User Manual of VerifyRealRoots

A MATLAB package for computing verified real solutions
of polynomial systems of equations and inequalities

Zhengfeng Yang

Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai 200062, China
zfyang@sei.ecnu.edu.cn; http: // faculty. ecnu. edu. cn/ yangzhengfeng

Lihong Zhi

Key Laboratory of Mathematics Mechanization,
Chinese Academy of Sciences,Beijing 100190, China
lzhi@mmrc.iss.ac.cn; http: // mmrc. iss. ac. cn/ ~ lzhi

1. Introduction

VerifyRealRoots is a Matlab package for computing verified real solutions of polynomial systems of equations and inequalities. Let $f_1, \ldots, f_m, g_1, \ldots, g_s$ be polynomials in $\mathbb{R}[x_1, \ldots, x_n]$, and $\mathcal{S} \subset \mathbb{R}^n$ be the semi-algebraic set defined by $f_1, \ldots, f_m, g_1, \ldots, g_s$:

$$\mathcal{S} = \{(a_1, \ldots, a_n) \in \mathbb{R}^n : f_i(a_1, \ldots, a_n) = 0, g_j(a_1, \ldots, a_n) > 0 \text{ for } 1 \leq i \leq m, 1 \leq j \leq s\}.$$

VerifyRealRoots aims for computing at least one verified real solution for the semi-algebraic set $\mathcal{S}$ using hybrid symbolic and numeric methods.

The directory in which you install VerifyRealRoots contains three subdirectories:

- `/src:` the source code of VerifyRealRoots;

- `/examples:` some benchmark examples;

- `/docs:` user manual and license file.

If you have any questions, please send email to `zfyang@sei.ecnu.edu.cn` and `lzhi@mmrc.iss.ac.cn`. Any comments or suggestions are greatly appreciated.

2. Configuration

2.1. System requirements

To install and run VerifyRealRoots, you need:

- MATLAB R2014b or later versions.

- gcc compiler

  - Windows Platform: Microsoft Windows with Microsoft Windows SDK 7.1 or Microsoft Visual C++ 2013 Compiler (Microsoft Visual Studio 2013).
  - Linux Platform: GNU C++ compiler gcc 4.4.x/4.5.x/4.6.x/4.7.x.
  - Mac OS X Platform: Xcode 5.x/6.x.

2.2. Installation instruction

You need install required software packages listed below and run the script `configure.m` to complete the configuration.

1. Download VerifyRealRoots.zip, and unpack it.

2. Download the required software packages for using VerifyRealRoots:

   - Download GloptiPoly 3.8. Unpack the zip file, and move the folder gloptipoly3 to ./VerifyRealRoots/src.

- Download the correct version of HOM4PS-2.0 for your operating system and unpack it, and move the bin folder to ./VerifyRealRoots/src/homotopy_aux and merge it to the existing bin folder.

- The three software above could be downloaded automatically in `configure.m`. If error happens while downloading automatically, it is necessary to do this manually.

3. Run `configure.m` to configure VerifyRealRoots

- Change the MATLAB current folder to VerifyRealRoots' folder, or add its folder to path, then type "configure" in Matlab Command Window.

- Type "testDemos" to check whether the installation of VerifyRealRoots is complete.

3. Computing Verified Real Solutions of Polynomial Systems

Main steps to compute verified real solutions of polynomial systems:

1. input the polynomial systems defined by equations and inequalities;

2. select the solver and initialize the option;

3. call main function `verifyrealroots` and get verified real solutions.

We also provide the online computation for VerifyRealRoots. You may try it on our website: `http://www.verifyrealroots.com`.

3.1. Polynomial systems

In VerifyRealRoots, at first we need define the variables, and then create the polynomial with the declared variables. For instance, to input a polynomial $f = x^2 + y^2$, you need type

```
>> syms x y;   % declare the variables
>> f=x^2+y^2; % input a polynomial
```

We input a polynomial system $H = \{F, G\}$, where $F = [f_1, \ldots f_m]$ contains polynomial equations $f_1 = 0, \ldots, f_m = 0$ and $G = [g_1, \ldots, g_s]$ contains polynomial inequalities $g_1 > 0, \ldots, g_s > 0$. If F or G is empty, we set $F = [\ ]$ or $G = [\ ]$.

Example 1 *Consider the following system*

$$\begin{cases} f_1 = x_1 + x_2 - 1 = 0, \\ f_2 = x_3 + x_4 - 1 = 0, \\ g_1 = 2x_1^3 - x_2 x_3 + 4x_4 - 1 > 0, \\ g_2 = x_1^2 + x_2^2 + x_3^2 - 1 > 0. \end{cases} \tag{1}$$

At first, we declare the variables x_1, x_2, x_3, x_4 by typing

```
>> syms x1 x2 x3 x4; % declare the variables
```

then construct the polynomial system by setting

```
>> f1 = x1 + x2 - 1;  %input the polynomials
>> f2 = x3 + x4 - 1;
>> g1 = 2*x1^3 - x2*x3 + 4*x4 - 1;
>> g2 = x1^2 + x2^2 + x3^2 - 1;
>> F = [f1, f2];  % input the polynomial equations
>> G = [g1, g2];  % input the inequalities
>> H = {F, G};    % construct the polynomial system
```

3.2. Option selections

The parameters of the options of VerifyRealRoots are stored in a Matlab data type structure. The structure is divided into fields which contain individual pieces of data. Two algorithms in VerifyRealRoots are provided to compute the verified real solutions for the given polynomial system. In the field option.solver, "Hom4ps" denotes the solver, which is based on the critical point method and the homotopy continuation method; "MMCRSolver" denotes the solver, which is based on the low-rank moment matrix completion method. After constructing a polynomial system, we can select one solver between "Hom4ps" and "MMCRSolver".

We list all options in Appendix. The parameter of the options will be assigned as the default value when it is unassigned.

For Example 1, we can type the following commands to set the options:

```
>> option.solver = 'MMCRSolver';  % select the solver
>> option.tol = 1e-5;             % set the tolerance
```

3.3. Getting solutions

After constructing a polynomial system and initializing option, we call the main function verifyrealroots by typing

```
>> [out, vars] = verifyrealroots(H, option); %call the main function
```

It returns the verified inclusions of real solutions and the order of the variables. For Example 1, we have

```
>> out{:}
ans =
   0.494621749849497   0.494621749849502
   0.505378250150498   0.505378250150503
   0.711471250340805   0.711471250340805
   0.288528749659195   0.288528749659195
>> vars
vars =
[ x1, x2, x3, x4]
```

The computed results can be stored in a specified file by calling the function `output` in verifyrealroots. For example, the above results are stored in a file named "result" by typing:

```
% write out and vars to a file named as 'result'  in the directory './res'.
>> output('res/result', out, vars);
```

3.4. Verification with given approximate solutions

If approximate solutions are given, we can input the approximate real solutions and the order of variables

```
>> xs={[0.49462; 0.50537; 0.71147; 0.28852],
       [-3.15811; -1.11511; 2.51531; 0.17714]};
       % input  approximate solutions
>>vars=[x1,x2,x3,x4]; % define the order of variables
```

and use the command

```
>> out = verifyrealroots(H, xs, vars); % call the main function
```

to get

```
>> out{:} % list the verified solutions

ans =
   0.494621749849497    0.494621749849502
   0.505378250150498    0.505378250150503
   0.711471250340805    0.711471250340805
   0.288528749659195    0.288528749659195
```

3.5. Try Online

VerifyRealRoots also provides the online computation. One can input the polynomial system and try verifyrealroot online via `http://www.verifyrealroots.com`. One need to edit the MATLAB script in the editor, and then run VerifyRealRoots online by clicking `Run` button on the upper-left corner of the browser. After the computation is complete, the output will appear on the right side of the browser.

Comparing with the standard installation, the online computation of verifyrealroots is much more time-consuming. To get better performance, we recommend the users to install VerifyRealRoots on their computers.

Appendix

In this section, we present the options of two solvers: Hom4ps and MMCRSolver, see [1] [2] for details.

Options	Details
demand	• *each_component*: is used for computing real solutions on each connected component of the variety defined by adding all minors. • *roots_existence* (default): construct the extended polynomial system by adding some minors.
projection	Construct a zero-dimensional polynomial system by fixing as many variables as possible. The default value is 'false'.
tol	The tolerance is selected to determine whether the complex root computed by Hom4ps is real. The default value is $1e-7$.

Table 1: The options for Hom4ps

Options	Details
order	The relaxation order. The default value is $$\max\left(\max_{1\leq i\leq m}(\lceil \deg(f_i)/2\rceil), \max_{1\leq j\leq s}(\lceil \deg(g_i)/2\rceil)\right).$$
method	Two methods to construct a square polynomial system when the original system is not square. • *nullspace*(default): use null vector of the Jacobian matrix to construct an extended regular system. • *fix*: fix some variable(s) as anchor points to construct a square system.
schur_type	Two methods for computing the Schur decomposition. • *full*: call the standard function 'schur' in MATLAB. • *partial* (default): call the function 'laneig' in PROPACK for partial Schur decomposition.
projection	Construct a zero-dimensional polynomial system by fixing as many variables as possible. The default value is 'false'.
continuation	Use Barzilai-Borwein continuation technique for solving the minimum-rank Gram matrix completion problems. The choices are '0' and '1'. The default value is '1'.
eta	The stepsize is a rational number less than 1. The default value is $1/4$.
tol	The tolerance for determining the numeric rank of the matrix.The default value is $1e-3$.

Table 2: The options for MMCRSolver

References

[1] MA, Y., AND ZHI, L. Computing real solutions of polynomial systems via low-rank moment matrix completion. In *ISSAC* (2012), ACM, pp. 249–256.

[2] YANG, Z., ZHI, L., AND ZHU, Y. Verified error bounds for real solutions of positive-dimensional polynomial systems. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation* (New York, NY, USA, 2013), ISSAC '13, ACM, pp. 371–378.