Noname manuscript No.
(will be inserted by the editor)

A Posteriori Direction Selection Rules for Interval Optimization Methods

T. Csendes[1], **R. Klatte**,[2] **and D. Ratz**[3]

[1] Institute of Informatics, University of Szeged, Szeged, Hungary
e-mail: `csendes@inf.u-szeged.hu`
[2] Faculty of Mathematics, Universität Karlsruhe, Karlsruhe, Germany, e-mail: `Rudi.Klatte@math.uni-karlsruhe.de`
[3] Department of Applied Informatics and Formal Descriptions, University of Karlsruhe, Karlsruhe, Germany, e-mail: `Dietmar.Ratz@aifb.uni-karlsruhe.de`

Received: June 15, 2000, accepted: ..., 2000

Abstract A posteriori direction selection rules are studied for interval global optimization methods to allow the user to choose the one best fitting for the problems at hand. Earlier interval subdivision direction selection rules involved easy to use a priori selection rules with more ore less argumentation why they can be appropriate. Although extensive computational investigations were made on the five most often used selection rules, the underlying reasons why these can improve the efficiency of the optimization algorithms remained unknown. The present paper aims to contribute to the understanding of the functioning of these interval subdivision direction selection rules.

Key words Subdivision direction selection, Global optimization, A posteriori rule.

1 Introduction.

Consider the global optimization problem in the usual form of

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

where $f(x) : \mathbb{R}^n \to \mathbb{R}$ is the nonlinear objective function, and $X \subseteq \mathbb{R}^n$ is the search interval.

Denote the set of compact intervals by $\mathbb{I} := \{[\underline{I}, \overline{I}] \mid \underline{I} \leq \overline{I},\ \underline{I}, \overline{I} \in \mathbb{R}\}$ and the set of n-dimensional intervals (also simply called intervals or boxes) by $\mathbb{I}^n$. Underline and overline denote the lower and upper bounds of the related intervals, respectively. We call a function $F : \mathbb{I}^n \to \mathbb{I}$ an *inclusion function* of $f : \mathbb{R}^n \to \mathbb{R}$ in X, if $x \in Y$ implies $f(x) \in F(Y)$ for each interval Y in X. In other words, $f(X) \subseteq F(X)$, where $f(X)$ is the range of $f(x)$ on X. The inclusion function of the gradient of $f(x)$ is denoted by $\nabla F(X)$, and its Hessian by $H(X)$. The width of an interval will be denoted by $w(X) = \max_{i=1,\ldots,n}(\overline{X_i} - \underline{X_i})$.

It is assumed that inclusion functions of the objective function and its first two derivatives are available. In general these can easily be generated by natural interval extension and by automatic differentiation [1,7,8,10]. The studied algorithm to solve problem (1) is as follows:

Step 0 Set $Y = X$ and $y = \underline{F}(X)$. Initialize the list $L = ((Y, y))$ and the cut-off level $\tilde{f} = \max F(X)$.
Step 1 Choose a coordinate direction $k \in \{1, 2, \ldots, n\}$.
Step 2 Bisect Y in direction k: $Y = V^1 \cup V^2$.
Step 3 Calculate $F(V^1)$ and $F(V^2)$, and set $v^i = \underline{F}(V^i)$ for $i = 1, 2$ and $\tilde{f} = \min\{\tilde{f}, \max F(V^1), \max F(V^2)\}$.
Step 4 Remove (Y, y) from the list L.
Step 5 Cut-off test: discard the pair (V^i, v^i) if $v^i > \tilde{f}$ (where $i \in \{1, 2\}$).
Step 6 Monotonicity test: discard any remaining pair (V^i, v^i) if $0 \notin F'_j(V^i)$ for any $j \in \{1, 2, \ldots, n\}$, and $i = 1, 2$.
Step 7 Use acceleration tests (as the concavity test or the interval Newton step) to delete pairs or shrink intervals of remaining pairs.
Step 8 Add the remaining pair(s) to the list L. If the list becomes empty, then STOP.
Step 9 Denote the pair with the smallest second element by (Y, y).
Step 10 If the width of $F(Y)$ and the width of Y are less than ε, then print $F(Y)$ and Y, STOP.
Step 11 Go to Step 1.

In the following we study possible implementations of Step 1, and their effects on the efficiency of the optimization algorithm.

2 A priori interval selection rules

Figure 1 illustrates possible inclusion values related to a subdivision.

All the rules select a direction with a merit function (expressing how good a given direction is for the current one-step-look-ahead decision):

$$k := \arg \max_{i=1}^{n} D(i), \tag{2}$$

where the merit function $D(i)$ is determined by the given rule. If many such optimal k indices exist then the algorithm can chose the smallest one, or it can select an optimal direction randomly.

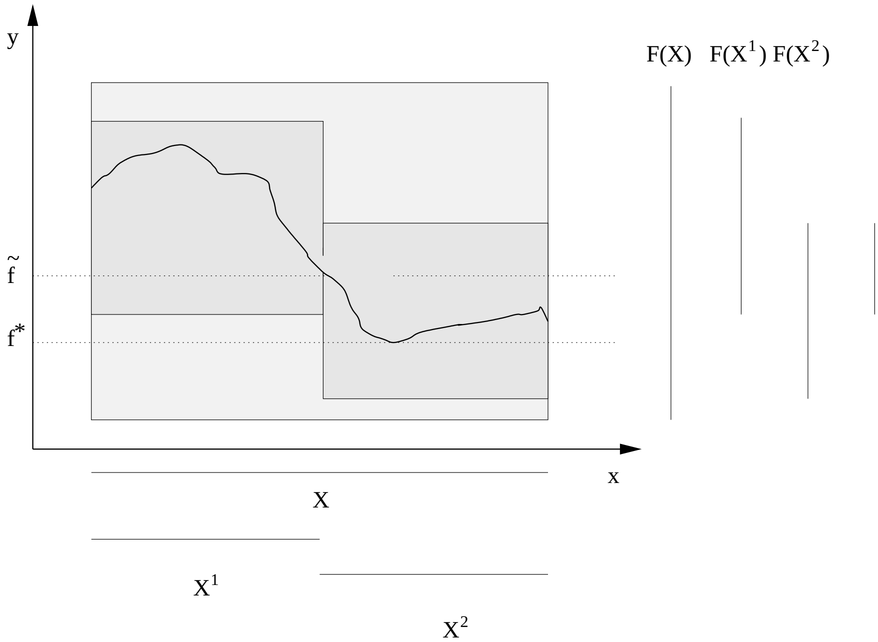

Fig. 1 An illustration of possible inclusion values related to a subdivision: X is the starting interval, X^1 and X^2 are the resulting subintervals, $F(X)$, $F(X^1)$, and $F(X^2)$ the respective inclusion function values. The last vertical section shows the intersection $F(X^1) \cap F(X^2)$. Also the global minimum value f^* and its current upper bound $\tilde{f}$ are highlighted.

2.1 Rule A

The first rule was the interval-width oriented rule. This rule chooses the coordinate direction with

$$D(i) := w(X_i). \tag{3}$$

This rule is justified by the idea that, if the original interval is subdivided in a uniform way, then the width of the current subintervals goes to zero most rapidly.

The algorithm with Rule A is convergent both with and without the monotonicity test [10]. This rule allows a relatively simple analysis of the convergence speed (as in [10], Chapter 3, Theorem 6).

2.2 Rule B

E. Hansen described another rule (initiated by G. W. Walster) [7]. The direct aim of this heuristic direction selection rule is to find the component for which

$$W_i = \max_{t \in X_i} f\,(m_1, \ldots, m_{i-1}, t, m_{i+1}, \ldots, m_n) -$$

$$\min_{t \in X_i} f(m_1, \ldots, m_{i-1}, t, m_{i+1}, \ldots, m_n)$$

is the largest (where $m_i = (\underline{X}_i + \overline{X}_i)/2$ is the midpoint of the interval X_i). The factor W_i, that should reflect how much f varies as x_i varies over X_i, is then approximated by $w(\nabla F_i(X))w(X_i)$ (where $\nabla F_i(X)$ denotes the i-th component of the inclusion of the gradient). The latter is not an upper bound for W_i (cf. [7] page 131 and Example 2 in Section 3 of [5]), yet it can be useful as a merit function.

Rule B selects the coordinate direction, for which (2) holds with

$$D(i) := w(\nabla F_i(X))w(X_i). \tag{4}$$

It should be noted that the basic bisection algorithm represents only one way in which Rule B was applied in [7]. There the subdivision was, e.g., also carried out for many directions in a single iteration step.

2.3 Rule C

The next rule was defined by D. Ratz [11]. The underlying idea was to minimize the width of the inclusion: $w(F(X)) = w(F(X) - F(m(X))) \approx w(\nabla F(X)(X - m(X))) = \sum_{i=1}^{n} w(\nabla F_i(X)(X_i - m(X_i)))$, where $m(X)$ is the midpoint of the interval X. Obviously, that component is to be chosen for which the term $w(\nabla F_i(X)(X_i - m(X_i)))$ is the largest. Thus, Rule C can also be formulated with (2) and

$$D(i) := w(\nabla F_i(X)(X_i - m(X_i))). \tag{5}$$

The important difference between (4) and (5) is that in Rule C the width of the multiplied intervals is maximized, not the multiplied widths of the respective intervals (and these are in general not equal). After a short calculation, the right-hand side of (5) can be written as

$$\max\{|\min \nabla F_i(X)|, |\max \nabla F_i(X)|\} w(X_i).$$

This corresponds to the maximum smear defined by R. B. Kearfott (used as a direction selection merit function solving systems of nonlinear equations [8]) for the case $F : \mathbb{I}^n \to \mathbb{I}$. It is easy to see that the Rules B and C give the same merit function value if and only if either $\underline{\nabla F}_i(X) = 0$ or $\overline{\nabla F}_i(X) = 0$.

2.4 Rule D

The fourth rule, Rule D is derivative-free like Rule A, and reflects the machine representation of the inclusion function $F(X)$ (see [7]). It is again defined by (2) and by

$$D(i) := \begin{cases} w(X_i) & \text{if } 0 \in X_i, \\ w(X_i)/\mathrm{mig\ X_i} & \text{otherwise,} \end{cases} \tag{6}$$

where mig X is the so-called mignitude of the interval X defined by: mig X := $\min_{x \in X} |x|$.

This rule may decrease the excess width $w(F(X)) - w(f(X))$ of the inclusion function that is caused in part by the floating point computer representation of real numbers. Consider the case when the component widths are of similar order, and the absolute value of one component is dominant. The subdivision of the latter component may result in a worse inclusion, since the raster points of the representable numbers are sparser in this direction.

2.5 Rule E

Similar to Rule C, the underlying idea of Rule E is to minimize the width of the inclusion, but this time based on second order information (suggested by D. Ratz [13]):

$$D(i) := w((X_i - m(X_i))(\nabla F_i(m(X)) + \frac{1}{2}\sum_{j=1}^{n}(H_{ij}(X)(X_i - m(X_i))))). \tag{7}$$

Many interval optimization codes use automatic differentiation to produce the gradient and Hessian values. For such an implementation the subdivision selection rule E is not much more expensive than the others.

3 A posteriori direction selection rules

In contrast to a priori direction selection rules that must be given as explicit decision rules that — for the sake of efficiency — do not require to evaluate the subdivisions in all the possible directions, a posteriori rules may formulate principles that can be checked seeing the results of all possible subdivisions. Note that although a posteriori rules can be optimal regarding the current subdivision step according to a given principle, the main aim, the better efficiency of the whole optimization procedure, not necessarily has to be achieved in this way. To avoid confusion, the a posteriori rules will be denoted by numbers.

Although these a posteriori rules are new, they have less practical significance, since they need the related function evaluations for all possible subdivision directions. In this way they serve better as investigation tools to find better a priori selection rules.

3.1 Rule 1

The first a posteriori direction selection rule reflects the usual reasoning behind the a priori rules, i.e. that direction should be chosen which allows the best inclusion of the objective function after the subdivision. In other terms, the union of the inclusion functions on the resulting subintervals must have minimal width:

$$k = \arg\min_i w(F(X^{1(i)}) \cup F(X^{2(i)})),$$

where $X^{1(i)}$ and $X^{2(i)}$ are the resulting intervals of the subdivision of the current interval X along the direction i. The arguments behind a priori rules B, C and E are the closest to this principle.

3.2 Rule 2

The second a posteriori rule does not seem to make much sense, since it requires that the intersection of the resulting subintervals must have the minimal width:

$$k = \arg\min_i w(F(X^{1(i)}) \cap F(X^{2(i)})).$$

Still it makes this rule interesting that it can cause quite different ranges for the resulting subintervals. This feature can be useful for the cut-off test, and thus also for the efficiency of the whole algorithm.

3.3 Rule 3

It is also a reasonable principle to follow that the subdivision should be such that the resulting subintervals present the best possible lower bound on the minimal objective function value on the current interval X:

$$k = \arg\max_i \min(\underline{F}(X^{1(i)}), \underline{F}(X^{2(i)})).$$

3.4 Rule 4

The last investigated a posteriori direction selection rule aims to find that direction according to which the worst of the resulting subinterval has the maximal lower bound on the objective function:

$$k = \arg\max_i \max(\underline{F}(X^{1(i)}), \underline{F}(X^{2(i)})).$$

This rule seems again surprising after Rule 3, still the possibly better deletion ratios with the cut-off test may justify it.

Since it is hard or even impossible to foresee the consequences of the suggested a posteriori interval subdivision direction selection rules for the efficiency of the underlying optimization algorithm, an extensive computational study was carried out to find the best rule and to check whether the earlier a priori rules can be related to the new a posteriori rules.

4 Computational results

4.1 Testing conditions

The numerical tests were made on a dual processor Sun Sparcstation 20/502 with 50 Mhz. The program was coded in Pascal-XSC [9] and used the Numerical Toolbox for Verified Computing [6]. The standard time unit (STU, 1000 evaluations of the non-interval Shekel-5 function) was 0.0072 sec. on the workstation used in the given software environment. We have obtained large CPU times measured in STU. These are in part due to the interval implementation [6,9] and the time consuming subdivisions in all possible directions in each step. This is why the CPU times of the single runs are now not much informative regarding the efficiency.

The numerical study involved the following 28 test functions with their usual search regions and parameterizations [5,7,15]: Shekel-5, Shekel-7, Shekel-10, Hartman-3, Hartman-6, Three-Hump-Camel-Back, Six-Hump-Camel-Back, Branin (RCOS), Rosenbrock, Levy-8 to Levy-16, Levy-18, Schwefel-2.1, Schwefel-3.1, Schwefel-3.2, Schwefel-2.5, Schwefel-2.14, Schwefel-2.18, Griewank-5, Griewank-7 and Goldstein-Price. This is basically the extended set of standard global optimization test problems, having the basic selection criterion that all the a priori selection rules should be applicable, i.e. the inclusion of the second derivative must be available in a way.

The inclusion functions were generated by natural interval extension with the necessary outside rounding, and the derivatives were obtained by forward mode automatic differentiation. The used interval software environment was tuned to good inclusion functions at the cost of larger computational complexity.

The stopping criterion of the algorithm required that the width of both the current interval and the inclusion of the objective function on it be at most $\epsilon = 10^{-2}$. This is close to usual settings (cf. [5,15]).

4.2 Numerical results

The numerical results are summarized in Table 1. We gave the most important efficiency indicators that can reflect the algorithmic improvements: NFE stands for the number of interval inclusion function evaluations, NGE for the number of gradient evaluations, NHE for the number of Hessian evaluations, and MLL for the for the maximal list length that was required to solve the given problems.

Due to the extensive testing, here you can find only comprised data as the sum of all function evaluations, and relative indicators: % means the relative figure compared to that of the algorithm variant 1, and AoP denotes the average of percentages measured on the single problems (in contrast to the % value that is based on the sums). The decision rule 1 was used as the basis of the comparisons since it reflects the usual reasoning behind the a priori rules.

Basically, the percentage figures (row %) reflect the relative amount of work to be done when the whole set of problems is to be solved, i.e. when the user has a set of problems similar to the test set, and the question is what is to be expected in terms of the given efficiency indicators — compared to the results obtained with rule 1. In contrast to that, the AoP figures express the expected changes for single problems: when the user has a problem similar to those in the test set, then the AoP relative figures tell how much improvement can be expected on the basis of the numerical test compared to the results obtained with rule 1 again.

We have tested two algorithms with all the four new a posteriori direction rules 1-4. The first algorithm used the standard splitting of the involved interval Newton step (with 2 resulting boxes), while the second one (alternative splitting) allowed at most $n+1$ resulting boxes for an interval Newton step [6].

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. The special splitting strategy suggested in [11] and also used in [6] uses *all* gaps in a special way. That means: if a gap is produced in the i-th component step, one part of the actual working box Y is stored by using one part of the component Y_i as i-th component of Y. The other part of Y_i is used to update Y and to go on with the next component steps of the interval Gauss-Seidel method.

In [12] and [14] the impact of gap-treating and box-splitting techniques were investigated. The results show clearly, that the special splitting strategy is more important for the efficiency of the algorithm than the use of preconditioners. Especially, best performance comes with special splitting.

The important question regarding this data is which a posteriori rule provides the best efficiency. The dimension of the tested problems are between 2 and 10, thus the most time consuming part was the evaluation of the Hessians. For the standard splitting the picture is not very clear: the Rule 2 is the best according to the sum of evaluations (11,279), while Rule 4 is the best according to the average of percentages (93%). On this basis, we can state that Rule 2 should be chosen when a set of problems is to be solved that is similar to the present test set, and Rule 4 when only one problem is to be solved. Although the same is true for the alternative splitting case (with $n+1$ resulting boxes), the differences are much smaller, and we can suggest Rule 4 from both point of view. This time the worst choice was Rule 3.

For the number of objective function and gradient evaluations, Rule 4 was a clear winner with standard splitting (with 95 and 83%, and 96 and 82%, respectively). For the alternative splitting again Rules 2 and 4 were the best in such a way that Rule 2 was better for the sum of function evaluations (97 and 98%, respectively), while Rule 4 the best for the average of percentages (89% in both cases). This together with the efficiency results for the Hessian evaluations allow us to conclude that the most efficient

Table 1 Summarized results of the numerical test. The two parts of the table contain results for the runs when standard splitting was used in the interval Newton step, and for those when maximal $n+1$ resulting boxes were allowed.

Efficiency indicator	Algorithm variants							
	standard splitting				n+1 resulting boxes			
	1	2	3	4	1	2	3	4
NFE Σ	39,750	38,518	40,109	37,895	35,904	34,765	46,526	35,067
%	100%	97%	101%	95%	100%	97%	130%	98%
AoP	100%	103%	105%	83%	100%	92%	217%	89%
best %	100%	22%	25%	26%	100%	26%	75%	27%
worst %	100%	460%	171%	109%	100%	162%	1517%	123%
NGE Σ	28,875	27,993	29,081	27,642	21,413	21,002	24,277	21,074
%	100%	97%	101%	96%	100%	98%	113%	98%
AoP	100%	98%	105%	82%	100%	90%	190%	89%
best %	100%	25%	21%	24%	100%	28%	76%	29%
worst %	100%	385%	188%	109%	100%	182%	1271%	117%
NHE Σ	11,574	11,279	11,633	35,871	7,198	7,046	8,280	7,075
%	100%	97%	101%	310%	100%	98%	115%	98%
AoP	100%	100%	104%	93%	100%	94%	170%	92%
best %	100%	24%	19%	24%	100%	29%	100%	29%
worst %	100%	396%	188%	107%	100%	190%	1126%	127%
MLL Σ	2,428	2,249	2,432	2,194	3,642	3,464	5,001	3,474
%	100%	93%	100%	90%	100%	95%	137%	95%
AoP	100%	92%	103%	71%	100%	91%	155%	84%
best %	100%	14%	8%	10%	100%	25%	100%	25%
worst %	100%	410%	207%	150%	100%	119%	957%	120%

implementation regarding the computational efforts was that with Rule 4. In other words, if we solve very hard problems, then Rule 4 is to be used to decide the direction of subdivisions in order to decrease the time complexity. The best and worst achieved relative performance (denoted by best % and worst %, respectively) shows a mixed picture, but basically again the rules 2 and 4 are the best in the overall evaluation. On the other hand, these values were mostly determined by extreme cases, by a few special problems.

The memory requirements are also important when solving hard optimization problems, but for this indicator, the question is better whether a problem can be solved at all with the given computational environment. Now the sum of the maximal list lengths (MLL) would reflect at one hand the total amount of memory units needed to solve all test problems at the same time (it is not very realistic), and a kind of compound, general space requirement. For the standard splitting algorithm variant Rule 4 is the most advantageous (90 and 71%), while for the alternative splitting Rules 2 and 4 share the first two positions. The best relative performance is good for the rules 2 and 4, although rule 3 performs best for the standard splitting.

Table 2 The ratio of the cases given in percentages when the decision of an a posteriori rule is the same as that of a given a priori rule. The two parts of the table contain results for the runs when standard splitting was used in the interval Newton step, and for those when maximal $n+1$ resulting boxes were allowed. The figures in bold face denote the largest values (at least 90% of the largest one) of the given set.

A priori rules	A posteriori rules							
	standard splitting				n+1 resulting boxes			
	1	2	3	4	1	2	3	4
A	40.4%	41.1%	39.2%	40.5%	32.5%	35.8%	26.2%	34.2%
B	**56.3%**	**57.3%**	**57.3%**	**56.1%**	**65.7%**	**62.9%**	**70.4%**	**64.0%**
C	**54.8%**	**56.2%**	**55.7%**	**54.8%**	**62.6%**	60.6%	**67.6%**	60.8%
D	40.4%	41.1%	39.2%	40.5%	32.5%	35.8%	26.2%	34.2%
E	**52.2%**	**52.6%**	**52.2%**	**52.2%**	54.1%	52.8%	54.8%	52.9%

Summing up the results of the numerical tests, we can conclude that both from the point of view of the computational and the memory complexity, Rule 4 is the best choice, closely followed by Rule 2. Rules 1 and 3 are worse. This result is quite surprising, since it contradicts the common belief that interval subdivision must produce the interval inclusion with the smallest width — that is expressed exactly by Rule 1. Instead, interval subdivision direction selection rules should maximize the lower bound of the worse subinterval (Rule 4), or minimize the width of the intersection of the resulting inclusion function values (Rule 2). The latter rules both support the cut-off test, i.e. it seems that according to the present numerical tests, the interval subdivision should help first of all to delete many subintervals as early as possible.

4.3 The coincidence of a priori and a posteriori rules according to the numerical test

After the discussion of the primary numerical results, the next question is which a priori applicable subdivision direction selection rules are the closest to the best a posteriori rules. Table 2 includes the corresponding data obtained. The percentages given here represent the ratio of the cases when the decision of an a posteriori rule was the same as that of a given a priori rule. In other words, we have counted those cases when the result of the related a priori and a posteriori rules were the same. The relative number is then listed in the table. No full coincidence was experienced. The two parts of the table contain again results for the runs when standard splitting was used in the interval Newton step, and for those when maximal $n+1$ resulting boxes were allowed. The figures in bold face denote the largest values (at least 90% of the largest one) of the given set.

According to the obtained results, the a priori Rule B is the closest to the a posteriori rules 2 and 4, both for the standard and for the alternative splitting algorithm variants, closely followed by Rule C (and to a less extent by Rule E). At one hand this conclusion is in accordance with the numerical results obtained for the a priori rules (see e.g. [4, 5, 15]), on the other hand in the latter studies Rule C was always a bit better than Rule B. This slight discrepancy may indicate a further improvement possibility in the a priori rules. On the other hand, although the present results on the a posteriori rules confute the earlier ideas on which the direction selection rules were based, still they confirm through the strong coincidence of Rules 2 and 4 with Rules B and C that these could be closely optimal for that purpose.

5 Summary

According to our investigations the most successful interval subdivision direction selection rules follow the principle of either maximizing the larger lower bound of the objective function on the resulting subintervals, or minimizing the width of the intersection of the resulting inclusion functions on the subintervals. In both cases, the algorithm tries to subdivide the current interval in such a way, that the new ranges of the objective function on the subintervals should be as different as possible. Out of the five a priori rules, Rule B and Rule C are the closest to these principles. This finding confirms the earlier results [3–5, 13, 15] on a priori rules on the one hand, and it opens new ways for further algorithmic improvements on the other hand.

Acknowledgement.

This work was supported by the Grants OMFB D-7/97, HUN97/008, FKFP 449/99, OTKA T 017241 and OTKA T032118. The computer programs were coded and run by Astrid Goos.

References

1. G. Alefeld and J. Herzberger. *Introduction to Interval Computations*, Academic Press, New York, 1983.
2. S. Berner, *New results on verified global optimization*, Computing, 57 (1996), pp. 323-343.
3. T. Csendes, *Interval analysis: subdivision directions in interval b&b methods.* Encyclopedia of Optimization, Kluwer, Dordrecht, accepted for publication.
4. T. Csendes and D. Ratz, *A review of subdivision direction selection in interval methods for global optimization.* ZAMM, 76(1996) Suppl. 1, pp. 319–322.
5. T. Csendes and D. Ratz, *Subdivision direction selection in interval methods for global optimization*, SIAM J. Numer. Anal., 34 (1997), pp. 922-938.
6. R. Hammer, M. Hocks, U. Kulisch, and D. Ratz. *Numerical Toolbox for Verified Computing I.*, Springer, Berlin, 1993.

7. E. Hansen. *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.
8. R. B. Kearfott. *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht, 1996.
9. R. Klatte, U. Kulisch, M. Neaga, D. Ratz, and Ch. Ullrich. *PASCAL–XSC — Language Reference with Examples*, Springer, New York, 1992.
10. H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester, 1988.
11. D. Ratz. *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen.* Dissertation, Universität Karlsruhe, Germany, 1992.
12. D. Ratz, *Box-Splitting Strategies for the Interval Gauss-Seidel Step in a Global Optimization Method*, Computing, 53 (1994) pp. 337–353.
13. D. Ratz, *On branching rules in second-order branch-and-bound methods for global optimization.* In Scientific Computing and Validated Numerics, G. Alefeld, A. Frommer, and B. Lang, eds., Akademie-Verlag, Berlin, 1996, pp. 221–227.
14. D. Ratz, *New Results on Gap-Treating Techniques in Extended Interval Newton Gauss-Seidel Steps for Global Optimization.* In: *Developments in Global Optimization*, I.M. Bomze, T. Csendes, R. Horst and P.M. Pardalos, eds., Kluwer Academic Publishers, Dordrecht, 1997, pp. 55–72.
15. D. Ratz and T. Csendes, *On the selection of subdivision directions in interval branch-and-bound methods for global optimization*, J. Global Optimization, 7 (1995), pp. 183–207.