An Algorithm for Checking Stability of Symmetric Interval Matrices

Jiri Rohn

Abstract— **A branch-and-bound algorithm for checking Hurwitz and Schur stability of symmetric interval matrices is proposed. The algorithm in a finite number of steps either verifies stability, or finds a symmetric matrix which is not stable. It can also be used for checking positive definiteness of nonsymmetric interval matrices.**

Keywords— **Interval matrix, symmetric matrix, stability, algorithm**

I. Introduction

A square matrix A is said to be (Hurwitz) stable if $\operatorname{Re}\lambda < 0$ for each eigenvalue λ of A. A square interval matrix

$$A^I = [\underline{A}, \overline{A}] := \{A;\ \underline{A} \le A \le \overline{A}\} \tag{1}$$

(componentwise inequalities) is called stable if each $A \in A^I$ is stable. Stability of interval matrices has been recently studied in robust control theory due to its close relationship to stability of a linear time-invariant system $\dot{x}(t) = Ax(t)$ under data perturbations.

In this paper we focus our attention on *symmetric* interval matrices; by definition, an interval matrix (1) is called symmetric if both the matrices $\underline{A}$ and $\overline{A}$ are symmetric. Hence, a symmetric interval matrix may contain nonsymmetric matrices. Necessary and sufficient conditions for stability of symmetric interval matrices, formulated in terms of stability of a finite subset of matrices in A^I, were given by Soh [7], Hertz [1], Wang and Michel [8] and Rohn [5]. In all the cases, the cardinality of the set of test matrices is exponential in the matrix size, therefore the conditions are hardly applicable for higher dimensions. This fact is explained by a recent result stating that checking stability of symmetric interval matrices is NP-hard (Rohn [6]); Nemirovskii [3] proved earlier that the problem of checking stability of general interval matrices is NP-hard.

In this paper we propose a branch-and-bound-type algorithm for checking stability of symmetric interval matrices, based on necessary and/or sufficient stability conditions. In view of the NP-hardness result, the algorithm cannot be expected to circumvent exponentiality in the verification process in general, but due to the built-in branch-and-bound strategy it reduces the amount of computations essentially in many cases. We wish to point out that, in contrast to stability checks based on sufficient conditions only, this algorithm always yields a result: in a finite number of steps it either verifies stability of A^I, or finds an unstable symmetric matrix in A^I. The branch-and-bound technique was used by Kokame and Mori [2] for checking stability of matrix polytopes; our approach makes use of the specific structure of interval matrices.

The paper is organized as follows. In section 2 we give the theoretical background of the algorithm, which itself is described in section 3, where it is also proved that it checks stability or finds an unstable symmetric matrix in a finite number of steps. An example is presented in section 4 and applications of the algorithm to some other problems (including Schur stability) are given in section 5.

The author is with the Faculty of Mathematics and Physics, Charles University, Prague, and with the Institute of Computer Science, Academy of Sciences, Prague, Czech Republic.

This work was supported by the Czech Republic Grant Agency under grant 201/93/0429.

II. Theoretical background

For an $n \times n$ symmetric interval matrix (1), using the set

$$Z_0 = \{z \in R^n;\ z_j \in \{-1, 0, 1\} \text{ for each } j\},$$

we define real matrices A_z, D_z, $z \in Z_0$ by

$$(A_z)_{ij} = \begin{cases} \frac{1}{2}(\underline{A}_{ij} + \overline{A}_{ij}) & \text{if} \quad z_i z_j = 0 \text{ and } i \ne j \\ \overline{A}_{ij} & \text{if} \quad z_i z_j = 1 \text{ or } i = j \\ \underline{A}_{ij} & \text{if} \quad z_i z_j = -1 \end{cases} \tag{2}$$

and

$$(D_z)_{ij} = \begin{cases} \frac{1}{2}(\overline{A}_{ij} - \underline{A}_{ij}) & \text{if } z_i z_j = 0 \text{ and } i \ne j \\ 0 & \text{if } z_i z_j \ne 0 \text{ or } i = j \end{cases} \tag{3}$$

$(i, j = 1, \ldots, n)$. It follows from the definition that for each $z \in Z_0$, both the matrices A_z and D_z are symmetric, $A_z \in A^I$ and $D_z \ge 0$. Let us additionally introduce the set

$$Z = \{z \in R^n;\ z_j \in \{-1, 1\} \text{ for each } j\},$$

so that $Z \subset Z_0$. First we have this necessary and sufficient condition:

Theorem 1: A symmetric interval matrix A^I is stable if and only if A_z is stable for each $z \in Z$, $z_1 = 1$.

Proof: It follows from (2) that for each $z \in Z$ the matrix A_z is of the form

$$(A_z)_{ij} = \begin{cases} \overline{A}_{ij} & \text{if} \quad z_i z_j = 1 \\ \underline{A}_{ij} & \text{if} \quad z_i z_j = -1 \end{cases} \tag{4}$$

hence Theorem 6 in [5] gives that A^I is stable if and only if each A_z, $z \in Z$ is stable. But since $A_z = A_{-z}$ due to (4), it is sufficient to consider the z's with $z_1 = 1$ only. ■

Let us note that the matrices A_z were defined for $z \in Z_0$ (for the purposes of the main algorithm in section 3), but only those satisfying $z \in Z$, $z_1 = 1$ are used in the necessary and sufficient condition of Theorem 1 which requires checking 2^{n-1} symmetric matrices for stability. In order to get this number possibly decreased, we shall employ the condition in frame of a branch-and-bound strategy. For this purpose we shall need a verifiable *sufficient* condition for stability of symmetric interval matrices which is provided in the next theorem. Here, I denotes the unit matrix and $\|D\|_1 = \max_j \sum_i |d_{ij}|$ for $D = (d_{ij})$.

Theorem 2: A symmetric interval matrix

$$[A - D, A + D]$$

is stable if the symmetric matrix

$$A + \|D\|_1 I \tag{5}$$

is stable.

Proof: Take a $z \in Z$ and consider the matrix A_z defined for $[\underline{A}, \overline{A}] := [A - D, A + D]$ by (2). Since D is symmetric and nonnegative, for each $x \in R^n$, $\|x\|_2 = 1$ we have

$$\begin{aligned} x^T(A_z - A)x &\le |x|^T D|x| \le \max_{\|y\|_2 = 1} y^T D y \\ &= \lambda_{\max}(D) = \varrho(D) \le \|D\|_1, \end{aligned}$$

which implies

$$x^T A_z x = x^T A x + x^T(A_z - A)x \le x^T(A + \|D\|_1 I)x < 0$$

in view of stability of (5). Hence for the maximal eigenvalue of the symmetric matrix A_z we have

$$\lambda_{\max}(A_z) = \max_{\|x\|_2=1} x^T A_z x < 0,$$

which means that A_z is stable. Hence each A_z, $z \in Z$ is stable, and from Theorem 1 we conclude that $[A-D, A+D]$ is stable. ■

Checking stability of a symmetric matrix $\tilde{A}$ can be performed in two ways: either by computing $\lambda_{\max}(\tilde{A})$ (see [4]) and checking its negativity, or by verifying that the matrix $-\tilde{A}$ is positive definite, using Sylvester determinant criterion.

Next we shall also need a verifiable sufficient condition for checking *instability* of a symmetric interval matrix A^I. According to Theorem 1, A^I is unstable if and only if $\lambda_{\max}(A_z) \geq 0$ for some $z \in Z$, $z_1 = 1$. The algorithm described below generates a sequence of matrices A_z with ascending values of $\lambda_{\max}(A_z)$. For a vector $x \in R^n$, we define its sign vector $\operatorname{sgn} x$ by

$$(\operatorname{sgn} x)_j = \begin{cases} 1 & \text{if } x_j \geq 0 \\ -1 & \text{if } x_j < 0, \end{cases}$$

hence $\operatorname{sgn} x \in Z$ for each x. We have this algorithm for checking instability of $A^I = [A-D, A+D]$:

compute $\lambda_{\max}(A)$ and a corresponding eigenvector x;
repeat
  $z := \operatorname{sgn} x$;
  compute $\lambda_{\max}(A_z)$ and a corresponding eigenvector x
until ($\lambda_{\max}(A_z) \geq 0$ or $D_{ij} z_i x_i z_j x_j \geq 0$ for each i, j);
if $\lambda_{\max}(A_z) \geq 0$ **then** {A^I is unstable}
else {instability was not verified}.

Theorem 3: The algorithm in a finite number of steps either verifies instability of A^I, or fails.

Proof: Since the two options for termination of the algorithm are obvious from its description, we must prove its finiteness only. To this end, we shall prove that for each successive z and z' generated by the algorithm we have

$$\lambda_{\max}(A_z) < \lambda_{\max}(A_{z'}). \tag{6}$$

This will imply that the sequence $\{\lambda_{\max}(A_z)\}$ is strictly increasing, hence the same z cannot be generated twice, and since the set Z is finite, the finiteness of the algorithm will follow. Denote $S = \operatorname{diag}\{z_1, \ldots, z_n\}$, i.e. S is the diagonal matrix with diagonal vector z. Then from (4) we have that A_z can be written as $A_z = A + SDS$. Let x be the eigenvector corresponding to $\lambda_{\max}(A_z)$ computed by the algorithm and normalized so that $\|x\|_2 = 1$. Then

$$\begin{aligned} \lambda_{\max}(A_z) &= x^T(A+SDS)x = x^T A x + \sum_{i,j} z_i x_i D_{ij} z_j x_j \\ &< x^T A x + \sum_{i,j} |x_i| D_{ij} |x_j| \end{aligned} \tag{7}$$

since the algorithm did not stop at z, hence it must have been $z_i x_i D_{ij} z_j x_j < 0$ for some i, j. Now, since $z' = \operatorname{sgn} x$, for $S' = \operatorname{diag}\{z'_1, \ldots, z'_n\}$ we have

$$\begin{aligned} x^T A x + \sum_{i,j} |x_i| D_{ij} |x_j| &= x^T A x + \sum_{i,j} z'_i x_i D_{ij} z'_j x_j \\ = \; x^T(A + S'DS')x &\leq \lambda_{\max}(A_{z'}) \end{aligned} \tag{8}$$

since $A_{z'} = A + S'DS'$, hence (7) and (8) imply (6), which completes the proof. ■

This algorithm proved to perform surprisingly effectively (see section 4). We ascribe this feature to the built-in "steepest ascent" strategy: as it can be seen from the proof, for current z and x the next z' computed by the algorithm satisfies

$$x^T A_{z'} x = \max\{x^T A_{\tilde{z}} x;\ \tilde{z} \in Z\},$$

i.e. the algorithm takes the steepest ascent in maximizing $x^T A_{\tilde{z}} x$ for fixed x.

III. The algorithm

We first give an informal description of the general algorithm. Starting from A^I, it proceeds by bisections towards the matrices A_z, $z \in Z$, $z_1 = 1$ used in the necessary and sufficient condition of Theorem 1. This is formally done by constructing successively matrices A_z, $z \in Z_0$, $z_1 = 1$ (starting from $z^* = (1, 0, \ldots, 0)^T$) and by replacing zeros in the z's by -1 or 1. If for some $z \in Z_0$ the symmetric interval matrix $[A_z - D_z, A_z + D_z] \subseteq A^I$ is found unstable, then the algorithm terminates; if $A_z + \|D_z\|_1 I$ is stable, then the interval matrix $[A_z - D_z, A_z + D_z]$ is stable (Theorem 2) and is removed from further considerations. If $A_z + \|D_z\|_1 I$ is not stable, then we must replace some zero entry in the current z by -1 and 1. To this end, we first find indices i, j, $i < j$, satisfying

$$(D_z)_{ij} = \max_{k<m} (D_z)_{km}. \tag{9}$$

Since A_z is stable, $A_z + \|D_z\|_1 I$ is not stable and D_z is symmetric, we have $(D_z)_{ij} > 0$ and $z_i z_j = 0$ (due to (3)). Set $h := i$ if $z_i = 0$ and $h := j$ otherwise, so that $z_h = 0$. Then we can construct a pair of new vectors z^1, $z^2 \in Z_0$ by

$$z^1_h = -1, z^1_k = z_k \;\text{ otherwise}$$

and

$$z^2_h = 1, z^2_k = z_k \;\text{ otherwise},$$

and insert them into the list L of items to be tested. Notice that since both A_z and D_z given by (2), (3) are uniquely determined by z, we can keep only the vector z in the list L; this reduces the storage requirement from $2n^2$ to n, thereby increasing significantly the size of interval matrices that can be processed on a given computer. The detailed description of the algorithm is as follows:

$L := \{(1, 0, \ldots, 0)^T\}$; *unstab:=false*;
repeat
  remove the topmost entry z from L;
  compute A_z, D_z by (2), (3);
  apply the algorithm of section 2 to check $[A_z - D_z, A_z + D_z]$ for instability;
  if $[A_z - D_z, A_z + D_z]$ is unstable **then** *unstab:=true*
  else
    if $A_z + \|D_z\|_1 I$ is unstable **then**
      find i, j satisfying (9);
      if $z_i = 0$ **then** $h := i$ **else** $h := j$;
      $z^1 := z$; $z^1_h := -1$;
      $z^2 := z$; $z^2_h := 1$;
      insert z^1, z^2 into L
until ($L = \emptyset$ or *unstab*);
if *unstab* **then** {A^I is unstable}
else {A^I is stable}.

We shall now prove that the algorithm yields an answer in a finite number of steps.

Theorem 4: For each symmetric interval matrix A^I, the algorithm after a finite number of steps either verifies stability of A^I, or finds an unstable symmetric matrix in A^I.

Proof: The algorithm starts from $z^* = (1, 0, \ldots, 0)^T$ and in each loop it replaces a zero entry in the current vector z by -1 and 1. Therefore the same z never reappears, and the number of steps of the algorithm is finite.

If $[A_z - D_z, A_z + D_z]$ was proved unstable for some $z \in Z_0$, then the algorithm of section 2 found an unstable symmetric matrix $A_{\tilde{z}}$ for some $\tilde{z} \in Z$, hence A^I is unstable. Thus to complete the proof, we must show that if $L = \emptyset$ at some step, then A^I is stable.

Take a $\tilde{z} \in Z$ with $\tilde{z}_1 = 1$. Since the algorithm started from z^* and proceeded by replacing zeros by -1's and 1's, it must have constructed, among others, a sequence of z's (with decreasing number of zeros), each of them having the property

$$z_i \neq 0 \Rightarrow z_i = \tilde{z}_i \tag{10}$$

for each i. Since $L = \emptyset$, the algorithm must have constructed at some stage a $z \in Z_0$ satisfying (10) such that $A_z + \|D_z\|_1 I$ was found stable, i.e. the interval matrix $[A_z - D_z, A_z + D_z]$ was proved to be stable. However, from the definition of A_z, D_z in (2), (3) it follows that (10) implies $A_{\tilde{z}} \in [A_z - D_z, A_z + D_z]$, hence $A_{\tilde{z}}$ is stable. Thus we have proved that $A_{\tilde{z}}$ is stable for each $\tilde{z} \in Z$ with $\tilde{z}_1 = 1$, which according to Theorem 1 means that A^I is stable. ■

IV. Example

To check the efficiency of the algorithm, we constructed a 7×7 matrix with integer coefficients randomly generated in the interval $[-9, 9]$:

$$A = \begin{pmatrix} 4 & 1 & 2 & -4 & -4 & 5 & -9 \\ 5 & 6 & 4 & -9 & -2 & 7 & 6 \\ -2 & 9 & 7 & -8 & 9 & -3 & 0 \\ 5 & -8 & 2 & -1 & -4 & 2 & 3 \\ -4 & -4 & 6 & 6 & 2 & 9 & 8 \\ -5 & 4 & 9 & -5 & 1 & -7 & 9 \\ 3 & -9 & 1 & -8 & -8 & 6 & -4 \end{pmatrix}.$$

This matrix is nonsingular, hence

$$\underline{A} = -A^T A$$

is symmetric and stable. We shall consider symmetric interval matrices of the form

$$A^I_\varepsilon = [\underline{A}, \underline{A} + \varepsilon\Delta]$$

where ε is a nonnegative parameter and

$$\Delta = \begin{pmatrix} 0 & 2 & 3 & 3 & 9 & 9 & 4 \\ 2 & 2 & 1 & 1 & 6 & 4 & 4 \\ 3 & 1 & 7 & 3 & 6 & 2 & 1 \\ 3 & 1 & 3 & 5 & 0 & 4 & 9 \\ 9 & 6 & 6 & 0 & 2 & 7 & 3 \\ 9 & 4 & 2 & 4 & 7 & 2 & 9 \\ 4 & 4 & 1 & 9 & 3 & 9 & 6 \end{pmatrix}$$

is a randomly generated symmetric matrix with integer coefficients in the interval $[0, 9]$. For each $\varepsilon > 0$, let us denote by $\nu(\varepsilon)$ the number of interval matrices $[A_z - D_z, A_z + D_z]$ checked for stability or instability by the main algorithm when applied to A^I_ε; hence, if A^I_ε is stable, then $\nu(\varepsilon)$ is exactly the number of z's inserted into the list L. It turns out that for $\varepsilon \in [0, 0.8]$ we always have $\nu(\varepsilon) = 1$, hence the sufficient condition of Theorem 2 checks stability immediately at the first interval matrix $[A_{z^*} - D_{z^*}, A_{z^*} + D_{z^*}]$, $z^* = (1, 0, \ldots, 0)^T$. Continuation for larger values of ε is summed up in the following table:

ε	stab./unstab.	$\nu(\varepsilon)$
0.8	stable	1
0.9	stable	23
1.0	stable	23
1.1	stable	35
1.2	stable	47
1.3	stable	67
1.4	stable	85
1.5	unstable	1

Notice that a direct application of the necessary and sufficient condition of Theorem 1 requires checking $2^6 = 64$ matrices for stability. If we increase ε from 0 on by step 0.01, then $\nu(\varepsilon) \leq 64$ for $\varepsilon \in [0, 1.28]$. Values of $\nu(\varepsilon)$ greater than 64 occur for $\varepsilon \in [1.29, 1.47]$, with the peak at $\nu(1.47) = 109$; hence, for $\varepsilon \in [1.29, 1.47]$ a direct application of Theorem 1 gives the result with a lower computational effort. Change to instability occurs between 1.47 and 1.48:

ε	stab./unstab.	$\nu(\varepsilon)$
1.41	stable	89
1.42	stable	89
1.43	stable	87
1.44	stable	91
1.45	stable	99
1.46	stable	99
1.47	stable	109
1.48	unstable	1
1.49	unstable	1
1.50	unstable	1

Particularly surprising here is the performance of the algorithm for checking instability from section 2 (last three lines of the table) which detects instability immediately at the first interval matrix $[A_{z^*} - D_{z^*}, A_{z^*} + D_{z^*}]$ in all three cases. A more detailed zooming shows that initially the number of interval matrices checked for instability is greater than 1, but decreases rapidly to 1 (dots indicate intervals of constant values of $\nu(\varepsilon)$):

ε	stab./unstab.	$\nu(\varepsilon)$
1.4766	stable	103
1.4767	unstable	16
...	...	...
1.4770	unstable	16
1.4771	unstable	22
...	...	...
1.4782	unstable	22
1.4783	unstable	16
1.4784	unstable	1
1.4785	unstable	1
...	...	...

From $\varepsilon = 1.479$ on, the algorithm needs to compute at most three values of $\lambda_{\max}(A_z)$ to detect instability. These results (also confirmed on other examples) indicate that the algorithm for checking instability might be very effective and deserves to be further studied.

V. Applications

The algorithm can also be used for solving some other problems that can be reformulated in terms of Hurwitz stability of symmetric interval matrices.

1. Schur stability of symmetric interval matrices. A symmetric interval matrix $A^I = [\underline{A}, \overline{A}]$ is called Schur stable if each *symmetric* $A \in A^I$ is Schur stable, i.e. satisfies $\varrho(A) < 1$ (ϱ is the spectral radius). As proved in [5], Theorem 10, a symmetric $A^I = [\underline{A}, \overline{A}]$ is Schur stable if and only if the two symmetric interval matrices

$$[\underline{A} - I, \overline{A} - I] \tag{11}$$

and

$$[-\overline{A} - I, -\underline{A} - I] \tag{12}$$

are Hurwitz stable. Hence Schur stability of A^I can be verified by applying the algorithm twice to the interval matrices (11) and (12). Checking Schur stability is NP-hard, cf. [6].

2. Positive definiteness of interval matrices. As proved in [5], Theorems 2 and 6, a general (not necessarily symmetric) square interval matrix A^I is positive definite (i.e., each $A \in A^I$ satisfies $x^T Ax > 0$ for each $x \neq 0$) if and only if the symmetric interval matrix

$$\left[-\frac{1}{2}(\overline{A} + \overline{A}^T), -\frac{1}{2}(\underline{A} + \underline{A}^T)\right] \tag{13}$$

is Hurwitz stable. Hence, applying the algorithm to the symmetric interval matrix (13), we can check positive definiteness of an interval matrix in a finite number of steps. The problem is again NP-hard in general [6].

3. Stability of nonsymmetric interval matrices. In [5], Theorem 7 it is proved that if the symmetric interval matrix

$$A_s^I = \left[\frac{1}{2}(\underline{A} + \underline{A}^T), \frac{1}{2}(\overline{A} + \overline{A}^T)\right] \tag{14}$$

is stable, then

$$A^I = [\underline{A}, \overline{A}]$$

is also stable; the converse statement, however, is generally not true (cf. a counterexample following Theorem 7 in [5]). Thus, if the algorithm applied to (14) verifies stability of A_s^I, then we have also verified stability of A^I. However, if A_s^I is proved to be unstable, then no conclusion concerning A^I can be drawn from this fact.

Acknowledgments

The author wishes to thank three anonymous referees for their constructive criticism that helped to improve essentially the results of this paper.

References

[1] D. Hertz, "The extreme eigenvalues and stability of real symmetric interval matrices," *IEEE Trans. Autom. Contr.*, vol. 37, pp. 532-535, 1992.

[2] H. Kokame and T. Mori, "A branch and bound method to check the stability of a polytope of matrices," in: *Robustness of Dynamic Systems with Parameter Uncertainties* (M. Mansour, S. Balemi and W. Truoel, eds.), Birkhäuser Verlag, Basel 1992, pp. 125-137.

[3] A. Nemirovskii, "Several NP-hard problems arising in robust stability analysis," *Math. Control Signals Syst.*, vol. 6, pp. 99-105, 1993.

[4] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ 1980.

[5] J. Rohn, "Positive definiteness and stability of interval matrices," *SIAM J. Matrix Anal. Appl.*, vol. 15, pp. 175-184, 1994.

[6] J. Rohn, "Checking positive definiteness or stability of symmetric interval matrices is NP-hard," *Commentat. Math. Univ. Carol.*, vol. 35, pp. 795-797, 1994.

[7] C. B. Soh, "Necessary and sufficient conditions for stability of symmetric interval matrices," *Int. J. Control*, vol. 51, pp. 243-248, 1990.

[8] K. Wang and A. Michel, "On sufficient conditions for the stability of interval matrices," *Syst. Control Lett.*, vol. 20, pp. 345-351, 1993.