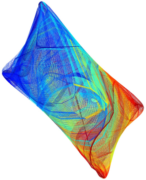
Bertini_real: Software for real algebraic sets

Daniel Brake
with
Dan Bates, Wenrui Hao, Jon Hauenstein,
Andrew Sommese, and Charles Wampler

08 August, 2014



Bertini_real Overview

Bertini_real is compiled command line software:

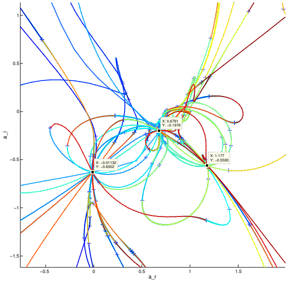
- uses Bertini as its homotopy continuation engine,
- performs numerical computations to produce a *cellular decomposition* of real algebraic components,
- uses Matlab for symbolic computations, such as deflation; as well as visualization,
- can decompose higher-dimensional curves and surfaces, including singularities,
- results are readily 3d printed.

Real Components

Bertini_real – Numerical Cellular Decomposition

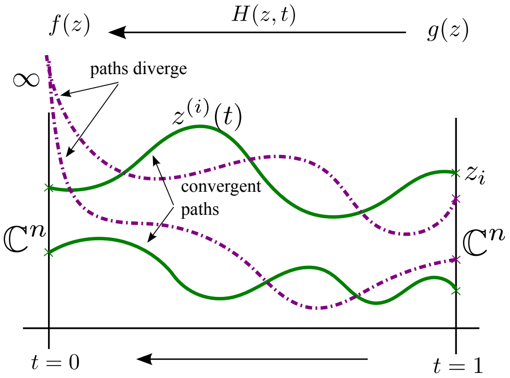
Setup

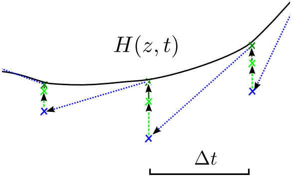
- Let f be a polynomial system with $\mathbb{R}$ coefficients, and $f : \mathbb{C}^N \to \mathbb{C}^n$.
- Let $V(f)$ be the variety of f.
- Consider $C \subseteq V(f)$ be a component of dimension k.
- If f is overdetermined, replace f by a randomized version of itself.



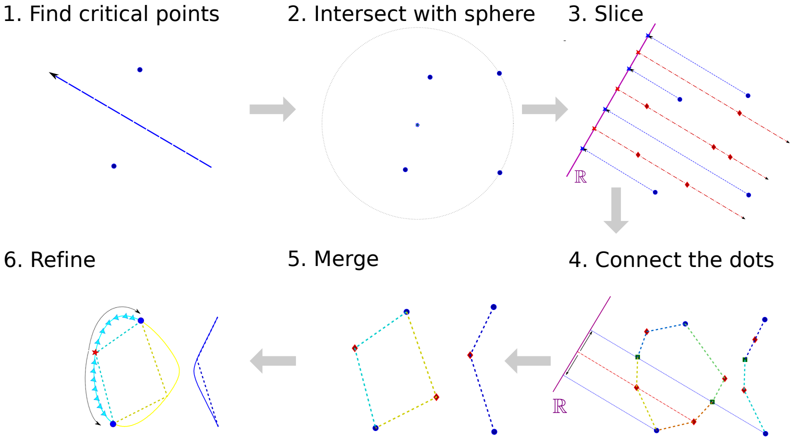
Our objective is to decompose the real part of C; i.e., $C \cap \mathbb{R}^N$

Homotopy Methods



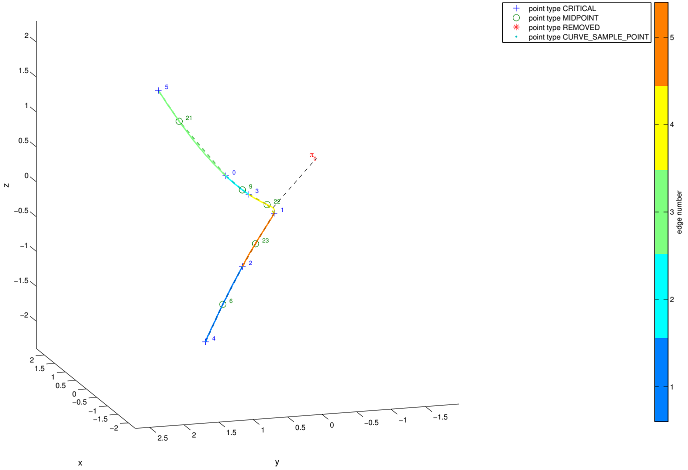


Curves



[Lu,Bates,Sommese,Wampler,2006]

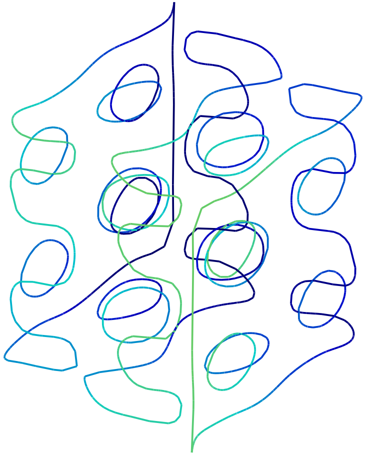
Curves



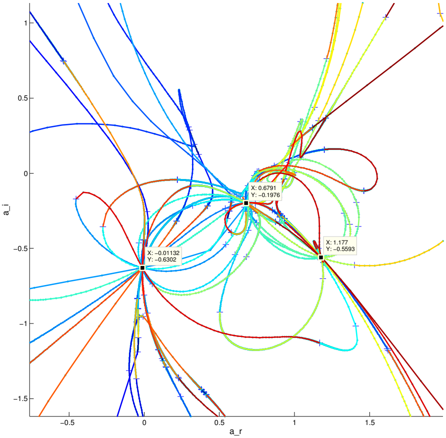
twisted cubic

$$f(x,y,z) = \begin{bmatrix} y - x^2 \\ z - x^3 \\ y^2 - xz \end{bmatrix}$$

Curves

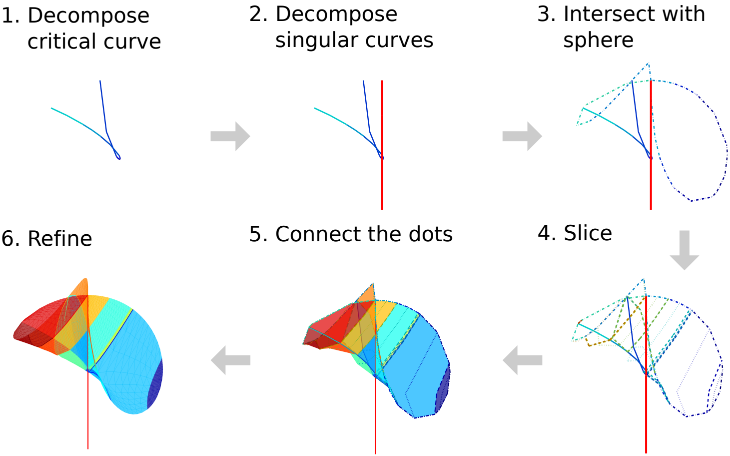


Curves



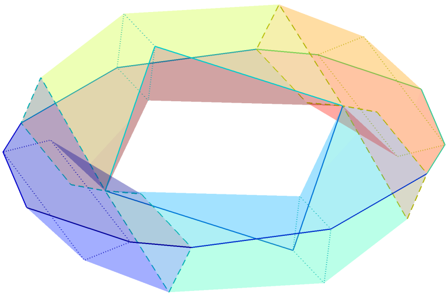
Burmester 3-3 curve
dimension 14, degree 630. In 2d projection, of degree 128.

Surface Decomposition



[Besana,Di Rocco,Hauenstein,Sommese,Wampler, 2012]

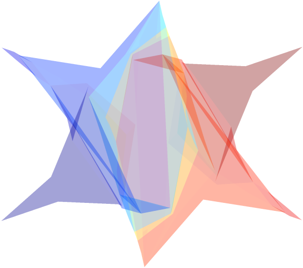
Surface examples



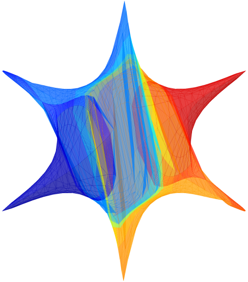
torus

$$f(x,y,z) = (x^2 + y^2 + z^2 + 2^2 - \frac{1}{2}^2)^2 - 16(x^2 + y^2)$$

Surface examples



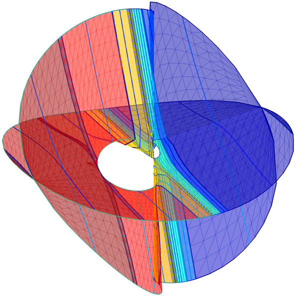
distel, unrefined



distel, refined

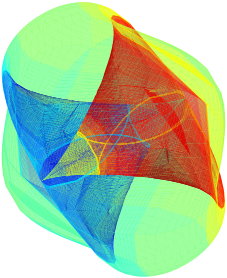
$$f(x, y, z) = x^2 + y^2 + z^2 + 1000(x^2 + y^2)(x^2 + z^2)(y^2 + z^2) - 1$$

Surface examples



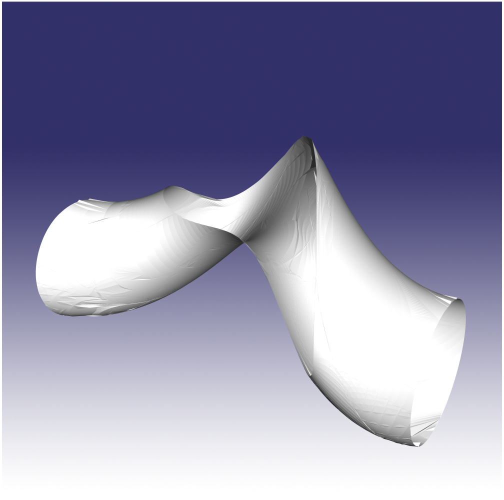
solitude

$$f(x,y,z) = x^2yz + xy^2 + y^3 + y^3z - x^2z^2$$



klein

$$f(x,y,z,w) = \begin{bmatrix} w^2 + x^2 + y^2 + z^2 - 1 \\ 2wyz - x(y^2 + z^2) \end{bmatrix}$$



Fivebar mechanism kinematics
Six-dimensional trigonometric system,
passed though an atan2 projection

Critical points of curves

Computing critical points of curves is easy.

Since f defines a dimension-one component, the working witness set comes with one random linear:

$$\begin{bmatrix} f \\ \mathcal{L}_1 \end{bmatrix}$$

Then we use regeneration to solve the system

$$\begin{bmatrix} f \\ v^\intercal \cdot \begin{bmatrix} Jf \\ \pi_1 \end{bmatrix} \\ \text{patch}_v \end{bmatrix}$$

[Hauenstein,Sommese,Wampler, 2009]

Regeneration to find crit

Regeneration uses products of linears to build up a start system for a homotopy. We're solving by homotoping as

$$H(x,v;t) = (1-t)\begin{bmatrix} f(x) \\ v^{\intercal}\cdot\begin{bmatrix} Jf(x) \\ J\pi_1 \end{bmatrix} \\ \text{patch}_v \end{bmatrix} + t\begin{bmatrix} f(x) \\ M_1(v)\prod_{i=1}^{\delta} L_{1,i}(x) \\ \vdots \\ M_N(v)\prod_{i=1}^{\delta} L_{N,i}(x) \\ \text{patch}_v \end{bmatrix} = 0$$

- δ is the maximum degree any polynomial in f.

Building a start system

- To form the start system and solutions, we move the given random complex $\mathcal{L}(x)$ to each $L_{j,i}$ one at a time, to find the x coordinates:

$$H(x;t) = (1-t)\begin{bmatrix} f \\ L_{j,i} \end{bmatrix} + t\begin{bmatrix} f \\ \mathcal{L}_1 \end{bmatrix} = 0$$

- Since only one $L_{j,i}$ vanishes for each start point, the remainder of the start functions must vanish due to $M_j(v)$, which are solved for v start values by matrix inversion:

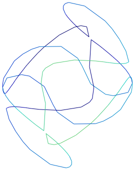
$$v = \begin{bmatrix} M_{1\neq j} \\ \vdots \\ M_{N\neq j} \\ \text{patch}_v \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Then we take the x solution from the top, and v from bottom, and concatenate to form the start point.

Crit points of surfaces

Computing critical points of surfaces is hard.

- Surfaces have *critical curves*.
- Surfaces also have critical points themselves.



Current surface critical method

Using the *determinantal* form of the criticality conditions:

witness set:

$$f_{\text{crit\_curve}} = \begin{bmatrix} f \\ \det \begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \\ \mathcal{L}_1 \end{bmatrix} \qquad f_{\text{crit\_crit}} = \begin{bmatrix} f \\ \det \begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \\ \det \begin{pmatrix} J \begin{pmatrix} f \\ \det \begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \end{pmatrix} \\ J\pi_1 \end{pmatrix} \\ \text{patch}_v \end{bmatrix}$$

- Determinant operation produces high degree polynomials, contributing to numerical issues.
- Using this formulation for dimension 3 decompositions will be even worse w.r.t. degree.
- Fortunately, we can still use the nullspace method for finding critical points of this curve.

Current surface critical method

Using the *determinantal* form of the criticality conditions:

witness set:

$$f_{\text{crit\_curve}} = \begin{bmatrix} f \\ \det\begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \\ \mathcal{L}_1 \end{bmatrix} \qquad f_{\text{crit\_crit}} = \begin{bmatrix} f \\ \det\begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \\ \det\begin{pmatrix} J\begin{pmatrix} f \\ \det\begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \end{pmatrix} \\ J\pi_1 \end{pmatrix} \\ \text{patch}_v \end{bmatrix}$$

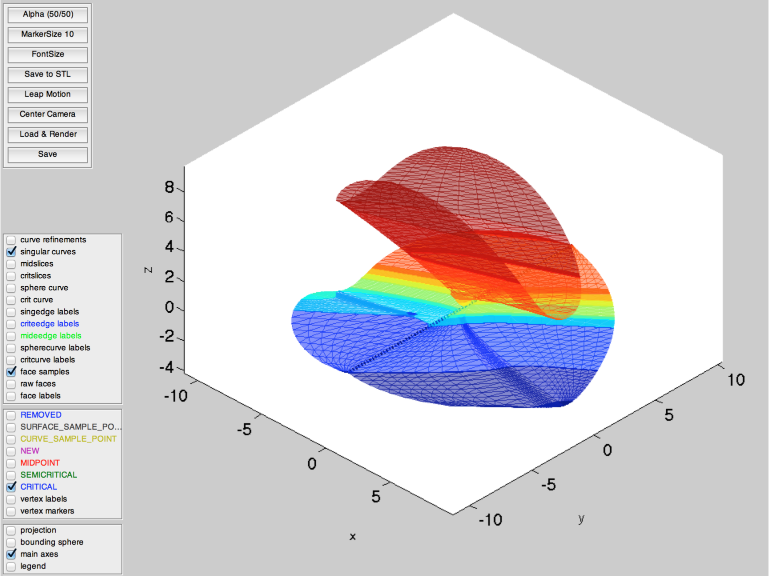
- Determinant operation produces high degree polynomials, contributing to numerical issues.
- Using this formulation for dimension 3 decompositions will be even worse w.r.t. degree.
- Fortunately, we can still use the nullspace method for finding critical points of this curve.

Crit points of crit curve

The actual system Bertini_real currently solves to obtain crit points of crit curve, by regeneration:

$$f_{\text{crit\_crit}} = \begin{bmatrix} f \\ \det \begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \\ v^\intercal \cdot J \begin{pmatrix} f \\ \det \begin{pmatrix} Jf \\ J\pi_1 \\ J\pi_2 \end{pmatrix} \\ J\pi_1 \end{pmatrix} \\ \text{patch}_v \end{bmatrix}$$

Matlab UI

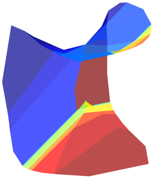

Alpha (50/50)
MarkerSize 10
FontSize
Save to STL
Leap Motion
Center Camera
Load & Render
Save
curve refinements
singular curves
midslices
critslices
sphere curve
crit curve
singedge labels
criteedge labels
mideedge labels
spherecurve labels
critcurve labels
face samples
raw faces
face labels
REMOVED
SURFACE_SAMPLE_PO...
CURVE_SAMPLE_POINT
NEW
MIDPOINT
SEMICRITICAL
CRITICAL
vertex labels
vertex markers
projection
bounding sphere
main axes
legend
8
6
4
2
0
-2
-4
z
-10
-5
0
5
x
10
5
0
-5
-10
y

3D Printing

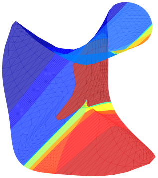
1. Run Bertini

```
****** Witness Set Decomposition ********

| dimension | components | classified
-----------------------------------------
|   2       |    1       |    4
-----------------------------------------
=========
******* Decomposition by Degree *********

Dimension 2: 1 classified component
-----------------------------------------
   degree 4: 1 component

*****************************************
```

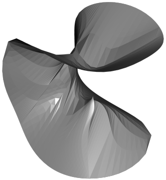
2. Run Bertini_real



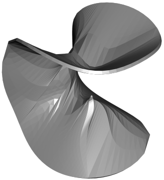
3. Refine



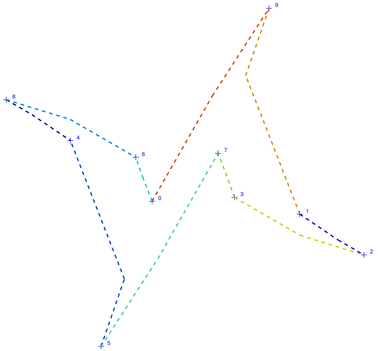
4. Process into `.stl`



5. Thicken surface

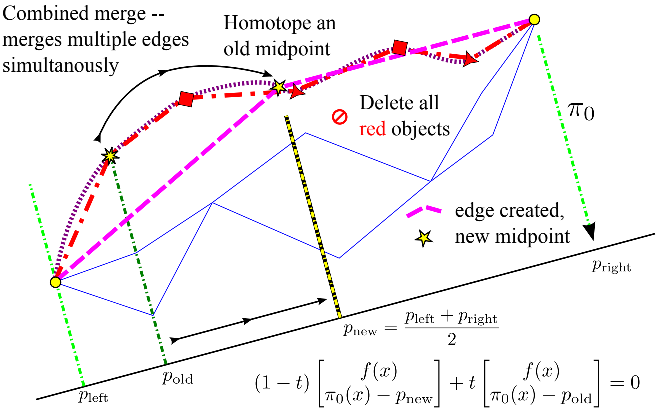


6. Print

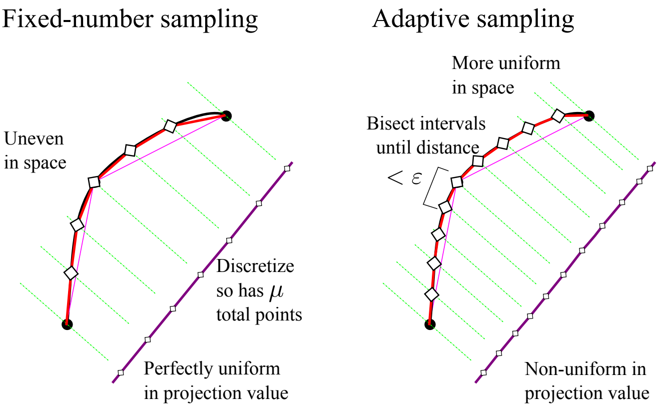


Thank you for your kind attention.

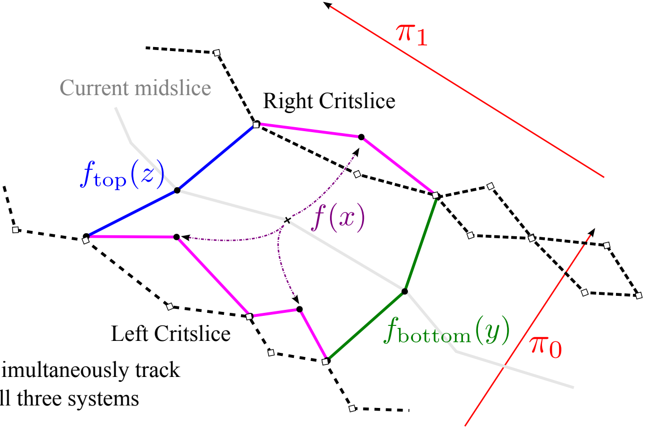
Curve Merging



Curve Sampling

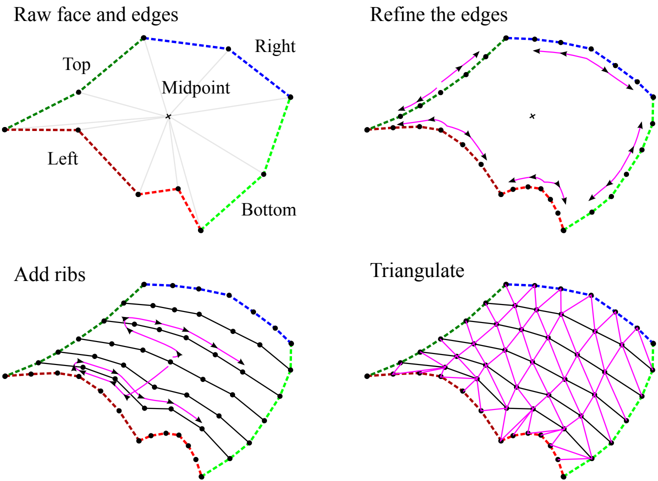


Midtracking



Simultaneously track
all three systems

Surface Refinement



Generic Decomposition

Input: polynomial system f, witness set W, projections $\pi_1, ...\pi_d$.
Output: cell decomposition D.

1: Compute W_K, a witness set for the critical set of dimension d.
2: Compute witness sets for singular sets, sphere intersection.
3: Compute critical points of everything possible.
4: Decompose S, the singular set.
5: Decompose K, the critical set.
6: Intersect with sphere.
7: Slice using linears at all critical points.
8: Slice between all critical points.
9: Glue it all together.

Method not implemented or written up.