Journal of Computational and Applied Mathematics 235 (2011) 2715–2730

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An algorithm for addressing the real interval eigenvalue problem

Milan Hladík[a,b,*], David Daney[b], Elias P. Tsigaridas[b]

[a] Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, Malostranské nám. 25, 118 00, Prague, Czech Republic

[b] INRIA Sophia-Antipolis Méditerranée, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France

ARTICLE INFO

Article history:
Received 20 October 2008
Received in revised form 28 June 2010

MSC:
65G40
65F15
15A18

Keywords:
Interval matrix
Real eigenvalue
Eigenvalue bounds
Regularity
Interval analysis

ABSTRACT

In this paper we present an algorithm for approximating the range of the real eigenvalues of interval matrices. Such matrices could be used to model real-life problems, where data sets suffer from bounded variations such as uncertainties (e.g. tolerances on parameters, measurement errors), or to study problems for given states.

The algorithm that we propose is a subdivision algorithm that exploits sophisticated techniques from interval analysis. The quality of the computed approximation and the running time of the algorithm depend on a given input accuracy. We also present an efficient C++ implementation and illustrate its efficiency on various data sets. In most of the cases we manage to compute efficiently the exact boundary points (limited by floating point representation).

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Computation of real eigenvalues is a ubiquitous operation in applied mathematics, not only because it is an important mathematical problem, but also due to the fact that such computations lie at the core of almost all engineering problems. However, in these problems, which are real-life problems, precise data are very rare, since the input data are influenced by diverse uncertainties.

We study these problems through models that reflect the real-life situations as well as possible. A modern approach is to consider that the parameters to be defined are not exact values, but a set of possible values. The nature of these variations is not physically homogeneous, mainly due to measurement uncertainties, or due to tolerances that come from fabrication and identification, or simply because we want to study the system in a set of continuous states.

Contrary to adopting a statistical approach, which, we have to note, is not always possible, it may be more simple or realistic to bound the variations of the parameters by intervals. Interval analysis turns out to be a very powerful technique for studying the variations of a system and for understanding its properties. One of the most important properties of this approach is the fact that it is possible to certify the results of all the states of a system.

Such an approach motivates us to look for an algorithm that computes rigorous bounds on eigenvalues of an interval matrix. Interval-based problems have been studied intensively in the past few decades, for example in control in order to analyse the stability of interval matrices [1]. The interval eigenvalue problem, in particular, also has a variety of applications throughout diverse fields of science. Let us mention automobile suspension systems [2], vibrating systems [3], principal component analysis [4], and robotics [5], for instance.

* Corresponding author at: Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, Malostranské nám. 25, 118 00, Prague, Czech Republic.

E-mail addresses: milan.hladik@matfyz.cz (M. Hladík), david.daney@sophia.inria.fr (D. Daney), elias.tsigaridas@sophia.inria.fr (E.P. Tsigaridas).

0377-0427/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2010.11.022

1.1. Motivation

As a motivating example, let us mention the following problem from robotics, that usually appears in experimental planning, e.g. [6]. We consider the following simple robotic mechanism. Let $X = (x, y)$ be a point in the plane, which is linked to two points, $M = (a, b)$ and $N = (c, d)$, using two prismatic joints, r_1 and r_2 respectively. In this case, the end-effector C has two degrees of freedom for moving in the plane. The joints r_1 and r_2 are measured in a range $[\min\{r_k\}, \max\{r_k\}]$, where $k = 1, 2$. The range of the joints is obtained due to mechanical constraints and describes the workspace of the mechanism.

If we are given r_1 and r_2 and we want to estimate the coordinates of X, then we solve the polynomial system $F_1 = F_2 = 0$, where $F_1 = |X - S|^2 - r_1^2$ and $F_2 = |X - T|^2 - r_2^2$, which describes the kinematics problem. For the calibration problem things are quite different [7,8]. In this case we want to compute, or estimate, the coordinates M and N as a function of several measurements of X, that is $X_1 = (x_1, y_1), X_2 = (x_2, y_2), X_3 = (x_3, y_3), \ldots$. This is so because M and N are not known exactly, due to manufacturing tolerances. We have four unknowns, a, b, c and d, expressed as a function of the measurements X_i, where $1 \leq i \leq n$. If $n \geq 2$, then we can compute a, b, c, and d using the classical approach of the least squares method. However, we have to take into account the noise in the measurements $l_{1,i}$ and $l_{2,i}$. To get a robust solution, we choose the position of the measurements by also selecting the values of $l_{1,i}, l_{2,i}$ in $[\min\{l_k\}, \max\{l_k\}]$, where $k = 1, 2$.

We estimate the several criteria of selection using the eigenvalues of the observability matrix [8], that is the eigenvalues of $J^T J$, where elements of J are partial derivatives of F_k with respect to kinematic parameters. Such an approach requires bounds on the eigenvalues of the observability matrix, which is what we propose in this paper. We present a detailed example in Example 5.

Further motivation comes from polynomial system real solving. Consider a system of polynomials in $\mathbb{R}[x_1, \ldots, x_n]$ and let I be the ideal that they define. The coordinates of the solutions of the system can be obtained as eigenvalues of the so called multiplication tables, e.g. [9]. That is for each variable x_i we can construct (using Gröbner basis or normal form algorithms) a matrix M_{x_i} that corresponds to the operator of multiplication by x_i in the quotient algebra $\mathbb{R}[x_1, \ldots, x_n]/I$. The eigenvalues of these matrices are the coordinates of the solutions; thus the real eigenvalues are the coordinates of the real solutions. If the coefficients of the polynomials are not known exactly, then we can consider the multiplications as interval matrices. For an algorithm for solving bivariate polynomial systems that is based on the eigenvalues and eigenvectors of the Bézoutian matrix, the reader may refer to [10]. For the great importance of eigenvalue computations in polynomial systems solving with inexact coefficients we refer the reader to [11].

1.2. Notation and preliminaries

In what follows we will use the following notation:

$\operatorname{sgn}(r)$	The sign of a real number r, i.e., $\operatorname{sgn}(r) = 1$ if $r \geq 0$, and $\operatorname{sgn}(r) = -1$ if $r < 0$
$\operatorname{sgn}(z)$	The sign of a vector z, i.e., $\operatorname{sgn}(z) = (\operatorname{sgn}(z_1), \ldots, \operatorname{sgn}(z_n))^T$
e	A vector of all ones (with convenient dimension)
$\operatorname{diag}(z)$	The diagonal matrix with entries $z_1, \ldots, z_n$
$\rho(A)$	The spectral radius of a matrix A
$A_{\bullet,i}$	The ith column of a matrix A
$\partial \mathcal{S}$	The boundary of a set $\mathcal{S}$
$\lvert \mathcal{S} \rvert$	The cardinality of a set $\mathcal{S}$

For basic interval arithmetic the reader may refer to e.g. [12–14]. A square interval matrix is defined as

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

where $\underline{A}, \overline{A} \in \mathbb{R}^{n\times n}$ and $\underline{A} \leq \overline{A}$ are given matrices and the inequalities are considered elementwise. By

$$A_c \equiv \frac{1}{2}(\underline{A} + \overline{A}), \qquad A_\Delta \equiv \frac{1}{2}(\overline{A} - \underline{A}),$$

we denote the midpoint and radius of $\boldsymbol{A}$, respectively. We use analogous notation for interval vectors. An interval linear system of equations

$$\boldsymbol{A}x = \boldsymbol{b},$$

is a short form for a set of systems

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

The set of all real eigenvalues of $\boldsymbol{A}$ is defined as

$$\Lambda := \{\lambda \in \mathbb{R}; Ax = \lambda x, x \neq 0, A \in \boldsymbol{A}\},$$

and is compact set. It seems that Λ is always composed of at most n compact real intervals, but this conjecture has not been proven yet and is proposed as an open problem.

In general, computing Λ is a difficult problem. Even checking whether $0 \in \Lambda$ is an NP-hard problem, since the problem is equivalent to checking regularity of the interval matrix $\boldsymbol{A}$, which is known to be NP-hard [15]. An *inner approximation* of Λ is any subset of Λ, and an *outer approximation* of Λ is a set containing Λ as a subset.

1.3. Previous work and our contribution

The problem of computing (the intervals of) the eigenvalues of interval matrices has been studied since the nineties. The first results were due to Deif [16] and Rohn and Deif [17]. They proposed formulae for calculating exact bounds; the former case bounds real and imaginary parts for complex eigenvalues, while the latter case bounds the real eigenvalues. However, these results apply only under certain assumptions on the sign pattern invariance of the corresponding eigenvectors; such assumptions are not easy to verify (cf. [18]). Other works by Rohn concern theorems for the real eigenvalues [19] and bounds of the eigenvalue set Λ [20]. An approximate method was given in [2]. The related topic of finding verified intervals of eigenvalues for real matrices is studied in [21].

If $\boldsymbol{A}$ has a special structure, then it is possible to develop stronger results, that is to compute tighter intervals for the eigenvalue set. This is particularly true when $\boldsymbol{A}$ is symmetric; we postpone this discussion to a forthcoming communication. Our aim is to consider the general case, and to propose an algorithm for the eigenvalue problem, when the input is a generic interval matrix, without any special property.

Several methods are known for computing the eigenvalues of scalar (non-interval) matrices. It is not possible to directly apply them to interval matrices, since this causes enormous overestimation of the computed eigenvalue intervals. For the same reason, algorithms that are based on the characteristic polynomial of $\boldsymbol{A}$ are rarely, if at all, used. Even though interval-valued polynomials can be handled efficiently [22], this approach cannot yield sharp bounds, due to the overestimation of the intervals that correspond to the coefficients of the characteristic polynomial.

A natural way of computing the set of the eigenvalue intervals Λ, is to try to solve directly the interval nonlinear system

$$Ax = \lambda x, \quad \|x\| = 1,\ A \in \boldsymbol{A},\ \lambda \in \boldsymbol{\lambda}^0, \tag{1}$$

where $\boldsymbol{\lambda}^0 \supseteq \Lambda$ is some initial outer estimation of the eigenvalue set, and $\|\cdot\|$ is any vector norm. Interval analysis techniques for solving nonlinear systems of equations with interval parameters are very developed nowadays [23,14]. Using filtering, diverse consistency checking, and sophisticated box splitting they achieve excellent results. However, the curse of dimensionality implies that these techniques are applicable only to problems of relative small size. Recall that the curse of dimensionality refers to the exponential increase of the volume, when additional dimensions are added to a problem. For the eigenvalue problem (1), this is particularly the case (cf. Section 4).

We present an efficient algorithm for approximating the set of intervals of the real eigenvalues of a (generic) interval matrix, Λ, within a given accuracy. Our approach is based on a branch and prune scheme. We use several interval analysis techniques to provide efficient tests for inner and outer approximations of the intervals in Λ.

The rest of the paper is structured as follows. In Section 2 we present the main algorithm, the performance of which depends on checking intervals for being outer (containing no eigenvalue) or inner (containing only eigenvalues). These tests are discussed in Sections 2.3 and 2.4, respectively. Using some known theoretical assertions we can achieve in most cases the exact bounds of the eigenvalue set. This is considered in Section 3. In Section 4 we present an efficient implementation of the algorithm and experiments on various data sets.

2. The general algorithm

The algorithm that we present is a subdivision algorithm, based on a branch and prune method [23]. The pseudo-code of the algorithm is presented in Algorithm 1. The input consists of an interval matrix $\boldsymbol{A}$ and a precision rate $\varepsilon > 0$. Notice that ε is not a direct measure of the approximation accuracy.

The output of the algorithm consists of two lists of intervals: L_{inn} which comprises intervals lying inside Λ, and L_{unc} which consists of intervals where we cannot decide whether they are contained in Λ or not, with the given required precision ε. The union of these two lists is an outer approximation of Λ.

The idea behind our approach is to subdivide a given interval that initially contains Λ until either we can certify that an interval is an inner or an outer one, or its length is smaller than the input precision ε. In the latter case, the interval is placed in the list L_{unc}.

The (practical) performance of the algorithm depends on the efficiency of its subroutines and more specifically on the subroutines that implement the inner and outer tests. This is discussed in detail in Sections 2.3 and 2.4.

2.1. Branching in detail

We may consider the process of the Algorithm 1 as a binary tree in which the root corresponds to the initial interval that contains Λ. At each step of the algorithm the inner and outer tests are applied to the tested interval. If both are unsuccessful and the length of the interval is greater than ε, then we split the interval into two equal intervals and the algorithm is applied to each of them.

There are two basic ways to traverse this binary tree, either depth-first or breadth-first. Even though from a theoretical point of view the two ways are equivalent, this is not the case from a practical point of view. The actual running time of an implementation of Algorithm 1 depends closely on the way that we traverse the binary tree. This is of no surprise. Exactly the same behavior is noticed in the problem of real root isolation of integer polynomials [24–26].

Algorithm 1 (Approximation of Λ)

```
1: compute initial bounds $\underline{\lambda}^0, \overline{\lambda}^0$ such that $\Lambda \subseteq \boldsymbol{\lambda}^{\mathbf{0}} := [\underline{\lambda}^0, \overline{\lambda}^0]$;
2: $L := \{\boldsymbol{\lambda}^{\mathbf{0}}\}, L_{\text{inn}} := \emptyset, L_{\text{unc}} := \emptyset$;
3: while $L \neq \emptyset$ do
4:   choose and remove some $\boldsymbol{\lambda}$ from $L$;
5:   if $\boldsymbol{\lambda} \cap \Lambda = \emptyset$ then
6:     {nothing};
7:   else if $\boldsymbol{\lambda} \subseteq \Lambda$ then
8:     $L_{\text{inn}} := L_{\text{inn}} \cup \{\lambda\}$;
9:   else if $\lambda_\Delta < \varepsilon$ then
10:    $L_{\text{unc}} := L_{\text{unc}} \cup \{\lambda\}$;
11:  else
12:    $\boldsymbol{\lambda}^1 := [\underline{\lambda}, \lambda_c], \boldsymbol{\lambda}^2 := [\lambda_c, \overline{\lambda}], L := L \cup \{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2\}$;
13:  end if
14: end while
15: return $L_{\text{inn}}$ and $L_{\text{unc}}$;
```

A closely related issue is the data structure that we use to implement the various lists of the algorithm and in particular L. Our experience suggests that we should implement L as a stack, so that the last inserted element to be chosen at step 4 is the next candidate interval $\boldsymbol{\lambda}$. Hereby, at step 12 we insert $\boldsymbol{\lambda}^2$ first, and $\boldsymbol{\lambda}^1$ afterwards.

Note that, in essence, the stack implementation of L closely relates to the depth-first search algorithm for traversing a binary tree. In this case, nodes correspond to intervals handled. Each node is a leaf if it is recognized as an outer or inner interval, or if it is small enough. Otherwise, it has two descendants: the left one is for the left part of the interval and the right one is for the right part.

The main advantage of the depth-first exploration of the tree, and consequently of the choice to use a stack to implement L, in stack implementation, is that it allows us to exhibit some useful properties of the tested intervals. For example, if a tested interval $\boldsymbol{\lambda}$ is an inner interval, then the next interval in the stack, which is adjacent to it, cannot be an outer interval. Thus, for this interval we can omit steps 5–6 of the algorithm. Similarly, when a tested interval is an outer interval, then the next in the stack cannot be inner. These kinds of properties allow us to avoid many needless computations in a lot of cases, and turn out to be very efficient in practice.

Another important consequence of the choice of traversing the tree depth-first is that it allows us to improve the time complexity of the inner tests. This is discussed in Section 2.4.

2.2. Initial bounds

During the first step of Algorithm 1 we compute an initial outer approximation of the eigenvalue set Λ, i.e. an interval that is guaranteed to contain the eigenvalue set. For this computation we use a method proposed in [20, Theorem 2]:

Theorem 1. *Let*

$$S_c := \frac{1}{2}\left(A_c + A_c^T\right),$$

$$S_\Delta := \frac{1}{2}\left(A_\Delta + A_\Delta^T\right).$$

Then $\Lambda \subseteq \boldsymbol{\lambda}^0 := [\underline{\lambda}^0, \overline{\lambda}^0]$, *where*

$$\underline{\lambda}^0 = \lambda_{\min}(S_c) - \rho(S_\Delta),$$

$$\overline{\lambda}^0 = \lambda_{\max}(S_c) + \rho(S_\Delta),$$

and $\lambda_{\min}(S_c), \lambda_{\max}(S_c)$ *denote the minimal and maximal eigenvalue of* S_c, *respectively.*

The aforementioned bounds are usually very tight, especially for symmetric interval matrices. Moreover, it turns out, as we will discuss in Section 4, that $\boldsymbol{\lambda}^0$ is an excellent starting point for our subdivision algorithm. Other bounds can be developed if we use Gerschgorin discs or Cassini ovals. None of these bounds, however, provide in practice approximations as sharp as the ones of Theorem 1.

2.3. The outer test

In this section, we propose several outer tests, which can be used in step 5 of Algorithm 1. Even though their theoretical (worst-case) complexities are the same, their performances in practice differ substantially.

Consider an interval matrix $\boldsymbol{A}$ and a real closed interval $\boldsymbol{\lambda}$. We want to decide whether $\boldsymbol{\lambda} \cap \Lambda = \emptyset$, that is, there is no matrix $A \in \boldsymbol{A}$ that has a real eigenvalue inside $\boldsymbol{\lambda}$. In this case, we say that $\boldsymbol{\lambda}$ is an *outer interval.*

The natural idea is to transform the problem to the problem of checking regularity of interval matrices. An interval matrix $\boldsymbol{M}$ is *regular* if every $M \in \boldsymbol{M}$ is nonsingular.

Proposition 1. *If the interval matrix* $\boldsymbol{A} - \boldsymbol{\lambda} I$ *is regular, then* $\boldsymbol{\lambda}$ *is an outer interval.*

Proof. Let $\lambda \in \boldsymbol{\lambda}$ and $A \in \boldsymbol{A}$. The real number λ is not an eigenvalue of A if and only if the matrix $A - \lambda I$ is nonsingular. Thus, if $\boldsymbol{A} - \boldsymbol{\lambda} I$ is regular, then for every $\lambda \in \boldsymbol{\lambda}$ and $A \in \boldsymbol{A}$ we have that $A - \lambda I$ is nonsingular (not conversely), and hence λ is not an eigenvalue. □

In general, Proposition 1 gives a sufficient but not necessary condition for checking the outer property (due to the dependences caused by multiple appearances of $\boldsymbol{\lambda}$). Nevertheless, the smaller the radius of $\boldsymbol{\lambda}$, the stronger the condition.

We now review some of the applicable conditions and methods. Recall that testing regularity of an interval matrix is an NP-hard problem [15]; therefore we exhibit a sufficient condition as well.

2.3.1. A sufficient regularity condition

There are diverse sufficient conditions for an interval matrix to be regular [27]. The very strong one, which turned out to very useful (cf. Section 4), is formulated below.

Proposition 2. *An interval matrix* $\boldsymbol{M}$ *is regular if* M_c *is nonsingular and* $\rho(|M_c^{-1}|M_\Delta) < 1$.

Proof. See e.g. [27, Corollary 3.2.]. □

2.3.2. The Jansson and Rohn method

Herein we recall the Jansson and Rohn method [28] for testing regularity of an interval matrix $\boldsymbol{M}$. Its great benefit is that the time complexity is not a priori exponential. Its modification is also is very useful for the inner test (Section 2.4). That is why we describe the method here in more detail.

Choose an arbitrary vector $b \in \mathbb{R}^n$ and consider the interval system of equations $\boldsymbol{M}x = b$. The solution set

$$X = \{x \in \mathbb{R}^n;\, Mx = b,\, M \in \boldsymbol{M}\}$$

is described by

$$|M_c x - b| \le M_\Delta |x|.$$

This solution set is formed by a union of convex polyhedra, since a restriction of X on an orthant is characterized by a linear system of inequalities

$$(M_c - M_\Delta \operatorname{diag}(z))x \le b, \qquad (M_c + M_\Delta \operatorname{diag}(z))x \ge b, \quad \operatorname{diag}(z)x \ge 0, \tag{2}$$

where $z \in \{\pm 1\}^n$ is a vector of signs corresponding to the orthant.

Regularity of $\boldsymbol{M}$ closely relates to unboundedness of X. Indeed, Jansson and Rohn [28] obtained the following result.

Theorem 2. *Let* C *be a component (maximal connected set) of* X*. Then* $\boldsymbol{M}$ *is regular if and only if* C *is bounded.*

The algorithm starts by selecting an appropriate vector b. The component C is chosen so that it includes the point $M_c^{-1}b$. We check the unboundedness of C by checking the unboundedness of (2), for each orthant that C intersects. The list L comprises the sign vectors (orthants) to be inspected, and V consists of the already visited orthants.

To speed up the process, we notice that there is no need to inspect all the neighboring orthants. It suffices to inspect only that orthants possibly connected to the actual one. Thus we can skip the ones that are certainly disconnected. Jansson and Rohn proposed an improvement in this way; we refer the reader to [28] for more details.

The performance of Algorithm 2 depends strongly on the choice of b. It is convenient to select b so that the solution set X intersects a (possibly) small number of orthants. The selection procedure of b, proposed by Jansson and Rohn, consists of a local search.

2.3.3. The ILS method

The ILS (interval linear system) method is a simple but efficient approach for testing regularity of an interval matrix $\boldsymbol{M}$. It is based on transforming the problem to solving an interval linear system and using an ILS solver. The more effective the ILS solver is, the more effective the ILS method.

Proposition 3. *The interval matrix* $\boldsymbol{M}$ *is regular if and only if the interval linear system*

$$\boldsymbol{M}x = 0, \quad x \ne 0 \tag{5}$$

has no solution.

Algorithm 2 (Jansson and Rohn method checking regularity of $\boldsymbol{M}$)

```
1: if $M_c$ is singular then
2:    return "$\boldsymbol{M}$ is not regular";
3: end if
4: select $b$;
5: $z := \operatorname{sgn}(A_c^{-1}b)$;
6: $L := \{z\}, V := \emptyset$;
7: while $L \neq \emptyset$ do
8:    choose and remove some $z$ from $L$;
9:    $V := V \cup \{z\}$;
10:   solve the linear program
          $\max \left\{z^T x;\ (M_c - M_\Delta \operatorname{diag}(z))x \leq b,\ (M_c + M_\Delta \operatorname{diag}(z))x \geq b,\ \operatorname{diag}(z)x \geq 0\right\};$   (3)
11:   if (3) is unbounded then
12:      return "$\boldsymbol{M}$ is not regular";
13:   else if (3) is feasible then
14:      $L := L \cup \left(N(z) \setminus (L \cup V)\right)\}$, where
          $N(z) = \{(z_1, \dots, z_{i-1}, -z_i, z_{i+1}, \dots, z_n)^T;\ 1 \leq i \leq n,\ \};$   (4)
15:   end if
16: end while
17: return "$\boldsymbol{M}$ is regular";
```

Algorithm 3 (ILS method)

```
1: for $i = 1, \dots, n$ do
2:    $\boldsymbol{b} := \boldsymbol{M}_{\bullet,i}$ {the $i$th column of $\boldsymbol{M}$};
3:    $\boldsymbol{M}' := \left(\boldsymbol{M}_{\bullet,1}, \dots, \boldsymbol{M}_{\bullet,i-1}, \boldsymbol{M}_{\bullet,i+1}, \dots, \boldsymbol{M}_{\bullet,n}\right)$ {the matrix $\boldsymbol{M}$ without the $i$th column};
4:    solve (approximately) the interval linear system
          $\boldsymbol{M}'x' = -\boldsymbol{b},\ -e \leq x' \leq e;$   (7)
5:    if (7) has possibly a solution then
6:       return "$\boldsymbol{\lambda}$ needn't be outer";
7:    end if
8: end for
9: return "$\boldsymbol{\lambda}$ is an outer interval";
```

As x can be normalized, we replace the inequation $x \neq 0$ by $\|x\|_\infty = 1$, where the maximum norm is defined as $\|x\|_\infty := \max\{|x|_i;\ i = 1, \dots, n\}$. Moreover, since both x and $-x$ satisfy (5), we derive the following equivalent formulation of (5):

$$\boldsymbol{M}x = 0, \qquad \|x\|_\infty = 1, \qquad x_i = 1 \quad \text{for some } i \in \{1, \dots, n\}, \tag{6}$$

the solvability of which can be tested using Algorithm 3.

The efficiency of the ILS method depends greatly on the selection of an appropriate ILS solver. It is not necessary to solve (7) exactly, as this is time-consuming. In fact, checking solvability is known to be NP-hard [29]. It suffices to exploit a (preferably fast) algorithm to produce an outer approximation of (6); that is, an approximation that contains the whole solution set. Experience shows that the algorithm proposed in [30] modified so as to work for overconstrained linear systems is a preferable choice. It is sufficiently fast and produces a good approximation of the solution set of (7).

2.3.4. *Direct enumeration*

The ILS method benefits us even when $\boldsymbol{M}$ is not recognized as a regular matrix. In this case, we have an approximation of the solution set of (7), at each iteration of step 4. By embedding them into n-dimensional space and joining them together, we get an outer approximation of the solution set of (6). This is widely usable; see also Section 3. We will present some more details of this procedure.

Let $\boldsymbol{v} \subseteq \mathbb{R}^n$ be an interval vector. We consider the *sign vector set* $\operatorname{sgn}(\boldsymbol{v})$, that is the set of vectors $z \in \{\pm 1\}^n$ with components

$$z_i = \begin{cases} +1, & \underline{v}_i \geq 0, \\ -1, & \underline{v}_i < 0,\ \overline{v}_i \leq 0, \\ \pm 1, & \text{otherwise } (\underline{v}_i < 0 < \overline{v}_i). \end{cases} \tag{8}$$

Clearly, the cardinality of $\mathrm{sgn}(\boldsymbol{v})$ is always a power of 2. Notice that this set does not always consist of the sign vectors of all $v \in \boldsymbol{v}$; the difference is caused when $\underline{v}_i < 0, \overline{v}_i = 0$ holds for some $i = 1, \ldots, n$.

Let $\boldsymbol{x}$ be an outer approximation of the solution set of (6), and let $Z := \mathrm{sgn}(\boldsymbol{x})$. As long as Z has reasonably small cardinality we can check the regularity of $\boldsymbol{M}$ by inspecting all the corresponding orthants and solving the linear programs of Eq. (3) with $b = 0$. There is no need to check the other orthants, since x is a solution of (6) if and only if it is a feasible solution of (3) with $b = 0, z = \mathrm{sgn}(x)$ and $z^T x > 0$. Algorithm 4 gives a formal description of this procedure.

Algorithm 4 (Direct enumeration via Z)

```
1: for all $z \in Z$ do
2:    solve the linear program (3) with $b = 0$;
3:    if (3) is unbounded then
4:       return "$\boldsymbol{M}$ is not regular";
5:    end if
6: end for
7: return "$\boldsymbol{M}$ is regular";
```

2.3.5. Practical implementation

Our implementation exhibits and combines all the methods mentioned in this section. We propose the following procedure (Algorithm 5) for the outer test of Algorithm 1:

Algorithm 5 (Outer test)

```
1: $\boldsymbol{M} := \boldsymbol{A} - \boldsymbol{\lambda} I$;
2: if $M_c$ is singular then
3:    return "$\boldsymbol{\lambda}$ is not an outer interval";
4: end if
5: if $\rho(|M_c^{-1}|M_\Delta) < 1$ then
6:    return "$\boldsymbol{\lambda}$ is an outer interval";
7: end if
8: call Algorithm 2 (Jansson and Rohn) with the number of iterations limited by a constant $K_3$;
9: if the number of iteration does not exceed $K_3$ then
10:    return its output;
11: end if
12: call Algorithm 3 (ILS method);
13: if Algorithm 3 recognize $\boldsymbol{\lambda}$ as an outer interval then
14:    return "$\boldsymbol{\lambda}$ is an outer interval";
15: end if
16: use the obtained approximation $\boldsymbol{x}$ to define $Z$;
17: if $|Z| < K_4$ then
18:    call Algorithm 4;
19:    return its output;
20: end if
21: return "no decision on $\boldsymbol{\lambda}$";
```

Jansson and Rohn method is very fast as long as radii of $\boldsymbol{M}$ are small and $\boldsymbol{\lambda}$ is not close to the border of Λ. If this is not the case, then it can be time-consuming. We limit the number of iterations of this procedure to K_3, where $K_3 := n^3$. If after this number of iterations the result is not conclusive, then we call the ILS method. Finally, if ILS does not succeed, then we compute Z, and if its cardinality is less than K_4, then we call Algorithm 4. Otherwise, we cannot reach a conclusion about $\boldsymbol{\lambda}$. We empirically choose $K_4 := 2^\alpha$ with $\alpha := 2\log(K_3 + 200) - 8$.

Notice that in step 2 of Algorithm 5 we obtain a little more information. Not only is $\boldsymbol{\lambda}$ not an outer interval, but also its half-intervals $[\underline{\lambda}, \lambda_c], [\lambda_c, \overline{\lambda}]$ cannot be outer.

Remark 1. The interval Newton method [13,14] applied to the nonlinear system

$$\boldsymbol{A}x = \lambda x, \qquad \|x\|_2 = 1$$

did not turn out to be efficient. Using the maximum norm was more promising; however, at each iteration the interval Newton method solves an interval linear system that is a consequence of (6), and therefore cannot yield better results than the ILS method.

2.4. The inner test

This section is devoted to the inner test (step 7 in Algorithm 1). We are given a real closed interval $\boldsymbol{\lambda}$ and an interval matrix $\boldsymbol{A}$. The question is whether every $\lambda \in \boldsymbol{\lambda}$ represents an eigenvalue of some $A \in \boldsymbol{A}$. If so, then $\boldsymbol{\lambda}$ is called an *inner interval*.

Using inner testing in interval-valued problems is not a common procedure. It depends greatly on the problem under consideration, since interval parameters are usually correlated, and such correlations are, in general, hard to deal with. However, utilization of inner testing provides two great benefits: it decreases the running time and allows us to measure the sharpness of the approximation.

Our approach is a modification of Jansson and Rohn method.

Proposition 4. *We have that $\lambda \in \mathbb{R}$ is an eigenvalue of some $A \in \boldsymbol{A}$ if the linear programming problem*

$$\max\left\{z^T x;\ (A_c - \lambda I - A_\Delta \mathrm{diag}(z))x \le b,\ (A_c - \lambda I + A_\Delta \mathrm{diag}(z))x \ge b,\ \mathrm{diag}(z)x \ge 0\right\} \tag{9}$$

is unbounded for some $z \in \{\pm 1\}^n$.

Proof. It follows from [28, Theorems 5.3 and 5.4]. □

Proposition 5. *We have that $\boldsymbol{\lambda}$ is an inner interval if the linear programming problem*

$$\max\left\{z^T x^1 - z^T x^2;\ (A_c - A_\Delta \mathrm{diag}(z))(x^1 - x^2) - \underline{\lambda} x^1 + \overline{\lambda} x^2 \le b,\ (A_c + A_\Delta \mathrm{diag}(z))(x^1 - x^2) - \overline{\lambda} x^1 \right.$$
$$\left. + \underline{\lambda} x^2 \ge b,\ \mathrm{diag}(z)(x^1 - x^2) \ge 0,\ x^1, x^2 \ge 0\right\} \tag{10}$$

is unbounded for some $z \in \{\pm 1\}^n$.

Proof. Let $z \in \{\pm 1\}^n$ and let (10) be unbounded. That is, there exists a sequence of feasible points $(x_k^1, x_k^2),\ k = 1, 2, \ldots,$ such that $\lim_{k\to\infty}(z^T x_k^1 - z^T x_k^2) = \infty$. We show that (9) is unbounded for every $\lambda \in \boldsymbol{\lambda}$, and thereby $\boldsymbol{\lambda}$ is an inner interval.

Let $\lambda \in \boldsymbol{\lambda}$ be arbitrary. Define a sequence of points $x_k := (x_k^1 - x_k^2),\ k = 1, 2, \ldots$. Every x_k is a feasible solution to (9), since

$$\begin{aligned}(A_c - \lambda I - A_\Delta \mathrm{diag}(z))x_k &= (A_c - \lambda I - A_\Delta \mathrm{diag}(z))(x_k^1 - x_k^2)\\ &\le (A_c - A_\Delta \mathrm{diag}(z))(x^1 - x^2) - \underline{\lambda} x^1 + \overline{\lambda} x^2 \le b,\end{aligned}$$

and

$$\begin{aligned}(A_c - \lambda I + A_\Delta \mathrm{diag}(z))x_k &= (A_c - \lambda I + A_\Delta \mathrm{diag}(z))(x_k^1 - x_k^2)\\ &\ge (A_c + A_\Delta \mathrm{diag}(z))(x^1 - x^2) - \overline{\lambda} x^1 + \underline{\lambda} x^2 \ge b,\end{aligned}$$

and

$$\mathrm{diag}(z)x_k = \mathrm{diag}(z)(x_k^1 - x_k^2) \ge 0.$$

Next, $\lim_{k\to\infty} z^T x_k = \lim_{k\to\infty} z^T (x_k^1 - x_k^2) = \infty$. Therefore the linear program (9) is unbounded. □

Proposition 5 gives us a sufficient condition for checking whether $\boldsymbol{\lambda}$ is an inner interval. The condition becomes stronger and stronger as $\boldsymbol{\lambda}$ becomes more narrow. The natural question is that of how to search for a sign vector z that guarantees the unboundedness of (10). We can modify the Jansson and Rohn method and inspect all orthants intersecting a given component. In our experience, slightly better results are obtained by the variation described by Algorithm 6.

This approach has several advantages. First, it solves the linear program (10), which has twice as many variables as (3), at most $(n + 1)$ times. Next, we can accelerate Algorithm 1 by means of the following properties:

- Algorithm 6 returns that $\boldsymbol{\lambda}$ is not an inner interval only if λ_c is not an eigenvalue. In this case, neither of the half-intervals $[\underline{\lambda}, \lambda_c]$ and $[\lambda_c, \overline{\lambda}]$ can be inner.
- If (10) is unbounded for some sign vector z, then it is sometimes probable that the interval adjacent to $\boldsymbol{\lambda}$ is also inner and the corresponding linear program is (10), unbounded for the same sign vector z. Therefore, the sign vector z is worth remembering for the subsequent iterations of step 3 in Algorithm 6. This is particularly valuable when the list L is implemented as a stack; see the discussion in Section 2.

2.5. Complexity

In this section we discuss the complexity of Algorithm 1. Recall that even testing the regularity of a matrix is an NP-hard problem; thus we cannot expect a polynomial algorithm. By $LP(m, n)$ we denote the (bit) complexity of solving a linear program with $O(m)$ inequalities and $O(n)$ unknowns.

Our algorithm is a subdivision algorithm. Its complexity is the number of tests it performs, times the complexity of each step. At each step we perform, in the worst case, an outer and an inner test.

Algorithm 6 (Inner test)

```
1: call Algorithm 2 (Jansson and Rohn) with $\boldsymbol{M} := \boldsymbol{A} - \lambda_c I$;
2: if $\boldsymbol{M}$ is regular then
3:    return $\boldsymbol{\lambda}$ is not inner interval;
4: end if
5: let $z$ be a sign vector for which (3) is unbounded;
6: solve (10);
7: if (10) is unbounded then
8:    return "$\boldsymbol{\lambda}$ is an inner interval";
9: else if (10) is infeasible then
10:    return "$\boldsymbol{\lambda}$ is possibly not inner";
11: end if
12: for all $y \in N(z)$ do
13:    solve (10) with $y$ as a sign vector;
14:    if (10) is unbounded then
15:       return "$\boldsymbol{\lambda}$ is an inner interval";
16:    end if
17: end for
18: return "$\boldsymbol{\lambda}$ is possibly not inner";
```

Let us first compute the number of steps of the algorithm. Let $\max\{\boldsymbol{A}_{ij}\} := \max\{\max\{\underline{A}_{ij}\}, \max\{\overline{A}_{ij}\}\} \leq 2^{\tau}$, i.e. we consider a bound on the absolute value on the numbers used to represent the interval matrix. From Section 2.2 we deduce that the real eigenvalue set of $\boldsymbol{A}$ is contained in an interval, centered at zero and with radius bounded by the sum of the spectral radii of S_S and S_Δ. Evidently the bound 2^{τ} holds for the elements of these matrices, as well. Since for an $n \times n$ matrix M the absolute value of its (possible complex) eigenvalues is bounded by $n \max_{ij} |M_{ij}|$, we deduce that the spectral radius of S_S and S_Δ is bounded by $n2^{\tau}$ and thus the real eigenvalues of $\boldsymbol{A}$ are in the interval $[-n2^{\tau+1}, n2^{\tau+1}]$. Let $\varepsilon = 2^{-k}$ be the input accuracy. In this case the total number of intervals that we need to test, or in other words the total number of steps that the algorithm performs, is $n2^{\tau+1}/2^{-k} = n2^{\tau+k+1}$.

It remains to compute the complexity of each step. At each step we perform an inner and an outer test. For each of these tests we should solve, in the worst case, $2^{O(n)}$ linear programs that consist of $O(n)$ variables and inequalities. The exponential number of linear programs is a consequence of the fact that we should enumerate all the vertices of a hypercube in n dimensions (refer to Algorithm 4).

Thus the total complexity of the algorithm is $O(n\, 2^{k+\tau+1}\, 2^n\, LP(n, n))$.

2.6. The interval hull

We can slightly modify Algorithm 1 to approximate the interval hull of Λ, i.e., the smallest interval containing Λ. Let λ^L (resp. λ^U) be the lower (resp. upper) boundary of Λ, i.e.,

$$\lambda^L := \inf\{\lambda;\, \lambda \in \Lambda\} \quad \text{and} \quad \lambda^U := \sup\{\lambda;\, \lambda \in \Lambda\}.$$

In order to compute λ^L, we consider the following modifications of Algorithm 1. We remove all the steps that refer to the list L_{inn}, and we change step 8 to

```
⋮
8:    return $L_{\text{unc}}$;
⋮
```

The modified algorithm returns L_{unc} as output. The union of all the intervals in L_{unc} is an approximation of the lower boundary point λ^L. If the list L_{unc} is empty, then the eigenvalue set Λ is empty, too.

An approximation of the upper boundary point, λ^U, can be computed as a negative value of the lower eigenvalue boundary point of the interval matrix $(-\boldsymbol{A})$.

3. Exact bounds

Algorithm 1 yields an outer and an inner approximation of the set of eigenvalues Λ. In this section we show how to use it for computing the exact boundary points of Λ. This *exactness* is limited by the use of floating point arithmetic. Rigorous results would be obtained by using interval arithmetic, but this is a direct modification of the proposed algorithm and we do not discuss it in detail.

As long as interval radii of $\boldsymbol{A}$ are small enough, we are able, in most of the cases, to determine the exact bounds in a reasonable time. Surprisingly, sometimes computing exact bounds is faster than high precision approximation.

We build on [19, Theorem 3.4]:

Theorem 3. *Let $\lambda \in \partial\Lambda$. Then there exist nonzero vectors $x, p \in \mathbb{R}^n$ and vectors $y, z \in \{\pm 1\}^n$ such that*

$$\begin{aligned}
&\big(A_c - \operatorname{diag}(y)A_\Delta\operatorname{diag}(z)\big)x = \lambda x,\\
&\big(A_c^T - \operatorname{diag}(z)A_\Delta^T\operatorname{diag}(y)\big)p = \lambda p,\\
&\operatorname{diag}(z)x \geq 0,\\
&\operatorname{diag}(y)p \geq 0.
\end{aligned} \tag{11}$$

Theorem 3 asserts that the boundary eigenvalues are produced by special matrices $A_{y,z} \in \boldsymbol{A}$ of the form of $A_{y,z} := A_c - \operatorname{diag}(y)A_\Delta\operatorname{diag}(z)$. Here, z is the sign vector of the right eigenvector x, and y is the sign vector of the left eigenvector p. Recall that a *right eigenvector* is a nonzero vector x satisfying $Ax = \lambda x$, and a *left eigenvector* is a nonzero vector p satisfying $A^Tp = \lambda p$.

In our approach, we are given an interval $\boldsymbol{\lambda}$ and we are trying to find outer approximations of the corresponding left and right eigenvectors, i.e. $\boldsymbol{p}$ and $\boldsymbol{x}$, respectively. If no component of $\boldsymbol{p}$ and $\boldsymbol{x}$ contains zero, then the sign vectors $y := \operatorname{sgn}(\boldsymbol{p})$ and $z := \operatorname{sgn}(\boldsymbol{x})$ are uniquely determined. In this case we enumerate all the eigenvalues of $A_{y,z}$. If only one of them belongs to $\boldsymbol{\lambda}$, then we have succeeded.

If the eigenvectors in $\boldsymbol{p}$ and $\boldsymbol{x}$ are normalized according to (5), then we must inspect not only $A_{y,z}$, but also $A_{-y,z}$ (the others, $A_{y,-z}$ and $A_{-y,-z}$, are not needed due to symmetry).

The formal description is given in Algorithm 7.

Algorithm 7 (Exact bound)

```
1: $\boldsymbol{M} := \boldsymbol{A} - \boldsymbol{\lambda}I$;
2: call Algorithm 3 (ILS method) with the input matrix $\boldsymbol{M}^T$ to obtain an outer approximation $\boldsymbol{p}$ of the corresponding solutions.
3: if $\underline{p}_i \leq 0 \leq \overline{p}_i$ for some $i = 1, \dots, n$ then
4:     return "bound is possibly not unique";
5: end if
6: $y := \operatorname{sgn}(\boldsymbol{p})$;
7: call Algorithm 3 (ILS method) with the input matrix $\boldsymbol{M}$ to obtain an outer approximation $\boldsymbol{x}$ of the corresponding solutions.
8: if $\underline{x}_i \leq 0 \leq \overline{x}_i$ for some $i = 1, \dots, n$ then
9:     return "bound is possibly not unique";
10: end if
11: $z := \operatorname{sgn}(\boldsymbol{x})$;
12: let $L$ be a set of all eigenvalues of $A_{y,z}$ and $A_{-y,z}$;
13: if $L \cap \boldsymbol{\lambda} = \emptyset$ then
14:     return "no boundary point in $\boldsymbol{\lambda}$";
15: else if $L \cap \boldsymbol{\lambda} = \{\lambda^*\}$ then
16:     return "$\lambda^*$ is a boundary point candidate";
17: else
18:     return "bound is possibly not unique";
19: end if
```

We now describe how to integrate this procedure into our main algorithm. Suppose that at some iteration of Algorithm 1 we have an interval $\boldsymbol{\lambda}^1$ recognized as outer. Suppose next that the following current interval $\boldsymbol{\lambda}^2$ is adjacent to $\boldsymbol{\lambda}^1$ (i.e., $\overline{\lambda}^1 = \underline{\lambda}^2$); it is not recognized as outer and it fulfills the precision test (step 9). According to the result of Algorithm 7 we distinguish three possibilities:

- If $L \cap \boldsymbol{\lambda}^2 = \emptyset$ then there cannot be any eigenvalue boundary point in $\boldsymbol{\lambda}^2$, and therefore it is an outer interval.
- If $L \cap \boldsymbol{\lambda}^2 = \{\lambda^*\}$ then λ^* is the exact boundary point required, and moreover $[\lambda^*, \overline{\lambda}^2] \subseteq \Lambda$.
- If $|L \cap \boldsymbol{\lambda}^2| > 1$ then the exact boundary point is $\lambda^* := \min\{\lambda;\ \lambda \in L \cap \boldsymbol{\lambda}^2\}$. However, we cannot say anything about the remaining interval $[\lambda^*, \overline{\lambda}^2]$.

A similar procedure is applied when $\boldsymbol{\lambda}^1$ is inner and $\boldsymbol{\lambda}^2$ is adjacent and narrow enough.

We can simply extend Algorithm 7 to the case where there are some zeros in the components of $\boldsymbol{p}$ and $\boldsymbol{x}$. In this case, the sign vectors y and z are not determined uniquely. Thus, we have to take into account the sets of all the possible sign vectors. Let $\boldsymbol{v}$ be an interval vector and $\operatorname{sgn}'(\boldsymbol{v})$ be a sign vector set, that is, the set of all sign vectors $z \in \{\pm 1\}^n$ satisfying

$$z_i = \begin{cases} +1, & \underline{v}_i > 0,\\ -1, & \overline{v}_i < 0,\\ \pm 1, & \text{otherwise } (\underline{v}_i \leq 0 \leq \overline{v}_i). \end{cases}$$

The definition of $\mathrm{sgn}'(\boldsymbol{v})$ slightly differs from that of $\mathrm{sgn}(\boldsymbol{v})$ in (8). Herein, we must take into account the both signs of z_i whenever $\boldsymbol{v}_i$ contains zero (even on a boundary). Assume

$$Y := \mathrm{sgn}'(\boldsymbol{p}), \qquad Z := \mathrm{sgn}'(\boldsymbol{x}).$$

Instead of two matrices, $A_{y,z}$ and $A_{-y,z}$, we must inspect all possible combinations with $y \in Y$ and $z \in Z$. In this way, step 12 of Algorithm 7 will we replaced by

```
⋮
12′:  $L := \{\lambda;\ \lambda$ is an eigenvalue of $A_{y,z}$ or of $A_{-y,z},\ y \in Y,\ z \in Z\}$;
⋮
```

The cardinality of Y is a power of 2, and the cardinality of Z as well. Since we have to enumerate eigenvalues of $|Y| \cdot |Z|$ matrices, step 12′ is tractable for only reasonably small sets Y and Z.

4. Numerical experiments

In this section we present results of some numerical experiments. They confirm the quality of the algorithm presented. We are able to determine the eigenvalue set exactly or at least very sharply for dimensions up to about 30. The running time depends heavily not only on the dimension, but also on the widths of matrix intervals.

We also compared our implementation with another techniques that solve directly the interval nonlinear system (1). It turned out that such techniques are comparable only for very small dimensions, i.e. ∼5. Results of our numerical experiments are displayed in tables that follow and can be interpreted using the following notation:

n	Problem dimension
ε	Precision
R	Maximal radius
"Exactness"	Indication of whether exact bounds of Λ were achieved; if not, we display the number of uncertain intervals
"Time"	Computing time in hours, minutes and seconds
"Hull time"	Computing time of the interval hull of Λ; see Section 2.6

Note that ε refers to the precision used in the step 9 of Algorithm 1. For the additional computation of exact boundary points we use $10^{-4}\varepsilon$ precision.

Generally, better results were obtained for smaller R, as both the Jansson and Rohn method and the sufficient regularity condition are more efficient for smaller radii of matrix intervals.

The results were carried on an Intel `Pentium(R) 4`, CPU 3.4 GHz, with 2 GB RAM, and the source code was written in C++ using `GLPK v.4.23` [31] for solving linear programs, `CLAPACK v.3.1.1` for its linear algebraic routines, and `PROFIL/BIAS v.2.0.4` [32] for interval arithmetics. Notice, however, that routines of `GLPK` and `CLAPACK` [33] do not produce verified solutions, and for real-life problems preferably verified software or interval arithmetic should be used.

Example 1 (*Random Matrices*)**.** The entries of the midpoint matrix A_c are chosen randomly with uniform distribution in $[-20, 20]$. The entries of the radius matrix A_Δ are chosen randomly with uniform distribution in $[0, R]$, where R is a positive real number. The results are displayed in Table 1.

Example 2 (*Random Symmetric Matrices*)**.** The entries of A_c and A_Δ are chosen randomly in the same manner as in Example 1; the only difference is that both of these matrices are composed to be symmetric. See Table 2 for the results.

Example 3 (*Random $\boldsymbol{A}^T\boldsymbol{A}$ Matrices*)**.** The entries of A_c and A_Δ are chosen randomly as in Example 1, and our algorithm is applied on the matrix generated by the product $\boldsymbol{A}^T\boldsymbol{A}$. In this case, the maximal radius value R is a bit misleading, since it refers to the original matrix $\boldsymbol{A}$ instead of the product used. The results are displayed in Table 3.

Example 4 (*Random Nonnegative Matrices*)**.** The entries of A_c and A_Δ are chosen randomly as in Example 1, and the eigenvalue problem is solved for its absolute value

$$|\boldsymbol{A}| := \{|A|;\ A \in \boldsymbol{A}\}.$$

The absolute value of an interval matrix is again an interval matrix and with entries

$$|\boldsymbol{A}|_{ij} = \begin{cases} \boldsymbol{A}_{ij} & \underline{A}_{ij} \geq 0, \\ -\boldsymbol{A}_{ij} & \overline{A}_{ij} \leq 0, \\ [0, \max(-\underline{A}_{ij}, \overline{A}_{ij})] & \text{otherwise.} \end{cases}$$

See Table 4 for the results.

Table 1
Random matrices.

n	ε	R	Exactness	Time	Hull time
5	0.1	1	Exact	2 s	1 s
10	0.1	0.1	Exact	7 s	2 s
10	0.1	0.5	Exact	9 s	4 s
10	0.1	1	Exact	16 s	1 s
10	0.1	5	Exact	1 min 12 s	1 min 11 s
15	0.1	0.1	Exact	37 s	5 s
15	0.1	0.5	Exact	10 min 29 s	6 s
15	0.1	0.5	Exact	20 min 54 s	35 s
15	0.1	1	Exact	7 min 59 s	1 min 12 s
20	0.1	0.1	Exact	2 min 16 s	10 s
20	0.1	0.1	Exact	7 min 27 s	39 s
20	0.1	0.5	Exact	21 min 6 s	46 s
25	0.1	0.01	Exact	5 min 46 s	23 s
25	0.1	0.05	Exact	10 min 39 s	1 min 34 s
30	0.01	0.01	Exact	14 min 37 s	54 s
30	0.01	0.1	Exact	48 min 31 s	29 s
40	0.01	0.01	Exact	–	2 min 20 s
40	0.01	0.05	Exact	–	1 h 42 min 36 s
40	0.01	0.1	Exact	–	1 h 52 min 15 s
50	0.01	0.01	Exact	–	9 min 25 s
50	0.01	0.1	2	–	21 min 34 s

Table 2
Random symmetric matrices.

n	ε	R	Exactness	Time	Hull time
5	0.1	1	Exact	3 s	1 s
10	0.1	0.1	Exact	11 s	1 s
10	0.1	0.5	Exact	17 s	1 s
10	0.1	1	2	2 min 18 s	1 s
10	0.1	5	2	11 s	10 s
15	0.1	0.1	Exact	3 min 51 s	1 s
15	0.1	0.5	6	31 min 43 s	4 s
20	0.1	0.01	Exact	2 min 25 s	3 s
20	0.1	0.05	Exact	6 min 39 s	4 s
20	0.1	0.1	Exact	27 min 48 s	8 s
20	0.1	0.1	10	40 min 19 s	8 s
25	0.1	0.01	Exact	7 min 51 s	12 s
25	0.1	0.05	Exact	1 h 59 min 11 s	11 s
30	0.01	0.1	Exact	–	29 s
40	0.01	0.1	Exact	–	6 min 15 s
50	0.01	0.01	Exact	–	1 min 23 s
50	0.01	0.1	Exact	–	1 h 2 min 43 s
100	0.01	0.01	Exact	–	34 min 5 s

Table 3
Random $\boldsymbol{A}^T\boldsymbol{A}$ matrices.

n	ε	R	Exactness	Time	Hull time
5	0.1	0.1	Exact	5 s	1 s
10	0.1	0.1	Exact	37 s	3 s
10	0.1	0.1	Exact	4 min 0 s	1 s
10	0.1	0.5	Exact	1 min 35 s	7 s
10	0.1	1	Exact	1 min 3 s	56 s
15	0.1	0.001	Exact	1 min 1 s	3 s
15	0.1	0.002	Exact	40 s	2 s
15	0.1	0.01	3	3 min 38 s	17 s
15	0.1	0.02	1	1 min 58 s	13 s
15	0.1	0.1	Exact	39 min 27 s	4 min 48 s
20	0.01	0.1	Exact	–	1 h 18 min 16 s

Figs. 1–4 present some examples of the eigenvalue set Λ. Intervals of Λ are colored red while the outer intervals are yellow and green; yellow color is for the intervals recognized by the sufficient regularity condition (step 5 of Algorithm 5), and green is for the remainder.

Table 4
Random nonnegative matrices.

n	ε	R	Exactness	Time	Hull time
10	0.01	0.1	Exact	13 s	1 s
10	0.01	1	Exact	8 s	1 s
15	0.01	0.1	Exact	2 min 22 s	6 s
15	0.01	0.1	Exact	47 s	4 s
15	0.01	0.5	Exact	1 min 53 s	27 s
15	0.01	1	Exact	57 s	37 s
15	0.01	5	Exact	–	1 h 8 min 49 s
20	0.01	0.1	Exact	3 min 55 s	9 s
20	0.01	0.5	Exact	8 min 36 s	1 min 19 s
25	0.01	0.1	Exact	51 min 58 s	12 s
30	0.01	0.01	Exact	19 min 47 s	49 s
30	0.01	0.1	Exact	–	37 min 44 s
40	0.01	0.01	Exact	–	2 min 41 s
40	0.01	0.05	Exact	–	15 min 57 s
50	0.01	0.1	Exact	–	2 h 2 min 22 s

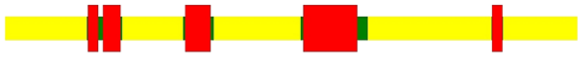

Fig. 1. Random matrix, $n = 30$, $R = 0.1$, computing time 48 min 31 s, initial approximation $[-86.888, 86.896]$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

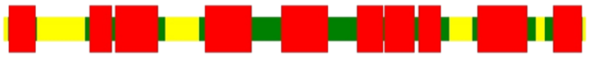

Fig. 2. Random symmetric matrix, $n = 15$, $R = 0.5$, computing time 13 min 48 s, initial approximation $[-60.614, 58.086]$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

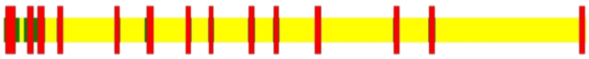

Fig. 3. Random $\boldsymbol{A}^T\boldsymbol{A}$ matrix, $n = 15$, $R = 0.02$, computing time 1 min 58 s, initial approximation $[-39.620, 5679.196]$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

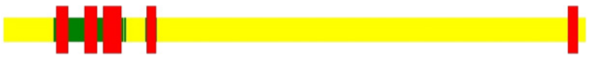

Fig. 4. Random nonnegative matrix, $n = 15$, $R = 0.2$, computing time 2 min 22 s, initial approximation $[-27.548, 144.164]$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Example 5 (*Interval Matrices in Robotics*)**.** The following problem usually appears in experimental planning, e.g. [6]. We consider a PRRRP planar mechanism, where P stands for a prismatic and R for a rotoid joint; for further details we refer the reader to [34].

We consider the mechanism of Fig. 5. Let $X = (x, y)$ be a point in the plane, which is linked to two points, M and N, using two fixed length bars so that it holds that $\|X - M\| = r_1$, $\|X - N\| = r_2$. We can move M (respectively N) between the fixed points $A = (0, 0)$ and $B = (L, 0)$ (respectively $C = (L + r_3, 0)$ and $D = (2.L + r_3, 0)$) using two prismatic joints, so $\|A - M\| = l_1$, $\|C - N\| = l_2$. In the example that we consider, Fig. 5, the points A, B, C, D are aligned.

If we control the length l_1 and l_2 by using two linear actuators we allow the displacement of the end-effector X to have two degrees of freedom in a planar workspace that is limited by the articular constraints $l_1, l_2 \in [0, L]$. The two equations $F_1(X, l_1) \equiv \|M - X\|^2 - r_1^2 = 0$ and $F_2(X, l_2) \equiv \|N - X\|^2 - r_2^2 = 0$ link the generalized coordinates (x, y) and the articular coordinates l_1, l_2.

The calibration of such a mechanism is not an easy problem due to assembly and manufacturing errors, and because the kinematic parameters, that is the lengths r_1, r_2 and r_3, are not well known.

The aim is to estimate them using several measurements, $k = 1, \dots, n$, of the end-effector X_k and the corresponding measurements of the articular coordinates $l_{k,1}, l_{k,2}$.

The identification procedure of r_1, r_2, and r_3 is based on a classical least square approach for the (redundant) system

$$F \equiv [F_{1,1}(X_1, l_{1,1}), F_{1,2}(X_1, l_{1,2}), \dots, F_{n,1}(X_n, l_{n,1}), F_{n,2}(X_n, l_{n,2})]^T = 0.$$

That is, we want to compute r_1, r_2, and r_3 that minimize the quantity F^TF.

A natural question is that of how to estimate the measurements positions inside the workspace [7] to improve the robustness of the numerical solution of a least square method. For this we can use the observability index [8], which is a square root of the smallest eigenvalue of J^TJ, where J is the identification Jacobian. It is defined by the derivatives of F_{1k}

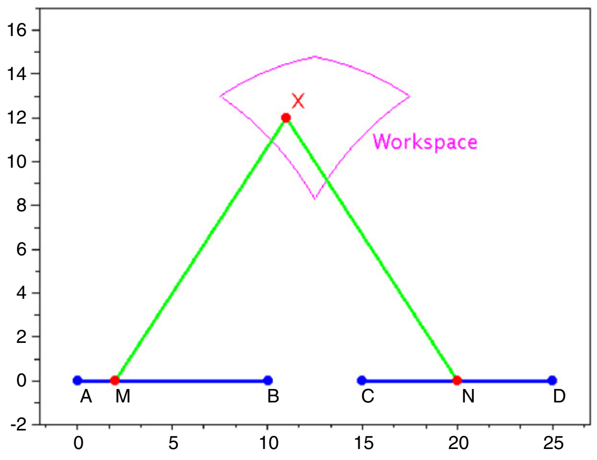

Fig. 5. PRRRP planar parallel robot.

and F_{2k} with respect to the kinematic parameters r_1, r_2, and r_3, that is

$$J = \begin{pmatrix} \frac{\partial F_{1,1}(X_1, l_{1,1})}{\partial r_1} & \frac{\partial F_{1,1}(X_1, l_{1,1})}{\partial r_2} & \frac{\partial F_{1,1}(X_1, l_{1,1})}{\partial r_3} \\ \frac{\partial F_{1,2}(X_1, l_{1,2})}{\partial r_1} & \frac{\partial F_{1,2}(X_1, l_{1,2})}{\partial r_2} & \frac{\partial F_{1,2}(X_1, l_{1,2})}{\partial r_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial F_{n,1}(X_n, l_{n,1})}{\partial r_1} & \frac{\partial F_{n,1}(X_n, l_{n,1})}{\partial r_2} & \frac{\partial F_{n,1}(X_n, l_{n,1})}{\partial r_3} \\ \frac{\partial F_{n,2}(X_n, l_{n,2})}{\partial r_1} & \frac{\partial F_{n,2}(X_n, l_{n,2})}{\partial r_2} & \frac{\partial F_{n,2}(X_n, l_{n,2})}{\partial r_3} \end{pmatrix}.$$

The observability index can be equivalently defined as the third-largest eigenvalue of the matrix

$$\begin{pmatrix} 0 & J \\ J^T & 0 \end{pmatrix}. \tag{12}$$

We employ this approach since it gives rise to more accurate estimates.

Recall that due to measurement errors, it is not possible to obtain the actual values of the kinematic parameters. However, if the set of measurements is chosen so as to maximize this index, the error of the end-effector positions after calibration is minimized.

We demonstrate our approach by setting $n = 2$. Let $r_1 = r_2 = 15$, $r_3 = 5$, and $L = 10$.

If $l_{1,1} \in [0, 5]$, $l_{1,2} \in [5, 10]$, $l_{2,1} \in [5, 10]$ and $l_{2,2} \in [0, 5]$ then

$$J = \begin{pmatrix} [-30, -30] & 0 & 0 \\ 0 & [-30, -30] & [-30, -30] \\ 0 & [15.2, 25] & [5, 14.8] \end{pmatrix},$$

and the third-largest eigenvalue λ_3 of (12) lies in the interval [0.25, 12.53].

Similarly, if $l_{1,1} \in [0, 2]$, $l_{1,2} \in [8, 10]$, $l_{2,1} \in [8, 10]$, and $l_{2,2} \in [0, 2]$, this is workspace 1, ws_1, in Fig. 6; then $\lambda_3 = [7.56, 12.53]$, where

$$J = \begin{pmatrix} [-30, -30] & 0 & 0 \\ 0 & [-30, -30] & [-30, -30] \\ 0 & [21, 25] & [5, 9] \end{pmatrix}.$$

If $l_{1,1} \in [4, 7]$, $l_{1,2} \in [9, 10]$, $l_{2,1} \in [9, 10]$, and $l_{2,2} \in [4, 7]$, this is workspace 2, ws_2, in Fig. 6; then $\lambda_3 = [2.52, 7.56]$, where

$$J = \begin{pmatrix} [-30, -30] & 0 & 0 \\ 0 & [-30, -30] & [-30, -30] \\ 0 & [17, 21] & [9, 13] \end{pmatrix}.$$

As regards the last two examples, it is always better to chose the measurement poses in ws_1, in red in Fig. 6, than the ones in ws_2, in blue in Fig. 6.

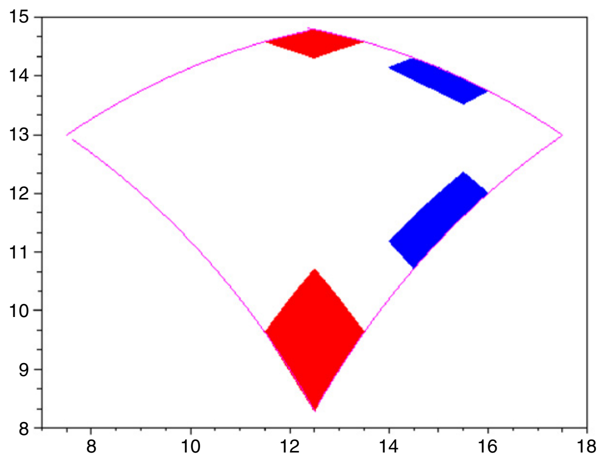

Fig. 6. Workspaces ws_1 in red and ws_2 in blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5. Conclusion

In this paper we considered the problem of computing the real eigenvalues of matrices with interval entries. Sharp approximation of the set of the (real) eigenvalues is an important subroutine in various engineering applications. We proposed an algorithm based on a branch and prune scheme and splitting only along one dimension (real axis) to compute the intervals of the real eigenvalues. The algorithm approximates the real eigenvalues with an accuracy depending on a given positive parameter ε.

Numerical experiments demonstrate that the algorithm is applicable in high dimensions. An exact bound can be achieved in real time up to the dimension of 30, but more or less sharp approximations can be produced in any dimension. To the best of our knowledge there is no comparable method for dimension greater that 5.

Our algorithm could be also seen as a first step of an algorithm that produces intervals (in the complex plane) that contain all the eigenvalues of a given interval matrix. This is work in progress.

References

[1] W. Karl, J. Greschak, G. Verghese, Comments on 'A necessary and sufficient condition for the stability of interval matrices', Internat. J. Control 39 (4) (1984) 849–851.
[2] Z. Qiu, P.C. Müller, A. Frommer, An approximate method for the standard interval eigenvalue problem of real non-symmetric interval matrices, Comm. Numer. Methods Engrg. 17 (4) (2001) 239–251.
[3] A.D. Dimarogonas, Interval analysis of vibrating systems, J. Sound Vibration 183 (4) (1995) 739–749.
[4] F. Gioia, C.N. Lauro, Principal component analysis on interval data, Comput. Statist. 21 (2) (2006) 343–363.
[5] D. Chablat, P. Wenger, F. Majou, J. Merlet, An interval analysis based study for the design and the comparison of 3-DOF parallel kinematic machines, Int. J. Robot. Res. 23 (6) (2004) 615–624.
[6] E. Walter, L. Pronzato, Identification of Parametric Models, Springer, Heidelberg, 1997.
[7] D. Daney, Y. Papegay, B. Madeline, Choosing measurement poses for robot calibration with the local convergence method and Tabu search, Int. J. Robot. Res. 24 (6) (2005) 501.
[8] A. Nahvi, J. Hollerbach, The noise amplification index for optimal pose selection in robot calibration, in: IEEE International Conference on Robotics and Automation, Citeseer, 1996, pp. 647–654.
[9] D. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms, 2nd ed., in: Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1997.
[10] L. Busé, H. Khalil, B. Mourrain, Resultant-based methods for plane curves intersection problems, in: V. Ganzha, E. Mayr, E. Vorozhtsov (Eds.), Proc. 8th Int. Workshop Computer Algebra in Scientific Computing, in: LNCS, vol. 3718, Springer, 2005, pp. 75–92.
[11] H. Stetter, Numerical Polynomial Algebra, Society for Industrial Mathematics, 2004.
[12] G. Alefeld, J. Herzberger, Introduction to Interval Computations, Academic Press, London, 1983.
[13] E. Hansen, G.W. Walster, Global Optimization Using Interval Analysis, 2nd ed., Marcel Dekker, New York, 2004, revised and expanded.
[14] A. Neumaier, Interval Methods for Systems of Equations, Cambridge University Press, Cambridge, 1990.
[15] S. Poljak, J. Rohn, Checking robust nonsingularity is NP-hard, Math. Control Signals Systems 6 (1) (1993) 1–9.
[16] A.S. Deif, The interval eigenvalue problem, Z. Angew. Math. Mech. 71 (1) (1991) 61–64.
[17] J. Rohn, A. Deif, On the range of eigenvalues of an interval matrix, Computing 47 (3–4) (1992) 373–377.
[18] A. Deif, J. Rohn, On the invariance of the sign pattern of matrix eigenvectors under perturbation, Linear Algebra Appl. 196 (1994) 63–70.
[19] J. Rohn, Interval matrices: singularity and real eigenvalues, SIAM J. Matrix Anal. Appl. 14 (1) (1993) 82–91.
[20] J. Rohn, Bounds on eigenvalues of interval matrices, ZAMM Z. Angew. Math. Mech. 78 (Suppl. 3) (1998) S1049–S1050.
[21] G. Alefeld, G. Mayer, Interval analysis: theory and applications, J. Comput. Appl. Math. 121 (1–2) (2000) 421–464.
[22] E.R. Hansen, G.W. Walster, Sharp bounds on interval polynomial roots, Reliab. Comput. 8 (2) (2002) 115–122.
[23] L. Jaulin, M. Kieffer, O. Didrit, É. Walter, Applied Interval Analysis. With Examples in Parameter and State Estimation, Robust Control and Robotics, Springer, London, 2001.
[24] F. Rouillier, Z. Zimmermann, Efficient isolation of polynomial's real roots, J. Comput. Appl. Math. 162 (1) (2004) 33–50.
[25] I.Z. Emiris, B. Mourrain, E.P. Tsigaridas, Real algebraic numbers: complexity analysis and experimentation, in: P. Hertling, C. Hoffmann, W. Luther, N. Revol (Eds.), Reliable Implementations of Real Number Algorithms: Theory and Practice, in: LNCS, vol. 5045, Springer-Verlag, 2008, pp. 57–82. Also available in: www.inria.fr/rrrt/rr-5897.html.

[26] W. Krandick, Isolierung reeller nullstellen von polynomen, in: J. Herzberger (Ed.), Wissenschaftliches Rechnen, Akademie-Verlag, Berlin, 1995, pp. 105–154.
[27] G. Rex, J. Rohn, Sufficient conditions for regularity and singularity of interval matrices, SIAM J. Matrix Anal. Appl. 20 (2) (1998) 437–445.
[28] C. Jansson, J. Rohn, An algorithm for checking regularity of interval matrices, SIAM J. Matrix Anal. Appl. 20 (3) (1999) 756–776.
[29] J. Rohn, Solvability of systems of interval linear equations and inequalities, in: M. Fiedler, J. Nedoma, J. Ramík, J. Rohn, K. Zimmermann (Eds.), Linear Optimization Problems with Inexact Data, Springer, New York, 2006, pp. 35–77 (Chapter 2).
[30] O. Beaumont, Algorithmique pour les intervalles: comment obtenir un résultat sűr quand les données sont incertaines, Ph.D. Thesis, Université de Rennes 1, Rennes, 1999.
[31] A. Makhorin, GLPK—GNU linear programming kit. http://www.gnu.org/software/glpk/.
[32] O. Knüppel, D. Husung, C. Keil, PROFIL/BIAS—a C++ class library. http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html.
[33] CLAPACK—linear algebra Package written for C. http://www.netlib.org/clapack/.
[34] X.-J. Liu, J. Wang, G. Pritschow, On the optimal kinematic design of the PRRRP 2-DOF parallel mechanism, Mech. Mach. Theory 41 (9) (2006) 1111–1130.