Solving bivariate systems and topology of plane curves

Yacine Bouzidi

VEGAS Team

Inria Nancy - Grand Est
`Yacine.bouzidi@inria.fr`

18 mars 2014

Loria
Laboratoire lorrain de recherche en informatique et ses applications

informatics mathematics
Inria

UNIVERSITÉ DE LORRAINE

Plan

1. Problem and motivation
2. Rational Univariate Representation
3. Theoretical worst-case complexity algorithm
4. Practical algorithm
5. Experiments
6. Conclusion and perspectives

Overview

1 Problem and motivation

2 Rational Univariate Representation

3 Theoretical worst-case complexity algorithm

4 Practical algorithm

5 Experiments

6 Conclusion and perspectives

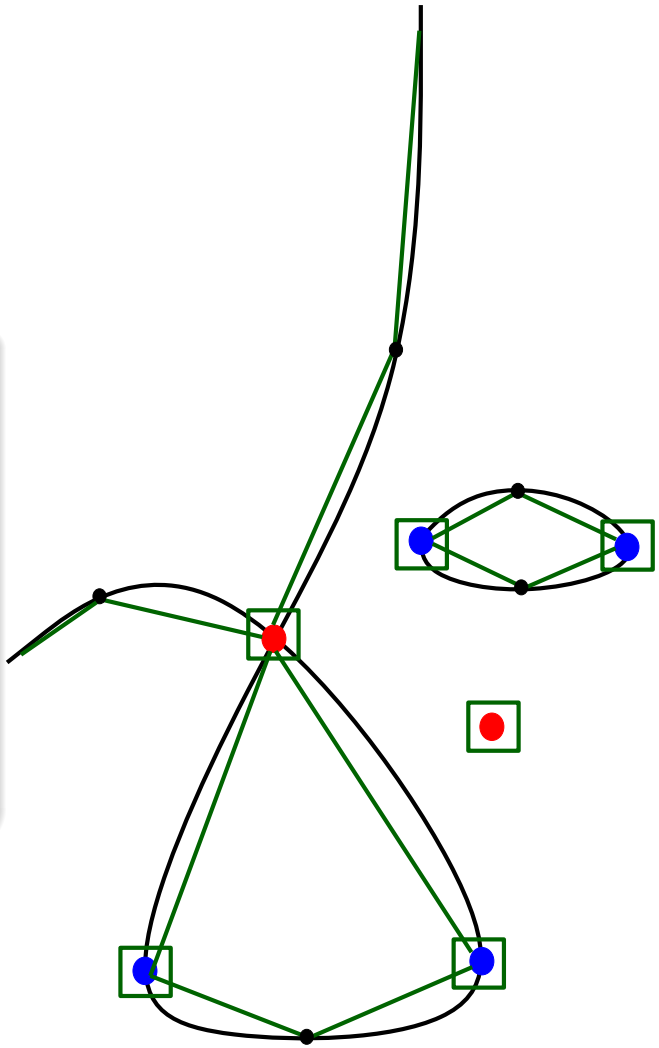
Topology of plane algebraic curves

Input : $f(x, y) \in \mathbb{Z}[x, y]$

Output : Isotopic approximation of $f(x, y) = 0$

Algorithm

1. Identify and approximate the critical points of f
   - Singular points and x-extreme points
     $f(x, y) = \frac{\partial f}{\partial y}(x, y) = 0$
2. Connect these points by means of polylines
   - Need the topology around the critical points



Topology of plane algebraic curves

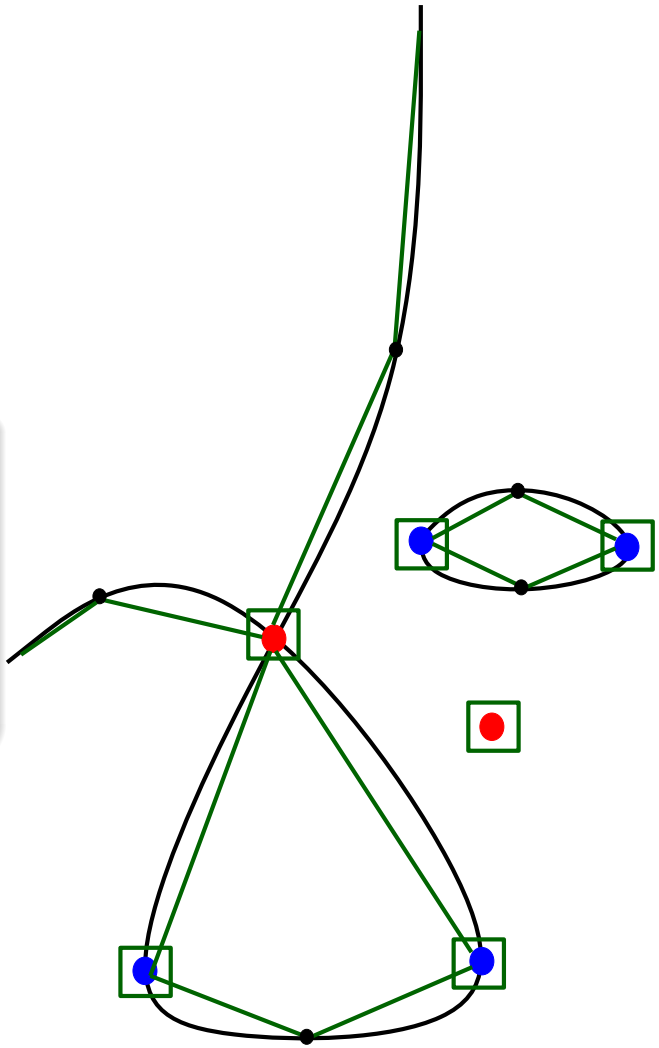
Input : $f(x, y) \in \mathbb{Z}[x, y]$

Output : Isotopic approximation of $f(x, y) = 0$

Main step

1. Identify and approximate the critical points of f
   - Singular points and x-extreme points
     $f(x, y) = \frac{\partial f}{\partial y}(x, y) = 0$

$\rightsquigarrow$ Solve systems of bivariate polynomials



Problem

Problems

- Solving systems of the form $\{P = Q = 0\}$ with $P, Q \in \mathbb{Z}[x, y]$: Isolating the real solutions
- Performing operations with the solutions (eg. IsZero, SignAt, etc)

Goals

- Correctness : Mathematically correct result
- Completeness : No restriction on the input
- Efficiency : In theory (bit complexity). In practice (running time)

Previous work : Solving algebraic systems

Numerical methods : + Fast / - Correctness

- Subdivision
- Homotopy continuation

Formal solutions : + Correctness / - Symbolic computation

- Resultant
- Triangular decomposition
- Rational parametrization
  - Gröbner basis + linear algebra
  - Geometric resolution
  - Chow form

Overview

1 Problem and motivation

2 Rational Univariate Representation

3 Theoretical worst-case complexity algorithm

4 Practical algorithm

5 Experiments

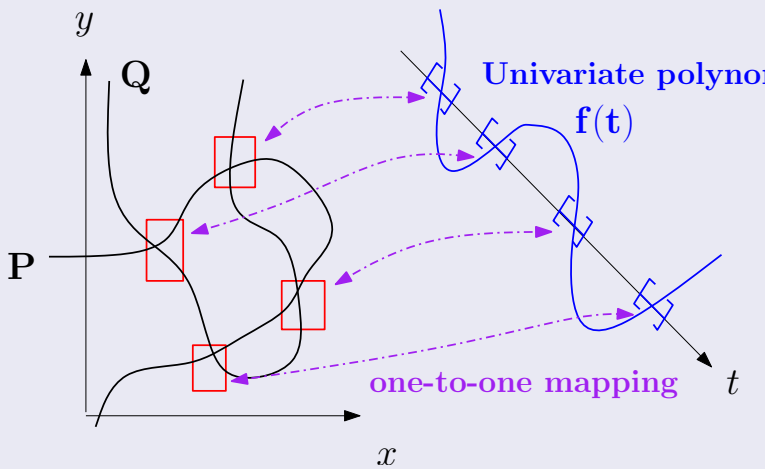
6 Conclusion and perspectives

Rational Univariate Representation

Definition (Rouillier 99)

Let $\langle P, Q\rangle$ be a zero-dim ideal and V its variety. A RUR of $\langle P, Q\rangle$ is given by :

- A linear form $x + ay$ that separates the points of V
- A one-to-one mapping between the roots of an univariate polynomial f and the solutions of V



$$\begin{array}{ccc} \mathbf{V}(\{\mathbf{P},\mathbf{Q}\}) & \rightarrow & \mathbf{V}(\mathbf{f}) \\ (\mathbf{x},\mathbf{y}) & \mapsto & \mathbf{x}+\mathbf{a}\mathbf{y} \\ \left(\frac{\mathbf{f_x(t)}}{\mathbf{f_1(t)}}, \frac{\mathbf{f_y(t)}}{\mathbf{f_1(t)}}\right) & \leftarrow\!\shortmid & \mathbf{t} \end{array}$$

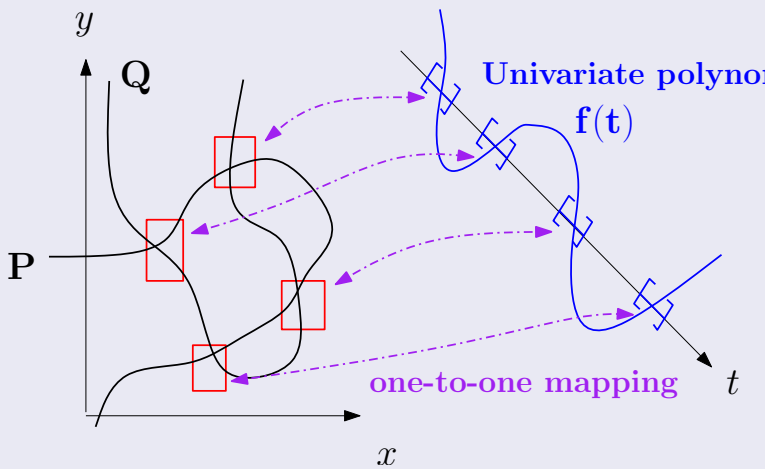
- Preserves the multiplicities

Rational Univariate Representation

Definition (Rouillier 99)

Let $\langle P, Q\rangle$ be a zero-dim ideal and V its variety. A RUR of $\langle P, Q\rangle$ is given by :

- A linear form $x + ay$ that separates the points of V
- A one-to-one mapping between the roots of an univariate polynomial f and the solutions of V



$$\begin{array}{rcl} \mathbf{V(\{P,Q\})} & \rightarrow & \mathbf{V(f)} \\ \mathbf{(x,y)} & \mapsto & \mathbf{x+ay} \\ \mathbf{\left(\frac{f_x(t)}{f_1(t)}, \frac{f_y(t)}{f_1(t)}\right)} & \leftarrow\!\shortmid & \mathbf{t} \end{array}$$

- Preserves the multiplicities

Advantages : simple numerical approximations

Notation

- $P, Q \in \mathbb{Z}[x, y]$
- d : the total degree of P and Q
- τ : the maximum bitsize of the coefficients of P and Q
- O_B : the bit complexity
- $\widetilde{O}$: omit polylogarithmic factors

Contribution (1/2)

Theoretical deterministic algorithm for computing the RUR

1. A new algorithm for computing a separating form
2. RUR via simple formulas
3. New bound on the bitsize of the RUR

Contribution (1/2)

Theoretical deterministic algorithm for computing the RUR

1. A new algorithm for computing a separating form $\widetilde{O}_B(d^8 + d^7\tau)$
   - Previous bound in $\widetilde{O}_B(d^{10} + d^9\tau)$ [Diochnos et al. 09]
2. RUR via simple formulas
3. New bound on the bitsize of the RUR

Contribution (1/2)

Theoretical deterministic algorithm for computing the RUR

1. A new algorithm for computing a separating form $\widetilde{O}_B(d^8 + d^7\tau)$
   - Previous bound in $\widetilde{O}_B(d^{10} + d^9\tau)$ [Diochnos et al. 09]
2. RUR via simple formulas : $\widetilde{O}_B(d^7 + d^6\tau)$
   - RUR via linear algebra in $\mathbb{Q}[x, y]/\langle P, Q\rangle$ [Rouillier. 96] : $\widetilde{O}_B(d^{14}\tau)$
   - Rational parametrizations [Gonzalez Vega et al. 96] : $\widetilde{O}_B(d^7 + d^6\tau)$
3. New bound on the bitsize of the RUR

Contribution (1/2)

Theoretical deterministic algorithm for computing the RUR

1. A new algorithm for computing a separating form $\widetilde{O}_B(d^8 + d^7\tau)$
   - Previous bound in $\widetilde{O}_B(d^{10} + d^9\tau)$ [Diochnos et al. 09]
2. RUR via simple formulas : $\widetilde{O}_B(d^7 + d^6\tau)$
   - RUR via linear algebra in $\mathbb{Q}[x, y]/\langle P, Q\rangle$ [Rouillier. 96] : $\widetilde{O}_B(d^{14}\tau)$
   - Rational parametrizations [Gonzalez Vega et al. 96] : $\widetilde{O}_B(d^7 + d^6\tau)$
3. New bound on the bitsize of the RUR : $\widetilde{O}(d^2 + d\tau)$
   - $\widetilde{O}(d^2\tau)$ [Dahan and Schost. 04] (radical systems)
   - d rational parametrizations of bitsize $\widetilde{O}(d^2 + d\tau)$

Contribution (2/2)

Practical probabilistic algorithms for computing the RUR

- Random linear form + multi-modular approach (CRT)
  - Monte-Carlo algorithm : $\widetilde{O}_B(d^6 + d^5\tau)$
  - Las-Vegas algorithms : $\widetilde{O}_B(d^7 + d^6\tau)$ and $\widetilde{O}_B(d^6 + d^5\tau)$

Contribution (2/2)

Practical probabilistic algorithms for computing the RUR

- Random linear form + multi-modular approach (CRT)
  - Monte-Carlo algorithm : $\widetilde{O}_B(d^6 + d^5\tau)$
  - Las-Vegas algorithms : $\widetilde{O}_B(d^7 + d^6\tau)$ and $\widetilde{O}_B(d^6 + d^5\tau)$

Numerical approximation

- Numerical approximations of the real solutions : $\widetilde{O}_B(d^6 + d^5\tau)$

Contribution (2/2)

Practical probabilistic algorithms for computing the RUR

- Random linear form + multi-modular approach (CRT)
  - Monte-Carlo algorithm : $\widetilde{O}_B(d^6 + d^5\tau)$
  - Las-Vegas algorithms : $\widetilde{O}_B(d^7 + d^6\tau)$ and $\widetilde{O}_B(d^6 + d^5\tau)$

Numerical approximation

- Numerical approximations of the real solutions : $\widetilde{O}_B(d^6 + d^5\tau)$

Implementation and Experiments

- Bivariate solver : **RS3** (F. Rouillier) + **AK2** in CGAL (Myself)
  - Las-Vegas algorithm + numerical approximation
- Topology computation : **ISOTOP2**
- Comparison with state-of-the-art implementations

Overview

1 Problem and motivation

2 Rational Univariate Representation

3 Theoretical worst-case complexity algorithm

4 Practical algorithm

5 Experiments

6 Conclusion and perspectives

Theoretical RUR computation

1. Compute a separating linear form $x + ay$
2. Compute the polynomials of the RUR associated to $x + ay$

Theoretical RUR computation

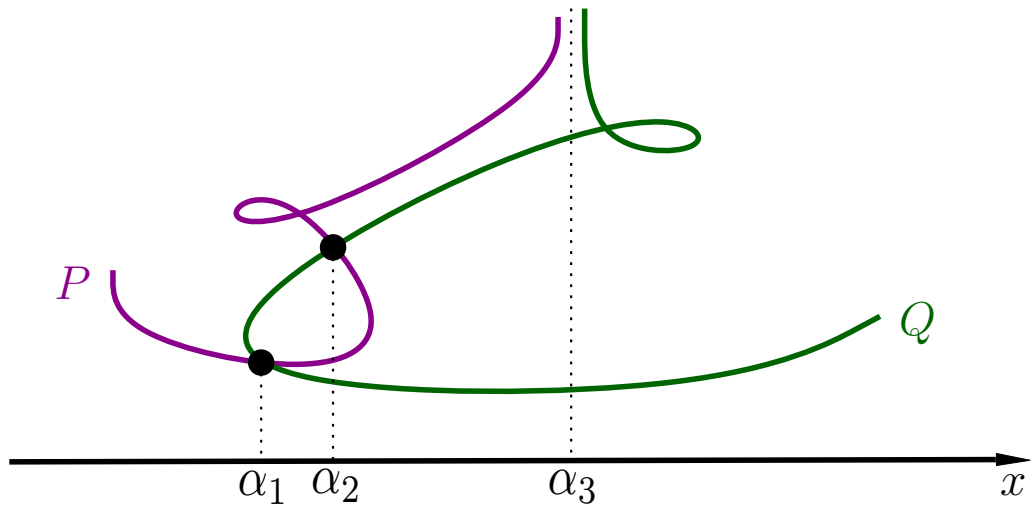
1. Compute a separating linear form $x + ay$
2. Compute the polynomials of the RUR associated to $x + ay$

Resultant

$$P = \sum_{i=0}^{p} a_i(x)y^i \quad \text{and} \quad Q = \sum_{i=0}^{q} b_i(x)y^i \quad \text{in} \quad \mathbb{Z}[x, y]$$

Geometric definition

$Res_y(P, Q) \in \mathbb{Z}[x]$ is a polynomial whose roots are the projections on the x-axis of the intersection points of P and Q (possibly at infinity)



Computing the RUR associated to $x + ay$

- $x + ay$ is a separating form of $V(\langle P, Q\rangle)$
- $t = x + sy$ and $R(t, s) = Res_y(P(t - sy, y), Q(t - sy, y))$
- $L_R(s)$ is the leading coefficient of $R(t, s)$

Theorem (RUR's formulas)

$$f(t) = \frac{R(t, a)}{L_R(a)} \qquad f_1(t) = \frac{f'(t)}{\gcd(f(t), f'(t))}$$

$$f_y(t) = \frac{\frac{\partial R}{\partial s}(t, a) - f(t)\frac{\partial L_R}{\partial s}(a)}{L_R(a)\gcd(f(t), f'(t))} \qquad f_x(t) = tf_1(t) - d_t(f)\overline{f(t)} - af_y(t).$$

Complexity analysis

Theorem

$P, Q \in \mathbb{Z}[x, y]$, degree : d, bitsize : τ, bitsize of $x + ay$: τ_a.

1. *The RUR of $\langle P, Q \rangle$ is computed in $\widetilde{O}_B(d^7 + d^6(\tau + \tau_a))$*
2. *The polynomials of the RUR have bitsize in $\widetilde{O}(d^2 + d(\tau + \tau_a))$*

Complexity analysis

Theorem

$P, Q \in \mathbb{Z}[x, y]$, degree : d, bitsize : τ, bitsize of $x + ay$: τ_a.

1. *The RUR of $\langle P, Q \rangle$ is computed in $\widetilde{O}_B(d^7 + d^6(\tau + \tau_a))$*
2. *The polynomials of the RUR have bitsize in $\widetilde{O}(d^2 + d(\tau + \tau_a))$*

Proof

RUR's polynomials are specialization at $s = a$ of some factor of $R(t, s)$ or one of its partial derivatives

- $R(t, s)$ has bitsize in $\widetilde{O}(d^2 + d\tau)$ (Hadamard bound)
- Any factor of $R(t, s)$ with integer coefficients has bitsize in $\widetilde{O}(d^2 + d\tau)$ (Mignotte)
- After specialization at a : $\widetilde{O}(d^2 + d(\tau + \tau_a))$

Theoretical RUR computation

1. Compute a separating linear form $x + ay$
2. Compute the polynomials of the RUR associated to $x + ay$

Theoretical RUR computation

1. Compute a separating linear form $x + ay$
2. Compute the polynomials of the RUR associated to $x + ay$

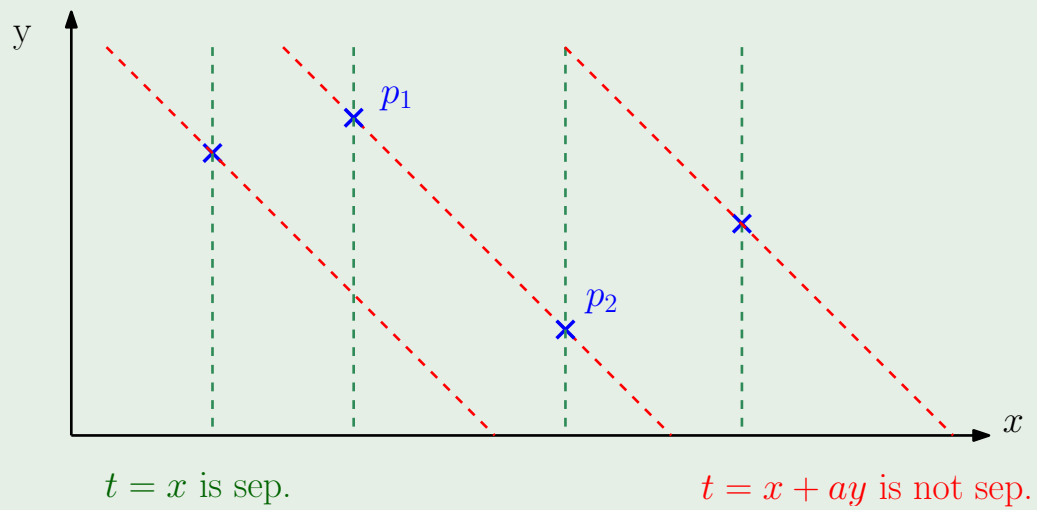
Separating linear form for bivariate systems

Definition

$t = x + ay$ separates the solutions of $V(P, Q)$

- If and only if, the map $(x, y) \mapsto x + ay$ is injective on $V(P, Q)$.
- If and only if the shear $t = x + ay, y = y$ sends the system in generic position with respect to the first coordinate t

Example

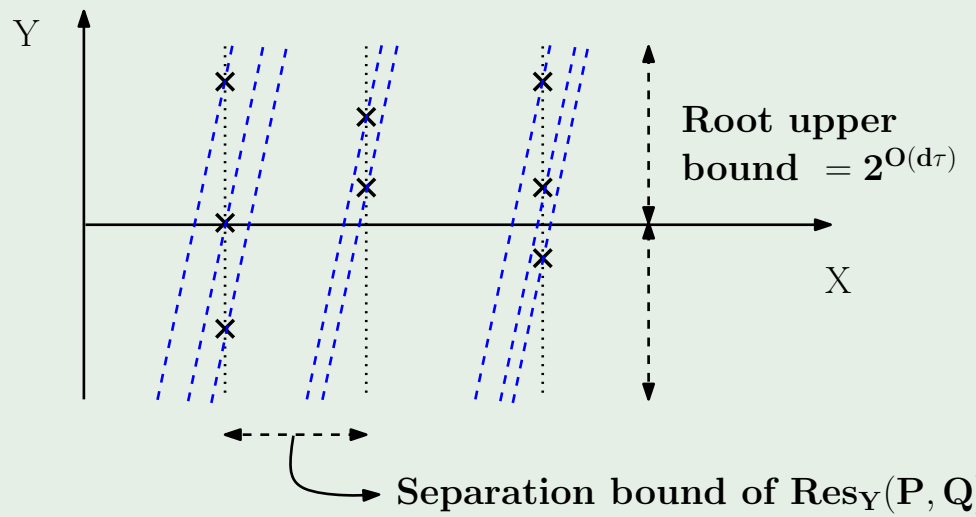


$$P, Q \in \mathbb{Z}[x, y]$$
$$V(P, Q) = \{p_1, p_2, \ldots\}$$

Previous work

- For a large enough a, $x + ay$ is separating

Example



[Cheng et al. ISSAC09]
adaptative version

Drawback : bitsize of a is in $\widetilde{O}(d^3\tau) \rightsquigarrow$ Impact the complexity of the RUR : $\widetilde{O}_B(d^7 + d^6(\tau + \tau_a))$.

Classical algorithm

- **A separating form with bitsize in $O(\log d) = \widetilde{O}(1)$:**

  There are at most $\binom{d^2}{2} < d^4$ bad choices of a which is the maximum number of alignements defined by at most d^2 solutions (Bézout's bound)

Classical algorithm

- Compute $R(t,s) = Res_y(P(t-sy,y), Q(t-sy,y))$
- For $d^4 > \binom{d^2}{2}$ choices of a
  - compute the polynomial $R(t,a)$, the specialization of $R(t,s)$ at a
  - compute $\overline{R(t,a)}$ the squarefree part of $R(t,a)$
- Select an a for which the degree of $\overline{R(t,a)}$ is maximal

Bit-complexity $\rightsquigarrow \widetilde{O}_B(d^{10} + d^9\tau)$ [Diochnos et al. 09]

Our algorithm

- Work over $\frac{\mathbb{Z}}{p\mathbb{Z}}$ to avoid coefficient swell
  - Consider the system $\{P_p = P \mod p, Q_p = Q \mod p\}$
- Previous algorithm then runs in $\widetilde{O}_B(d^8)$
- **Problem :** $x + ay$ is separating over $\frac{\mathbb{Z}}{p\mathbb{Z}}$ does not imply that it is also separating over $\mathbb{Z}\ldots$ Except under some conditions !

Theorem

Let p be a prime such that $Lc_y(P(t - sy, y))Lc_y(Q(t - sy, y))$ *do not vanish modulo p and* $\#V(\langle P_p, Q_p\rangle) = \#V(\langle P, Q\rangle)$ *then,*
$x + ay$ *separates* $V(\langle P_p, Q_p\rangle) \Rightarrow x + ay$ *separates* $V(\langle P, Q\rangle)$

Goal : compute a prime p satisfying the above conditions

Our algorithm

Lemma 1

Let p be a prime number that does not cancel some leading coefficients then, $\#V(\langle P_p, Q_p\rangle) \leqslant \#V(\langle P, Q\rangle)$

Lemma 2

There exists at most $\widetilde{\Theta}(d^4 + d^3\tau)$ prime numbers s.t. $\#V(\langle P_p, Q_p\rangle) < \#V(\langle P, Q\rangle)$

Algorithm : Computing good prime

- For $\widetilde{\Theta}(d^4 + d^3\tau)$ p that do not cancel some leading coefficients
  - Compute $\#V(\langle P_p, Q_p\rangle) \rightsquigarrow \widetilde{O}_B(d^4)$
- Choose p that maximizes $\#V(\langle P_p, Q_p\rangle)$

Overall bit complexity : $\widetilde{O}_B(d^8 + d^7\tau)$

Triangular decomposition

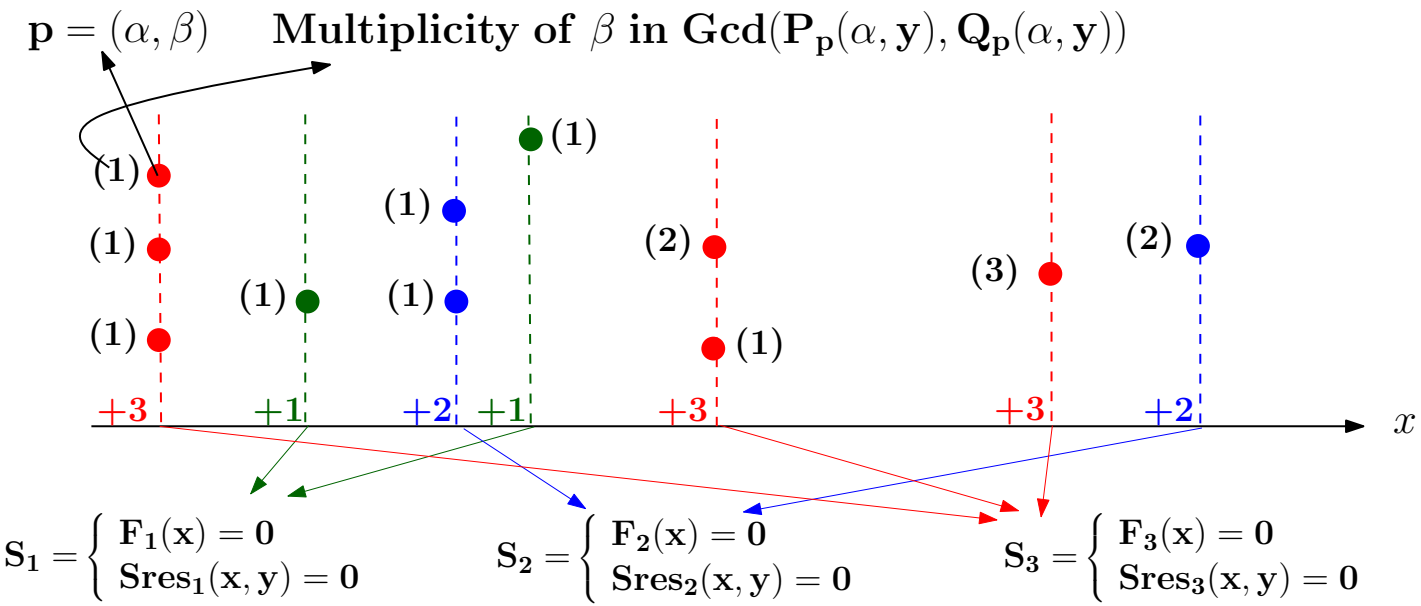
Triangular decomposition [Gonzalez-Vega and El Kahoui. 96]

- Input : $\{P, Q\}$, Output : a set of $S_i = \begin{cases} F_i(x) \\ Sres_i(x, y) \end{cases}$ such that :
  - $F_i(x)$ is a factor of $Res_y(P, Q)$
  - $\forall \alpha$ root of F_i : the deg of $\gcd(P(\alpha, y), Q(\alpha, y)) = Sres_i(\alpha, y)$ is i.

Triangular decomposition

Triangular decomposition [Gonzalez-Vega and El Kahoui. 96]

- Input : $\{P, Q\}$, Output : a set of $S_i = \begin{cases} F_i(x) \\ Sres_i(x, y) \end{cases}$ such that :
  - $F_i(x)$ is a factor of $Res_y(P, Q)$
  - $\forall \alpha$ root of F_i : the deg of $\gcd(P(\alpha, y), Q(\alpha, y)) = Sres_i(\alpha, y)$ is i.



Computing $\#V(\langle P_p, Q_p\rangle)$

$\mu(\alpha, \beta)$: the multiplicity of β in $\gcd(P_p(\alpha, y), Q_p(\alpha, y))$

- $\#V(\langle P_p, Q_p\rangle) = \sum_{(\alpha,\beta)\in V} \mu(\alpha, \beta) - \sum_{(\alpha,\beta)\in V} (\mu(\alpha, \beta) - 1)$

Algorithm : Computing $\#V(\langle P_p, Q_p\rangle)$

- Triangular decomposition of $\{P_p, Q_p\} \rightsquigarrow \{F_i(x), Sres_i(x, y)\}_{i\in\mathcal{I}}$

  $\rightsquigarrow \sum_{(\alpha,\beta)\in V} \mu(\alpha, \beta) = \sum_{i\in\mathcal{I}} i \times \deg(F_i)$

- Triangular decomposition of $\{Sres_i, \frac{\partial Sres_i}{\partial y}\} \rightsquigarrow \{F_{i,j}(x), Sres_{i,j}(x, y)\}_{j\in\mathcal{J}_i}$

  $\rightsquigarrow \sum_{(\alpha,\beta)\in V} (\mu(\alpha, \beta) - 1) = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}_i} j \times \deg(F_{i,j})$

Computing $\#V(\langle P_p, Q_p\rangle)$

$\mu(\alpha, \beta)$: the multiplicity of β in $\gcd(P_p(\alpha, y), Q_p(\alpha, y))$

- $\#V(\langle P_p, Q_p\rangle) = \sum_{(\alpha,\beta)\in V} \mu(\alpha, \beta) - \sum_{(\alpha,\beta)\in V} (\mu(\alpha, \beta) - 1)$

Algorithm : Computing $\#V(\langle P_p, Q_p\rangle)$

- Triangular decomposition of $\{P_p, Q_p\} \rightsquigarrow \{F_i(x), Sres_i(x, y)\}_{i\in\mathcal{I}}$

  $\rightsquigarrow \sum_{(\alpha,\beta)\in V} \mu(\alpha, \beta) = \sum_{i\in\mathcal{I}} i \times \deg(F_i)$

- Triangular decomposition of $\{Sres_i, \frac{\partial Sres_i}{\partial y}\} \rightsquigarrow \{F_{i,j}(x), Sres_{i,j}(x, y)\}_{j\in\mathcal{J}_i}$

  $\rightsquigarrow \sum_{(\alpha,\beta)\in V} (\mu(\alpha, \beta) - 1) = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}_i} j \times \deg(F_{i,j})$

Computing $\#V(\langle P_p, Q_p\rangle)$

$\mu(\alpha, \beta)$: the multiplicity of β in $\gcd(P_p(\alpha, y), Q_p(\alpha, y))$

- $\#V(\langle P_p, Q_p\rangle) = \sum_{(\alpha,\beta)\in V} \mu(\alpha, \beta) - \sum_{(\alpha,\beta)\in V} (\mu(\alpha, \beta) - 1)$

Algorithm : Computing $\#V(\langle P_p, Q_p\rangle)$

- Triangular decomposition of $\{P_p, Q_p\} \rightsquigarrow \{F_i(x), Sres_i(x, y)\}_{i\in\mathcal{I}}$

  $\rightsquigarrow \sum_{(\alpha,\beta)\in V} \mu(\alpha, \beta) = \sum_{i\in\mathcal{I}} i \times \deg(F_i)$

- Triangular decomposition of $\{Sres_i, \frac{\partial Sres_i}{\partial y}\} \rightsquigarrow \{F_{i,j}(x), Sres_{i,j}(x, y)\}_{j\in\mathcal{J}_i}$

  $\rightsquigarrow \sum_{(\alpha,\beta)\in V} (\mu(\alpha, \beta) - 1) = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}_i} j \times \deg(F_{i,j})$

Summary

The whole RUR computation algorithm

1. Compute a separating form $x + ay$, a in $\widetilde{O}(1) \rightsquigarrow \widetilde{O}_B(d^8 + d^7\tau)$
2. Compute the RUR associated to $x + ay \rightsquigarrow \widetilde{O}_B(d^7 + d^6\tau)$

Separating form computation is still the bottleneck in the worst case

Overview

1 Problem and motivation

2 Rational Univariate Representation

3 Theoretical worst-case complexity algorithm

4 Practical algorithm

5 Experiments

6 Conclusion and perspectives

Drawback of the previous approach

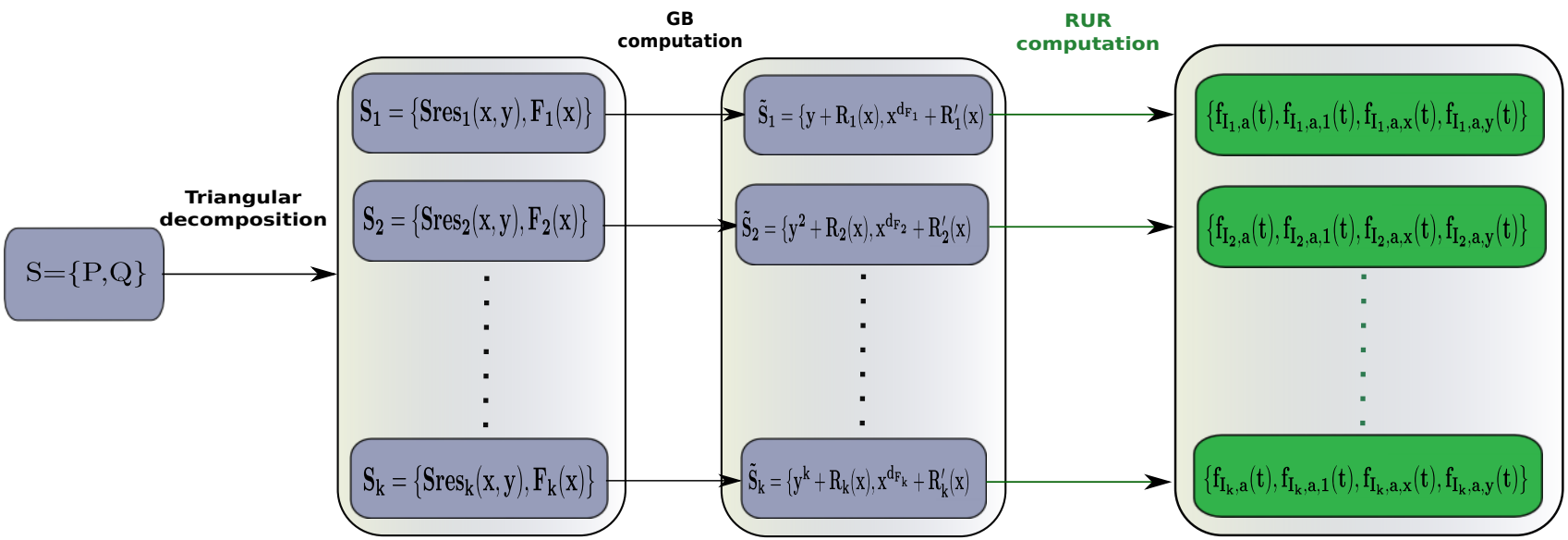
Is the previous approach efficient in practice ? No !

- Deterministic search for a separating form
- Resultant of polynomials in three variables

Our algorithm : A decomposition into several RURs

Outline

1. Random choice of a linear form
2. Triangular decomposition of the input system
3. Gröbner bases of the triangular sub-systems
4. Rational Univariate Representations



Our algorithm : details

Triangular decomposition of $\{P, Q\}$ [El kahoui and Gonzalez-Vega. 96]

- Output : A set of $S_k = \{F_k(x), Sres_k(x, y)\}$

The multiplicity of (α, β) in $\langle S_k \rangle$ is the multiplicity of β in $\gcd(P(\alpha, y), Q(\alpha, y))$

Lexicographic Gröbner Bases

Inverting the leading coefficient of $Sres_k(x, y)$ in S_k

Rational Univariate Representations

Arithmetic complexity improvement from $O(D^3)$ [Rouillier. 99], to $\widetilde{O}(D^2)$ where $D = O(d^2)$ is the number of solutions

An overall arithmetic complexity in $\widetilde{O}(d^4)$

Multi-modular algorithm

The coefficients in the Gröbner bases are much larger than those in the RURs : negative impact on efficiency

Chinese Remainder Theorem-based approach

1. Compute the RURs of $\{P_p, Q_p\}$ for a set of primes p *s.t.* $\prod_{p \in A} p \geqslant 2^m$ where m is a bound on the bitsize of the RURs of $\{P, Q\}$.
2. Apply the CRT to the resulting RURs in order to obtain the RURs of $\{P, Q\}$

Multi-modular algorithm

The coefficients in the Gröbner bases are much larger than those in the RURs : negative impact on efficiency

Chinese Remainder Theorem-based approach

1. Compute the RURs of $\{P_p, Q_p\}$ for a set of primes p *s.t.* $\prod_{p \in A} p \geqslant 2^m$ where m is a bound on the bitsize of the RURs of $\{P, Q\}$.
2. Apply the CRT to the resulting RURs in order to obtain the RURs of $\{P, Q\}$

Possibly wrong result

- Bad random linear form
- Bad prime numbers (unlucky)

Monte-Carlo algorithm

Monte-Carlo algorithm

1. Select randomly a linear form $x + ay$
2. Select a set of $2m$ primes, m is a bound on the bitsize of the RURs
3. Compute the RURs of $\{P, Q\}$ using the CRT approach

Monte-Carlo algorithm

Monte-Carlo algorithm

1. Select randomly a linear form $x + ay$
2. Select a set of $2m$ primes, m is a bound on the bitsize of the RURs
3. Compute the RURs of $\{P, Q\}$ using the CRT approach

Complexity and probability of success

Complexity :

$$\begin{array}{ccc} \text{Compute the RURs modulo one prime} & \times & \text{the number of primes} = 2m \\ \downarrow & & \downarrow \\ \widetilde{O}_B(d^4) & \times & \widetilde{O}(d^2 + d\tau) \end{array}$$

$$\rightsquigarrow \widetilde{O}_B(d^6 + d^5\tau)$$

Monte-Carlo algorithm

Monte-Carlo algorithm

1. Select randomly a linear form $x + ay$
2. Select a set of $2m$ primes, m is a bound on the bitsize of the RURs
3. Compute the RURs of $\{P, Q\}$ using the CRT approach

Complexity and probability of success

Complexity : $\rightsquigarrow \widetilde{O}_B(d^6 + d^5\tau)$

Probability of success :

Choose linear forms and prime numbers in sets *s.t.*

- The probability that the linear form is **separating** is larger than $\frac{1}{2}$
- The probability that m prime numbers are **lucky** is larger than $\frac{1}{2}$

The probability of success is larger than $\frac{1}{4}$.

Checking the result

Lemma

The set of the obtained solutions (counted with multiplicities) **cannot be** a strict subset of the set of the actual solutions.

Corollary

$$\text{Incorrect result} \Longrightarrow \begin{cases} \text{Some solution is not solution of } \{P, Q\} \\ \text{or} \\ \text{Some multiplicity is too large} \end{cases}$$

Checking the result

Lemma

The set of the obtained solutions (counted with multiplicities) **cannot be** a strict subset of the set of the actual solutions.

Corollary

Incorrect result $\Longrightarrow$ $\begin{cases} \text{Some solution is not solution of } \{P, Q\} \\ \text{or} \\ \text{Some multiplicity is too large} \end{cases}$

Sufficient condition for correcteness : $\forall\ (\alpha, \beta)$ of multiplicity μ, check that

- $P(\alpha, \beta) = Q(\alpha, \beta) = 0$
- $\frac{\partial^k P}{\partial^k y}(\alpha, \beta) = \frac{\partial^k Q}{\partial^k y}(\alpha, \beta) = 0$ for $k = 2, \ldots, \mu - 1$

Check the RURs by substitution $\rightsquigarrow \widetilde{O}_B(d^7 + d^6\tau)$

Las-Vegas algorithm

Las-Vegas algorithm

- For pairs of : linear form, set of prime numbers
  - Run the Monte-Carlo algorithm until the check is positive

- After one iteration, probability that the result is correct is larger than $\frac{1}{4}$

$\rightsquigarrow$ The Monte-Carlo algorithm runs at most four times on average before the check is positive

Expected bit complexity in $\widetilde{O}_B(d^7 + d^6\tau)$

Las-Vegas algorithm

Las-Vegas algorithm

- For pairs of : linear form, set of prime numbers
  - Run the Monte-Carlo algorithm until the check is positive

- After one iteration, probability that the result is correct is larger than $\frac{1}{4}$

$\rightsquigarrow$ The Monte-Carlo algorithm runs at most four times on average before the check is positive

Expected bit complexity in $\widetilde{O}_B(d^7 + d^6\tau)$

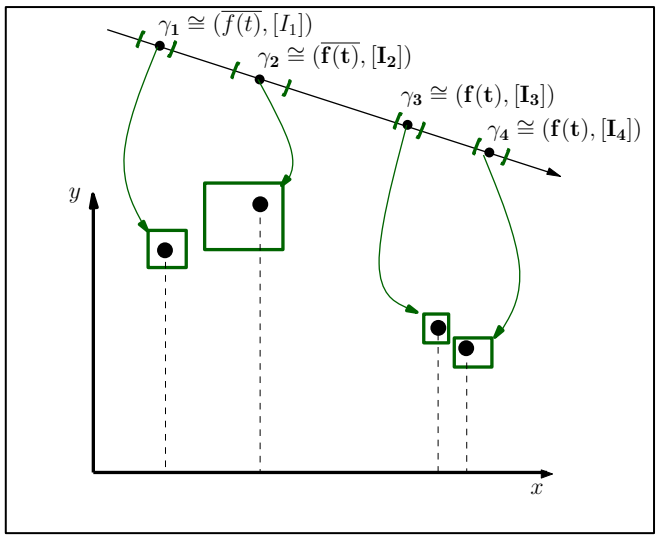
- Alternative Las-Vegas algorithm in $\widetilde{O}_B(d^6 + d^5\tau)$: Not implemented yet !

Certified numerical approximation

Goal : Disjoint boxes of the real solutions of $\langle P, Q \rangle$

Algorithm

- Isolate the real roots of $f(t) \rightsquigarrow$ intervals $I_1, \ldots, I_k$
- Compute the images of these intervals through $(\frac{f_x(t)}{f_1(t)}, \frac{f_y(t)}{f_1(t)}) \rightsquigarrow$ boxes $B_1, \ldots, B_k$
- Refine the intervals $I_1, \ldots, I_k$ until the boxes $B_1, \ldots, B_k$ are disjoint

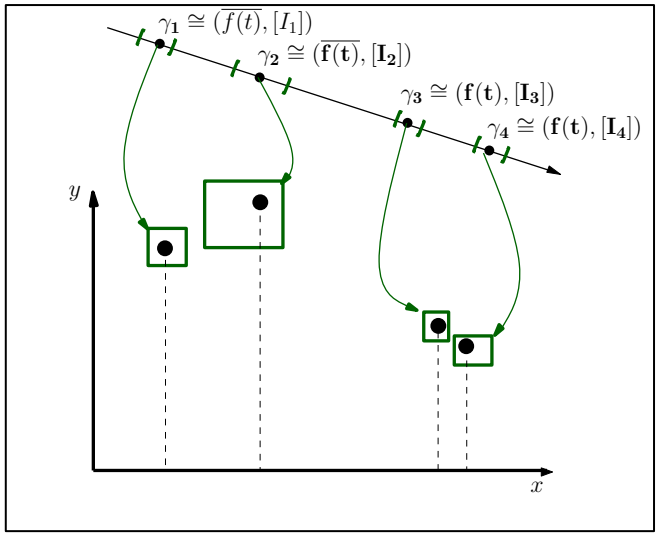


Certified numerical approximation

Goal : Disjoint boxes of the real solutions of $\langle P, Q \rangle$

Algorithm

- Isolate the real roots of $f(t) \rightsquigarrow$ intervals $I_1, \ldots, I_k$
- Compute the images of these intervals through $(\frac{f_x(t)}{f_1(t)}, \frac{f_y(t)}{f_1(t)}) \rightsquigarrow$ boxes $B_1, \ldots, B_k$
- Refine the intervals $I_1, \ldots, I_k$ until the boxes $B_1, \ldots, B_k$ are disjoint



Complexity

Sufficient condition for disjointness : Sum of the precisions in $\widetilde{O}(d^4 + d^3\tau)$

- Isolation and refinement : $\widetilde{O}_B(d^6 + d^5\tau)$ [Pan 01][Mehlhorn et al. 13]
- Evaluation : $\widetilde{O}_B(d^6 + d^5\tau)$ using amortized bounds

Overview

1 Problem and motivation

2 Rational Univariate Representation

3 Theoretical worst-case complexity algorithm

4 Practical algorithm

5 Experiments

6 Conclusion and perspectives

Some experiments

We compare our solver **RS3** with

- **IsolateRC**, the solver of the maple package *Regular Chains* [Li et al. 11]
- **LGP** [Cheng et al. 09]

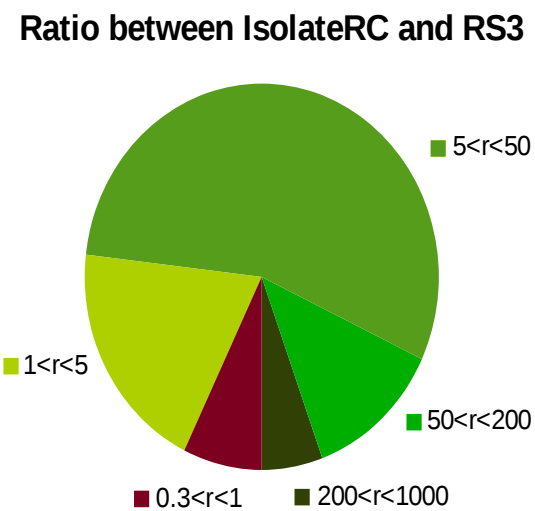
Input : $\{P, Q\}$ Output : Isolating boxes of the real solutions of $\{P, Q\}$

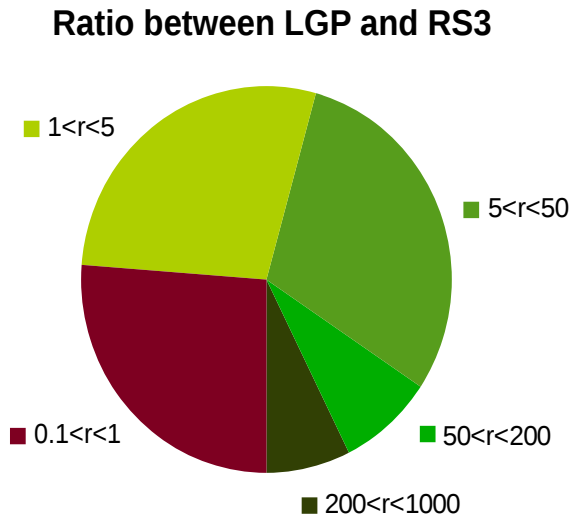
Some experiments

We compare our solver **RS3** with

- **IsolateRC**, the solver of the maple package *Regular Chains* [Li et al. 11]
- **LGP** [Cheng et al. 09]

Input : $\{P, Q\}$ Output : Isolating boxes of the real solutions of $\{P, Q\}$





Some experiments

We compare our solver **RS3** with

- **IsolateRC**, the solver of the maple package *Regular Chains* [Li et al. 11]
- **LGP** [Cheng et al. 09]

Input : $\{P, Q\}$ Output : Isolating boxes of the real solutions of $\{P, Q\}$

RS3 vs IsolateRC

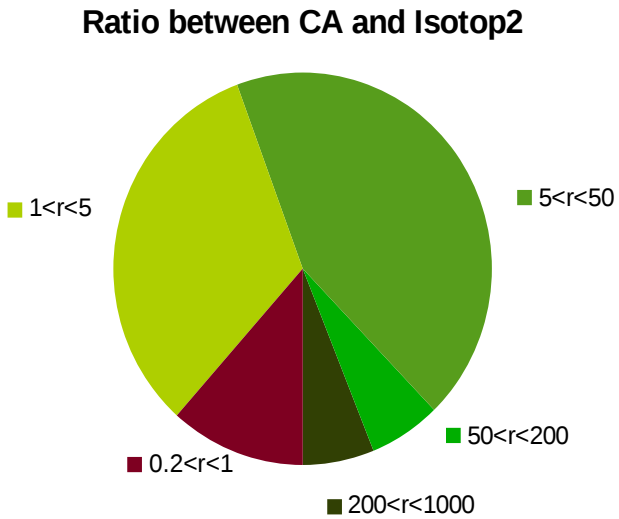
- Computing the RURs in **RS3** is comparable to the triangular decomposition part in **IsolateRC**
- The Isolation part is much faster in **RS3**

Conclusion : the overhead of the symbolic computation of the RURs is small compared to the benefit it yields for the isolation step !

Some experiments

We compare **ISOTOP2** with

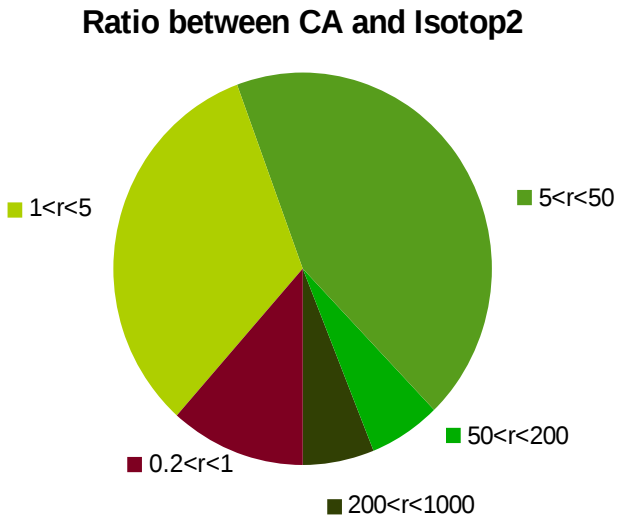
- **CA** of the arrangement package of *CGAL* [Eigenwillig et al. 07]



Some experiments

We compare **ISOTOP2** with

- **CA** of the arrangement package of *CGAL* [Eigenwillig et al. 07]

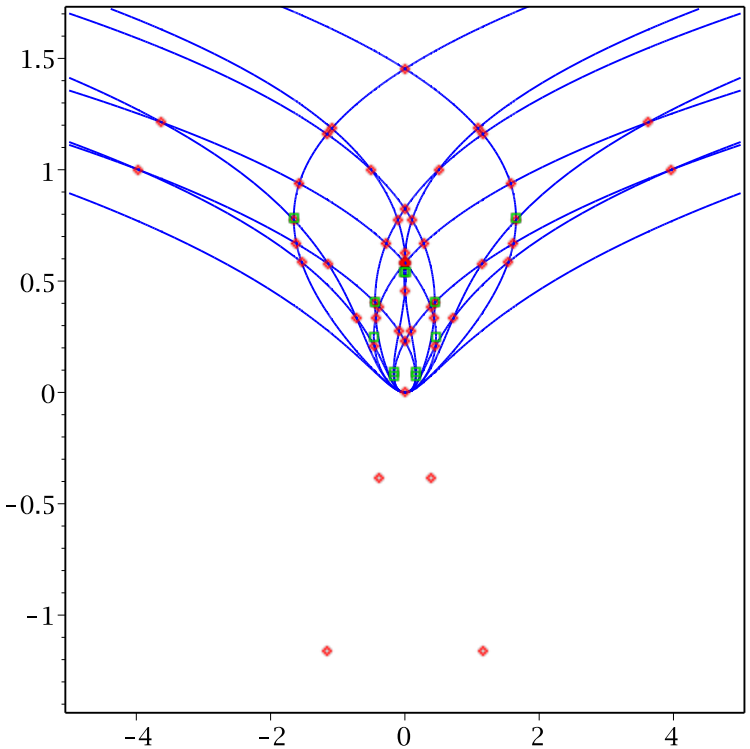


- **FastAnalysis** [Berberich et al. 11] (GPU resultant computation)
  - Comparable behavior with **FastAnalysis** with small advantage for **FastAnalysis**

The tables of benchmarks can be found in my thesis manuscript

Some examples

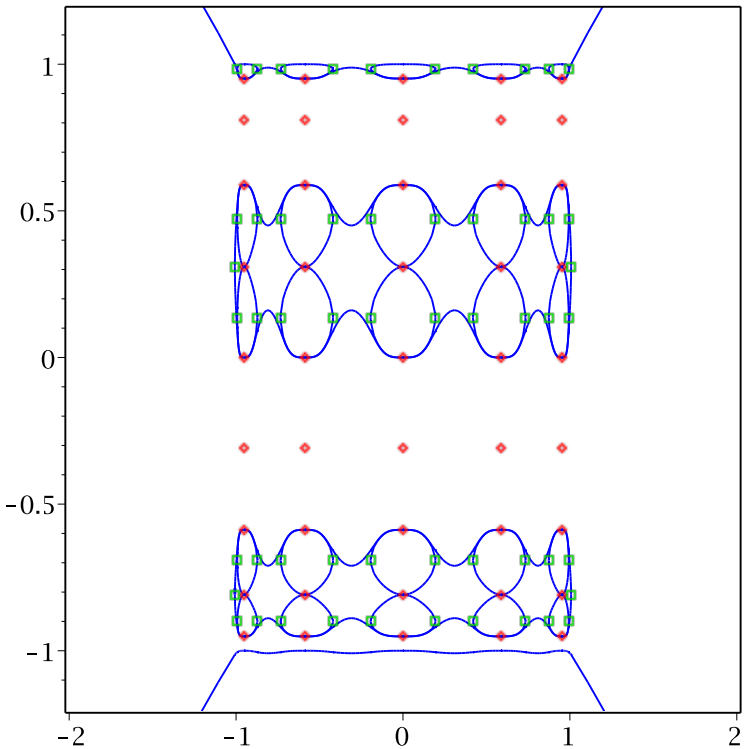
curve	d	τ	$\#V$	**Is (3T)**	FastAna	CA
swin	40	32	63	**6s**	7s	311
chal_12b	40	41	99	**49s**	44s	t/o
FTT	40	39	62	**49s**	32s	t/o
spider	36	248	38	**114s**	26s	t/o



The curves are taken from [Berberich et al. 11] and Timeout set to 30min

Some examples

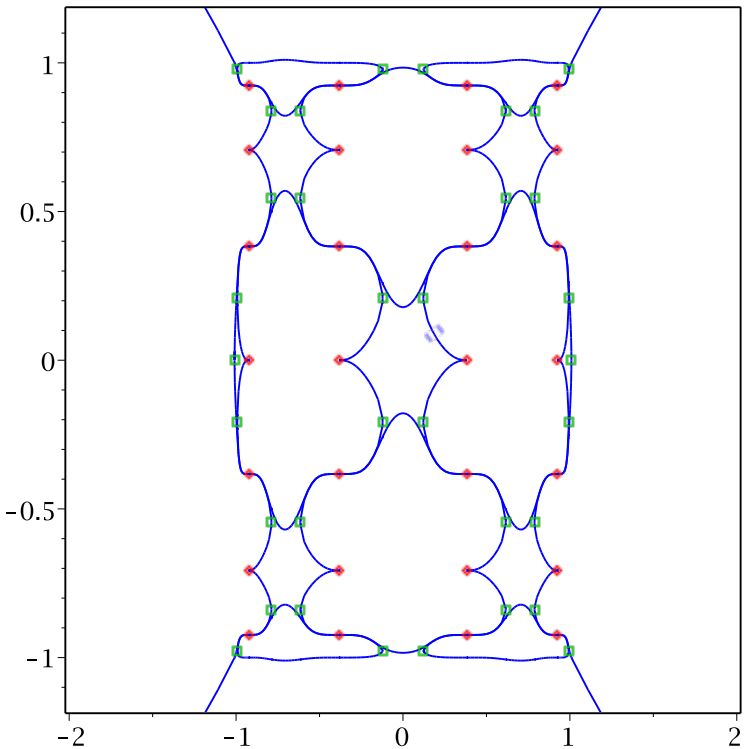
curve	d	τ	$\#V$	**Is (3T)**	FastAna	CA
swin	40	32	63	**6s**	7s	311
chal_12b	40	41	99	**49s**	44s	t/o
FTT	40	39	62	**49s**	32s	t/o
spider	36	248	38	**114s**	26s	t/o



The curves are taken from [Berberich et al. 11] and Timeout set to 30min

Some examples

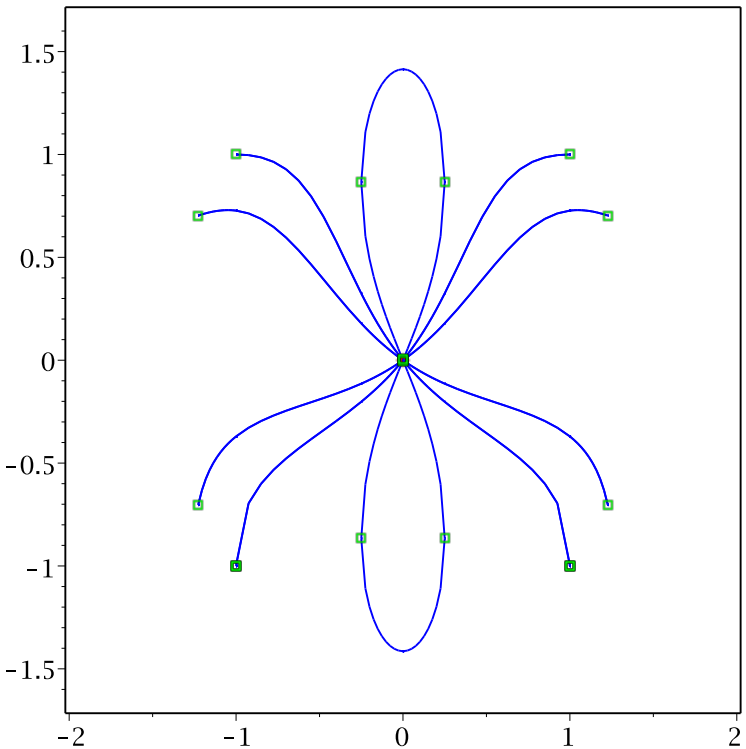
curve	d	τ	$\#V$	**Is (3T)**	FastAna	CA
swin	40	32	63	**6s**	7s	311
chal_12b	40	41	99	**49s**	44s	t/o
FTT	40	39	62	**49s**	32s	t/o
spider	36	248	38	**114s**	26s	t/o



The curves are taken from [Berberich et al. 11] and Timeout set to 30min

Some examples

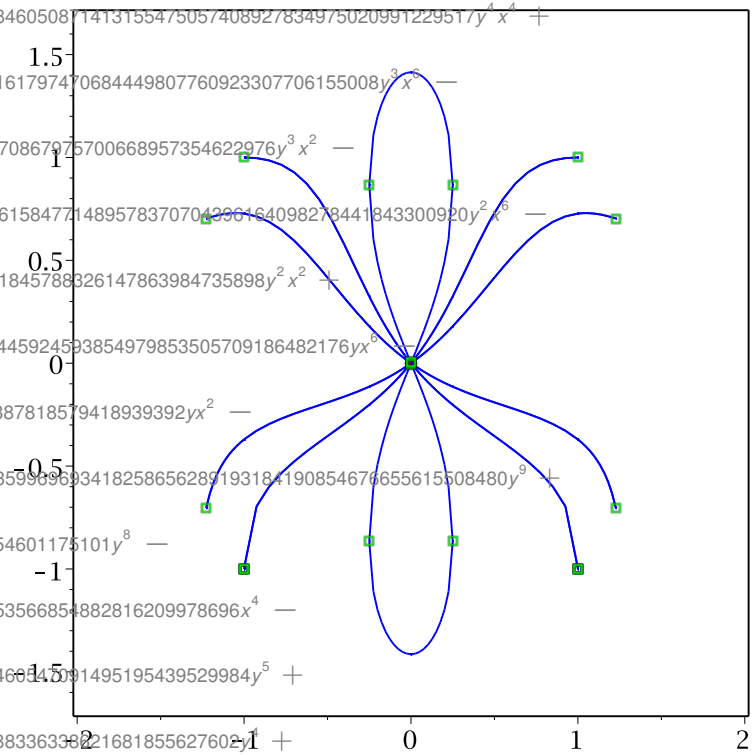
curve	d	τ	$\#V$	**ls (3T)**	FastAna	CA
swin	40	32	63	**6s**	7s	311
chal_12b	40	41	99	**49s**	44s	t/o
FTT	40	39	62	**49s**	32s	t/o
spider	36	248	38	**114s**	26s	t/o



The curves are taken from [Berberich et al. 11] and Timeout set to 30min

Some examples

curve	d	τ	$\#V$	Is (3T)	FastAna	CA
swin	40	32	63	**6s**	7s	311
chal_12b	40	41	99	**49s**	44s	t/o
FTT	40	39	62	**49s**	32s	t/o
spider	36	248	38	**114s**	26s	t/o



Overview

1 Problem and motivation

2 Rational Univariate Representation

3 Theoretical worst-case complexity algorithm

4 Practical algorithm

5 Experiments

6 Conclusion and perspectives

Summary of contributions

1. New algorithm for computing a separating form
   - Improves by a factor d^2 the best known complexity
2. New bound on the bitsize of the polynomials of the RUR
3. Efficient Las-Vegas algorithm for computing a decomposition into RURs

Summary of contributions

1. New algorithm for computing a separating form
2. New bound on the bitsize of the polynomials of the RUR
   - Same order than the squarefree of the resultant
3. Efficient Las-Vegas algorithm for computing a decomposition into RURs

Summary of contributions

1. New algorithm for computing a separating form
2. New bound on the bitsize of the polynomials of the RUR
3. Efficient Las-Vegas algorithm for computing a decomposition into RURs
   - Good complexity bounds (Matches the isolation of the resultant)
   - Excellent practical behavior

Summary of contributions

1. New algorithm for computing a separating form
2. New bound on the bitsize of the polynomials of the RUR
3. Efficient Las-Vegas algorithm for computing a decomposition into RURs

A whole bivariate solver in

- Worst-case complexity : $\widetilde{O}_B(d^8 + d^7\tau)$
  - Match the best known complexity [Emeliyanenko and sagraloff. 12]
- Expected complexity : $\widetilde{O}_B(d^6 + d^5\tau)$

Since December...

A new result

- A new algorithm for computing a separating linear form in $\widetilde{O}_B(d^7 + d^6\tau)$ (worst-case) and in $\widetilde{O}_B(d^5 + d^4\tau)$ (expected)
  - Idea : compute a separating form for the set of critical points of PQ which contains all the solutions of $\{P, Q\}$

Since December...

A new result

- A new algorithm for computing a separating linear form in $\widetilde{O}_B(d^7 + d^6\tau)$ (worst-case) and in $\widetilde{O}_B(d^5 + d^4\tau)$ (expected)
  - Idea : compute a separating form for the set of critical points of PQ which contains all the solutions of $\{P, Q\}$

Ongoing work :

- Separating form + rational parametrization in $\widetilde{O}_B(d^6 + d^5\tau)$ (worst-case) and $\widetilde{O}_B(d^5 + d^4\tau)$ (expected)

Since December...

A new result

- A new algorithm for computing a separating linear form in $\widetilde{O}_B(d^7 + d^6\tau)$ (worst-case) and in $\widetilde{O}_B(d^5 + d^4\tau)$ (expected)
  - Idea : compute a separating form for the set of critical points of PQ which contains all the solutions of $\{P, Q\}$

Ongoing work :

- Separating form + rational parametrization in $\widetilde{O}_B(d^6 + d^5\tau)$ (worst-case) and $\widetilde{O}_B(d^5 + d^4\tau)$ (expected)

Future work :

- Computing arrangement of curves
- Tackle the 3D world : for both solving and topology computation

Since December...

A new result

- A new algorithm for computing a separating linear form in $\widetilde{O}_B(d^7 + d^6\tau)$ (worst-case) and in $\widetilde{O}_B(d^5 + d^4\tau)$ (expected)
  - Idea : compute a separating form for the set of critical points of PQ which contains all the solutions of $\{P, Q\}$

Ongoing work :

- Separating form + rational parametrization in $\widetilde{O}_B(d^6 + d^5\tau)$ (worst-case) and $\widetilde{O}_B(d^5 + d^4\tau)$ (expected)

Future work :

- Computing arrangement of curves
- Tackle the 3D world : for both solving and topology computation

Thank you !