VERSOFT: Guide

Rohn, Jiří
2011

Institute of Computer Science
Academy of Sciences of the Czech Republic

VERSOFT: Guide

Jiří Rohn

Technical report No. V-1118

27.07.2011

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

VERSOFT: Guide

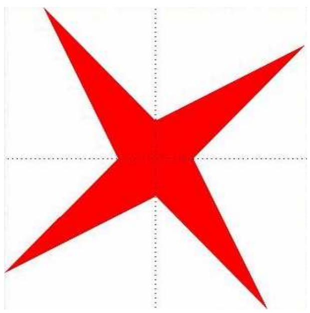
Jiří Rohn[1]

Technical report No. V-1118

27.07.2011

Abstract:

This is a very concise introduction to VERSOFT, a collection of 61 verification files for computing verified solutions of various numerical linear algebraic problems having exact or interval-valued data.



Keywords:
VERSOFT, verification software, guide.[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 WHAT IS VERSOFT?

VERSOFT is a collection of 61 verification files for computing verified solutions of various numerical linear algebraic problems having exact or interval-valued data.

2 WHERE CAN IT BE FOUND?

It is freely available at
http://uivtx.cs.cas.cz/~rohn/matlab/ .

3 WHAT DOES "VERIFIED" MEAN?

A [yes/no] result is called verified if it holds as a mathematical truth despite being achieved by computation in finite precision arithmetic. Usually the result is given in the form "$x \in [x1, x2]$", where x (be it a matrix, vector, or number) is the exact solution and $[x1, x2]$ is an interval (matrix, vector, number) which is guaranteed to contain x and whose bounds consist of floating-point numbers.

Example: verified eigenvalues of a 3-by-3 matrix.

```
>> format long, A=[1 2 3; 4 5 6; 7 8 9], lam=diag(vereig(A))
A =
     1     2     3
     4     5     6
     7     8     9
intval lam =
             [  16.11684396980703,  16.11684396980705]
             [  -0.00000000000001,   0.00000000000001]
             [  -1.11684396980705,  -1.11684396980704]
```

The three eigenvalues are verified to belong to the intervals given. Observe that the second eigenvalue, which is zero, is enclosed by a tiny interval. Hence the result does not give us the right to assert that the matrix is singular (but this can be verified by another means, see VERFULLCOLRANK).

4 FILE NAMES

All the M-file names (except one) bear the prefix VER (for VERified). The second part is very often simply the name of the corresponding MATLAB function, as e.g. VERRANK, VERDET, VERPINV, VEREIG, VERCHOL, VERQR, VERLSQ, VERLINPROG, VERQUADPROG, etc. The names not falling into this category are constructed in a similar way, as e.g. VERINVERSE, VERPOSDEF (VERified POSitive DEFiniteness), VERTHINSVD (VERified THIN SVD), VERSPECTDEC (VERified SPECTral DEComposition), VERFULLCOLRANK (VERified FULL COLumn RANK), VERMATFUN (VERified MATrix FUNction), VERLINSYS (VERified solution(s) of a LINear SYStem), VERREGSING (VERified REGularity/SINGularity of an interval matrix), VERTOLSOL (VERified TOLerance SOLution), etc.

5 INPUT DATA

There is a basic distinction between real (noninterval) data only, and interval data. Files constructed for interval data also admit real data, but not conversely. Typically, only the data of the problem are required; various control variables (as maximal number of iterations, tolerance, etc.) are built in. There is only one exception regarding files for solving NP-hard problems (as VERINTERVALHULL, VERREGSING, VERPOSDEF, etc.) that offer the possibility to input an additional time variable "t" (timing) which, when equal to 1, produces screen output of the form

```
Expected remaining time: 2502 sec.
```

which can help you to decide whether is pays off to continue the computation, or not.

6 OUTPUT DATA

The basic rule is that the output is always either verified, or consists of NaN's (of the respective size). Thus,

```
>> A=[1 2 3 4; 5 6 7 8; 9 10 11 12], X=verpinv(A)
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
intval X =
Columns 1 through 2
[  -0.37500000000001,  -0.37499999999999] [  -0.10000000000001, -0.09999999999999]
[  -0.14583333333334,  -0.14583333333333] [  -0.03333333333334, -0.03333333333333]
[   0.08333333333333,   0.08333333333334] [   0.03333333333333,  0.03333333333334]
[   0.31249999999999,   0.31250000000001] [   0.09999999999999,  0.10000000000001]
Column 3
[   0.17499999999999,   0.17500000000001]
[   0.07916666666666,   0.07916666666667]
[  -0.01666666666667,  -0.01666666666666]
[  -0.11250000000001,  -0.11249999999999]
```

computes a verified pseudoinverse of A, whereas an attempt to compute a verified QR decomposition of the same matrix fails:

```
>> format short, A=[1 2 3 4; 5 6 7 8; 9 10 11 12], [Q,R]=verqr(A)
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
intval Q =
[       NaN,        NaN] [       NaN,        NaN] [       NaN,        NaN]
[       NaN,        NaN] [       NaN,        NaN] [       NaN,        NaN]
[       NaN,        NaN] [       NaN,        NaN] [       NaN,        NaN]
```

```
intval R =
[      NaN,         NaN] [      NaN,         NaN] [      NaN,         NaN] [      NaN,         NaN]
[   0.0000,      0.0000] [      NaN,         NaN] [      NaN,         NaN] [      NaN,         NaN]
[   0.0000,      0.0000] [   0.0000,      0.0000] [      NaN,         NaN] [      NaN,         NaN]
```

(R is predefined to be upper triangular, therefore the zeros). Some files (but by far not all of them) contain an output variable E which gives some reasons for NaN output. Care should be taken of the fact that a NaN output also occurs (without further comments) when input data are wrong. For example,

```
>> A=[1 2;3 4], lam=vereigsym(A)
A =
     1     2
     3     4
intval lam =
             [       NaN,         NaN]
             [       NaN,         NaN]
```

results in NaN's because of an attempt to apply a file designed for symmetric matrices to a nonsymmetric matrix.

7 SHORTEST FILE

The shortest file is PLUSMINUSONESET, consisting of 10 code lines only. This is also the single M-file in VERSOFT not bearing the prefix VER.

8 LONGEST FILE

The longest file is INTERVALHULL, referred to by VERINTERVALHULL. It has almost 1000 lines and its construction lasted from 2003 to 2007 (being approximately as time-consuming as all the other files taken together).

9 EXAMPLES

Further examples can be found at
http://uivtx.cs.cas.cz/~rohn/matlab/examples/examples.html .

10 THE BASIC REFERENCE

For problems with real data only, there is no single basic reference. For problems with interval data you may consult the "Handbook" available at
http://uivtx.cs.cas.cz/~rohn/handbook/ .

11 THE LANGUAGE IN WHICH IT IS WRITTEN

VERSOFT is written in INTLAB, a MATLAB toolbox created by Siegfried M. Rump [2]. It can be downloaded from
http://www.ti3.tu-harburg.de/rump/intlab
(observe the license). An INTLAB primer can be found at
http://uivtx.cs.cas.cz/~rohn/matlab/primer/intlab_primer.html.

12 AUTHOR'S NOTE: M-FILES VS. P-FILES

In its current state (version 10) VERSOFT consists of 53 MATLAB/INTLAB m-files and 8 p-files, i.e., content-obscured but executable files. These eight files were made public in p-coded form because they were based on unpublished algorithms, as e.g. those for computing verified eigenvalues and eigenvectors of general complex matrices (vereig.p), computing verified singular value decomposition (verthinsvd.p), verified linear (in)dependence of vectors (zd.p), etc. This approach by the author was prompted by then-blazing affair of patent applications by two US mathematicians who had successfully patented the basic formulae of interval arithmetic and several interval algorithms, thereby creating doubts about the future of free research as such [The author cannot help but quote here Georg Cantor, "*Das Wesen der Mathematik liegt in ihrer Freiheit.*"]; the whole affair can be followed on the pages of the *reliable_computing* net.

Bibliography

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

[2] S. Rump, *INTLAB - INTerval LABoratory*, in Developments in Reliable Computing, T. Csendes, ed., Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104. http://www.ti3.tu-harburg.de/rump/. 5