Excluding regions using Sobol sequences in an interval branch-and-prune method

Bartłomiej Jacek Kubica

Institute of Control and Computation Engineering
Warsaw University of Technology

SCAN 2012
Novosibirsk

Background

- Interval methods provide us several powerful tools for solving nonlinear systems, e.g.:
  - various kinds of interval Newton operator,
  - various consistency operators,
  - other constraint propagation/satisfaction tools,
  - ...
- Question: **What is crucial for the efficiency (or its lack) of an interval method for solving a specific problem?**

Background

- Interval methods provide us several powerful tools for solving nonlinear systems, e.g.:
  - various kinds of interval Newton operator,
  - various consistency operators,
  - other constraint propagation/satisfaction tools,
  - ...
- Question: **What is crucial for the efficiency (or its lack) of an interval method for solving a specific problem?**
  - Answer: developing a proper heuristic for **choosing**, **parameterizing** and **arranging** adequate tools to process specific data.

Considered algorithm

- We try to solve a system of nonlinear equations.
- Focus on:
  - Tools targeted for underdetermined systems (more variables than equations).
  - Multithreaded safety.
- Used tools:
  - Branch-and-prune schema.
  - Interval Newton operators (switching between two versions: Ncmp and GS).
  - Shared-memory parallelization using Intel TBB (Threading Building Blocks).
- Advanced heuristics:
  - Switching Newton operators.
  - Choosing the component for bisection.

Previous papers

- B. J. Kubica, *Interval methods for solving underdetermined nonlinear equations systems*, SCAN 2008 Proceedings, Reliable Computing, Vol. 15, pp. 207 – 217 (2011).
- B. J. Kubica, *Performance inversion of interval Newton narrowing operators*, KAEiOG 2009 Proceedings, Zeszyty Naukowe PW. Elektronika, Vol. 169, pp. 111 – 119 (2009).
- B. J. Kubica, *Shared-memory parallelization of an interval equations systems solver – comparison of tools*, KAEiOG 2009 Proceedings, *ibidem*, pp. 121 – 128.
- B. J. Kubica, *Intel TBB as a tool for parallelization of an interval solver of nonlinear equations systems*, ICCE WUT technical report no 09-02, 2010.
- B. J. Kubica, *Tuning the multithreaded interval method for solving underdetermined systems of nonlinear equations*, PPAM 2011 Proceedings, LNCS, Vol. 7204, pp. 467 – 476 (2012).

The idea for improvement

- We are solving the equations system:
  $$\left(f_1(x), \ldots, f_m(x)\right)^T = 0$$
- The Newton step (a basic tool for equations systems) is time consuming.
- The use of this tool should concentrate on regions around the solution manifold.
- Other regions, i.e., regions where $f_i(x) > \varepsilon$ or $f_i(x) < -\varepsilon$ (for some *i*) can (and should) be deleted earlier – by some cheaper test, if possible.

What tools can be used?

- Solving tolerance problems: $f_i(x) \in [\varepsilon, +\infty]$,
  $f_i(x) \in [-\infty, -\varepsilon]$.
  - Linear – methods of Shary, Sharaya, Rohn...
  - Nonlinear?
- Epsilon-inflation.
- Initial choice of “seeds” of exclusion regions:
  - Random.
  - Deterministic.

Shary's method for the linear tolerance problem

- Consider the tolerable solution set (TSS) of the linear interval system: $\boldsymbol{A}\, x = \boldsymbol{b}$.
- We have a point t, from the interior of the TSS.
- Then, the following set is contained in TSS:
  $\boldsymbol{U} = t + r\,\boldsymbol{e}$, where:

$$\boldsymbol{e} = \left([-1,1],\ldots,[-1,1]\right)^T,$$

$$r = \min_{1 \le i \le m} \min_{A \in vert\, \boldsymbol{A}} \frac{\operatorname{rad} \boldsymbol{b}_i - \left| \operatorname{mid} \boldsymbol{b}_i - \sum_{j=1}^{n} a_{ij} t_j \right|}{\sum_{j=1}^{n} \left| a_{ij} \right|}.$$

Adaptation of Shary's method

… which in our case has to be modified to:

$$r = \frac{\left|\tilde{b}_i - \sum_{j=1}^{n} a_{ij} t_j\right|}{\sum_{j=1}^{n} \left|a_{ij}\right|}, \text{ where } \tilde{b}_i = \underline{b}_i \text{ or } \tilde{b}_i = \overline{b}_i.$$

And for our case it results in:

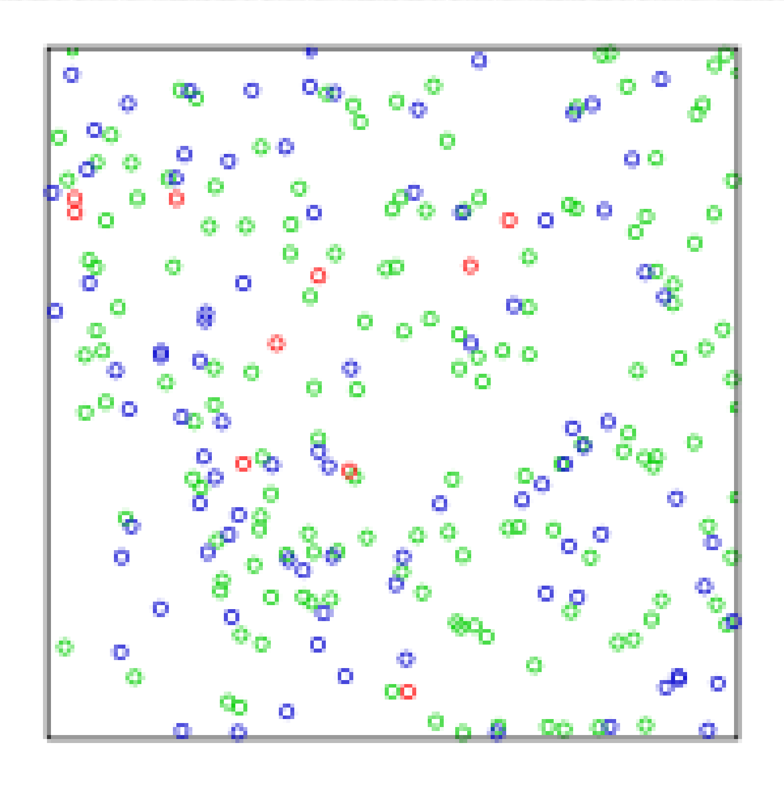
$$r = \frac{\left|f_i(t)\right| - \varepsilon}{\sum_{j=1}^{n} \left|a_{ij}\right|}.$$

Sobol sequences

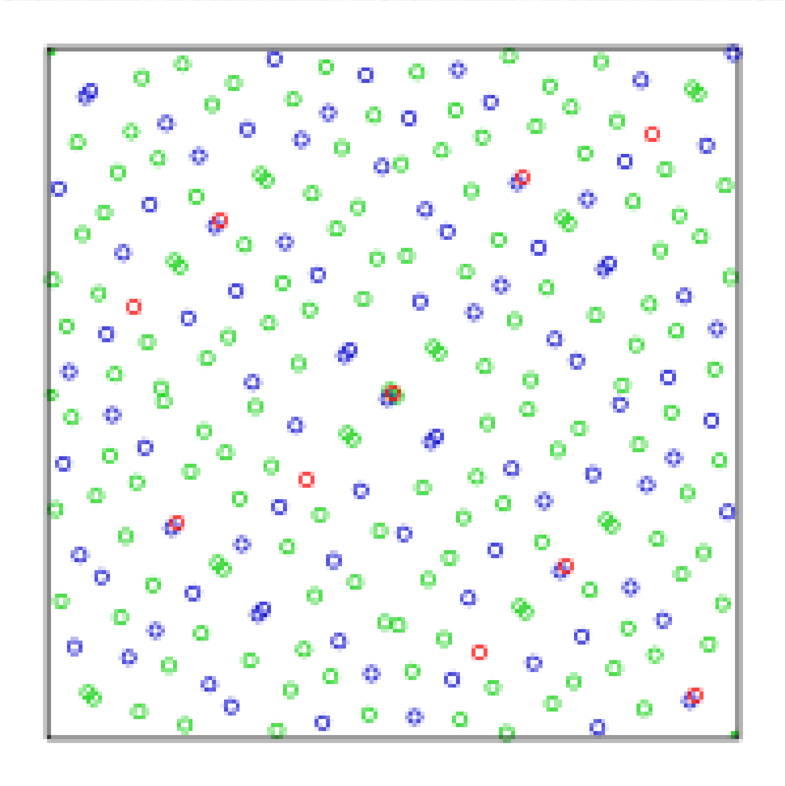
- An example of low-discrepancy sequences.
- Proposed in 1967.
- Efficient algorithms for generation: Gray code, by I. A. Antonov and V. M. Saleev.
- Efficient and convenient free and open source implementations, e.g., the one of Stephen Joe and Frances Y. Kuo: http://web.maths.unsw.edu.au/~fkuo/sobol.

Илья Меерович Соболь

Random (pseudo-random) sequences vs Sobol sequences



A pseudo-random sequence in 2D



A Sobol sequence in 2D

(pictures from the Wikipedia article on Sobol sequences)

Details

- For higher dimensions Sobol sequences require a large number of points to fill the space densely.
- But we do not need to fill the space, just to plant seeds in many different places.
- In our experiments the number of chosen points equal to n (the number of variables) performed the best.
  - At least usually.
  - There were exceptions to it.
- Sobol sequences performed much better than pseudo-random ones:
  - Better speedups.
  - More predictable behavior.

Details

- So, we propose the following "initial exclusion phase" for the branch-and-prune algorithm.
- Using the Sobol (or other) sequence, we chose *n* points from within the considered domain.
- We compute the value of one of the functions $f_i(x)$ at the chosen point $x^{(k)}$.
- If $f_i(x^{(k)}) \in [-\varepsilon, \varepsilon]$, then the point is ignored.
- The linear tolerance problem (using Shary's method) is solved for a problem $f_i(x) \in [\varepsilon, \infty]$ or $f_i(x) \in [-\infty, -\varepsilon]$, linearized around $x^{(k)}$.
- Optionally, we expand the computed region, using epsilon-inflation.

Details

- Then, the computed regions are removed from the problem domain, by a well-known procedure to compute the complement of a box (or set of boxes).
- The preprocessing phase can be parallelized easily (we use **`tbb::parallel_for`** for this purpose).
- Yet, the parallelization seems irrelevant as its time can be neglected with respect to the overall computation time.
- Preliminary results: B. J. Kubica, *Exclusion regions in the interval solver of underdetermined nonlinear systems*, ICCE internal report 12-01.

Implementation & experiments

- Environment:
  - 16 cores: 8 dual core AMD Opterons 8218, 2.6 GHz.
  - 8 threads used actually.
  - Fedora Linux 15.
  - Linux kernel 2.6.43.8.
  - Glibc 2.14.
  - GCC 4.6.3.
- Used libraries:
  - C-XSC 2.5.3.
  - TBB 4.0 update 5.
  - OpenBLAS 0.2.2.
  - Joe & Kuo Sobol sequence generator.

Test problems

Hippopede – 2 equations in 3 variables:

$$x_1^2 + x_2^2 - x_3 = 0,$$
$$x_2^2 + x_3^2 - 1.1\, x_3 = 0,$$
$$x_1 \in [-1{,}5,\, 1{,}5], \quad x_2 \in [-1, 1], \quad x_3 \in [0, 4].$$

Broyden – N equations in N variables:

$$x_i \cdot (2 + 5\, x_i^2) + 1 - \sum_{j \in J_i} x_j \cdot (1 + x_j) = 0, \quad j = 1, \dots, N,$$
$$J_i = \{\, j \,|\, j \neq i \text{ and } \max\{1, i-5\} \le j \le \min\{N, i+1\}\},$$
$$x_i \in [-100, 101], \quad i = 1, \dots, N.$$

Test problems

Rheinboldt – 5 equations in 8 variables:

$$-3.933\,x_1+0.107\,x_2+0.126\,x_3-9.99\,x_5-45.83\,x_7-7.64\,x_8\,+$$
$$-0.727\,x_2x_3+8.39\,x_3x_4-684.4\,x_4x_5+63.5\,x_4x_7=0,$$
$$-0.987\,x_2-22.95\,x_4-28.37\,x_6+0.949\,x_1x_3+0.173\,x_1x_5=0,$$
$$0.002\,x_1-0.235\,x_3+5.67\,x_5+0.921\,x_7-6.51\,x_8-0.716\,x_1x_2\,+$$
$$-1.578\,x_1x_4+1.132\,x_4x_7=0,$$
$$x_1-x_4-0.168\,x_6-x_1x_2=0,$$
$$-x_3-0.196\,x_5-0.0071\,x_7+x_1x_4=0,$$
$$x_i\in[-2,2],\quad i=1,\dots,8.$$

Computational results

	Hippopede	Rheinboldt	Broyden12	Broyden16
fun evals	1 184 664	213 645 211	23 364 196	7 975 494 792
grad evals	1 361 152	128 791 915	8 625 492	2 139 405 184
bisecs	329 911	12 225 817	337 884	66 082 093
ver.boxes	21 672	486 738	1	1
pos.boxes	149 952	7 684 286	0	0
time (sec.)	< 1	232	21	6911
fun evals	560 712	186 210 881	19 432 059	4 705 422 366
grad evals	639 616	112 809 925	6 722 376	1 257 731 440
bisecs	151 299	10 688 351	264 036	38 905 745
ver.boxes	14 557	425 256	1	1
pos.boxes	63 297	6 828 040	0	0
time (sec.)	< 1	202	16	4036

Conclusions

- Using the “initial exclusion phase” seems worthwhile and Sobol sequences perform well for “planting seeds”.
- Epsilon-inflation should be used with it.
- Speedups seem to be pretty random, but evident; very impressive for some test problems.
  - 10-30%, typically.
  - Occasionally, no speedup or a minor slowdown. :-(
  - But sometimes, the efficiency doubles!!!
- For some reason, the number of “seeds” equal to the number of variables performs best (but there are exceptions to it).