Every Two Square Matrices of the Same Size Have Some Solution in Common

Rohn, Jiří
2011

Institute of Computer Science
Academy of Sciences of the Czech Republic

Every Two Square Matrices of the Same Size Have Some Solution in Common

Jiří Rohn

Technical report No. V-1106

01.04.2011

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

Every Two Square Matrices of the Same Size Have Some Solution in Common

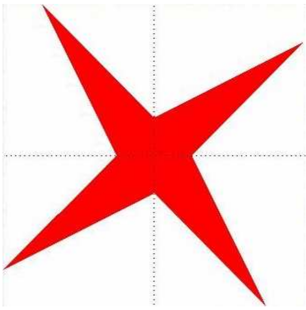
Jiří Rohn[1]

Technical report No. V-1106

01.04.2011

Abstract:

We describe a MATLAB file (only 23 lines long) for computing solution of a two-matrix alternative. Given two square matrices $A, B \in \mathbb{R}^{n\times n}$, it computes a nontrivial solution either to $|Ax| \leq |B||x|$, or to $|Ax| > |B||x|$.



Keywords:
Two-matrix alternative, 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

In [2], Corollary 4.1, we proved the following result which we call a "two-matrix alternative":

Theorem 1. *For each $A, B \in \mathbb{R}^{n\times n}$, at least one of the inequalities*

$$|Ax| \le |B||x|, \tag{1.1}$$

$$|Ax| > |B||x| \tag{1.2}$$

has a nontrivial solution.

Here, both the absolute value as well as the two types of inequalities are understood entrywise. This is a little bit surprising result, considering the full generality of the data, and can be verbally interpreted as "every two square matrices of the same size have some solution in common", thereby justifying the title of the report.

The theoretical proof given in [2] gives little clue as to how to compute a solution of either (1.1), or (1.2). In this report we describe a MATLAB file for solving this problem, available at http://uivtx.cs.cas.cz/∼rohn/matlab/others/twomatralt.m. The algorithm not only finds a nontrivial solution of (1.1) or (1.2), but it itself is also nontrivial despite consisting of only 23 lines of the source code.

2 Description

The file is invoked by

```
[xle,xgt,iter]=twomatralt(A,B) .
```

Here `xle` is the solution of (1.1), if found (solution of the Less or Equal inequality), `xgt` is the solution of (1.2), if found (solution of the Greater Than inequality), and `iter` is the number of iterations. The algorithm proceeds by solving the absolute value equation

$$Ax - |B||x| = e, \tag{2.1}$$

where e is the vector of all ones, using the embedded file `absvaleqn1063s`. If a solution x is found, then $Ax = |B||x| + e > |B||x| \ge 0$, hence $Ax > 0$, so that $|Ax| = Ax > |B||x|$ and we can put `xgt=x`. If x is not found, then `absvaleqn1063s` outputs a singular matrix S satisfying $|A - S| \le |B|$, see [3]. If we take any $x \ne 0$ with $Sx = 0$, then $|Ax| = |Ax - Sx| \le |A - S||x| \le |B||x|$ and we can put `xle=x`. In infinite precision arithmetic the inner algorithm always outputs exactly one nonempty variable, either x or S, which guarantees that in this case exactly one of `xgt`, `xle` is found. This may not to be the case of finite precision computations where it may happen that `xle=xgt=[]` (for a finite-precision matrix S regarded as singular by the algorithm it may be `null(S)=[]` in MATLAB, so that a null vector cannot be found), but in fact this feature has not been observed in practice so far.

3 Examples

We illustrate the behavior of the algorithm on two 500×500 randomly generated examples that can be rerun because `rand('state',i)` is used (`i=1` in the first example and `i=2` in the second one).

```
>> tic, n=500; rand('state',1); A=2*rand(n,n)-1; B=(1/n)*(2*rand(n,n)-1);...
>> [xle,xgt,iter]=twomatralt(A,B); toc
Elapsed time is 8.596867 seconds.
>> if ~isempty(xgt), resxgt=min(abs(A*xgt)-abs(B)*abs(xgt)),...
>> else resxle=min(abs(B)*abs(xle)-abs(A*xle)), end
resxgt =
    1.0000
```

Here a solution `xgt` has been found. The positivity of `resxgt` confirms that it really solves (1.2); the solution couldnot be written down here for obvious space reasons.

```
>> tic, n=500; rand('state',2); A=2*rand(n,n)-1; B=(1/n)*(2*rand(n,n)-1);...
>> [xle,xgt,iter]=twomatralt(A,B); toc
Elapsed time is 23.173219 seconds.
>> if ~isempty(xgt), resxgt=min(abs(A*xgt)-abs(B)*abs(xgt)),...
>> else resxle=min(abs(B)*abs(xle)-abs(A*xle)), end
resxle =
    0.0128
```

Here a solution `xle` has been found. The nonnegativity of `resxle` confirms that it solves (1.1).

4 The file

Following we give a screenshot of the MATLAB editor showing the file which itself can be downloaded from http://uivtx.cs.cas.cz/~rohn/matlab/others/twomatralt.m .

```
function [xle,xgt,iter]=twomatralt(A,B) % TWO-MATRix ALTernative
% xle solves abs(A*x)<=abs(B)*abs(x), x~=0, xgt solves abs(A*x)>abs(B)*abs(x), (one empty), iter # of iterations
[xgt,S,iter]=absvaleqn1063s(A,-abs(B),ones(size(A,1),1));
if ~isempty(xgt), xle=[]; return, else xle=null(S); xle=xle(:,1); xgt=[]; return, end
function [x,S,iter]=absvaleqn1063s(A,B,b) % ABSolute VALue EQuatioN as in report 1063, short version
b=b(:); n=length(b); I=eye(n,n); ep=n*(max([norm(A,inf) norm(B,inf) norm(b,inf)]))*eps; x=[]; S=[]; iter=0;
if rank(A)<n, S=A; return, end
x=A\b; z=zeros(n,1);
for j=1:n, if x(j)<0, z(j)=-1; else z(j)=1; end, end
if rank(A+B*diag(z))<n, S=A+B*diag(z); x=[]; return, end
x=(A+B*diag(z))\b; C=-inv(A+B*diag(z))*B; X=zeros(n,n); r=zeros(1,n);
while any(z.*x<-ep)
    k=find(z.*x<-ep,1); iter=iter+1;
    if 1+2*z(k)*C(k,k)<=0, tau=(-1)/(2*z(k)*C(k,k)); S=A+B*(diag(z)-2*tau*z(k)*I(:,k)*I(k,:)); x=[]; return, end
    if ((k<n)&&(all(r(k)>r(k+1:n))))||((k==n)&&(r(k)>0))
        x=x-X(:,k);
        for j=1:n, if x(j)<0, z(j)=-1; else z(j)=1; end, end
        ct=A*x; jm=abs(B)*abs(x); y=zeros(1,n);
        for i=1:n, if jm(i)>ep, y(i)=ct(i)/jm(i); else y(i)=1; end, end
        S=A-diag(y)*abs(B)*diag(z); x=[]; return
    end
    X(:,k)=x; r(k)=iter; z(k)=-z(k); alpha=2*z(k)/(1-2*z(k)*C(k,k)); x=x+alpha*x(k)*C(:,k); C=C+alpha*C(:,k)*C(k,:);
end
```

Bibliography

[1] W. Barth and E. Nuding, *Optimale Lösung von Intervallgleichungssystemen*, Computing, 12 (1974), pp. 117–125. 1

[2] J. Rohn, *Regularity of interval matrices and theorems of the alternatives*, Reliable Computing, 12 (2006), pp. 99–105. 2

[3] 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