Developments in Global Optimization
Bomze, Csendes, Horst, Pardalos (Eds.)
pp. 55–72

New Results on Gap-Treating Techniques in Extended Interval Newton Gauss-Seidel Steps for Global Optimization

DIETMAR RATZ dietmar.ratz@math.uni-karlsruhe.de
Institut für Angewandte Mathematik, Universität Karlsruhe, D-76128 Karlsruhe, Germany

Abstract. Interval-branch-and-bound methods for global optimization very often incorporate interval Newton Gauss-Seidel steps to reduce the widths of the boxes resulting from the basic branch-and-bound method. These steps try to determine the roots of the gradient of the objective function, whereas various other techniques eliminate the regions containing roots which do not correspond to global optimizers.

The interval Newton Gauss-Seidel step uses so-called extended interval arithmetic which allows the division by intervals containing zero. The latter may produce gaps in the resulting coordinate intervals, which can be used to split the resulting box of the interval Gauss-Seidel step.

We investigate the impact of gap-treating and box-splitting techniques which make use of branching rules, i.e. rules for selecting the subdivision direction in the underlying branch-and-bound method. Supplementing earlier studies ([3], [12]), the investigated model algorithm (similar to that in [5]) now uses the enclosure of the Hessian matrix to incorporate a second-order branching rule. We propose a strategy, a sorted interval Gauss-Seidel step, which improves the overall efficiency of the interval Newton Gauss-Seidel step and therefore of our global optimization method. We present results of computational experiments with standard global optimization problems.

Keywords: Global optimization, interval arithmetic, branch-and-bound, interval Newton Gauss-Seidel step

1. Introduction

Let $f : D \to \mathbb{R}$ be a twice continuously differentiable function, and let $D \supseteq [x] \in I\mathbb{R}^n$. We address the problem of finding all points x^* in the interval vector $[x]$ such that

$$f(x^*) = \min_{x \in [x]} f(x).$$

We are interested in both the global minimizers x^* and the minimum value $f^* = f(x^*)$.

We use the branch-and-bound approach described in [5] and [11] with several modifications. Our method starts from an initial box $[x] \in I\mathbb{R}^n$, subdivides $[x]$, stores the subboxes in a list L, and discards subintervals which are guaranteed not to contain a global minimizer, until the desired accuracy (width) of the interval vectors in the list is achieved. The tests we use to discard or to prune pending subboxes are cut-off test, monotonicity test, concavity test, and interval Newton Gauss-Seidel step. For details on these tests and on the method itself, see [5].

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,

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

We denote real numbers by $x, y, \ldots$ and real bounded and closed interval vectors 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 $I\mathbb{R} := \{[\underline{a}, \overline{a}] \mid \underline{a} \leq \overline{a},\ \underline{a}, \overline{a} \in \mathbb{R}\}$ and the set of n-dimensional interval vectors (also called boxes) by $I\mathbb{R}^n$. For real vectors and interval vectors the notations

$$x = (x_i), \quad x_i \in \mathbb{R}, \qquad \text{and} \qquad [x] = ([x]_i), \quad [x]_i \in I\mathbb{R}$$

are used.

The diameter (or width) of the interval $[x]$ is defined by $d([x]) = \overline{x} \Leftrightarrow \underline{x}$ if $[x] \in I\mathbb{R}$. The midpoint of the interval $[x]$ is defined by $m([x]) = (\underline{x} + \overline{x})/2$ if $[x] \in I\mathbb{R}$, and $m([x]) = (m([x]_i))$, if $[x] \in I\mathbb{R}^n$.

We call a function $F : I\mathbb{R}^n \to I\mathbb{R}$ an *inclusion function* of $f : \mathbb{R}^n \to \mathbb{R}$ in $[x] \in I\mathbb{R}^n$, if $x \in [x]$ implies $f(x) \in F([x])$. In other words, $f_{\mathrm{rg}}([x]) \subseteq F([x])$, where $f_{\mathrm{rg}}([x])$ is the range of the function f on $[x]$. The inclusion function of the gradient of f and the Hessian of f are denoted by $\boldsymbol{\nabla} F$ and $\boldsymbol{\nabla}^2 F$. It is assumed in the following that the inclusion functions have the *isotonicity* property, i.e. $[x] \subseteq [y]$ implies $F([x]) \subseteq F([y])$.

Moreover, we use the following notations for $[a] \in I\mathbb{R}$ and $[x] \in I\mathbb{R}^n$:

$$\begin{aligned}
&\overset{\circ}{[a]} && := \{a \in [a] \mid \underline{a} < a < \overline{a}\} && \text{(Interior of } [a]\text{)},\\
&\overset{\circ}{[x]} && := \{x \in [x] \mid \underline{x_i} < x_i < \overline{x_i} \ \text{ for all } i\} && \text{(Interior of } [x]\text{)},\\
&\partial([a]) && := \{\underline{a}, \overline{a}\} && \text{(Boundary of } [a]\text{)},\\
&\partial([x]) && := \{x \in [x] \mid x_i \in \partial([x]_i) \ \text{ for any } i\} && \text{(Boundary of } [x]\text{)},\\
&[x] \overset{\circ}{\subset} [y] && \Leftrightarrow \ [x] \subseteq \overset{\circ}{[y]} && \text{(Inner inclusion)}.
\end{aligned}$$

Throughout the whole paper, we assume that there exists a stationary point $x^* \in [x]$ for which $f(x^*) = f^*$, since we do not do anything special to handle boundary points in these studies.

2. Main Global Optimization Algorithm

In the following, we give a simplified algorithmic description and an overview of our global optimization method. We use the notations from [5].

ALGORITHM 1. GlobalOptimize $(F, [x], \varepsilon, L_{\rm res}, [f^*])$

1. $\widetilde{f} := \overline{F}(m([x]))$; $[y] := [x]$; $L := \{\}$; $L_{\rm res} := \{\}$;
2. **repeat**
   (A) FindComponents $([y], k_1, k_2)$; Branch $([y], k_1, k_2, [U]_1, [U]_2, [U]_3, [U]_4)$;
   (B) **for** $i := 1$ **to** 4 **do**
      i. **if** $\widetilde{f} < \underline{F}([U]_i)$ **then next**$_i$;
      ii. **if** MonotonicityTest $(\nabla F([U]_i))$ **then next**$_i$;
      iii. **if** ConcavityTest $(\nabla^2 F([U]_i))$ **then next**$_i$;
      iv. IntervalNewtonGaussSeidelStep $(F, [U]_i, \nabla^2 F([U]_i), [V], p)$;
      v. **for** $j := 1$ **to** p **do if** $\widetilde{f} \geq \underline{F}([V]_j)$ **then** $L := L \uplus ([V]_j, \underline{F_V})$;
   (C) **while** $(L \neq \{\})$ **do**
      i. $([y], \underline{F_y}) :=$ PopHead (L);
      ii. $\widetilde{f} := \min\{\widetilde{f}, \overline{F}(m([y]))\}$; CutOffTest $(L, \widetilde{f})$;
      iii. **if** Accept $(F, [y], \varepsilon)$ **then** $L_{\rm res} := L_{\rm res} \uplus ([y], \underline{F_y})$ **else goto** 2(a);
   until $(L = \{\})$;
3. $([y], \underline{F_y}) :=$ Head $(L_{\rm res})$; $[f^*] := [\underline{F_y}, \widetilde{f}]$; **return** $L_{\rm res}$, $[f^*]$;

Algorithm 1 first computes an upper bound $\widetilde{f}$ for the global minimum value and initializes the working list L and the result list $L_{\rm res}$. The main iteration (Step 2) starts with a multisection of $[y]$. Then we apply a range check, the monotonicity test, the concavity test, and the interval Newton step to the multisected boxes $[U_1]$, $[U_2]$, $[U_3]$, and $[U_4]$. The interval Newton step results in p boxes, to which we apply a range check. If the current box $[V]_j$ is still a candidate for a minimizer, we store it in L in Step 2(B)v. Note that the boxes are stored as pairs $([y], \underline{F_y})$ in list L sorted in *nondecreasing* order with respect to the lower bounds $\underline{F_y} = \underline{F([y])}$ and in *decreasing* order with respect to the ages of the boxes in L (cf. [11]).

In Step 2(C), we remove the first element from the list L, i.e. the element of L with the smallest $\underline{F_y}$ value, and we perform the cut-off test. Then, if the desired accuracy is achieved for $[y]$, we store $[y]$ in the result list $L_{\rm res}$. Otherwise, we go to the branching step. When the iteration stops because the pending list L is empty, we compute a final enclosure $[f^*]$ for the global minimum value and return $L_{\rm res}$ and $[f^*]$.

The method can be improved by incorporating an approximate local search procedure to try to decrease the value $\widetilde{f}$. See [7] for the description of such local search procedures. For our studies in this paper, we do not apply any local method. We also do not apply any boundary treating, so we assume that all x^* lie in the interior of $[x]$.

3. Use of Branching Rules

As demonstrated in [3] and [12], the determination of "optimal" components for subdividing the current box $[y]$ in Step 2(A) of Algorithm 1 plays an important role. Moreover, the corresponding rules for selecting the subdivision direction can also be helpful in connection with the interval Newton Gauss-Seidel step, as we shall see later.

In Algorithm 1, a multisection is used, so each of these branching rules selects directions k_1 and k_2 with $D(k_1) \geq D(k_2) \geq D(i)$ for all $i = 1, \ldots, n$ and $i \notin \{k_1, k_2\}$, where $D(i)$ is fixed by the given rule. For the current study, we investigate four rules (we leave out Rule D from [3] and [12]):

Rule A: $D(i) := d([y]_i)$

Rule B: $D(i) := d(g_i([y])) \cdot d([y]_i)$ (cf. [7])

Rule C: $D(i) := d\left(g_i([y]) \cdot ([y]_i \Leftrightarrow c_i)\right)$ (cf. [10])

Rule E: $D(i) := d\left(([y]_i \Leftrightarrow c_i) \cdot \left(G_i(c) + \frac{1}{2}\sum_{j=1}^{n}(H_{ij}([y]) \cdot ([y]_i \Leftrightarrow c_i))\right)\right)$.

Here, $G = \boldsymbol{\nabla} F$, $H = \boldsymbol{\nabla}^2 F$, and $c = m([y])$.

Similar to Rule C (cf. [10]), the underlying idea of the new Rule E is to minimize

$$\begin{aligned} d(F([y])) &= d(F([y]) \Leftrightarrow f(c)) \\ &\approx d\left(([y] \Leftrightarrow c)^T \cdot \left(\boldsymbol{\nabla} f(c) + \frac{1}{2}\boldsymbol{\nabla}^2 F([y]) \cdot ([y] \Leftrightarrow c)\right)\right) \\ &= d\sum_{i=1}^{n}\left(([y]_i \Leftrightarrow c_i) \cdot \left(\frac{\partial F}{\partial x_i}([y]) + \frac{1}{2}\sum_{j=1}^{n}\frac{\partial^2 F([y])}{\partial x_i \partial x_j} \cdot ([y]_j \Leftrightarrow c_j)\right)\right). \end{aligned}$$

The proofs of convergence of the underlying branch-and-bound (subdivision) algorithm with Rules A, B, and C can be found in [12], the proof for Rule E (recently proposed in [13]) can be found in [2].

4. Interval Newton Gauss-Seidel Step (INGSS)

In Algorithm 1, we apply one step of the extended interval Newton Gauss-Seidel method (cf. [1]) to the nonlinear system $\boldsymbol{\nabla} f(y) = 0$ with $y \in [y]$. The subbox $[y]$ is a candidate box for enclosing a minimizer x^*, which we have assumed must satisfy $\boldsymbol{\nabla} f(x^*) = 0$. One step of the extended interval Newton Gauss-Seidel method shall improve (prune) the enclosure $[y]$ by formally solving the system $g = [H] \cdot (c \Leftrightarrow y)$, where $c = m([y])$, $g = \boldsymbol{\nabla} f(c)$, and $[H] = \boldsymbol{\nabla}^2 F([y])$.

Usually, this method works better if we first apply a *preconditioning*, by using a special matrix $R \in \mathbb{R}^{n \times n}$ for computing $b := R \cdot g$ and $[A] := R \cdot [H]$. Then

we consider the system $b = [A] \cdot (c \Leftrightarrow y)$, and we compute the new box $N'_{\mathrm{GS}}([y])$ according to

$$
\begin{aligned}
&[z] := [y], \\
&[z]_i := \Big(c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} [A]_{ij} \cdot ([z]_j \Leftrightarrow c_j) \Big) \Big/ [A]_{ii} \Big) \cap [z]_i, \quad i = 1, \ldots, n, \\
&N'_{\mathrm{GS}}([y]) := [z].
\end{aligned}
$$

The interval Newton Gauss-Seidel step (abbreviated by INGSS) in this form (assuming that $0 \notin [A]_{ii}$) has the following properties (see [7] or [9] for proofs):

THEOREM 3.1 *Let $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function, and let $[x] \in I\mathbb{R}^n$ be an interval vector with $[x] \subseteq D$. Then $N'_{\mathrm{GS}}([x])$ has the following properties:*

1. *Every zero $x^* \in [x]$ of ∇f satisfies $x^* \in N'_{\mathrm{GS}}([x])$.*
2. *If $N'_{\mathrm{GS}}([x]) = \emptyset$, then there exists no zero of ∇f in $[x]$.*
3. *If $N'_{\mathrm{GS}}([x]) \overset{\circ}{\subset} [x]$, then there exists a unique zero of ∇f in $[x]$.*

In the one-dimensional case with $f : \mathbb{R} \to \mathbb{R}$ and $[y] \in I\mathbb{R}$, the interval Newton Gauss-Seidel step reduces to the interval Newton step

$$
N'([y]) := N'_{\mathrm{GS}}([y]) = \Big(c \Leftrightarrow \frac{f'(c)}{F''([y])} \Big) \cap [y]
$$

Using standard interval arithmetic, the interval Newton step assumes $0 \notin F''([y])$. Like in classical Newton's method, the interval Newton step can be geometrically interpreted as drawing two lines from the midpoint $(c, f'(c))$ and intersecting them with the x-axis. These lines have the slope $\underline{g}$ (a lower bound of the slopes of f' in $[y]$) and $\overline{g}$ (an upper bound of the slopes of f' in $[y]$), respectively, where $[g] = F''([y])$.

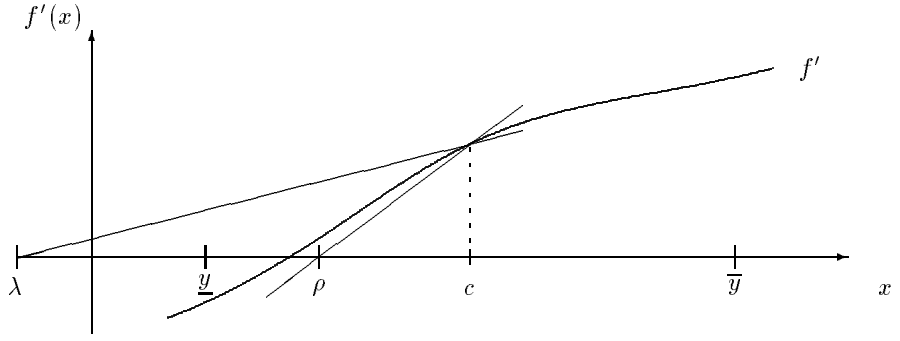

Figure 1. Interval Newton step with $0 \notin F''([y])$

The points of intersection with the x-axis, i.e. λ and ρ, form the new interval $[\lambda, \rho]$. Figure 1 demonstrates this interval Newton step resulting in

$$N'([y]) = [\lambda, \rho] \cap [y] = [\underline{y}, \rho].$$

If the intersection is empty, we know that there is no root of f' in $[y]$.

5. Box-Splitting and Gap-Treating

Using extended interval arithmetic (see [5] or [7] for details), we are able to treat the case $0 \in F''([y])$ that occurs, for example, if there are several zeros of f'' in the interval $[y]$. In this case, $N'([y])$ is given by one or two intervals resulting from the interval division.

In Figure 2, we illustrate one extended interval Newton step geometrically. Again we draw lines through the point $(c, f'(c))$. The first line with the smallest (negative lower bound) slope of f' in $[y]$ intersects the x-axis in point ρ. The line with the largest (positive upper bound) slope intersects the x-axis in point λ. Therefore, we get

$$N'([y]) = ([\Leftarrow\!\infty, \rho] \cup [\lambda, \infty]) \cap [y] = [\underline{y}, \rho] \cup [\lambda, \overline{y}],$$

and we "punched" out a gap in the original interval $[y]$ which is now split.

In the multi-dimensional case, we must apply extended interval arithmetic if $0 \in [A]_{ii}$ for some i. In this case, a gap can be produced in the corresponding components $[z]_i$ of $[z]$. Therefore, the interval Gauss-Seidel step may result in the union of several boxes $[V]_i \in I\!I\!R^n$, $i = 1, \ldots, p$, and we have $N'_{\mathrm{GS}}([y]) = [V]_1 \cup \ldots \cup [V]_p$, so $[V] \in I\!I\!R^{p \times n}$. Since it is not necessary to compute the $[y]_i$ in fixed order $i = 1, \ldots, n$ in a practical realization of the interval Newton Gauss-Seidel method, very often the Hansen/Greenberg realization [6] is used. That is, we first perform the single component steps of the Gauss-Seidel step for all i with $0 \notin [A]_{ii}$ and then for the remaining indices with $0 \in [A]_{ii}$ by using extended interval arithmetic.

Nevertheless, if $0 \in [A]_{ii}$ for *several* components i, then the extended interval divisions in the interval Newton Gauss-Seidel method possibly produce *several* gaps

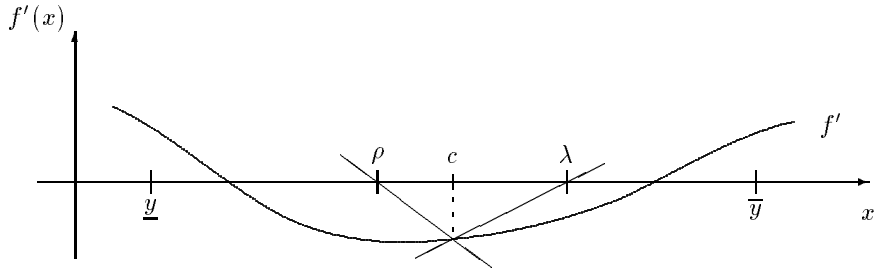

Figure 2. Extended interval Newton step with $0 \in F''([y])$

in the current box $[y]$. So we must split the result $N'_{\mathrm{GS}}([y])$ into two or more boxes. In this case, different splitting techniques may be applied resulting in different values for $[V]$ and p. We give two examples, which are used in the current study:

$p \leq 2$ *Compute all possible gaps in* $[y]$*, and finally use only the largest gap to split* $[y]$. This technique is known from Hansen/Greenberg [6], and the Newton step results in at most 2 boxes, thus $N'_{\mathrm{GS}}([x]) = [V]_1 \cup [V]_2$.

$p \leq n+1$ *Compute every gap, and use it* immediately *to split* $[y]$ *in a special way.* For this special splitting technique introduced in [10] the Newton step results in at most $n+1$ boxes, thus $N'_{\mathrm{GS}}([x]) = [V]_1 \cup \ldots \cup [V]_{n+1}$.

In our special technique with $p \leq n+1$, we use each gap to store one part of the current box $[y]$ by using one part of the component $[y]_i$ and to update $[y]$ with the other part of $[y]_i$, before continuing with the next component step of the interval Gauss-Seidel method. That is, we perform one component step according to the scheme:

1. Compute $[y]_i = [w] \cup [v]$.
2. If $[w] = [v] = \emptyset$, then stop {no solution in $[y]$}.
3. If $[v] \neq \emptyset$, then set $[y]_i := [v]$ and store $[y]$.
4. Set $[y]_i := [w]$ and continue with next i.

In some bad cases, we only get $\Big(b_i + \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} [A]_{ij} \cdot ([y]_j \Leftrightarrow c_j)\Big) \Big/ [A]_{ii} = (\Leftrightarrow\infty, \infty)$ and no gap occurs, so $[y]_i := [y]_i \cap (\Leftrightarrow\infty, \infty)$ remains unchanged. In these cases, we introduce "gaps" of width zero by splitting $[y]_i = [w] \cup [v]$ with $[w] := [\underline{y}_i, m([y]_i)]$ and $[v] := [m([y]_i), \overline{y}_i]$, that is we do a bisection.

6. Sorted Interval Newton Gauss-Seidel Step (SINGSS)

We investigate the branching rules applied in the main Algorithm 1 in connection with the interval Gauss-Seidel step. We use these rules to compute a sorting vector $s = (s_1, s_2, \ldots, s_n)$ with $s_i \in \{1, \ldots, n\}$ and $s_i \neq s_j$ for $i \neq j$, which satisfies $D(s_i) \geq D(s_{i+1})$, $i = 1, \ldots, n \Leftrightarrow 1$ for the corresponding direction selection rule $D(\ldots)$. Then, we perform the *sorted* interval Newton Gauss-Seidel step (SINGSS) according to

$$[z] := [y]$$

$$[z]_{s_i} := \Big(c_{s_i} \Leftrightarrow \Big(b_{s_i} + \sum_{\substack{j=1 \\ j \neq s_i}}^{n} [A]_{s_i,j} \cdot ([z]_j \Leftrightarrow c_j)\Big) \Big/ [A]_{s_i s_i}\Big) \cap [z]_{s_i}, \; i = 1, \ldots, n$$

$$N'_{\mathrm{SGS}}([y]) := [z]$$

incorporating the Hansen/Greenberg realization and different splitting techniques. Note that $N'_{\mathrm{SGS}}([y])$ is the union of several boxes in the general case. This SINGSS aims at splitting the box $[y]$ in those components first, which would be chosen for multisection by the specified branching rule.

We now give an algorithmic description of the SINGSS.

ALGORITHM 2. SIntervalNewtonGaussSeidelStep $(F, [y], [H], [V], p)$

1. Compute preconditioner R and sorting vector s;
2. $c := m([y]); \quad [A] := R \cdot [H]; \quad b := R \cdot \boldsymbol{\nabla} f(c); \quad p := 0;$
3. **for** $l := 1$ **to** n **do** {Component steps for $0 \notin [A]_{ii}$}
   (A) $i := s_l;$ **if** $(0 \in [A]_{ii})$ **then next**$_l$;
   (B) $[y]_i := \Big(c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} [A]_{ij} \cdot ([y]_j \Leftrightarrow c_j)\Big) \Big/ [A]_{ii}\Big) \cap [y]_i;$
   (C) **if** $[y]_i = \emptyset$ **then return** ;
4. **for** $l := 1$ **to** n **do** {Component steps for $0 \in [A]_{ii}$}
   (A) $i := s_l;$ **if** $(0 \notin [A]_{ii})$ **then next**$_l$;
   (B) $[w] \cup [v] := \Big(c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} [A]_{ij} \cdot ([y]_j \Leftrightarrow c_j)\Big) \Big/ [A]_{ii}\Big) \cap [y]_i;$
   (C) **if** $([w] = [v] = \emptyset)$ **then return** ;
   (D) **if** $([v] \neq \emptyset)$ **then** {Store part of $[y]$ in $[V]_p$}
   $[y]_i := [v]; \quad p := p + 1; \quad [V]_p := [y];$
   (E) $[y]_i := [w];$
5. $p := p + 1; \quad [V]_p := [y];$
6. **return** $[V], p;$ {Result: $N'_{\mathrm{SGS}}([y]) = \bigcup_{j=1}^{p} [V]_j$}

The following theorem summarizes the properties of our sorted interval Newton Gauss-Seidel step with special splitting.

THEOREM 4.1 *Let $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function, and let $[x] \in I\mathbb{R}^n$ be an interval vector with $[x] \subseteq D$. Then $N'_{\mathrm{SGS}}([x])$ computed by Algorithm 2 (including sorting and special splitting technique) has the following properties:*

1. *Every zero $x^* \in [x]$ of $\boldsymbol{\nabla} f$ satisfies $x^* \in N'_{\mathrm{SGS}}([x])$.*
2. *If $N'_{\mathrm{SGS}}([x]) = \emptyset$, then there exists no zero of $\boldsymbol{\nabla} f$ in $[x]$.*
3. *If $N'_{\mathrm{SGS}}([x]) \overset{\circ}{\subset} [x]$, then there exists a unique zero of $\boldsymbol{\nabla} f$ in $[x]$.*

Proof:

1. Let $, _i([y]) = \Big(c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} [A]_{ij} \cdot ([y]_j \Leftrightarrow c_j) \Big) \Big/ [A]_{ii} \Big) \cap [y]_i$ be the new value for $[y]_i$ computed in one component step of the SINGSS applied to $[y] \in I\!I\!R^n$ for $i \in \{1, \ldots, n\}$, and let $[y]^{(0)}$ be the updated value of $[y]$ after the complete Step 3 of Algorithm 2. Moreover, let $\mathcal{S} = \{s_{l_1}, s_{l_2}, \ldots s_{l_m}\}$ with $m \leq n$ be the set of those components of the sorting vector s for which extended interval division must be applied when computing $, _i$, that is, $0 \in [A_{ii}]$ for all $i \in \mathcal{S}$ and $, _i([y]) = [w] \cup [v]$ with $[w], [v] \in I\!I\!R$. For simplicity of the proof, we use $[v] = \emptyset$ if in fact no splitting occurs in $, _i([y])$, although denoting this a splitting.

   According to Algorithm 2, the result of the k-th extended component step of the SINGSS (i.e. Step 4) is given by

$$N'^{(k)}_{\mathrm{SGS}}([y]^{(k-1)} := [V]_k \cup [y]^{(k)},$$

   where

$$[V]_{ki} \cup [y]_i^{(k)} = , _i([y]^{(k-1)}) \quad \text{and} \quad i = s_{l_k},$$

   and where $[y]^{(k)}$ is the current value of $[y]$ after the k-th update in Step 4(E) of Algorithm 2, i.e. after k splittings. Then

$$[z]^{(k)} = [V]_1 \cup [V]_2 \cup \ldots \cup [V]_k \cup [y]^{(k)}$$

   is the current enclosure of the true solution set after k splittings, and with $[V]_{m+1} = [y]^{(m)}$ according to Step 5 of our algorithm we have

$$[z]^{(m)} = \bigcup_{j=1}^{m+1} [V]_j = N'_{\mathrm{SGS}}([x]).$$

   Now let $x^* \in [x]$ be a zero of $\boldsymbol{\nabla} f$.

   a) $x^* \in [y]^{(0)} = [z]^{(0)}$ according to Theorem 3.1.

   b) Given an arbitrary $k \in \{0, \ldots, m \Leftrightarrow 1\}$, we have two cases for $x^* \in [z]^{(k)}$:

      i) $x^* \in [V]_1 \cup \ldots \cup [V]_k$: In this case, it follows immediately that

$$x^* \in [V]_1 \cup \ldots \cup [V]_k \cup [V]_{k+1} \subseteq [z]^{(k+1)}.$$

      ii) $x^* \in [y]^{(k)}$: In this case, with $i = s_{l_{k+1}}$ there exists an $A^* \in [A]$ with $A^*(c \Leftrightarrow x^*) = b$.

For $A^*_{ii} \neq 0$ we get

$$\begin{aligned} x^*_i &= c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} A^*_{ij} \cdot (x^*_j \Leftrightarrow c_j) \Big) \Big/ A^*_{ii} \\ &\in c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} [A]_{ij} \cdot ([y]^{(k)}_j \Leftrightarrow c_j) \Big) \Big/ [A]_{ii}, \end{aligned}$$

and, since $x^*_i \in [y]^{(k)}_i$,

$$\begin{aligned} x^*_i &\in \Big(c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} [A]_{ij} \cdot ([y]^{(k)}_j \Leftrightarrow c_j) \Big) \Big/ [A]_{ii} \Big) \cap [y]^{(k)}_i \\ &= \, , _i([y]^{(k)}) \\ &= [V]_{k+1,i} \cup [y]^{(k+1)}_i, \end{aligned}$$

and thus $x^* \in [V]_{k+1} \cup [y]^{(k+1)} = N'^{(k+1)}_{\mathrm{SGS}}([y]^{(k)})$.

For $A^*_{ii} = 0$, a necessary consequence is that

$$\begin{aligned} 0 &= b_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} A^*_{ij} \cdot (x^*_j \Leftrightarrow c_j) \\ &\in b_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} [A]_{ij} \cdot ([y]^{(k)}_j \Leftrightarrow c_j), \end{aligned}$$

and the definition of extended interval division implies that

$$c_i \Leftrightarrow \Big(b_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} [A]_{ij} \cdot ([y]^{(k)}_j \Leftrightarrow c_j) \Big) \Big/ [A]_{ii} = (\Leftrightarrow\infty, \infty)$$

and thus

$$x^*_i \in [y]^{(k)} = (\Leftrightarrow\infty, \infty) \cap [y]^{(k)} = \, , _i([y]^{(k)}) = [V]_{k+1,i} \cup [y]^{(k+1)}_i.$$

Again we have $x^* \in [V]_{k+1} \cup [y]^{(k+1)} = N'^{(k+1)}_{\mathrm{SGS}}([y]^{(k)})$.

So, with $x^* \in [z]^{(k)}$, we have that $x^* \in [z]^{(k+1)}$ for $k = 0, \ldots, m \Leftrightarrow 1$.

Combining a) and b), $x^* \in [x]$ implies that $x^* \in [z]^{(m)} = N'_{\mathrm{SGS}}([x])$.

2. Assuming $x^* \in [x]$, $N'_{\mathrm{SGS}}([x]) = \emptyset$ contradicts Proposition 1 of the theorem.

3. According to the definition of extended interval operations (c.f. [5] or [7] for details), we know that $\partial([y]_i) \cap \, , _i([y]) \neq \emptyset$. That is, whenever extended interval division is applied $N'_{\mathrm{SGS}}([x]) \cap \partial([x]) \neq \emptyset$. Since $N'_{\mathrm{SGS}}([x]) \overset{\circ}{\subset} [x]$, we know that no extended interval operation occurred in the SINGSS, and therefore Proposition 3 of Theorem 3.1 completes the proof.

■

7. Numerical Experiences

For our tests, we used the group of test functions given in [12]. We carried out the numerical tests on an HP 9000/730 equipped with PASCAL–XSC [8] using the basic toolbox modules for automatic differentiation and extended interval arithmetic [5]. Our test suite compared the methods with branching rules A, B, C, and E combined with the usual splitting technique ($p \leq 2$) and with the special splitting technique ($p \leq n+1$, "0-width-gaps").

In the following, we list the complete results for 10 test problems. Important columns of the corresponding tables are the runtime (in STUs), the storage space or maximum list length (LL) and the Eeff_1 and Eeff_2 values. The latter combine the three values for the number of function (FE), gradient (GE), and Hessian (HE) evaluation to single values approximating the total evaluation effort in terms of objective function evaluations by

$$\text{Eeff}_1 = \text{FE} + n \cdot \text{GE} + \frac{n \cdot (n+1)}{2} \cdot \text{HE}$$

and

$$\text{Eeff}_2 = \text{FE} + \min\{4, n\} \cdot \text{GE} + n \cdot \text{HE}$$

(with respect to forward (Eeff_1) and backward (Eeff_2) mode of automatic differentiation, see [4] for details).

Results for problem **Shekel10** ($n = 4$)								
$p \leq$	Rule	STUs	FE	GE	HE	Eeff_1	Eeff_2	LL
2	A	2.09	132	106	42	976	724	17
	B	2.12	133	108	43	995	737	17
	C	1.68	112	86	32	776	584	15
	E	1.68	112	86	32	776	584	15
n+1	A	1.45	144	62	22	612	480	33
	B	1.71	169	70	26	709	553	39
	C	1.31	129	56	19	543	429	31
	E	1.33	129	56	19	543	429	31

Results for problem **Hartman3** ($n = 3$)								
$p \leq$	Rule	STUs	FE	GE	HE	E_eff_1	E_eff_2	LL
2	A	3.25	266	152	51	1028	875	18
	B	1.96	145	99	33	640	541	12
	C	1.73	131	85	29	560	473	10
	E	1.78	131	88	30	575	485	10
n+1	A	3.46	300	163	45	1059	924	26
	B	2.32	200	109	30	707	617	24
	C	1.78	154	79	24	535	463	24
	E	1.80	154	79	24	535	463	24

Results for problem **Hartman6** ($n = 6$)								
$p \leq$	Rule	STUs	FE	GE	HE	Eeff_1	Eeff_2	LL
2	A	40.11	1762	959	366	15202	7794	115
	B	26.61	1141	668	239	10168	5247	78
	C	22.32	963	574	195	8502	4429	62
	E	24.14	1014	611	212	9132	4730	70
$n+1$	A	37.94	2357	697	205	10844	6375	360
	B	24.93	1542	510	143	7605	4440	143
	C	24.25	1496	491	129	7151	4234	235
	E	21.90	1377	439	118	6489	3841	139

Results for problem **Rosenbrock** ($n = 2$)								
$p \leq$	Rule	STUs	FE	GE	HE	E_eff_1	E_eff_2	LL
2	A	0.22	217	143	71	716	645	15
	B	0.16	144	106	52	512	460	12
	C	0.16	144	106	52	512	460	12
	E	0.17	144	106	52	512	460	12
$n+1$	A	0.12	133	69	33	370	337	17
	B	0.10	106	53	25	287	262	11
	C	0.10	106	53	25	287	262	11
	E	0.10	106	53	25	287	262	11

Results for problem **Levy8** ($n = 3$)								
$p \leq$	Rule	STUs	FE	GE	HE	E_eff_1	E_eff_2	LL
2	A	1.49	76	59	21	379	316	11
	B	1.47	76	58	21	376	313	11
	C	1.47	76	58	21	376	313	11
	E	1.48	76	58	21	376	313	11
$n+1$	A	1.17	77	41	13	278	239	18
	B	0.98	68	32	12	236	200	21
	C	0.98	68	32	12	236	200	21
	E	0.98	68	32	12	236	200	21

Results for problem **Levy12** ($n = 10$)								
$p \leq$	Rule	STUs	FE	GE	HE	Eeff_1	Eeff_2	LL
2	A	24.71	246	205	76	6476	1826	43
	B	23.90	239	200	74	6309	1779	39
	C	23.90	239	200	74	6309	1779	39
	E	23.90	239	200	74	6309	1779	40
$n+1$	A	19.19	401	106	36	3441	1185	238
	B	17.39	376	97	33	3161	1094	231
	C	18.19	391	101	34	3271	1135	231
	E	16.76	367	89	32	3017	1043	228

Results for problem **Schwefel3.2** ($n = 3$)								
$p \leq$	Rule	STUs	FE	GE	HE	E_eff$_1$	E_eff$_2$	LL
2	A	0.33	171	109	45	768	633	9
	B	0.25	110	84	36	578	470	9
	C	0.26	110	86	36	584	476	9
	E	0.26	110	84	36	578	470	9
n+1	A	0.22	122	70	29	506	419	13
	B	0.17	78	54	23	378	309	13
	C	0.17	82	52	22	370	304	12
	E	0.18	78	54	23	378	309	12

Results for problem **Griewank5** ($n = 5$)								
$p \leq$	Rule	STUs	FE	GE	HE	E_eff$_1$	E_eff$_2$	LL
2	A	7.04	220	181	80	2325	1344	34
	B	6.82	218	176	78	2268	1312	34
	C	6.84	218	177	78	2273	1316	34
	E	7.01	219	179	79	2299	1330	34
n+1	A	5.12	305	100	41	1420	910	86
	B	5.05	301	100	41	1416	906	86
	C	4.84	295	95	40	1370	875	86
	E	4.96	294	95	40	1369	874	87

Results for problem **Griewank7** ($n = 7$)								
$p \leq$	Rule	STUs	FE	GE	HE	Eeff$_1$	Eeff$_2$	LL
2	A	15.57	304	255	114	5281	2122	60
	B	15.12	301	249	111	5152	2074	61
	C	15.23	302	251	112	5195	2090	61
	E	15.36	301	249	111	5152	2074	61
n+1	A	11.04	483	125	52	2814	1347	216
	B	11.24	493	129	54	2908	1387	212
	C	10.96	477	125	52	2808	1341	211
	E	10.94	472	123	51	2761	1321	202

Results for problem **Ratz4** ($n = 2$)								
$p \leq$	Rule	STUs	FE	GE	HE	Eeff$_1$	Eeff$_2$	LL
2	A	6.83	850	540	242	2656	2414	68
	B	6.81	838	544	230	2616	2386	60
	C	6.37	802	508	214	2460	2246	56
	E	6.31	802	496	210	2424	2214	56
n+1	A	6.97	1031	449	176	2457	2281	102
	B	5.13	733	358	140	1869	1729	68
	C	5.12	726	359	142	1870	1728	73
	E	5.15	726	359	142	1870	1728	73

As an example, we take the last problem (Ratz4) to demonstrate the behavior of the splitting an the influence of the rules. This problem considers the function

$$f(x) = \sin(x_1^2 + 2x_2^2)\exp(\Leftrightarrow x_1^2 \Leftrightarrow x_2^2)$$

in the starting region $[x]_i = [\Leftrightarrow 3, 3]$, $i = 1, \ldots, 2$. The following pictures are snapshots of the boxes in the working list L after 125 iterations of the main algorithm. The first four pictures in Figure 3 correspond to the method with the usual splitting technique and the different sorting rules, the pictures in Figure 4 correspond to the method with the special splitting technique.

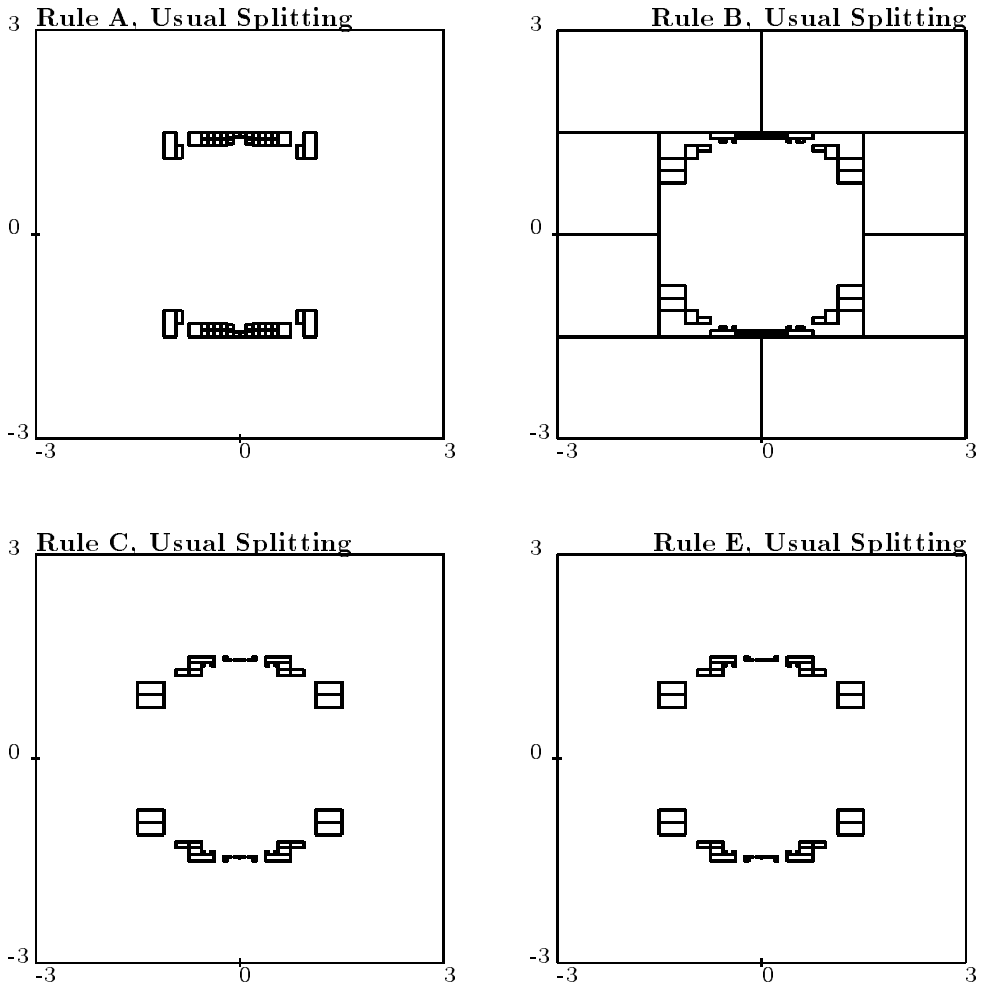

Figure 3. Boxes in the working list after 125 iterations with usual splitting

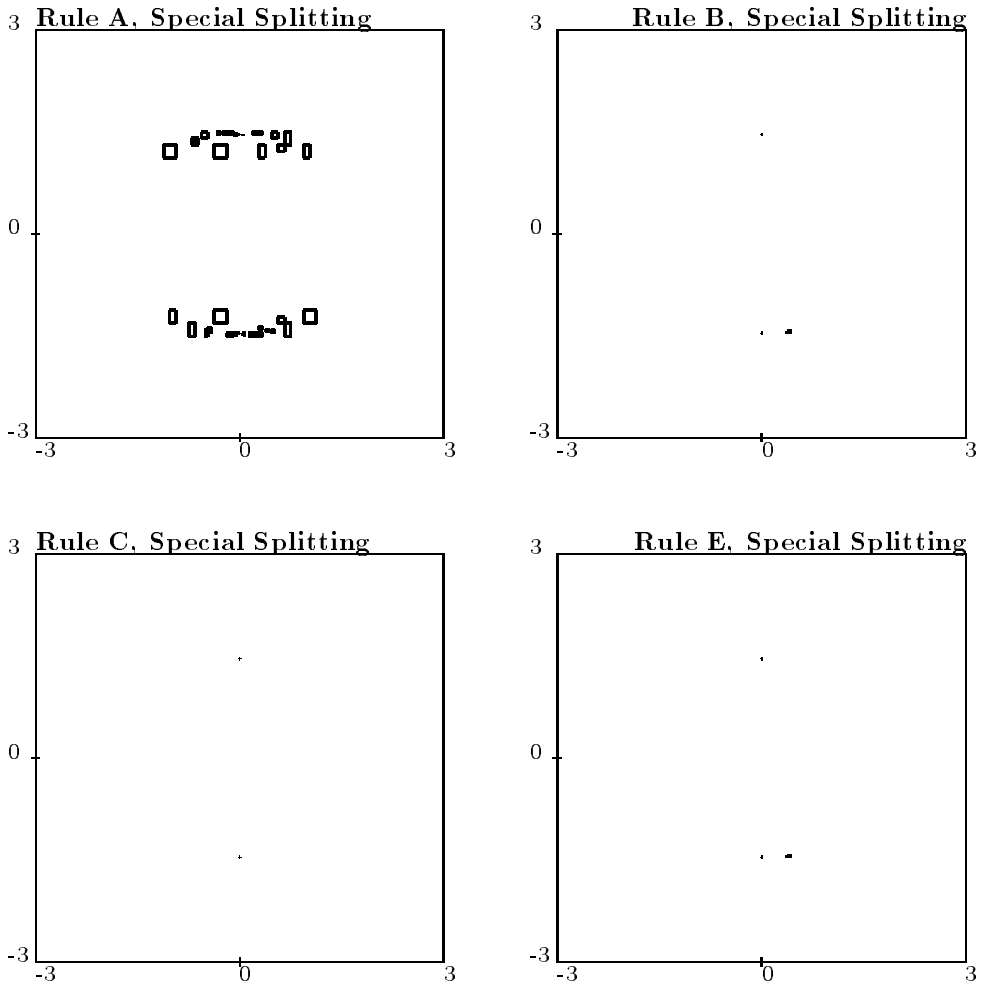

Figure 4. Boxes in the working list after 125 iterations with special splitting

Finally, we give an overview on the results for the complete test set by listing the necessary resources (execution time, evaluation efforts, and maximum list length) for the different variants of our method (Table 1). The values given are relative values (in percent) with respect to the method with Rule A and usual splitting (which is used as reference value corresponding to 100%). The table gives the best values the worst values, and the average values (for all test problems) achieved in the whole test set.

Table 1. Results for the complete test set (relative values)

$p \leq$	Rule	Values	STUs	Eeff_1	Eeff_2	LL
2	B	best	60.3%	62.3%	61.8%	66.7%
		average	95.0%	94.8%	94.8%	102.4%
		worst	131.3%	129.9%	129.4%	178.3%
	C	best	53.2%	54.5%	54.1%	53.9%
		average	93.9%	95.0%	95.0%	98.5%
		worst	137.1%	136.1%	136.8%	169.6%
	E	best	54.8%	55.9%	55.4%	55.6%
		average	95.3%	95.1%	95.1%	99.1%
		worst	141.7%	134.9%	134.5%	157.9%
$n+1$	A	best	54.5%	51.7%	52.2%	100.0%
		average	82.0%	75.0%	78.6%	221.2%
		worst	113.6%	111.2%	111.5%	553.5%
	B	best	45.5%	40.1%	40.6%	66.7%
		average	76.5%	69.6%	72.9%	206.4%
		worst	108.3%	104.7%	105.3%	537.2%
	C	best	45.5%	40.1%	40.6%	66.7%
		average	73.1%	67.5%	70.7%	207.4%
		worst	101.5%	104.0%	104.7%	537.2%
	E	best	45.5%	40.1%	40.6%	71.4%
		average	75.8%	67.8%	71.1%	202.7%
		worst	100.0%	100.0%	100.0%	530.2%

8. Conclusion

Studying the numerical results for the four branching rules combined with different splitting techniques, we recognize that there are test problems for which Rule B, Rule C, and Rule E are much more efficient than Rule A. On the other hand, there are also some problems where the new rules are worse. On average, the branching rules alone lead to an improvement of about 10%.

The special splitting technique improves the performance of the global optimization method significantly, by drastically decreasing the evaluation effort. The price to pay for this improvement is an increasing storage space. Further improvement is due to the branching rules B, C, and E, used as sorting rules in the interval Newton Gauss-Seidel step. This holds for the best cases, the average, and for the worst cases.

Summarizing the consequences of the numerical tests, we can conclude that for Rules B, C, and E combined with the special splitting technique we can expect an average improvement of about 25% in the efficiency of the method, keeping in mind that on average there is approximately a doubling in the necessary storage space.

References

1. Alefeld, G., Herzberger, J. (1983), *Introduction to Interval Computations*. Academic Press, New York.
2. Berner, S. (1996), *New Results on Verified Global Optimization*. Submitted for publication in Computing, Springer-Verlag, Wien.
3. Csendes, T., Ratz, D. (1995), *Subdivision Direction Selection in Interval Methods for Global Optimization*. SIAM Journal of Numerical Analysis, accepted for publication.
4. Fischer, H.-C. (1990), *Schnelle automatische Differentiation, Einschließungsmethoden und Anwendungen*. Dissertation, Universität Karlsruhe.
5. Hammer, R., Hocks, M., Kulisch, U., Ratz, D. (1993), *Numerical Toolbox for Verified Computing I – Basic Numerical Problems*. Springer-Verlag, Heidelberg, New York.
6. Hansen, E., Greenberg, R. (1983), *An Interval Newton Method*. Applied Mathematics and Computations **12**, 89–98.
7. Hansen, E. (1992), *Global Optimization Using Interval Analysis*. Marcel Dekker, New York.
8. Klatte, R., Kulisch, U., Neaga, M., Ratz, D., Ullrich, Ch. (1992), *PASCAL–XSC – Language Reference with Examples*. Springer-Verlag, New York.
9. Neumaier, A. (1990), *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge.
10. Ratz, D. (1992), *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*. Dissertation, Karlsruhe.
11. Ratz, D. (1994), *Box-Splitting Strategies for the Interval Gauss-Seidel Step in a Global Optimization Method*. Computing **53**, 337–353, Springer-Verlag, Wien.
12. Ratz, D., 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.
13. Ratz, D. (1996), *On Branching Rules in Second-Order Branch-and-Bound Methods for Global Optimization*. In: Alefeld, G., Frommer, A. und Lang, Bruno. (Eds.), *Scientific Computing and Validated Numerics*, 221–227, Akademie-Verlag, Berlin.