






The *IRIT* multivariate solver - matlab interface

General Information


This is a matlab interface to the multivariate polynomial solver of IRIT. This solver is for real roots of sets of non-linear polynomial equations with the following properties:

1.  All real solutions that can be isolated from their neighbors by the subdivision tolerance (i.e. the distance between the two real solution is larger than that tolerance) are guaranteed to be detected and extracted.
2.  Singularities (higher order zero contacts) are not handled but the location of the singularity can be isolated to within the subdivision tolerance.
3.  Solution space can be zero dimensional (n equations in n variables) in which case a list of solution points in \mathbb{R}^n will be returned.
4.  Solution space can be one dimensional (n equations in $n+1$ variables) in which case a list of solution polylines in $\mathbb{R}^{(n+1)}$ will be returned.
5.  All variables must be of the form x_1 to x_{18} .

In practice, the time complexity depends on n , the number of equations, on the degrees of the polynomials and the size and type (zero- or one-dimensional) of the solution space. As a rule of thumb and assuming a small (<10) number of solutions, for $n < 5$ and degrees no more than 10, solution should be extract in less than a second. If all constraints are quadratic, interactive rate can be achieved also for $n < 10$. Among other choices, the user may choose the domain in which solutions are searched, as well as the subdivision tolerance (the domain size at which subdivision is terminated, if no other termination condition is fulfilled, i.e. when two roots are very close to each other or at a higher order contact). Remember that the ratio between the domain size and the subdivision tolerance has effect on the run-times. By default the domain is $[0, 1]$ in each dimension and the subdivision tolerance is 0.001, giving a reasonable default ratio. However, if the required domain is much different than $[0,1]$, by several orders of magnitude, consider reformulating the problem, as it shall be a more numerically robust to solve the problem in the domain $[0, 1]$ or close to it.

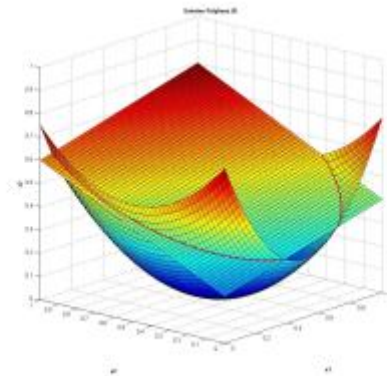
Examples


Click on the following images for a full size picture In all examples the domain is $[0,1]^n$ unless otherwise stated. Rough time estimates are for a modern Windows workstations as of 2014.

1.  (Univariate) Solution of two equations in (x_1, x_2, x_3) - intersection of a plane and a Paraboloid (a fraction of a second to solve):

$$2(x_1 - 0.5)^2 + (x_2 - 0.5)^2 - x_3 = 0$$

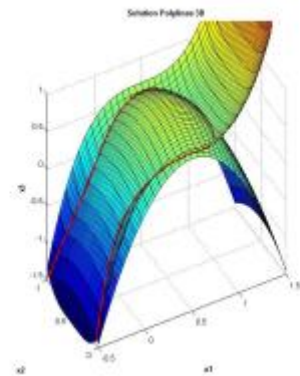
$$0.2 * x_1 + 0.4 * x_2 - x_3 + 0.2 = 0$$




2.  (Univariate) Solution of two equations in (x_1, x_2, x_3) , and domain $[-0.5, 1.5] \times [0, 1] \times [-1.5, 1]$ - intersection of a general quartic by cubic surface with a paraboloid (a few seconds to solve):

$$2(x_1 - 0.5)^3 + 5(x_2 - 0.5)^4 - x_3 + 0.35 = 0$$

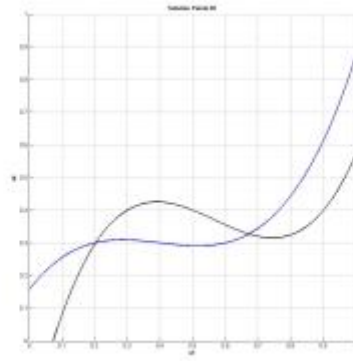
$$-(2 * (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.8) - x_3 = 0$$




3.  (Zero dimensional) Solution of two equations in (x_1, x_2) - intersection of two cubic curves (a fraction of a second to solve):

$$3(x_1 - 0.2) * (x_1 - 0.4) * (x_1 - 0.6) + 0.3 - x_2 = 0$$

$$5(x_1 - 0.3) * (x_1 - 0.5) * (x_1 - 0.9) + 0.4 - x_2 = 0$$

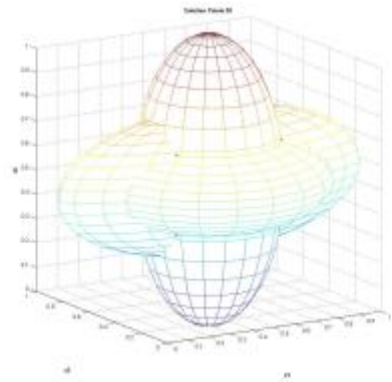



4.  (Zero dimensional) Solution of intersecting three ellipsoids in (x_1, x_2, x_3) (a fraction of a second to solve):

$$((x_1-0.5)^2) / 0.3^2 + ((x_2-0.5)^2) / 0.3^2 + ((x_3-0.5)^2) / 0.6^2 - 1 = 0$$

$$((x_1-0.5)^2) / 0.7^2 + ((x_2-0.5)^2) / 0.2^2 + ((x_3-0.5)^2) / 0.3^2 - 1 = 0$$

$$((x_1-0.5)^2) / 0.5^2 + ((x_2-0.5)^2) / 0.5^2 + ((x_3-0.5)^2) / 0.2^2 - 1 = 0$$



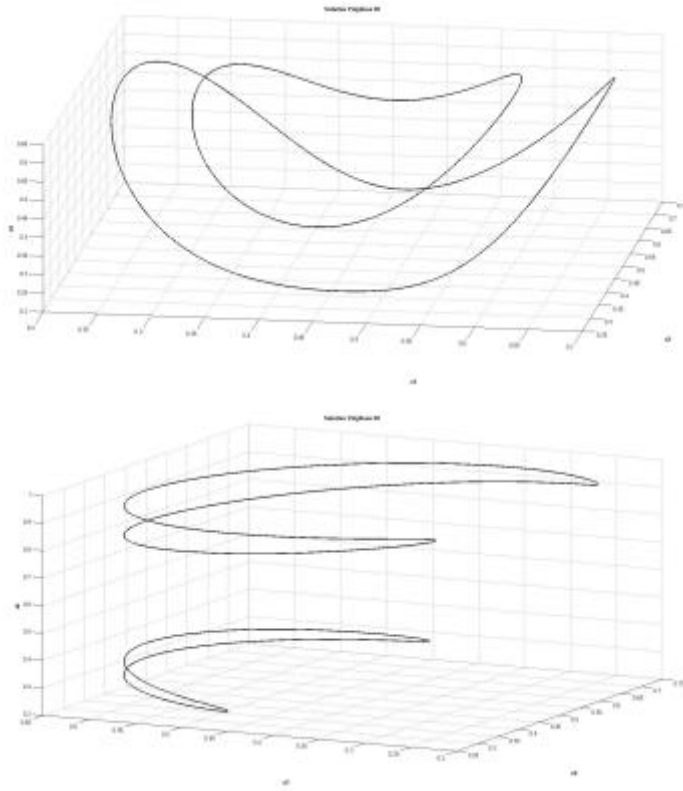
5.  (Univariate) Solution of intersecting three surfaces. The first is the Ding-Dong surface and the other two are quadratic hyper-surfaces in (x_1, x_2, x_3, x_4) (a few seconds to solve):

$$(x_1-0.5)^2 + (x_2-0.5)^2 + (x_3-0.5)^2 - (1-x_4+0.5)*(x_4-0.5)^2 = 0$$


$$(x_2-0.5)^2 + (x_3-0.4)^2 - 1/16 = 0$$

$$(x_1-0.4)^2 + (x_4-0.6)^2 - 1/9 = 0$$

Shown are projections of the univariate solutions to \mathbb{R}^3 , with left figure featuring dimensions (1,2,3) and right figure dimensions (2,3,4):





Installation

1.  Download [this zip file](#). This zip contains matlab code at the source level and the [IRIT](#) modeling kernel, written in C and compiled (on VS2008) as a single DLL, for both 32 bits and 64 bits. See more on IRIT below, including all source code.


Remark: Standard Matlab 2013b version we used for testing provides the C runtime environments for VS2008 enabling the use of the IRIT compiled DLL as is. If, for some reason, your Matlab installation does not provide the VS2008 run time, you might also be required to download and install (run) once the [Microsoft VS 2008 run time environment](#) to execute the IRIT DLLS and run this solver.


The zip file contains:

Main Functions:

-  IritPolynomialSolver.m - Matlab interface to the polynomial equations' solver.
-  IritPlotSolution.m - Matlab file to plot the solution generated by the solver.



Other files and directories:

-  irit*.dll - the necessary Irit compiled dll files for running the mex, for both 32 bits and 64 bits.

 IritPolynomialSolverMex.mexw32 and IritPolynomialSolverMex.mexw64 - The compiled Mex engine for the polynomial solver, for 32bit or 64 bit architecture, respectively. The mex files were built using Matlab versions 2011b (for 64bit) and 2012a (for 32bit), and also tested on 2013 versions.

 IritPolynomialSolverTest.m - Test program of the solver.

 readme.txt - Simple help, similar to this.

2.  Unpack the IritPolynomialSolver.zip file at a new directory of your selection, called "main directory" hereafter.
3.  In MATLAB (under the "Current Folder" section), right click on the IritPolynomialSolver main directory and select "Add to Path -> Selected Folders and Subfolders". This will enable you to use the Irit solver, regardless of the location of the Current Folder. For this to hold in future Matlab sessions, select "Set path" from the Matlab menu (depending on your Matlab version) and "Save".

Basic Use

There are two matlab functions in the interface for the solver:

1. IritPolynomialSolver for procuding the solution, and
2. IritPlotSolution for plotting it.

To see documentation of these functions, run:

1. > help IritPolynomialSolver
2. > help IritPlotSolution

To see examples of the solver in action, run the script:

1. > IritPolynomialSolverTest

For examples on how to use the solver, check out the example functions in IritPolynomialSolverTest.m.

Licensing

BECAUSE IRIT AND ITS SUPPORTING INTERFACE TOOLS AS DOCUMENTED IN THIS DOCUMENT ARE LICENSED FREE OF CHARGE FOR NON COMMERCIAL USE, I PROVIDE ABSOLUTELY NO WARRANTY, TO THE EXTENT PERMITTED BY APPLICABLE STATE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING, I [GERSHON ELBER](#) PROVIDE THE IRIT PROGRAM AND ITS SUPPORTING TOOLS "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND

FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THESE PROGRAMS IS WITH YOU. SHOULD THE IRIT PROGRAMS PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL GERSHON ELBER, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR A FAILURE OF THE PROGRAMS TO OPERATE WITH PROGRAMS NOT DISTRIBUTED BY GERSHON ELBER) THE PROGRAMS, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

IRIT is a freeware geometric solid modeler. It is not public domain since we hold copyrights on it. However, unless you are to sell or attempt to make money from any part of this code and/or any result you got with this environment, you are free to make anything you want with it. In order to use IRIT commercially, you must license it first - contact us in such a case.

More Information

More on the foundations of the IRIT multivariate solver can be found in the following publications:

1. Gershon Elber and Myung-Soo Kim. "Geometric Constraint Solver using Multivariate Rational Spline Functions." The Sixth ACM/IEEE Symposium on Solid Modeling and Applications, Ann Arbor, Michigan, pp 1-10, June 2001.
2. Iddo Hanniel and Gershon Elber. "Subdivision Termination Criteria in Subdivision Multivariate Solvers." Computer Aided Design, Vol 39, pp 369-378, 2007.
3. Michael Barton, Gershon Elber, and Iddo Hanniel. "Topologically Guaranteed Univariate Solutions of Underconstrained Polynomial Systems via No-loop and Single-component Tests." Computer Aided Design, Vol 43, No 8, pp 1035-1044, August 2011.
4. Jonathan Mizrahi and Gershon Elber. "Optimizing Subdivision Solvers of Piecewise Polynomial Systems." The XIV IMA conference on Mathematics of Surfaces, Birmingham, England, September 2013.

You can find more on IRIT geometric modeling environment, in which the multivariate solver is developed, in [here](#). Contact us at

gershon@cs.technion.ac.il

Acknowledgement

This interface was created with the help of Jonathan Mizrahi and Ron Zisman.