Bertini_Real

User's Manual

Manual by
Pierce Cunneen & Daniel Brake
University of Notre Dame
ACMS

July 24, 2015

Contents

1 Introduction

Welcome to Bertini_real, software for real algebraic geometry. This manual is intended to help the user operate this piece of numerical software, to obtain useful and high-quality results from decomposing real algebraic curves and surfaces.

Bertini_real is compiled software, links against a parallel version of Bertini 1 compiled as a library, and requires Matlab and the Symbolic Computation toolbox. It also requires several other libraries, including a few from Boost, and an installation of MPI. All libraries should be compiled using the same compilers.

1.1 Contact

Acknowledgements

- This research utilized the CSU ISTeC Cray HPC System supported by NSF Grant CNS-0923386.
- This material is based upon work supported by the National Science Foundation under Grants No. DMS-1025564 and DMS-1115668.

1.2 License

2 Quick Start

Bertini_real can be downloaded from http://bertinireal.com/download.html. Use of Bertini_real depends on Bertini, which itself has several important dependencies (see section 3) Once installed, you can run Bertini_real on an input file from the command line. After navigating to the working directory of the input file, the flow of Bertini_real is as follows:

1. Run Bertini on an input file using the "tracktype: 1" setting. This is done by typing in the command line: `bertini` with an input file named 'input'. Bertini will produce a Numerical Irreducible Decomposition that will be used by Bertini_real.

2. Run Bertini_real on the same input file. Similarly, just type `bertini_real` in the command line. Bertini_real will provide a cellular decomposition of the real portion of a one- or two-dimensional complex algebraic set.

3. Visualize the results of Bertini_real in MATLAB. Enter MATLAB and call `gather_br_samples`, which parses the output results of into a .mat file, and then call `bertini_real_plotter`, which will plot the curve or surface in MATLAB (N.B. The MATLAB executable must be on the path to the input file for Bertini_real to run).

3 Compilation and Installation

3.1 Installation

Before installing Bertini_real, you must first be sure to have several libraries and dependencies that the software requires.

First, you must install Bertini (as a library). The Bertini source code can be found at https://bertini.nd.edu/download.html. Download the Bertini source code using the ./configure && make && make install process.

Bertini itself has the following dependencies: a C++ compiler capable of the C++ 11 standard, an MPI (such as MPICH2), Boost >= 1.53, MPFR, and GMP. Instructions specifically for mac users are listed below. If on Linux, use the package manager provided (e.g. apt-get). Unfortunately, Windows users are unsupported at this time, except possibly through Cygwin or a virtual machine. If interested in porting Bertini and Bertini_real to windows, please contact Dr. Brake at dbrake@nd.edu. Bertini_real also is dependent on MATLAB. Once Bertini and all the necessary dependencies are installed, navigate to the directory containing Bertini_real and install Bertini_real via the ./configure && make && make install process.

3.2 Installation of Bertini/Bertini_real on macIntosh

If you are using a mac, we encourage the use of Homebrew (http://brew.sh) to install these packages. After installing Homebrew itself, installing the previously listed dependencies becomes simple. In terminal, just type, `brew search` ____ to list packages related to ____, where ____ is your search (for example, GMP, Boost, or MPICH2). To download via Homebrew, type in terminal: `brew install` ____.

4 Using Bertini_Real

4.1 Input File

Running Bertini_real starts with creating a valid input file for Bertini to run on. With regards to Bertini_real, the important point about the input file is the tracktype setting. The input file must have the following configuration setting: `tracktype:1`, which specifies a basic positive dimensional run for Bertini. Using this setting allows for the necessary output files to be produced for Bertini_real to run. For more information on the structure and syntax of input files, please see the Bertini User Manual (https://bertini.nd.edu/BertiniUsersManual.pdf).

4.2 Running Bertini

Once a valid input file has been created, Bertini can be run on the input file. This simply amounts to typing on the command line: `bertini filename`. If the input file is named `input`, then no filename is needed. Running Bertini will produce the witness points, the points on the curve or surface, which Bertini_real will later track. Running Bertini also produces the witness linears which are used in the regeneration and slicing steps of the algorithm. The necessary file for Bertini_real is called `witness_data`.

4.3 Running Bertini_real

The next step is to call Bertini_real on the command line. This is done simply by calling `bertini_real` from the command line. If the input file is called anything other than `input`, than the `-input` or `-i` option followed by the filename must be used (more information below). Bertini_real uses the tracker options for Bertini, which are set at the top of the input file. We suggest the following configuration options in the input file for Bertini_real: sharperndigits > 0, imagthreshold $\sim$ 1e-5. Other options can improve performance and tighten up the produced decomposition. It is important to note that for Bertini_real to run, the MATLAB executable must be on the path

4.4 Command Line Options

Below are the command line options for Bertini_real. These are placed after the initial `bertini_real` command. Each command starts with a single dash, and any required arguments should be placed after. For example, if the user wanted to run Bertini_real decomposing a specific component, he or she would type: `bertini_real -component x`, where x is the integer index of the component to decompose.

- `-component`, `-comp`, or `-c`
  Required argument is the integer index of the component for Bertini_real to decompose (e.g `-component 1`)
- `-debug`
  If used, program will pause for 30 seconds before running for debugging purposes. No required argument.
- `-dim` or `-d`
  Required argument is the target dimension for Bertini_real to shoot for
- `-gammatrick` or `-g`
  Indicator for whether Bertini_real should use the gamma trick in a particular solver. Required argument is either 1 (if you'd like Bertini_real to use the gamma trick) or 0 (if not).

- `-help` or `-h`
  Displays a help message containing the version of Bertini_real, where Bertini_real can be found online, support information, and finally the command line options.

- `-input` or `-i`
  Used if input file is named something other than 'input'. Required argument is the filename.

- `-mode` or `-m`
  Sets the mode of Bertini_real to be used. Required argument is the mode of operation, and there are currently two valid modes (`bertini_real` and `crit`). `bertini_real` is the default mode.

- `-nostifle` or `-ns`
  If used, screen output will not be stifled. No required argument

- `-nomerge` or `-nm`
  Indicates that Bertini_real should not merge ends. No required argument.

- `-output`, `-out`, or `-o`
  Required argument is the name of the output directory.

- `-projection`, `-pi`, or `-p`
  Indicator for whether to read the projection from a file, rather than randomly choose it. Required argument is the filename.

- `-quick` or `-q`
  Sets the level of quickness for the solver. The quicker the solver, the less robust. ⋆

- `-veryquick` or `-vq`
  Sets the level of quickness for the solver. The quicker the solver, the less robust.

- `-sphere` or `-s`
  Sets indicator that Bertini_real should use sphere created by user rather than just compute sphere. Required argument is the name of the file for Bertini_real to read

- `-verb`
  Required argument is the level of the verbosity you'd like to set.

- `-version` or `-v`
  Displays the version of Bertini_real running on your computer. Has no required argument.

5 Troubleshooting

6 Visualization

After running Bertini_real, the output results can be visualized in MATLAB. First, open MATLAB and call `gather_br_samples`. This parses the output from Bertini_real into a .mat file. Then, call `bertini_real_plotter`, which creates a handle class object and facilitates selection of parts of the decomposition to view. There are many options, all of which are documents and displayed via `help bertini_real_plotter` in MATLAB. To run bertini_real_plotter with a specific option, type in MATLAB `bertini_real_plotter('option', 'option_argument')`, where the option_argument will vary depending on the option you decide to alter. The options are listed below.

- `'autosave'`
  By default, the autosave option is on. Thus, the autosave option can be used if you do not want a figure to be automatically saved to the working directory upon creation. Either 'false' or 0 can be used to disable autosave. (e.g. `bertini_real_plotter('autosave', 'false')`).

- `'colormap'`
  Users can alter the colormap used by MATLAB with the 'colormap' option by providing a handle to another color map. A full list of built-in colormaps can be found online on the MATLAB help site.
  (e.g. Using the summer colormap: `bertini_real_plotter('colormap', @summer)`)

- `'curve'` or `'curves'`
  By default, the figure created by `bertini_real_plotter` allows the user to display raw curves on the figure. The user can disable this option by setting 'curve' or 'curves' to false. To disable curves, the user can use 'n', 'no', 'none', 'false', and 0.
  (e.g `bertini_real_plotter('curve', 'false')`)

- `'faces'`
  By default, the figure created in MATLAB will show both the raw curves and faces. By setting 'faces' to false, only the option to display the raw curves will be given. To disable faces, you can use 'n', 'no', 'none', 'false', and 0.
  (e.g `bertini_real_plotter('faces', 'none')`)

- `'filename'` or `'file'`
  By default, `bertini_real_plotter` looks for any .mat file that begins with `BRinfo` . If more than one of those files exists, it looks for the most recent one and uses that in the visualization. If you'd like to specify a specific file to visualize, whether it's an older file from a previous run of `gather_br_samples` a file with a different filename structure alltogether, the filename option allows this.
  (e.g. `bertini_real_plotter('filename', 'Example_File_Name.mat')`)

- `'labels'`
  By default, the figure created by `bertini_real_plotter` allows the user to apply labels to the figure upon creation. If you would like to disable this option, you can use the labels option with any of the following arguments 'n' ,'no', 'none', 'false', or 0.
  (e.g. `bertini_real_plotter('labels', 'none')`).

- `'linestyle'`
  Used to change the line style of lines in the MATLAB figure. Options include: '-' (solid line), '–' (dashed line), ':' (dotted line), and '-.' (dash-dot line).
  (e.g. `bertini_real_plotter('linestyle', ':')`)

- `'monocolor'` or `'mono'`
  Used to create a mono-color figure. Required argument is either a RGB triple (row vector with three columns with each entry ranging from zero to one) or one of the following options: 'r' (red), 'g' (green) 'b' (blue), 'm' (magenta), 'c' (cyan) 'y' (yellow), and 'k' (black). For example, creating a red-only figure would be done by typing in MATLAB: `bertini_real_plotter('mono', 'r')`. By default the mono-color option is off.

- `'proj'` ⋆ Unsu

- `'vertices'` or `'vert'`
  By default, the figure created in MATLAB will allow the user to place vertex markers and labels on the figure. Setting 'vertices' to false disables this option. To disable vertices, you

can use ‘n’, ‘no’, ‘none’, ‘false’, and 0.
(e.g `bertini_real_plotter('vertices', 0)`)

7 3D Printing

A Output Formats

A.1 .curve

(num_variables total) num_vertices num_edges
num_V0 num_V1 num_midpts num_newpts

indices of V0
indices of V1
indices of midpoints
indices of added_points
projection excluding the homogeneous 0 coordinate.

A.2 .edge

A.3 .vert