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

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 Solving the Absolute Value Inequality

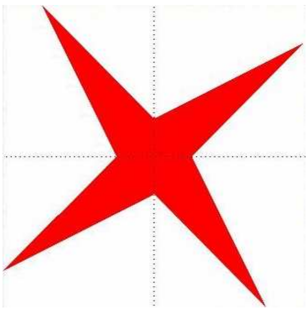
Jiří Rohn[1]

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 $|Ax| \le |B||x|$, or states that no such solution exists.



Keywords:
Absolute value inequality, solution, algorithm.[2]

---

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

[2]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 [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 $x = [\,]$).*

[3] It 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** $x =$ **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) $S =$ **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.

function $S =$ **singreg** $(\mathbf{A})$
% $S \neq [\,]$: S is a singular matrix in $\mathbf{A}$.
% $S = [\,]$: no singular matrix in $\mathbf{A}$ exists.
$S = [\,]$; $n = \text{size}(\mathbf{A}, 1)$; $e = (1, \ldots, 1)^T \in \mathbb{R}^n$;
if A_c is singular, $S = A_c$; **return**, **end**
$R = A_c^{-1}$; $D = \Delta|R|$;
if $D_{kk} = \max_j D_{jj} \geq 1$
  $x = R_{\bullet k}$;
  for $i = 1 : n$
    if $(\Delta|x|)_i > 0$, $y_i = (A_c x)_i / (\Delta|x|)_i$; **else** $y_i = 1$; **end**
    if $x_i \geq 0$, $z_i = 1$; **else** $z_i = -1$; **end**
  end
  $S = A_c - T_y \Delta T_z$; **return**
end
if $\varrho(D) < 1$, **return**, **end**
$b = e$;
$x = Rb$; $\gamma = \min_k |x_k|$;
for $i = 1 : n$
  for $j = 1 : n$
    $x' = x - 2 b_j R_{\bullet j}$;
    if $\min_k |x'_k| > \gamma$, $\gamma = \min_k |x'_k|$; $x = x'$; $b_j = -b_j$; **end**
  end
end
$[\mathbf{x}, S] =$ **intervalhull** $(\mathbf{A}, [b, b])$;

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

function $[\mathbf{x}, S] =$ **intervalhull** $(\mathbf{A}, \mathbf{b})$
% Computes either the interval hull $\mathbf{x}$
% of the solution set of $\mathbf{A}x = \mathbf{b}$,
% or a singular matrix $S \in \mathbf{A}$.
$\mathbf{x} = [\,]$; $S = [\,]$;
if A_c is singular, $S = A_c$; **return**, **end**
$x_c = A_c^{-1} b_c$; $z = \mathrm{sgn}(x_c)$; $\underline{x} = x_c$; $\overline{x} = x_c$;
$Z = \{z\}$; $D = \emptyset$;
while $Z \neq \emptyset$
  select $z \in Z$; $Z = Z - \{z\}$; $D = D \cup \{z\}$;
  $[Q_z, S] =$ **qzmatrix** $(\mathbf{A}, z)$;
  if $S \neq [\,]$, $\mathbf{x} = [\,]$; **return**, **end**
  $[Q_{-z}, S] =$ **qzmatrix** $(\mathbf{A}, -z)$;
  if $S \neq [\,]$, $\mathbf{x} = [\,]$; **return**, **end**
  $\overline{x}_z = Q_z b_c + |Q_z|\delta$;
  $\underline{x}_z = Q_{-z} b_c - |Q_{-z}|\delta$;
  if $\underline{x}_z \leq \overline{x}_z$
    $\underline{x} = \min(\underline{x}, \underline{x}_z)$; $\overline{x} = \max(\overline{x}, \overline{x}_z)$;
    for $j = 1 : n$
      $z' = z$; $z'_j = -z'_j$;
      if $((\underline{x}_z)_j (\overline{x}_z)_j \leq 0$ **and** $z' \notin Z \cup D)$
        $Z = Z \cup \{z'\}$;
      end
    end
  end
end
$\mathbf{x} = [\underline{x}, \overline{x}]$;
function $[Q_z, S] =$ **qzmatrix** $(\mathbf{A}, z)$
% Computes either a solution Q_z
% of the equation $QA_c - |Q|\Delta T_z = I$,
% or a singular matrix $S \in \mathbf{A}$.
for $i = 1 : n$
  $[x, S] =$ **absvaleqn** $(A_c^T, -T_z \Delta^T, e_i)$;
  if $S \neq [\,]$, $S = S^T$; $Q_z = [\,]$; **return**
  end
  $(Q_z)_{i\bullet} = x^T$;
end
$S = [\,]$;

Figure 3.3: An algorithm for computing the interval hull.

function $[x, S] =$ **absvaleqn** (A, B, b)
% Finds either a solution x to $Ax + B|x| = b$, or
% a singular matrix S satisfying $|S - A| \leq |B|$.
$x = [\,];\ S = [\,];\ i = 0;\ r = 0 \in \mathbb{R}^n;\ X = 0 \in \mathbb{R}^{n \times n};$
if A is singular, $S = A$; **return**, **end**
$z = \mathrm{sgn}(A^{-1}b);$
if $A + BT_z$ is singular, $S = A + BT_z$; **return**, **end**
$x = (A + BT_z)^{-1}b;$
$C = -(A + BT_z)^{-1}B;$
while $z_j x_j < 0$ for some j
  $i = i + 1;$
  $k = \min\{j \mid z_j x_j < 0\};$
  if $1 + 2z_k C_{kk} \leq 0$
    $S = A + B(T_z + (1/C_{kk})e_k e_k^T);$
    $x = [\,]$; **return**
  end
  if $((k < n$ **and** $r_k > \max\limits_{k<j} r_j)$ **or** $(k = n$ **and** $r_n > 0))$
    $x = x - X_{\bullet k};$
    for $j = 1 : n$
      if $(|B||x|)_j > 0,\ y_j = (Ax)_j/(|B||x|)_j$; **else** $y_j = 1$; **end**
    end
    $z = \mathrm{sgn}(x);$
    $S = A - T_y|B|T_z;$
    $x = [\,]$; **return**
  end
  $r_k = i;$
  $X_{\bullet k} = x;$
  $z_k = -z_k;$
  $\alpha = 2z_k/(1 - 2z_k C_{kk});$
  $x = x + \alpha x_k C_{\bullet k};$
  $C = C + \alpha C_{\bullet k} C_{k \bullet};$
end

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