratz, D. and Csendes, T. (1995), On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *Journal of Global Optimization* 7: 183–207.
14. Ratz, D. (1996), An optimized interval slope arithmetic and its application. *Forschungsschwerpunkt Computerarithmetik, Intervallrechnung und Numerische Algorithmen mit Ergebnisverifikation*, Bericht 4/1996.
15. Rump, S.M. (1996), Expansion and estimation of the range of nonlinear functions, *Mathematics of Computations* 65: 1503–1512.
16. Törn, A. and Žilinskas, A. (1989), *Global Optimization*, Lecture Notes in Computer Science, No. 350, Springer Verlag, Berlin.