A brief tour of Bertini_real

with trailing notes on some challenges to success

Danielle Brake

June 5, 2018

Outline

Implementation - Bertini_real [BBH+17]

- Command-line MPI-parallel program.
- Uses Bertini 1 as its path tracker.
- C++ code, with options for Matlab or Python for symbolic operations.
- Matlab and Python visualization suites.

```
 BertiniReal(TM) v1.6.0

 D.A. Brake with
 D.J. Bates, W. Hao, J.D. Hauenstein,
 A.J. Sommese, C.W. Wampler

(using GMP v6.1.2, MPFR v3.1.5)


  Library-linked Bertini(TM) v1.6
   (February 27, 2017)

 D.J. Bates, J.D. Hauenstein,
 A.J. Sommese, C.W. Wampler

(using GMP v6.1.2, MPFR v3.1.5)
```

Requirements

For now, you have to install Bertini_real (and Bertini 1) from source.

- To install
  - Modern C++ compiler (C++11 minimum)
  - Boost (not old), MPI (your choice)
  - Autotools (sorry)
  - Bertini 1 – compiled and installed from source, + its dependencies
- To use
  - Either Matlab + symbolic toolbox,
  - or Python + sympy
- To visualize
  - Full suite – Matlab + a few things from the exchange
  - Partially implemented suite – Python's Matplotlib

sorry, no native windows compilation at the moment.

Basic calling sequence

Here's a quick summary. See the user manual, please

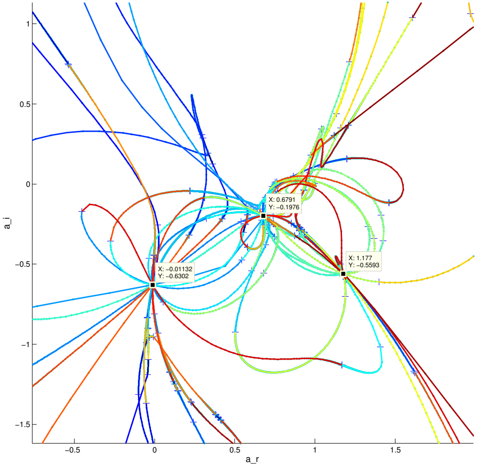
1. Make plain-text Bertini 1 input file. Move to that directory. For sanity, limit yourself to one bertini input file per directory.
2. `bertini` – computes complex witness set
3. `bertini_real -opt1 arg1 -opt2 arg2` – computes real cell decomposition
4. Visualize, probably in Matlab cuz python viz is incomplete so far
5. Tell me how it goes!
    - issues at `github.com/ofloveandhate/bertini_real` or
    - email to `brakeda@uwec.edu`

Cautions and disclaimers

- Bertini_real is numerical software, and does not compute a certified correct decomposition.
- The default settings for Bertini do not work well for Bertini_real. Some almost always need tweeking.
- See below and the manual for notes on problems and solutions.
- There are known areas that need improvement. Again, see below and the manual.

Outline

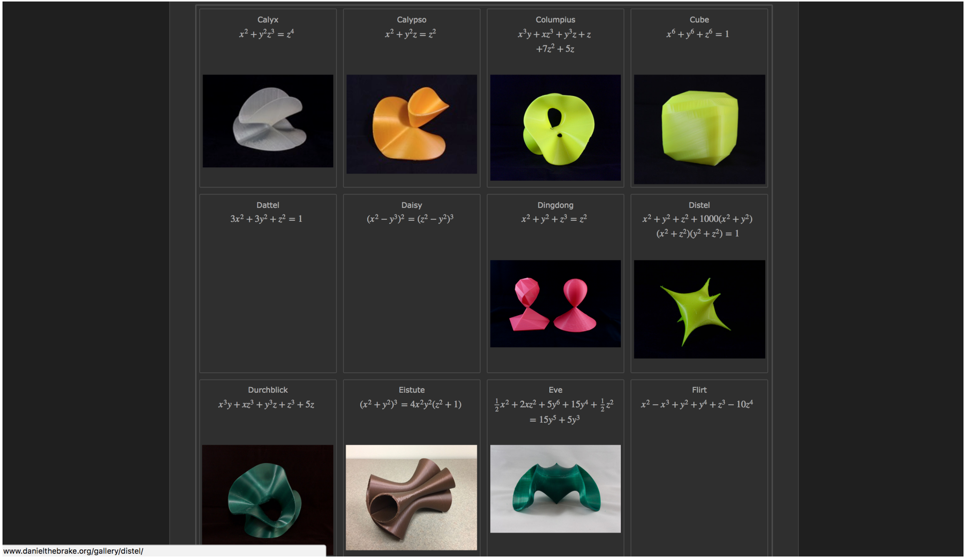
Application - Kinematics



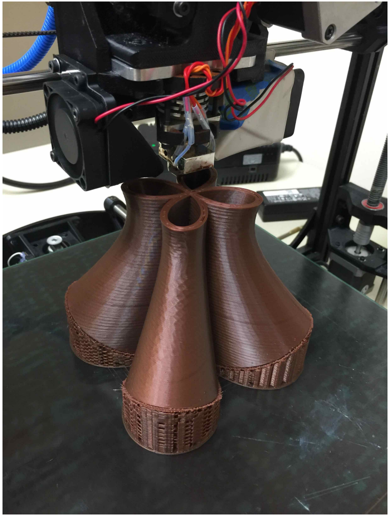
A curve, of kinematic mechanisms, of degree 630 / 128 in the meaningful projection.

Application - reproducing Herwig Hauser's gallery

Application - reproducing Herwig Hauser's gallery



Motivation - reproducing Herwig Hauser's gallery



Outline

Principle – implicit function theorem

An object $\mathcal{C}$ may be parameterized such that:

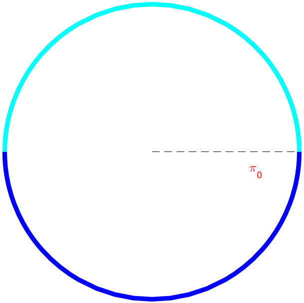
- the number of parameters is equal to $\dim(\mathcal{C})$
- the parameterization is well-defined *almost everywhere*
- the parameterization is not 'special'

Example: the circle defined

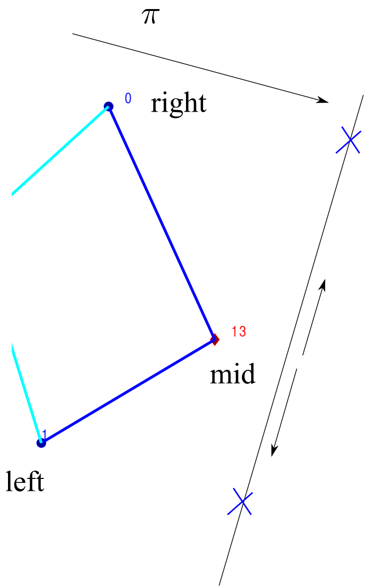
$$x^2 + y^2 - 1 = 0$$

can be parameterized by x:

$$y = \pm\sqrt{1 - x^2}$$

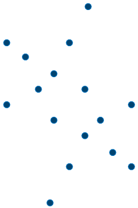


Edges – Structure of computed object

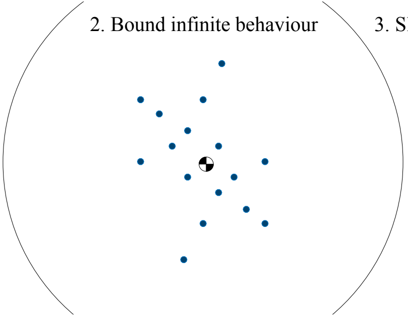


Curves – What Bertini_real does

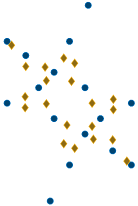
1. Compute critical points



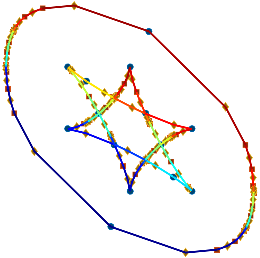
2. Bound infinite behaviour



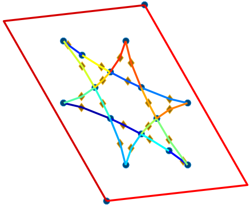
3. Slice between critical points



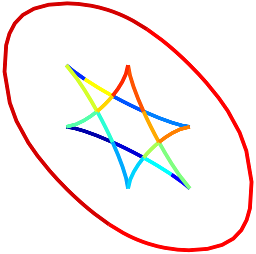
4. Connect the dots



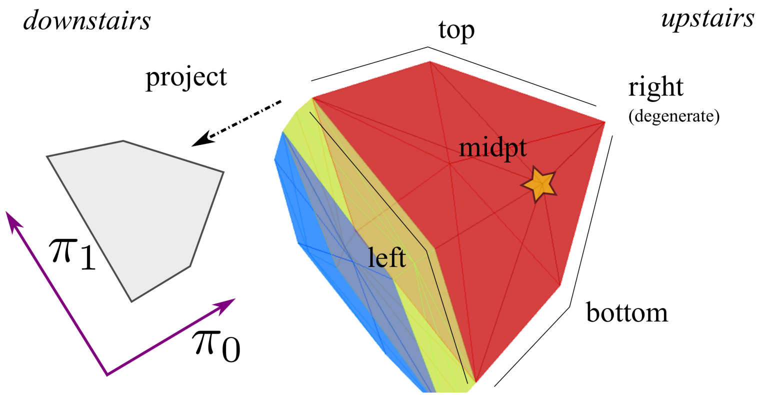
5. Merge (optional)



6. Smooth (optional)

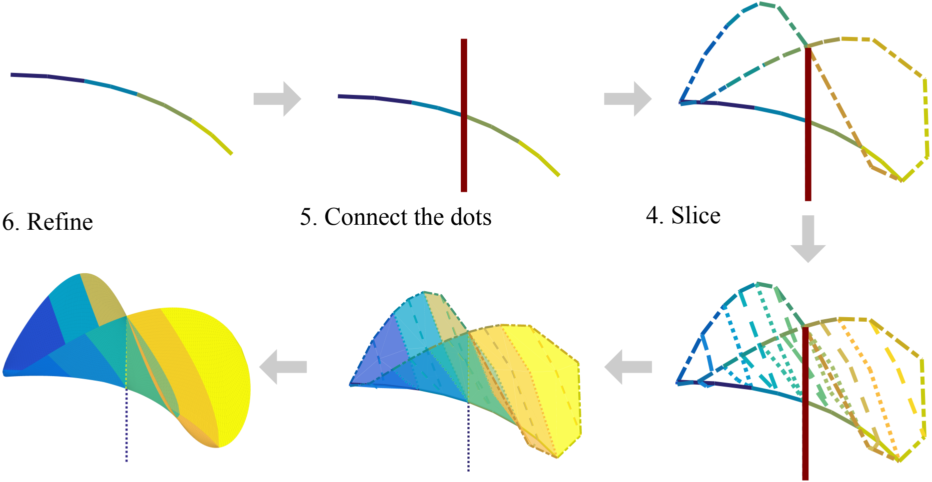


Faces – Structure of computed object



Surfaces – What Bertini_real does

1. Decompose critical curve
2. Decompose singular curves
3. Intersect with sphere
4. Slice
5. Connect the dots
6. Refine



Outline

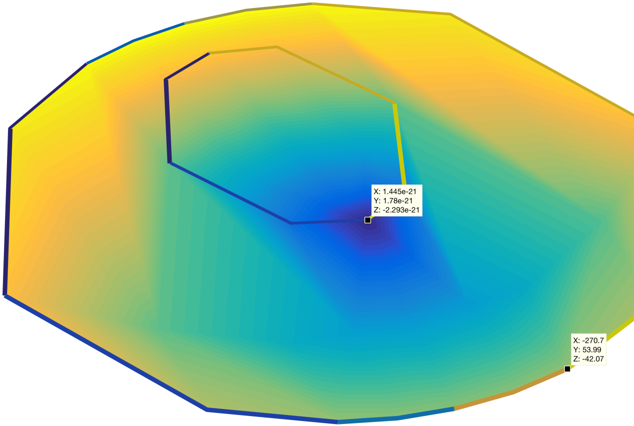
Choosing a projection

The decomposition algorithms depend on the component being in *general position*. Accomplished by using a random real projection. Problems with random projections:

- Some cause the critical points on the critical curve to be very far from the origin.
  You can easily get things like critical points very near the origin ($\|x\| \approx$1e-6), and very far away ($\|x\| \approx$1e3). Now you have a scaling problem!
- Some cause multiple critical points to have similar projection values. Near an axis projection, or near symmetry.
  Should we do a π_1 surface slice between `1.451331542e1` and `1.451331601e1`???

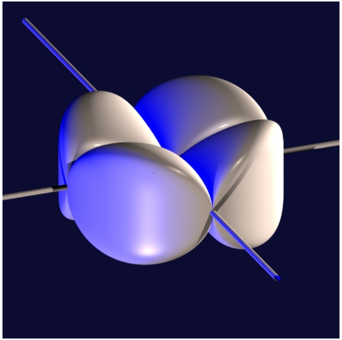
Scale

- How to deal with points close to the origin?
- How to deal with points close together?
- How to decide if two points are the same point, or not?



Singular on singular

- Surfaces can have non-isolated singularities.
  Must curve-decompose singular curves
  $\Rightarrow$ deflate witness points for the singular curves.
- No good deflation implementation exists, that I've found.[1]
- Combinatorial growth in number of polynomials in deflating system.



$$x^4y^2 + y^4x^2 - x^2y^2 + z^6 = 0$$

- Buggle, only degree 6, but I still fail to decompose it.
- Starts with 1 polynomial, then adds 3, then adds 13, then adds 358, then adds thousands and never terminates.

[1]Best notes I've found on improving isosingular deflation are in [AHS17]

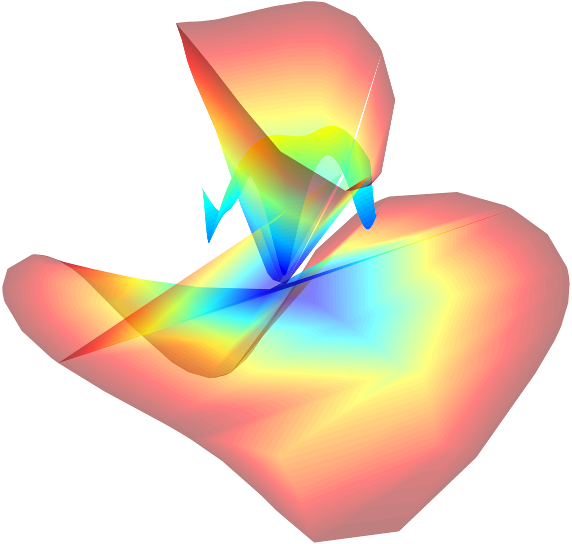
On classifying points as singular

In Bertini, one can classify (end)points as singular based on:

1. Multiplicity > 1
2. Condition number of Jacobian at endpoint
3. Rank of Jacobian at endpoint

2 and 3 are precarious, depending on arbitrary tolerances.

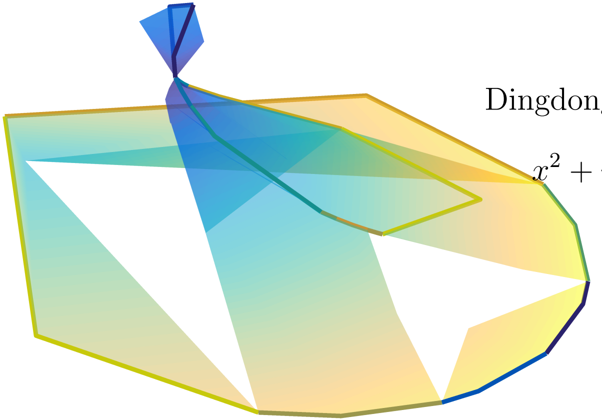
What went wrong?



Calypso

$$x^2 + y^2 z - z^2$$

What went wrong?

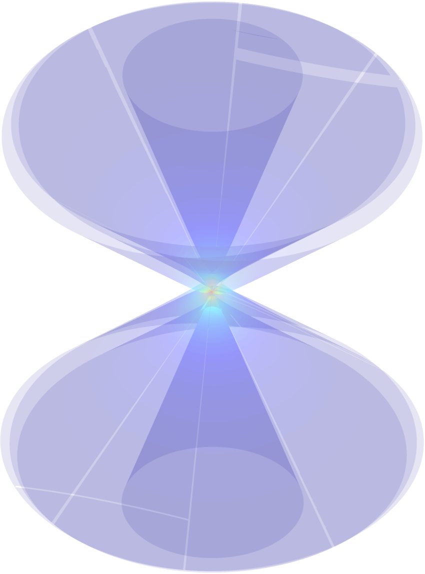


Dingdong

$$x^2 + y^2 + z^3 - z^2 = 0$$

What went wrong?

What went wrong?



Outline

My current typical fixes

- Tighter tracking tolerances – walk a tighter line
- ODE predictor – use higher order, accept higher cost
- Use few Newton correction iterations – require correction each step
- Use the Cauchy endgame for well-separated roots, power series for poorly?
  Use many sample points (hopefully at least half as many as the cycle number?), and
  enter the endgame late.

My current typical fixes (2)

- Always allow higher precision. Safety digits in adaptive precision increase robustness near singularities.
- Do not flag points as singular except if multiple.
  New problem: multiplicity-one singularities are undetectable.
- Over-sharpen. But don't depend on sharpening, because methods for sharpening singular points are not (yet) well-exposed.
- Lower the maximum step size up front
  $\Rightarrow$ increase minimum number of steps

Choose a projection - or not

To solve the projection problem, I take these steps:

1. Run decomposition several times. Hope completes correctly.
2. If not, choose a special projection. Probably onto two variables. Hope completes correctly.
3. If not, choose a less special projection. Probably integer coefficients, always orthogonal. Hope completes correctly.
4. If not, go back to random, play games with settings. Use more processors, wait longer.

Use relative comparisons

Use the relative norm and relative difference.
Donald Knuth knows this, and Boost.UnitTest specifically recommends we use the relative difference for unit testing code.

- Compare u and v, by scaling by $\|u\|$, $\|v\|$. Unless, of course, you shouldn't. See [Squassabia 2000]
- Beware comparing numbers at different *precisions*.
- Beware comparing numbers at different *accuracies*.
- Be careful comparing to zero.

I yearn for a system which will tag points with the accuracy with which they were computed, and prevent garbage comparisons.

Idea: keep it real

Any little bit of imaginary part creeps in and infects all subsequent real computations.

- $\Rightarrow$ I want a pure-real tracker, that throws when domain errors arise.
- $\Rightarrow$ I also need to replace the default complex patch equation from Bertini's NID with a real patch, so the homotopy is entirely real.
- This would be particularly helpful when doing the `ConnectTheDots` routines in Bertini_real. Easy to step off the path, I think a real-only tracker would solve some problems.

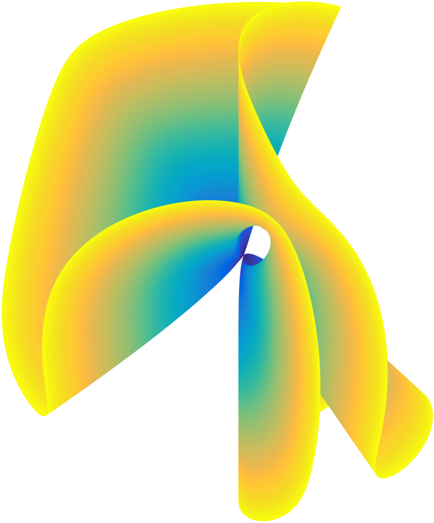
Idea: decompose in zones

To solve the scale problem, we could

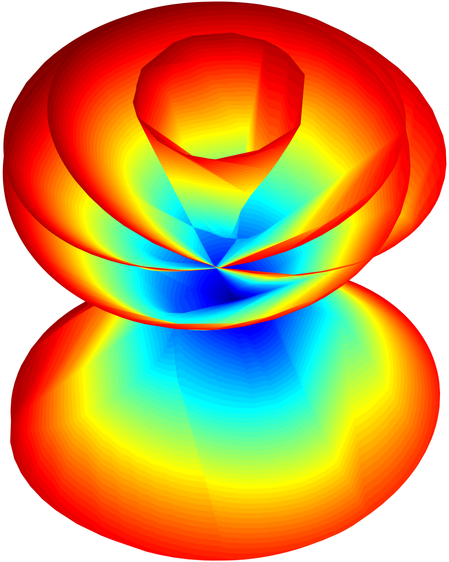
1. Compute critical points;
2. For each critical point, center, scale, decompose;
3. Glue together decompositions.

Sounds like a complicated bit of code, so it's not being done.

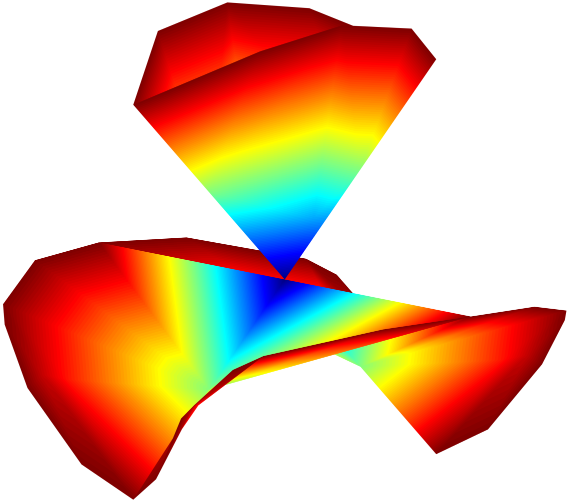
Success!



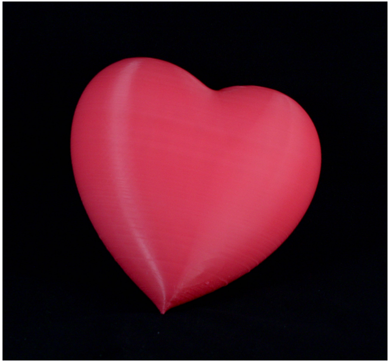
Success!



Success!



Thank you for your kind attention!



- Tulay Ayyildiz Akoglu, Jonathan D Hauenstein, and Ágnes Szántó, *Certifying solutions to overdetermined and singular polynomial systems over q*, Journal of Symbolic Computation (2017).
- Daniel A Brake, Daniel J Bates, Wenrui Hao, Jonathan D Hauenstein, Andrew J Sommese, and Charles W Wampler, *Algorithm 976: Bertini_real: Numerical decomposition of real algebraic curves and surfaces*, ACM Transactions on Mathematical Software (TOMS) **44** (2017), no. 1, 10.