

# An Algorithm for Computing the Hull of the Solution Set of Interval Linear Equations

Jiri Rohn<sup>a,b,1</sup>

<sup>a</sup>*Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic*  
<sup>b</sup>*School of Business Administration, Anglo-American University, Prague, Czech Republic*

---

## Abstract

Described is a not-a-priori-exponential algorithm 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 computes the interval hull of the solution set of the system of interval linear equations  $\mathbf{A}x = \mathbf{b}$ , or finds a singular matrix  $S \in \mathbf{A}$ .

*Keywords:* interval linear equations, solution set, interval hull, algorithm, absolute value inequality

*2008 MSC:* 65G40, 65G20

---

## 1. Introduction

This paper is dedicated to one of the most classical problems in interval analysis, namely computation of the interval hull of the solution set of a system of interval linear equations (which is an NP-hard problem). After introducing the notations being used in Section 2, we describe in Section 3 the problem itself and some results relevant to it. In Section 4 we quote a basic underlying result for computing enclosures of the solution set, which is then improved in Section 5 to yield instead the interval hull itself. In the following Section 6 we explain how to solve efficiently the absolute value matrix equations formulated in the main result of Section 5, and in Section 7 we give a detailed MATLAB-style description of the algorithm which, as the reader will see, is rather complex, but according to this author's experience, also efficient, and has been included into the INTERVALHULL.P file of the free software package VERSOFT [1].

## 2. Notations

We use the following notations. The  $i$ th row of a matrix  $A$  is denoted by  $A_{i\bullet}$ , the  $j$ th column by  $A_{\bullet j}$ . Matrix inequalities, as  $A \leq B$  or  $A < B$ , are understood componentwise.

---

*Email address:* rohn@cs.cas.cz (Jiri Rohn)

*URL:* <http://uivtx.cs.cas.cz/~rohn> (Jiri Rohn)

<sup>1</sup>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.

*Preprint submitted to Linear Algebra and Its Applications*

*February 11, 2011*

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. Minimum or maximum of a finite set of vectors is also understood componentwise.  $I$  is the unit matrix,  $e_j$  is the  $j$ th column of  $I$ , and  $e = (1, \dots, 1)^T$  is the vector of all ones.  $Y_n = \{y \mid |y| = e\}$  is the set of all  $\pm 1$ -vectors in  $\mathbb{R}^n$ , so that its cardinality is  $2^n$ . Vectors  $y, z \in Y_n$  are called adjacent if they differ in exactly one entry. Obviously,  $y, z \in Y_n$  are adjacent if and only if  $y = z - 2z_j e_j$  for some  $j$ . For each  $x \in \mathbb{R}^n$  we define its sign vector  $\text{sgn}(x)$  by

$$(\text{sgn}(x))_i = \begin{cases} 1 & \text{if } x_i \geq 0, \\ -1 & \text{if } x_i < 0 \end{cases} \quad (i = 1, \dots, n), \quad (1)$$

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

$$T_z = \text{diag}(z_1, \dots, z_n) = \begin{pmatrix} z_1 & 0 & \dots & 0 \\ 0 & z_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & z_n \end{pmatrix}, \quad (2)$$

and  $\mathbb{R}_z^n = \{x \mid T_z x \geq 0\}$  is the orthant prescribed by the  $\pm 1$ -vector  $z \in Y_n$ . An interval matrix is a set of matrices

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

and an interval vector is a one-column interval matrix

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

### 3. The problem

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 solution set of the system of interval linear equations  $\mathbf{A}x = \mathbf{b}$  is defined as

$$\mathbf{X}(\mathbf{A}, \mathbf{b}) = \{x \mid Ax = b \text{ for some } A \in \mathbf{A}, b \in \mathbf{b}\}.$$

The Oettli-Prager theorem [2] asserts that the solution set is described by

$$\mathbf{X}(\mathbf{A}, \mathbf{b}) = \{x \mid |A_c x - b_c| \leq \Delta|x| + \delta\}.$$

If  $\mathbf{A}$  is regular (i.e., each  $A \in \mathbf{A}$  is nonsingular), then  $\mathbf{X}(\mathbf{A}, \mathbf{b})$  is compact and connected (Beeck [3]); if  $\mathbf{A}$  is singular (i.e., it contains a singular matrix), then each component of  $\mathbf{X}(\mathbf{A}, \mathbf{b})$  is unbounded (Jansson [4]). The solution set is generally of a complicated nonconvex structure. In practical computations, therefore, we look for an *enclosure* of it, i.e., for an interval vector  $\mathbf{x}$  satisfying

$$\mathbf{X}(\mathbf{A}, \mathbf{b}) \subseteq \mathbf{x}.$$

If  $\mathbf{A}$  is regular, then the intersection of all enclosures of  $\mathbf{X}(\mathbf{A}, \mathbf{b})$  forms an interval vector which is called the *interval hull* (sometimes simply “hull”) of  $\mathbf{X}(\mathbf{A}, \mathbf{b})$  and is denoted by  $\mathbf{x}(\mathbf{A}, \mathbf{b})$ . Obviously,

$$\mathbf{x}(\mathbf{A}, \mathbf{b}) = [\underline{x}, \bar{x}],$$

where

$$\underline{x}_i = \min\{x_i \mid x \in \mathbf{X}(\mathbf{A}, \mathbf{b})\}, \quad (3)$$

$$\bar{x}_i = \max\{x_i \mid x \in \mathbf{X}(\mathbf{A}, \mathbf{b})\} \quad (4)$$

( $i = 1, \dots, n$ ). If  $\mathbf{A}$  is singular, then  $\mathbf{X}(\mathbf{A}, \mathbf{b})$  is either empty (a rare case), or unbounded and the interval hull is not defined in this case.

Computing the interval hull is NP-hard (original proof by Rohn and Kreinovich in [5], simplified proof in [6], Theorem 2.38). Exponential algorithms for its computation were suggested by Oettli [7], Nickel [8] and Rohn [9]. Jansson [4] was first to propose the idea of “going along the orthants” having a nonempty intersection with  $\mathbf{X}(\mathbf{A}, \mathbf{b})$ , and bound the intersections, which are convex polytopes, by a linear programming technique. His method has the additional advantage that it does not require verification of regularity/singularity in advance because it is verified (or disproved) on the way. In this paper we describe an algorithm based on a modification of his going-along-the-orthants idea which, however, employs a new technique of enclosing the intersections of the solution set with orthants by means of solutions of certain absolute value matrix equations. The main advantage of this approach consists in the fact that it allows for a verified implementation (where the result is true despite being achieved by means of finite precision arithmetic). This implementation was done in the INTERVALHULL.P file of the freely available verification software VERSOFT [1] written in INTLAB, a MATLAB toolbox developed by Rump [10]. INTERVALHULL, which was created between 2000 and 2008 and whose source file has about 1000 lines (partly due to additional verification routines included), is available there as a p-coded file only; here we disclose the algorithm behind it for the first time.

#### 4. Underlying result

The starting point of our considerations is the following result which has been proved in [11, Thm. 3] (without using  $\bar{x}_z$  and  $\underline{x}_z$  that are introduced here in (7), (8), so that the original formulation was more cumbersome).

**Theorem 1.** *Let  $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$  be an  $n \times n$  interval matrix,  $\mathbf{b} = [b_c - \delta, b_c + \delta]$  an interval  $n$ -vector, and let  $Z$  be a subset of  $Y_n$  having the following properties:*

(a)  $\text{sgn}(x_0) \in Z$  for some  $x_0 \in \mathbf{X}(\mathbf{A}, \mathbf{b})$ ,

(b) for each  $z \in Z$  the inequalities

$$(QA_c - I)T_z \geq |Q|\Delta, \quad (5)$$

$$(QA_c - I)T_{-z} \geq |Q|\Delta \quad (6)$$

have matrix solutions  $Q_z$  and  $Q_{-z}$ , respectively; denote

$$\bar{x}_z = Q_z b_c + |Q_z| \delta, \quad (7)$$

$$\underline{x}_z = Q_{-z} b_c - |Q_{-z}| \delta, \quad (8)$$

(c) if  $z \in Z$ ,  $\underline{x}_z \leq \bar{x}_z$ , and  $(\underline{x}_z)_j (\bar{x}_z)_j \leq 0$  for some  $j$ , then  $z - 2z_j e_j \in Z$ .

Then  $\mathbf{A}$  is regular and

$$\mathbf{X}(\mathbf{A}, \mathbf{b}) \subseteq \left[ \min_{z \in Z_1} \underline{x}_z, \max_{z \in Z_1} \bar{x}_z \right] \quad (9)$$

holds, where

$$Z_1 = \{ z \in Z \mid \underline{x}_z \leq \bar{x}_z \}.$$

There are several facts, established in [11], needed for understanding the procedure involved:

- (i) the set  $Z$  of  $\pm 1$ -vectors  $z$  representing the orthants  $\mathbb{R}_z^n$  is constructed recursively in steps (a) and (c),
- (ii)  $\mathbf{X}(\mathbf{A}, \mathbf{b}) \cap \mathbb{R}_z^n = \emptyset$  for each  $z \in Z - Z_1$ ,
- (iii)  $\mathbf{X}(\mathbf{A}, \mathbf{b}) \cap \mathbb{R}_z^n \subseteq [\underline{x}_z, \bar{x}_z]$  for each  $z \in Z_1$ ,
- (iv) if the procedure of constructing  $Z$  is brought to conclusion (i.e., all the  $Q_z$ 's and  $Q_{-z}$ 's needed have been found), then

$$\mathbf{X}(\mathbf{A}, \mathbf{b}) \subseteq \bigcup_{z \in Z_1} [\underline{x}_z, \bar{x}_z]. \quad (10)$$

In the main result of this paper we show that if the inequalities (5), (6) are always solved as equations, then the enclosure in (9) becomes the interval hull of  $\mathbf{X}(\mathbf{A}, \mathbf{b})$ .

## 5. Main result

**Theorem 2.** Let  $\mathbf{A} = [A_c - \Delta, A_c + \Delta]$  be an  $n \times n$  interval matrix,  $\mathbf{b} = [b_c - \delta, b_c + \delta]$  an interval  $n$ -vector, and let  $Z$  be a subset of  $Y_n$  having the following properties:

(a)  $\text{sgn}(x_0) \in Z$  for some  $x_0 \in \mathbf{X}(\mathbf{A}, \mathbf{b})$ ,

(b') for each  $z \in Z$  the equations

$$QA_c - |Q|\Delta T_z = I, \quad (11)$$

$$QA_c - |Q|\Delta T_{-z} = I \quad (12)$$

have matrix solutions  $Q_z$  and  $Q_{-z}$ , respectively; denote

$$\bar{x}_z = Q_z b_c + |Q_z| \delta, \quad (13)$$

$$\underline{x}_z = Q_{-z} b_c - |Q_{-z}| \delta, \quad (14)$$

(c) if  $z \in Z$ ,  $\underline{x}_z \leq \bar{x}_z$ , and  $(\underline{x}_z)_j (\bar{x}_z)_j \leq 0$  for some  $j$ , then  $z - 2z_j e_j \in Z$ .

Then  $\mathbf{A}$  is regular and

$$\mathbf{x}(\mathbf{A}, \mathbf{b}) = \left[ \min_{z \in Z_1} \underline{x}_z, \max_{z \in Z_1} \bar{x}_z \right] \quad (15)$$

holds, where

$$Z_1 = \{ z \in Z \mid \underline{x}_z \leq \bar{x}_z \}.$$

PROOF. If  $Q_z$  and  $Q_{-z}$  solve (11) and (12), respectively, then they satisfy

$$\begin{aligned}(Q_z A_c - I)T_z &= |Q_z|\Delta, \\ (Q_{-z} A_c - I)T_{-z} &= |Q_{-z}|\Delta,\end{aligned}$$

so that they are solutions of (5), (6). Thus, the assumption (b') implies validity of the assumption (b) of Theorem 1, hence all three assumptions of Theorem 1 are met and consequently  $\mathbf{A}$  is regular and

$$\mathbf{X}(\mathbf{A}, \mathbf{b}) \subseteq [\min_{z \in Z_1} \underline{x}_z, \max_{z \in Z_1} \bar{x}_z] \quad (16)$$

holds. Denote

$$[\underline{x}, \bar{x}] = \mathbf{x}(\mathbf{A}, \mathbf{b}), \quad (17)$$

$$[\underline{\underline{x}}, \bar{\bar{x}}] = [\min_{z \in Z_1} \underline{x}_z, \max_{z \in Z_1} \bar{x}_z]. \quad (18)$$

In view of (16),  $[\underline{\underline{x}}, \bar{\bar{x}}]$  is an enclosure of  $\mathbf{X}(\mathbf{A}, \mathbf{b})$  and since  $\mathbf{x}(\mathbf{A}, \mathbf{b})$  is the intersection of all the enclosures, we have

$$[\underline{x}, \bar{x}] \subseteq [\underline{\underline{x}}, \bar{\bar{x}}],$$

in particular

$$\bar{x} \leq \bar{\bar{x}}. \quad (19)$$

Take an  $i \in \{1, \dots, n\}$ . Then (19) implies  $\bar{x}_i \leq \bar{\bar{x}}_i$ . To prove the converse inequality, let  $z \in Z_1$  be such that

$$\bar{\bar{x}}_i = (\bar{x}_z)_i$$

(by (18)), and define

$$q^T = e_i^T Q_z$$

and

$$y = \text{sgn}(q),$$

so that  $|q| = T_y q$ . Premultiplying the equation (11) by  $e_i^T$ , we get

$$q^T A_c - |q|^T \Delta T_z = q^T A_c - q^T T_y \Delta T_z = q^T (A_c - T_y \Delta T_z) = e_i^T,$$

hence

$$q^T = e_i^T (A_c - T_y \Delta T_z)^{-1}$$

(since  $y, z$  are  $\pm 1$ -vectors, so that  $A_c - T_y \Delta T_z \in \mathbf{A}$ , and we have already proved that  $\mathbf{A}$  is regular). Now we have

$$\begin{aligned}\bar{\bar{x}}_i &= (\bar{x}_z)_i = e_i^T (Q_z b_c + |Q_z|\delta) = q^T b_c + q^T T_y \delta \\ &= e_i^T (A_c - T_y \Delta T_z)^{-1} (b_c + T_y \delta) = x_i,\end{aligned}$$

where  $x$  is the solution of

$$(A_c - T_y \Delta T_z)x = b_c + T_y \delta$$

with  $A_c - T_y \Delta T_z \in \mathbf{A}$  (as we already know) and  $b_c + T_y \delta \in \mathbf{b}$  (since  $y$  is a  $\pm 1$ -vector). This shows that  $x \in \mathbf{X}(\mathbf{A}, \mathbf{b})$ , hence  $\bar{\bar{x}}_i$  is attained over  $\mathbf{X}(\mathbf{A}, \mathbf{b})$ , and in view of (4) this means that  $\bar{\bar{x}}_i \leq \bar{x}_i$ , so that  $\bar{\bar{x}}_i = \bar{x}_i$ . Since  $i$  was arbitrary, we finally obtain that

$$\bar{\bar{x}} = \bar{x}.$$

The equality  $\underline{\underline{x}} = \underline{x}$  is proved in a similar way. This shows that

$$\mathbf{x}(\mathbf{A}, \mathbf{b}) = [\underline{\underline{x}}, \bar{\bar{x}}],$$

which completes the proof.

## 6. Computation of the matrices $Q_z$

In the light of the main result, what remains to be resolved is the problem of solving the matrix equation

$$QA_c - |Q|\Delta T_z = I \quad (20)$$

(and the related equation (12) which is of the same form). Take an  $i \in \{1, \dots, n\}$  and put

$$x^T = Q_{i\bullet},$$

then  $x$  solves

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

and thus also

$$A_c^T x - T_z \Delta^T |x| = e_i. \quad (21)$$

The equation (21) is an equation of the form

$$Ax + B|x| = b \quad (22)$$

( $A, B \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^n$ ), which is called an *absolute value equation*. In [12], [13] we have developed a very efficient algorithm **absvaleqn** for its solution, reproduced here in Fig. 3 in a MATLAB-like form. It is called by

$$[x, S] = \mathbf{absvaleqn}(A, B, b)$$

and its main feature is that for each data  $A, B, b$ , it produces in a finite number of steps either a solution  $x$  of the equation (22), or a singular matrix  $S$  satisfying  $|S - A| \leq |B|$  (but not both; the unassigned variable is outputted as  $[\ ]$ , an empty matrix/vector). Thus, our equation (21) can be solved by

$$[x, S] = \mathbf{absvaleqn}(A_c^T, -T_z \Delta^T, e_i)$$

and the output is either a vector  $x$  with  $x^T = (Q_z)_{i\bullet}$ , or a singular matrix  $S$  satisfying

$$|S - A_c^T| \leq |-T_z \Delta^T| = \Delta^T,$$

so that

$$|S^T - A_c| \leq \Delta$$

and  $S^T$  is a singular matrix in  $\mathbf{A}$ . In this way we may compute the matrix  $Q_z$  row-by-row. This procedure is formalized in the MATLAB-like function **qzmatrix** placed in Fig. 2 as a subfunction of the main function **intervalhull** to be explained in the next section. From what has been said above it follows that *the function **qzmatrix** called by*

$$[Q_z, S] = \mathbf{qzmatrix}(\mathbf{A}, z)$$

*in a finite number of steps either finds a solution  $Q_z$  of the matrix equation (20), or produces a singular matrix  $S \in \mathbf{A}$ .*

## 7. The algorithm

The algorithm **intervalhull**, described in MATLAB-like style in Figs. 1 through 3, consists of the main function **intervalhull** and of two subfunctions **qzmatrix** and **absvaleqn**. The main function **intervalhull** closely copies Theorem 2, the subfunction **qzmatrix** is based on the results of Section 6, and the subfunction **absvaleqn** comes from [13]. Since the subfunction **absvaleqn** terminates in a finite number of steps (Theorem 3 in [13]) and the same thus holds for **qzmatrix**, we have this finite termination theorem.

**Theorem 3.** *For each  $n \times n$  interval matrix  $\mathbf{A}$  and for each interval  $n$ -vector  $\mathbf{b}$  the algorithm **intervalhull** (Figs. 1–3) in a finite number of steps either computes the interval hull  $\mathbf{x}$  of the solution set of the system of interval linear equations  $\mathbf{A}\mathbf{x} = \mathbf{b}$ , or produces a singular matrix  $S \in \mathbf{A}$  (but not both).*

The algorithm is not-a-priori-exponential since it requires  $2n \cdot \text{card}(Z)$  calls of the subfunction **absvaleqn** which itself takes about  $0.1 \cdot n$  steps on the average [12] (where under a “step” we understand a single pass through the **while ... end** loop between the lines (11) and (31) of the function **absvaleqn** in Fig. 3). For example, if the solution set  $\mathbf{X}(\mathbf{A}, \mathbf{b})$  is a part of the interior of a single orthant  $\mathbb{R}_z^n$ , then only two matrices  $Q_z$  and  $Q_{-z}$  are to be computed, so that only  $2n$  calls of **absvaleqn** are made.

## 8. Implementation

The algorithm has been implemented in the VERINTERVALHULL.M [14] function of the freely available verification software package VERSOFT [1].

## 9. Acknowledgement

The author wishes to thank the referee for his comments.

## References

- [1] J. Rohn, VERSOFT: Verification software in MATLAB/INTLAB, <http://uivtx.cs.cas.cz/~rohn/matlab> (2009).
- [2] W. Oettli, W. Prager, Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides, *Numerische Mathematik* 6 (1964) 405–409.
- [3] H. Beeck, Charakterisierung der Lösungsmenge von Intervallgleichungssystemen, *Zeitschrift für Angewandte Mathematik und Mechanik* 53 (1973) T181–T182.

```

(01) function [x, S] = intervalhull (A, b)
(02) % Computes either the interval hull x
(03) % of the solution set of  $\mathbf{Ax} = \mathbf{b}$ ,
(04) % or a singular matrix  $S \in \mathbf{A}$ .
(05) x = []; S = [];
(06) if  $A_c$  is singular,  $S = A_c$ ; return, end
(07)  $x_c = A_c^{-1}b_c$ ;  $z = \text{sgn}(x_c)$ ;  $\underline{x} = x_c$ ;  $\bar{x} = x_c$ ;
(08)  $Z = \{z\}$ ;  $D = \emptyset$ ;
(09) while  $Z \neq \emptyset$ 
(10)   select  $z \in Z$ ;  $Z = Z - \{z\}$ ;  $D = D \cup \{z\}$ ;
(11)    $[Q_z, S] = \text{qzmatrix}(\mathbf{A}, z)$ ;
(12)   if  $S \neq []$ ,  $\mathbf{x} = []$ ; return, end
(13)    $[Q_{-z}, S] = \text{qzmatrix}(\mathbf{A}, -z)$ ;
(14)   if  $S \neq []$ ,  $\mathbf{x} = []$ ; return, end
(15)    $\bar{x}_z = Q_z b_c + |Q_z| \delta$ ;
(16)    $\underline{x}_z = Q_{-z} b_c - |Q_{-z}| \delta$ ;
(17)   if  $\underline{x}_z \leq \bar{x}_z$ 
(18)      $\underline{x} = \min(\underline{x}, \underline{x}_z)$ ;  $\bar{x} = \max(\bar{x}, \bar{x}_z)$ ;
(19)     for  $j = 1 : n$ 
(20)        $z' = z$ ;  $z'_j = -z'_j$ ;
(21)       if  $((\underline{x}_z)_j (\bar{x}_z)_j \leq 0$  and  $z' \notin Z \cup D$ )
(22)          $Z = Z \cup \{z'\}$ ;
(23)       end
(24)     end
(25)   end
(26) end
(27)  $\mathbf{x} = [\underline{x}, \bar{x}]$ ;

```

Figure 1: An algorithm for computing the interval hull.

- [4] C. Jansson, Calculation of exact bounds for the solution set of linear interval systems, *Linear Algebra and Its Applications* 251 (1997) 321–340.
- [5] J. Rohn, 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) 415–420.
- [6] M. Fiedler, J. Nedoma, J. Ramík, J. Rohn, K. Zimmermann, *Linear Optimization Problems with Inexact Data*, Springer-Verlag, New York, 2006.
- [7] W. Oettli, On the solution set of a linear system with inaccurate coefficients, *SIAM Journal on Numerical Analysis* 2 (1965) 115–118.
- [8] K. Nickel, Die Überschätzung des Wertebereichs einer Funktion in der Intervallrechnung mit Anwendungen auf lineare Gleichungssysteme, *Computing* 18 (1977) 15–36.
- [9] J. Rohn, Systems of linear interval equations, *Linear Algebra and Its Applications* 126 (1989) 39–78.
- [10] S. Rump, INTLAB - INTerval LABoratory, in: T. Csendes (Ed.), *Developments in Reliable Computing*, Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104, <http://www.ti3.tu-harburg.de/rump/>.
- [11] J. Rohn, A general method for enclosing solutions of interval linear equations, to appear in *Optimization Letters*.
- [12] J. Rohn, An algorithm for solving the absolute value equation, *Electronic Journal of Linear Algebra* 18 (2009) 589–599, [http://www.math.technion.ac.il/iic/ela/ela-articles/articles/vol18\\_pp589-599.pdf](http://www.math.technion.ac.il/iic/ela/ela-articles/articles/vol18_pp589-599.pdf).

```

(01) function  $[Q_z, S] = \mathbf{qzmatrix}(\mathbf{A}, z)$ 
(02) % Computes either a solution  $Q_z$ 
(03) % of the equation  $QA_c - |Q|\Delta T_z = I$ ,
(04) % or a singular matrix  $S \in \mathbf{A}$ .
(05) for  $i = 1 : n$ 
(06)      $[x, S] = \mathbf{absvaleqn}(A_c^T, -T_z \Delta^T, e_i)$ ;
(07)     if  $S \neq []$ ,  $S = S^T$ ;  $Q_z = []$ ; return
(08)     end
(09)      $(Q_z)_{i\bullet} = x^T$ ;
(10) end
(11)  $S = []$ ;

```

Figure 2: An algorithm for computing the matrix  $Q_z$ .

- [13] 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, <http://uivtx.cs.cas.cz/~rohn/publist/absvaleqnreport.pdf> (January 2010).
- [14] J. Rohn, VERINTERVALHULL: Verified interval hull of the solution set of a system of interval linear equations, <http://uivtx.cs.cas.cz/~rohn/matlab/verintervalhull.html> (2008).

```

(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 ∈ ℝn; X = 0 ∈ ℝn×n;
(05) if A is singular, S = A; return, end
(06) z = sgn(A-1b);
(07) if A + BTz is singular, S = A + BTz; return, end
(08) x = (A + BTz)-1b;
(09) C = -(A + BTz)-1B;
(10) while zjxj < 0 for some j
(11)     i = i + 1;
(12)     k = min{j | zjxj < 0};
(13)     if 1 + 2zkCkk ≤ 0
(14)         S = A + B(Tz + (1/Ckk)ekekT);
(15)         x = []; return
(16)     end
(17)     if ((k < n and rk > maxk<j rj) or (k = n and rn > 0))
(18)         x = x - X•k;
(19)         for j = 1 : n
(20)             if (|B||x|)j > 0, yj = (Ax)j/(|B||x|)j; else yj = 1; end
(21)         end
(22)         z = sgn(x);
(23)         S = A - Ty|B|Tz;
(24)         x = []; return
(25)     end
(26)     rk = i;
(27)     X•k = x;
(28)     zk = -zk;
(29)     α = 2zk/(1 - 2zkCkk);
(30)     x = x + αxkC•k;
(31)     C = C + αC•kCk•;
(32) end

```

Figure 3: An algorithm for solving an absolute value equation.