# Performance inversion of interval Newton narrowing operators

Bartłomiej Jacek Kubica[1*]

[1] Warsaw University of Technology, Institute of Control and Computation Engineering, ul. Nowowiejska 15/19, 00–665 Warsaw, Poland, email: bkubica@elka.pw.edu.pl

Key words: interval computations, underdetermined systems of equations, Newton operators

**Abstract** This paper describes a phenomenon of performance inversion of Newton operators – more precise operators might result in longer computation of a branch-and-prune method. Examples are presented and possible reasons of this behavior are discussed.

## 1 Introduction

In previous papers of the author (see [4], [5]) an interval solver of underdetermined equations systems was presented. A prepared reader might notice a strange phenomenon in the numerical results presented there – in some cases the use of more precise narrowing operators leads to slowdown of the algorithm. This phenomenon – we shall call it "performance inversion" was not analyzed or named there. It is going to be considered in this paper in details.

The mechanism the solver is based on are 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; interested reader is referred to several papers and textbooks, e.g. [1], [2], [7].

The solver was based on the branch-and-prune schema that can be expressed by the following pseudocode:

`IBP` ($\boldsymbol{x}^{(0)}$; `f`)
// $\boldsymbol{x}^{(0)}$ is the initial box, `f`$(\cdot)$ is the interval extension of the function $f\colon \mathbb{R}^n \to \mathbb{R}^m$
// $L_{ver}$ 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
$L = L_{ver} = L_{pos} = \emptyset$ ;
$\boldsymbol{x} = \boldsymbol{x}^{(0)}$ ;
`loop`

*The research has been supported by the Polish Ministry of Science and Higher Education under grant N N514 416934. The computer on which experiments were performed is shared with the Institute of Computer Science of our University. Thanks to Jacek Błaszczyk for maintaining it.

process the box $\boldsymbol{x}$, using the rejection/reduction tests ;
`if` ($\boldsymbol{x}$ does not contain solutions) `then` discard $\boldsymbol{x}$ ;
`else if` ($\boldsymbol{x}$ is verified to contain a segment of solution manifold) `then` push ($L_{ver}$, $\boldsymbol{x}$) ;
`else if` (tests subdivided $\boldsymbol{x}$ into $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$) `then`
    $\boldsymbol{x} = \boldsymbol{x}^{(1)}$ ;
    push ($L$, $\boldsymbol{x}^{(2)}$) ;
    `cycle loop` ;
`else if` ($\boldsymbol{x}$ is small enough) `then` push ($L_{pos}$, $\boldsymbol{x}$) ;
`if` ($\boldsymbol{x}$ was discarded or stored) `then`
    $\boldsymbol{x}$ = pop ($L$) ;
    `if` ($L$ was empty) `then exit loop` ;
`else`
    bisect ($\boldsymbol{x}$), obtaining $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$;
    $\boldsymbol{x} = \boldsymbol{x}^{(1)}$ ;
    push ($L$, $\boldsymbol{x}^{(2)}$) ;
`end if` ;
`end loop`

The paper [4] considered different variants of the "rejection/reduction tests" – these were several variants of interval Newton operators. Let us focus on two of them.

## 2 Newton operators

Consider a system of $m$ equations in $n > m$ variables:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ \ldots \\ f_m(x_1, x_2, \ldots, x_n) = 0 \end{cases} .$$

Such system is linearized and an iteration of Gauss-Seidel type is performed on it.

As the system is underdetermined, we can modify only $m$ out of $n$ variables in each step – without the loss of generality let us assume we modify variables with indexes $i = 1, \ldots, m$, treating the remaining variables ($i = m+1, \ldots, n$) as parameters.

The iteration can be constructed in a few ways. Let us consider two of them.

`GS_I_step` $\left(\boldsymbol{x}, \check{x}\right)$
`for` $i = 1, \ldots, m$ `do`
    compute $Y_{i:}$, the $i$-th row of the preconditioning matrix $Y$ ;
    $\boldsymbol{x}^{\text{new}} = \check{x}_i - \left(Y_{i:} \cdot \mathtt{f}(\check{x}_1, \ldots, \check{x}_n) \; + \sum_{j=1, j\neq i}^{n} Y_{i:} \cdot \boldsymbol{A}_{j:} \cdot (\boldsymbol{x}_j - \check{x}_j)\right) / (Y_{i:} \cdot A_{i:})$ ;
    `if` $\left(\boldsymbol{x}^{\text{new}} \cap \boldsymbol{x}_i = \emptyset\right)$ `then return`, signaling no solutions ;
    `replace` $\boldsymbol{x}_i$ by $\left(\boldsymbol{x}^{\text{new}} \cap \boldsymbol{x}_i\right)$ ;
`end for` ;
`end GS_I_step`

`GS_II_step` $\left(\boldsymbol{x}, \check{x}\right)$
`for` $i = 1, \ldots, m$ `do`
compute $Y_{i:}$, the $i$-th row of the preconditioning matrix $Y$ ;
$\boldsymbol{x}^{\text{new}} = \check{x}_i - \left(Y_{i:} \cdot \mathsf{f}(\check{x}_1, \ldots, \check{x}_m, \boldsymbol{x}_{m+1}, \ldots, \boldsymbol{x}_n) \; + \sum_{j=1, j\neq i}^{m} Y_{i:} \cdot \boldsymbol{A}_{j:} \cdot (\boldsymbol{x}_j - \check{x}_j)\right) / (Y_{i:} \cdot A_{i:})$ ;
`if` $\left(\boldsymbol{x}^{\text{new}} \cap \boldsymbol{x}_i = \emptyset\right)$ `then return`, signaling no solutions ;
`replace` $\boldsymbol{x}_i$ by $\left(\boldsymbol{x}^{\text{new}} \cap \boldsymbol{x}_i\right)$ ;
`end for` ;
`end GS_II_step`

In both cases $\boldsymbol{A}_{ij} = \frac{\partial \mathsf{f}_i(\boldsymbol{x})}{\partial x_j}$.

What is the difference between two above pseudocodes ? In `GS_I` we have a non-square $(n \times m)$ matrix $\boldsymbol{A}$ and "thin" free summand $\boldsymbol{b} = Y_{i:} \cdot \mathsf{h}(\check{x}_1, \ldots, \check{x}_n)$. In `GS_II` $\boldsymbol{A}$ is a square $m \times m$ matrix and the free summand $\boldsymbol{b} = Y_{i:} \cdot \mathsf{h}(\check{x}_1, \ldots, \check{x}_m, \boldsymbol{x}_{m+1}, \ldots \; , \boldsymbol{x}_n)$ is not thin.

It is actually equivalent to using the mean value form and a natural interval extension of a function $\widetilde{f}_i(x_{m+1}, \ldots, x_n) = f_i(\check{x}_1, \ldots, \check{x}_m; x_{m+1}, \ldots, x_n)$.

Which of them is better ? We shall consider it in Section 4. Now, let us describe test problems that will be used to compare the operators.

## 3 Test problems

The following problems were considered in numerical experiments.

The Puma problem ([6]) arose in the inverse kinematics of a 3R robot and is one of typical benchmarks for nonlinear system solvers:

$$
\begin{aligned}
& x_1^2 + x_2^2 - 1 = 0 \; , \;\; x_3^2 + x_4^2 - 1 = 0 \; , \\
& x_5^2 + x_6^2 - 1 = 0 \; , \;\; x_7^2 + x_8^2 - 1 = 0 \; , \\
& 0.004731 x_1 x_3 - 0.3578 x_2 x_3 - 0.1238 x_1 - 0.001637 x_2 - 0.9338 x_4 + x_7 = 0 \; , \\
& 0.2238 x_1 x_3 + 0.7623 x_2 x_3 + 0.2638 x_1 - 0.07745 x_2 - 0.6734 x_4 - 0.6022 = 0 \; , \\
& x_6 x_8 + 0.3578 x_1 + 0.004731 x_2 = 0 \; , \\
& -0.7623 x_1 + 0.2238 x_2 + 0.3461 = 0 \; , \\
& x_1, \ldots, x_8 \in [-1, 1] \; .
\end{aligned}
\tag{1}
$$

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 last equation was dropped (as in [4]). The variant with 7 equations was considered in numerical experiments as second test problem. Accuracy $\varepsilon = 10^{-4}$ was set.

The second problem is a single circle on the plane:

$$
x_1^2 + x_2^2 - 4 = 0 \; , \;\; x_1, x_2 \in [-3, 5] \; . \tag{2}
$$

And another curve on a plane; it is the sum of two concentric circles ([4]):

$$
(x_1^2 + x_2^2 - 4) \cdot (x_1^2 + x_2^2 - 1) = 0 \; , \;\; x_1, x_2 \in [-3, 5] \; . \tag{3}
$$

## 4 Efficiency of the operators

Now, let us discuss the performance of both Newton operators. Tables with computational results are given in next section. This layout might be inconvenient a bit, but – as actual experiments that were done rely on the analysis here – it seems the only natural schema.

According to textbooks, e.g. [1], [2], [7] the mean value form is inferior with respect to natural interval extension for large intervals and superior for "sufficiently small" ones. On the other hand, for relatively simple functions and sparse systems of equations one might expect the `GS_II` method to be more precise.

So, let us have a look at the experimental results. Consider the Puma problem (all test problems are described in Section 3). Let us investigate the performance of both methods. For typical parameters of the program (first two columns of Table 1), the "better" variant `GS_II` requires more than ten times more gradient evaluations than the "worse" one ! What happened ?!

We already said that the mean value form is sometimes more precise than the natural extension. This might explain the observed phenomenon. Maybe we should switch from one interval extension to the another as diameters of the boxes decrease ?

It does not seem to be the case. Let us check a simpler example – Problem (2). Now both variables appear only once in the formula and the natural extension gives us exact bounds; the mean value form cannot be more precise. Again the `GS_II` is slower – it requires about 50 times more gradient evaluations ! Clearly, we have a situation, where a more precise operator results in a worse efficiency of the algorithm – as declared in Section 1, we shall call it the *performance inversion* of the operators.

The third example – Problem (3) behaves in the same way. To make things even less understandable, the idea of switching the interval extension actually gives us much improvements – even for Problem (2). So, what is the reason of this surprising behavior ?

### Other possible reasons

Let us consider other hypotheses.

**Newton operator vs. bisection.** One of the explanations might be the interference between the Newton operator and bisection. As Newton operators cut parts of boxes that do not contain solutions, solutions are quite likely to lie near the center of the resulting box. The better Specific operator is, the more likely the solution will be in the midpoint.

But after the Newton step bisection is usually done, splitting the box in the middle. Hence solutions will lie on boundaries on the considered boxes, making it more difficult to verify them.

If this is the reason of observed performance inversion, replacing bisection with e.g. trisection should help.

As can be seen in corresponding columns of Tables 1-3 this variant has been tested. Results have been surprising – the phenomenon disappeared for Problem (3), but not for other two examples. Moreover for Problems (1) and (3) results of algorithms based on trisection were worse than switching the operators with bisection.

So, such interference cannot be the only reason of performance inversion.

**Selecting verified boxes.** Using the Newton operators, the algorithm tries to verify if a box contains a segment of the solution manifold. If it does, such box is not processed further, but is stored as a verified solution. This way relatively large boxes can sometimes be removed from further processing.

This feature also seems likely to cause the performance inversion – some operators might reduce the boxes less efficiently, but instead verify them to contain solutions.

Results in Tables 4-6 falsify this hypothesis, too. They present results for the variant of our algorithms that do not verify boxes to contain the solution, but only discard infeasible boxes and store small ones. Differences between operators `GS_I` and `GS_II` are definitely smaller than in Tables 1-3, but the performance inversion phenomenon is still clearly observed.

## 5 Numerical experiments

Numerical experiments were performed on a computer with 16 cores, i. e. 8 Dual-Core AMD Opterons 8218 with 2.6GHz clock. The machine ran under control of a Fedora 10 Linux operating system. The solver was implemented in C++, using C-XSC 2.2.3 library for interval computations. The GCC 4.3.2 compiler was used.

Computations were done in parallel, using TBB library (see [9] and [5]). All runs of the program were done with 8 threads – numbers of objective evaluations, gradient evaluations, bisections, etc. do not rely on the number of threads, so the number giving higher observed efficiency (see [5]) has been chosen.

Tables 1-6 contain basic computational results for all three test problems. The numbers after method's name mean the number of boxes on which a box is subdivided; 2 means bisection, 3 – trisection. The annotation NVB means "no verified boxes", i. e. the variant of the algorithm that does not select boxes that are verified to contain solutions.

**Table 1.** Computational results for Problem (1) without the last equation, $\varepsilon = 10^{-4}$.

| method | `GS_I`, 2 | `GS_II`, 2 | switching, 2 | `GS_I`, 3 | `GS_II`, 3 | switching, 3 |
|---|---|---|---|---|---|---|
| fun.evals | 3539627 | 39040127 | 3078915 | 3254272 | 45806362 | 3005674 |
| grad.evals | 4022004 | 47802356 | 3492916 | 4028066 | 65069242 | 3700200 |
| bisecs | 279139 | 3410079 | 242415 | 171276 | 3071724 | 154590 |
| bis.Newt. | 274 | 270 | 262 | 398 | 518 | 484 |
| del.Newt. | 72988 | 900076 | 63652 | 98174 | 2121032 | 89096 |
| pos.boxes | 116076 | 1245564 | 101936 | 98704 | 1218112 | 85708 |
| verif.boxes | 21440 | 12964 | 17948 | 35532 | 52984 | 35644 |
| Leb.pos. | 4e-32 | 2e-29 | 2e-32 | 1e-32 | 7e-30 | 1e-32 |
| Leb.verif. | 3e-11 | 6e-12 | 1e-11 | 2e-11 | 3e-10 | 3e-13 |
| time (sec.) | 28 | 428 | 26 | 29 | 546 | 26 |

**Table 2.** Computational results for Problem (2), $\varepsilon = 10^{-5}$.

| method | GS_I, 2 | GS_II, 2 | switching, 2 | GS_I, 3 | GS_II, 3 | switching, 3 |
|---|---|---|---|---|---|---|
| fun.evals | 1126 | 66335 | 621 | 575 | 1514 | 1024 |
| grad.evals | 1336 | 66426 | 698 | 722 | 1688 | 1198 |
| bisecs | 465 | 32862 | 249 | 148 | 248 | 199 |
| bis.Newt. | 1 | 1 | 1 | 0 | 0 | 0 |
| del.Newt. | 0 | 0 | 0 | 0 | 0 | 0 |
| pos.boxes | 34 | 31660 | 13 | 12 | 22 | 16 |
| verif.boxes | 224 | 1114 | 162 | 139 | 302 | 210 |
| Leb.pos. | 8e-10 | 1e-6 | 2e-10 | 2e-10 | 4e-10 | 2e-10 |
| Leb.verif. | 3.6512 | 2.2074 | 2.2108 | 2.2713 | 1.8462 | 1.8712 |
| time (sec.) | 0.02 | 0.4 | 0.02 | 0.02 | 0.03 | 0.02 |

**Table 3.** Computational results for Problem (3), $\varepsilon = 10^{-5}$.

| method | GS_I, 2 | GS_II, 2 | switching, 2 | GS_I, 3 | GS_II, 3 | switching, 3 |
|---|---|---|---|---|---|---|
| fun.evals | 6784 | 140979 | 5442 | 10165 | 6909 | 9752 |
| grad.evals | 8006 | 141922 | 6438 | 12034 | 8338 | 11509 |
| bisecs | 2576 | 69909 | 2234 | 1921 | 1691 | 1834 |
| bis.Newt. | 4 | 27 | 27 | 2 | 21 | 21 |
| del.Newt. | 0 | 0 | 0 | 0 | 0 | 0 |
| pos.boxes | 160 | 65605 | 146 | 200 | 196 | 192 |
| verif.boxes | 1200 | 3390 | 1121 | 1777 | 1780 | 1742 |
| Leb.pos. | 3e-9 | 2e-6 | 3e-9 | 2e-9 | 4e-9 | 3e-9 |
| Leb.verif. | 0.6039 | 0.5789 | 0.5788 | 0.7075 | 0.6077 | 0.6616 |
| time (sec.) | 0.08 | 1.29 | 0.08 | 0.10 | 0.09 | 0.1 |

## 6 Analysis and further research

For all three investigated test problems the phenomenon of performance inversion has been observed. Changing bisection to trisection allowed to prevent it only in one case. On the other hand switching between both Newton operators gave us improvement in all three cases – it resulted in an algorithm quicker than both GS_I and GS_II always.

Tables 7-9 showed that the phenomenon also disappears for larger values of the accuracy $\varepsilon$, i. e. for larger diameters of considered boxes.

We have not found a satisfying explanation of the phenomenon of observed performance inversion, yet. Please note that in [4] a similar behavior occurred also in comparison between the componentwise and traditional Newton operators for problem (3), so the occurrence is not uncommon.

Also several other questions have to be answered. In particular: is the observed phenomenon specific to underdetermined problems or it can be encountered also for well-determined ones ? And how to choose interval tools to optimize performance of the

**Table 4.** Computational results for Problem (1) without the last equation, $\varepsilon = 10^{-4}$, NVB.

| method | GS_I, 2 | GS_II, 2 | switching, 2 | GS_I, 3 | GS_II, 3 | switching, 3 |
|---|---|---|---|---|---|---|
| fun.evals | 15253875 | 45827803 | 14981015 | 18236848 | 64106490 | 18275978 |
| grad.evals | 16176580 | 54923540 | 15851444 | 20286546 | 86037042 | 20357932 |
| bisecs | 1155195 | 3922839 | 1131983 | 965760 | 4096656 | 969102 |
| bis.Newt. | 274 | 270 | 262 | 398 | 518 | 484 |
| del.Newt. | 143716 | 943340 | 135588 | 274022 | 2615000 | 272492 |
| pos.boxes | 879940 | 1680380 | 872312 | 1365084 | 2445896 | 1368776 |
| verif.boxes | 0 | 0 | 0 | 0 | 0 | 0 |
| Leb.pos. | 2e-31 | 2e-29 | 2e-31 | 2e-31 | 9e-30 | 7e-32 |
| Leb.verif. | 0 | 0 | 0 | 0 | 0 | 0 |
| time (sec.) | 110 | 484 | 106 | 133 | 703 | 139 |

**Table 5.** Computational results for Problem (2), $\varepsilon = 10^{-5}$, NVB.

| method | GS_I, 2 | GS_II, 2 | switching, 2 | GS_I, 3 | GS_II, 3 | switching, 3 |
|---|---|---|---|---|---|---|
| fun.evals | 2954862 | 3132347 | 3130697 | 3355390 | 3363966 | 3364030 |
| grad.evals | 2955318 | 3132522 | 3130958 | 3356021 | 3364493 | 3364700 |
| bisecs | 1477657 | 1566259 | 1565477 | 1118673 | 1121497 | 1121566 |
| bis.Newt. | 1 | 1 | 1 | 0 | 0 | 0 |
| del.Newt. | 0 | 0 | 0 | 0 | 0 | 0 |
| pos.boxes | 1477204 | 1566087 | 1565219 | 2236717 | 2242469 | 2242464 |
| verif.boxes | 0 | 0 | 0 | 0 | 0 | 0 |
| Leb.pos. | 4e-5 | 3e-5 | 3e-5 | 2e-5 | 2e-5 | 2e-5 |
| Leb.verif. | 0 | 0 | 0 | 0 | 0 | 0 |
| time (sec.) | 11 | 14 | 12 | 14 | 12 | 14 |

overall algorithm ?

These topics should be investigated in the future.

## 7 Conclusions

A phenomenon of efficiency inversion of some Newton operators was considered. A few possible explanations were considered, specifically:

- properties of natural and mean value interval extensions – their different precision on boxes of different diameters,
- interference between the Newton operator and bisection,
- influence of the procedure of verifying relatively large boxes to contain a segment of the solution manifold.

**Table 6.** Computational results for Problem (3), $\varepsilon = 10^{-5}$, NVB.

| method | GS_I, 2 | GS_II, 2 | switching, 2 | GS_I, 3 | GS_II, 3 | switching, 3 |
|---|---|---|---|---|---|---|
| fun.evals | 4464956 | 4506800 | 4507378 | 5019094 | 4632018 | 4632368 |
| grad.evals | 4467642 | 4508470 | 4509926 | 5025150 | 4636031 | 4637963 |
| bisecs | 2233816 | 2254207 | 2254935 | 1675048 | 1545329 | 1545973 |
| bis.Newt. | 4 | 27 | 27 | 2 | 21 | 21 |
| del.Newt. | 0 | 0 | 0 | 0 | 0 | 0 |
| pos.boxes | 2231136 | 2252566 | 2252416 | 3344044 | 3086668 | 3086374 |
| verif.boxes | 0 | 0 | 0 | 0 | 0 | 0 |
| Leb.pos. | 5e-5 | 5e-5 | 5e-5 | 4e-5 | 4e-5 | 4e-5 |
| Leb.verif. | 0 | 0 | 0 | 0 | 0 | 0 |
| time (sec.) | 23 | 25 | 19 | 27 | 26 | 22 |

**Table 7.** Number of gradient evaluations for Problem (1) for different values of $\varepsilon$.

| method $\setminus$ $\varepsilon$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | 0.1 | 1.0 |
|---|---|---|---|---|---|
| GS_I | 4022004 | 821744 | 318808 | 167748 | 26796 |
| GS_II | 47802356 | 4851420 | 686924 | 195580 | 25060 |
| GS_I, NVB | 16176580 | 1543892 | 359604 | 167804 | 26796 |
| GS_II, NVB | 54923540 | 5314148 | 712124 | 195580 | 25060 |

**Table 8.** Number of gradient evaluations for Problem (2) for different values of $\varepsilon$.

| method $\setminus$ $\varepsilon$ | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | 0.1 | 1.0 |
|---|---|---|---|---|---|---|
| GS_I | 1336 | 1044 | 750 | 566 | 320 | 76 |
| GS_II | 66426 | 8632 | 1192 | 405 | 154 | 34 |
| GS_I, NVB | 2955318 | 370006 | 24478 | 3154 | 494 | 86 |
| GS_II, NVB | 3132522 | 358552 | 25534 | 3160 | 390 | 50 |

**Table 9.** Number of gradient evaluations for Problem (3) for different values of $\varepsilon$.

| method $\setminus$ $\varepsilon$ | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | 0.1 | 1.0 |
|---|---|---|---|---|---|---|
| GS_I | 8006 | 6738 | 5046 | 3482 | 922 | 178 |
| GS_II | 141922 | 22412 | 4516 | 2051 | 793 | 162 |
| GS_I, NVB | 4467642 | 558458 | 37538 | 5546 | 1010 | 178 |
| GS_II, NVB | 4508470 | 559242 | 36948 | 5298 | 940 | 162 |

It was shown that the phenomenon of performance inversion cannot be fully explained by any of the above features. As all of them seem to accelerate it, eliminating any of them does not remove the phenomenon.

As performance of the algorithm varies quite significantly (even 50 times), it seems necessary to elaborate useful heuristics to choose interval tools (variant of Newton operator, variant of bisection/multisection, etc.) suitable for a specific problem.

## Bibliography


[1] E. Hansen, G. W. Walster, "*Global Optimization Using Interval Analysis, Second Edition: Revised and Expanded*", Marcel Dekker, New York, 2004.

[2] R. B. Kearfott, "*Rigorous Global Search: Continuous Problems*", Kluwer Academic Publishers, Dordrecht, 1996.

[3] R. B. Kearfott, M. T. Nakao, A. Neumaier, S. M. Rump, S. P. Shary, P. van Hentenryck, "*Standardized notation in interval analysis*", available on the web at `http://www.mat.univie.ac.at/~neum/software/int/notation.ps.gz` .

[4] B. J. Kubica, "*Interval methods for solving underdetermined nonlinear equations systems*", presented at SCAN 2008, El Paso, Texas, 2008.

[5] B. J. Kubica, "*Shared-Memory Parallelization of an Interval Equations Systems Solver – Comparison of Tools*", 2009, this volume.

[6] A. Neumaier, "*The Enclosure of Solutions of Parameter-Dependent Systems of Equations*", in *Reliability in Computing* (ed. Moore, R), Academic Press, 1988.

[7] A. Neumaier, "*Interval methods for systems of equations*", Cambridge University Press, Cambridge, 1990.

[8] C-XSC interval library, `http://www.xsc.de` .

[9] Intel Threading Building Blocks `http://www.threadingbuildingblocks.org` .