ZAMM **76**, S1, 323–326 (1996)

Ratz, D.

Improved Techniques for Gap-Treating and Box-Splitting in Interval Newton Gauss-Seidel Steps for Global Optimization with Validation

Interval global optimization algorithms often incorporate an interval Newton Gauss-Seidel step to rapidly reduce the widths of the boxes resulting from the underlying generalized bisection method. It aims at determining the roots of the gradient of the objective function, whereas various other techniques eliminate regions containing roots which do not correspond to global optimizers. The interval Newton Gauss-Seidel step uses 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. We investigate the impact of different gap-treating and box-splitting techniques producing different numbers of subboxes, and we propose strategies which improve the overall efficiency of the interval Newton Gauss-Seidel step and therefore of global optimization methods. We present results of computational experiments with standard global optimization problems.

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

$$f(x^*) = \min_{x \in [x]} f(x). \tag{1}$$

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

We use the branch-and-bound method described in [8] and [3] with several modifications. Our method starts from an initial box $[x] \in I\mathbb{R}^n$, subdivides $[x]$ and stores the subboxes in a list L, and discards subintervals which are guaranteed not to contain a global minimizer, until the desired accuracy of the intervals 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 [3].

2. Global Optimization Algorithm

In the following, we give a simplified algorithmic description and an overview on our global optimization method.

Algorithm 1: GlobalOptimize $(f, [x], \varepsilon, L_{\text{res}}, [f^*])$

1. $\widetilde{f} := \overline{f_\diamond}(m([x]))$; $[y] := [x]$; $L := \{\}$; $L_{\text{res}} := \{\}$;
2. **repeat**
   (a) $k :=$ OptimalComponent $([y])$; Bisection $([y], k, [u]_1, [u]_2)$; $Bisect := false$;
   (b) **for** $i := 1$ **to** 2 **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. IntervalGaussSeidelStep $(f, [u]_i, [H], [V], p)$;
      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** $(L \neq \{\})$ **and** (**not** $Bisect$) **do**
      i. $([y], \underline{f_y}) :=$ PopHead (L); $\widetilde{f} := \min\{\widetilde{f}, \overline{f_\diamond}(m([y]))\}$; CutOffTest $(L, \widetilde{f})$;
      ii. **if** Accept $(f, [y], \varepsilon)$ **then** $L_{\text{res}} := L_{\text{res}} \uplus ([y], \underline{f_y})$ **else** $Bisect := true$;

   until (**not** $Bisect$);
3. $([y], \underline{f_y}) :=$ Head (L_{res}); $[f^*] := [\underline{f_y}, \widetilde{f}]$; **return** L_{res}, $[f^*]$;

In Algorithm 1, we first compute an upper bound $\widetilde{f}$ for the global minimum value and initialize L and $L_{\rm res}$. Step 2 is the main iteration starting with a bisection of $[y]$. Then we apply a range check, the monotonicity test, the concavity test, and the interval Newton step to the bisected boxes $[u_1]$ and $[u_2]$. The interval Newton step results in p boxes, to which we apply a range check. If the actual box $[V]_j$ is still a candidate for a minimizer, we store it in L. 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* oder with respect to the ages of the boxes in the list (cf. [8]).

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 bisection step. When the iteration stops because the pending list L is empty, we compute a final enclosure $[f^*]$ for the global minimum value, and we return $L_{\rm res}$ and $[f^*]$.

The method can be improved by incorporating an approximate local search procedure trying to decrease the value $\widetilde{f}$. See [5] 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. Interval Gauss-Seidel Step

In our global optimization method, we apply one step of the extended interval Newton Gauss-Seidel method (cf. [1]) to the nonlinear system $\nabla f(y) = 0$ with $y \in [y]$. The subbox $[y]$ is a candidate box for enclosing a minimizer x^*, which we have assumed must satisfy $\nabla f(x^*) = 0$. One step of the extended interval Newton Gauss-Seidel method shall improve the enclosure $[y]$ by formally solving the system $g = [H] \cdot (c - y)$, where $c = m([y])$, $g = \nabla f(c)$, and $[H] = \nabla^2 f([y])$. This method works better if we first apply a *preconditioning*, by using a special matrix $R \in I\!R^{n \times n}$ for computing $b := R \cdot g$ and $[A] := R \cdot [H]$, and consider then the system

$$b = [A] \cdot (c - y).$$

Then, we compute $N'_{\rm GS}([y])$ according to

$$\left.\begin{aligned} [z] &:= [y] \\ [z]_i &:= \left(c_i - \left(b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} [A]_{ij} \cdot ([z]_j - c_j) \right) \Big/ [A]_{ii} \right) \cap [z]_i, \qquad i = 1, \ldots, n \\ N'_{\rm GS}([y]) &:= [z] \end{aligned}\right\}. \tag{2}$$

If $0 \in [A]_{ii}$ for some i, extended interval arithmetic (see [3] for details) is applied. 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 $N'_{\rm GS}([y]) = [V]_1 \cup \ldots \cup [V]_p$, where $[V]_i \in I\!I\!R^n$, $i = 1, \ldots, p$, that is $[V] \in I\!I\!R^{p \times n}$. Different splitting strategies for treating the gaps may be applied resulting in different values for $[V]$ and p.

We summarize the most important properties of the interval Newton Gauss-Seidel step in

Theorem 1. *Let $f : D \subseteq I\!R^n \to I\!R$ be a twice continuously differentiable function, and let $[x] \in I\!I\!R^n$ be an interval vector with $[x] \subseteq D$. Then $N'_{\rm GS}([x])$ has the following properties:*

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

For proofs, see [5]. Note that $p = 0$ in case 2 and $p = 1$ in case 3 of Theorem 1.

In a practical realization of the interval Newton Gauss-Seidel method (2), it is not necessary to compute the $[y]_i$ in fixed order $i = 1, \ldots, n$. A well-known strategy is the Hansen/Greenberg realization [4], which first performs 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.

4. Splitting Strategies

If $0 \in [A]_{ii}$ for several components i, then the extended interval divisions in the interval Newton Gauss-Seidel method possibly produces several gaps in the actual box $[y]$. Therefore, we have to split the result $N'_{\rm GS}([y])$ in two or more boxes. In this case, different splitting strategies may be applied. We give four examples:

$p \le 2$ *Compute all possible gaps in* $[y]$*, and finally use only the largest gap to split* $[y]$. This strategy is known from Hansen/Greenberg [4], and the Newton step results in at most 2 boxes, thus $N'_{\mathrm{GS}}([x]) = [V]_1 \cup [V]_2$.

$p \le 8$ *Compute all possible gaps in* $[y]$*, and finally use at most 3 gaps to split* $[y]$. This strategy was suggested by Hansen [5], and the Newton step results in at most $2^3 = 8$ boxes, thus $N'_{\mathrm{GS}}([x]) = [V]_1 \cup \ldots \cup [V]_8$.

$p \le 2^n$ *Compute all possible gaps in* $[y]$*, and finally use them all to split* $[y]$. As far as we know, nobody uses this strategy, because the Newton step results in at most 2^n boxes causing a proliferation of subboxes, thus $N'_{\mathrm{GS}}([x]) = [V]_1 \cup \ldots \cup [V]_{2^n}$.

$p \le n+1$ *Compute every gap, and use it* immediately *to split* $[y]$ *in a special way.* For this special splitting strategy introduced in [7] 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 strategy with $p \le n+1$, we use each gap to store one part of the actual box $[y]$ by using one part of the component $[y]_i$ and the other part of $[y]_i$ to update $[y]$ 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 $[y]_i = [v]_i \cup [w]_i$.
2. If $[v]_i = [w]_i = \emptyset$, then stop {no solution in $[y]$}.
3. If $[w]_i \neq \emptyset$, then set $[y]_i := [w]_i$ and store $[y]$.
4. Set $[y]_i := [v]_i$ and continue with next i.

Example 1. We handle a box $[y]$ of dimension $n = 3$ assuming that we produce a gap in each component. Incorporating the above strategy, the three component steps of the method split $[y]$ in the following manner:

$$
\begin{array}{ccccccc}
[y] = \begin{pmatrix} [y]_1 \\ [y]_2 \\ [y]_3 \end{pmatrix} & \xrightarrow{1} & \begin{pmatrix} [v]_1 \\ [y]_2 \\ [y]_3 \end{pmatrix} & \xrightarrow{2} & \begin{pmatrix} [v]_1 \\ [v]_2 \\ [y]_3 \end{pmatrix} & \xrightarrow{3} & \begin{pmatrix} [v]_1 \\ [v]_2 \\ [v]_3 \end{pmatrix} \\
\text{storing} \quad \downarrow 1 & & \downarrow 2 & & \downarrow 3 & & \downarrow 4 \\
\begin{pmatrix} [w]_1 \\ [y]_2 \\ [y]_3 \end{pmatrix} =: [V]_1 & & \begin{pmatrix} [v]_1 \\ [w]_2 \\ [y]_3 \end{pmatrix} =: [V]_2 & & \begin{pmatrix} [v]_1 \\ [v]_2 \\ [w]_3 \end{pmatrix} =: [V]_3 & & \begin{pmatrix} [v]_1 \\ [v]_2 \\ [v]_3 \end{pmatrix} =: [V]_4
\end{array}
$$

In many cases, we get $\left(b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} [A]_{ij} \cdot ([y]_j - c_j) \right) \Big/ [A]_{ii} = (-\infty, \infty)$ and no gap occurs, so $[y]_i := [y]_i \cap (-\infty, \infty)$ is unchanged. In these cases, we introduce "gaps" of width zero by splitting $[y]_i = [v]_i \cup [w]_i$ with $[v]_i := [\underline{y}_i, m([y]_i)]$ and $[w]_i := [m([y]_i), \overline{y}_i]$, that is we do a bisection. All four splitting strategies can benefit from this trick.

5. Branching Rules and Sorted Interval Gauss-Seidel Step

In Algorithm 1, different subdivision direction selection rules can be applied to determine an "optimal" component k for bisection of the current box $[y]$ (see [2] and [9]). Each of these rules selects a direction k with $D(k) = \max_{i=1}^{n} D(i)$, where $D(i)$ is determined by the given rule. We investigated two rules in connection with the interval Gauss-Seidel step: Rule A with $D(i) := d([y]_i)$ and Rule C with $D(i) := d(g_i([y]) \cdot ([y]_i - m([y]_i)))$, where $g([y]) = \boldsymbol{\nabla} f([y])$. 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 $D(s_i) \ge D(s_{i+1})$, $i = 1, \ldots, n-1$ for the corresponding direction selection rule $D(\ldots)$. Then, we perform the *sorted* interval Newton Gauss-Seidel step according to

$$
\left.
\begin{aligned}
[z] &:= [y] \\
[z]_{s_i} &:= \left(c_{s_i} - \left(b_{s_i} + \sum_{\substack{j=1 \\ j \neq s_i}}^{n} [A]_{s_i,j} \cdot ([z]_j - c_j) \right) \Big/ [A]_{s_i s_i} \right) \cap [z]_{s_i}, \qquad i = 1, \ldots, n \\
N'_{GS}([y]) &:= [z]
\end{aligned}
\right\} \tag{3}
$$

incorporating the Hansen/Greenberg realization and different splitting strategies.

6. Numerical Experiences

For testing we used the two groups of standard test functions and some new functions (see [5] and [9] for details on the functions). We carried out the numerical tests on a HP 9000/730 equipped with PASCAL–XSC [6] using the basic toolbox modules for automatic differentiation and extended interval arithmetic [3].

In two test suites we compared the methods with no sorting, sorting rule A, and sorting rule C combined with different splitting strategies. In the first test suite we used the "0-width-gap" technique only in the special splitting strategy, in the second test suite, we used it in all four splitting strategies.

For the first test suite, the special splitting strategy improves the performance of the global optimization method significantly, by drastically decreasing the number of Hessian and gradient evaluations and only slightly increasing storage space. Further improvement is due to the sorting rule C, which can improve the efficiency of the algorithm independently of the splitting strategies used. When using the "0-width-gap" technique, the splitting with $p \leq 8$ can also improve the performance, whereas the splittings with $p \leq 2$ or $p \leq 2^n$ often produce bad results.

We list the results for two test functions in the following. Important columns are the runtime (in STUs), the storage space or list length (LL) and the E_eff values combining the three values FE, GE, and HE to a single value approximating the total evaluation effort in terms of objective function evaluations (see [8] for details). The complete results for all test functions together with the source code of our test program can be obtained by anonymous ftp from `iamk4515.mathematik.uni-karlsruhe.de (129.13.129.15)` in directory `pub/documents/ratz/iciam.95`.

Hartman 3 ($n = 3$)

Splittings	Sorting	STUs	FE	GE	HE	E_eff$_1$	E_eff$_2$	LL
$p \leq 2$	C	1.08	87	161	45	840	705	12
	A	1.93	158	283	83	1505	1256	21
	no	1.95	158	283	83	1505	1256	21
$p \leq 8$	C	1.11	87	161	45	840	705	12
	A	1.96	158	283	83	1505	1256	21
	no	1.95	158	283	83	1505	1256	21
$p \leq 2^n$	C	1.18	87	161	45	840	705	12
	A	1.95	158	283	83	1505	1256	21
	no	1.93	158	283	83	1505	1256	21
$p \leq n+1$	C	0.65	82	79	21	445	382	19
	A	1.30	169	161	43	910	781	35
	no	1.83	237	243	55	910	1131	47

Hartman 3 ($n = 3$) **0-width-gaps**

Splittings	Sorting	STUs	FE	GE	HE	E_eff$_1$	E_eff$_2$	LL
$p \leq 2$	C	1.16	116	154	42	830	704	25
	A	3.36	346	469	112	2425	2089	62
	no	3.39	350	471	113	2441	2102	62
$p \leq 8$	C	0.64	97	68	19	415	358	35
	A	1.21	177	139	33	792	693	55
	no	1.23	181	141	34	808	706	55
$p \leq 2^n$	C	0.69	97	68	19	415	358	35
	A	1.21	177	139	33	792	693	55
	no	1.23	181	141	34	808	706	55
$p \leq n+1$	C	0.65	82	79	21	445	382	19
	A	1.30	169	161	43	910	781	35
	no	1.83	237	243	55	910	1131	47

Levy 11 ($n = 8$)

Splittings	Sorting	STUs	FE	GE	HE	E_eff$_1$	E_eff$_2$	LL
$p \leq 2$	C	9.26	123	221	74	4555	1599	23
	A	9.29	123	221	74	4555	1599	24
	no	9.29	123	221	74	4555	1599	24
$p \leq 8$	C	9.24	123	221	74	4555	1599	23
	A	9.26	123	221	74	4555	1599	24
	no	9.26	123	221	74	4555	1599	24
$p \leq 2^n$	C	9.29	123	221	74	4555	1599	23
	A	9.26	123	221	74	4555	1599	24
	no	9.26	123	221	74	4555	1599	24
$p \leq n+1$	C	4.12	148	71	23	1544	616	86
	A	4.77	163	87	27	1831	727	103
	no	5.09	178	85	28	1831	742	99

Levy 11 ($n = 8$) **0-width-gaps**

Splittings	Sorting	STUs	FE	GE	HE	E_eff$_1$	E_eff$_2$	LL
$p \leq 2$	C	5.60	105	120	42	2577	921	43
	A	5.80	109	122	44	2669	949	48
	no	5.80	109	122	44	2669	949	48
$p \leq 8$	C	4.91	177	78	27	1773	705	112
	A	5.16	184	83	29	1892	748	118
	no	5.16	184	83	29	1892	748	118
$p \leq 2^n$	C	41.03	1874	48	15	2798	2186	1432
	A	38.96	1830	56	17	2890	2190	1518
	no	39.16	1830	56	17	2890	2190	1518
$p \leq n+1$	C	4.12	148	71	23	1544	616	86
	A	4.77	163	87	27	1831	727	103
	no	5.09	178	85	28	1831	742	99

7. References

1 Alefeld, G., Herzberger, J.: *Introduction to Interval Computations.* Academic Press, New York, 1983.

2 Csendes, T., Ratz, D.: *Subdivision Direction Selection in Interval Methods for Global Optimization.* SIAM Journal of Numerical Analysis, accepted for publication, 1995.

3 Hammer, R., Hocks, M., Kulisch, U., Ratz, D.: *Numerical Toolbox for Verified Computing I – Basic Numerical Problems.* Springer-Verlag, Heidelberg, New York, 1993.

4 Hansen, E., Greenberg, R.: *An Interval Newton Method.* Applied Mathematics and Computations **12**, 89–98, 1983.

5 Hansen, E.: *Global Optimization Using Interval Analysis.* Marcel Dekker, New York, 1992.

6 Klatte, R., Kulisch, U., Neaga, M., Ratz, D., Ullrich, Ch.: *PASCAL–XSC – Language Reference with Examples.* Springer-Verlag, New York, 1992.

7 Ratz, D.: *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen.* Dissertation, Karlsruhe, 1992.

8 Ratz, D.: *Box-Splitting Strategies for the Interval Gauss-Seidel Step in a Global Optimization Method.* Computing **53**, 337–353, Springer-Verlag, Wien, 1994.

9 Ratz, D., Csendes, T.: *On the Selection of Subdivision Directions in Interval Branch-and-Bound Methods for Global Optimization.* Journal of Global Optimization, accepted for publication, 1995.

Address: Dr. Dietmar Ratz, Universität Karlsruhe, Institut für Angewandte Mathematik, D-76128 Karlsruhe