



národní
úložiště
šedé
literatury

An Algorithm for Solving the Absolute Value Equation: An Improvement

Rohn, Jiří
2010

Dostupný z <http://www.nusl.cz/ntk/nusl-41158>

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

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

Datum stažení: 29.09.2024

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 Equation:
An Improvement**

Jiří Rohn

Technical report No. V-1063

20.01.2010



Institute of Computer Science
Academy of Sciences of the Czech Republic

An Algorithm for Solving the Absolute Value Equation: An Improvement

Jiří Rohn¹

Technical report No. V-1063

20.01.2010

Abstract:

Presented is an algorithm which for each $A, B \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ in a finite number of steps either finds a solution of the equation $Ax + B|x| = b$, or finds a singular matrix S satisfying $|S - A| \leq |B|$.

Keywords:

Absolute value equation, algorithm, singularity.

¹Supported by the Czech Republic Grant Agency under grants 201/09/1957 and 201/08/J020, and by the Institutional Research Plan AV0Z10300504.

1 Introduction

In our earlier paper [1] we presented an algorithm (Fig. 3.1 below) which for each $A, B \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ in a finite number of steps either finds a solution of the absolute value equation

$$Ax + B|x| = b, \quad (1.1)$$

or states existence of a singular matrix S satisfying

$$|S - A| \leq |B|, \quad (1.2)$$

and, in most cases, also finds such an S . The cases when existence of a matrix S satisfying (1.2) is stated, but S itself is not found, are extremely rare, but they still exist. Among 100,000 randomly generated 5×5 examples, the author has found only *one* example of such type, namely the one given in Section 5. In this paper we present an improvement of the previous algorithm (Fig. 4.1) which eliminates occurrences of the above-described situations. The improved algorithm for each data $A, B \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ in a finite number of steps either finds an x satisfying (1.1), or finds a singular matrix S satisfying (1.2). As we shall see, an essential redesigning of a part of the algorithm was necessary to achieve this purpose.

We use the following notations. $A_{k\bullet}$ and $A_{\bullet k}$ denote the k th row and the k th column of a matrix A , respectively. Matrix inequalities, as $A \leq B$ 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, e_k is the k 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 ± 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 $\text{sgn}(x)$ by

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

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

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

2 Auxiliary result

The following result gives an explicit way how to construct a required singular matrix S under circumstances that may occur during the algorithm.

Proposition 1. *Let*

$$(A + BT_{z'})x' = (A + BT_{z''})x''$$

hold for some $z', z'' \in Y_n$ and $x' \neq x''$ such that for each ℓ , $z'_\ell z''_\ell = -1$ implies $x'_\ell x''_\ell \leq 0$. Then for $x = x' - x''$ the matrix

$$S = A - T_y |B| T_z, \quad (2.1)$$

where y is given by

$$y_j = \begin{cases} (Ax)_j / (|B||x|)_j & \text{if } (|B||x|)_j > 0, \\ 1 & \text{if } (|B||x|)_j = 0 \end{cases} \quad (j = 1, \dots, n) \quad (2.2)$$

and

$$z = \text{sgn}(x), \quad (2.3)$$

is a singular matrix satisfying $|S - A| \leq |B|$ and $Sx = 0$.

Proof. From the proof of Proposition 2.4 in [1] it follows that under our assumptions on z', x', z'', x'' there holds

$$|Ax| \leq |B||x|,$$

where $x = x' - x''$, and Corollary 2.3 in [1] implies that the matrix S constructed by (2.1), (2.2) and (2.3) is singular and satisfies $|S - A| \leq |B|$ and $Sx = 0$. \square

3 The former algorithm

In [1] we proposed the **signaccord** algorithm (Fig. 3.1). It was supported there by the following theorem.

Theorem 2. *For each $A, B \in \mathbb{R}^{n \times n}$ and each $b \in \mathbb{R}^n$, the **signaccord** algorithm (Fig. 3.1) in a finite number of steps either finds a solution x of the equation (1.1), or states existence of a singular matrix S satisfying (1.2) (and, in most cases, also finds such an S).*

As we shall see in Section 5, the possibility of stating existence of a singular matrix without actually finding such a matrix is not excluded. This may happen if the condition

$$\log_2 p_k > n - k \quad (3.1)$$

is satisfied at some step; then the algorithm terminates in the fourth **if ... end** statement without having constructed a singular matrix S satisfying $|S - A| \leq |B|$ (although its existence is guaranteed).

```

function [x, S, flag] = signaccord (A, B, b)
% Finds a solution to Ax + B|x| = b, or states
% singularity of [A - |B|, A + |B|].
x = []; S = []; flag = 'singular';
if A is singular, S = A; return, end
p = 0 ∈ ℝn;
z = sgn(A-1b);
if A + BTz is singular, S = A + BTz; return, end
x = (A + BTz)-1b;
C = -(A + BTz)-1B;
while zjxj < 0 for some j
    k = min{j | zjxj < 0};
    if 1 + 2zkCkk ≤ 0
        S = A + B(Tz + (1/Ckk)ekekT);
        x = [];
        return
    end
    pk = pk + 1;
    if log2pk > n - k, x = []; return, end
    zk = -zk;
    α = 2zk/(1 - 2zkCkk);
    x = x + αxkC•k;
    C = C + αC•kCk•;
end
flag = 'solution';

```

Figure 3.1: The former **signaccord** algorithm from [1].

4 The improved algorithm

Here we describe the improved algorithm **absvaleqn** (Fig. 4.1) which (in infinite precision arithmetic) gives a result for any data.

Theorem 3. *For each $A, B \in \mathbb{R}^{n \times n}$ and each $b \in \mathbb{R}^n$, the algorithm **absvaleqn** (Fig. 4.1) in a finite number of steps either finds a solution x of the equation (1.1), or finds a singular matrix S satisfying (1.2).*

The improvement is placed in between the lines (17) and (25) of the algorithm where a previously missing singular matrix S is constructed along the lines of Proposition 1. The newly added variable r provides for finite termination of the algorithm. The proof is omitted here, but it can be inferred from the proof of Theorem 3.1 in [1].

5 Example

The following randomly generated example was mentioned in the Introduction.

```
A =
  78.2134  -31.1765   60.6102  -37.0822   56.8726
  58.2907   43.4605   19.6398   -9.8557   78.7528
  70.4107  -10.3979  -91.2714   76.0946   63.0426
 -87.0915  -40.7813   43.1212   18.4124   66.3227
 -15.8190  -97.4141   84.0572  -17.1518   71.9448
B =
  48.7043  -11.4057  -45.9936  -32.0912  -48.5738
 -17.5735  -30.9182   46.6939   -5.8549   5.7216
  34.4625   4.9679   5.6077  -42.2342  -32.9722
  27.4187  43.0308   8.4773   38.7742   -6.8549
 -45.7192  -18.8891   32.3623   9.3232  -15.2663
b =
  34.9380
  81.5419
 -19.1015
  89.3878
  -5.9995
```

Running the former algorithm in MATLAB, we obtain

```
>> [x,S,flag]=signaccord(A,B,b)
x =
     []
S =
     []
flag =
interval matrix singular
```

The reason for the premature termination is the fact that after the seventh iteration we have $p_5 = 2$, hence the condition (3.1) is satisfied and the algorithm exits the **while** loop.

On the contrary, the improved algorithm, also after seven iterations, produces a singular matrix.

```

>> [x,S]=absvleqn(A,B,b)
x =
    []
S =
    29.5091   -19.7708    29.6467   -4.9910    8.2988
    75.8642    74.3787    51.0747   -4.0008   84.4744
    35.9482   -15.3658   -87.4962   118.3288   30.0704
   -114.5102  -83.8121    48.8282   -20.3618   59.4678
    29.9002   -78.5250   105.8439  -26.4750   56.6785

```

We can check that the computed matrix S satisfies (1.2) (up to rounding errors) and is rank deficient.

```

>> abs(B)-abs(S-A)
ans =
     0         0   15.0301         0         0
  0.0000  -0.0000   15.2590         0   0.0000
     0         0    1.8325         0         0
     0         0    2.7703         0   -0.0000
     0         0   10.5756         0   -0.0000

>> rank(S)
ans =
     4

```



```

(01) function  $[x, S] = \text{absvaleqn}(A, B, b)$ 
(02) % Finds either a solution  $x$  to  $Ax + B|x| = b$ , or
(03) % a singular matrix  $S$  satisfying  $|S - A| \leq |B|$ .
(04)  $x = []$ ;  $S = []$ ;  $i = 0$ ;  $r = 0 \in \mathbb{R}^n$ ;  $X = 0 \in \mathbb{R}^{n \times n}$ ;
(05) if  $A$  is singular,  $S = A$ ; return, end
(06)  $z = \text{sgn}(A^{-1}b)$ ;
(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)      $k = \min\{j \mid z_j x_j < 0\}$ ;
(13)     if  $1 + 2z_k C_{kk} \leq 0$ 
(14)          $S = A + B(T_z + (1/C_{kk})e_k e_k^T)$ ;
(15)          $x = []$ ; return
(16)     end
(17)     if  $((k < n \text{ and } r_k > \max_{k < j} r_j) \text{ or } (k = n \text{ and } r_n > 0))$ 
(18)          $x = x - X_{\bullet k}$ ;
(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)          $z = \text{sgn}(x)$ ;
(23)          $S = A - T_y |B| T_z$ ;
(24)          $x = []$ ; return
(25)     end
(26)      $r_k = i$ ;
(27)      $X_{\bullet k} = x$ ;
(28)      $z_k = -z_k$ ;
(29)      $\alpha = 2z_k / (1 - 2z_k C_{kk})$ ;
(30)      $x = x + \alpha x_k C_{\bullet k}$ ;
(31)      $C = C + \alpha C_{\bullet k} C_{k \bullet}$ ;
(32) end

```

Figure 4.1: The improved algorithm `absvaleqn`.

Bibliography

- [1] J. Rohn. An algorithm for solving the absolute value equation. *Electronic Journal of Linear Algebra*, 18:589–599, 2009.