Géométrie algorithmique non linéaire et courbes algébriques planaires

Luis Peñaranda

INRIA Nancy-Grand Est

3 décembre 2010

Presentation outline

introduction
- research field
- the problem
- previous work

algorithmic issues
- overview
- details
- complexity

implementation issues
- isotop
- cgal algebraic kernel

conclusion

Where are we standing?

- exact geometric computing
  - computational geometry
  - computer algebra tools

Where are we standing?

- exact geometric computing
  - computational geometry
  - computer algebra tools

- a long history
  - robot motion planning
  - CAGD

Where are we standing?

- exact geometric computing
  - computational geometry
  - computer algebra tools
- a long history
  - robot motion planning
  - CAGD
- recent advances in real solving

Exact geometric computing

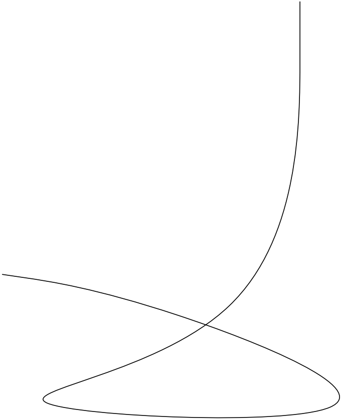
- the general problem
  - algorithms often assume that numbers are real
  - computers do not like real numbers
  - inconsistencies

Exact geometric computing

- the general problem
  - algorithms often assume that numbers are real
  - computers do not like real numbers
  - inconsistencies
- in computational geometry
  - numerical errors often lead to crash
    - exact arithmetic
    - filtered arithmetic

Topology and some geometry of real algebraic plane curves

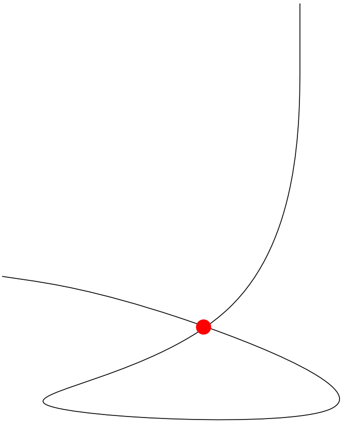
input curve: $f(x, y) = 0$ with $f \in \mathbb{Q}[x, y]$



Topology and some geometry of real algebraic plane curves

input curve: $f(x, y) = 0$ with $f \in \mathbb{Q}[x, y]$

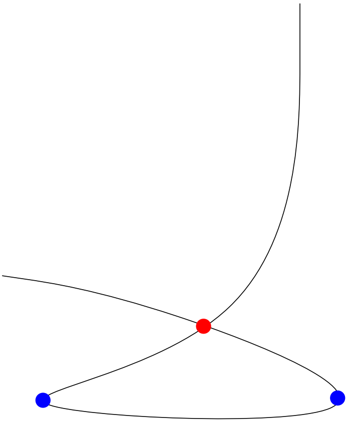
- identify and localize
  - singular points,



Topology and some geometry of real algebraic plane curves

input curve: $f(x, y) = 0$ with $f \in \mathbb{Q}[x, y]$

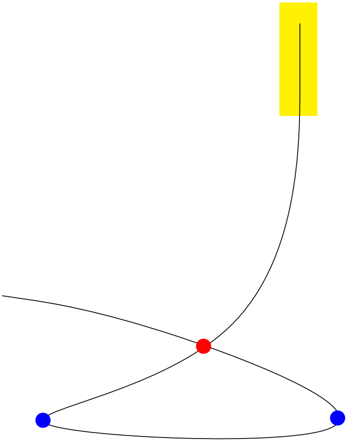
- identify and localize
  - singular points,
  - x-extreme points,



Topology and some geometry of real algebraic plane curves

input curve: $f(x, y) = 0$ with $f \in \mathbb{Q}[x, y]$

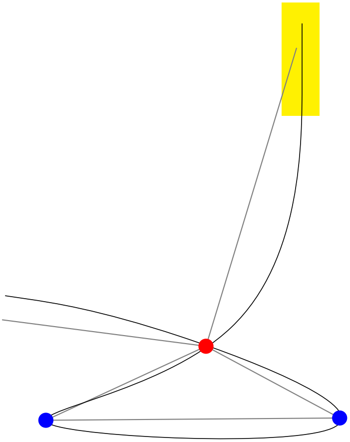
- identify and localize
  - singular points,
  - x-extreme points,
  - vertical asymptotes.



Topology and some geometry of real algebraic plane curves

input curve: $f(x, y) = 0$ with $f \in \mathbb{Q}[x, y]$

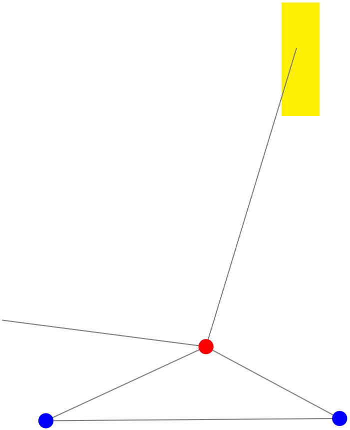
- identify and localize
  - singular points,
  - x-extreme points,
  - vertical asymptotes.
- isotopic approximation of the curve by an arrangement of polylines



Topology and some geometry of real algebraic plane curves

input curve: $f(x, y) = 0$ with $f \in \mathbb{Q}[x, y]$

- identify and localize
  - singular points,
  - x-extreme points,
  - vertical asymptotes.
- isotopic approximation of the curve by an arrangement of polylines
- results in the original coordinate system of the plane



Applications

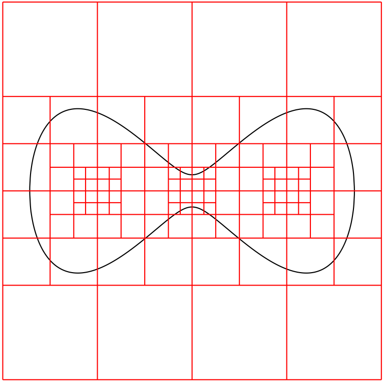
- curve plotting
- computing arrangements of algebraic curves

Applications

- curve plotting
- computing arrangements of algebraic curves

the x-extreme points of each curve have to be computed in the same coordinate system

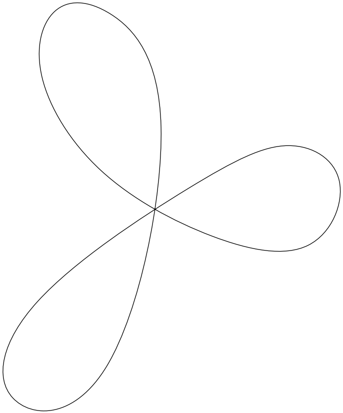
Previous work: subdivision techniques



- fast
- localizable (computation in a given box)
- not certified, unless they
- reach the separation bound

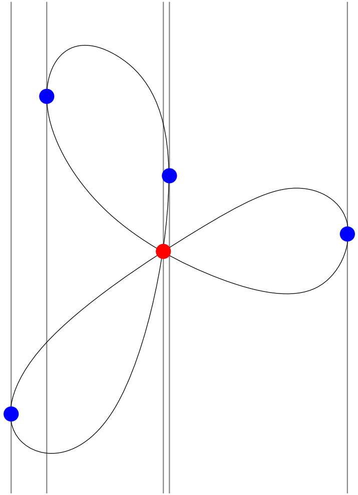
[Lorensen & Cline, 1987]
[Alberti, Mourrain & Wintz, 2008]
[Burr, Choi, Galehouse & Yap, 2008]
[Lin & Yap, 2009]

Cylindrical algebraic decomposition methods



Cylindrical algebraic decomposition methods

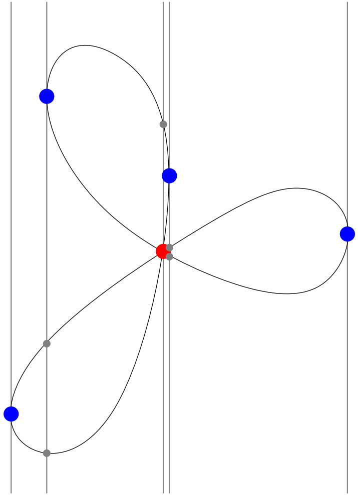
1. projection



Cylindrical algebraic decomposition methods

1. projection

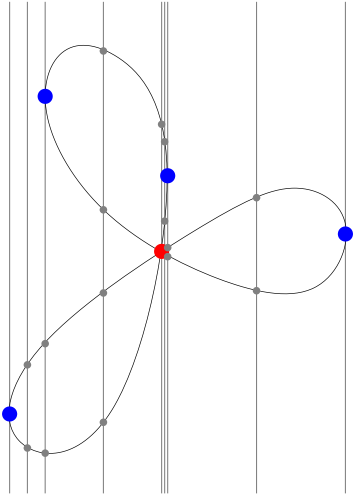
2. lifting



Cylindrical algebraic decomposition methods

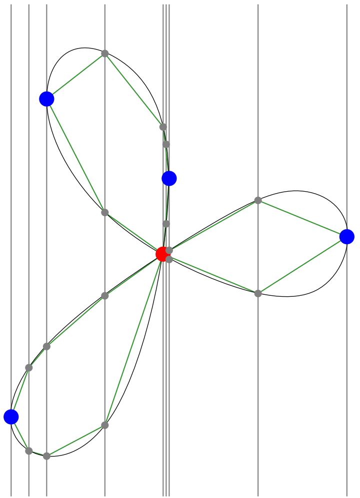
1. projection

2. lifting



Cylindrical algebraic decomposition methods

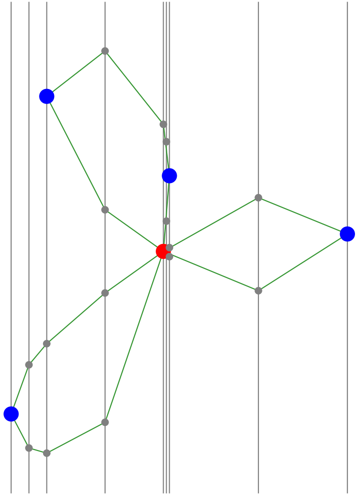
1. projection
2. lifting
3. adjacencies



Cylindrical algebraic decomposition methods

1. projection
2. lifting
3. adjacencies

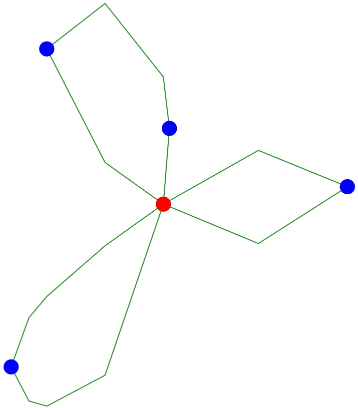
- assume generic position, or detect it, shear and shear back



Cylindrical algebraic decomposition methods

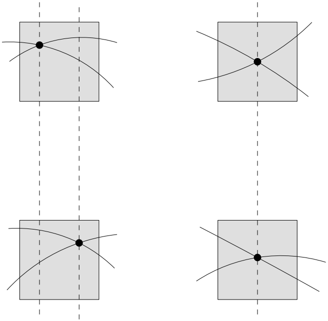
1. projection
2. lifting
3. adjacencies

- assume generic position, or detect it, shear and shear back
- Collins, 1984
- Cad2d [Brown & al.]
  Top [Gonzalez Vega & Necula, 2002]
  Insulate [Seidel & Wolpert, 2005]
  CA [Eigenwillig, Kerber & Wolpert, 2007]



Our algorithm

- it's not a CAD
  - decomposition of the plane into rectangles: the rectangle containing each critical point may overlap in x
  - non-genericity of x-overlapping boxes is not an issue



generic non-generic

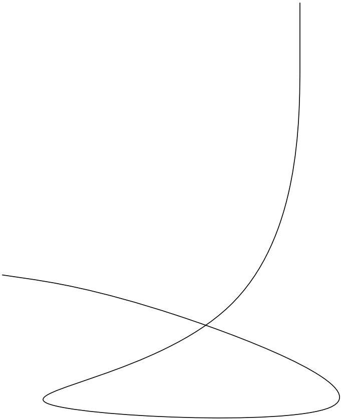
Our algorithm

- it's not a CAD
  - decomposition of the plane into rectangles: the rectangle containing each critical point may overlap in x
  - non-genericity of x-overlapping boxes is not an issue

- replaces sub-resultant sequences and computations with algebraic coefficient polynomials by
  - Gröbner bases
  - Rational Univariate Representations

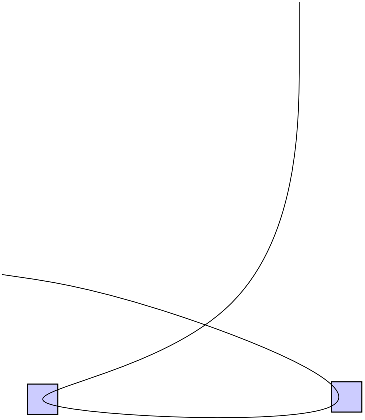
Algorithm outline

1. compute isolating boxes for critical points:



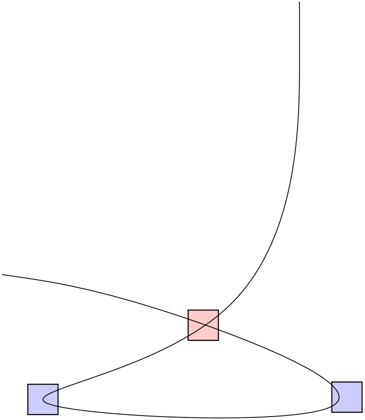
Algorithm outline

1. compute isolating boxes for critical points:
   - extreme points,



Algorithm outline

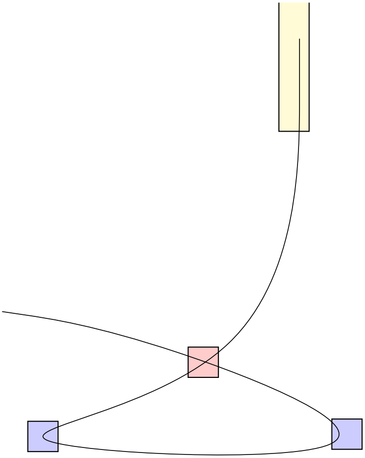
1. compute isolating boxes for critical points:
   - extreme points,
   - singular points and



Algorithm outline

1. compute isolating boxes for critical points:
   - extreme points,
   - singular points and
   - asymptotes;

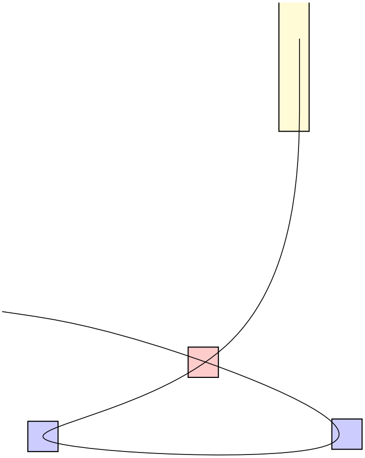
   and refine them until they are disjoint



Algorithm outline

1. compute isolating boxes for critical points:
   - extreme points,
   - singular points and
   - asymptotes;

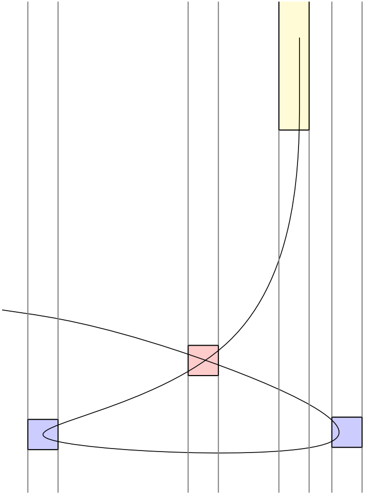
   and refine them until they are disjoint
2. determine the topology in the rectangles of critical points



Algorithm outline

1. compute isolating boxes for critical points:
   - extreme points,
   - singular points and
   - asymptotes;

   and refine them until they are disjoint
2. determine the topology in the rectangles of critical points
3. compute the topology in the rest of the rectangles



Notation

curve: square free polynomial $f \in \mathbb{Q}[x, y]$

Notation

curve: square free polynomial $f \in \mathbb{Q}[x, y]$

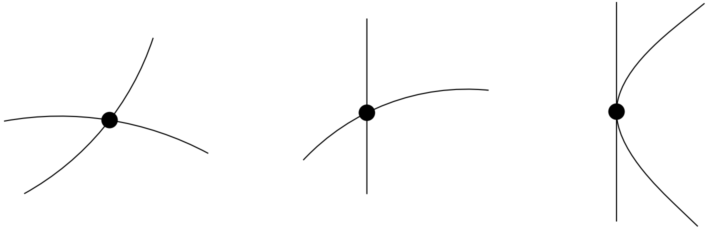
a point $\mathbf{p} = (\alpha, \beta) \in \mathbb{C}^2$ is *x-critical* if $f(\mathbf{p}) = f_y(\mathbf{p}) = 0$,

Notation

curve: square free polynomial $f \in \mathbb{Q}[x, y]$

a point $\mathbf{p} = (\alpha, \beta) \in \mathbb{C}^2$ is *x-critical* if $f(\mathbf{p}) = f_y(\mathbf{p}) = 0$,

- *singular* if $f_x(\mathbf{p}) = 0$, and

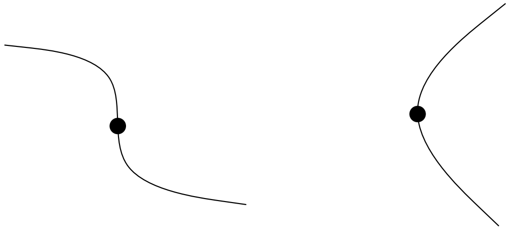


Notation

curve: square free polynomial $f \in \mathbb{Q}[x, y]$

a point $\mathbf{p} = (\alpha, \beta) \in \mathbb{C}^2$ is *x-critical* if $f(\mathbf{p}) = f_y(\mathbf{p}) = 0$,

- *singular* if $f_x(\mathbf{p}) = 0$, and
- *x-extreme* if $f_x(\mathbf{p}) \neq 0$ (i.e. x-critical and non-singular)



The Rational Univariate Representation

S is a bivariate system, RUR $\rightsquigarrow$ univariate polynomial f, such that

$$t \text{ root of } f \iff \left(\frac{g_x(t)}{h(t)}, \frac{g_y(t)}{h(t)}\right) \text{ root of } S$$
$$g_x, g_y, h \in \mathbb{Q}(t)$$

The Rational Univariate Representation

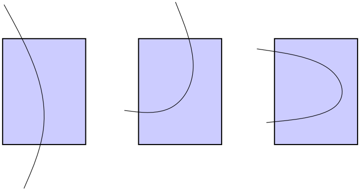
S is a bivariate system, RUR $\rightsquigarrow$ univariate polynomial f, such that

$$t \text{ root of } f \iff \left(\frac{g_x(t)}{h(t)}, \frac{g_y(t)}{h(t)}\right) \text{ root of } S$$
$$g_x, g_y, h \in \mathbb{Q}(t)$$

- the RUR preserves multiplicities
- we obtain the RUR from the Gröbner basis of S
- the roots of f are isolated with Descartes' method
- interval arithmetic for computing the separating boxes of the roots of the system S

Topology at extreme points

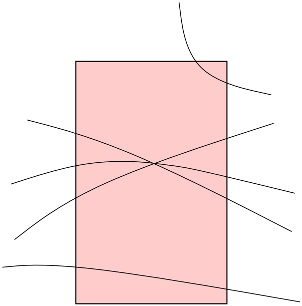
1. isolate the extreme system $S_e = \begin{cases} f(x,y) = 0 \\ \frac{\partial f}{\partial y} = 0 \\ \frac{\partial f}{\partial x} \neq 0 \end{cases}$
2. refine boxes to get only two crossings on the border



3. store the multiplicities in the system S_e for the connection step (see later)

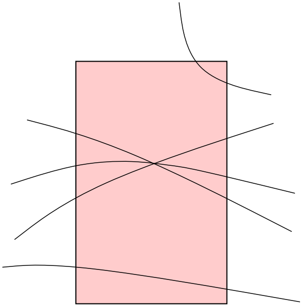
Topology at singularities

1. isolate singular points in boxes



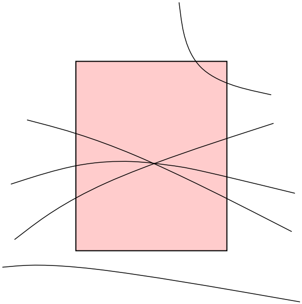
Topology at singularities

1. isolate singular points in boxes
2. compute multiplicities k in fibers



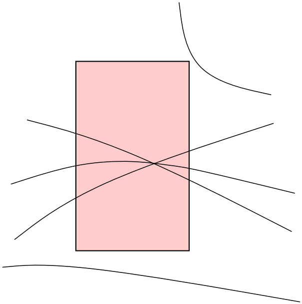
Topology at singularities

1. isolate singular points in boxes
2. compute multiplicities k in fibers
3. refine the box to avoid the curve $f_{y^k} = \frac{\partial^k f}{\partial y^k}$ [Seidel & Wolpert, 2005]



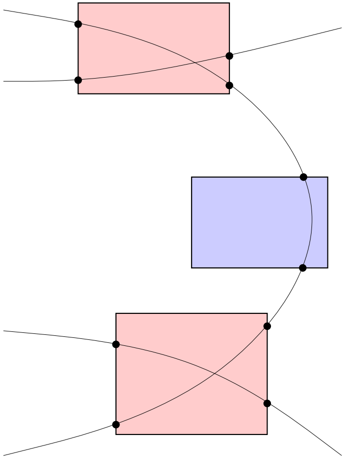
Topology at singularities

1. isolate singular points in boxes
2. compute multiplicities k in fibers
3. refine the box to avoid the curve $f_{y^k} = \frac{\partial^k f}{\partial y^k}$ [Seidel & Wolpert, 2005]
4. refine the box to avoid top/bottom crossings



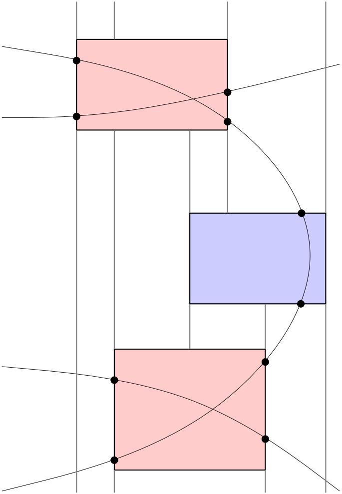
Rectangle decomposition of the plane

- the topology is known inside critical boxes



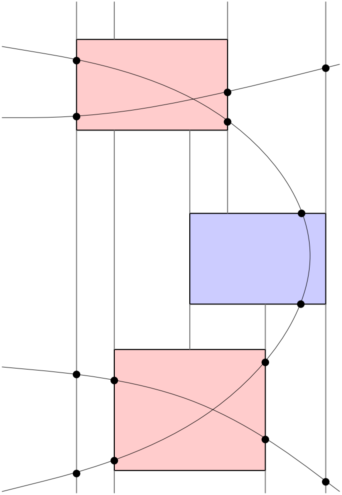
Rectangle decomposition of the plane

- the topology is known inside critical boxes
- compute a vertical decomposition of the plane with respect to these boxes



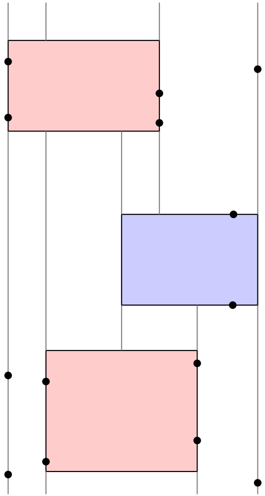
Rectangle decomposition of the plane

- the topology is known inside critical boxes
- compute a vertical decomposition of the plane with respect to these boxes
- compute intersections of the curve with the decomposition

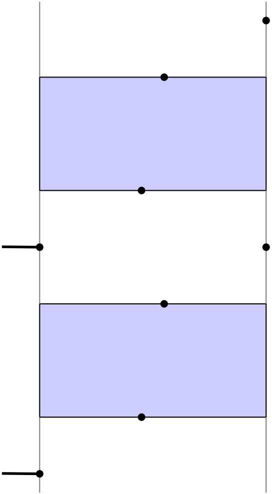


Rectangle decomposition of the plane

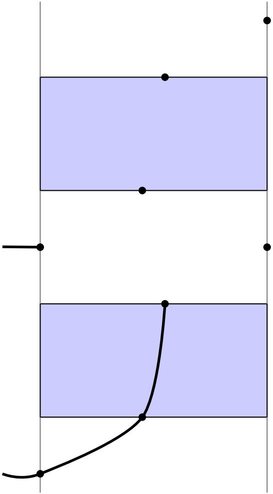
- the topology is known inside critical boxes
- compute a vertical decomposition of the plane with respect to these boxes
- compute intersections of the curve with the decomposition



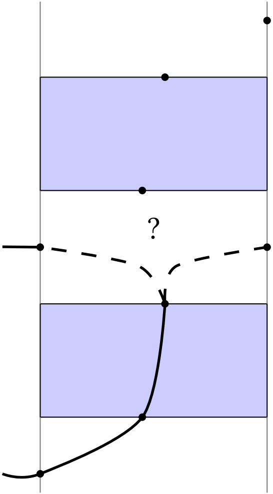
Greedy connection algorithm using multiplicities



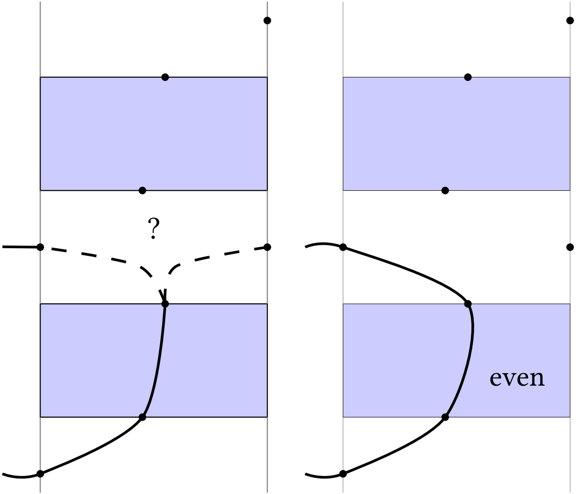
Greedy connection algorithm using multiplicities



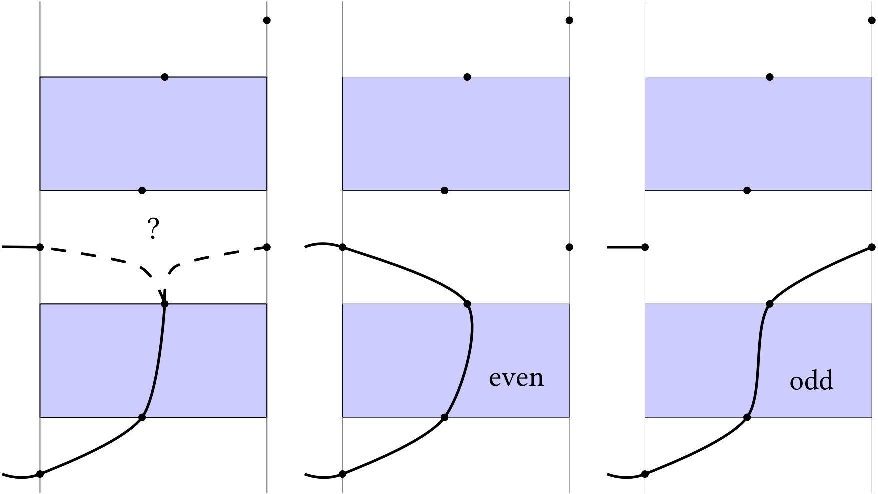
Greedy connection algorithm using multiplicities



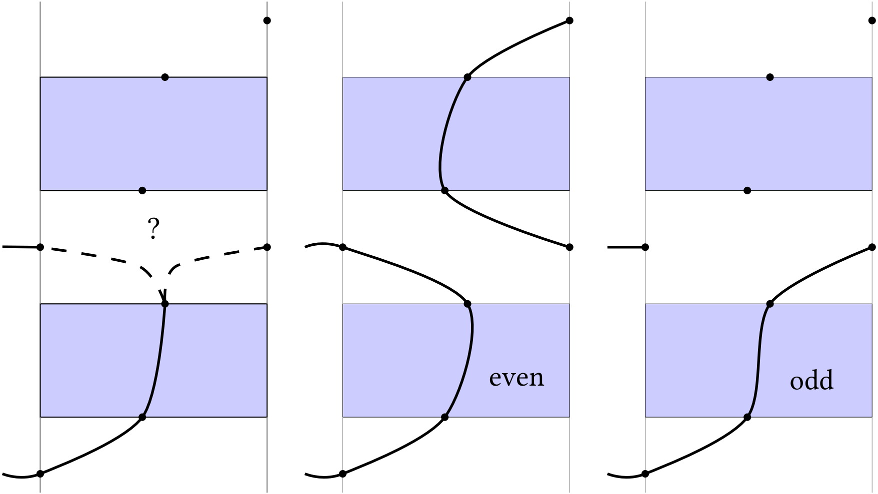
Greedy connection algorithm using multiplicities



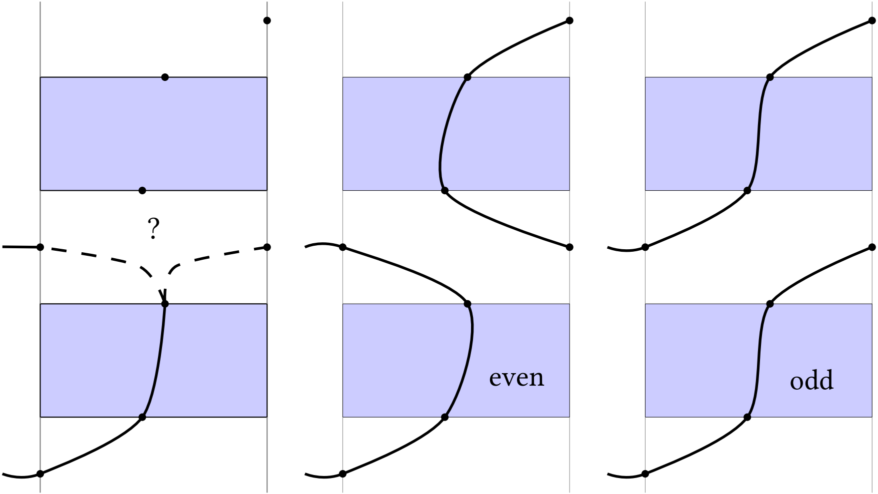
Greedy connection algorithm using multiplicities



Greedy connection algorithm using multiplicities



Greedy connection algorithm using multiplicities



Complexity analysis

- the algorithm runs in $\widetilde{\mathcal{O}}_B(R\,d^{22}\,\tau^2)$, where
  - R: number of real critical points,
  - d: degree of the polynomial f,
  - τ: maximum coefficient bitsize of f.

Complexity analysis

- the algorithm runs in $\widetilde{\mathcal{O}}_B(R\, d^{22}\, \tau^2)$, where
  - R: number of real critical points,
  - d: degree of the polynomial f,
  - τ: maximum coefficient bitsize of f.
- if $N = \max\{d, \tau\}$ and R is $\mathcal{O}(d^2)$, we have $\widetilde{\mathcal{O}}_B(N^{26})$

Complexity analysis

- the algorithm runs in $\widetilde{\mathcal{O}}_B(R\, d^{22}\, \tau^2)$, where
  - R: number of real critical points,
  - d: degree of the polynomial f,
  - τ: maximum coefficient bitsize of f.

- if $N = \max\{d, \tau\}$ and R is $\mathcal{O}(d^2)$, we have $\widetilde{\mathcal{O}}_B(N^{26})$

- does not reflect practical performance

Implementation in Maple

Isotop: +7000 lines of Maple code, using

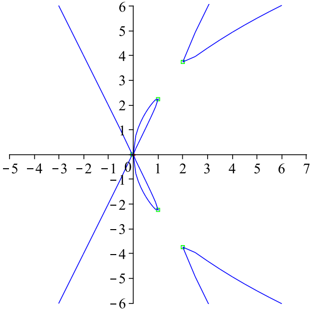
- FGB for Gröbner basis (Faugère)
- RS for RUR and isolation (Rouillier)

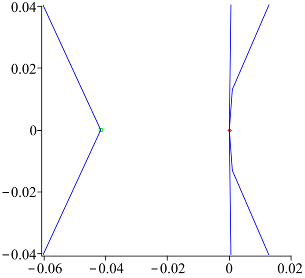
- complete: handles vertical asymptotes and vertical components
- certified

- http://vegas.loria.fr/isotop

Isotop interface

```
ISOTOP:-topology_real_curve(y^4 - 6*y^2*x + x^2 - 4*y^2*x^2 + 24*x^3,
                            verbosity=0,
                            precision=10,
                            plot_graph=true,
                            nb_splits=10);
```



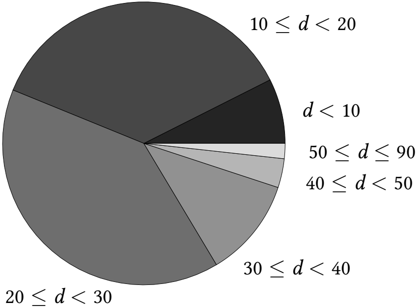


Isotop experiments

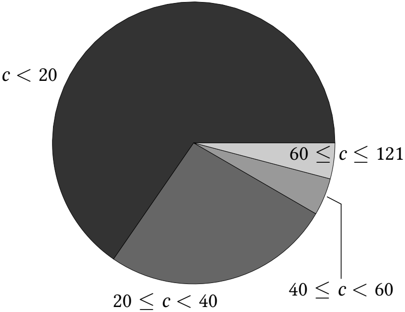
we ran large-scale tests, testing around 600 curves

- random curves
- ACS curves
- O. Labs' tough curves
- resultants of degree-3 random surfaces
- n translations $\prod_{j=0}^{n} f(x, y + j)$
- symmetric polynomials $f^2(x, y) + f^2(x, -y)$

Isotop experiments: input curves

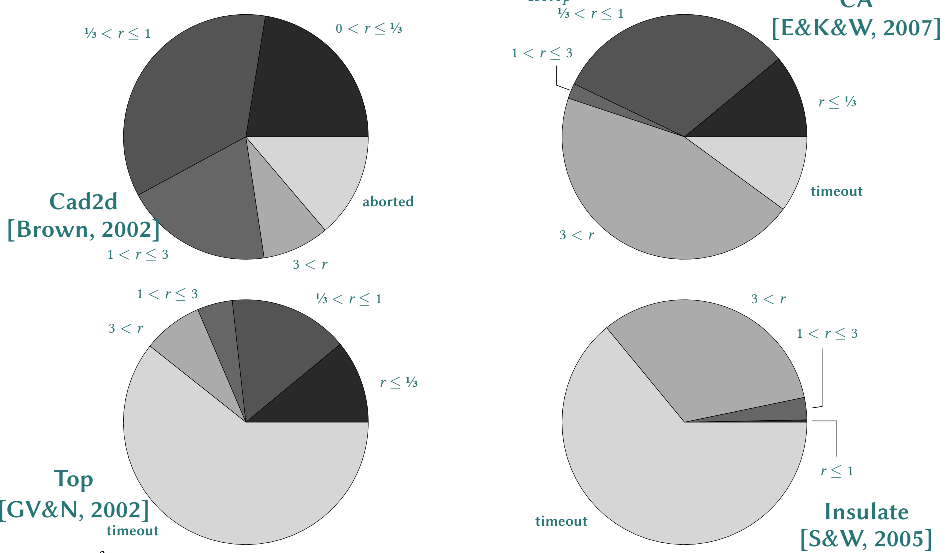


degrees



number of critical points

Experiments: results, $r = \frac{time}{time_{Isotop}}$



Experiments: conclusions

- faster for
  - non-generic curves
  - high degree curves

Experiments: conclusions

- faster for
  - non-generic curves
  - high degree curves

- slower for
  - random curves
  - curves with high-tangency points

Experiments: conclusions

- faster for
  - non-generic curves
  - high degree curves
- slower for
  - random curves
  - curves with high-tangency points
- why?

Algebraic algorithms in CGAL

- Isotop is in Maple, but
  - it is not a standard in the CG community
  - Maple programs cannot be used as libraries

Algebraic algorithms in CGAL

- Isotop is in Maple, but
  - it is not a standard in the CG community
  - Maple programs cannot be used as libraries

- in general
  - lack of algebro-geometric tools in C or C++

Algebraic algorithms in CGAL

- Isotop is in Maple, but
  - it is not a standard in the CG community
  - Maple programs cannot be used as libraries

- in general
  - lack of algebro-geometric tools in C or C++

- CGAL
  - C++ library
  - standard in the community
  - generic programming

Algebraic algorithms in CGAL

- Isotop is in Maple, but
  - it is not a standard in the CG community
  - Maple programs cannot be used as libraries
- in general
  - lack of algebro-geometric tools in C or C++
- CGAL
  - C++ library
  - standard in the community
  - generic programming
- equip CGAL with algebraic tools
  - also useful for future algorithms

Algebraic tools in CGAL

- specific non-linear objects, particular algorithms
  - arrangements of conics

Algebraic tools in CGAL

- specific non-linear objects, particular algorithms
  - arrangements of conics

- specific non-linear objects, kernels
  - circles
  - spheres

Algebraic tools in CGAL

- specific non-linear objects, particular algorithms
  - arrangements of conics
- specific non-linear objects, kernels
  - circles
  - spheres
- curves of arbitrary degree, algebraic kernels
  - univariate and bivariate
  - many variables

Algebraic Kernel

combines algebra and geometry for manipulating non-linear objects

Algebraic Kernel

combines algebra and geometry for manipulating non-linear objects

- features
  - root finding
  - algebraic number comparison
  - all related polynomial operations

Algebraic Kernel

combines algebra and geometry for manipulating non-linear objects

- features
  - root finding
  - algebraic number comparison
  - all related polynomial operations

- concepts and models

Algebraic Kernel

combines algebra and geometry for manipulating non-linear objects

- features
  - root finding
  - algebraic number comparison
  - all related polynomial operations
- concepts and models
- model of univariate algebraic kernel

Tools we use

- GMP
  - GNU multiple-precision number library

Tools we use

- GMP
  - GNU multiple-precision number library
- RS
  - univariate polynomials with integer coefficients
  - interval Descartes algorithm
  - coded in C
  - memory management
  - multiple platforms (Unix, Mac OS, Win)

Tools we use

- GMP
  - GNU multiple-precision number library
- RS
  - univariate polynomials with integer coefficients
  - interval Descartes algorithm
  - coded in C
  - memory management
  - multiple platforms (Unix, Mac OS, Win)
- MPFR
  - arbitrary multiple-precision floating-point numbers

Tools we use

- GMP
  - GNU multiple-precision number library
- RS
  - univariate polynomials with integer coefficients
  - interval Descartes algorithm
  - coded in C
  - memory management
  - multiple platforms (Unix, Mac OS, Win)
- MPFR
  - arbitrary multiple-precision floating-point numbers
- MPFI
  - arbitrary multiple-precision floating-point intervals

Our algebraic kernel

- 8000 lines of code

Our algebraic kernel

- 8000 lines of code

- root isolation
  - uses RS
  - gives as result algebraic numbers
    - isolating interval: MPFI
    - pointer to a polynomial

Our algebraic kernel

- 8000 lines of code

- root isolation
  - uses RS
  - gives as result algebraic numbers
    - isolating interval: MPFI
    - pointer to a polynomial

- comparison of algebraic numbers
  - easy when intervals do not overlap
  - otherwise, test for equality
    - greatest common divisor (gcd)
    - algebraic number refinement

Auxiliar operations

- gcd
  - bottleneck of the implementation (used for comparisons and square free factorizations)
  - two modular implementations
  - fast detection of coprime polynomials

Auxiliar operations

- gcd
  - bottleneck of the implementation (used for comparisons and square free factorizations)
  - two modular implementations
  - fast detection of coprime polynomials
- refinement
  - bisection
  - quadratic refinement

Benchmarks

- software
  - MPII's algebraic kernel (using CORE NT)
  - Synaps/Mathemagix code (using NCF2 and GMP NT)
  - our algebraic kernel

Benchmarks

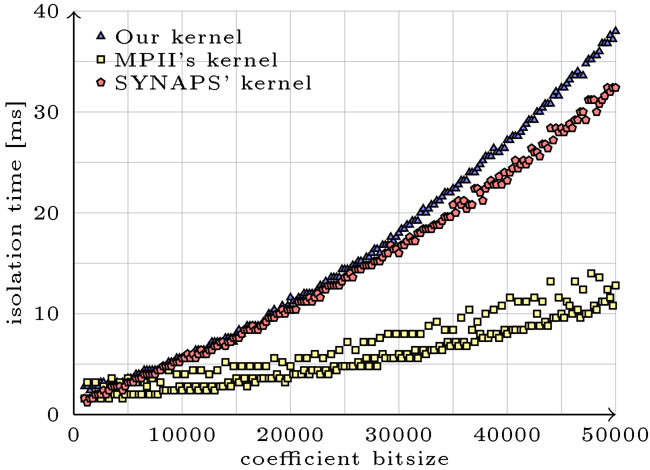
- software
  - MPII's algebraic kernel (using CORE NT)
  - Synaps/Mathemagix code (using NCF2 and GMP NT)
  - our algebraic kernel
- functionalities
  - root isolation
  - algebraic number comparison
  - application: arrangement construction

Benchmark data

- first time such a big amount of data for polynomials is tested
- 60,000 polynomials (3.8 Gb)
- several weeks in total

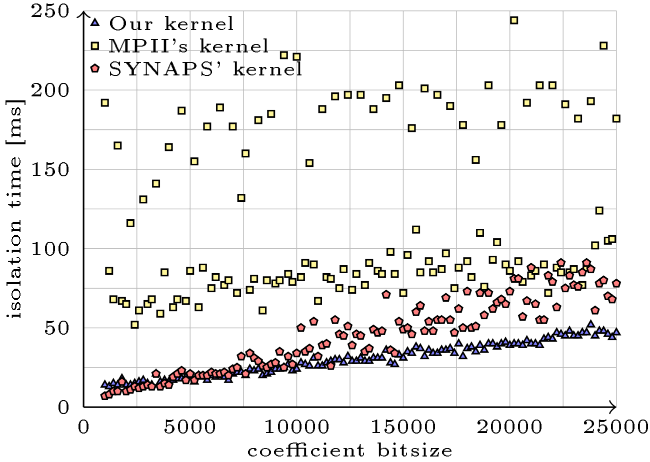
Root isolation: varying bitsize

degree-12 random polynomials



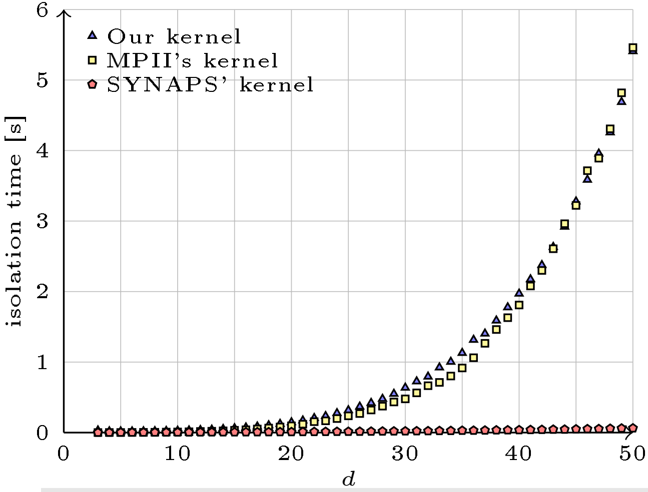
Root isolation: varying bitsize II

degree-100 random polynomials



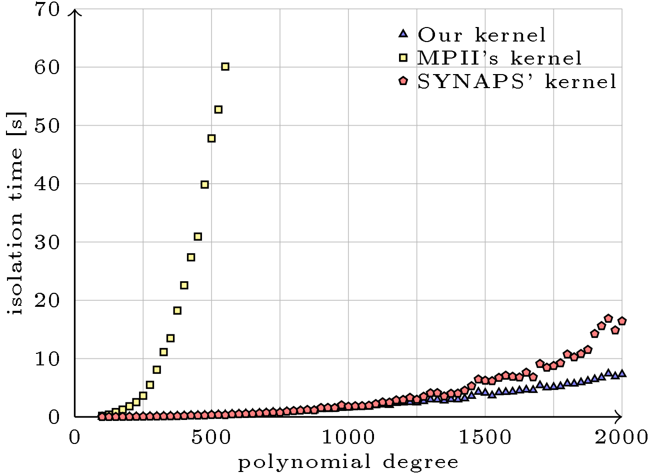
Root isolation: Mignotte polynomials

$f = x^d - 2(kx-1)^2$



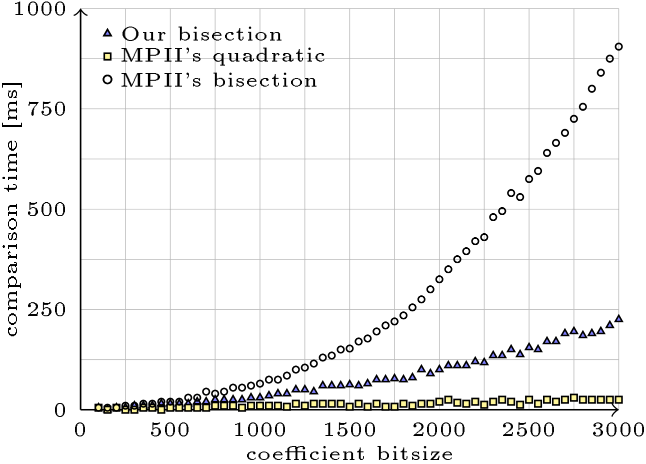
Root isolation: varying degree

bitsize-1000 random polynomials



Algebraic number comparison

almost-identical polynomials of degree 20



Arrangement benchmarks

test programs

- CGAL's arrangement package (Tel-Aviv University)
- parameterised with
  - a traits class that uses CORE
  - a new traits class for our kernel

Arrangement benchmarks

test programs

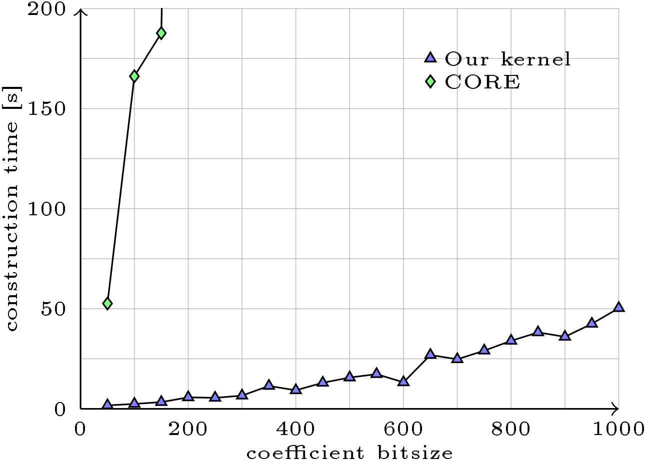
- CGAL's arrangement package (Tel-Aviv University)
- parameterised with
  - a traits class that uses CORE
  - a new traits class for our kernel

test data

- generate n random polynomials
- shift them vertically m times
- $n(m+1)$ polynomials of bitsize τ and degree d
- we fix $n = 5$ and $m = 4$ here

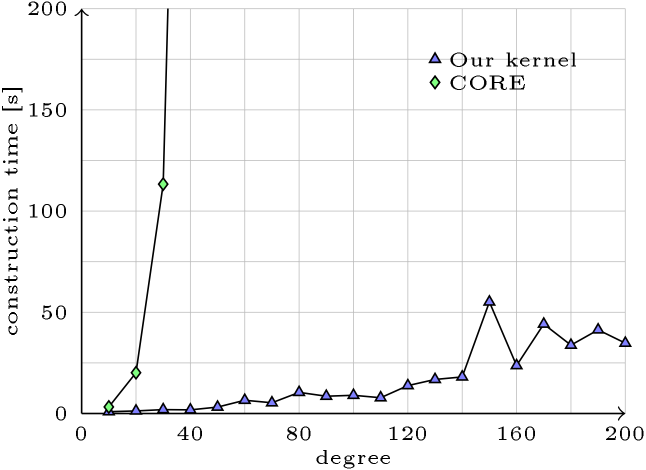
Arrangements: varying bitsize

$d = 20$



Arrangements: varying degree

$\tau = 32$



Experimentation highlights

- our algebraic kernel is included in CGAL 3.6
- MPII kernel is unstable due to external library issues
- Synaps kernel is not complete

Experimentation highlights

- our algebraic kernel is included in CGAL 3.6
- MPII kernel is unstable due to external library issues
- Synaps kernel is not complete
- root isolation
  - our kernel is faster for high degrees and bitsizes
  - Synaps isolation performs much better in Mignotte's polynomials

Experimentation highlights

- our algebraic kernel is included in CGAL 3.6
- MPII kernel is unstable due to external library issues
- Synaps kernel is not complete
- root isolation
  - our kernel is faster for high degrees and bitsizes
  - Synaps isolation performs much better in Mignotte's polynomials
- algebraic number refinement
  - MPII quadratic refinement is really fast

Experimentation highlights

- our algebraic kernel is included in CGAL 3.6
- MPII kernel is unstable due to external library issues
- Synaps kernel is not complete
- root isolation
  - our kernel is faster for high degrees and bitsizes
  - Synaps isolation performs much better in Mignotte's polynomials
- algebraic number refinement
  - MPII quadratic refinement is really fast
- arrangement experiments
  - validate the algebraic kernel approach

Conclusions

- algorithm development
  - curve topology analysis
  - no special treatment of non-generic cases,
  - results in the original coordinate system
  - uses Rational Univariate Representations, to avoid sub-resultant sequences

Conclusions

- algorithm development
  - curve topology analysis
  - no special treatment of non-generic cases,
  - results in the original coordinate system
  - uses Rational Univariate Representations, to avoid sub-resultant sequences

- implementation
  - Maple implementation of the topology algorithm
  - CGAL univariate algebraic kernel
  - thorough benchmarking

Conclusions

- algorithm development
  - curve topology analysis
  - no special treatment of non-generic cases,
  - results in the original coordinate system
  - uses Rational Univariate Representations, to avoid sub-resultant sequences

- implementation
  - Maple implementation of the topology algorithm
  - CGAL univariate algebraic kernel
  - thorough benchmarking

- analysis of algorithms
  - output-sensitive complexity analysis

Perspectives

- improve handling of some curves
  - algebraic approach that is *always* efficient
  - arrangements of curves
- topology of surfaces, meshing
- include Isotop in Maple
- bivariate and multivariate algebraic kernel
- tighter complexity bounds