An Algorithm for Finding a Singular Matrix in an Interval Matrix

Rohn, Jiří
2010

Institute of Computer Science
Academy of Sciences of the Czech Republic

An Algorithm for Finding a Singular Matrix in an Interval Matrix

Jiří Rohn

Technical report No. V-1087

15.11.2010

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

An Algorithm for Finding a Singular Matrix in an Interval Matrix

Jiří Rohn[1]

Technical report No. V-1087

15.11.2010

Abstract:

Described is a not-a-priori-exponential algorithm which in a finite number of steps either finds a singular matrix in a given interval matrix, or states that no such singular matrix exists.

Keywords:
Interval matrix, singularity, regularity, algorithm.

[1]This work was supported by the Czech Republic Grant Agency under grants 201/09/1957 and 201/08/J020, and by the Institutional Research Plan AV0Z10300504.

1 Introduction and notations

In this paper we use the following notations. The ith row of a matrix A is denoted by $A_{i\bullet}$, the jth column by $A_{\bullet j}$. Matrix inequalities, as $A \le B$ or $A < B$, are understood componentwise. The absolute value of a matrix $A = (a_{ij})$ is defined by $|A| = (|a_{ij}|)$. The same notations also apply to vectors that are considered one-column matrices. I is the unit matrix, and $e = (1, \ldots, 1)^T$ is the vector of all ones. $Y_n = \{y \mid |y| = e\}$ is the set of all ± 1-vectors in $\mathbb{R}^n$, so that its cardinality is 2^n. For each $x \in \mathbb{R}^n$ we define its sign vector $\mathrm{sgn}(x)$ by

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

so that $\mathrm{sgn}(x) \in Y_n$. For each $y \in \mathbb{R}^n$ we denote

$$T_y = \mathrm{diag}\,(y_1, \ldots, y_n) = \begin{pmatrix} y_1 & 0 & \ldots & 0 \\ 0 & y_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & y_n \end{pmatrix},$$

and $\mathbb{R}^n_y = \{x \mid T_y x \ge 0\}$ is the orthant prescribed by the ± 1-vector $y \in Y_n$. $\varrho(A)$ stands for the spectral radius of A. An interval matrix is a set of matrices

$$\mathbf{A} = \{\, A \mid |A - A_c| \le \Delta \,\} = [A_c - \Delta, A_c + \Delta],$$

and an interval vector is a one-column interval matrix

$$\mathbf{b} = \{\, b \mid |b - b_c| \le \delta \,\} = [b_c - \delta, b_c + \delta].$$

This paper is dedicated to a simple-to-formulate, yet a difficult problem: given a square interval matrix $\mathbf{A}$, find a singular matrix $S \in \mathbf{A}$ or prove that no such singular matrix exists. This problem in full generality has not been given much attention in interval analysis so far, apparently because of its difficulty: already in 1993 it has been proved by Poljak and Rohn [6] that even the problem of checking *existence* of a singular matrix in $\mathbf{A}$ (not to speak of finding one) is NP-complete. Yet this problem has important applications, as e.g. finding a not positive definite matrix in a not positive definite interval matrix [9], [13], finding a nonpositive principal minor in a matrix which is not a P-matrix [18], [12], finding a nontrivial solution of an absolute value inequality $|Ax| \le |B||x|$ [15], or solving the system $-e \le Ax \le e$, $\|x\|_1 \ge 1$ [2], [11].

The solution of this problem, presented in this paper, relies on a recently published algorithm **intervalhull** [7], invoked by

$$[\mathbf{x}, S] = \textbf{intervalhull}\,(\mathbf{A}, \mathbf{b})$$

which for each $n \times n$ interval matrix $\mathbf{A}$ and for each interval n-vector $\mathbf{b}$ in a finite number of steps either finds a singular matrix $S \in \mathbf{A}$, or computes the so-called interval hull $\mathbf{x}$, which is the intersection of all interval vectors enclosing the solution set $X = \{\, x \mid Ax = b$ for some $A \in \mathbf{A}$, $b \in \mathbf{b}\,\}$. Since existence of a singular matrix in $\mathbf{A}$ implies unboundedness of the solution set X and thereby nonexistence of the interval hull (Jansson [4]), successful

computation of the interval hull implies nonexistence of a singular matrix in $\mathbf{A}$. Since the algorithm **intervalhull** in infinite precision arithmetic always yields exactly one output (either $\mathbf{x}$, or S; the second one is always empty), this gives us a direct way how to solve our problem fully in a finite number of steps.

We present two versions of the algorithm. The first version, called the "short" one (Fig. 2.1), is based on the idea just described, and has the attractive feature of simplicity. It works as required and yields the result for any data, but it is still rather slow for many examples. Therefore in the "long" version (Fig. 3.1) we add two sufficient conditions that in many cases help to avoid using the algorithm **intervalhull** at all, and one heuristic for finding a proper value of the right-hand side $\mathbf{b}$ in case **intervalhull** is applied.

The resulting algorithm is not-a-priori-exponential. That means, the number of steps taken is not always exponential and depends on the actual input $\mathbf{A}$; but of course it is exponential in the worst case. The important feature of not-a-priori-exponentiality is based on the idea of C. Jansson described in his paper [4]. Roughly said, the idea consists in avoiding as many orthants as possible while constructing the interval hull. This idea is incorporated into the algorithm **intervalhull**, Fig. 5.1, see the construction of the sets Z and D there. But construction of a singular matrix S is hidden in its subalgorithm **absvaleqn**, Fig. 5.2 (lines (14) and (19)-(23) there), wherefrom it is transferred to **intervalhull** itself.

2 The algorithm: short version

For better understanding, we first present a short version of the algorithm (Fig. 2.1).

```
(01)  function S = singregshort (A)
(02)  % S ≠ []: S is a singular matrix in A.
(03)  % S = []: no singular matrix in A exists.
(04)   n = size(A, 1); e = (1, ..., 1)^T ∈ R^n;
(05)   [x, S] = intervalhull (A, [e, e]);
```

Figure 2.1: An algorithm for finding a singular matrix (short version).

As explained in the Introduction, this algorithm is already fully operational and, in infinite precision arithmetic, in a finite number of steps produces exactly one of the two possible outputs described in lines (02), (03). However, in many examples it turns out to be still too slow.

3 The algorithm: long version

To speed up performance of the algorithm, we add three new features explained in the proof of Theorem 1 below. The final "long" version of the algorithm is given in Fig. 3.1.

The following finite termination theorem describes the basic properties of the algorithm.

Theorem 1. *For each square interval matrix* $\mathbf{A}$ *the algorithm* **singreg** *(Fig. 3.1) in a finite, but not-a-priori-exponential number of steps either finds a singular matrix* $S \in \mathbf{A}$ *(the case of* $S \neq [\,]$*), or states that no such singular matrix exists (the case of* $S = [\,]$*).*

```
(01)  function S = singreg (A)
(02)  % S ≠ [ ]: S is a singular matrix in A.
(03)  % S = [ ]: no singular matrix in A exists.
(04)   S = [ ]; n = size(A, 1); e = (1, ..., 1)^T ∈ R^n;
(05)   if A_c is singular, S = A_c; return, end
(06)   R = A_c^{-1}; D = Δ|R|;
(07)   if D_kk = max_j D_jj ≥ 1
(08)      x = R_{•k};
(09)      for i = 1 : n
(10)         if (Δ|x|)_i > 0, y_i = (A_c x)_i/(Δ|x|)_i; else y_i = 1; end
(11)         if x_i ≥ 0, z_i = 1; else z_i = −1; end
(12)      end
(13)      S = A_c − T_y Δ T_z; return
(14)   end
(15)   if ϱ(D) < 1, return, end
(16)   b = e;
(17)   x = Rb; γ = min_k |x_k|;
(18)   for i = 1 : n
(19)      for j = 1 : n
(20)         x' = x − 2b_j R_{•j};
(21)         if min_k |x'_k| > γ, γ = min_k |x'_k|; x = x'; b(j) = −b(j); end
(22)      end
(23)   end
(24)   [x, S] = intervalhull (A, [b, b]);
```

Figure 3.1: An algorithm for finding a singular matrix.

Proof. The not-a-priori-exponentiality of the algorithm follows from the same property of the algorithm **intervalhull** stated in Theorem 2 in the Appendix. The long version of the algorithm consists of several blocks.

The block of lines (04)-(06) is preparatory. Its main purpose consists in computation of the matrix $D = \Delta|A_c^{-1}|$ to be used in the next two blocks.

Block (07)-(14): If $\max_j D_{jj} \geq 1$, then $\mathbf{A}$ contains a singular matrix ([8], Corollary 5.2, (iii)), and a specific singular matrix S in $\mathbf{A}$ (line (13)) is constructed along the proofs of Theorem 12 and of Proposition 10 in [10].

Block (15): If $\varrho(D) < 1$, then, by Beeck's result in [1], no singular matrix exists in $\mathbf{A}$.

Block (16)-(23): This is a heuristic coming from [5] aimed at finding a b which would possibly minimize the number of orthants intersected by the solution set of the system of interval linear equations $\mathbf{A}x = [b, b]$. This is done by maximizing, in n^2 successive steps, the value of $\min_k(A_c^{-1}b)_k$. It is easy to see that at each step after updating in line (21) we have $x = A_c^{-1}b$ for the current b.

Block (24): According to the basic property of the algorithm **intervalhull** stated in Theorem 2, the algorithm in a finite number of steps either finds a singular matrix $S \in \mathbf{A}$, in which case we are done, or computes the interval hull $\mathbf{x}$ of the solution set of $\mathbf{A}x = [b, b]$, in

which case the solution set is bounded and therefore no singular matrix in $\mathbf{A}$ exists (Jansson [4]). This completes the proof. □

In order to avoid computation of the spectral radius, the condition $\varrho(D) < 1$ in line (15) may be replaced by a weaker condition $\|D\|_1 < 1$. This is so because $\varrho(D) \leq \|D\|_1$ for each square D, see [3].

4 Implementation

The algorithm has been implemented in the VERREGSING.M [14] function of the freely available verification software package VERSOFT [16]. The results are *verified* there, i.e., they hold as mathematical truths despite being achieved by finite precision computations. In particular, a singular matrix is outputted there as a tiny floating-point interval matrix verified to contain a real singular matrix. This is achieved by using sometimes complicated verification procedures whose inclusion here would lead us well beyond the scope of the present paper. This is why we presented the results in infinite precision arithmetic only.

5 Appendix

To keep the paper self-contained, we append here full descriptions of the algorithm **intervalhull** and of its two subalgorithms **qzmatrix** and **absvaleqn** (Figs. 5.1 and 5.2) as they appear in [7]. The following theorem was proved in [7]; for explanation of **absvaleqn**, see [15] and [17].

Theorem 2. *For each $n \times n$ interval matrix $\mathbf{A}$ and for each interval n-vector $\mathbf{b}$ the algorithm* **intervalhull** *(Figs. 5.1 and 5.2) in a finite, but not-a-priori-exponential number of steps either computes the interval hull $\mathbf{x}$ of the solution set of the system of interval linear equations $\mathbf{A}x = \mathbf{b}$ (the case of $\mathbf{x} \neq [\,]$, $S = [\,]$), or produces a singular matrix $S \in \mathbf{A}$ (the case of $\mathbf{x} = [\,]$, $S \neq [\,]$).*

```
(01)  function [x, S] = intervalhull (A, b)
(02)  % Computes either the interval hull x
(03)  % of the solution set of Ax = b,
(04)  % or a singular matrix S ∈ A.
(05)   x = [ ]; S = [ ];
(06)   if A_c is singular, S = A_c; return, end
(07)   x_c = A_c^{-1} b_c; z = sgn(x_c); x_ = x_c; x¯ = x_c;
(08)   Z = {z}; D = ∅;
(09)   while Z ≠ ∅
(10)      select z ∈ Z; Z = Z − {z}; D = D ∪ {z};
(11)      [Q_z, S] = qzmatrix (A, z);
(12)      if S ≠ [ ], x = [ ]; return, end
(13)      [Q_{−z}, S] = qzmatrix (A, −z);
(14)      if S ≠ [ ], x = [ ]; return, end
(15)      x¯_z = Q_z b_c + |Q_z|δ;
(16)      x_z = Q_{−z} b_c − |Q_{−z}|δ;
(17)      if x_z ≤ x¯_z
(18)         x_ = min(x_, x_z); x¯ = max(x¯, x¯_z);
(19)         for j = 1 : n
(20)            z′ = z; z′_j = −z′_j;
(21)            if ((x_z)_j (x¯_z)_j ≤ 0 and z′ ∉ Z ∪ D)
(22)               Z = Z ∪ {z′};
(23)            end
(24)         end
(25)      end
(26)   end
(27)   x = [x_, x¯];
(01)  function [Q_z, S] = qzmatrix (A, z)
(02)  % Computes either a solution Q_z
(03)  % of the equation QA_c − |Q|ΔT_z = I,
(04)  % or a singular matrix S ∈ A.
(05)  for i = 1 : n
(06)      [x, S] = absvaleqn (A_c^T, −T_z Δ^T, e_i);
(07)      if S ≠ [ ], S = S^T; Q_z = [ ]; return
(08)      end
(09)      (Q_z)_{i•} = x^T;
(10)  end
(11)  S = [ ];
```

Figure 5.1: An algorithm for computing the interval hull.

```
(01)  function [x, S] = absvaleqn (A, B, b)
(02)  % Finds either a solution x to Ax + B|x| = b, or
(03)  % a singular matrix S satisfying |S − A| ≤ |B|.
(04)  x = [ ]; S = [ ]; i = 0; r = 0 ∈ R^n; X = 0 ∈ R^{n×n};
(05)  if A is singular, S = A; return, end
(06)  z = sgn(A^{-1}b);
(07)  if A + BT_z is singular, S = A + BT_z; return, end
(08)  x = (A + BT_z)^{-1}b;
(09)  C = −(A + BT_z)^{-1}B;
(10)  while z_j x_j < 0 for some j
(11)     i = i + 1;
(12)     k = min{j | z_j x_j < 0};
(13)     if 1 + 2z_k C_kk ≤ 0
(14)        S = A + B(T_z + (1/C_kk)e_k e_k^T);
(15)        x = [ ]; return
(16)     end
(17)     if ((k < n and r_k > max_{k<j} r_j) or (k = n and r_n > 0))
(18)        x = x − X_{•k};
(19)        for j = 1 : n
(20)           if (|B||x|)_j > 0, y_j = (Ax)_j/(|B||x|)_j; else y_j = 1; end
(21)        end
(22)        z = sgn(x);
(23)        S = A − T_y|B|T_z;
(24)        x = [ ]; return
(25)     end
(26)     r_k = i;
(27)     X_{•k} = x;
(28)     z_k = −z_k;
(29)     α = 2z_k/(1 − 2z_k C_kk);
(30)     x = x + αx_k C_{•k};
(31)     C = C + αC_{•k}C_{k•};
(32)  end
```

Figure 5.2: An algorithm for solving an absolute value equation.

Bibliography

[1] 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. 4

[2] M. Fiedler, J. Nedoma, J. Ramík, J. Rohn, and K. Zimmermann, *Linear Optimization Problems with Inexact Data*, Springer-Verlag, New York, 2006. 2

[3] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985. 5

[4] C. Jansson, *Calculation of exact bounds for the solution set of linear interval systems*, Linear Algebra and Its Applications, 251 (1997), pp. 321–340. 2, 3, 5

[5] 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. 4

[6] S. Poljak and J. Rohn, *Checking robust nonsingularity is NP-hard*, Mathematics of Control, Signals, and Systems, 6 (1993), pp. 1–9. 2

[7] J. Rohn, *An algorithm for computing the hull of the solution set of interval linear equations*. Submitted. 2, 5

[8] J. Rohn, *Systems of linear interval equations*, Linear Algebra and Its Applications, 126 (1989), pp. 39–78. 4

[9] J. Rohn, *Positive definiteness and stability of interval matrices*, SIAM Journal on Matrix Analysis and Applications, 15 (1994), pp. 175–184. 2

[10] 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://uivtx.cs.cas.cz/~rohn/publist/92.pdf`. 4

[11] J. Rohn, *VERBASINTNPPROB: Verified solution of the basic NP-complete problem of interval computations*, 2008. `http://uivtx.cs.cas.cz/~rohn/matlab/verbasintnpprob.html`. 2

[12] J. Rohn, *VERPMAT: Verified P-property of a real matrix*, 2008. `http://uivtx.cs.cas.cz/~rohn/matlab/verpmat.html`. 2

[13] J. Rohn, *VERPOSDEF: Verified positive definiteness of an interval matrix*, 2008. `http://uivtx.cs.cas.cz/~rohn/matlab/verposdef.html`. 2

[14] J. Rohn, *VERREGSING: Verified regularity/singularity of an interval matrix*, 2008. http://uivtx.cs.cas.cz/~rohn/matlab/verregsing.html. 5

[15] J. Rohn, *An algorithm for solving the absolute value equation*, Electronic Journal of Linear Algebra, 18 (2009), pp. 589–599. http://www.math.technion.ac.il/iic/ela/ela-articles/articles/vol18_pp589-599.pdf. 2, 5

[16] J. Rohn, *VERSOFT: Verification software in MATLAB/INTLAB*, 2009. http://uivtx.cs.cas.cz/~rohn/matlab. 5

[17] J. Rohn, *An algorithm for solving the absolute value equation: An improvement*, Technical Report 1063, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, January 2010. http://uivtx.cs.cas.cz/~rohn/publist/absvaleqnreport.pdf. 5

[18] S. M. Rump, *On P-matrices*, Linear Algebra and Its Applications, 363 (2003), pp. 237–250. 2