Institute of Computer Science
Academy of Sciences of the Czech Republic

A Handbook of Results on Interval Linear Problems

Dedicated in memoriam to my parents

Mrs. Nina Rohnová and Mr. Robert Rohn

Jiří Rohn
http://uivtx.cs.cas.cz/~rohn

Technical report No. V-1163

07.04.2005 / 23.09.2012

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: +420 266 051 111, fax: +420 286 585 789,
e-mail:rohn@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

A Handbook of Results on Interval Linear Problems

Dedicated in memoriam to my parents

Mrs. Nina Rohnová and Mr. Robert Rohn

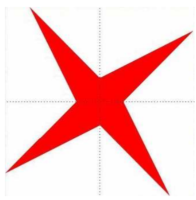
Jiří Rohn
http://uivtx.cs.cas.cz/~rohn

Technical report No. V-1163

07.04.2005 / 23.09.2012

Abstract:

This text surveys important results on interval matrices, interval linear equations (both square and rectangular), and interval linear programming (without proofs). It is based on a "one-topic-one-page" approach, in which each topic is allotted the space of one page only, and contains MATLAB-like descriptions of 15 basic algorithms. The bibliography contains direct links to many papers quoted. Verification versions of some of the algorithms presented in the text can be found in the VERSOFT software package. The Handbook was finalized on 07.04.2005 and was left as an internet text only; here it is published in its original 2005 form as a technical report. During the years 2005-2012 many of the results have been improved; they can be found at author's web page http://uivtx.cs.cas.cz/~rohn/publist/000home.htm. [1]



Keywords:
Interval linear problems, auxiliary results, interval matrices, systems of interval linear equations (square case), systems of interval linear equations and inequalities (rectangular case), interval linear programming, algorithms, and many others.

[1]Above: logo of interval computations and related areas (depiction of the solution set of the system $[2,4]x_1 + [-2,1]x_2 = [-2,2]$, $[-1,2]x_1 + [2,4]x_2 = [-2,2]$ (Barth and Nuding [6])).

Motto

Was sich überhaupt sagen läßt, läßt sich klar sagen;
und wovon man nicht reden kann, darüber muß man schweigen.

L. Wittgenstein, Tractatus logico-philosophicus,
Routledge & Kegan Paul Ltd., London 1922

Contents

Preface

Aim. It has been the aim of this text to present a selection of important results on interval linear problems in a unified and concise way.

Philosophy. The philosophy behind the text is the "one-topic-one-page" approach, in which each topic is allotted the space of one page only.

Layout. There are basically two types of problems handled in interval analysis: *decision problems* (as checking whether an interval matrix is regular), and *computational problems* (as computation of the inverse of a regular interval matrix). For decision problems I was using the following page layout:

Definition. (basic notion of the page)

Problem. (problem formulation)

Necessary and sufficient condition.

Complexity.

Sufficient condition. (if the problem is NP-hard)

Algorithm. (reference to Chapter 7, or to the above sufficient condition)

Comment. (if necessary)

Operation. (the way the algorithm is operating)

Special features. (explanations, connections or related results of particular interest)

References. (sources of given or related results; without names)

and for the computational problems a similar layout

Definition.

Problem.

Formula(e). (formula(e) used in the algorithm)

Complexity.

Algorithm.

Comment.

Operation.

Special features.

References.

Sometimes some of the headings are missing. Occasionally, when necessary, I also added another headings, as **Intro**, **Fact**, **Idea**, **Formulae for enclosures**, **Apology**[2], etc.

Algorithms. It has been my second goal to present not-a-priori-exponential algorithms for solving NP-hard problems. They are those forming the branch starting with **signaccord** in the scheme on p. 51.

Algorithm form. All the algorithms are gathered in Chapter 7. They are described in the form of MATLAB-like functions, but with formulae written in the usual mathematical way. In particular, the output variable *flag* always gives a verbal description of the output.

Hyperlinks. The source text contains hyperlinks that make it easy to flip through it simply by clicking on links colored in magenta, or, in the Contents, in blue. In particular, each item in the bibliography is appended with numbers (in magenta) of the pages where it is referenced from. (This feature also allows you to verify that all the bibliographical items have been referenced.) My own papers listed can be downloaded directly by clicking on the respective URLs in the bibliography.[3]

Prague, Easter 2005

Jiri Rohn
(rohn@cs.cas.cz)

[2] On p. 40.
[3] Unfortunately, this works only in the dvi file, not in the pdf file.

Chapter 1

Notations

Subject. Notations used are introduced and summarized in this chapter.

1.1 Basic notations

1.1.1 Linear algebraic notations

Notation for matrices. The ith row of a matrix A is denoted by $A_{i\bullet}$, the jth column by $A_{\bullet j}$. For two matrices A, B of the same size, inequalities like $A \le B$ or $A < B$ are understood componentwise. A is called nonnegative if $0 \le A$ and symmetric if $A^T = A$; A^T is the transpose of A. $A \circ B$ denotes the Hadamard (entrywise) product of $A, B \in \mathbb{R}^{m\times n}$, i.e., $(A \circ B)_{ij} = A_{ij}B_{ij}$ for each i, j. The absolute value of a matrix $A = (a_{ij})$ is defined by $|A| = (|a_{ij}|)$. Maximum (or minimum) of two matrices A, B is taken componentwise, i.e., $(\max\{A, B\})_{ij} = \max\{A_{ij}, B_{ij}\}$ for each i, j.

Properties. The following properties are valid whenever the respective operations and inequalities are defined: (i) $A \le B$ and $0 \le C$ imply $AC \le BC$, (ii) $A \le |A|$, (iii) $|A| \le B$ if and only if $-B \le A \le B$, (iv) $|A+B| \le |A|+|B|$, (v) if $A\circ B \ge 0$, then $|A+B| = |A|+|B|$, (vi) if $|A - B| < |B|$, then $A \circ B > 0$, (vii) $||A| - |B|| \le |A - B|$, (viii) $|AB| \le |A||B|$.

Notation for vectors. The same notations and results also apply to vectors which are always considered one-column matrices. Hence, for $a = (a_i)$ and $b = (b_i)$, $a^Tb = \sum_i a_ib_i$ is the scalar product whereas ab^T is the matrix (a_ib_j).

Notation. I denotes the unit matrix, e_j is the jth column of I, $e = (1, \ldots, 1)^T$ is the vector of all ones and $E = ee^T \in \mathbb{R}^{m\times n}$ is the matrix of all ones (in these cases we do not designate explicitly the dimension which can always be inferred from the context).

1.1.2 Specific notations

Notation specific for this text. Throughout the text, important role is played by the set Y_m of all ±1 vectors in $\mathbb{R}^m$, i.e., $Y_m = \{y \in \mathbb{R}^m\,;\ |y| = e\}$. Obviously, the cardinality of Y_m is 2^m. For each $x \in \mathbb{R}^m$ we define its sign vector $\operatorname{sgn} x$ by

$$(\operatorname{sgn} x)_i = \begin{cases} 1 & \text{if } x_i \ge 0, \\ -1 & \text{if } x_i < 0 \end{cases} \qquad (i = 1, \ldots, m),$$

so that $\operatorname{sgn} x \in Y_m$. For a given vector $y \in \mathbb{R}^m$ we denote

$$T_y = \begin{pmatrix} y_1 & 0 & \ldots & 0 \\ 0 & y_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & y_m \end{pmatrix}. \tag{1.1}$$

With a few exceptions we use the notation T_y for vectors $y \in Y_m$ only, in which case we have $T_{-y} = -T_y$, $T_y^{-1} = T_y$ and $|T_y| = I$. For each $x \in \mathbb{R}^m$ we can write $|x| = T_zx$, where $z = \operatorname{sgn} x$; in the proofs[1] this trick is often used to remove the absolute value of a vector. Notice that $T_zx = (z_ix_i)_{i=1}^m = z \circ x$.

All notations are summed up on pp. 9-10.

[1]Omitted here.

1.2 Summary: Linear algebraic notations

A	matrix
$A_{i\bullet}$	the ith row of A
$A_{\bullet j}$	the jth column of A
A^{-1}	inverse matrix
A^{+}	the Moore-Penrose inverse of A
A^T	transpose of A
$\\|A\\|_{\infty,1}$	$= \max_{\\|x\\|_\infty=1} \\|Ax\\|_1$
$A \le B$	$A_{ij} \le B_{ij}$ for each i, j
$A < B$	$A_{ij} < B_{ij}$ for each i, j
$A \ge B$	$\Leftrightarrow B \le A$
$A > B$	$\Leftrightarrow B < A$
$A \circ B$	$= (a_{ij}b_{ij})$ for $A = (a_{ij})$, $B = (b_{ij})$ (Hadamard product)
a	column vector
a^Tb	$= \sum_i a_ib_i$ (scalar product)
ab^T	outer product ($(ab^T)_{ij} = a_ib_j$ for each i, j)
$\operatorname{Conv} X$	the convex hull of X
$\det A$	determinant of A
E	$= ee^T \in \mathbb{R}^{m\times n}$ (the matrix of all ones)
e	$= (1, 1, \ldots, 1)^T$
e_j	the jth column of the unit matrix I
I	unit (or identity) matrix
$\lambda_i(A)$	the ith eigenvalue of a symmetric A ($\lambda_1(A) \ge \ldots \ge \lambda_n(A)$)
$\max\{A, B\}$	componentwise maximum of matrices (vectors)
$\min\{A, B\}$	componentwise minimum of matrices (vectors)
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^{m\times n}$	the set of $m \times n$ real matrices
$\mathbb{R}^n$	real vector space
$\varrho(A)$	spectral radius of A

1.3 Summary: Specific notations

Notations marked in red are important and occur frequently.

$\mathbf{A}$	interval matrix
$\lvert A\rvert$	absolute value of a matrix ($\lvert A\rvert = (\lvert a_{ij}\rvert)$ for $A = (a_{ij})$)
$\underline{A}$	lower bound of an interval matrix $\mathbf{A} = [\underline{A}, \overline{A}]$
$\overline{A}$	upper bound of an interval matrix $\mathbf{A} = [\underline{A}, \overline{A}]$
A_c	midpoint matrix of an interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$
$\mathbf{A}_s$	$= [(\underline{A} + \underline{A}^T)/2, (\overline{A} + \overline{A}^T)/2]$ for $\mathbf{A} = [\underline{A}, \overline{A}]$ (symmetrization)
A_{yz}	$= A_c - T_y \Delta T_z$
A_{-yz}	$= A_{-y,z}$
$\frac{a}{0}$	$= 0$ for $a = 0$, $= \infty$ for $a > 0$ (case $a < 0$ does not occur)
$\mathbf{b}$	interval vector
$\underline{b}$	lower bound of an interval vector $\mathbf{b} = [\underline{b}, \overline{b}]$
$\overline{b}$	upper bound of an interval vector $\mathbf{b} = [\underline{b}, \overline{b}]$
b_c	midpoint vector of an interval vector $\mathbf{b} = [b_c - \delta, b_c + \delta]$
b_y	$= b_c + T_y \Delta$
δ	radius vector of an interval vector $\mathbf{b} = [b_c - \delta, b_c + \delta]$
Δ	radius matrix of an interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$
$f(A, b, c)$	optimal value of a linear programming problem
$\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})$	lower bound of the range of the optimal value of an interval linear programming problem
$\overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})$	upper bound of the range of the optimal value of an interval linear programming problem
$\varrho_0(A)$	real spectral radius of A (maximum of moduli of *real* eigenvalues) $= 0$ if no real eigenvalue exists
$\mathbb{R}^n_z$	$= \{x \in \mathbb{R}^n ; T_z x \geq 0\}$ (z-orthant, $z \in Y_n$)
$\operatorname{sgn} x$	sign vector of a vector x ($(\operatorname{sgn} x)_i = 1$ if $x_i \geq 0$, $(\operatorname{sgn} x)_i = -1$ otherwise)
T_y	the diagonal matrix with diagonal vector y
X	the solution set of $\mathbf{A}x = \mathbf{b}$
$\lvert x\rvert$	absolute value of a vector ($\lvert x\rvert = (\lvert x_i\rvert)$ for $x = (x_i)$)
$[\underline{x}, \overline{x}]$	the interval hull of the solution set X
$[\underline{\underline{x}}, \overline{\overline{x}}]$	enclosure of X (in particular, that by Hansen-Bliek-Rohn)
Y_m	the set of all ± 1-vectors in $\mathbb{R}^m$

Chapter 2

Auxiliary results

Recommendation. Please, read the Preface (pp. 5-6) first.

Subject. Three auxiliary results (of noninterval character) are presented in this chapter.

2.1 The set Y_n

Definition. Y_n is the set of all ± 1-vectors in $\mathbb{R}^n$ (there are 2^n of them).

Problem. Generate Y_n vector by vector so that each two successive vectors differ in exactly one entry.

Algorithm. See p. 52.

Comment. In the algorithm description, y is the generated vector and z is an auxiliary $(0,1)$-vector used for determining the index k for which y_k should be changed to $-y_k$.

Operation. *For each $n \geq 1$ the algorithm at the output yields the set $Y = Y_n$.*

Special features. This algorithm is employed as a subroutine in exhaustive algorithms that require to perform some operation for all $y \in Y_n$ (see the scheme on p. 51). The set Y_n itself is not constructed, the operation is applied to successively generated vectors.

References. [108].

2.2 The norm $\|A\|_{\infty,1}$

Definition. For $A \in \mathbb{R}^{m\times n}$ we define (see e.g. [35])

$$\|A\|_{\infty,1} = \max_{\|x\|_\infty = 1} \|Ax\|_1.$$

Problem. Compute $\|A\|_{\infty,1}$ for a given A.

Formula. *For each $A \in \mathbb{R}^{m\times n}$ we have*

$$\|A\|_{\infty,1} = \max_{y \in Y_n} \|Ay\|_1.$$

Complexity. *Computing $\|A\|_{\infty,1}$ is NP-hard. Even more, checking whether $\|A\|_{\infty,1} \geq 1$ is NP-complete.*

Algorithm. See p. 53.

Comment. This algorithm uses implicitly the algorithm **ynset** for generating the set Y_n (p. 52). This simplifies computation of the new Ay' from the old Ay. Also, since $\|A(-y)\|_1 = \|Ay\|_1$, only y's with $y_n = 1$ are considered.

Operation. The algorithm computes $\|A\|_{\infty,1}$ in a number of steps exponential in n.

Special features. When studying complexity of interval linear problems, we often encounter this norm (see the survey [105]). Since its computation is NP-hard, the norm forms one of two main tools for establishing NP-hardness of interval linear problems (the second such a tool is a related problem whether $-e \leq Ax \leq e$, $\|x\|_1 \geq 1$ has a solution, see [108]).

References. [107], [105], [108], [35], [25].

2.3 The equation $Ax + B|x| = b$ and the sign accord algorithm

Problem. Given $A, B \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, find a solution to the nonlinear equation

$$Ax + B|x| = b. \tag{2.1}$$

Idea. If we knew the sign vector $z = \operatorname{sgn} x$ of the solution x of (2.1), we could rewrite (2.1) as $(A + BT_z)x = b$ and solve it for x as $x = (A + BT_z)^{-1}b$. However, we know neither x, nor z; but we do know that they should satisfy $T_z x = |x| \geq 0$, i.e., $z_j x_j \geq 0$ for each j (a situation we call a sign accord of z and x). In its kernel form the sign accord algorithm computes the z's and x's repeatedly until a sign accord occurs. A combinatorial argument

```
z = sgn (A^{-1}b);
x = (A + BT_z)^{-1}b;
while z_j x_j < 0 for some j
   k = min{j ; z_j x_j < 0};
   z_k = -z_k;
   x = (A + BT_z)^{-1}b;
end
```

Figure 2.1: The kernel of the sign accord algorithm (p. 54).

is used to prove that in case of regularity of $[A - |B|, A + |B|]$, a sign accord is achieved within a prespecified number of steps, so that crossing this number indicates singularity of $[A - |B|, A + |B|]$ (see p. 17 for regularity and singularity).

Complexity. *The problem of checking whether (2.1) has a solution is NP-complete.*

Algorithm. See p. 54.

Comment. The matrix C in the algorithm description is used for updating x according to the Sherman-Morrison formula.

Operation. *For each $A, B \in \mathbb{R}^{n\times n}$ and each $b \in \mathbb{R}^n$, the sign accord algorithm (p. 54) in a finite number of steps either finds a solution of the equation (2.1), or states singularity of the interval matrix $[A - |B|, A + |B|]$ (and, in certain cases, finds a singular matrix $A_s \in [A - |B|, A + |B|]$).*

Comment. If $[A - |B|, A + |B|]$ is regular, then the algorithm finds a solution of (2.1) which, moreover, is *unique*. In case of singularity the algorithm may state singularity without having found a singular matrix, but such cases are rather rare; in most cases it finds a singular matrix as well.

Special features. The sign accord algorithm is the fundamental building block for construction of other algorithms presented in Chapter 7. (See the scheme on p. 51.)

References. [92].

Chapter 3

Interval matrices

Subject. In this chapter we consider various properties of square $n \times n$ interval matrices. Rectangular interval matrices are handled only in the last Section 3.19.

3.1 Interval matrices: definition and basic notations

Definition. If $\underline{A}$, $\overline{A}$ are two matrices in $\mathbb{R}^{m\times n}$, $\underline{A} \le \overline{A}$, then the set of matrices

$$\mathbf{A} = [\underline{A}, \overline{A}] = \{A\,;\ \underline{A} \le A \le \overline{A}\}$$

is called an interval matrix, and the matrices $\underline{A}$, $\overline{A}$ are called its bounds.

Comment. Hence, if $\underline{A} = (\underline{a}_{ij})$ and $\overline{A} = (\overline{a}_{ij})$, then $\mathbf{A}$ is the set of all matrices $A = (a_{ij})$ satisfying

$$\underline{a}_{ij} \le a_{ij} \le \overline{a}_{ij} \tag{3.1}$$

for $i = 1, \dots, m$, $j = 1, \dots, n$. It is worth noting that each coefficient may attain any value in its interval (3.1) independently of the values taken on by other coefficients. Notice that interval matrices are typeset in boldface letters.

Notation. In many cases it is more advantageous to express the data in terms of the center matrix

$$A_c = \tfrac{1}{2}(\underline{A} + \overline{A}) \tag{3.2}$$

and of the radius matrix

$$\Delta = \tfrac{1}{2}(\overline{A} - \underline{A}), \tag{3.3}$$

which is always nonnegative.

Comment. From (3.2), (3.3) we easily obtain that

$$\underline{A} = A_c - \Delta,$$

$$\overline{A} = A_c + \Delta,$$

so that $\mathbf{A}$ can be given either as $[\underline{A}, \overline{A}]$, or as $[A_c - \Delta, A_c + \Delta]$. In the sequel *we employ both forms and we switch freely between them* according to which one is more useful in the current context.

Matrices A_{yz} (important). Given an $m \times n$ interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$, we define matrices

$$A_{yz} = A_c - T_y \Delta T_z$$

for each $y \in Y_m$ and $z \in Y_n$ (T_y is given by (1.1)).

Explanation. The definition implies that

$$(A_{yz})_{ij} = (A_c)_{ij} - y_i \Delta_{ij} z_j = \begin{cases} \overline{a}_{ij} & \text{if } y_i z_j = -1, \\ \underline{a}_{ij} & \text{if } y_i z_j = 1 \end{cases}$$

($i = 1, \dots, m$, $j = 1, \dots, n$), so that $A_{yz} \in \mathbf{A}$ for each $y \in Y_m$, $z \in Y_n$.

Special features. This finite set of matrices from $\mathbf{A}$ (of cardinality at most 2^{m+n-1} because $A_{yz} = A_{-y,-z}$ for each $y \in Y_m$, $z \in Y_n$; the bound is attained if $\Delta > 0$) plays an important role because it turns out that many problems with interval-valued data can be characterized in terms of these matrices, thereby obtaining finite characterizations of problems involving infinitely many sets of data.

Special cases. We write A_{-yz} instead of $A_{-y,z}$. In particular, we have $A_{-yz} = A_c + T_y \Delta T_z$, $A_{ye} = A_c - T_y \Delta$, $A_{ez} = A_c - \Delta T_z$, $A_{ee} = \underline{A}$ and $A_{-ee} = \overline{A}$.

3.2 Regularity

Definition. A square interval matrix $\mathbf{A}$ is called regular if each $A \in \mathbf{A}$ is nonsingular, and it is said to be singular otherwise (i.e., if it contains a singular matrix).

Problem. Check regularity of $\mathbf{A}$.

Necessary and sufficient conditions. *For a square interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$, the following assertions are equivalent:*

(i) $\mathbf{A}$ *is regular,*
(ii) the inequality $|A_c x| \leq \Delta|x|$ *has only the trivial solution,*
(iii) $(\det A_{yz})(\det A_{y'z'}) > 0$ *for each*[1] $y, z, y', z' \in Y_n$,
(iv) A_c *is nonsingular and*[2] $\max_{y,z \in Y_n} \varrho_0(A_c^{-1} T_y \Delta T_z) < 1$,
(v) for each $z \in Y_n$ *the equation* $QA_c - |Q|\Delta T_z = I$ *has a unique matrix solution* Q_z.

Complexity. *Checking regularity of interval matrices is a co-NP-complete problem.*

Sufficient regularity condition. *An interval matrix* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *is regular if*

$$\varrho(|A_c^{-1}|\Delta) < 1 \tag{3.4}$$

holds.[3]

Comment. The condition (3.4) can be verified in polynomial time since it is equivalent to $(I - |A_c^{-1}|\Delta)^{-1} \geq 0$.

Sufficient singularity condition. *An interval matrix* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *is singular if*

$$\max_j (|A_c^{-1}|\Delta)_{jj} \geq 1$$

holds.

Algorithm. See p. 55.

Comment. The algorithm is based on another principles and employs the procedure **hull** (see p. 62), but at the start it checks the above two sufficient conditions.

Operation. *The algorithm in a finite number of steps checks regularity or singularity of* $\mathbf{A}$.

Special features. Among many properties of regular interval matrices, probably the most important one is the unique solvability of the equation $Ax + B|x| = b$ (p. 14) in conjunction with the sign accord algorithm (p. 54) for finding its solution.

References. [9], [12], [92], [110], [41].

[1] $A_{yz} = A_c - T_y \Delta T_z$, see p. 10
[2] ϱ_0 is the *real* spectral radius, see p. 10.
[3] Interval matrices satisfying (3.4) are called strongly regular.

3.3 Finding a singular matrix

Fact. By definition (p. 17), a singular interval matrix $\mathbf{A}$ contains a singular matrix. The algorithm **regularity** (p. 55) is capable of detecting singularity of $\mathbf{A}$, but it does not find a singular matrix in $\mathbf{A}$.

Problem. Find a singular matrix in a singular interval matrix $\mathbf{A}$.

Idea. By the assertion (iii) on p. 17, singularity of $\mathbf{A}$ is equivalent to existence of $y, z, y', z' \in Y_n$ such that

$$(\det A_{yz})(\det A_{y'z'}) \leq 0. \tag{3.5}$$

Since the ± 1-vectors (y^T, z^T) can be ordered in such a way that each two successive vectors differ in exactly one entry (p. 12), the inequality (3.5) must occur for some ± 1-vectors (y^T, z^T), (y'^T, z'^T) differing in just one entry.

Formulae. *Let (3.5) hold for some ± 1-vectors (y^T, z^T), (y'^T, z'^T) differing in exactly one entry. Then we have:*

(a) if $y'_i \neq y_i$ for some i, then $A_s = A_c - (T_y - 2\tau e_i e_i^T)\Delta T_z$ is a singular matrix in $\mathbf{A}$, where $\tau = -y_i/(2(A_c D)_{ii} - 2)$,
(b) if $z'_j \neq z_j$ for some j, then $A_s = A_c - T_y \Delta(T_z - 2\tau e_j e_j^T)$ is a singular matrix in $\mathbf{A}$, where $\tau = -z_j/(2(DA_c)_{jj} - 2)$.

Algorithm. See p. 56.

Comment. The algorithm successively generates all the ± 1-vectors (y^T, z^T) using implicitly the algorithm **ynset** (pp. 12, 52) as a subroutine. $\det A_{y'z'}$ is evaluated from $\det A_{yz}$ with the help of the Sherman-Morrison determinant formula which also proves that the matrix A_s constructed in (a) or (b) above is singular.

Operation. *The algorithm in a finite number of steps checks regularity or singularity of $\mathbf{A}$ and in the latter case it also constructs a singular matrix $A_s \in \mathbf{A}$.*

Comment. The algorithm is heavily exponential. It is therefore recommended to check first singularity by the algorithm **regularity**, and if singularity is detected, to use the current algorithm for finding a singular matrix.

Special features. The above cases (a), (b) show that if $\mathbf{A}$ is singular, then it contains a singular matrix in a certain "normal form" $A_s = A_c - T_y \Delta T_z$, where all entries of y, z are ± 1 with exception of one which belongs to $[-1, 1]$.

References. [92], [96].

3.4 Q_z matrices

Fact. According to the assertion (v) on p. 17, $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ is regular if and only if for each $z \in Y_n$ the equation

$$QA_c - |Q|\Delta T_z = I \tag{3.6}$$

has a unique matrix solution Q_z.

Problem. Given a regular $\mathbf{A}$, compute Q_z for a given $z \in Y_n$.

Formula. *For each i, $(Q_z)_{i\bullet} = x^T$, where x is the solution of*

$$A_c^T x - T_z \Delta^T |x| = e_i$$

and can be found by the sign accord algorithm (see pp. 14, 54).

Complexity. Unknown.

Algorithm. See p. 57.

Operation. *The algorithm in a finite number of steps either computes a solution to (3.6), or states singularity of* $\mathbf{A}$.

Comment. If $\mathbf{A}$ is regular, then the computed solution of (3.6) is equal to Q_z. But it may happen that the algorithm finds a solution to (3.6) even in case of singularity.

Special features. Matrices Q_z are the main tool for construction of a not-a-priori-exponential algorithm for computing the hull, see p. 62.

References. [92].

3.5 Inverse interval matrix

Definition. For a regular $\mathbf{A}$ we define the inverse interval matrix as $\mathbf{A}^{-1} = [\underline{B}, \overline{B}]$, where

$$\underline{B} = \min\{A^{-1}\,;\, A \in \mathbf{A}\},$$

$$\overline{B} = \max\{A^{-1}\,;\, A \in \mathbf{A}\}$$

(componentwise).

Problem. Given a regular $\mathbf{A}$, compute $\mathbf{A}^{-1}$.

Formulae. *Let $\mathbf{A}$ be regular. Then for its inverse $\mathbf{A}^{-1} = [\underline{B}, \overline{B}]$ we have*

$$\underline{B} = \min_{z \in Y_n} Q_z = \min_{y,z \in Y_n} A_{yz}^{-1},$$

$$\overline{B} = \max_{z \in Y_n} Q_z = \max_{y,z \in Y_n} A_{yz}^{-1}$$

(componentwise).

Complexity. *Computing the inverse interval matrix is NP-hard.*

Algorithm. See p. 58.

Operation. *The algorithm in a finite number of steps either computes $\mathbf{A}^{-1}$, or states singularity of $\mathbf{A}$.*

Special features. As in real numerical analysis, computation of $\mathbf{A}^{-1}$ should be avoided whenever possible. In particular, an interval linear system $\mathbf{A}x = \mathbf{b}$ should *never* be solved as $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

References. [92], [97], [18].

3.6 Enclosure of the inverse interval matrix

Definition. An interval matrix $[\underline{\underline{B}}, \overline{\overline{B}}]$ satisfying $\mathbf{A}^{-1} \subseteq [\underline{\underline{B}}, \overline{\overline{B}}]$ is called an enclosure of the inverse interval matrix.

Problem. Given a regular $\mathbf{A}$, compute an enclosure of its inverse.

Comment. This weakened requirement is a consequence of the NP-hardness of computing the exact interval inverse, see p. 20.

Formula. *Let* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *satisfy*[4] $\varrho(|A_c^{-1}|\Delta) < 1$. *Then we have*

$$\mathbf{A}^{-1} \subseteq [\min\{\underset{\sim}{B}, T_\nu \underset{\sim}{B}\}, \max\{\widetilde{B}, T_\nu \widetilde{B}\}],$$

where

$$\begin{aligned}
M &= (I - |A_c^{-1}|\Delta)^{-1}, \\
\mu &= (M_{11}, \ldots, M_{nn})^T, \\
T_\nu &= (2T_\mu - I)^{-1}, \\
\underset{\sim}{B} &= -M|A_c^{-1}| + T_\mu(A_c^{-1} + |A_c^{-1}|), \\
\widetilde{B} &= M|A_c^{-1}| + T_\mu(A_c^{-1} - |A_c^{-1}|).
\end{aligned}$$

Comment. This is the Hansen-Bliek-Rohn enclosure (p. 40) applied to interval linear systems $\mathbf{A}x = [e_j, e_j]$ for $j = 1, \ldots, n$. It can be used only when $\varrho(|A_c^{-1}|\Delta) < 1$.

Complexity. This enclosure is computed in polynomial time.

Algorithm. Use the above formulae.

Operation. The algorithm in a finite number of steps either computes an enclosure, or fails (due to $\varrho(|A_c^{-1}|\Delta) \geq 1$).

Special features. Computing this enclosure requires inverting two real matrices only (inverting $2T_\mu - I$ is trivial because it is a diagonal matrix).

References. [108], [95], [34], [17].

[4] See p. 17.

3.7 Inverse stability

Definition. A regular interval matrix $\mathbf{A}$ is called inverse stable[5] if $|A^{-1}| > 0$ for each $A \in \mathbf{A}$.

Comment. Due to the continuity of the determinant, this means that for each i, j, either $(A^{-1})_{ij} < 0$ for each $A \in \mathbf{A}$, or $(A^{-1})_{ij} > 0$ for each $A \in \mathbf{A}$. Thus we can also say that inverse stability is equivalent to existence of a matrix Z such that[6] $Z \circ A^{-1} > 0$ for each $A \in \mathbf{A}$.

Problem. Check inverse stability of a regular $\mathbf{A}$.

Necessary and sufficient condition. $\mathbf{A}$ *is inverse stable if and only if there exists a matrix* Z *such that* $Z \circ A_{yz}^{-1} > 0$ *for each* $y, z \in Y_n$.

Complexity. Unknown.

Sufficient condition. *If* $[\underline{\underline{B}}, \overline{\overline{B}}]$ *is an enclosure of the inverse interval matrix (see p. 21) and* $\underline{\underline{B}} \circ \overline{\overline{B}} > 0$*, then* $\mathbf{A}$ *is inverse stable.*

Algorithm. Use the above sufficient condition.

Operation. The algorithm is polynomial-time, but it fails if $\varrho(|A_c^{-1}|\Delta) \geq 1$ or $\underline{\underline{B}} \circ \overline{\overline{B}} \not> 0$.

Special features. *If* $\mathbf{A}$ *is inverse stable, then the coefficients of its inverse* $\mathbf{A}^{-1} = [\underline{B}, \overline{B}]$ *are given by the explicit formulae*

$$
\begin{aligned}
\underline{B}_{ij} &= (A^{-1}_{-y(i),z(j)})_{ij} \\
\overline{B}_{ij} &= (A^{-1}_{y(i)z(j)})_{ij}
\end{aligned}
$$

$(i, j = 1, \ldots, n)$*, where* $y(i) = \operatorname{sgn}(A_c^{-1})_{i\bullet}$ *and* $z(j) = \operatorname{sgn}(A_c^{-1})_{\bullet j}$ *for each* i, j.

References. [92], [97].

[5] Meant: inverse sign stable.

[6] For clarity, Z may be "normalized" to satisfy $|Z| = E$, but it is not necessary. "$\circ$" denotes the Hadamard product, see p. 9.

3.8 Inverse sign pattern

Definition. Let $\mathbf{A}$ be regular. If there exist (fixed) $z, y \in Y_n$ such that $T_z A^{-1} T_y \geq 0$ holds for each $A \in \mathbf{A}$, then $\mathbf{A}$ is said to be of the inverse sign pattern (z, y).

Comment. In other words, for each i, j we have $(A^{-1})_{ij} z_i y_j \geq 0$ for each $A \in \mathbf{A}$, so that $z_i y_j$ prescribes the sign of $(A^{-1})_{ij}$.

Problem. For given $z, y \in Y_n$, check whether $\mathbf{A}$ is of the inverse sign pattern (z, y).

Necessary and sufficient condition. $\mathbf{A}$ *is of the inverse sign pattern* (z, y) *if and only if*

$$T_z A_{yz}^{-1} T_y \geq 0, \tag{3.7}$$

$$T_z A_{-yz}^{-1} T_y \geq 0 \tag{3.8}$$

hold.[7]

Complexity. The problem can be solved in polynomial time.

Algorithm. Check the above two conditions.

Operation. Checking requires inverting two real matrices only.

Special features. This is a generalization of inverse nonnegativity (p. 24). E.g. for $z = y = (1, -1, 1, \ldots, (-1)^{n-1})^T$ we get the "chequer-board" inverse sign pattern, etc.

References. [92], [26].

[7] Which implicitly asserts that the two conditions (3.7), (3.8) imply regularity of $\mathbf{A}$.

3.9 Inverse nonnegativity

Definition. A regular interval matrix $\mathbf{A}$ is called inverse nonnegative if $A^{-1} \geq 0$ for each $A \in \mathbf{A}$.

Problem. Check whether a given $\mathbf{A}$ is inverse nonnegative.

Necessary and sufficient condition. *A square interval matrix* $\mathbf{A} = [\underline{A}, \overline{A}]$ *is inverse nonnegative if and only if* $\underline{A}^{-1} \geq 0$ *and* $\overline{A}^{-1} \geq 0$.[8]

Complexity. The problem can be solved in polynomial time.

Algorithm. Check the above two conditions.

Operation. Two inversions needed.

Special features. *If* $\mathbf{A} = [\underline{A}, \overline{A}]$ *is inverse nonnegative, then* $\mathbf{A}^{-1} = [\overline{A}^{-1}, \underline{A}^{-1}]$.

Comment. In a similar way we may define $\mathbf{A}$ to be inverse positive if $A^{-1} > 0$ for each $A \in \mathbf{A}$. Then $\mathbf{A}$ is inverse positive if and only if $\underline{A}^{-1} > 0$ and $\overline{A}^{-1} > 0$.

References. [56], [90].

[8]Which implicitly asserts that nonnegative invertibility of $\underline{A}$ and $\overline{A}$ implies regularity of $\mathbf{A}$.

3.10 Radius of regularity

Convention. In this section (only) we use the convention $\frac{0}{0} = 0$, $\frac{a}{0} = \infty$ for $a > 0$.

Definition. For a square interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$, the number

$$d(\mathbf{A}) = \inf\{\varepsilon \geq 0\,;\, [A_c - \varepsilon\Delta, A_c + \varepsilon\Delta] \text{ is singular}\} \tag{3.9}$$

is called the radius of regularity[9] of $\mathbf{A}$.

Comment. Hence, $d(\mathbf{A}) \in [0, \infty]$. If $d(\mathbf{A})$ is finite, then the infimum in (3.9) is attained as minimum.

Problem. Given $\mathbf{A}$, compute $d(\mathbf{A})$.

Formulae. *For each square interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ we have*[10]

$$d(\mathbf{A}) = \inf_{x \neq 0} \max_i \frac{|A_c x|_i}{(\Delta|x|)_i} = \frac{1}{\max\limits_{y,z \in Y_n} \varrho_0(A_c^{-1} T_y \Delta T_z)}, \tag{3.10}$$

the second formula assuming nonsingularity of A_c.

Comment. In the first formula in (3.10), "$x \neq 0$" can be replaced by "$\|x\| = 1$" in any vector norm.

Complexity. *Computing $d(\mathbf{A})$ is NP-hard, even in the case*[11] $\Delta = E$.

Bounds. *If A_c is nonsingular, then*

$$\frac{1}{\varrho(|A_c^{-1}|\Delta)} \leq d(\mathbf{A}) \leq \frac{1}{\max\limits_j (|A_c^{-1}|\Delta)_{jj}}. \tag{3.11}$$

Algorithm. Starting from the bounds (3.11) (if finite), use the method of halving the interval in conjunction with the algorithm **regularity**, p. 55.

Comment. In the neighbourhood of $d(\mathbf{A})$ the algorithm is likely to behave exponentially and the computation is likely to be slow.

Special features. $d(\mathbf{A}) = 1/\varrho(|A_c^{-1}|\Delta)$ *if A_c is nonsingular and $T_z A_c^{-1} T_y \geq 0$ holds for some $z, y \in Y_n$.*

Comment. The topic was further investigated in [22], [116], [117], [115], and has found applications in control theory.

References. [79], [80], [91], [22], [116], [117], [115], [3], [5], [16], [19], [23], [83].

[9] Also "radius of nonsingularity", and even "radius of singularity".
[10] From conditions (ii), (iv) on p. 17.
[11] The first NP-hardness result for an interval problem, see [79], [80].

3.11 Real eigenvalues

Definition. A real number[12] λ is called a real eigenvalue of $\mathbf{A}$ if it is a real eigenvalue of some $A \in \mathbf{A}$.

Problem. Check whether a given $\lambda \in \mathbb{R}$ is a real eigenvalue of $\mathbf{A}$.

Necessary and sufficient condition. *A $\lambda \in \mathbb{R}$ is a real eigenvalue of $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ if and only if the interval matrix*

$$[(A_c - \lambda I) - \Delta, (A_c - \lambda I) + \Delta] \tag{3.12}$$

is singular.

Complexity. *The problem is NP-hard. (It is NP-hard even for $\lambda = 0$, see p. 17.)*

Sufficient conditions. *If $\lambda \in \mathbb{R}$ is not an eigenvalue of A_c, then*[13]*:*

(a) if $\max_j(|(A_c - \lambda I)^{-1}|\Delta)_{jj} \geq 1$, then λ is a real eigenvalue of $\mathbf{A}$,
(b) if $\varrho(|(A_c - \lambda I)^{-1}|\Delta) < 1$, then λ is not a real eigenvalue of $\mathbf{A}$.

Algorithm. Check singularity of (3.12) by the algorithm **regularity** (p. 55).

Operation. The algorithm solves the problem in a finite number of steps.

References. [96], [92], [84].

[12] We consider the real eigenproblem only; complex eigenvalues seemingly cannot be handled effectively by our methods.

[13] See p. 17.

3.12 Real eigenvectors

Definition. A real vector x is called a real eigenvector of $\mathbf{A}$ if it is a real eigenvector of some $A \in \mathbf{A}$.

Problem. Check whether a given real vector x is a real eigenvector of $\mathbf{A}$.

Necessary and sufficient condition. *A vector* $0 \neq x \in \mathbb{R}^n$ *is a real eigenvector of* $\mathbf{A}$ *if and only if it satisfies*[14]

$$T_z A_{zz} x x^T T_z \leq T_z x x^T A_{-zz}^T T_z,$$

where $z = \operatorname{sgn} x$*.*

Complexity. The problem can be solved in polynomial time.

Algorithm. Check the above condition.

Special features. While checking real eigenvalues is NP-hard (p. 26), checking real eigenvectors is a polynomial-time problem. This is certainly a surprising and unexpected result. For another kind of such a distinction, see p. 50.

References. [96].

---

[14] $A_{zz} = A_c - T_z \Delta T_z$ and $A_{-zz} = A_c + T_z \Delta T_z$, see p. 10.

3.13 Real eigenpairs

Definition. If $\lambda \in \mathbb{R}$ and $x \in \mathbb{R}^n$, then the pair (λ, x) is called a real eigenpair of $\mathbf{A}$ if it is a real eigenpair of some $A \in \mathbf{A}$.

Problem. Given $\lambda \in \mathbb{R}$ and $x \in \mathbb{R}^n$, check whether (λ, x) is a real eigenpair of $\mathbf{A}$.

Necessary and sufficient condition. *If $\lambda \in \mathbb{R}$ and $0 \neq x \in \mathbb{R}^n$, then (λ, x) is a real eigenpair of $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ if and only if*

$$|(A_c - \lambda I)x| \leq \Delta|x| \tag{3.13}$$

holds.

Complexity. Verification can be performed in polynomial time.

Algorithm. Check the above condition.

Special features. It follows[15] from (3.13) that (λ, x), $x \neq 0$, *is a real eigenpair of* $\mathbf{A}$ *if and only if*

$$\max_{x_i \neq 0} \frac{((T_z A_c T_z - \Delta)|x|)_i}{|x_i|} \leq \lambda \leq \min_{x_j \neq 0} \frac{((T_z A_c T_z + \Delta)|x|)_j}{|x_j|}$$

holds, where $z = \operatorname{sgn} x$. This shows the range of all real eigenvalues λ of $\mathbf{A}$ belonging to the same real eigenvector x.

References. [96].

[15] See [96].

3.14 Eigenvalues of symmetric matrices

Fact. A symmetric matrix $A \in \mathbb{R}^{n\times n}$ has all eigenvalues real. They are (usually) ordered in a nonincreasing sequence as $\lambda_1(A) \geq \ldots \geq \lambda_n(A)$.

Definition. A square interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ is called symmetric if both A_c and Δ are symmetric (so that it may also contain nonsymmetric matrices).

Fact. *If* $\mathbf{A}$ *is symmetric, then for each* $i \in \{1, \ldots, n\}$ *the set*

$$\{\lambda_i(A)\,;\, A \in \mathbf{A},\ A \text{ symmetric}\}$$

is a compact interval. We denote this interval by $[\underline{\lambda}_i(\mathbf{A}), \overline{\lambda}_i(\mathbf{A})]$.

Problem. Given a symmetric $\mathbf{A}$, compute the intervals $[\underline{\lambda}_i(\mathbf{A}), \overline{\lambda}_i(\mathbf{A})]$, $i = 1, \ldots, n$.

Formulae for the extremal eigenvalues. Unfortunately, formulae are available only for the extremal eigenvalues so far[16]: *For each symmetric* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *there holds*[17]

$$\overline{\lambda}_1(\mathbf{A}) = \max_{\|x\|_2=1} (x^T A_c x + |x|^T \Delta |x|) = \max_{z\in Y_n} \lambda_1(A_{-zz}),$$

$$\underline{\lambda}_n(\mathbf{A}) = \min_{\|x\|_2=1} (x^T A_c x - |x|^T \Delta |x|) = \min_{z\in Y_n} \lambda_n(A_{zz}).$$

Complexity. *Computing* $\overline{\lambda}_1(\mathbf{A})$, $\underline{\lambda}_n(\mathbf{A})$ *is NP-hard.*

Reformulation of the problem. Due to the above difficulties, we reformulate the problem as follows: given a symmetric $\mathbf{A}$, compute enclosures of the intervals $[\underline{\lambda}_i(\mathbf{A}), \overline{\lambda}_i(\mathbf{A})]$, $i = 1, \ldots, n$.

Formulae for enclosures. *For a symmetric* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *we have*[18]

$$[\underline{\lambda}_i(\mathbf{A}), \overline{\lambda}_i(\mathbf{A})] \subseteq [\lambda_i(A_c) - \varrho(\Delta), \lambda_i(A_c) + \varrho(\Delta)] \qquad (i = 1, \ldots, n). \tag{3.14}$$

Algorithm. Use the above formulae.

Comment. It is an unpleasant feature that all the intervals in (3.14) have the same radius. But nothing better seems to be available.

Operation. Computing the enclosures requires computation of all the eigenvalues of A_c and of the spectral radius of Δ.

Special features. In particular, *for each eigenvalue* $\lambda_i(A)$ *of each symmetric* $A \in \mathbf{A}$ *there holds*

$$\lambda_n(A_c) - \varrho(\Delta) \leq \lambda_i(A) \leq \lambda_1(A_c) + \varrho(\Delta).$$

These bounds are useful for solving problems formulated in terms of extremal eigenvalues (as e.g. positive (semi)definiteness or Hurwitz stability).

References. [103], [106], [99], [28].

---

[16] As far as known to me.
[17] $A_{zz} = A_c - T_z \Delta T_z$, see p. 10.
[18] Consequence of the Wielandt-Hoffman theorem, see [28].

3.15 Positive semidefiniteness

Definition. A symmetric interval matrix (see p. 29) is said to be positive semidefinite if $x^T Ax \geq 0$ holds for each $A \in \mathbf{A}$ and each x.

Problem. Given a symmetric $\mathbf{A}$, check it for positive semidefiniteness.

Necessary and sufficient conditions. *For a symmetric interval matrix* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$*, the following assertions are equivalent:*

(i) $\mathbf{A}$ *is positive semidefinite,*
(ii) $x^T A_c x - |x|^T \Delta |x| \geq 0$ *for each* x*,*
(iii) each A_{zz}*,* $z \in Y_n$*, is positive semidefinite.*[19]

Complexity. *Checking positive semidefiniteness is NP-hard.*

Sufficient condition. *If*

$$\varrho(\Delta) \leq \lambda_n(A_c),$$

then $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *is positive semidefinite.*

Algorithm. Check the above sufficient condition.

Comment. Employing the necessary and sufficient condition (iii) results in an exponential number of operations and can be hardly recommended.

References. [101], [54].

[19]Each matrix $A_{zz} = A_c - T_z \Delta T_z$, $z \in Y_n$, is symmetric.

3.16 Positive definiteness

Definition. A symmetric interval matrix (see p. 29) is said to be positive definite if $x^T Ax > 0$ holds for each $A \in \mathbf{A}$ and each $x \neq 0$.

Problem. Given a symmetric $\mathbf{A}$, check it for positive definiteness.

Necessary and sufficient conditions. *For a symmetric interval matrix* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$*, the following assertions are equivalent:*

(i) $\mathbf{A}$ *is positive definite,*
(ii) $x^T A_c x - |x|^T \Delta |x| > 0$ *for each* $x \neq 0$*,*
(iii) each A_{zz}*,* $z \in Y_n$*, is positive definite,*[20]
(iv) $\mathbf{A}$ *is regular (see p. 17) and* A_c *is positive definite.*

Complexity. *Checking positive definiteness is NP-hard.*

Sufficient condition. *If*

$$\varrho(\Delta) < \lambda_n(A_c),$$

then $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *is positive definite.*

Algorithm. See p. 59.

Comment. The algorithm is based on the above necessary and sufficient condition (iv), and also employs the sufficient condition.

Operation. The algorithm in a finite number of steps checks positive definiteness of $\mathbf{A}$.

Special features. The connection of positive definiteness with regularity in the above condition (iv) is worth noticing.

References. [101], [99].

[20] Each matrix $A_{zz} = A_c - T_z \Delta T_z$, $z \in Y_n$, is symmetric.

3.17 Hurwitz stability

Definition. A square matrix A is called Hurwitz stable if $\mathrm{Re}\,\lambda < 0$ for each eigenvalue λ of A.

Definition. A square interval matrix $\mathbf{A}$ is called Hurwitz stable if each $A \in \mathbf{A}$ is Hurwitz stable.

Problem. Given $\mathbf{A}$, check it for Hurwitz stability.

A negative result. For a general square interval matrix $\mathbf{A}$, Hurwitz stability of all vertex matrices[21] of $\mathbf{A}$ is not sufficient for Hurwitz stability of $\mathbf{A}$ (it was wrongly stated so in [14], but shown to be erroneous in [43] and independently in [4]). However, such a characterization is possible for symmetric interval matrices.

Necessary and sufficient condition. *A* symmetric *interval matrix* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *is Hurwitz stable if and only if the interval matrix* $[-A_c - \Delta, -A_c + \Delta]$ *is positive definite.*[22]

Complexity. *Checking Hurwitz stability is NP-hard (even for symmetric interval matrices).*

Sufficient condition. *Let* $\mathbf{A} = [\underline{A}, \overline{A}]$ *be a (nonsymmetric) square interval matrix. If the symmetric interval matrix*

$$\mathbf{A}_s = [(\underline{A} + \underline{A}^T)/2, (\overline{A} + \overline{A}^T)/2]$$

is Hurwitz stable, then $\mathbf{A}$ *is Hurwitz stable.* Many other sufficient conditions are surveyed in [60].

Algorithm. See p. 60.

Comment. The algorithm employs both the necessary and sufficient condition and the sufficient condition.

Operation. *If* $\mathbf{A}$ *is symmetric, then the algorithm in a finite number of steps checks Hurwitz stability of* $\mathbf{A}$. It fails to give any result if $\mathbf{A}$ is nonsymmetric and $\mathbf{A}_s$ is not Hurwitz stable.

Special features. All the properties of interval matrices considered in this chapter so far were characterized in terms of the matrices A_{yz}, $y, z \in Y_n$ (see p. 10). Hurwitz stability is the first exception.

References. [101], [99], [14], [43], [4], [60], [68].

---

[21] Vertex matrix of $\mathbf{A} = [\underline{A}, \overline{A}]$ is any matrix A satisfying $A_{ij} \in \{\underline{A}_{ij}, \overline{A}_{ij}\}$ for each i, j; each A_{yz} is a vertex matrix, see p. 16.

[22] See p. 31.

3.18 Schur stability

Definition. A square matrix A is called Schur stable if $\varrho(A) < 1$.

Definition. A symmetric interval matrix $\mathbf{A}$ is called Schur stable if each *symmetric* $A \in \mathbf{A}$ is Schur stable.

Comment. Hence, we do not take into account the nonsymmetric matrices contained in $\mathbf{A}$. The reasons for it are purely technical.

Problem. Given a symmetric $\mathbf{A}$, check it for Schur stability.

Necessary and sufficient condition. *A symmetric interval matrix* $\mathbf{A} = [\underline{A}, \overline{A}]$ *is Schur stable if and only if the symmetric interval matrices* $[\underline{A} - I, \overline{A} - I]$ *and* $[-\overline{A} - I, -\underline{A} - I]$ *are Hurwitz stable.*

Complexity. *Checking Schur stability of symmetric interval matrices is NP-hard.*

Algorithm. See p. 61.

Operation. *The algorithm in a finite number of steps checks Schur stability of a symmetric interval matrix* $\mathbf{A}$.

References. [101], [99].

3.19 Full column rank

Definition. A matrix $A \in \mathbb{R}^{m\times n}$ is said to have full column rank if $\operatorname{rank}(A) = n$ (or, equivalently, if $Ax = 0$ implies $x = 0$).

Definition. An $m \times n$ interval matrix $\mathbf{A}$ is said to have full column rank if each $A \in \mathbf{A}$ has full column rank.

Comment. This is the only property in this chapter formulated for rectangular interval matrices.

Problem. Check whether a given $m \times n$ interval matrix $\mathbf{A}$ has full column rank.

Necessary and sufficient condition. $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *has full column rank if and only if the inequality*

$$|A_c x| \le \Delta |x|$$

has only the trivial solution $x = 0$.

Complexity. *Checking full column rank is NP-hard (it is NP-hard even in the square case).*

Sufficient condition. *Let* A_c *have full column rank and let*

$$\varrho(|(A_c^T A_c)^{-1} A_c^T|\Delta) < 1.$$

Then $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *has full column rank.*

Comment. $(A_c^T A_c)^{-1} A_c^T$ is the Moore-Penrose inverse A_c^+ of A_c.

Algorithm. Check the above sufficient condition.

Special features. For square interval matrices, this notion is equivalent to regularity (see p. 17).

References. [104].

Chapter 4

Interval linear equations (square case)

Subject. In this chapter we consider interval linear equations $\mathbf{A}x = \mathbf{b}$ with a square $n \times n$ interval matrix $\mathbf{A}$.

4.1 Interval vectors: definition and basic notations

Definition. An interval vector is a one-column interval matrix

$$\mathbf{b} = \{b\,;\, \underline{b} \leq b \leq \overline{b}\},$$

where $\underline{b}$, $\overline{b} \in \mathbb{R}^m$, $\underline{b} \leq \overline{b}$.

Notation. We again use the center vector

$$b_c = \tfrac{1}{2}(\underline{b} + \overline{b})$$

and the nonnegative radius vector

$$\delta = \tfrac{1}{2}(\overline{b} - \underline{b}).$$

Comment. We employ both forms $\mathbf{b} = [\underline{b}, \overline{b}] = [b_c - \delta, b_c + \delta]$. Notice that interval matrices and vectors are typeset in boldface letters.

Vectors b_y. For an m-dimensional interval vector $\mathbf{b} = [b_c - \delta, b_c + \delta]$, in analogy with the matrices A_{yz} (p. 16), we define vectors

$$b_y = b_c + T_y \delta$$

for each $y \in Y_m$.

Explanation. Then for each such a y we have

$$(b_y)_i = (b_c)_i + y_i \delta_i = \begin{cases} \underline{b}_i & \text{if } y_i = -1, \\ \overline{b}_i & \text{if } y_i = 1 \end{cases}$$

$(i = 1, \ldots, m)$, so that $b_y \in \mathbf{b}$ for each $y \in Y_m$. In particular, $b_{-e} = \underline{b}$ and $b_e = \overline{b}$. Together with matrices A_{yz}, vectors b_y are used in finite characterizations of interval problems having right-hand sides.

4.2 The solution set

Definition. Given an $n \times n$ interval matrix $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ and an interval n-vector $\mathbf{b} = [b_c - \delta, b_c + \delta]$, the set

$$X = \{x\,;\ Ax = b \text{ for some } A \in \mathbf{A},\ b \in \mathbf{b}\}$$

is called the solution set of the (formally written) interval linear system $\mathbf{A}x = \mathbf{b}$.

Problem. Describe the solution set of $\mathbf{A}x = \mathbf{b}$.

Formula.[1] *We have*

$$X = \{x\,;\ |A_c x - b_c| \le \Delta|x| + \delta\}.$$

Comment. Observe that no assumptions concerning $\mathbf{A}$ or $\mathbf{b}$ are made.

Complexity. Verifying whether a given x belongs to X can be performed in polynomial time.

Special features. The solution set is nonconvex in general, but its intersection with each orthant is a convex polyhedron (possibly empty). If $\mathbf{A}$ is regular, then X is compact and connected [10]; if $\mathbf{A}$ is singular, then *each* component of X is unbounded [40].

References. [77], [71], [10], [40], [108].

[1]The Oettli-Prager theorem [77].

4.3 The hull

Fact. If $\mathbf{A}$ is regular, then the solution set X is compact (p. 37) and therefore bounded.

Definition. If $\mathbf{A}$ is regular, then the interval vector $[\underline{x}, \overline{x}]$ given by

$$\begin{aligned} \underline{x}_i &= \min_{x \in X} x_i, \\ \overline{x}_i &= \max_{x \in X} x_i \qquad (i = 1, \dots, n), \end{aligned}$$

(i.e., the narrowest interval vector containing the solution set X) is called the interval hull[2] of the solution set X.

Problem. Compute the interval hull of the solution set X of an interval linear system $\mathbf{A}x = \mathbf{b}$ with $\mathbf{A}$ regular.

Formulae.[3] *Let Z be any subset of Y_n such that for each $x \in X$ there exists a $z \in Z$ with $T_z x \geq 0$. If for each $z \in Z$ the equations*

$$QA_c - |Q|\Delta T_z = I, \tag{4.1}$$
$$QA_c + |Q|\Delta T_z = I \tag{4.2}$$

have solutions[4] *Q_z and Q_{-z}, respectively, then $\mathbf{A}$ is regular*[5] *and for the interval hull $[\underline{x}, \overline{x}]$ there holds*

$$\begin{aligned} \underline{x} &= \min_{z \in Z} (Q_{-z} b_c - |Q_{-z}|\delta), \\ \overline{x} &= \max_{z \in Z} (Q_z b_c + |Q_z|\delta) \end{aligned}$$

(componentwise).

Comment. The first assumption concerning Z is satisfied e.g. for $Z = Y_n$. The algorithm referenced below attempts to make Z as small as possible. For Q_z matrices, see pp. 19 and 57.

Complexity. *Computing the hull of the solution set is an NP-hard problem.*

Algorithm. See p. 62.

Operation. *The algorithm in a finite number of steps either computes the hull, or states singularity of $\mathbf{A}$.*

Special features. This is a not-a-priori-exponential algorithm. Its number of steps depends on the cardinality of the set Z. For example, if $X \subset (\mathbb{R}^n_z)^\circ$, then $Z = \{z\}$ and only two matrices (Q_z and Q_{-z}) are to be computed, see p. 39; if $0 \in X^\circ$, then $Z = Y_n$ and we must compute 2^n of them (the superscript "$\circ$" denotes the interior).

References. (The algorithm has not been published.) [92], [40], [110], [108], [76], [6], [11], [2], [71].

---

[2] Or simply "hull".

[3] The result is formulated in this somewhat complicated form in order to circumvent the assumption of regularity of $\mathbf{A}$ which is verified (or disproved) on the way.

[4] If (4.1) or (4.2) does not have a solution, then $\mathbf{A}$ is singular, see p. 19.

[5] Which, in turn, guarantees that the solutions Q_z, Q_{-z} of (4.1), (4.2) are unique, see p. 19.

4.4 The solution set lying in a single orthant

Formulae. *Let* $\mathbf{A}$ *be regular. Then* $X \subset (\mathbb{R}^n_z)^\circ$ *holds*[6] *for some* $z \in Y_n$ *if and only if*

$$\begin{aligned} T_z(A_c^{-1}b_c) &> 0, && (4.3)\\ T_z(Q_{-z}b_c - |Q_{-z}|\delta) &> 0, && (4.4)\\ T_z(Q_z b_c + |Q_z|\delta) &> 0. && (4.5) \end{aligned}$$

In this case the hull $[\underline{x}, \overline{x}]$ *is given by*

$$\begin{aligned} \underline{x} &= Q_{-z}b_c - |Q_{-z}|\delta, && (4.6)\\ \overline{x} &= Q_z b_c + |Q_z|\delta. && (4.7) \end{aligned}$$

Algorithm. If (4.3)- (4.5) are satisfied, then the algorithm **hull** (see p. 62) detects this situation and computes the hull directly by (4.6)-(4.7).

Operation. In this case the algorithm requires computing two matrices (Q_z and Q_{-z}) only.

Special features. This is a rare case when the bounds of the hull can be given explicitly by closed-form formulae.

References. (Unpublished.) [6], [11], [90].

[6] $(\mathbb{R}^n_z)^\circ$ is the interior of $\mathbb{R}^n_z$.

4.5 Enclosure of the solution set

Definition. An interval vector $[\underline{\underline{x}}, \overline{\overline{x}}]$ satisfying $X \subseteq [\underline{\underline{x}}, \overline{\overline{x}}]$ is called an enclosure of the solution set X.

Problem. Given an interval linear system $\mathbf{A}x = \mathbf{b}$ with regular $\mathbf{A}$, compute an enclosure of its solution set X.

Comment. This weakened requirement is a consequence of the NP-hardness of computing the interval hull of the solution set, see p. 38.

Formulae. *Let* $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ *satisfy* $\varrho(|A_c^{-1}|\Delta) < 1$ *(see p. 17). Then the interval vector* $[\underline{\underline{x}}, \overline{\overline{x}}]$ *computed by the following formulae is an enclosure*[7] *of the solution set* X*:*

$$
\begin{aligned}
M &= (I - |A_c^{-1}|\Delta)^{-1}, \\
\mu &= (M_{11}, \dots, M_{nn})^T, \\
T_\nu &= (2T_\mu - I)^{-1}, \\
x_c &= A_c^{-1} b_c, \\
x^* &= M(|x_c| + |A_c^{-1}\delta|), \\
\underset{\sim}{x} &= -x^* + T_\mu(x_c + |x_c|), \\
\tilde{x} &= \quad x^* + T_\mu(x_c - |x_c|), \\
\underline{\underline{x}} &= \min\{\underset{\sim}{x}, T_\nu \underset{\sim}{x}\}, \\
\overline{\overline{x}} &= \max\{\tilde{x}, T_\nu \tilde{x}\}.
\end{aligned}
$$

Complexity. This enclosure is computed in polynomial time.

Algorithm. See p. 63, up to the line "*flag* = *'enclosure computed'*;" (the rest of the algorithm is explained on p. 41).

Comment. The algorithm works only under the condition $\varrho(|A_c^{-1}|\Delta) < 1$.

Operation. The algorithm in a finite number of steps either computes an enclosure, or fails.

Special features. *The bounds given by the HBR enclosure are always at least as good as the componentwise Bauer-Skeel bounds*[8]*, and they are better* in each entry *provided* $(|A_c^{-1}|\Delta)_{ii} > 0$ *holds for each* i*.*

Apology. At this place I apologize to all colleagues who have ever written papers on enclosures for not having quoted their results here. Because of the "one-topic-one-page" approach I could choose only one type, and I opted for the HBR enclosure because of its special properties (p. 41).

References. [31], [15], [95], [108], [100], [74], [72]; [8], [130], [131]; [1], [2], [7], [12], [24], [26], [27], [29], [30], [32], [33], [36], [50], [53], [55], [61], [62], [63], [64], [67], [69], [71], [73], [81], [85], [112], [113], [114], [119], [124], [127], [129].

---

[7] Called the Hansen-Bliek-Rohn enclosure (abbreviated as HBR).

[8] For the Bauer-Skeel bounds, see e.g. [131].

4.6 Overestimation of the HBR enclosure

Fact. By definition (see p. 40), any enclosure $[\underline{\underline{x}}, \overline{\overline{x}}]$ satisfies $[\underline{x}, \overline{x}] \subseteq [\underline{\underline{x}}, \overline{\overline{x}}]$, where $[\underline{x}, \overline{x}]$ is the interval hull.

Problem. Determine the overestimation of the enclosure.[9]

Comment. Such information is usually not available. The HBR enclosure was chosen for inclusion here because of possessing this particular property.

Formulae. *Under assumption and notations from p. 40, let* $[\underline{x}, \overline{x}]$ *be the interval hull and* $[\underline{\underline{x}}, \overline{\overline{x}}]$ *the HBR enclosure. Then for each* $i \in \{1, \ldots, n\}$ *we have*

$$\underline{\underline{x}}_i \le \underline{x}_i \le \underline{\underline{x}}_i + \underline{d}_i, \tag{4.8}$$

$$\overline{\overline{x}}_i - \overline{d}_i \le \overline{x}_i \le \overline{\overline{x}}_i, \tag{4.9}$$

where

$$\begin{aligned}
\underline{d}_i &= e_i^T (I - |A_c^{-1} T_{\underline{z}} \Delta|)^{-1} |(T_{\underline{z}} A_c^{-1} T_{\underline{z}} - |A_c^{-1}|)(\underline{\xi}_i \Delta M e_i + \Delta x^* + \delta)|,\\
\overline{d}_i &= e_i^T (I - |A_c^{-1} T_{\overline{z}} \Delta|)^{-1} |(T_{\overline{z}} A_c^{-1} T_{\overline{z}} - |A_c^{-1}|)(\overline{\xi}_i \Delta M e_i + \Delta x^* + \delta)|,\\
\underline{\xi}_i &= (|\underline{\underline{x}}| + \underline{\underline{x}} - x_c - |x_c|)_i,\\
\overline{\xi}_i &= (|\overline{\overline{x}}| - \overline{\overline{x}} + x_c - |x_c|)_i
\end{aligned}$$

and $\underline{z}$, $\overline{z}$ *are given by*

$$\underline{z}_j = \begin{cases} \operatorname{sgn}(x_c)_j & \text{if } j \ne i, \\ -1 & \text{if } j = i, \end{cases}$$

$$\overline{z}_j = \begin{cases} \operatorname{sgn}(x_c)_j & \text{if } j \ne i, \\ 1 & \text{if } j = i \end{cases}$$

$(j = 1, \ldots, n)$.

Comment. Computing $\underline{d}$, $\overline{d}$ requires computation of up to $2n$ inverses (but it usually pays off). If this number is considered too large, the matrices $(I - |A_c^{-1} T_{\underline{z}} \Delta|)^{-1}$, $(I - |A_c^{-1} T_{\overline{z}} \Delta|)^{-1}$ in the formulae for $\underline{d}_i$, $\overline{d}_i$ can be replaced by the matrix M, and the whole theorem will remain in force.

Complexity. Vectors $\underline{d}$, $\overline{d}$ can be computed in polynomial time.

Algorithm. See p. 63, the lines after "*flag* = *'enclosure computed'*;".

Operation. *The algorithm in a finite number of steps computes nonnegative vectors* $\underline{d}$, $\overline{d}$ *satisfying (4.8), (4.9).*

Special features. If A_c is a diagonal matrix with positive diagonal entries, then $T_{\underline{z}} A_c^{-1} T_{\underline{z}} - |A_c^{-1}| = T_{\overline{z}} A_c^{-1} T_{\overline{z}} - |A_c^{-1}| = 0$ and consequently $\underline{d} = \overline{d} = 0$, so that $[\underline{x}, \overline{x}] = [\underline{\underline{x}}, \overline{\overline{x}}]$. Hence, *in this case the HBR enclosure yields the exact interval hull.*

References. (Unpublished.) [31], [15], [95], [74], [72].

[9] Of course, without computing the hull, which is an NP-hard problem.

Chapter 5

Interval linear equations and inequalities (rectangular case)

Subject. In this chapter we consider systems of interval linear equations $\mathbf{A}x = \mathbf{b}$ (or systems of interval linear inequalities $\mathbf{A}x \leq \mathbf{b}$) with a rectangular $m \times n$ interval matrix $\mathbf{A}$.

5.1 (Z, z)-solutions

Intro. Let $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ be an $m \times n$ interval matrix and $\mathbf{b} = [b_c - \delta, b_c + \delta]$ an interval m-vector. Under an interval linear system $\mathbf{A}x = \mathbf{b}$ we understand the family of all systems $Ax = b$ with $A \in \mathbf{A}$, $b \in \mathbf{b}$.

Definition. Let $|Z| = E \in \mathbb{R}^{m\times n}$ and $|z| = e \in \mathbb{R}^m$. A vector $x \in \mathbb{R}^n$ is said to be a (Z, z)-solution of a system $\mathbf{A}x = \mathbf{b}$ if for each $A_{ij} \in [\underline{A}_{ij}, \overline{A}_{ij}]$ with $Z_{ij} = -1$ and for each $b_i \in [\underline{b}_i, \overline{b}_i]$ with $z_i = -1$ there exist $A_{ij} \in [\underline{A}_{ij}, \overline{A}_{ij}]$ with $Z_{ij} = 1$ and $b_i \in [\underline{b}_i, \overline{b}_i]$ with $z_i = 1$ such that $Ax = b$ holds.[1]

Problem. Given Z and z, describe the set of all (Z, z)-solutions of $\mathbf{A}x = \mathbf{b}$.

Comment. Despite the complexity of the definition, it turns out that description of (Z, z)-solutions becomes wonderfully simple as soon as the Hadamard product is employed.

Formula.[2] *A vector $x \in \mathbb{R}^n$ is a (Z, z)-solution of $\mathbf{A}x = \mathbf{b}$ if and only if it satisfies*

$$|A_c x - b_c| \le (Z \circ \Delta)|x| + z \circ \delta. \tag{5.1}$$

Complexity. Complexity of checking whether a system $\mathbf{A}x = \mathbf{b}$ has a (Z, z)-solution depends on the choice of Z and z; see pp. 44 and 45 for two opposite examples.

Algorithm. For verification whether a given x is a (Z, z)-solution of $\mathbf{A}x = \mathbf{b}$, check (5.1).

Special features. This is a generalization of the Oettli-Prager theorem [77], p. 37 (which can be obtained from (5.1) by putting $Z = E$ and $z = e$). Both its formulation and proof were not straightforward. Shary presented his definition of (Z, z)-solutions, which he called "∀∃-solutions", in [125]. His formulation of the result contained interval arithmetic operations. A formula not using these operations and proved from the Oettli-Prager theorem was given in this author's letter to Shary and Lakeyev [102]. The final step towards utmost simplicity by employing the Hadamard product was done by Lakeyev in [57].

References. [125], [57], [102], [77].

[1] Thus "-1" corresponds to "∀" and "1" to "∃". It could be argued that the reverse order would be more natural, but we would have to pay for it by introducing minus signs into the main formula (5.1).

[2] By Shary, Lakeyev and Rohn.

5.2 Tolerance solutions

Definition. A $(-E, e)$-solution (see p. 43) is called a tolerance solution of $\mathbf{A}x = \mathbf{b}$. In other words, x is a tolerance solution if it satisfies

$$\{Ax\,;\ A \in \mathbf{A}\} \subseteq \mathbf{b}.$$

Problem. Describe the set of tolerance solutions of $\mathbf{A}x = \mathbf{b}$.

Formula. *For the set X_{tol} of tolerance solutions of $\mathbf{A}x = \mathbf{b}$ we have*

$$\begin{aligned} X_{\mathrm{tol}} &= \{x\,;\ |A_c x - b_c| \le -\Delta|x| + \delta\} \\ &= \{x_1 - x_2\,;\ \overline{A}x_1 - \underline{A}x_2 \le \overline{b},\ \underline{A}x_1 - \overline{A}x_2 \ge \underline{b},\ x_1 \ge 0,\ x_2 \ge 0\}. \end{aligned} \tag{5.2}$$

Complexity. *Checking whether a system $\mathbf{A}x = \mathbf{b}$ has a tolerance solution can be performed in polynomial time.*

Algorithm. Use a polynomial-time linear programming algorithm to check whether the system of linear inequalities in (5.2) has a solution.

Operation. The algorithm in a finite number of steps checks whether $\mathbf{A}x = \mathbf{b}$ has a tolerance solution (and, in the positive case, also finds such a solution).

Special features. Introduction of the notion of tolerance solutions (as early as in 1970's) was motivated by considerations concerning crane construction [75] and input-output planning with inexact data of the socialist economy of former Czechoslovakia [86].

References. [75], [86], [89], [70], [21], [47], [45], [46], [128], [118], [121], [122], [123], [58].

5.3 Control solutions

Definition. An $(E, -e)$-solution (see p. 43) is called a control solution of $\mathbf{A}x = \mathbf{b}$. In other words, x is a control solution if it satisfies

$$\mathbf{b} \subseteq \{Ax\,;\, A \in \mathbf{A}\}.$$

Problem. Describe the set of control solutions of $\mathbf{A}x = \mathbf{b}$.

Formula. *For the set X_{con} of control solutions of $\mathbf{A}x = \mathbf{b}$ we have*

$$X_{\text{con}} \quad = \quad \{x\,;\, |A_c x - b_c| \leq \Delta|x| - \delta\}.$$

Complexity. *The problem of checking whether a system $\mathbf{A}x = \mathbf{b}$ has a control solution is NP-complete.*

Special features. Control solutions were introduced in [120]. The choice of the word "control" was probably motivated by the fact that each vector $b \in \mathbf{b}$ can be reached by Ax when properly controlling the coefficients of A within $\mathbf{A}$.

References. [120], [123], [126], [58], [127].

5.4 Strong solvability of equations

Definition. Let $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$ be an $m \times n$ interval matrix and $\mathbf{b} = [b_c - \delta, b_c + \delta]$ an interval m-vector. We say that the system $\mathbf{A}x = \mathbf{b}$ is strongly solvable if *each* system $Ax = b$ with $A \in \mathbf{A}$, $b \in \mathbf{b}$ has a solution.

Problem. Check whether a given system $\mathbf{A}x = \mathbf{b}$ is strongly solvable.

Necessary and sufficient condition. *A system $\mathbf{A}x = \mathbf{b}$ is strongly solvable if and only if for each $y \in Y_m$ the system*[3]

$$A_{ye}x^1 - A_{-ye}x^2 = b_y, \tag{5.3}$$

$$x^1 \geq 0,\ x^2 \geq 0 \tag{5.4}$$

has a solution x_y^1, x_y^2. Moreover, if this is the case, then for each $A \in \mathbf{A}$, $b \in \mathbf{b}$ the system $Ax = b$ has a solution in the set $\mathrm{Conv}\{x_y^1 - x_y^2\,;\, y \in Y_m\}$.

Complexity. *The problem of checking strong solvability of interval linear equations is NP-hard.*

Algorithm. See p. 64.

Comment. The algorithm uses a (not specified) polynomial-time linear programming subroutine for solving the system (5.3), (5.4).

Operation. *The algorithm in a finite number of steps checks strong solvability of $\mathbf{A}x = \mathbf{b}$.*

Special features. The proof of the above necessary and sufficient condition is nontrivial and uses a new existence theorem for systems of linear equations [93], [94].

References. [109], [108], [93], [94].

[3] $A_{ye} = A_c - T_y\Delta$, $A_{-ye} = A_c + T_y\Delta$ and $b_y = b_c + T_y\delta$, see p. 10.

5.5 Strong solvability of inequalities

Definition. Let $\mathbf{A} = [\underline{A}, \overline{A}]$ be an $m \times n$ interval matrix and $\mathbf{b} = [\underline{b}, \overline{b}]$ an interval m-vector. We say that the system $\mathbf{A}x \leq \mathbf{b}$ is strongly solvable if *each* system $Ax \leq b$ with $A \in \mathbf{A}$, $b \in \mathbf{b}$ has a solution.

Problem. Check whether a given system $\mathbf{A}x \leq \mathbf{b}$ is strongly solvable.

Necessary and sufficient condition. *A system $\mathbf{A}x \leq \mathbf{b}$ is strongly solvable if and only if the system*

$$\overline{A}x^1 - \underline{A}x^2 \leq \underline{b}, \tag{5.5}$$

$$x^1 \geq 0,\ x^2 \geq 0 \tag{5.6}$$

has a solution.

Complexity. *The problem of checking strong solvability of interval linear inequalities can be solved in polynomial time.*

Algorithm. See p. 65.

Comment. The algorithm uses a (not specified) polynomial-time linear programming subroutine for solving the system (5.5), (5.6).

Operation. The algorithm in a finite number of steps checks strong solvability of $\mathbf{A}x \leq \mathbf{b}$.

Special features. If a system $\mathbf{A}x \leq \mathbf{b}$ is strongly solvable, then all the systems $Ax \leq b$, $A \in \mathbf{A}$, $b \in \mathbf{b}$, have a common solution (a nontrivial fact), which is called a strong solution of $\mathbf{A}x \leq \mathbf{b}$. The algorithm on p. 65 finds a strong solution if it exists. Also, observe the difference: checking strong solvability of interval linear equations is NP-hard, whereas the same problem for interval linear inequalities is solvable in polynomial time.

References. [111], [108].

Chapter 6

Interval linear programming

Subject. This last, and shortest, chapter is dedicated to a single topic, namely the range of the optimal value of an interval linear programming problem.

6.1 Reminder: optimal value of a linear program

Definition. The value[1]

$$f(A, b, c) = \inf\{c^T x\,;\, Ax = b, x \geq 0\}$$

is called the optimal value of a linear program

$$\text{minimize } c^T x$$

subject to

$$Ax = b,\ x \geq 0.$$

Comment. Hence, $f(A, b, c) \in [-\infty, \infty]$.

Problem. Given A, b, c, compute $f(A, b, c)$.

Complexity. *The problem can be solved in polynomial time.*

Algorithm. The first polynomial-time linear programming algorithm was described by Khachiyan in [48]. Many of them exist nowadays; see e.g. [78].

Special features. A polynomial-time linear programming subroutine is implicitly used in the algorithms on pp. 64, 65 and 66.

References. [20], [48], [44], [78].

[1]In linear programming only finite value of $f(A, b, c)$ is accepted as the optimal value; we use this formulation for the sake of utmost generality of the results.

6.2 Range of the optimal value

Definition. Let $\mathbf{A} = [\underline{A}, \overline{A}] = [A_c - \Delta, A_c + \Delta]$ be an $m \times n$ interval matrix and let $\mathbf{b} = [\underline{b}, \overline{b}] = [b_c - \delta, b_c + \delta]$ and $\mathbf{c} = [\underline{c}, \overline{c}]$ be an m-dimensional and n-dimensional interval vector, respectively. The *family* of linear programming problems

$$\min\{c^T x\,;\, Ax = b, x \geq 0\} \tag{6.1}$$

with data satisfying

$$A \in \mathbf{A},\, b \in \mathbf{b},\, c \in \mathbf{c} \tag{6.2}$$

is called an *interval linear programming problem.*

Definition. The interval $[\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}), \overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})]$, where

$$\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}) = \inf\{f(A, b, c)\,;\, A \in \mathbf{A}, b \in \mathbf{b}, c \in \mathbf{c}\},$$

$$\overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}) = \sup\{f(A, b, c)\,;\, A \in \mathbf{A}, b \in \mathbf{b}, c \in \mathbf{c}\},$$

is called the range of the optimal value of the interval linear programming problem (6.1), (6.2).

Comment. The endpoints of $[\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}), \overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})]$ may be $\pm\infty$.

Problem. Given $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$, compute $[\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}), \overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})]$.

Formula. *We have*

$$\begin{aligned} \underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}) &= \inf\{\underline{c}^T x\,;\, \underline{A}x \leq \overline{b},\, \overline{A}x \geq \underline{b},\, x \geq 0\}, \\ \overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}) &= \sup_{y \in Y_m} f(A_{ye}, b_y, \overline{c}). \end{aligned} \tag{6.3}$$

Comment. Hence, solving only one linear programming problem is needed to evaluate $\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})$, whereas up to 2^m of them are to be solved to compute $\overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})$ according to (6.3). Although the set Y_m is finite, we use "sup" here because some of the values may be infinite. Notice the absence of any assumptions: the result is fully general.

Complexity. *Computing $\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})$ can be performed in polynomial time, whereas computation of $\overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})$ is NP-hard.*

Algorithm. See p. 66.

Operation. *The algorithm computes the range of the optimal value in a finite number of steps.*

Special features. *If $\underline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c})$ is finite, then*

$$\overline{f}(\mathbf{A}, \mathbf{b}, \mathbf{c}) = \sup\{b_c^T p + \delta^T |p|\,;\, A_c^T p - \Delta^T |p| \leq \overline{c}\},$$

so that in this case the upper bound can be computed by solving one nonlinear programming problem.

References. [87], [88], [108], [59], [52], [13], [65], [66], [37], [38], [39], [42], [98], [88], [49], [51], [67], [82], [132].

Chapter 7

Algorithms

Subject. Here we give MATLAB-like descriptions of fifteen basic algorithms that have been referred to in the previous chapters.

Scheme. The following scheme demonstrates the interdependence of the algorithms ($\mathbf{a} \to \mathbf{b}$ means that the algorithm **a** is used as a subroutine in the algorithm **b**). It explains the central role played by the algorithms **ynset**, **signaccord** and **hull**.

$$\begin{tikzcd}
\textbf{norminfone} & & \\
\textbf{ynset} \arrow[u] \arrow[r] \arrow[d] & \textbf{strosolveq} & \\
\textbf{range} & \textit{linear programming} \arrow[l] \arrow[u] \arrow[r] & \textbf{strosolvin}
\end{tikzcd}$$

$$\begin{tikzcd}
\textbf{signaccord} \arrow[r] & \textbf{qzmatrix} \arrow[r] & \textbf{hull} \arrow[r] \arrow[d] & \textbf{inverse} & \\
& & \textbf{regularity} \arrow[r] & \textbf{posdefness} \arrow[r] & \textbf{hurwitzstab} \arrow[d] \\
& & & & \textbf{schurstab}
\end{tikzcd}$$

The algorithms **singular** and **hbr** do not use subroutines.

Algorithm description. For algorithm form, see p. 6. In particular, [ ] denotes the empty matrix or vector (which is not used in linear algebra, but is a useful programming tool); it is assigned to matrices or vectors that have not been computed.

7.1 An algorithm for generating the set Y_n

function $Y = \textbf{ynset}(n)$
$z = 0 \in \mathbb{R}^n$; $y = e \in \mathbb{R}^n$; $Y = \{y\}$;
while $z \neq e$
    $k = \min\{i\,;\, z_i = 0\}$;
    for $i = 1 : k-1,\ z_i = 0$; **end**
    $z_k = 1$;
    $y_k = -y_k$;
    $Y = Y \cup \{y\}$;
end

Figure 7.1: An algorithm for generating the set Y_n (p. 12).

7.2 An algorithm for computing the norm $\|A\|_{\infty,1}$

$$
\begin{array}{l}
\textbf{function } \nu = \textbf{norminfone}\,(A) \\
\quad y = e \in \mathbb{R}^n;\ z = 0 \in \mathbb{R}^{n-1}; \\
\quad x = Ay; \\
\quad \nu = \|x\|_1; \\
\quad \textbf{while } z \neq e \\
\qquad k = \min\{i\,;\ z_i = 0\}; \\
\qquad x = x - 2y_k A_{\bullet k}; \\
\qquad \nu = \max\{\nu, \|x\|_1\}; \\
\qquad \textbf{for}\ \ i = 1 : k-1,\ z_i = 0;\ \textbf{end} \\
\qquad z_k = 1; \\
\qquad y_k = -y_k; \\
\quad \textbf{end}
\end{array}
$$

Figure 7.2: An algorithm for computing the norm $\|A\|_{\infty,1}$ (p. 13).

7.3 The sign accord algorithm

function $[x, flag, A_s] =$ **signaccord** (A, B, b)
% Finds a solution to $Ax + B|x| = b$ or states
% singularity of $[A - |B|, A + |B|]$.
$x = [\,]$; $flag =$ *'singular'*; $A_s = [\,]$;
if A is singular, $A_s = A$; **return**, **end**
$p = 0 \in \mathbb{R}^n$;
$x = A^{-1}b$;
$z = \operatorname{sgn} x$;
if $A + BT_z$ is singular, $A_s = A + BT_z$; $x = [\,]$; **return**, **end**
$x = (A + BT_z)^{-1}b$;
$C = -(A + BT_z)^{-1}B$;
while $z_j x_j < 0$ for some j
    $k = \min\{j\,;\, z_j x_j < 0\}$;
    if $1 + 2z_k C_{kk} \le 0$
        $\tau = (-1)/(2z_k C_{kk})$;
        $A_s = A + B(T_z - 2\tau z_k e_k e_k^T)$;
        $x = [\,]$;
        return
    end
    $p_k = p_k + 1$;
    $z_k = -z_k$;
    if $\log_2 p_k > n - k$, $x = [\,]$; **return**, **end**
    $\alpha = 2z_k/(1 - 2z_k C_{kk})$;
    $x = x + \alpha x_k C_{\bullet k}$;
    $C = C + \alpha C_{\bullet k} C_{k \bullet}$;
end
$flag =$ *'solution found'*;

Figure 7.3: The sign accord algorithm (p. 14).

Comment. After each updating of x and C there holds $x = (A + BT_z)^{-1}b$ and $C = -(A + BT_z)^{-1}B$ for the current z. The variable p_k registers the number of occurrences of k; if $p_k > 2^{n-k}$ for some k, then $[A - |B|, A + |B|]$ is singular (see [92]).

7.4 An algorithm for checking regularity

function $flag = $ **regularity** $(\mathbf{A})$
if A_c is singular, $flag =$ *'singular'*; **return**, **end**
$R = A_c^{-1}$;
if $\varrho(|R|\Delta) < 1$, $flag =$ *'regular'*; **return**, **end**
if $\max_j (|R|\Delta)_{jj} \geq 1$, $flag =$ *'singular'*; **return**, **end**
$b = e$; $\gamma = \min_k |Rb|_k$;
for $i = 1 : n$
  for $j = 1 : n$
    $b' = b$; $b'_j = -b'_j$;
    if $\min_k |Rb'|_k > \gamma$, $\gamma = \min_k |Rb'|_k$; $b = b'$; **end**
  end
end
$[\underline{x}, \overline{x}, flag] =$ **hull** $(\mathbf{A}, [b, b])$;
if $flag =$ *'hull computed'*, $flag =$ *'regular'*; **return**
end

Figure 7.4: An algorithm for checking regularity (p. 17).

Comment. Both the **for** loops may be omitted without affecting functioning of the algorithm. They form only an empirical tool [41] aimed at diminishing the number of orthants to be visited by the subroutine **hull**.

7.5 An algorithm for finding a singular matrix

$$
\begin{array}{l}
\textbf{function } [flag, A_s] = \textbf{singular}\,(\mathbf{A})\\
flag = \text{'}singular\text{'};\ A_s = [\,];\\
\textbf{if } A_c \text{ is singular, } A_s = A_c;\ \textbf{return}, \textbf{end}\\
y = e \in \mathbb{R}^n;\ z = e \in \mathbb{R}^n;\ t = 0 \in \mathbb{R}^{2n-1};\\
\textbf{if } \underline{A} \text{ is singular, } A_s = \underline{A};\ \textbf{return}, \textbf{end}\\
D = \underline{A}^{-1};\\
\textbf{while } t \neq e\\
\quad k = \min\{i\,;\, t_i = 0\};\\
\quad \textbf{for } i = 1 : k-1,\ t_i = 0;\ \textbf{end}\\
\quad t_k = 1;\\
\quad \textbf{if } k \leq n\\
\qquad i = k;\ p = e_i^T A_c D - e_i^T;\\
\qquad \textbf{if } 2p_i + 1 \leq 0\\
\qquad\quad \tau = -y_i/(2p_i);\\
\qquad\quad A_s = A_c - (T_y - 2\tau e_i e_i^T)\Delta T_z;\ \textbf{return}\\
\qquad \textbf{end}\\
\qquad \alpha = 2/(2p_i + 1);\\
\qquad D = D - \alpha D e_i p;\\
\qquad y_i = -y_i;\\
\quad \textbf{else}\\
\qquad j = k - n;\ p = D A_c e_j - e_j;\\
\qquad \textbf{if } 2p_j + 1 \leq 0\\
\qquad\quad \tau = -z_j/(2p_j);\\
\qquad\quad A_s = A_c - T_y \Delta (T_z - 2\tau e_j e_j^T);\ \textbf{return}\\
\qquad \textbf{end}\\
\qquad \alpha = 2/(2p_j + 1);\\
\qquad D = D - \alpha p e_j^T D;\\
\qquad z_j = -z_j;\\
\quad \textbf{end}\\
\textbf{end}\\
flag = \text{'}regular\text{'};
\end{array}
$$

Figure 7.5: An algorithm for finding a singular matrix (p. 18).

Comment. After each updating of y or z there holds $D = A_{yz}^{-1}$.

7.6 An algorithm for computing Q_z

$$
\begin{array}{|l|}
\hline
\textbf{function } [Q_z, flag] = \textbf{qzmatrix}\,(\mathbf{A}, z) \\
\textbf{for } i = 1 : n \\
\quad [x, flag] = \textbf{signaccord}\,(A_c^T, -T_z\Delta^T, e_i); \\
\quad \textbf{if } flag = \text{'}singular\text{'},\ Q_z = [\,];\ \textbf{return} \\
\quad \textbf{end} \\
\quad (Q_z)_{i\bullet} = x^T; \\
\textbf{end} \\
flag = \text{'}solution\ computed\text{'}; \\
\hline
\end{array}
$$

Figure 7.6: An algorithm for computing Q_z (p. 19).

7.7 An algorithm for computing the inverse

function $[\underline{B}, \overline{B}, flag] = \mathbf{inverse}\,(\mathbf{A})$
for $j = 1 : n$
    $[\underline{x}, \overline{x}, flag] = \mathbf{hull}\,(\mathbf{A}, [e_j, e_j]);$
    if $flag =$ '*singular*', $\underline{B} = [\,]; \overline{B} = [\,];$ **return**
    end
    $\underline{B}_{\bullet j} = \underline{x}; \overline{B}_{\bullet j} = \overline{x};$
end
$flag =$ '*inverse computed*';

Figure 7.7: An algorithm for computing the inverse (p. 20).

7.8 An algorithm for checking positive definiteness

function $flag =$ **posdefness** $(\mathbf{A})$
if A_c is not positive definite
  $flag = 'not\ positive\ definite'$; **return**
end
if $\lambda_{\min}(A_c) > \varrho(\Delta)$
  $flag = 'positive\ definite'$; **return**
end
$flag =$ **regularity** $(\mathbf{A})$;
if $flag = 'regular'$, $flag = 'positive\ definite'$; **return**
else $flag = 'not\ positive\ definite'$; **return**
end

Figure 7.8: An algorithm for checking positive definiteness (p. 31).

7.9 An algorithm for checking Hurwitz stability

```
function flag = hurwitzstab (A)
 A'_c = (A_c + A_c^T)/2; Δ' = (Δ + Δ^T)/2;
 flag = posdefness ([−A'_c − Δ', −A'_c + Δ']);
 if flag = 'positive definite'
    flag = 'Hurwitz stable'; return
 else
    if (A'_c = A_c and Δ' = Δ)
       flag = 'not Hurwitz stable'; return
    else
       flag = 'Hurwitz stability not verified'; return
    end
 end
```

Figure 7.9: An algorithm for checking Hurwitz stability (p. 32).

7.10 An algorithm for checking Schur stability

function $flag = \textbf{schurstab}\,(\mathbf{A})$
if $(A_c^T \neq A_c$ **or** $\Delta^T \neq \Delta)$
    $flag =$ '*Schur stability not verified*'; **return**
end
$flag = \textbf{hurwitzstab}\,([A_c - I - \Delta, A_c - I + \Delta])$;
if $flag =$ '*not Hurwitz stable*'
    $flag =$ '*not Schur stable*'; **return**
end
$flag = \textbf{hurwitzstab}\,([-A_c - I - \Delta, -A_c - I + \Delta])$;
if $flag =$ '*not Hurwitz stable*'
    $flag =$ '*not Schur stable*'; **return**
end
$flag =$ '*Schur stable*';

Figure 7.10: An algorithm for checking Schur stability (p. 33).

7.11 An algorithm for computing the hull

function $[\underline{x}, \overline{x}, flag] = \mathbf{hull}\,(\mathbf{A}, \mathbf{b})$
  if A_c is singular
    $\underline{x} = [\,];\ \overline{x} = [\,];\ flag = \text{'}singular\text{'};$ **return**
  end
  $\underline{x} = A_c^{-1} b_c;\ \overline{x} = \underline{x};$
  $z = \operatorname{sgn} \underline{x};\ Z = \{z\};\ D = \emptyset;$
  while $Z \neq \emptyset$
    select $z \in Z;\ Z = Z - \{z\};\ D = D \cup \{z\};$
    $[Q_{-z}, flag] = \mathbf{qzmatrix}(\mathbf{A}, -z);$
    if $flag = \text{'}singular\text{'}$, $\underline{x} = [\,];\ \overline{x} = [\,];$ **return**, **end**
    $\underset{\sim}{x} = Q_{-z} b_c - |Q_{-z}|\delta;$
    $[Q_z, flag] = \mathbf{qzmatrix}(\mathbf{A}, z);$
    if $flag = \text{'}singular\text{'}$, $\underline{x} = [\,];\ \overline{x} = [\,];$ **return**, **end**
    $\tilde{x} = Q_z b_c + |Q_z|\delta;$
    if $\underset{\sim}{x} \leq \tilde{x}$
      $\underline{x} = \min\{\underline{x}, \underset{\sim}{x}\};$
      $\overline{x} = \max\{\overline{x}, \tilde{x}\};$
      for $j = 1 : n$
        $z' = z;\ z'_j = -z'_j;$
        if $(\underset{\sim}{x}_j \tilde{x}_j \leq 0$ **and** $z' \notin Z \cup D),\ Z = Z \cup \{z'\};$ **end**
      end
    end
  end
$flag = \text{'}hull\ computed\text{'};$

Figure 7.11: An algorithm for computing the hull (p. 38).

Comment. Z is the set of sign vectors of orthants to be visited; D is the set of those that already have been visited.

7.12 The Hansen-Bliek-Rohn enclosure algorithm

function $[\underline{\underline{x}}, \overline{\overline{x}}, \underline{d}, \overline{d}, flag] = \mathbf{hbr}\,(\mathbf{A}, \mathbf{b})$
if (A_c is singular **or** $I - |A_c^{-1}|\Delta$ is singular **or** $(I - |A_c^{-1}|\Delta)^{-1} \not\geq I$)
  $\underline{\underline{x}} = [\,]$; $\overline{\overline{x}} = [\,]$; $\underline{d} = [\,]$; $\overline{d} = [\,]$; $flag =$ *'enclosure not computed'*;
  return
end
$M = (I - |A_c^{-1}|\Delta)^{-1}$;
$\mu = (M_{11}, \dots, M_{nn})^T$;
$T_\nu = (2T_\mu - I)^{-1}$;
$x_c = A_c^{-1} b_c$;
$x^* = M(|x_c| + |A_c^{-1}\delta|)$;
$\underset{\sim}{x} = -x^* + T_\mu(x_c + |x_c|)$;
$\tilde{x} = \quad x^* + T_\mu(x_c - |x_c|)$;
$\overline{\overline{x}} = \max\{\tilde{x}, T_\nu \tilde{x}\}$;
$\underline{\underline{x}} = \min\{\underset{\sim}{x}, T_\nu \underset{\sim}{x}\}$;
$flag =$ *'enclosure computed'*;
$z = \operatorname{sgn} x_c$;
$\overline{\xi} = |\overline{\overline{x}}| - \overline{\overline{x}} + x_c - |x_c|$;
$\underline{\xi} = |\underline{\underline{x}}| + \underline{\underline{x}} - x_c - |x_c|$;
for $i = 1 : n$
  $z' = z$; $z'_i = -1$; $N = (I - |A_c^{-1} T_{z'} \Delta|)^{-1}$;
  $\underline{d}_i = (N|(T_{z'} A_c^{-1} T_{z'} - |A_c^{-1}|)(\underline{\xi}_i \Delta M e_i + \Delta x^* + \delta)|)_i$;
  $z'_i = 1$; $N = (I - |A_c^{-1} T_{z'} \Delta|)^{-1}$;
  $\overline{d}_i = (N|(T_{z'} A_c^{-1} T_{z'} - |A_c^{-1}|)(\overline{\xi}_i \Delta M e_i + \Delta x^* + \delta)|)_i$;
end

Figure 7.12: The Hansen-Bliek-Rohn enclosure algorithm (pp. 40, 41).

7.13 An algorithm for checking strong solvability of equations

At the start of the algorithm the equations of the system $\mathbf{A}x = \mathbf{b}$ should be reordered in such a way that the matrix $(\Delta\ \delta)$ has first q rows nonzero ($0 \le q \le m$) and the remaining $m - q$ rows zero.

function $flag = \mathbf{strosolveq}(\mathbf{A}, \mathbf{b})$
reorder the equations;
$z = 0 \in \mathbb{R}^q$; $y = e \in \mathbb{R}^q$; $flag =$ *'strongly solvable'*;
$A = \underline{A}$; $B = \overline{A}$; $b = \overline{b}$;
if $Ax^1 - Bx^2 = b$, $x^1 \ge 0$, $x^2 \ge 0$ is not solvable
    $flag =$ *'not strongly solvable'*; **return**
end
while $z \ne e$
    $k = \min\{i\,;\, z_i = 0\}$;
    for $i = 1 : k - 1$, $z_i = 0$; **end**
    $z_k = 1$;
    $y_k = -y_k$;
    if $y_k = 1$
        $A_{k\bullet} = \underline{A}_{k\bullet}$; $B_{k\bullet} = \overline{A}_{k\bullet}$; $b_k = \overline{b}_k$;
    else
        $A_{k\bullet} = \overline{A}_{k\bullet}$; $B_{k\bullet} = \underline{A}_{k\bullet}$; $b_k = \underline{b}_k$;
    end
    if $Ax^1 - Bx^2 = b$, $x^1 \ge 0$, $x^2 \ge 0$ is not solvable
        $flag =$ *'not strongly solvable'*; **return**
    end
end

Figure 7.13: An algorithm for checking strong solvability of equations (p. 46).

Comment. After each updating of A, B and b there holds $A = A_{ye}$, $B = A_{-ye}$, $b = b_y$ for the current y.

7.14 An algorithm for checking strong solvability of inequalities

function $[x, flag] = \textbf{strosolvin}(\mathbf{A}, \mathbf{b})$
solve the system $\overline{A}x^1 - \underline{A}x^2 \leq \underline{b}$, $x^1 \geq 0$, $x^2 \geq 0$;
if it has a solution x^1, x^2
    $x = x^1 - x^2$; $flag = \text{'}strong\ solution\ found\text{'}$;
else
    $x = [\,]$; $flag = \text{'}not\ strongly\ solvable\text{'}$;
end

Figure 7.14: An algorithm for checking strong solvability of inequalities (p. 47).

7.15 An algorithm for computing the range of the optimal value

At the start of the algorithm the equations of the system $\mathbf{A}x = \mathbf{b}$ should be reordered in such a way that the matrix $(\Delta\ \delta)$ has first q rows nonzero $(0 \le q \le m)$ and the remaining $m - q$ rows zero.

function $[\underline{f}, \overline{f}, flag] = \mathbf{range}(\mathbf{A}, \mathbf{b}, \mathbf{c})$
reorder the equations;
compute $\underline{f} = \inf\{\underline{c}^T x\,;\ \underline{A}x \le \overline{b}, \overline{A}x \ge \underline{b}, x \ge 0\}$;
$z = 0 \in \mathbb{R}^q$; $y = e \in \mathbb{R}^q$;
$A = \underline{A}$; $b = \overline{b}$; $\overline{f} = f(A, b, \overline{c})$;
while $(z \neq e$ **and** $\overline{f} < \infty)$
$k = \min\{i\,;\ z_i = 0\}$;
for $i = 1$ **to** $k - 1$, $z_i = 0$; **end**
$z_k = 1$;
$y_k = -y_k$;
if $y_k = 1$
$A_{k\bullet} = \underline{A}_{k\bullet}$; $b_k = \overline{b}_k$;
else
$A_{k\bullet} = \overline{A}_{k\bullet}$; $b_k = \underline{b}_k$;
end
$\overline{f} = \max\{\overline{f}, f(A, b, \overline{c})\}$;
end
$flag =$ *'range computed'*;

Figure 7.15: An algorithm for computing the range of the optimal value (p. 50).

Comment. After each updating of A and b there holds $A = A_{ye}$, $b = b_y$ for the current y.

Bibliography

[1] J. Albrecht, *Monotone Iterationsfolgen und ihre Verwendung zur Lösung linearer Gleichungssysteme*, Numerische Mathematik, 3 (1961), pp. 345–358. 40

[2] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983. 38, 40

[3] B. R. Barmish, *New Tools for Robustness of Linear Systems*, MacMillan, New York, 1994. 25

[4] B. R. Barmish and C. V. Hollot, *Counter-example to a recent result on the stability of interval matrices by S. Białas*, International Journal of Control, 39 (1984), pp. 1103–1104. 32

[5] I. Bar-On, B. Codenotti and M. Leoncini, *Checking robust nonsingularity of tridiagonal matrices in linear time*, BIT, 36 (1996), pp. 206–220. 25

[6] W. Barth and E. Nuding, *Optimale Lösung von Intervallgleichungssystemen*, Computing, 12 (1974), pp. 117–125. 1, 38, 39

[7] H. Bauch, K.-U. Jahn, D. Oelschlägel, H. Süsse, and V. Wiebigke, *Intervallmathematik*, Teubner, Leipzig, 1987. 40

[8] F. L. Bauer, *Genauigkeitsfragen bei der Lösung linearer Gleichungssysteme*, Zeitschrift für Angewandte Mathematik und Mechanik, 46 (1966), pp. 409–421. 40

[9] M. Baumann, *A regularity criterion for interval matrices*, in Collection of Scientific Papers Honouring Prof. Dr. K. Nickel on Occasion of his 60th Birthday, Part I, J. Garloff et al., eds., Freiburg, 1984, Albert-Ludwigs-Universität, pp. 45–50. 17

[10] H. Beeck, *Charakterisierung der Lösungsmenge von Intervallgleichungssystemen*, Zeitschrift für Angewandte Mathematik und Mechanik, 53 (1973), pp. T181–T182. 37

[11] H. Beeck, *Zur scharfen Aussenabschätzung der Lösungsmenge bei linearen Intervallgleichungssystemen*, Zeitschrift für Angewandte Mathematik und Mechanik, 54 (1974), pp. T208–T209. 38, 39

[12] H. Beeck, *Zur Problematik der Hüllenbestimmung von Intervallgleichungssystemen*, in Interval Mathematics, K. Nickel, ed., Lecture Notes in Computer Science 29, Berlin, 1975, Springer-Verlag, pp. 150–159. 17, 40

[13] H. Beeck, *Linear programming with inexact data*, Technical Report TUM–ISU–7830, Technical University of Munich, Munich, 1978. 50

[14] S. Białas, *A necessary and sufficient condition for the stability of interval matrices*, International Journal of Control, 37 (1983), pp. 717–722. 32

[15] C. Bliek, *Computer Methods for Design Automation*, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, July 1992. 40, 41

[16] R. D. Braatz, P. M. Young, J. C. Doyle and M. Morari, *Computational complexity of mu calculation*, IEEE Transactions on Automatic Control, 39 (1994), pp. 1000–1002. 25

[17] G. F. Corliss, *Industrial applications of interval techniques*, in Computer Arithmetic and Self-Validating Numerical Methods, C. Ullrich, ed., no. 7 in Notes and Reports in Mathematics in Science and Engineering, San Diego, 1990, Academic Press, pp. 91–113. 21

[18] G. E. Coxson, *Computing exact bounds on elements of an inverse interval matrix is NP-hard*, Reliable Computing, 5 (1999), pp. 137–142. 20

[19] G. E. Coxson and C. L. DeMarco, *Computing the real structured singular value is NP-hard*, Report ECE-92-4, Department of Electrical and Computer Engineering, University of Wisconsin, Madison, 1992. 25

[20] G. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, 1963. 49

[21] A. Deif, *Sensitivity Analysis in Linear Systems*, Springer-Verlag, Berlin, 1986. 44

[22] J. W. Demmel, *The componentwise distance to the nearest singular matrix*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 10–19. 25

[23] J. C. Doyle, *Analysis of feedback systems with structured uncertainties*, IEE Proceedings, Part D, 129 (1982), pp. 242–250. 25

[24] A. Frommer and G. Mayer, *A new criterion to guarantee the feasibility of the interval Gaussian algorithm*, SIAM Journal on Matrix Analysis and Applications, 14 (1993), pp. 408–419. 40

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979. 13

[26] J. Garloff, *Totally nonnegative interval matrices*, in Interval Mathematics 1980, K. Nickel, ed., New York, 1980, Academic Press, pp. 317–327. 23, 40

[27] D. Gay, *Solving interval linear equations*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 858–870. 40

[28] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1996. 29

[29] E. Hansen, *On linear algebraic equations with interval coefficients*, in Topics in Interval Analysis, E. Hansen, ed., Oxford, 1969, Oxford University Press, pp. 33–46. 40

[30] E. Hansen, *On the solution of linear algebraic equations with interval coefficients*, Linear Algebra and Its Applications, 2 (1969), pp. 153–165. 40

[31] E. R. Hansen, *Bounding the solution of interval linear equations*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 1493–1503. 40, 41

[32] G. Heindl, *Some inclusion results based on a generalized version of the Oettli-Prager theorem*, Zeitschrift für Angewandte Mathematik und Mechanik, Supplement 3, 76 (1996), pp. 263–266. 40

[33] G. Heindl, *An improvement of the componentwise error estimate corresponding to a well-known inclusion result for solutions of systems of linear equations*, in Recent Advances in Numerical Methods and Applications, O. P. Iliev, ed., Singapore, 1999, World Scientific, pp. 355–361. 40

[34] J. Herzberger and D. Bethke, *On two algorithms for bounding the inverse of an interval matrix*, Interval Computations, 1 (1991), pp. 44–53. 21

[35] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996. 13

[36] K.-U. Jahn, *Eine Theorie der Gleichungssysteme mit Intervallkoeffizienten*, Zeitschrift für Angewandte Mathematik und Mechanik, 54 (1974), pp. 405–412. 40

[37] C. Jansson, *Zur linearen Optimierung mit unscharfen Daten*, PhD thesis, University of Kaiserslautern, 1985. 50

[38] C. Jansson, *A self-validating method for solving linear programming problems with interval input data*, Computing Supplementum, 6 (1988), pp. 33–46. 50

[39] C. Jansson, *On self-validating methods for optimization problems*, in Topics in Validated Computations, J. Herzberger, ed., Amsterdam, 1994, North-Holland, pp. 381–438. 50

[40] C. Jansson, *Calculation of exact bounds for the solution set of linear interval systems*, Linear Algebra and Its Applications, 251 (1997), pp. 321–340. 37, 38

[41] C. Jansson and J. Rohn, *An algorithm for checking regularity of interval matrices*, SIAM Journal on Matrix Analysis and Applications, 20 (1999), pp. 756–776. http://www.cs.cas.cz/~rohn/publist/95.ps 17, 55

[42] C. Jansson and S. M. Rump, *Rigorous solution of linear programming problems with uncertain data*, ZOR-Methods and Models of Operations Research, 35 (1991), pp. 87–111. 50

[43] W. C. Karl, J. P. Greschak, and G. C. Verghese, *Comments on "A necessary and sufficient condition for the stability of interval matrices"*, International Journal of Control, 39 (1984), pp. 849–851. 32

[44] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395. 49

[45] B. Kelling, *Methods of solution of linear tolerance problems with interval arithmetic*, in Computer Arithmetic and Enclosure Methods, L. Atanassova and J. Herzberger, eds., Amsterdam, 1992, North-Holland, pp. 269–277. 44

[46] B. Kelling, *Geometrische Untersuchungen zur eingeschränkten Lösungsmenge linearer Intervallgleichungssysteme*, Zeitschrift für Angewandte Mathematik und Mechanik, 74 (1994), pp. 625–628. 44

[47] B. Kelling and D. Oelschlägel, *Zur Lösung von linearen Toleranzproblemen*, Wissenschaftliche Zeitschrift TH Leuna-Merseburg, 33 (1991), pp. 121–131. 44

[48] L. G. Khachiyan, *A polynomial algorithm in linear programming*, Doklady Akademii Nauk SSSR, 244 (1979), pp. 1093–1096. 49

[49] J. Koníčková, *Optimalizační úlohy s nepřesnými daty*, PhD thesis, Charles University, Prague, 1996. 50

[50] J. Koníčková, *Solution of systems of interval linear equations*, Zeitschrift für Angewandte Mathematik und Mechanik, Supplement 3, 80 (2000), pp. S807–S808. 40

[51] J. Koníčková, *Sufficient condition of basis stability of an interval linear programming problem*, Zeitschrift für Angewandte Mathematik und Mechanik, Supplement 3, 81 (2001), pp. S677–S678. 50

[52] R. Krawczyk, *Fehlerabschätzung bei linearer Optimierung*, in Interval Mathematics, K. Nickel, ed., Lecture Notes in Computer Science 29, Berlin, 1975, Springer-Verlag, pp. 215–222. 50

[53] R. Krawczyk, *Intervalliterationsverfahren*, Bericht 186, Mathematisch-Statistische Sektion im Forschungszentrum Graz, Graz, 1982. 40

[54] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer Academic Publishers, Dordrecht, 1998. 30

[55] I. Kuperman, *Approximate Linear Algebraic Equations and Roundoff Error Estimation*, PhD thesis, University of Witwatersrand, Johannesburg, 1967. 40

[56] J. Kuttler, *A fourth-order finite-difference approximation for the fixed membrane eigenproblem*, Mathematics of Computation, 25 (1971), pp. 237–256. 24

[57] A. V. Lakeyev, *Vychislitel'naya slozhnost' ocenivaniya obobshchennykh mnozhestv resheniy interval'nych lineynykh sistem*, in Trudy XI mezhdunarodnoi Baikalskoi shkoly-seminara "Metody optimizacii i ich prilozheniya", Irkutsk, 1998, pp. 115–118. 43

[58] A. V. Lakeyev and S. I. Noskov, *On the set of solutions of a linear equation with intervally defined operator and right-hand side (in Russian)*, Siberian Mathematical Journal, 35 (1994), pp. 1074–1084. 44, 45

[59] B. Machost, *Numerische Behandlung des Simplexverfahrens mit intervallanalytischen Methoden*, Berichte der GMD, GMD, Bonn, 1970. 50

[60] M. Mansour, *Robust stability of interval matrices*, Proceedings of the 28th Conference on Decision and Control, Tampa, FL, (1989), pp. 46–51. 32

[61] G. Mayer, *Epsilon-inflation in verification algorithms*, Journal of Computational and Applied Mathematics, 60 (1995), pp. 147–169. 40

[62] G. Mayer and J. Rohn, *On the applicability of the interval Gaussian algorithm*, Reliable Computing, 4 (1998), pp. 205–222. http://www.cs.cas.cz/˜rohn/publist/mayer.ps 40

[63] R. E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, 1966. 40

[64] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, 1979. 40

[65] F. Mráz, *Úloha lineárního programování s intervalovými koeficienty.* West Bohemian University, 1993. Habilitationsschrift. 50

[66] F. Mráz, *Calculating the exact bounds of optimal values in LP with interval coefficients*, Annals of Operations Research, 81 (1998), pp. 51–62. 50

[67] J. Nedoma, *Vague matrices in linear programming*, Annals of Operations Research, 47 (1993), pp. 483–496. 40, 50

[68] A. Nemirovskii, *Several NP-hard problems arising in robust stability analysis*, Mathematics of Control, Signals, and Systems, 6 (1993), pp. 99–105. 32

[69] A. Neumaier, *Linear interval equations*, in Interval Mathematics 1985, K. Nickel, ed., Lecture Notes in Computer Science 212, Berlin, 1985, Springer-Verlag, pp. 109–120. 40

[70] A. Neumaier, *Tolerance analysis with interval arithmetic*, Freiburger Intervall-Berichte 86/9, Albert-Ludwigs-Universität, Freiburg, 1986. 44

[71] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990. 37, 38, 40

[72] A. Neumaier, *A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations*, Reliable Computing, 5 (1999), pp. 131–136. 40, 41

[73] K. Nickel, *Die Überschätzung des Wertebereichs einer Funktion in der Intervallrechnung mit Anwendungen auf lineare Gleichungssysteme*, Computing, 18 (1977), pp. 15–36. 40

[74] S. Ning and R. B. Kearfott, *A comparison of some methods for solving linear interval equations*, SIAM Journal on Numerical Analysis, 34 (1997), pp. 1289–1305. 40, 41

[75] E. Nuding and J. Wilhelm, *Über Gleichungen und über Lösungen*, Zeitschrift für Angewandte Mathematik und Mechanik, 52 (1972), pp. T188–T190. 44

[76] W. Oettli, *On the solution set of a linear system with inaccurate coefficients*, SIAM Journal on Numerical Analysis, 2 (1965), pp. 115–118. 38

[77] W. Oettli and W. Prager, *Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides*, Numerische Mathematik, 6 (1964), pp. 405–409. 37, 43

[78] M. Padberg, *Linear Optimization and Extensions*, Springer-Verlag, Berlin, 1999. 49

[79] S. Poljak and J. Rohn, *Radius of nonsingularity*, Research Report, KAM Series 88–117, Faculty of Mathematics and Physics, Charles University, Prague, December 1988. http://www.cs.cas.cz/˜rohn/publist/38.doc 25

[80] S. Poljak and J. Rohn, *Checking robust nonsingularity is NP-hard*, Mathematics of Control, Signals, and Systems, 6 (1993), pp. 1–9. http://www.cs.cas.cz/˜rohn/publist/63.doc 25

[81] K. Reichmann, *Abbruch beim Intervall-Gauss-Algorithmus*, Computing, 22 (1979), pp. 355–361. 40

[82] J. Renegar, *Some perturbation theory for linear programming*, Mathematical Programming, 65 (1994), pp. 73–91. 50

[83] G. Rex and J. Kupferschmidt, *What is the radius of singularity of a real matrix?*, Zeitschrift für Angewandte Mathematik und Mechanik, 81 (2001), pp. 985–986. 25

[84] G. Rex and J. Rohn, *Sufficient conditions for regularity and singularity of interval matrices*, SIAM Journal on Matrix Analysis and Applications, 20 (1999), pp. 437–445. http://www.cs.cas.cz/˜rohn/publist/94.ps 26

[85] H.-G. Rex, *Zur Lösungseinschließung linearer Gleichungssysteme*, Wissenschaftliche Zeitschrift, Technische Hochschule Leipzig, 15 (1991), pp. 441–447. 40

[86] J. Rohn, *Input-output planning with inexact data*, Freiburger Intervall-Berichte 78/9, Albert-Ludwigs-Universität, Freiburg, 1978. http://www.cs.cas.cz/˜rohn/publist/11.doc 44

[87] J. Rohn, *Duality in interval linear programming*, in Interval Mathematics 1980, K. Nickel, ed., New York, 1980, Academic Press, pp. 521–529. http://www.cs.cas.cz/˜rohn/publist/12.doc 50

[88] J. Rohn, *Interval linear systems*, Freiburger Intervall-Berichte 84/7, Albert-Ludwigs-Universität, Freiburg, 1984. http://www.cs.cas.cz/˜rohn/publist/25.doc 50

[89] J. Rohn, *Inner solutions of linear interval systems*, in Interval Mathematics 1985, K. Nickel, ed., Lecture Notes in Computer Science 212, Berlin, 1986, Springer-Verlag, pp. 157–158. http://www.cs.cas.cz/˜rohn/publist/31.doc 44

[90] J. Rohn, *Inverse-positive interval matrices*, Zeitschrift für Angewandte Mathematik und Mechanik, 67 (1987), pp. T492–T493. http://www.cs.cas.cz/˜rohn/publist/32.doc 24, 39

[91] J. Rohn, *Nearness of matrices to singularity*, Research Report, KAM Series 88–79, Faculty of Mathematics and Physics, Charles University, Prague, 1988. http://www.cs.cas.cz/~rohn/publist/37.doc 25

[92] J. Rohn, *Systems of linear interval equations*, Linear Algebra and Its Applications, 126 (1989), pp. 39–78. http://www.cs.cas.cz/~rohn/publist/47.doc 14, 17, 18, 19, 20, 22, 23, 26, 38, 54

[93] J. Rohn, *Characterization of a linear program in standard form by a family of linear programs with inequality constraints*, Ekonomicko-matematický obzor, 26 (1990), pp. 71–73. http://www.cs.cas.cz/~rohn/publist/49.doc 46

[94] J. Rohn, *An existence theorem for systems of linear equations*, Linear and Multilinear Algebra, 29 (1991), pp. 141–144. http://www.cs.cas.cz/~rohn/publist/54.doc 46

[95] J. Rohn, *Cheap and tight bounds: The recent result by E. Hansen can be made more efficient*, Interval Computations, 4 (1993), pp. 13–21. http://www.cs.cas.cz/~rohn/publist/69.ps 21, 40, 41

[96] J. Rohn, *Interval matrices: Singularity and real eigenvalues*, SIAM Journal on Matrix Analysis and Applications, 14 (1993), pp. 82–91. http://www.cs.cas.cz/~rohn/publist/61.doc 18, 26, 27, 28

[97] J. Rohn, *Inverse interval matrix*, SIAM Journal on Numerical Analysis, 30 (1993), pp. 864–870. http://www.cs.cas.cz/~rohn/publist/65.doc 20, 22

[98] J. Rohn, *Stability of the optimal basis of a linear program under uncertainty*, Operations Research Letters, 13 (1993), pp. 9–12. http://www.cs.cas.cz/~rohn/publist/64.doc 50

[99] J. Rohn, *Checking positive definiteness or stability of symmetric interval matrices is NP-hard*, Commentationes Mathematicae Universitatis Carolinae, 35 (1994), pp. 795–797. http://www.cs.cas.cz/~rohn/publist/74.ps 29, 31, 32, 33

[100] J. Rohn, *A perturbation theorem for linear equations*, Commentationes Mathematicae Universitatis Carolinae, 35 (1994), pp. 213–214. http://www.cs.cas.cz/~rohn/publist/77.ps 40

[101] J. Rohn, *Positive definiteness and stability of interval matrices*, SIAM Journal on Matrix Analysis and Applications, 15 (1994), pp. 175–184. http://www.cs.cas.cz/~rohn/publist/68.doc 30, 31, 32, 33

[102] J. Rohn. E-mail letter to S. P. Shary and A. V. Lakeyev of November 18, 1995. http://www.cs.cas.cz/~rohn/publist/LettShaLak.ps 43

[103] J. Rohn, *Checking properties of interval matrices*, Technical Report 686, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, September 1996. http://www.cs.cas.cz/~rohn/publist/92.ps 29

[104] J. Rohn, *Enclosing solutions of overdetermined systems of linear interval equations*, Reliable Computing, 2 (1996), pp. 167–171. http://www.cs.cas.cz/~rohn/publist/88.ps 34

[105] J. Rohn, *Complexity of some linear problems with interval data*, Reliable Computing, 3 (1997), pp. 315–323. http://www.cs.cas.cz/~rohn/publist/96.ps 13

[106] J. Rohn, *Bounds on eigenvalues of interval matrices*, Zeitschrift für Angewandte Mathematik und Mechanik, Supplement 3, 78 (1998), pp. S1049–S1050. http://www.cs.cas.cz/~rohn/publist/97.ps 29

[107] J. Rohn, *Computing the norm* $\|A\|_{\infty,1}$ *is NP-hard*, Linear and Multilinear Algebra, 47 (2000), pp. 195–204. http://www.cs.cas.cz/~rohn/publist/norm.ps 13

[108] J. Rohn, *Systems of interval linear equations and inequalities (rectangular case)*, Technical Report 875, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, September 2002. http://www.cs.cas.cz/~rohn/publist/chapters.ps 12, 13, 21, 37, 38, 40, 46, 47, 50

[109] J. Rohn, *Solvability of systems of linear interval equations*, SIAM Journal on Matrix Analysis and Applications, 25 (2003), pp. 237–245. http://www.cs.cas.cz/~rohn/publist/solvability.pdf 46

[110] J. Rohn and V. Kreinovich, *Computing exact componentwise bounds on solutions of linear systems with interval data is NP-hard*, SIAM Journal on Matrix Analysis and Applications, 16 (1995), pp. 415–420. http://www.cs.cas.cz/~rohn/publist/72.ps 17, 38

[111] J. Rohn and J. Kreslová, *Linear interval inequalities*, Linear and Multilinear Algebra, 38 (1994), pp. 79–82. http://www.cs.cas.cz/~rohn/publist/71.ps 47

[112] S. M. Rump, *Solving algebraic problems with high accuracy*, in A New Approach to Scientific Computation, U. Kulisch and W. Miranker, eds., New York, 1983, Academic Press, pp. 51–120. 40

[113] S. M. Rump, *On the solution of interval linear systems*, Computing, 47 (1992), pp. 337–353. 40

[114] S. M. Rump, *Verification methods for dense and sparse systems of equations*, in Topics in Validated Computations, J. Herzberger, ed., Amsterdam, 1994, North-Holland, pp. 63–135. 40

[115] S. M. Rump, *The distance between regularity and strong regularity*, in Scientific Computing and Validated Numerics, G. Alefeld, A. Frommer and B. Lang, eds., Mathematical Research, Vol. 90, Berlin, 1996, Akademie Verlag, pp. 105–117. 25

[116] S. M. Rump, *Bounds for the componentwise distance to the nearest singular matrix*, SIAM Journal on Matrix Analysis and Applications, 18 (1997), pp. 83–103. 25

[117] S. M. Rump, *Almost sharp bounds for the componentwise distance to the nearest singular matrix*, Linear and Multilinear Algebra, 42 (1998), pp. 93–108. 25

[118] S. P. Shary, *O nekotorykh metodakh resheniya lineinoi zadachi o dopuskakh*, Preprint 6, Siberian Branch of the Soviet Academy of Sciences, Krasnoyarsk, 1989. 44

[119] S. P. Shary, *A new class of algorithms for optimal solution of interval linear systems*, Interval Computations, 2 (1992), pp. 18–29. 40

[120] S. P. Shary, *On controlled solution set of interval algebraic systems*, Interval Computations, 6 (1992), pp. 66–75. 45

[121] S. P. Shary, *Solving the tolerance problem for interval linear systems*, Interval Computations, 2 (1994), pp. 6–26. 44

[122] S. P. Shary, *Solving the linear interval tolerance problem*, Mathematics and Computers in Simulation, 39 (1995), pp. 53–85. 44

[123] S. P. Shary, *Algebraic approach to the interval linear static identification, tolerance and control problems, or One more application of Kaucher arithmetic*, Reliable Computing, 2 (1996), pp. 3–33. 44, 45

[124] S. P. Shary, *Algebraic solutions to interval linear equations and their applications*, in Numerical Methods and Error Bounds, G. Alefeld and J. Herzberger, eds., Mathematical Research, Vol. 89, Berlin, 1996, Akademie Verlag, pp. 224–233. 40

[125] S. P. Shary, *A new approach to the analysis of static systems under interval uncertainty*, in Scientific Computing and Validated Numerics, G. Alefeld, A. Frommer, and B. Lang, eds., Berlin, 1996, Akademie Verlag, pp. 118–132. 43

[126] S. P. Shary, *Controllable solutions sets to interval static systems*, Applied Mathematics and Computation, 86 (1997), pp. 185–196. 45

[127] S. P. Shary, *A new technique in systems analysis under interval uncertainty and ambiguity*, Reliable Computing, 8 (2002), pp. 321–418. 40, 45

[128] V. V. Shaydurov and S. P. Shary, *Resheniye interval'noi algebraicheskoi zadachi o dopuskakh*, Preprint 5, Siberian Branch of the Soviet Academy of Sciences, Krasnoyarsk, 1988. 44

[129] Y. I. Shokin, *Interval'nyi analiz*, Nauka, Novosibirsk, 1981. 40

[130] R. D. Skeel, *Scaling for numerical stability in Gaussian elimination*, Journal of the ACM, 26 (1979), pp. 494–526. 40

[131] G. W. Stewart, *Matrix Algorithms, Volume I: Basic Decompositions*, SIAM, Philadelphia, 1998. 40

[132] A. A. Vatolin, *On the linear programming problems with interval coefficients (in Russian)*, Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 24 (1984), pp. 1629–1637. 50