Using Quadratic Approximations in an Interval Method for Solving Underdetermined and Well-Determined Nonlinear Systems

Bartłomiej Jacek Kubica(✉)

Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland
bkubica@elka.pw.edu.pl

Abstract. This paper considers quadratic approximation as a narrowing tool in an interval branch-and-prune method. We seek the roots of such an approximate equation – a quadratic equation with interval parameters. Heuristics to decide, when to use the developed operator, are proposed. Numerical results for some benchmark problems are presented and analyzed.

Keywords: Nonlinear equations systems · Interval computations · Quadratic approximation · Interval quadratic equation · Heuristic

1 Introduction

In the paper [11] the author considered an interval solver for nonlinear systems – targeted mostly at underdetermined equations systems – and its shared-memory parallelization. In subsequent papers several improvements have been considered, including various parallelization tools [12] and using sophisticated tools and heuristics to increase the efficiency of the solver [13]. As indicated in [13] and [14], the choice of proper tools and the proper heuristic, for their selection and parameterization, appeared to have a dramatic influence on the efficiency of the algorithm.

2 Generic Algorithm

The solver uses interval methods.They are based on interval arithmetic operations and basic functions operating on intervals instead of real numbers (so that result of an operation on numbers belong to the result of operation on intervals, containing the arguments). We shall not define interval operations here; the interested reader is referred to several papers and textbooks, e.g., [8,9,19].

The solver is based on the branch-and-prune (B&P) schema that can be expressed by the following pseudocode:

```
IBP ($\mathbf{x}^{(0)}$; f)
+//+$\mathbf{x}^{(0)}$ is the initial box, f(·) is the interval extension of the function $f{:}\,\mathbb{R}^n \rightarrow \mathbb{R}^m$
// $L_{ver}$ is the list of boxes verified to contain a segment of the solution manifold
// $L_{pos}$ is the list of boxes that possibly contain a segment of the solution manifold
$L = L_{ver} = L_{pos} = \emptyset$ ;
$\mathbf{x} = \mathbf{x}^{(0)}$ ;
loop
    process the box $\mathbf{x}$, using the rejection/reduction tests ;
    if ($\mathbf{x}$ does not contain solutions) then discard $\mathbf{x}$ ;
    else if ($\mathbf{x}$ is verified to contain a segment of the solution manifold) then
        push ($L_{ver}$, $\mathbf{x}$) ;
    else if (the tests resulted in two subboxes of $\mathbf{x}$: $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$) then
        $\mathbf{x} = \mathbf{x}^{(1)}$ ;
        push ($L$, $\mathbf{x}^{(2)}$) ;
        cycle loop ;
    else if ($\mathbf{x}$ is small enough) then push ($L_{pos}$, $\mathbf{x}$) ;
    if ($\mathbf{x}$ was discarded or stored) then
        $\mathbf{x}$ = pop ($L$) ;
        if ($L$ was empty) then exit loop ;
    else
        bisect ($\mathbf{x}$), obtaining $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$;
        $\mathbf{x} = \mathbf{x}^{(1)}$ ;
        push ($L$, $\mathbf{x}^{(2)}$) ;
    end if ;
end loop
end IBP
```

The “rejection/reduction tests”, mentioned in the algorithm are described in previous papers (specifically [13] and [14]), i.e.:

- switching between the componentwise Newton operator (for larger boxes) and Gauss-Seidel with inverse-midpoint preconditioner, for smaller ones,
- the sophisticated heuristic to choose the bisected component [13],
- an initial exclusion phase of the algorithm (deleting some regions, not containing solutions) – based on Sobol sequences [14].

Other possible variants (see, e.g., [11]) are not going to be considered.

3 Quadratic Approximations

3.1 Motivation

As stated above, the main tools used to narrow boxes in the branch-and-prune process are various forms of the Newton operator. This operator requires the computation of derivatives of (at least some of) the functions $f_i(\cdot)$. This computation – usually performed using the automatic differentiation process – is relatively costly. Because of that, in [14] the author proposed a heuristic using the Newton operator only for boxes that can be suspected to lie in the vicinity of a solution (or the solution manifold – in the underdetermined case). For other areas we try to base on 0th-order information only, i.e., function values and not gradients (or other derivatives). In [14] an "initial exclusion phase" was proposed, when regions are deleted using Sobol sequences, inner solution of the tolerance problem [19] and ε-inflation.

In [16] a similar approach (not using Sobol sequences, though) was considered for another problem – seeking Pareto sets of a multicriterion problem. Both papers show that this approach can improve the branch-and-bound type algorithms' efficiency dramatically – at least for some problems.

However, as for some areas the Newton operators do not perform well, we can try to use 2nd (or even higher) order information there. Obviously, this requires Hesse matrix computations, which is very costly, but can be worthwhile.

3.2 Quadratic Approximation

Each of the functions $f_i(x_1, \ldots, x_n)$ can be approximated by the 2nd order Taylor polynomial:

$$x \in \mathbf{x} \;\rightarrow\; f_i(x) \in \mathsf{f}_i(\check{x}) + \mathsf{g}(\check{x})^T \cdot (x - \check{x}) + \frac{1}{2} \cdot (x - \check{x})^T \cdot \mathsf{H}(\mathbf{x}) \cdot (x - \check{x}), \qquad (1)$$

where $\mathsf{g}(\cdot)$ and $\mathsf{H}(\cdot)$ are interval extensions of the gradient and Hesse matrix of $f_i(\cdot)$, respectively.

By choosing a variable x_j, we can obtain a univariate formula: $x \in \mathbf{x} \rightarrow f_i(x) \in \mathbf{a}\boldsymbol{v}_j^2 + \mathbf{b} \cdot \boldsymbol{v}_j + \mathbf{c}$, where $v_j = x - \check{x}_j$.

Obviously:

$$\begin{aligned}
\mathbf{a} &= \frac{1}{2}\mathsf{H}_{jj}(\mathbf{x}), \\
\mathbf{b} &= \mathsf{g}_j(\check{x}) + \sum_{k \neq j} \mathsf{H}_{jk}(\mathbf{x}) \cdot (\mathbf{x}_k - \check{x}_k), \\
\mathbf{c} &= \frac{1}{2} \sum_{k \neq j} \Big(\mathsf{g}_k(\check{x}) \cdot (\mathbf{x}_k - \check{x}_k) + \mathsf{H}_{jk}(\mathbf{x}) \cdot (\mathbf{x}_k - \check{x}_k)^2 \Big) \\
&\quad + \sum_{k=1, k \neq j}^{n} \; \sum_{l=k+1}^{n} \mathsf{H}_{jk}(\mathbf{x}) \cdot (\mathbf{x}_k - \check{x}_k) \cdot (\mathbf{x}_l - \check{x}_l).
\end{aligned}$$

3.3 Interval Quadratic Equations

A quadratic equation is a well-known equation type of the form $ax^2 + bx + c = 0$. Methods of solving this equation in real numbers are common knowledge, nowadays.

How can such methods be generalized to the case when $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are intervals and we have an interval $\mathbf{x}$ of possible values of the variable? Below, we present two solutions: a straightforward one and a more sophisticated one, based on the algorithm, described by Hansen [8].

The Straightforward Approach. This approach is a simple (yet not naïve) “intervalization” of the point-wise algorithm. It is provided by the author, but it also resembles techniques of [6].

Please note, it is assumed that the interval $\mathbf{a}$ is either strictly positive or strictly negative; for $a = 0$ the formulae for the quadratic equation do not make sense – even using extended interval arithmetic is of little help.

So, as for the non-interval case, we start with computing the discriminant of the equation: $\boldsymbol{\Delta} = \mathbf{b}^2 - 4\mathbf{ac}$. If all possible values of $\boldsymbol{\Delta}$ are guaranteed to be negative, i.e., $\overline{\Delta} < 0$ then for no quadratic approximation can there be any solutions, and we can discard the box $\mathbf{x}$. Otherwise, we set: $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} \cap [0, +\infty]$ and compute the values:

$$\mathbf{x}^{(1)} = \frac{-\mathbf{b} - \sqrt{\boldsymbol{\Delta}}}{2\mathbf{a}}, \qquad \mathbf{x}^{(2)} = \frac{-\mathbf{b} + \sqrt{\boldsymbol{\Delta}}}{2\mathbf{a}}. \tag{2}$$

Please note, we cannot use the Viete formula $x^{(1)}x^{(2)} = \frac{c}{a}$ to compute one of the roots – $\mathbf{c}$ (and the other root) will often contain zero. However, we can use the Viete formula for narrowing:

$$\mathbf{x}^{(1)} \leftarrow \mathbf{x}^{(1)} \cap \frac{\mathbf{c}}{\mathbf{ax}^{(2)}}, \qquad \mathbf{x}^{(2)} \leftarrow \mathbf{x}^{(2)} \cap \frac{\mathbf{c}}{\mathbf{ax}^{(1)}}. \tag{3}$$

The two “interval solutions $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ can either be disjoint or not (if $0 \in \boldsymbol{\Delta}$ then they always coincide, but for strictly positive $\boldsymbol{\Delta}$ they can coincide, also). Now we can have the following possibilities:

– two disjoint intervals, both having nonempty intersections with $\mathbf{x}$ – the domain has been split, as for the Newton operator with extended arithmetic,
– two disjoint intervals, but only one of them has a nonempty intersection with $\mathbf{x}$ – then we can contract the domain; if the interval solution belongs to the interior of $\mathbf{x}$, we can prove the existence of a solution (as for the Newton operator),
– two coinciding intervals and at least one of them coincides with $\mathbf{x}$ – we narrow the domain, but cannot prove the existence or uniqueness of a solution,
– intervals disjoint with $\mathbf{x}$ – then we can discard this area, obviously.

Hansen’s Approach. The book of Hansen and Walster [8] presents a sophisticated and efficient approach to solve quadratic equations with interval coefficients. This approach is applicable if $0 \in \mathbf{a}$, also.

The essence is to consider upper and lower functions of $\mathsf{f}(x) = [\underline{f}(x), \overline{f}(x)]$ and find x’s, where $\underline{f}(x) \leq 0 \leq \overline{f}(x)$. It is assumed that $\overline{a} > 0$; if it is not, we can multiply both sides of the equation by (-1).

Please note, the upper and lower functions can be expressed as follows:

$$\underline{f}(x) = \begin{cases} \underline{a}x^2 + \underline{b}x + \underline{c} \text{ for } & x \geq 0 \\ \underline{a}x^2 + \overline{b}x + \underline{c} \text{ for } & x < 0 \end{cases}, \tag{4}$$

$$\overline{f}(x) = \begin{cases} \overline{a}x^2 + \overline{b}x + \overline{c} \text{ for } & x \geq 0 \\ \overline{a}x^2 + \underline{b}x + \overline{c} \text{ for } & x < 0 \end{cases}. \tag{5}$$

The condition $\overline{a} > 0$ implies that the upper function $\overline{f}(x)$ is convex. The lower function can be either convex, concave or neither convex nor concave.

The algorithm can be presented as follows:

```
initialize the list $S$ of points (represented by narrow intervals);
compute roots of $\underline{f}(x)$ and put them to the list $S$;
similarly, compute roots of $\overline{f}(x)$ and put them to the list $S$;
put $-\infty$ and/or $+\infty$ to $S$ if $\underline{f}(x) \leq 0 \leq \overline{f}(x)$ is fulfilled for these limits;
sort the list $L$ with respect to lower bounds of the entries;
if (there are no entries in $S$) then the equation has no solutions;
if (there are exactly two entries in $S$: $s_1$ and $s_2$) then
    the equation has one interval solution $[s_1, s_2]$;
if (there are exactly four entries in $S$: $s_1, \ldots, s_4$) then
    the equation has two interval solutions: $[s_1, s_2]$ and $[s_3, s_4]$;
if (there are exactly six entries in $S$: $s_1, \ldots, s_6$) then
    the equation has three interval solutions: $[s_1, s_2]$, $[s_3, s_4]$ and $[s_5, s_6]$;
```

In [8] it is specified that double roots should be stored twice on the list. Our implementation does not distinguish the cases when the discriminant $\Delta > 0$ or $\Delta = 0$, as it would be very difficult from the numerical point of view. So, the double root can be represented by two very close, probably coinciding, but different intervals.

Also, it is proven in the book that other numbers of solutions are not possible. The case of three interval solutions can occur when $\underline{f}(x)$ is nonconvex and it has four roots – two positive and two negative ones.

3.4 When to Use the Quadratic Approximation?

As stated above, crucial for designing successful interval algorithms is the choice of a proper heuristic to choose, arrange and parameterize interval tools for a specific box.

It seems reasonable to formulate the following advice *a priori*:

– not to use it when traditional (less costly) Newton operators, perform well,
– not to use it on too wide boxes – the ranges of approximations will grow at least in a quadratic range with the size,
– probably, also not to use it on too narrow boxes – higher order Taylor models do not have a better convergence than 1st order centered forms [18],
– if the Newton operator could not reduce one of the components as there were zeros in both the nominator and the denominator of the formula (see, e.g., [13]); this indicates that the box might contain a singular (or near-singular) point – we assume the Newton operator sets the flag `singular` in such a case.

In particular, the following heuristic appeared to perform well:

```
heuristic_for_use_of_quadratic_approximation
perform the Newton operator of some type on $\mathbf{x}$;
if ($\mathbf{x}$ was split or a solution has been verified) then return;
if (diameters of less than $m$ components of $\mathbf{x}$ do not exceed the value
     $\max\left(\frac{16.0}{n}, 1.0\right)$) then return; // the box is too large
if (diameters of more than $n - m$ components of $\mathbf{x}$ exceed the value
     $\frac{2.5}{2+n}$) then return; // the box is too small
if (some component of $\mathbf{x}$ has been narrowed on both sides) then return;
for ($k = 1$; $k \leq m$; ++$k$) do
     if (the $k$-th equation has at least one quadratic term) then
          compute the Hesse matrix of $f_k(\cdot)$ on $\mathbf{x}$;
          compute the function value and gradient at the midpoint of $\mathbf{x}$;
          for each variable, compute coefficients $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ of the quadratic
          approximation and try to solve the equation;
     end if
end for
end heuristic_for_use_of_quadratic_approximation
```

The above procedure does not take singularities into account. We can modify it, by changing the proper line to:

```
if (not singular and diameters of more than $n - m$ components of $\mathbf{x}$ exceed
     the value $\frac{2.5}{2+n}$) then return;
```

Both versions of the heuristic will be called: “basic” and “singularity checking” respectively, in the remainder.

4 Computational Experiments

Numerical experiments were performed on a computer with 16 cores, i.e., 8 Dual-Core AMD Opterons 8218 with 2.6 GHz clock. The machine ran under control of a Fedora 15 Linux operating system with the GCC 4.6.3, glibc 2.14 and the Linux kernel 2.6.43.8.

The solver is written in C++ and compiled using GCC compiler. The C-XSC library (version 2.5.3) [1] was used for interval computations. The parallelization (8 threads) was done with TBB 4.0, update 3 [2]. OpenBLAS 0.1 alpha 2.2 [3] was linked for BLAS operations.

According to previous experience (see [12]), 8 parallel threads were used to decrease computation time. Please note that parallelization does not affect the number of iterations, but the execution time only.

The following test problems were considered.

The first one is called the Hippopede problem [11,17] – two equations in three variables. Accuracy $\varepsilon = 10^{-7}$ was set.

The second problem, called Puma, arose in the inverse kinematics of a 3R robot and is one of typical benchmarks for nonlinear system solvers [4]. In the above form it is a well-determined (8 equations and 8 variables) problem with 16 solutions that are easily found by several solvers. To make it underdetermined the two last equations were dropped (as in [11]). The variant with 6 equations was considered in numerical experiments as the third test problem. Accuracy $\varepsilon = 0.05$ was set.

The third problem is well-determined – it is called Box3 [4] and has three equations in three variables. Accuracy ε was set to 10^{-5}.

The fourth one is a set of two equations – a quadratic one and a linear one – in five variables [7]. It is called the Academic problem. Accuracy $\varepsilon = 0.05$

The fifth problem is called the Brent problem – it is a well-determined algebraic problem, supposed to be "difficult" [5]. Presented results have been obtained for $N = 10$; accuracy was set to 10^{-7}.

And the last one is a well-determined one – the well-known Broyden-banded system [4,11]. In this paper we consider the case of $N = 16$. The accuracy $\varepsilon = 10^{-6}$ was set.

Results are given in Tables 1–4. The following notation is used in the tables:

- fun.evals, grad.evals, Hesse evals – numbers of functions evaluations, its gradients and Hesse matrices evaluations,
- bisecs – the number of boxes bisections,
- preconds – the number of preconditioning matrix computations (i.e., performed Gauss-Seidel steps),
- bis. Newt, del. Newt – numbers of boxes bisected/deleted by the Newton step,
- q.solv – the number of quadratic equations the algorithm was trying to solve,
- q.del.delta – the number of boxes deleted, because the discriminant of the quadratic equation was negative,
- q.del.disj. – the number of boxes deleted, because the solutions of a quadratic equation were disjoint with the original box,
- q.bisecs – the number of boxes bisected by the quadratic equations solving procedure,
- pos.boxes, verif.boxes – number of elements in the computed lists of boxes containing possible and verified solutions,
- Leb.pos., Leb.verif. – total Lebesgue measures of both sets,
- time – computation time in seconds.

5 Analysis of the Results

The proposed new version of our B&P algorithm resulted in an excellent speedup for the Brent problem (for one of the algorithm versions – for the Hippopede problem, also) and a significant one for Broyden16. For Puma6 the improvement was marginal and for problems Box3 and Academic, the new algorithm performed slightly worse than the version from [14]. It is worth noting that for the Academic problem, although the computation time was slightly worse, the accuracy was a bit better.

Table 1. Computational results for the algorithm version from [14]

Problem	Hippopede	Puma6	Box3	Academic	Brent10	Broyden16
fun. evals	440450	3620757	2387475	5568107	43399916	2894815943
grad.evals	502708	3162744	2278533	4776696	65109780	835991376
Hesse evals	—	—	—	—	—	—
bisections	115947	263181	379718	1193829	2822816	25765546
preconds	219599	447788	523947	2165486	2709154	8542793
bis. Newt.	13	99	27	92	432298	357371
del. Newt.	24209	53491	236390	208841	441141	18290280
pos.boxes	43210	184888	0	886722	473	0
verif.boxes	17069	2520	1	91	805	1
Leb.poss.	8e-18	3e-9	0.0	0.028	9e-82	0.0
Leb.verif.	0.003	3e-7	1e-25	1e-5	3e-69	4e-137
time (s)	<1	4	2	8	86	2752

Table 2. Computational results for the algorithm version using the quadratic approximation as described in Sect. 3

Problem	Hippopede	Puma6	Box3	Academic	Brent10	Broyden16
fun. evals	240659	3509119	2365265	5652078	43055720	2670617370
grad.evals	271652	3103980	2297800	4850533	64434484	787247696
Hesse evals	44	3786	32734	2547	7134	4135728
bisections	60353	257957	377474	1211743	2788715	24112367
preconds	119833	442924	523643	2197282	2672319	8736413
bis. Newt.	9	103	27	58	432276	358069
del. Newt.	14130	52539	235653	212840	439606	17048221
q.solv.	81	7480	65327	12691	19577	22768329
q.del.delta	0	0	0	0	0	9634
q.del.disj.	0	8	135	0	92	11924
q.bisecs	0	4	0	0	23	698
pos.boxes	21254	181240	0	898837	476	0
verif.boxes	9230	2424	1	88	803	1
Leb.poss.	4e-18	2e-9	0.0	0.027	9e-82	0.0
Leb.verif.	0.005	3e-7	1e-25	7e-6	3e-69	3e-137
time (s)	<1	4	2	7	87	2627

It seems, it is difficult to improve the performance of the algorithm, using the 2nd order information – yet possible, at least for some problems.

It is worth noting that the algorithm cannot improve the performance on bilinear problems – like the Rheinboldt problem, considered in the author's earlier papers (e.g., [11,13,14]). In such cases, we can try to transform the problem using symbolic techniques, e.g., the Gröbner basis theory (see, e.g., [15] and the references therein), but performance of this approach has yet to be investigated.

Table 3. Computational results for the algorithm version using the quadratic approximation solved by the Hansen method [8] and the basic heuristic

Problem	Hippopede	Puma6	Box3	Academic	Brent10	Broyden16
fun. evals	243336	3385431	2206661	5662094	42671592	2263804495
grad.evals	274676	2941828	2160304	4859064	63752945	691351646
Hesse evals	46	3586	29890	2822	5945	464702
bisections	60912	244445	355031	1213849	2757100	21219462
preconds	121182	417284	494074	2200090	2638126	5865559
bis. Newt.	10	111	28	46	429876	368942
del. Newt.	14174	52131	220406	212714	436998	14621392
q.solv.	87	7152	238676	14100	16410	2517097
q.del.delta	0	0	0	0	0	7915
q.del.disj.	0	8	105	0	91	7465
q.bisecs	1	4	0	0	83	11571
pos.boxes	21288	171496	0	900178	473	0
verif.boxes	9546	2384	1	100	804	1
Leb.poss.	4e-18	3e-9	0.0	0.027	9e-82	0.0
Leb.verif.	0.005	3e-7	1e-25	7e-6	3e-69	3e-137
time (s)	<1	4	2	7	84	2304

Table 4. Computational results for the algorithm version using the Hansen method [8] and the singularity checking version of the heuristic

Problem	Hippopede	Puma6	Box3	Academic	Brent10	Broyden16
fun. evals	556401	3314879	1959329	5612877	18681704	2251193535
grad.evals	635136	2927917	2036653	4870740	18875731	673122158
Hesse evals	1034	28890	539611	54090	571691	666926
bisections	150912	241173	249471	1203934	606847	20641684
preconds	277789	413004	411630	2179986	257325	5077374
bis. Newt.	7	111	26	43	307988	370729
del. Newt.	35268	50611	243524	2041267	318117	14409198
q.solv.	2063	57624	1079109	270431	1537925	3608072
q.del.delta	0	0	0	0	107	14095
q.del.disj.	0	144	107	0	20346	11195
q.bisecs	1	4	0	3	15847	13642
pos.boxes	65184	168944	0	897399	394	0
verif.boxes	11392	1920	1	99	811	1
Leb.poss.	5e-18	3e-9	0.0	0.027	2e-83	0.0
Leb.verif.	0.002	6e-9	1e-25	8e-6	2e-48	2e-144
time (s)	<1	4	3	7	31	2251

Performance of the method for the Hippopede problem is surprising – the algorithm version using the straightforward approach is the best there and the "singularity checking" heuristic version performs the worst. This remains to be carefully investigated in the future.

6 Conclusions

The proposed additional tool for interval branch-and-prune procedures, using quadratic approximations, allows us to improve the performance for some problems. The obtained improvement was minor for some cases, but significant, e.g., for the Broyden-banded problem and dramatic for the hard Brent problem. When the use of this tool is crucial, is going to be the subject of future research.

References

1. C-XSC interval library. http://www.xsc.de
2. Intel Threading Building Blocks. http://www.threadingbuildingblocks.org
3. OpenBLAS library. http://xianyi.github.com/OpenBLAS/
4. Non-polynomial nonlinear system benchmarks. https://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/node2.html
5. Difficult benchmark problems. http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/node6.html
6. Domes, F., Neumaier, A.: Constraint propagation on quadratic constraints. Constraints **15**(3), 404–429 (2010)
7. Goldsztejn, A., Jaulin, L.: Inner and outer approximations of existentially quantified equality constraints. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 198–212. Springer, Heidelberg (2006)
8. Hansen, E., Walster, W.: Global Optimization Using Interval Analysis. Marcel Dekker, New York (2004)
9. Kearfott, R.B.: Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht (1996)
10. Kearfott, R.B., Nakao, M.T., Neumaier, A., Rump, S.M., Shary, S.P., van Hentenryck, P.: Standardized notation in interval analysis. http://www.mat.univie.ac.at/~neum/software/int/notation.ps.gz (2002)
11. Kubica, B.J.: Interval methods for solving underdetermined nonlinear equations systems. SCAN 2008 Proceedings. Reliable Comput. **15**(3), 207–217 (2011).
12. Kubica, B.J.: Shared-memory parallelization of an interval equations systems solver - comparison of tools. Pr. Nauk Politech. Warszawskiej. Elektron. **169**, 121–128 (2009)
13. Kubica, B.J.: Tuning the multithreaded interval method for solving underdetermined systems of nonlinear equations. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2011, Part II. LNCS, vol. 7204, pp. 467–476. Springer, Heidelberg (2012)
14. Kubica, B. J.: Excluding regions using Sobol sequences in an interval branch-and-prune method for nonlinear systems. Presented ta SCAN2012 Conference, submitted to Reliable Computing.
15. Kubica, B.J., Malinowski, K.: An interval global optimization algorithm combining symbolic rewriting and componentwise Newton method applied to control a class of queueing systems. Reliable Comput. **11**(5), 393–411 (2005)
16. Kubica, B.J., Woźniak, A.: Tuning the interval algorithm for seeking Pareto sets of multi-criteria problems. In: Manninen, P., Öster, P. (eds.) PARA 2012. LNCS, vol. 7782, pp. 504–517. Springer, Heidelberg (2013)

17. Neumaier, A.: The enclosure of solutions of parameter-dependent systems of equations. In: Moore, R. (ed.) Reliability in Computing. Academic Press, San Diego (1988)
18. Neumaier, A.: Taylor forms - use and limits. Reliable Comput. **9**, 43–79 (2003)
19. Shary, S. P.: Finite-difference Interval Analysis. XYZ (2010) (in Russian)