World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:4, No:11, 2010

A contractor for the symmetric solution set

Milan Hladík

Abstract—The symmetric solution set Σ_{sym} is the set of all solutions to the linear systems $Ax = b$, where A is symmetric and lies between some given bounds $\underline{A}$ and $\overline{A}$, and b lies between $\underline{b}$ and $\overline{b}$. We present a contractor for Σ_{sym}, which is an iterative method that starts with some initial enclosure of Σ_{sym} (by means of a cartesian product of intervals) and sequentially makes the enclosure tighter. Our contractor is based on polyhedral approximation and solving a series of linear programs. Even though it does not converge to the optimal bounds in general, it may significantly reduce the overestimation. The efficiency is discussed by a number of numerical experiments.

Keywords—Linear interval systems, solution set, interval matrix, symmetric matrix.

I. Introduction

Let us consider an $n \times n$ linear system of equations

$$Ax = b,$$

where A and b perturb within some given bounds as follows:

$$\underline{A} \le A \le \overline{A} \quad \text{and} \quad \underline{b} \le b \le \overline{b},$$

and the relations are understood componentwise. Introducing the interval matrix

$$\boldsymbol{A} := [\underline{A}, \overline{A}] = \{A \in \mathbb{R}^{n\times n} \mid \underline{A} \le A \le \overline{A}\},$$

and the interval vector

$$\boldsymbol{b} := [\underline{b}, \overline{b}] = \{b \in \mathbb{R}^n \mid \underline{b} \le b \le \overline{b}\},$$

the interval linear system reads

$$Ax = b,\ A \in \boldsymbol{A},\ b \in \boldsymbol{b}.$$

The solution set Σ to this interval system is defined as the set of all possible solutions, that is

$$\Sigma := \{x \mid Ax = b,\ A \in \boldsymbol{A},\ b \in \boldsymbol{b}\}$$

It is known that checking non-emptiness of Σ is NP-hard problem as well as computing its interval hull (the smallest interval vector containing Σ) [12]. To find a tight enclosure to Σ is a basic problem in the discipline of interval computing. We focus on the symmetric solution set Σ_{sym}, which is defined as

$$\Sigma_{sym} := \{x \mid Ax = b,\ A \in \boldsymbol{A},\ A = A^T,\ b \in \boldsymbol{b}\}.$$

Linear systems with symmetric matrices arise naturally in many engineering problems. Interval analysis approach was used e.g. in truss mechanics [18] and nodal analysis for linear electrical circuits [11], [29]. Symmetric matrices appear in eigenvalue problems [14]; our approach enables to compute tight approximation of eigenvectors of symmetric interval matrices. Another kinds of applications involve Markov chains [9], for instance.

The dependences due to the symmetry of A induce that Σ_{sym} is very hard to sharply approximate, and it is still a challenging problem. As $\Sigma_{sym} \subseteq \Sigma$, any enclosure of Σ is also an enclosure of Σ_{sym}. However, the symmetry must be taken into account since the overestimation of Σ over Σ_{sym} caused by relaxing the symmetry condition can be arbitrarily large [13]. Characterization of Σ_{sym} was developed in [3], [4], [5], [7], [13].

Due to the mentioned difficulties, it is not surprising that there are few methods for finding a tight enclosure of Σ_{sym}. Basically, there were two approaches studied. The Cholesky method adapted to interval data was studied in [6], [7], [8] and extended to block matrices in [27], [28]. Inclusion methods for linear interval systems with more general dependences were discussed e.g. by Jansson [15], Rump [25], Popova et al. [20], [21], [22] or by Kolev [16], [17].

Let us introduce some notations. The midpoint and the radius of an interval matrix $\boldsymbol{A}$ is defined respectively as

$$A^c := \frac{1}{2}(\underline{A} + \overline{A}), \quad A^\Delta := \frac{1}{2}(\overline{A} - \underline{A}),$$

and similarly for interval vectors. By an enclosure of a set we mean an arbitrary superset. The sign of a real number x is defined as $\operatorname{sgn}(x) = 1$ if $x \ge 0$ and $\operatorname{sgn}(x) = -1$ otherwise. Strict lexicographic ordering of vectors is denoted by $\prec_{\text{lex}}$; it means $u \prec_{\text{lex}} v$ if for some k we have $u_i = v_i$, $i < k$, and $u_k < v_k$. Finally, $A_{i,\bullet}$ isolates the i-th row of a matrix A, and $\operatorname{diag}(z)$ stands for a diagonal matrix with entries $z_1, \ldots, z_n$.

II. Method

Our method is based on a description of Σ_{sym} by means of a nonlinear system of inequalities [13]. The system is linearized by a McCormick-like method [1] and the polyhedral approximation of Σ_{sym} is used for iterative contracting in a similar way as presented in [10]: we calculate an interval hull of the polyhedral approximation and use the bounds for the next iteration. The basic scheme of the algorithm is presented below.

Let us discuss the steps of Algorithm 1 in detail. Step 1 requires some initial interval enclosure of Σ_{sym}. Since $\Sigma_{sym} \subseteq \Sigma$ we can utilize any interval enclosure to Σ as well. There are numbers of solvers for interval linear systems [2], [19], [23].

The improvement in Step 7 is measured by the ratio of component sums of $(x^i)^\Delta$ and $(x^{i-1})^\Delta$, respectively. We

M. Hladík is with the Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25, 11800, Prague, Czech Republic, e-mail: (see milan.hladik@matfyz.cz)

Algorithm 1 (Symmetric solution set contractor)

1: Compute an initial interval enclosure $\boldsymbol{x}^0 \supseteq \Sigma_{sym}$;
2: $i := 0$;
3: **repeat**
4: compute a polyhedral enclosure $\mathcal{P}$ of Σ_{sym} by using $\boldsymbol{x}^i$;
5: $i := i + 1$;
6: compute the interval hull $\boldsymbol{x}^i$ of $\mathcal{P}$;
7: **until** improvement is nonsignificant;
8: **return** $\boldsymbol{x}^i$;

terminate the loop when

$$\frac{\sum_{j=1}^{n} (x_j^i)^\Delta}{\sum_{j=1}^{n} (x_j^{i-1})^\Delta} > 0.99$$

comes true.

In step 6, the interval hull $\boldsymbol{x}^i$ of $\mathcal{P}$ is easily calculated by solving $2n$ linear programs: the bounds of $\boldsymbol{x}^i$ consists of the minimum and the maximum of each coordinate over $\mathcal{P}$.

Step 4 will be more troublesome. First we remind the description of Σ_{sym} by Hladík [13].

Theorem 1 (Hladík, 2008)**.** *The symmetric solution set Σ_{sym} is described by the following system of inequalities*

$$A^\Delta |x| + b^\Delta \geq |b^c - A^c x|, \tag{1}$$

$$\sum_{i,j=1}^{n} a_{ij}^\Delta |x_i x_j (p_i - q_j)| + \sum_{i=1}^{n} b_i^\Delta |x_i (p_i + q_i)| \geq \left| \sum_{i=1}^{n} (b_i^c - A_{i,\bullet}^c x) x_i (p_i - q_i) \right| \tag{2}$$

for all vectors $p, q \in \{0,1\}^n \setminus \{0,1\}$ such that

$$p \prec_{\text{lex}} q \text{ and } (p = 1 - q \vee \exists i : p_i = q_i = 0). \tag{3}$$

In order to enclose Σ_{sym} by a polyhedral set we have to linearize the constraints (1)–(2). Linearization of bilinear terms is done by means of Adjiman et al [1], and linearization of absolute value is adopted from Beaumont [10].

Theorem 2 (Adjiman et al., 1998)**.** *For every $x \in \boldsymbol{x} \subset \mathbb{R}$ and $y \in \boldsymbol{y} \subset \mathbb{R}$ we have*

$$\begin{aligned} xy &\leq \overline{x}y + \underline{y}x - \overline{x}\underline{y}, \\ xy &\leq \underline{x}y + \overline{y}x - \underline{x}\overline{y}, \\ xy &\geq \underline{x}y + \underline{y}x - \underline{x}\underline{y}, \\ xy &\geq \overline{x}y + \overline{y}x - \overline{x}\overline{y}. \end{aligned}$$

Theorem 3 (Beaumont, 1998)**.** *For every $x \in \boldsymbol{x} \subset \mathbb{R}$ with $\underline{x} < \overline{x}$ we have*

$$|x| \leq \alpha x + \beta, \tag{4}$$

where

$$\alpha = \frac{|\overline{x}| - |\underline{x}|}{\overline{x} - \underline{x}} \quad \text{and} \quad \beta = \frac{\overline{x}|\underline{x}| - \underline{x}|\overline{x}|}{\overline{x} - \underline{x}}.$$

Moreover, if $\underline{x} \geq 0$ or $\overline{x} \leq 0$ then (4) holds as equation.

Let $\boldsymbol{x}^k$ be an interval enclosure of Σ_{sym}. The first system (1) in the description of Σ_{sym} is linearized by the Beaumont method [10]. Define vectors $\alpha, \beta \in \mathbb{R}^n$ componentwise by

$$\alpha_i := \begin{cases} \frac{|\overline{x}_i^k| - |\underline{x}_i^k|}{\overline{x}_i^k - \underline{x}_i^k} & \text{if } \underline{x}_i^k < \overline{x}_i^k, \\ \operatorname{sgn}(\overline{x}_i^k) & \text{if } \underline{x}_i^k = \overline{x}_i^k, \end{cases}$$

$$\beta_i := \begin{cases} \frac{\overline{x}_i^k |\underline{x}_i^k| - \underline{x}_i^k |\overline{x}|^k}{\overline{x}_i^k - \underline{x}_i^k} & \text{if } \underline{x}_i^k < \overline{x}_i^k, \\ 0 & \text{if } \underline{x}_i^k = \overline{x}_i^k. \end{cases}$$

Then each solution to (1) satisfies the system of $2n$ inequalities [10]

$$\begin{aligned} (A^c - A^\Delta \operatorname{diag}(\alpha))\, x &\leq \overline{b} + A^\Delta \beta, \\ (-A^c - A^\Delta \operatorname{diag}(\alpha))\, x &\leq -\underline{b} + A^\Delta \beta. \end{aligned}$$

Now let turn our attention to the second system (2). Let $p, q \in \{0,1\}^n \setminus \{0,1\}$ satisfying (3), and we linearize the terms of (1) as follows:

1. The second term in (2) is easy to linearize. By Theorem 3, for each summand of the sum we have

$$b_i^\Delta |x_i (p_i + q_i)| \leq b_i^\Delta |(p_i + q_i)| \alpha_i x_i + b_i^\Delta |(p_i + q_i)| \beta_i.$$

2. A linear upper bound for the first term in (2) is obtained by applying both Theorems 2 and 3:

$$\begin{aligned} &|x_i x_j (p_i - q_j)| \\ &\leq |p_i - q_j| (\alpha_i x_i + \beta_i)(\alpha_j x_j + \beta_j) \qquad (5) \\ &= |p_i - q_j| (\alpha_i \alpha_j x_i x_j + \beta_j \alpha_i x_i + \beta_i \alpha_j x_j + \beta_i \beta_j) \\ &\leq \begin{cases} |p_i - q_j| \big(\alpha_i \alpha_j (\overline{x}_i x_j + \underline{x}_j x_i - \overline{x}_i \underline{x}_j) + \\ \qquad + \beta_j \alpha_i x_i + \beta_i \alpha_j x_j + \beta_i \beta_j\big) & \text{if } \alpha_i \alpha_j \geq 0, \\ |p_i - q_j| \big(\alpha_i \alpha_j (\underline{x}_i x_j + \overline{x}_j x_i - \underline{x}_i \overline{x}_j) + \\ \qquad + \beta_j \alpha_i x_i + \beta_i \alpha_j x_j + \beta_i \beta_j\big) & \text{if } \alpha_i \alpha_j \geq 0, \\ |p_i - q_j| \big(\alpha_i \alpha_j (\underline{x}_i x_j + \underline{x}_j x_i - \underline{x}_i \underline{x}_j) + \\ \qquad + \beta_j \alpha_i x_i + \beta_i \alpha_j x_j + \beta_i \beta_j\big) & \text{if } \alpha_i \alpha_j < 0, \\ |p_i - q_j| \big(\alpha_i \alpha_j (\overline{x}_i x_j + \overline{x}_j x_i - \overline{x}_i \overline{x}_j) + \\ \qquad + \beta_j \alpha_i x_i + \beta_i \alpha_j x_j + \beta_i \beta_j\big) & \text{if } \alpha_i \alpha_j < 0. \end{cases} \qquad (6) \end{aligned}$$

In any case, whether $\alpha_i \alpha_j$ is negative or non-negative, we have the choice of two inequalities which to use. We discuss this point later.

3. The inequality (2) has the equivalent form

$$\sum_{i,j=1}^{n} a_{ij}^\Delta |x_i x_j (p_i - q_j)| + \sum_{i=1}^{n} b_i^\Delta |x_i (p_i + q_i)| \geq \sum_{i=1}^{n} (b_i^c - A_{i,\bullet}^c x) x_i (p_i - q_i), \tag{7}$$

$$\sum_{i,j=1}^{n} a_{ij}^\Delta |x_i x_j (p_i - q_j)| + \sum_{i=1}^{n} b_i^\Delta |x_i (p_i + q_i)| \geq -\sum_{i=1}^{n} (b_i^c - A_{i,\bullet}^c x) x_i (p_i - q_i). \tag{8}$$

The right-hand side of the first inequality is linearized in the following way

$$\sum_{i=1}^{n}(b_i^c - A_{i,\bullet}^c x)x_i(p_i - q_i) = \sum_{i=1}^{n} b_i^c x_i(p_i - q_i) - \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}^c x_i x_j (p_i - q_i),$$

where the bilinear term is approximated

$$-a_{i,j}^c(p_i - q_i)x_i x_j \geq \begin{cases} -a_{i,j}^c(p_i - q_i)(\overline{x}_i x_j + \underline{x}_j x_i - \overline{x}_i \underline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) \geq 0, \\ -a_{i,j}^c(p_i - q_i)(\underline{x}_i x_j + \overline{x}_j x_i - \underline{x}_i \overline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) \geq 0, \\ -a_{i,j}^c(p_i - q_i)(\underline{x}_i x_j + \underline{x}_j x_i - \underline{x}_i \underline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) < 0, \\ -a_{i,j}^c(p_i - q_i)(\overline{x}_i x_j + \overline{x}_j x_i - \overline{x}_i \overline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) < 0. \end{cases} \tag{9}$$

For the right-hand side of (8) we proceed accordingly:

$$-\sum_{i=1}^{n}(b_i^c - A_{i,\bullet}^c x)x_i(p_i - q_i) = -\sum_{i=1}^{n} b_i^c x_i(p_i - q_i) + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}^c x_i x_j (p_i - q_i),$$

where

$$a_{i,j}^c(p_i - q_i)x_i x_j \geq \begin{cases} a_{i,j}^c(p_i - q_i)(\underline{x}_i x_j + \underline{x}_j x_i - \underline{x}_i \underline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) \geq 0, \\ a_{i,j}^c(p_i - q_i)(\overline{x}_i x_j + \overline{x}_j x_i - \overline{x}_i \overline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) \geq 0, \\ a_{i,j}^c(p_i - q_i)(\overline{x}_i x_j + \underline{x}_j x_i - \overline{x}_i \underline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) < 0, \\ a_{i,j}^c(p_i - q_i)(\underline{x}_i x_j + \overline{x}_j x_i - \underline{x}_i \overline{x}_j) \\ \qquad \text{if} \quad a_{i,j}^c(p_i - q_i) < 0. \end{cases} \tag{10}$$

In (6) we have two inequalities which we can choose and the same is true for (9) and (10). The choice is independent for $\mathcal{O}(n^2)$ terms. So, if we considered all possible cases then the resulting system would consist of $\mathcal{O}(2^{n^2})$ inequalities. This is a tremendous number, and we had rather proceeded in another way. We chose randomly one of the pair independently one after another, and we did this run twice. Thus, for fixed p and q, we linearized (1) by four inequalities.

It remains to select appropriate binary vectors p and q. Again, we cannot consider all possibilities defined by (3) since it would result in an exponentially large system. In our experience, the following set of choices is efficient; therein, e_k denotes the k-th Cartesian unit vector.

(S1) $p = e_k$ and $q = e_l$, where $k = 1, \ldots, n$, $l = k+1, \ldots, n$;
(S2) $p = e_k$ and $q = 1 - p$, where $k = 1, \ldots, n$;
(S3) make $\frac{1}{4}n^2 + 2n$ random selections of $p, q \in \{0,1\}^n$ with probabilities:

$$P(p_i = 0) = \frac{4}{7},\ P(p_i = 1) = \frac{3}{7},$$
$$P(q_i = 0) = P(q_i = 1) = \frac{1}{2}.$$

We consider more sophisticated selections, too. The interval hull $\boldsymbol{x}^i$ of $\mathcal{P}$ (step 6 of Algorithm 1) is determined by solving $2n$ linear programs. Each of the linear programs yields an extremal point of $\mathcal{P}$ which is on the boundary of $\boldsymbol{x}^i$. For each of such extremal points we construct binary vectors p and q such that inequalities (7)–(8) are violated by as large amount as possible. We discuss two ways: a simple fast heuristic and a more expensive optimal selection.

Let x be any extremal point under discussion. We want to find p and q such that the difference of the left-hand side and right-hand side of the inequality (7) is as small as possible. This inequality takes the form

$$\sum_{i,j=1}^{n} c_{ij}|p_i - q_j| + \sum_{i=1}^{n} d_i p_i + \sum_{i=1}^{n} f_i q_i \geq 0, \tag{11}$$

where

$$c_{ij} = a_{ij}^{\Delta}|x_i x_j|,$$
$$d_i = b_i^{\Delta}|x_i| - b_i^c x_i + \sum_{j=1}^{n} a_{ij}^c x_i x_j,$$
$$f_i = b_i^{\Delta}|x_i| + b_i^c x_i - \sum_{j=1}^{n} a_{ij}^c x_i x_j.$$

The first idea is to choose p and q such that the second and the third sum in (11) is minimal. That is, for $i = 1, \ldots, n$, we put

(S4) $p_1 = 1$ if $d_i < 0$ and $p_i = 0$ otherwise; $q_1 = 1$ if $f_i < 0$ and $q_i = 0$ otherwise.

This simple heuristic doesn't yield optimal solution in general. To compute optimal vectors p and q we have to consider the optimization problem

$$\min \sum_{i,j=1}^{n} c_{ij}|p_i - q_j| + \sum_{i=1}^{n} d_i p_i + \sum_{i=1}^{n} f_i q_i$$
$$\text{subject to} \quad p, q \in \{0,1\}^n.$$

This integer nonlinear programming problem can be reformulated as

$$\min \sum_{i,j=1}^{n} c_{ij} r_{ij} + \sum_{i=1}^{n} d_i p_i + \sum_{i=1}^{n} f_i q_i$$
$$\text{subject to} \quad r_{ij} \geq p_i - q_j,\ r_{ij} \geq -p_i + q_j,\ p_i, q_j, r_{ij} \in \{0,1\}.$$

The constraints force the additional variable r_{ij} to be at least $|p_i - q_j|$. Since we have a minimization problem and $c_{ij} \geq 0$ for all $i, j \in \{1, \ldots, n\}$, the optimal solution always satisfies $r_{ij} = |p_i - q_j|$. Moreover, since the constraint matrix is totally unimodular, we can relax integrality conditions on variables, and obtain a linear program

(S5) $\min \sum_{i,j=1}^{n} c_{ij} r_{ij} + \sum_{i=1}^{n} d_i p_i + \sum_{i=1}^{n} f_i q_i$
subject to $r_{ij} \geq p_i - q_j,\ r_{ij} \geq -p_i + q_j,\ 0 \leq p_i, q_j, r_{ij} \leq 1$.

Even though linear programs are efficiently solvable, we have a quadratic number of variables and calculation of p and q is not very cheap. We discuss performance in detail in Example 3.

In the following section, we test our contractor on a number of examples. We employed selection rules (S1)–(S4), which results in $\binom{n}{2}+n+(\frac{1}{4}n^2+2n)+2n$ instances of p and q. Thus the total number of inequalities that define the polyhedron $\mathcal{P}$ is

$$2n + 4\Big(\binom{n}{2} + n + (\tfrac{1}{4}n^2 + 2n) + 2n\Big) = 3n^2 + 20n.$$

In Example 3 we compare it with a variant incorporating also the selection rule (S5). In this case, we have $2n$ more instances of p and q, and the total number of inequalities is

$$2n + 4\Big(\binom{n}{2} + n + (\tfrac{1}{4}n^2 + 2n) + 2n + 2n\Big) = 3n^2 + 28n.$$

III. Numerical experiments

The computations presented in this section were carried out in MATLAB 7.7.0.471 (R2008b) on a machine with AMD Athlon 64 X2 Dual Core Processor 4400+, CPU 2.2 GHz, with 1004 MB RAM. Interval arithmetics and basic interval functions were provided by the interval toolbox INTLAB v5.3 [26], and some extended interval functions by the Rohn's package VERSOFT 10 [24]. However, for the sake of simplicity, the non-verified floating point arithmetics was used for the real value calculation.

Example 1. Consider an example by Alefeld and Mayer [6]

$$\boldsymbol{A} = \begin{pmatrix} 4 & [-1,1] \\ [-1,1] & 4 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 6 \\ 6 \end{pmatrix}.$$

Let an initial enclosure of Σ_{sym} be $\boldsymbol{x}^0 = ([0,3],\ [0,3])^T$. Calling Algorithm 1 we obtain a sequence of improving enclosures as follows:

$$\begin{aligned}
\boldsymbol{x}^0 &= ([0,3],\ [0,3])^T,\\
\boldsymbol{x}^1 &= ([1.0588, 2.0001],\ [1.0588, 2.0001])^T,\\
\boldsymbol{x}^2 &= ([1.1670, 2.0001],\ [1.1670, 2.0001])^T,\\
\boldsymbol{x}^3 &= ([1.1945, 2.0000],\ [1.1945, 2.0000])^T,\\
\boldsymbol{x}^4 &= ([1.1991, 2.0000],\ [1.1991, 2.0000])^T.
\end{aligned}$$

In four iterations we have a tight enclosure $([1.1991, 2.0000], [1.1991, 2.0000])^T$ of Σ_{sym}. Note that the real interval hull of Σ_{sym} is $([\frac{6}{5}, 2], [\frac{6}{5}, 2])^T$ and the interval hull of Σ is $([\frac{18}{17}, 2], [\frac{18}{17}, 2])^T$. According to [6], Cholesky method yields $([1,2], [\frac{9}{8}, 2])^T$ and Gaussian elimination yields $([1,2], [\frac{18}{17}, 2])^T$. Thus we improved both enclosures.

Example 2. Consider the Behnke example from Rump [25, pg. 48]

$$\boldsymbol{A} = \begin{pmatrix} 3 & [1,2] \\ [1,2] & 3 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} [10, 10.5] \\ [10, 10.5] \end{pmatrix}$$

Let an initial enclosure of Σ_{sym} be $\boldsymbol{x}^0 = ([0.8, 4],\ [0.8, 4])^T$. Calling Algorithm 1 we obtain the following sequence of improving enclosures:

$$\begin{aligned}
\boldsymbol{x}^0 &= ([0.8, 4],\ [0.8, 4])^T,\\
\boldsymbol{x}^1 &= ([1.2857, 3.0715],\ [1.2857, 3.0715])^T,\\
\boldsymbol{x}^2 &= ([1.5496, 2.8411],\ [1.5496, 2.8411])^T,\\
\boldsymbol{x}^3 &= ([1.6722, 2.7589],\ [1.6722, 2.7589])^T,\\
\boldsymbol{x}^4 &= ([1.7192, 2.7348],\ [1.7192, 2.7348])^T,\\
\boldsymbol{x}^5 &= ([1.7350, 2.7280],\ [1.7350, 2.7280])^T,\\
\boldsymbol{x}^6 &= ([1.7401, 2.7260],\ [1.7401, 2.7260])^T.
\end{aligned}$$

In six iterations we have a tight enclosure $([1.7401, 2.7260], [1.7401, 2.7260])^T$ of Σ_{sym}. Compare it with the interval hull of Σ_{sym}, which is $([1.8, 2.6875], [1.8100, 2.688])^T$ (and not $([1.8100, 2.688], [1.8100, 2.688])^T$ stated in [25]). The interval hull of Σ is $([\frac{9}{7}, \frac{43}{14}], [\frac{9}{7}, \frac{43}{14}])^T \simeq ([1.285, 3.072], [1.285, 3.072])^T$. The Rump inclusion method gives $([1.623, 2.932], [1.623, 2.932])^T$.

Example 3. We carried out a number of randomly generated examples. They are summarized in Table I and II. The dimension is denoted by n. The random interval matrices were generated in the following way. First, entries of A^c were chosen randomly and independently in $[-10, 10]$ with uniform distribution, and then we set $A^c := A^c + (A^c)^T + 20nI$, where I stands for the identity matrix. Thus $\boldsymbol{A}$ is symmetric and more or less diagonally dominant, which forces $\boldsymbol{A}$ to consist of nonsingular matrices. Entries of the radius matrix A^Δ were generated randomly independently and uniformly in $[-R, R]$, where $R > 0$ was a parameter. Again, we symmetrize $A^\Delta := A^\Delta + (A^\Delta)^T$. Similarly we proceeded in case of the right-hand side interval vector $\boldsymbol{b}$. Entries of the midpoint vector b^c came randomly from $[-10n, 10n]$ and entries of the radius vector from $[0, R]$.

In each setting of n and R we carried out a sequence of runs the number of which is denoted by "iterations". Based on a sequence of runs we determined the average computing time and cut off quotients. Cut off quotient is defined naturally as a fraction of the volume that is cut off, i.e.

$$\frac{\prod_{i=1}^{n} (x_i^0)^\Delta - \prod_{i=1}^{n} (x_i^*)^\Delta}{\prod_{i=1}^{n} (x_i^0)^\Delta} 100\%,$$

where $\boldsymbol{x}^0$ is an initial interval enclosure of Σ_{sym} and $\boldsymbol{x}^*$ is the resulting enclosure returned by our algorithm.

The initial solutions were obtained by calling the functions `verifylss` and `verintervalhull`. The former is from the INTLAB toolbox and computes fast interval enclosure of

Σ, while the latter VERSOFT function calculates the verified interval hull of Σ.

Table I contains results for the variant of the contractor that incorporates selection rules (S1)–(S4), while Table II employs the set of selection rules (S1)–(S5).

The last column in Table I reveals that the resulting interval enclosures of Σ_{sym} are strictly inside the interval hull of Σ. It means, our results are significantly better than an arbitrarily accurate solver for Σ. This also certifies that dependencies in the description of Σ_{sym} must be taken into account, otherwise the solution is heavily overestimated.

Table I shows that the computation is rather slow. Nevertheless, it is tractable for quite high dimensions: Providing we limit the number of iterations of the main loop of Algorithm 1 (steps 3–7), and we use an efficient linear programming solver then our method runs in polynomial time.

IV. Conclusion

We presented a method for contracting an interval enclosure of the symmetric solution set. It was based on a linear relaxation. Tight approximation would lead to exponentially large system of inequalities, so, in order to be our approach tractable, we selected the most promising ones. This selection was based on numerical experience as well as theoretical analysis. However, another way of selection may possibly improve the contractor efficiency.

Even though the computational cost of the proposed algorithm is rather high, its complexity is still polynomial. Hence we can solve much larger systems than the traditional branch & bound approach was able to deal with.

References

[1] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs – I. Theoretical advances. *Comput. Chem. Eng.*, 22(9):1137–1158, 1998.

[2] G. Alefeld and J. Herzberger. *Introduction to interval computations*. Computer Science and Applied Mathematics. Academic Press, New York, 1983.

[3] G. Alefeld, V. Kreinovich, and G. Mayer. On the shape of the symmetric, persymmetric, and skew-symmetric solution set. *SIAM J. Matrix Anal. Appl.*, 18(3):693–705, 1997.

[4] G. Alefeld, V. Kreinovich, and G. Mayer. The shape of the solution set for systems of interval linear equations with dependent coefficients. *Math. Nachr.*, 192:23–36, 1998.

[5] G. Alefeld, V. Kreinovich, and G. Mayer. On the solution sets of particular classes of linear interval systems. *J. Comput. Appl. Math.*, 152(1-2):1–15, 2003.

[6] G. Alefeld and G. Mayer. The cholesky method for interval data. *Linear Algebra Appl.*, 194:161–182, 1993.

[7] G. Alefeld and G. Mayer. On the symmetric and unsymmetric solution set of interval systems. *SIAM J. Matrix Anal. Appl.*, 16(4):1223–1240, 1995.

[8] G. Alefeld and G. Mayer. New criteria for the feasibility of the cholesky method with interval data. *SIAM J. Matrix Anal. Appl.*, 30(4):1392–1405, 2008.

[9] R. Araiza, G. Xiang, O. Kosheleva, and D. Škulj. Under interval and fuzzy uncertainty, symmetric markov chains are more difficult to predict. In M. Reformat and M. R. Berthold, editors, *Proceedings of the 26th International Conference of the North American Fuzzy Information Processing Society NAFIPS'2007*, pages 526–531, San Diego, California, 2007.

[10] O. Beaumont. Solving interval linear systems with linear programming techniques. *Linear Algebra Appl.*, 281(1-3):293–309, 1998.

[11] A. Dreyer. Interval analysis of linear analog circuits. In *Proceedings of the 12th GAMM–IMACS Symposium on Scientific Computing, Computer Artithmetic and Validated Numerics, SCAN 2006.*, page 14, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[12] M. Fiedler, J. Nedoma, J. Ramík, J. Rohn, and K. Zimmermann. *Linear optimization problems with inexact data*. Springer, New York, 2006.

[13] M. Hladík. Description of symmetric and skew-symmetric solution set. *SIAM J. Matrix Anal. Appl.*, 30(2):509–521, 2008.

[14] M. Hladík, D. Daney, and E. Tsigaridas. An algorithm for the real interval eigenvalue problem. Research Report RR-6680, INRIA, France, October 2008. http://hal.inria.fr/inria-00329714/en/.

[15] C. Jansson. Interval linear systems with symmetric matrices, skew-symmetric matrices and dependencies in the right hand side. *Comput.*, 46(3):265–274, 1991.

[16] L. V. Kolev. An improved interval linearization for solving nonlinear problems. *Numer. Algorithms*, 37(1-4):213–224, 2004.

[17] L. V. Kolev. Improvement of a direct method for outer solution of linear parametric systems. *Reliab. Comput.*, 12(3):193–202, 2006.

[18] Z. Kulpa, A. Pownuk, and I. Skalna. Analysis of linear mechanical structures with uncertainties by means of interval methods. *Comput. Assist. Mech. Eng. Sci.*, 5(4):443–477, 1998.

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

[20] E. D. Popova. Parametric interval linear solver. *Numer. Algorithms*, 37(1-4):345–356, 2004.

[21] E. D. Popova. Solving linear systems whose input data are rational functions of interval parameters. *Lecture Notes in Computer Science*, 4310:345–352, 2007.

[22] E. D. Popova and W. Krämer. Inner and outer bounds for the solution set of parametric linear systems. *J. Comput. Appl. Math.*, 199(2):310–316, 2007.

[23] J. Rohn. Systems of linear interval equations. *Linear Algebra Appl.*, 126(C):39–78, 1989.

[24] J. Rohn. VERSOFT: Verification software in MATLAB / INTLAB, version 10, 2009. http://uivtx.cs.cas.cz/rohn/matlab/.

[25] S. M. Rump. Verification methods for dense and sparse systems of equations. In J. Herzberger, editor, *Topics in Validated Computations*, Studies in Computational Mathematics, pages 63–136, Amsterdam, 1994. Elsevier. Proceedings of the IMACS-GAMM International Workshop on Validated Computations, University of Oldenburg.

[26] S. M. Rump. Intlab – interval laboratory, the matlab toolbox for verified computations, version 5.3, 2006. http://www.ti3.tu-harburg.de/rump/intlab/.

[27] U. Schäfer. Two ways to extend the Cholesky decomposition to block matrices with interval entries. *Reliab. Comput.*, 8(1):1–20, 2002.

[28] U. Schäfer. Aspects for a block version of the interval Cholesky algorithm. *J. Comput. Appl. Math.*, 152(1-2):481–491, 2003.

[29] C.-J. R. Shi and M. W. Tian. Simulation and sensitivity of linear analog circuits under parameter variations by robust interval analysis. *ACM Transactions on Design Automation of Electronic Systems*, 4(3):280–312, 1999.

TABLE I
AVERAGE COMPUTING TIME AND CUT OFF QUOTIENTS FOR RANDOM RUNS AND SELECTION RULES (S1)–(S4)

n	R	iterations	average time	`verifylss` cut off	`verintervalhull` cut off
5	0.1	100	5.13 s	21.8 %	18.8 %
5	0.5	100	5.52 s	29.5 %	15.6 %
5	1	100	5.71 s	38.0 %	13.1 %
10	0.1	100	56.3 s	36.1 %	31.3 %
10	0.5	100	54.5 s	47.5 %	25.5 %
10	1	100	55.4 s	57.8 %	19.4 %
15	0.1	100	218 s	43.6 %	37.1 %
15	0.5	100	222 s	59.6 %	31.5 %
15	1	100	211 s	72.2 %	23.9 %
20	0.1	50	604 s	51.7 %	44.1 %
20	0.5	50	600 s	68.3 %	36.3 %
20	1	50	573 s	80.9 %	26.5 %
25	0.1	50	1318 s	57.7 %	49.3 %
25	0.5	50	1312 s	75.3 %	41.0 %
25	1	50	1250 s	86.9 %	30.8 %

TABLE II
AVERAGE COMPUTING TIME AND CUT OFF QUOTIENTS FOR RANDOM RUNS AND SELECTION RULES (S1)–(S5)

n	R	iterations	average time	`verifylss` cut off	`verintervalhull` cut off
5	0.1	100	6.47 s	22.1 %	19.1 %
5	0.5	100	6.29 s	30.0 %	16.5 %
5	1	100	6.27 s	40.0 %	15.0 %
10	0.1	100	63.9 s	36.5 %	31.6 %
10	0.5	100	64.1 s	47.5 %	25.5 %
10	1	100	60.6 s	59.2 %	20.2 %
15	0.1	100	296 s	44.9 %	38.5 %
15	0.5	100	295 s	59.8 %	31.9 %
15	1	100	270 s	72.4 %	24.0 %
20	0.1	50	1025 s	52.1 %	44.5 %
20	0.5	50	1085 s	68.8 %	37.2 %
20	1	50	1045 s	81.0 %	27.6 %
25	0.1	50	3385 s	58.0 %	49.5 %
25	0.5	50	3430 s	75.7 %	41.5 %
25	1	50	3238 s	87.2 %	31.3 %