Regularizing numerical cell decompositions

Dani(el) Brake

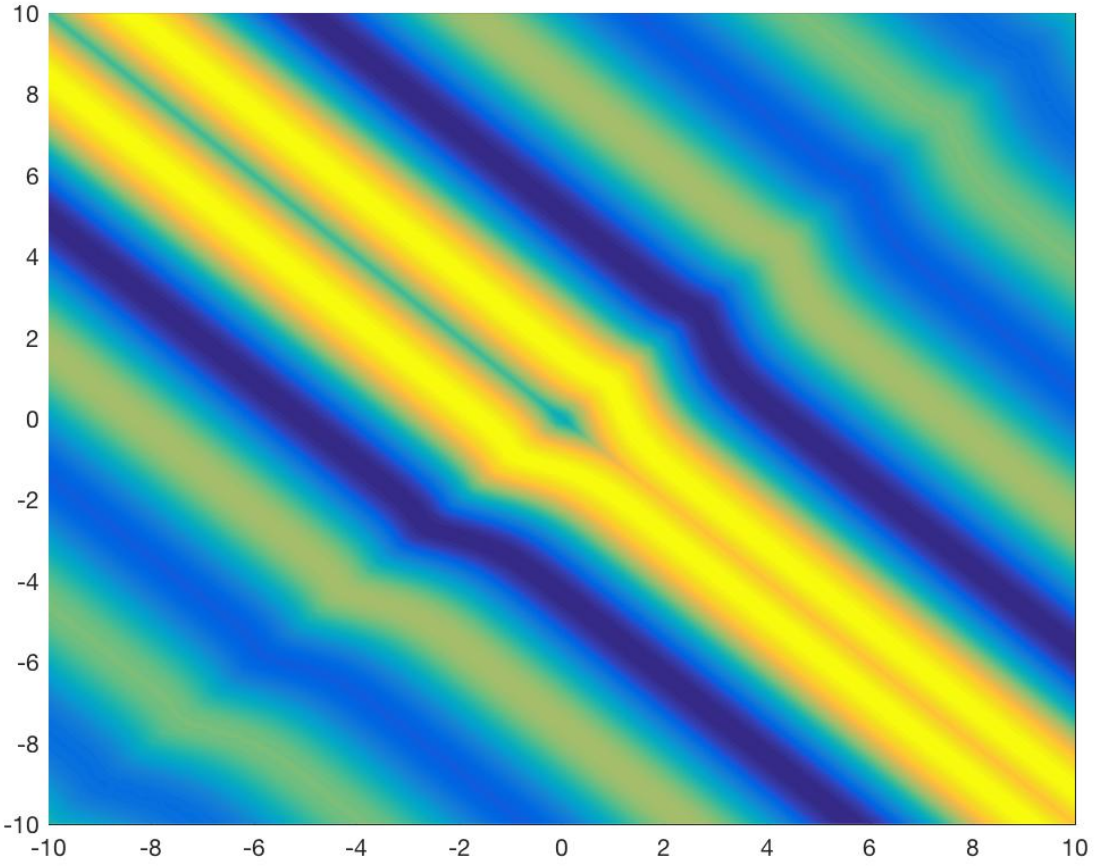
8 October, 2016

Joint work with
Jon Hauenstein and Charles Wampler
thanks to Nick Trefethen

UNIVERSITY OF
NOTRE DAME
College of Science

Outline

Example 0



$$f(x_1, x_2) = \text{besselj}(1, (\text{abs}(x_1 + x_2)) + \text{abs}(\text{erfc}(x_1 - x_2)^{1/3} - 1))$$

Project goals

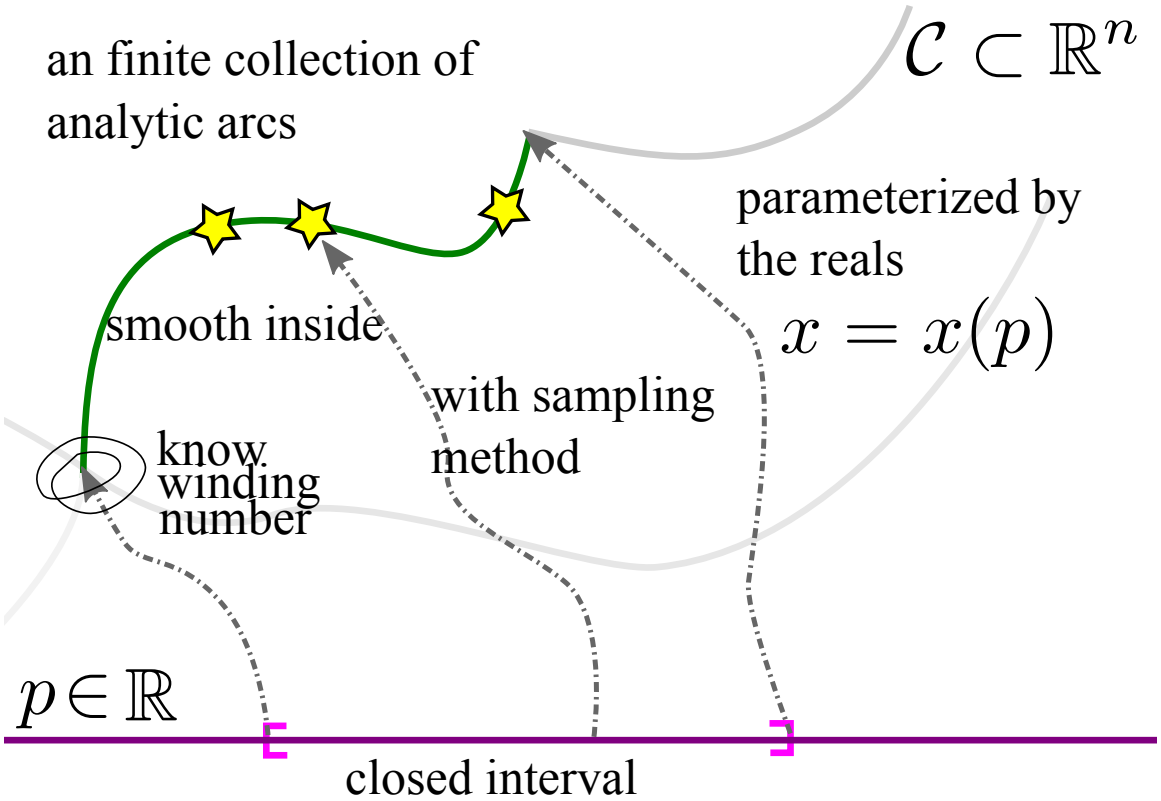
Our main goals are to:

- enable computation of analytic functions over algebraic sets
- be able to solve algebraically constrainted real optimization problems
- gain deeper understanding of the cycle number
- make fun pictures ;)

In this particular project, we are talking about real functions over real algebraic curves

Outline

The 5 data we need



1. A finite set of analytic arcs.
2. Arcs parameterized on a closed interval of $\mathbb{R}$
3. Arcs smooth inside interval. Can be singular at endpoints.
4. Know the winding number at each end. Also is cycle number.
5. Have sampling method to produce new points on the curve.

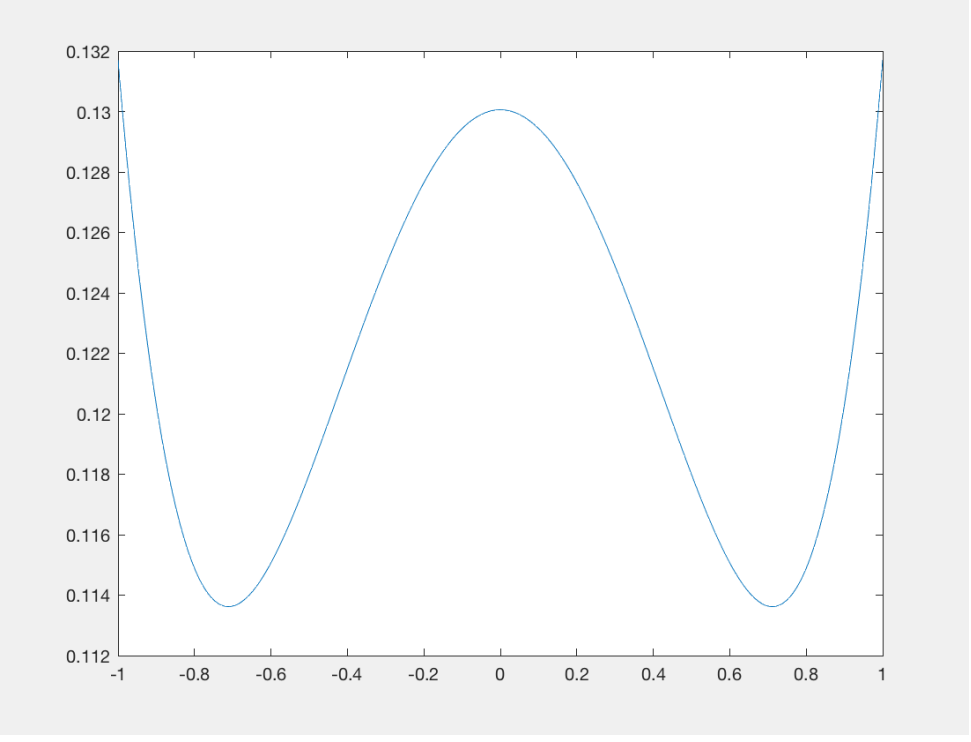
Outline

MATLAB library for using numerical computation using functions.

- Overloads all arithmetic operators. Also plot, max, min, roots, and *many* more.
- Uses function handles for evaluating the functions it approximates.
- Uses minimal number of sample points to represent a function to a given tolerance.
- Many user-tunable parameters.
  - Splitting, for non-smooth functions
  - Tolerance
  - Max number points, etc.
- Works over closed interval of $\mathbb{R}$.
- And so much more.

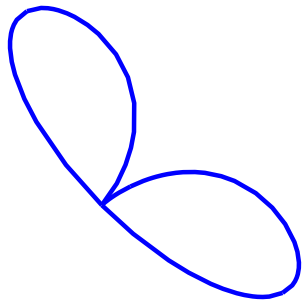
chebfun.org

```
>> f = @(x1) besselj(1,(log(sin(x1)./x1+0.3*cosh(x1.^2))));
>> cf = chebfun(f);
>> plot(cf)
>> cf

cf =

   chebfun column (1 smooth piece)
       interval       length     endpoint values
[      -1,       1]       29      0.13     0.13
vertical scale = 0.13
```



Outline

Puiseux Series



$$f = x^4 + y^4 - (x-y)^2(x+y)$$

Along the cusp, we have Puiseux series

$$(x,y) \quad = \quad \left(s, \quad s \pm s^{3/2} + \frac{3}{4} s^2 \pm \frac{45}{32} s^{5/2} + \ldots \right),$$

But we really need a power series representation on the curve...

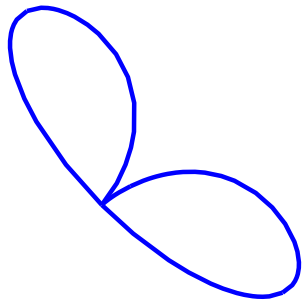
Winding or cycle number

The winding number, also cycle number, c is conceptually many things. Fix an algebraic component $\mathcal{D}$ and a point α.

- The number of times a closed loop walked around α on $\mathcal{D}$ encircles α.
- The number of sheets in a locally irreducible family that come together at α. If α is a smooth point, then $c = 1$.
- The denominator in a Puiseux series. We can regularize the series by substituting in the correct power of some other variable.

$$f(s) = \sum_j \square s^{j/c}$$

Regularize using the cycle number



We can regularize the cusp on the quartic, knowing the cycle number is 2. Substitute

$$s = t^2$$

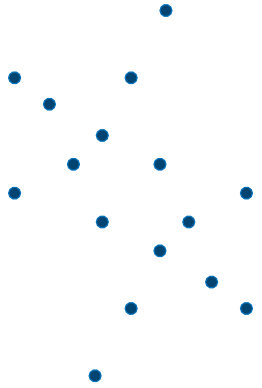
to get

$$(x, y) \;=\; \left(t^2,\;\; t^2 \,\pm\, t^3 \,+\, \frac{3}{4}\, t^4 \,\pm\, \frac{45}{32}\, t^5 \,+\, \ldots \right),$$

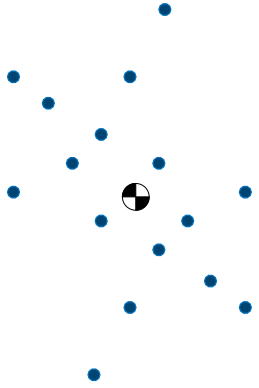
Outline

Curves

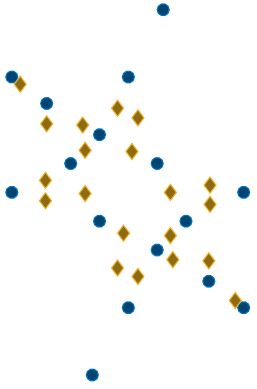
1. Compute critical points



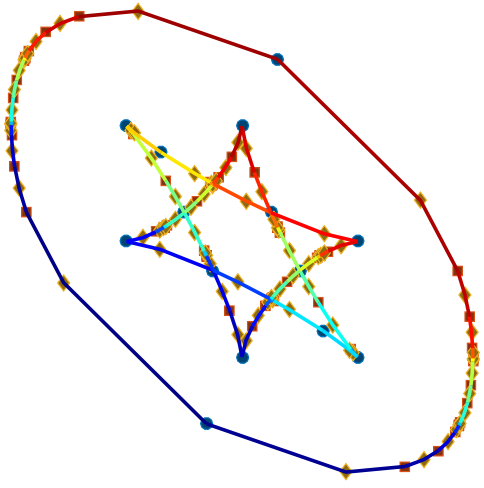
2. Bound infinite behaviour



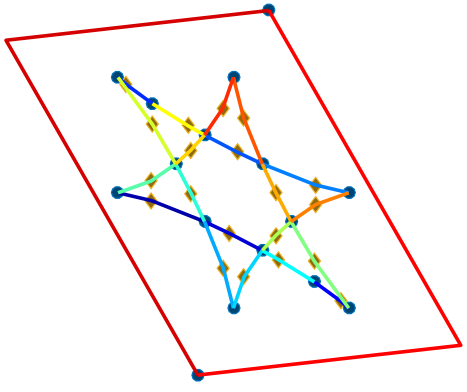
3. Slice between critical points



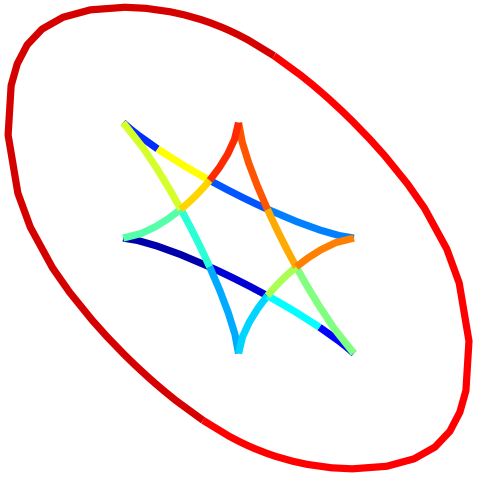
4. Connect the dots



5. Merge (optional)



6. Smooth (optional)



Outline

An tale of two softwares

- We use Chebfun as an engine for computing things like `roots` and `max` of arbitrary functions over $\mathbb{R}$.
- Bertini_real acts as an engine to provide the necessary data.
- Thus, we can do things like `roots` over arbitrary functions $f : \mathbb{R}^n \to \mathbb{R}$, with a curve $\mathcal{C} \subset \mathbb{R}^n$.

The interface between them is essentially Matlab code, coupled with Bertini and Bertini_real.

- a function for evaluation of points on $\mathcal{C}$ which we use as a function handle for Chebfun.
- wrapper functions for doing operations such as `max` on and edge-by-edge basis.
- other niceties as necessary.

Some code

```
ii = 1:length(cf_coord)
p = cf_coord(ii);
pi = pi_out + (pi_mid - pi_out) * (1-p)^c;

fid = fopen('final_parameters','w');
fprintf(fid,'1\n\n');

fprintf(fid,'%1.16f 0.0\n',pi);
fclose(fid);

bertini('filename','input_BrCf_slice','stifle');
s = get_generic_solns('real_finite_solutions');
solutions(ii,:) = real(s(coordinate_indices
```

Some more code

```
evalme = 'useme = func(';
for jj = 1:this.BRinfo.num_variables-1
    evalme = sprintf('%s this.edge_chebfuns(%i,%i).right',evalme,ii,jj);
    if jj ~= this.BRinfo.num_variables-1
        evalme = sprintf('%s, ',evalme);
    else
        evalme = sprintf('%s);',evalme);
    end
end
eval(evalme);

[val,pos] = m_or_m(useme,varargin{:});
values_by_edge{ii,2} = val;
values = [values;val];
if need_positions
    callme = MakeBrCfHandle(this,ii,1:this.BRinfo.num_variables-1,'right');
    pts_on_curve = callme(pos);
    positions_by_edge{ii,2} = pts_on_curve;
    positions = [positions;pts_on_curve];
end
```

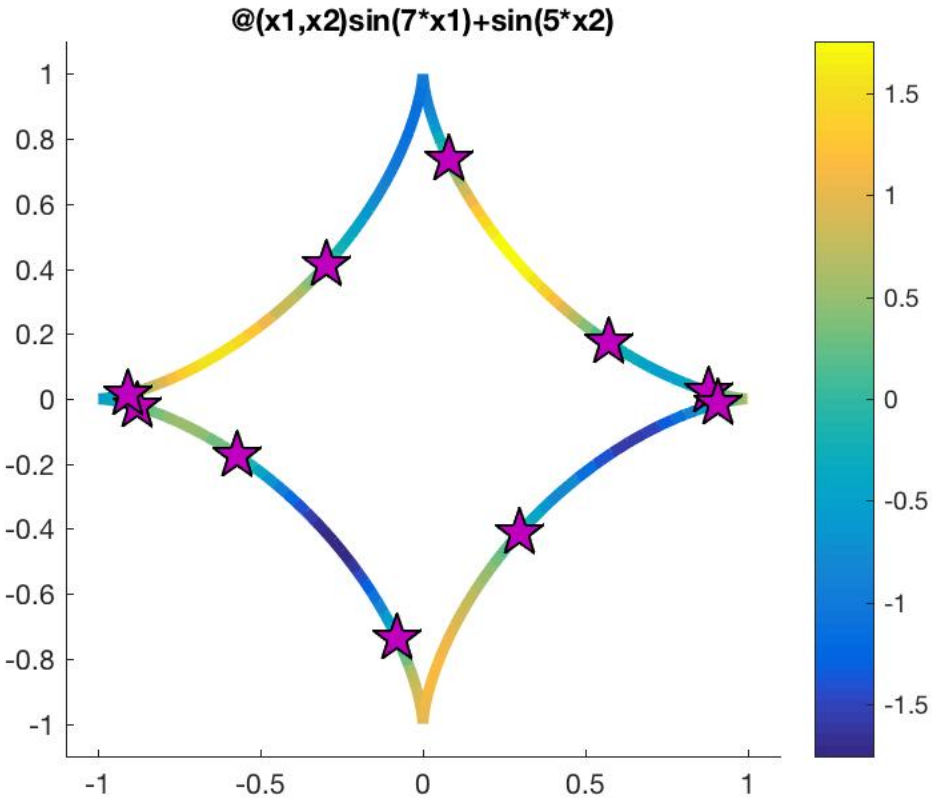
Example 1

here's a reasonable function:
$f(x_1, x_2) = sin(5x_1) + sin(7x_2)$

We can find plot f over $\mathcal{C}$, the astroid curve defined by

$$(x^2 + y^2 - 1)^3 + 27x^2y^2 = 0$$

and its roots.

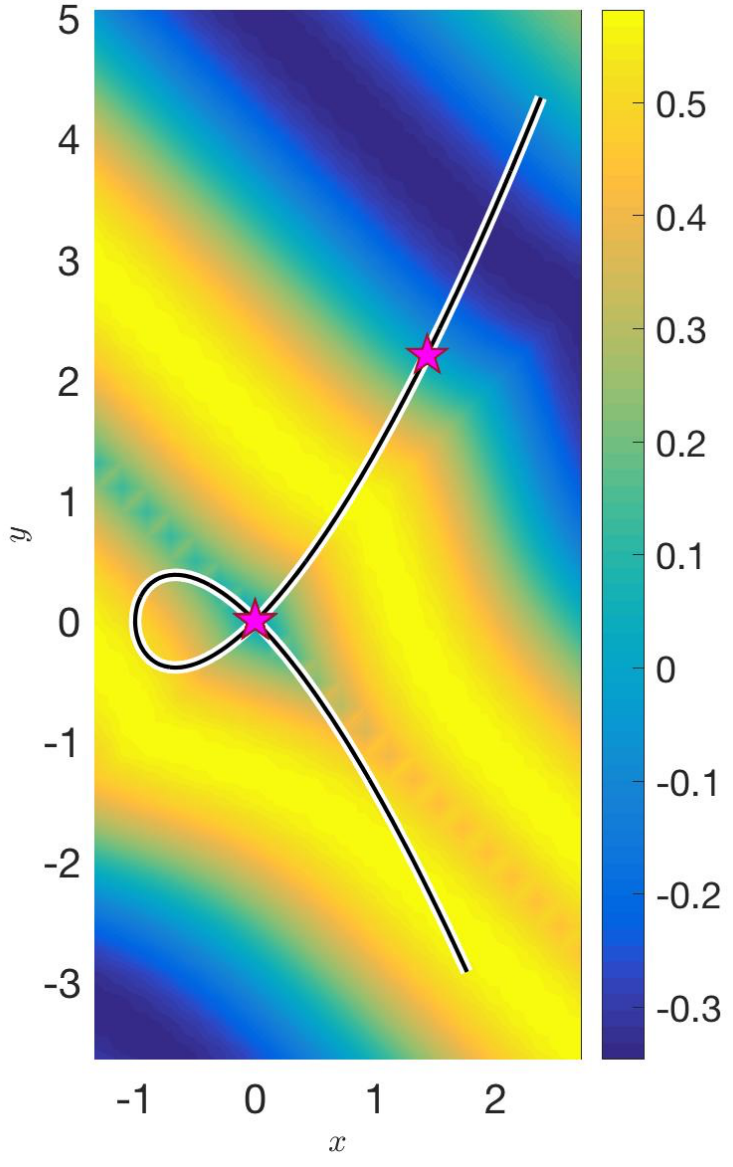


Example 2

consider this crazy function:
$f(x_1, x_2) = \text{besselj}(1, (\text{abs}(x_1 + x_2)) + \text{abs}(\text{erfc}(x_1 - x_2)^{1/3} - 1))$

We can find roots of f over $\mathcal{C}$, the alpha curve defined by.

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



Thank you for your kind attention