



národní
úložiště
šedé
literatury

An Algorithm for Solving the Absolute Value Inequality

Rohn, Jiří
2011

Dostupný z <http://www.nusl.cz/ntk/nusl-42774>

Dílo je chráněno podle autorského zákona č. 121/2000 Sb.

Tento dokument byl stažen z Národního úložiště šedé literatury (NUŠL).

Datum stažení: 04.05.2024

Další dokumenty můžete najít prostřednictvím vyhledávacího rozhraní [nusl.cz](http://www.nusl.cz) .



Institute of Computer Science
Academy of Sciences of the Czech Republic

An Algorithm for Solving the Absolute Value Inequality

Jiří Rohn

Technical report No. V-1107

21.04.2011



Institute of Computer Science
Academy of Sciences of the Czech Republic

An Algorithm for Solving the Absolute Value Inequality

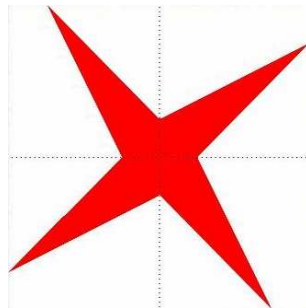
Jiří Rohn¹

Technical report No. V-1107

21.04.2011

Abstract:

Described is a not-a-priori-exponential algorithm which in a finite number of steps either finds a nontrivial solution of the inequality $|Ax| \leq |B||x|$, or states that no such solution exists.



Keywords:

Absolute value inequality, solution, algorithm.²

¹This work was supported by the Institutional Research Plan AV0Z10300504.

²Above: 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 [1])).

1 Introduction

We are interested here in finding a nontrivial solution of the inequality

$$|Ax| \leq |B||x| \tag{1.1}$$

where $A, B \in \mathbb{R}^{n \times n}$ and both the inequality as well as the absolute value are understood entrywise. As evidenced in the software package VERSOFT [4], this inequality, called an absolute value inequality, has numerous applications due to the following fundamental result:

Proposition 1. *A vector $x \neq 0$ solves (1.1) if and only if it is a null vector of some singular matrix S satisfying*

$$|S - A| \leq |B|. \tag{1.2}$$

Thus, for instance, an interval matrix $[A_c - \Delta, A_c + \Delta]$ is singular if and only if the inequality $|A_c x| \leq \Delta|x|$ has a nontrivial solution. Since the problem of checking singularity of interval matrices is NP-complete [2], it follows that the problem of checking existence of a nontrivial solution of (1.1) is NP-complete as well.

In this report we bring a rather complicated algorithm for finding a nontrivial solution of (1.1), which has two basic advantages. First, it is not-a-priori-exponential; in fact, it is capable of solving even problems with large matrices in acceptable time, depending on the data structure. Second, in infinite precision arithmetic it always produces full answer: it either finds a nontrivial solution to (1.1), or it proves that no such solution exists.

The algorithm is presented in self-contained form (i.e., with all its subalgorithms) in Section 3.³ In Section 2 we give its overall description and we prove a finite termination theorem.

2 Description

Full description of the algorithm appears in Section 3 (Figs. 3.1 through 3.4). In fact, it is a hierarchy of algorithms working in this way:

absvalineq calls **singreg**,

singreg calls **intervalhull**,

intervalhull calls **qzmatrix** and **absvaleqn**.

The algorithm **singreg** is described in [6], **intervalhull** and **qzmatrix** in [5], and **absvaleqn** in [3], [7]. Hence we are left with explanation of the behavior of the main algorithm **absvalineq** (Fig. 3.1).

Theorem 2. *For any pair of matrices $A, B \in \mathbb{R}^{n \times n}$ the algorithm **absvalineq** (Fig. 3.1) in a finite, but not-a-priori-exponential number of steps either finds a nontrivial solution of the inequality $|Ax| \leq |B||x|$ (the case of $x \neq []$), or states that no such solution exists (the case of $x = []$).*

³It is placed at the rear of the paper in order not to be intertwined with the text.

Proof. As it can be seen from Fig. 3.1, line (04), the function **absvalineq** applies the subfunction **singreg** to the interval matrix $[A - |B|, A + |B|]$. According to the main result in [6], this subfunction in a finite, but not-a-priori-exponential number of steps either finds a singular matrix S satisfying (1.2) (the case of $S \neq []$), or proves that no such matrix exists (the case of $S = []$). The rest follows from Proposition 1. ■

Example. Consider an example with two 500×500 matrices (computation has been performed on a relatively slow netbook):

```
>> tic, n=500; rand('state',1); A=2*rand(n,n)-1; B=2*rand(n,n)-1;
>> x=absvalineq(A,B); toc
```

Elapsed time is 16.832303 seconds.

```
>> isempty(x)
ans =
    0
```

Nonemptiness of x (which is too long to be displayed here) indicates that a solution has been found.

```
>> min(abs(B)*abs(x)-abs(A*x))
ans =
    8.0415
```

Positiveness of this number confirms that the vector $|B||x| - |Ax|$ is indeed nonnegative (even positive).

3 Algorithm

(01)	function $x = \text{absvalineq}(A, B)$
(02)	% $x \neq []$: x solves $ Ax \leq B x $, $x \neq 0$.
(03)	% $x = []$: $ Ax \leq B x $, $x \neq 0$ has no solution.
(04)	$S = \text{singreg}([A - B , A + B])$;
(05)	if $S \neq []$
(06)	find an $x \neq 0$ satisfying $Sx = 0$;
(07)	else
(08)	$x = []$;
(09)	end

Figure 3.1: An algorithm for solving an absolute value inequality.

```

(01) function  $S = \text{singreg}(\mathbf{A})$ 
(02) %  $S \neq []$ :  $S$  is a singular matrix in  $\mathbf{A}$ .
(03) %  $S = []$ : no singular matrix in  $\mathbf{A}$  exists.
(04)  $S = []$ ;  $n = \text{size}(\mathbf{A}, 1)$ ;  $e = (1, \dots, 1)^T \in \mathbb{R}^n$ ;
(05) if  $A_c$  is singular,  $S = A_c$ ; return, end
(06)  $R = A_c^{-1}$ ;  $D = \Delta|R|$ ;
(07) if  $D_{kk} = \max_j D_{jj} \geq 1$ 
(08)    $x = R_{\bullet k}$ ;
(09)   for  $i = 1 : n$ 
(10)     if  $(\Delta|x|)_i > 0$ ,  $y_i = (A_c x)_i / (\Delta|x|)_i$ ; else  $y_i = 1$ ; end
(11)     if  $x_i \geq 0$ ,  $z_i = 1$ ; else  $z_i = -1$ ; end
(12)   end
(13)    $S = A_c - T_y \Delta T_z$ ; return
(14) end
(15) if  $\rho(D) < 1$ , return, end
(16)  $b = e$ ;
(17)  $x = Rb$ ;  $\gamma = \min_k |x_k|$ ;
(18) for  $i = 1 : n$ 
(19)   for  $j = 1 : n$ 
(20)      $x' = x - 2b_j R_{\bullet j}$ ;
(21)     if  $\min_k |x'_k| > \gamma$ ,  $\gamma = \min_k |x'_k|$ ;  $x = x'$ ;  $b_j = -b_j$ ; end
(22)   end
(23) end
(24)  $[\mathbf{x}, S] = \text{intervalhull}(\mathbf{A}, [b, b])$ ;

```

Figure 3.2: An algorithm for finding a singular matrix in an interval matrix.

```

(01) function [x, S] = intervalhull (A, b)
(02) % Computes either the interval hull x
(03) % of the solution set of  $\mathbf{A}x = \mathbf{b}$ ,
(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)  $\mathbf{x} = [\underline{x}, \bar{x}]$ ;
(01) function  $[Q_z, S] = \text{qzmatrix}(\mathbf{A}, z)$ 
(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)    $[x, S] = \text{absvaleqn}(A_c^T, -T_z \Delta^T, e_i)$ ;
(07)   if  $S \neq []$ ,  $S = S^T$ ;  $Q_z = []$ ; return
(08)   end
(09)    $(Q_z)_{i\bullet} = x^T$ ;
(10) end
(11)  $S = []$ ;

```

Figure 3.3: 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 ∈ 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 3.4: An algorithm for solving an absolute value equation.

Bibliography

- [1] W. Barth and E. Nuding, *Optimale Lösung von Intervallgleichungssystemen*, Computing, 12 (1974), pp. 117–125. [1](#)
- [2] S. Poljak and J. Rohn, *Checking robust nonsingularity is NP-hard*, Mathematics of Control, Signals, and Systems, 6 (1993), pp. 1–9. [2](#)
- [3] 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](#)
- [4] J. Rohn, *VERSOFT: Verification software in MATLAB/INTLAB*, 2009. <http://uivtx.cs.cas.cz/~rohn/matlab>. [2](#)
- [5] J. Rohn, *An algorithm for computing the hull of the solution set of interval linear equations*, Technical Report 1074, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, April 2010. <http://uivtx.cs.cas.cz/~rohn/publist/intervalhull.pdf>. [2](#)
- [6] J. Rohn, *An algorithm for finding a singular matrix in an interval matrix*, Technical Report 1087, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, November 2010. <http://uivtx.cs.cas.cz/~rohn/publist/singreg.pdf>. [2](#), [3](#)
- [7] 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>. [2](#)