# Bivariate systems and topology of plane curves: algebraic and numerical methods

Marc Pouget

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

# Bivariate systems & topology of plane curves: algebraic and numerical methods

## THÈSE

soutenue le 24 juin 2022

pour l'obtention d'une

**Habilitation de l'Université de Lorraine**

**(mention informatique)**

par

Marc Pouget

**Composition du jury**

| | |
|---|---|
| *Rapporteurs :* | Évelyne Hubert, Directrice de Recherche INRIA, Sophia Antipolis<br>Sonia Pérez-Díaz, Professeure à l'Université d'Alcalá, Espagne<br>Éric Schost, Professeur à l'Université de Waterloo, Canada |
| *Examinateurs :* | Damien Chablat, Directeur de Recherche CNRS, Université de Nantes, Président du jury<br>Nicolas Delanoue, Maître de conférence, HDR, Université d'Angers<br>Marine Minier, Professeure à l'Université de Lorraine, Marraine scientifique |

**Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503**

Mis en page avec la classe thesul.

# Résumé

Le travail présenté dans cette thèse appartient au domaine de la géométrie computationnelle non linéaire en petite dimension. Plus précisément, il se concentre sur la résolution de systèmes bivariés et le calcul de la topologie des courbes dans le plan. Lorsque l'entrée est donnée par des polynômes, les outils naturels proviennent du calcul formel. Nos contributions sont des algorithmes dont l'efficacité a été prouvée dans un cadre déterministe ou Las Vegas, ainsi qu'un logiciel efficace pour le dessin certifié de la topologie d'une courbe algébrique plane. Lorsque les données d'entrée ne sont pas limitées aux polynômes mais sont données par des fonctions d'intervalles, nous concevons des algorithmes basés sur des approches numériques certifiées utilisant la subdivision et l'arithmétique d'intervalles. L'entrée doit alors satisfaire certaines hypothèses génériques et nos algorithmes sont certifiés dans le sens où ils se terminent si et seulement si les hypothèses sont satisfaites.



# Abstract

The work presented in this thesis belongs to the domain of non-linear computational geometry in low dimension. More precisely it focuses on solving bivariate systems and computing the topology of curves in the plane. When the input is given by polynomials, the natural tools come from computer algebra. Our contributions are algorithms proven efficient in a deterministic or a Las Vegas settings together with a practical efficient software for topology certified drawing of a plane algebraic curve. When the input is not restricted to be polynomials but given by interval functions, we design algorithms based on certified numerical approches using subdivision and interval arithmetic. The input is then required to fulfill some generic assumptions and our algorithms are certified in the sense that they terminate if and only if the assumptions are satisfied.

# Table of contents

# Chapter 1

# Introduction

## 1.1 Context and theme of research

Curved objects are ubiquitous in the world we live in. Yet, in spite of decades of research in several communities, curved objects are far from being robustly and efficiently manipulated by geometric algorithms. On way is to discretize a curved object and work on a piecewise linear representation since algorithms on linear objects are often simpler and efficient. Still such a discretization may not represent faithfully the topology or the geometry of the original object or may require a large number of linear elements. Dramatic improvements can be accomplished when the right mathematics and computer science are put into motion, as shown for instance by my work on certified drawing of plane curves. In this direction, many problems are of fundamental nature and solutions have potential industrial impact in Computer Aided Design and Robotics for instance. Intersecting parametric surfaces or meshing singular curves and surfaces in a certified manner are important examples of such problems.

Among the main characteristics of my research are: the belief that mathematics (topology, geometry, algebra or numerical analysis) and computer science (computer algebra, interval analysis, geometric algorithms), when cleverly set into motion, can dramatically enhance the algorithmic knowledge concerning curved objects; the understanding that practical efficiency is as important as theoretical complexity in algorithmic design; and the willingness to cover the whole spectrum of exact geometric computing going from a proven characterization of geometric degeneracies to the production of reliable software.

More precisely, I'm interested in computational geometry with an algebraic flavour, also termed non-linear computational geometry. In particular, I'm investigating symbolic (computer algebra) and certified numerical methods for the computation of the topology and the geometric approximation of curves and surfaces. For symbolic methods, this implies the use of exact computation combined with multimodular techniques for efficiency. By certified numerical algorithms, I mean algorithms that are proven correct even in the presence of rounding error. In particular, the theory of transversality and interval arithmetic are our basic building blocks to achieve certification.

**Outline of the thesis.** Chapter 2 details my algorithms for solving bivariate polynomial systems with algebraic methods. Chapter 3 focuses on Isotop my software for computing the topology of a real algebraic plane curve. Chapter 4 details my approach theoretically based on transversality theory and algorithmically based on subdivision for computing the topology of a curve defined via interval functions. The following sections of this chapter introduce the contributions of each of these chapters. My research project is presented in Chapter 5 and my publications are listed in Chapter 6.

## 1.2 Solving bivariate algebraic systems

The collaboration I started with the computer algebra community at the end of my PhD opened my mind to the field of effective algebraic geometry. This field was already well established with algorithmic tools and complexity analysis in a general setting for polynomial systems of arbitrary dimensions. Nonetheless, I realized that there was room for research in this field, so that I started a thematic move to investigate computer algebra tools applied to computational geometry with the following ideas in mind:

- Algorithms can be improved when restricted to objects of low dimensions, typically curves and surfaces.

- The complexity has to be refined to take into account the bitsize of the input and intermediate results, this is called the bit complexity hereafter.

- Keeping in mind the certification goal, probabilistic algorithms in the Monte Carlo sense are not satisfactory, one should aim for Las Vegas algorithms: the running time is a random variable (with a well studied expectation) but the result is always correct.

- In practice, the input may not be in a required "generic position" and specific analysis has to be conducted to detect a degenerate position and to deal with such input.

A key computer algebra sub-problem for the curve topology computation is the isolation of singular points, which amounts to solving bivariate algebraic systems (the equations of the curve and its partial derivatives). Given a system of bivariate polynomials, the aim is to compute a convenient representation of its solutions to enable further processing needed in geometric algorithms: comparison of solutions, evaluation of another polynomial at a solution and eventually approximating the location of the solutions in $\mathbb{R}^2$. I focused on the problem of computing a *Rational Univariate Representation (RUR)* of the solutions, that is, roughly speaking, a univariate polynomial and two rational functions that map the roots of the univariate polynomial to the two coordinates of the solutions of the system. I also addressed the problem of computing a *linear separating form* of a system of two bivariate polynomials with integer coefficients, that is a linear combination of the variables that takes different values when evaluated at the distinct solutions of the system. The computation of such linear forms is at the core of most algorithms that solve algebraic systems by computing rational parameterizations of the solutions and, surprisingly, it was the bottleneck of these algorithms in terms of worst-case bit complexity. Finally, I also worked on the problem of computing *isolating boxes* of the solutions from the RURs.

**Discussion on contributions.** Let $P$ and $Q$ be two coprime polynomials in $\mathbb{Z}[x, y]$ of degree bounded by $d$ and bitsize bounded by $\tau$. In the following, the notation $\widetilde{O}$ refers to the complexity where poly-logarithmic factors are omitted and $O_B$ refers to the bit complexity. I present three algorithms, one for each of the main steps of solving a bivariate system $\{P, Q\}$ via RURs, that is, computing (i) a separating linear form $x + ay$ with $a \in \{0, \ldots, 2d^4\}$, (ii) a RUR decomposition of the system, and (iii) isolating boxes of the solutions. Each of these algorithms has worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ and I also present Las Vegas variants of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ of my two first algorithms. I do not present a Las Vegas variant of my last algorithm for computing isolating boxes but it should be noticed that the complexity of that subdivision-based algorithm actually depends on the distances between the solutions and thus its worst-case complexity is not always reached.

These complexities are not proved to be optimal but it should be stressed that, even in the simpler setting where the ideal is known to be radical, the complexities for problem (i) match the best known worst-case and expected bit complexities for checking whether a linear form is separating.[1] This also holds for problem (ii) since computing a RUR decomposition *requires* computing a separating form. Similarly, for problem (iii), observe that my bound also matches the best known complexity for the isolation of the roots of a given polynomial of degree and bitsize comparable to those of the resultant of the input polynomials. All these observations hint that it is likely difficult to improve the Las Vegas and worst-case complexities of my sequence of algorithms for solving bivariate systems.

In addition, when one only requires the result to be certified with some probability, Monte Carlo algorithms come into the picture. Doing so, Mehrabi and Schost [MS16] compute a RUR of the radical of the input system with probability of success at least $1 - 1/2^p$ in a bit complexity that is the sum of $\widetilde{O}_B(d^{2+\varepsilon}(d^2 + d\tau + dp + p^2))$, for $\varepsilon > 0$ arbitrarily small, and of the complexity of computing a random prime smaller than $M = (2^p d\tau)^{O(1)}$. This result is remarkable since the whole complexity is $\widetilde{O}_B(d^{4+\varepsilon} + d^{3+\varepsilon}\tau)$ when $p$ is a constant, which almost matches the worst-case upper and lower bounds $\widetilde{O}(d^4 + d^3\tau)$ and $\widetilde{\Omega}(d^4)$ on the size of the output. However the drawback of this approach is that, so far, neither the separating form nor the computed RUR can be checked for correctness with a better bit complexity than those I present, that is $\widetilde{O}_B(d^5 + d^4\tau)$ expected and $\widetilde{O}_B(d^6 + d^5\tau)$ in the worst case.

[1] In this simpler case, the problem amounts to shearing the coordinate system, then computing a resultant and checking whether it is squarefree. The best known complexities for that problem are $\widetilde{O}_B(d^5 + d^4\tau)$ in the Las Vegas setting and $\widetilde{O}_B(d^6 + d^5\tau)$ in the worst case. Indeed, after shearing, the input polynomials have degree $d$ and bitsize $\widetilde{O}(d + \tau)$. The best known complexity for computing their resultant is $\widetilde{O}_B(d^4(d + \tau))$ in the worst case and, up to our knowledge, there is no better complexity even in the Las Vegas setting. Finally, the squarefreeness test amounts to a gcd computation and the best-known complexities are $\widetilde{O}_B(d^4 + d^3\tau)$ in the Las Vegas setting and $\widetilde{O}_B(d^6 + d^5\tau)$ in the worst case.

These complexity improvements have been made possible by the use of multimodular techniques. This required a precise study of the bitsize of the computed objects: RURs and triangular systems, together with the conditions to choose prime numbers to compute in associated prime field and ensure a correct reconstruction. In particular, I present an amortized analysis of the classical triangular decomposition via subresultants of bivariate systems [GVEK96], proving that the decomposition can be computed in $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case, which improves by a factor $d$ the state-of-the-art analysis [DET09, Proof of Theorem 19]. The multimodular techniques are much relevant both in theory for the correctness and the complexity, and in practice since they allow parallel computations.

Finally, one notes that, for computing a separating linear form of an *arbitrary* system $\{P, Q\}$, my best complexity algorithm is likely to be purely theoretical because (i) considering the system $\{PQ, \frac{\partial PQ}{\partial y}\}$ instead of $\{P, Q\}$ essentially doubles the degree of the input polynomials, and (ii) the shearing of the coordinate systems, done to avoid vertical asymptotes, spoils the sparsity of the coefficients and increases their bitsize, which is not efficient in practice. To solve these drawbacks in practice, I first show how the triangular decomposition can be performed even in the presence of asymptotes without shearing and without impacting the complexity (Section 2.3.1). Second, I show how to further split the decomposition into its generic and non-generic parts, so that to avoid shearing the input system but only sub-systems when needed. I also show that the multiplicity information required by my application to curve drawing can be preserved by this practical method (Section 2.3.2).

**Advised PhD and collaborations.** Among my co-authors are Yacine Bouzidi, a PhD I advised with Sylvain Lazard (Inria Nancy); Jean-Charles Faugère (Inria Paris), Fabrice Rouillier (Inria Paris), Guillaume Moroz (Inria Nancy) and Mickael Sagraloff (MPII Saarbrücken) contributed their expertise in computer algebra.

**Outline of Chapter 2.** Chapter 2 introduces my algorithms for solving bivariate polynomial systems together with their complexities. The best algorithms in terms of asymptotic complexity are not always the best in practice and several options had to be tested on an extensive benchmark of thousands of curves. These tests were used to fine tune both the ISOTOP and the RS3 software (RS3 being the algebraic solver used by ISOTOP). I detail these practical choices in Section 2.3. Classical notation and definitions are recalled in Section 2.1.

The publications associated with this chapter are [BLPR13b*, BLPR13c*, BLM+14*, BLPR15*, BLM+16*, LPR17*].

## 1.3 ISOTOP algorithm: topology of algebraic curves

At the end of my Ph.D., focussed on discrete differential geometry on surfaces defined by meshes or point clouds, I began considering parametric surfaces defined by polynomial functions that are commonly used in computer aided design. I thus changed my point of view from differential geometry and approximation theory to computational algebraic geometry and computer algebra. My contribution was to study ridge curves, i.e. curves of extremal principal curvatures, in this new setting. I proved that ridge curves can be encoded by an implicit polynomial curve in the parametric domain and I provided semi-algebraic tests to distinguish further subclasses of ridge curves [CFPR06*]. I then started a collaboration with F. Rouillier and J-C Faugère from the Salsa Inria Rocquencourt team to design an algorithm certifying the topology of ridge curves. Due to the high degree of ridge curves in terms of the degree of the input parameterization of the surface, specific geometric knowledge had to be taken into account to leverage the available tools of computer algebra [CFPR08*].

I have been working, together with several members of my Inria team and F. Rouillier from the Ouragan (formerly Salsa) Inria team, on the problem of drawing plane algebraic curves with a certified topology. Precisely, given the implicit equation of a plane algebraic curve, one wants to compute a polygonal approximation of the curve with guaranteed topology. This problem is important as it allows plotting algebraic curves in a certified way, that is, in particular, without missing branches or self-intersections. For example, a certified plot of a discriminant curve could be the only admissible answer that can be proposed for engineering problems that need the resolution of parametric algebraic systems: this curve (and the connected components of its counterpart) defines a partition of the parameter space in regions in which the solutions are numerically stable and topologically simple, see for instance references in [MZ19] from the robotics community. Standard software tools do not provide such certified plots, see e.g. Figure 1.1. Several existing mathematical software packages: Maple, Mathematica, Matlab, etc.,

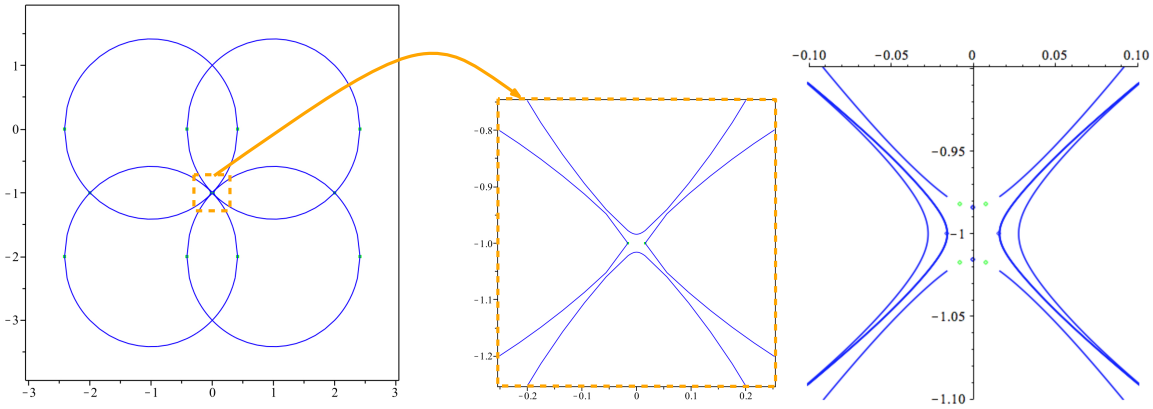



Figure 1.1: Left: a certified drawing output by Isotop of a perturbation of a "product" of 4 circles of equation $[(x+1)^2+y^2-2]\,[(x-1)^2+y^2-2]\,[(x+1)^2+(y+2)^2-2]\,[(x-1)^2+(y+2)^2-2]\,-10^{-6}=0$. Center: a zoom with the correct topology. Right: a zoom computed by the Maple command `plot_real_curve` with an incorrect topology.

enable an approximate visualization of curves. However, they provide no certification guarantee regarding the correctness of the visualization. These systems produce visual representations which may suffer from important artifacts due to an improper sampling of the curve. As a result, the topological characteristics of the visual representations can be arbitrarily far from those of the actual curve (different numbers of connected components, intersections, etc.). My Isotop software uses a radically different approach by first computing the singularities and at least one point per connected component. The visualization can then be computed at any level of detail by connecting these feature points without artifacts. As opposed to non-certified visualizations, Isotop outputs a drawing witnessing all components of the curve with a precision adapted to distinguish all singularities.

My contribution began with a new geometric algorithm (a) exploiting Teissier formula (Lemma 38) to compute multiplicities in fibers, these multiplicities are critical for the isolation of the singular points of the curve; (b) handling degeneracies, such as asymptotes or multiple singular points in the same fiber, without using a change of coordinates [CLP+08*, CLP+09*, CLP+10*]. It is worth noting that changing the coordinates does not change the topology of the curve and previous work generally used such an approach since it simplifies a lot the algorithmic but has a dramatic impact on the practical complexity. In addition, it solves a slightly different problem which is irrelevant for most applications: the coordinates are parameters of a specific problem and it makes no sense to mix them.

I spent a significative amount of time in 2008-9 implementing the Isotop software and I have been continuously improving and maintaining it since then. In particular, I worked with Benjamin Dexheimer, an Inria engineer, on developing an interactive visualization and a new webserver.[2] Isotop motivated the development of a dedicated algebraic solver with the ideas presented in Section 2.3 and conversely the design of Isotop followed the evolution of the solver. On the other hand, F. Rouillier (Ouragan Inria team, Paris) is the only developer of this algebraic solver Real Solving RS3. The experiments performed on an extensive database of curves show that Isotop outperforms other software for most classes of examples [BLPR15*].

**Advised PhD and collaborations.** On this topic, I co-advised the PhD of Luis Peñaranda with Sylvain Lazard. I also collaborated with two postdocs, Jinsan Cheng and Elias Tsigaridas, on the first version of our algorithm and software.

**Outline of Chapter 3.** Chapter 3 introduces the state of the art for the computation of the topology of an algebraic curve and details the contributions of Isotop.
The publications associated with this chapter are [CLP+08*, CLP+09*, CLP+10*, BLPR11*, LPR17*].

[2] https://isotop.gamble.loria.fr/

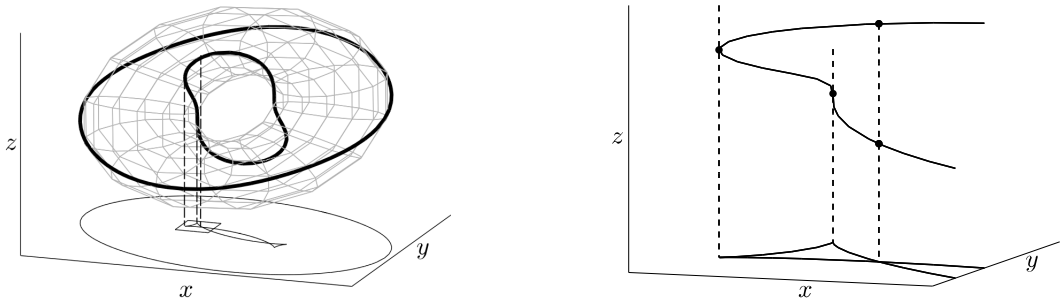



Figure 1.2: Left: a torus $P = 0$, in bold line the curve $P = P_z = 0$, its apparent contour, and a zoom zone. Right: a zoom, with preimages by the projection of cusp and node singularities.

## 1.4 Reliable numeric algorithms for the topology of plane projections of smooth curves

In parallel to the above algebraic approach for plane curves, I have investigated certified numerical approaches. The usual advantages of numerical methods are their efficiency and locality: one can focus on a small region and the complexity is expected to be correlated to the geometric complexity of the curve only in that region. This is in contrast to algebraic approaches whose complexities depend on global parameters such as the degree and the bitsize of the coefficients. The main drawback of numerical approaches, like tracking or subdivision, is that they are not certified in general except for non-singular curves [BT06]. I have therefore been focusing on the design of certified numerical methods for some classes of singular curves.

One motivation comes from mechanical design applications, where the space of configurations of a robot (i.e. the values of the articular parameters together with the corresponding position of the end effector of the robot) is encoded by a system of equations and projections on subspaces of parameters are to be studied. The application goal is thus to extend the simulation environment of researchers in robotics with innovative tools, offering them better assistance in the design of new mechanisms. I first studied singular curves that are projections of higher dimensional smooth ones that commonly appear in the analysis of two degrees of freedom robots.

In [IMP16*, IMP15a*, IMP18*], I proposed a solution for the projection in the plane of generic smooth curves in 3D. Such projections are generically singular but the singularities are only nodes or ordinary cusps, see Figure 1.2. The key point was to show that the singularities can be described as the regular[3] solutions of square systems since this is a requirement to isolate them with certified numerical tools such as the interval Newton method. In [IMP16*], my contribution was in the transition from symbolic to numeric: I was still using subresultants to design a square and regular system in $\mathbb{R}^2$. In [IMP15a*], I removed this symbolic projection step and went in the other direction by lifting the singularities in $\mathbb{R}^4$. In [IMP18*], I proposed data structure answering point location queries with respect to the subdivision of the plane induced by the projected curve. This data structure is composed of an approximation of the space curve together with a topological representation of its projection. In [KLMP21*], I extended the method to the case of a curve in arbitrary codimension. The notion of transversality from singularity analysis is used to prove the genericity of the conditions. These conditions are then checked via a semi-algorithm in the sense that the semi-algorithm terminates if and only if the conditions are satisfied.

This work is the first certified numerical approach for computing the topology of a class of singular curves and it handles in practice examples that are not reachable by algebraic methods (Section 4.6). The associated software is still a prototype, but I distribute one of its building blocks, SubdivisionSolver, which is a very efficient certified multi-precision numerical solver for regular square systems of polynomial equations.

**Advised PhD and collaborations.** This research was done in collaboration with Guillaume Moroz with whom I coordinated the ANR JCJC SingCAST. I co-advised the PhD of George Krait with Sylvain Lazard and Guillaume Moroz. I also collaborated with a postdoc, Rémi Imbach, and an engineer, Mohamed Eissa.

[3] A solution to a system is regular when the Jacobian determinant does not vanish.

**Outline of Chapter 4.** Chapter 4 details my certified numerical approach for computing the topology the projection of a smooth curve. Section 4.2 presents the generic assumptions under which the singularities of the projection can be encoded by a square and regular system (Section 4.3). Section 4.4 presents the semi-algorithms to check the assumptions and isolate the singularities. A data structure enabling location queries is proposed to encode the projected curve (in Section 4.5). I present experiments on a high degree curve in $\mathbb{R}^4$ and on a classical robotic mechanism in Section 4.6.

The publications associated with this chapter are [IMP16*, IMP15a*, IMP18*, KLMP21*].

# Chapter 2

# Solving bivariate algebraic systems

Section 2.2 presents my algorithms for solving zero-dimensional systems via rational univariate representations: computing triangular decompositions (Section 2.2.2), separating forms (Section 2.2.3), RUR decompositions (Section 2.2.4) and isolating the solutions (Section 2.2.5). The emphasis is put on best complexities either in the worst-case or in expectation for Las-Vegas algorithms. The results mainly come from [BLPR15*, BLM+16*]. Since the best algorithms in theory may not be the best in practice, Section 2.3 presents the practical versions that I developed [BLPR11*, LPR17*], in particular for my Isotop software computing the topology of an algebraic plane curve. Section 2.1 recalls classical notation and definitions used in the next two chapters.

On this topic, I co-advised the PhD of Yacine Bouzidi with Sylvain Lazard. I also collaborated with Guillaume Moroz and Fabrice Rouillier.

Given two coprime polynomials $P$ and $Q$ in $\mathbb{Z}[x,y]$ of degree bounded by $d$ and bitsize bounded by $\tau$, we address the problem of solving the system $\{P,Q\}$. We are interested in certified numerical approximations or, more precisely, isolating boxes of the solutions. We are also interested in computing, as intermediate symbolic objects, rational parameterizations of the solutions, and in particular Rational Univariate Representations (RURs), which can easily turn many queries on the system into queries on univariate polynomials. Such representations require the computation of a separating form for the system, that is a linear combination of the variables that takes different values when evaluated at the distinct solutions of the system.

We present new algorithms for computing linear separating forms, RUR decompositions and isolating boxes of the solutions. We show that these three algorithms have worst-case bit complexity $\widetilde{O}_B(d^6+d^5\tau)$, where $\widetilde{O}$ refers to the complexity where polylogarithmic factors are omitted and $O_B$ refers to the bit complexity. We also present probabilistic Las-Vegas variants of our two first algorithms, which have expected bit complecity $\widetilde{O}_B(d^5+d^4\tau)$. A key ingredient of our proofs of complexity is an amortized analysis of the triangular decomposition algorithm via subresultants, which is of independent interest. To apply a multimodular scheme to the RUR decomposition, we derive a tight bound on its bitsize.

Finally, focusing on a practical and efficient solver tailored for our Isotop software, we depart from the presented best complexity algorithms. In particular, we avoid a global shearing of the coordinate system from the beginning and instead first decompose further the triangular systems in their generic and non-generic parts. Independently, to handle the degenerate case of vertical asymptotes of a curve, we extend the triangular decomposition in a recursive way without increasing its complexity.

## 2.1 Notation and definitions

For convenience, we recall notation and some classical material about subresultants, gcds, lucky primes for gcd computations and multiplicities. This section can thus be skipped for a first reading and we invite the reader to come back to it when needed later.

The bitsize of an integer $p$ is the number of bits needed to represent it, that is $\lfloor \log p \rfloor + 1$ (log refers to the logarithm in base 2). The bitsize of a rational is the maximum bitsize of its numerator and its denominator. The bitsize of a polynomial with integer coefficients is the *maximum* bitsize of its coefficients. As mentioned earlier, $O_B$ refers to the bit complexity and $\widetilde{O}$ and $\widetilde{O}_B$ refer to complexities where polylogarithmic factors are omitted, i.e., $f(n) \in \widetilde{O}(g(n))$ if $f(n) \in O(g(n)\log^k(n))$ for some $k \in \mathbb{N}$.

In the following, $\mu$ is a prime number and we denote by $\mathbb{F}_\mu$ the quotient $\mathbb{Z}/\mu\mathbb{Z}$. We denote by $\phi_\mu$: $\mathbb{Z} \to \mathbb{F}_\mu$ the reduction modulo $\mu$, and extend this definition to the reduction of polynomials with integer coefficients. We denote by $\mathbb{D}$ a unique factorization domain, typically $\mathbb{Z}[x,y]$, $\mathbb{Z}[x]$, $\mathbb{F}_\mu[x]$, $\mathbb{Z}$ or $\mathbb{F}_\mu$. We also denote by $\mathbb{F}$ a field, typically $\mathbb{Q}$, $\mathbb{C}$, or $\mathbb{F}_\mu$.

For any polynomial $P$ in $\mathbb{D}[x]$, let $\mathrm{Lc}_x(P)$ denote its leading coefficient with respect to the variable $x$ and $\mathrm{d}_x(P)$ its degree with respect to $x$. The degree of a polynomial refers to its *total* degree, unless specified otherwise. For simplicity, we often refer to a curve of equation $H(x,y)=0$ as to curve $H$ and we also refer to a system $\{P=0, Q=0\}$ as to system $\{P,Q\}$. For any curve defined by $H(x,y)$ in $\mathbb{D}[x,y]$, we call the critical points of $H$ with respect to $x$ or more shortly the critical points of $H$, the points that are solutions of the system $\{H, \frac{\partial H}{\partial y}\}$. The solutions of a system of polynomials are always considered in the algebraic closure of the fraction field of $\mathbb{D}$.

### 2.1.1 Subresultant and gcd

We first recall the concept of *polynomial determinant* of a matrix which is used in the definition of subresultants. Let $M$ be an $m \times n$ matrix with $m \leqslant n$ and $M_i$ be the square submatrix of $M$ consisting of the first $m-1$ columns and the $i$-th column of $M$, for $i = m, \ldots, n$. The *polynomial determinant* of $M$ is the polynomial defined as $\det(M_m)y^{n-m} + \det(M_{m+1})y^{n-(m+1)} + \cdots + \det(M_n)$.

Let $P = \sum_{i=0}^{p} a_i y^i$ and $Q = \sum_{i=0}^{q} b_i y^i$ be two polynomials in $\mathbb{D}[y]$ and assume, without loss of generality, that $a_p b_q \neq 0$ and $p \geqslant q$.

The Sylvester matrix of $P$ and $Q$, $\mathrm{Syl}_y(P,Q)$ is the $(p+q)$-square matrix whose rows are $y^{q-1}P, \ldots, P$, $y^{p-1}Q, \ldots, Q$ considered as vectors in the basis $y^{p+q-1}, \ldots, y, 1$.

$$\mathrm{Syl}_y(P,Q) = \overbrace{\begin{pmatrix} a_p & a_{p-1} & \cdots & \cdots & a_0 & & & \\ & a_p & a_{p-1} & \cdots & \cdots & a_0 & & \\ & & \ddots & & & & \ddots & \\ & & & a_p & a_{p-1} & \cdots & \cdots & a_0 \\ b_q & b_{q-1} & \cdots & & b_0 & & & \\ & b_q & b_{q-1} & \cdots & & b_0 & & \\ & & \ddots & & & & \ddots & \\ & & & \ddots & & & & \ddots \\ & & & & b_q & b_{q-1} & \cdots & b_0 \end{pmatrix}}^{\text{p+q columns}} \left.\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}\right\} \begin{matrix} \text{q rows} \\ \\ \\ \\ \text{p rows} \\ \\ \\ \\ \end{matrix}$$

For $i = 0, \ldots, \min(q, p-1)$, let $\mathrm{Syl}_{y,i}(P,Q)$ be the $(p+q-2i)\times(p+q-i)$ matrix obtained from $\mathrm{Syl}_y(P,Q)$ by deleting the $i$ last rows of the coefficients of $P$, the $i$ last rows of the coefficients of $Q$, and the $i$ last columns.

**Definition 1.** *([Kah03, §3]). For $i = 0, \ldots, \min(q, p-1)$, the $i$-th polynomial subresultant of $P$ and $Q$, denoted by $\mathrm{Sres}_{y,i}(P,Q) = \mathrm{sres}_{y,i}(P,Q)y^i + \mathrm{sres}_{y,i,i-1}(P,Q)y^{i-1} + \cdots + \mathrm{sres}_{y,i,0}(P,Q)$ is the polynomial determinant of $\mathrm{Syl}_{y,i}(P,Q)$.*

For practical consideration, when $q = p$, we define the $q$-th polynomial subresultant of $P$ and $Q$ as $Q$.[4] The polynomial $\mathrm{Sres}_i(P,Q)$ has degree at most $i$ in $y$ and it can be written as $\mathrm{sres}_{y,i}(P,Q)y^i + \mathrm{sres}_{y,i,i-1}(P,Q)y^{i-1} + \cdots + \mathrm{sres}_{y,i,0}(P,Q)$, where the coefficient of its monomial of degree $i$ in $y$, $\mathrm{sres}_i(P,Q)$, is called the $i$-th *principal subresultant coefficient*. Unless specified otherwise, the subresultants are always considered with respect to the variable $y$ and then, for simplicity, we do not explicitly refer to the variable in the notation. Note that $\mathrm{Sres}_0(P,Q) = \mathrm{sres}_0(P,Q)$ is the *resultant* of $P$ and $Q$ with respect to $y$, which we also denote by $\mathrm{Res}_y(P,Q)$. Again, when the resultant is considered with respect to $y$, we omit the reference to the variable and denote it by $\mathrm{Res}(P,Q)$.

The matricial definition of subresultants implies the so-called *specialization property* of subresultants, that is $\phi(\mathrm{Sres}_i(P,Q)) = \mathrm{Sres}_i(\phi(P),\phi(Q))$ for any morphism $\phi$ between $\mathbb{D}$ and another unique factorization domain $\mathbb{D}'$ such that none of the leading coefficients of $P$ and $Q$ vanishes through $\phi$. More generally, the equality holds up to a non-zero multiplicative constant in $\mathbb{D}'$ when only one of the leading coefficients vanishes [Kah03, Lemmas 2.3, 3.1].

[4] It can be observed that, when $p > q$, the $q$-th subresultant is equal to $b_q^{p-q-1}Q$, however it is not defined when $p = q$. In this case, El Kahoui suggests to extend the definition to $b_q^{-1}Q$ assuming that the domain $\mathbb{D}$ is integral. However, $b_q^{-1}$ does not necessarily belong to $\mathbb{D}$, which is not practical. Note that it is important to define the $q$-th subresultant to be a multiple of $Q$ so that Lemma 2 holds when $P(\alpha, y)$ and $Q(\alpha, y)$ have same degree and are multiple of one another.

We state in Lemma 2 a classical fundamental property of subresultants which is instrumental in the triangular decomposition algorithm. For clarity, we state this property for bivariate polynomials $P = \sum_{i=0}^{p} a_i y^i$ and $Q = \sum_{i=0}^{q} b_i y^i$ in $\mathbb{D}[x, y]$, with $p \geqslant q$.

Before stating Lemma 2, we recall that a greatest common divisor (gcd) of $P$ and $Q$ is a polynomial in $\mathbb{D}[x, y]$ that divides $P$ and $Q$ such that any common divisor of $P$ and $Q$ also divides the gcd in $\mathbb{D}[x, y]$. The greatest common divisor is unique only up to the multiplication by an invertible element of $\mathbb{D}$. When $\mathbb{D}$ is equal to $\mathbb{Z}$, the gcd of $P$ and $Q$ is unique up to its sign and we refer to any of them as *the* gcd for simplicity. On the other hand, when $\mathbb{D}$ is a field, we refer to the monic gcd (with respect to a given ordering of the variables) as to *the* gcd. Furthermore, in the sequel, we sometimes compare gcds defined in $\mathbb{F}_\mu[x, y]$ and the reduction modulo $\mu$ of gcds defined in $\mathbb{Z}[x, y]$; for simplicity, we often say they are equal if they are equal up to the multiplication by a non-zero constant in $\mathbb{F}_\mu$. Note finally that if $P$ and $Q$ are coprime in $\mathbb{Z}[x, y]$, then they define a zero-dimensional system.

**Lemma 2.** *For any $\alpha$ such that $a_p(\alpha)$ and $b_q(\alpha)$ do not both vanish, the first subresultant polynomial $\mathrm{Sres}_k(P, Q)(\alpha, y)$ (for $k$ increasing) that does not identically vanish is of degree $k$, and it is the gcd of $P(\alpha, y)$ and $Q(\alpha, y)$ (up to the multiplication by a non-zero constant in the fraction field of $\mathbb{D}(\alpha)$).*

We recall complexity results, using fast algorithms, on subresultants and gcd computations.

**Lemma 3** ([BPR06, Prop. 8.46] [Rei97, §8] [vzGG13, §11.2])**.** *Let $P$ and $Q$ be in $\mathbb{Z}[x_1, \ldots, x_n][y]$ (n fixed) with coefficients of bitsize at most $\tau$ such that their degrees in $y$ are bounded by $d_y$ and their degrees in the other variables are bounded by $d$.*

- *The coefficients of $\mathrm{Sres}_i(P, Q)$ have bitsize in $\widetilde{O}(d_y\tau)$.*
- *The degree in $x_j$ of $\mathrm{Sres}_i(P, Q)$ is at most $2d(d_y - i)$.*
- *For any $i \in \{0, \ldots, \min(\mathrm{d}_y(P), \mathrm{d}_y(Q))\}$, the polynomial $\mathrm{Sres}_i(P, Q)$ can be computed in $\widetilde{O}(d^n d_y^{n+1})$ arithmetic operations and $\widetilde{O}_B(d^n d_y^{n+2}\tau)$ bit operations. These complexities also hold for the computation of the sequence of principal subresultant coefficients $\mathrm{sres}_i(P, Q)$.*[5]

In the univariate case, we need a refinement of the previous lemma in the case of two polynomials with different degrees and bitsizes. In addition, we often consider the gcd of two univariate polynomials $P$ and $Q$ and the gcd-free part of $P$ with respect to $Q$, that is $\frac{P}{\gcd(P,Q)}$. Note that when $Q = P'$, the latter is the squarefree part of $P$. Since the gcd and gcd-free part can be computed via subresultants, we summarize all these complexity results in the following lemma.

**Lemma 4** ([LR01] [vzGG13, §11.2])**.** *Let $P$ and $Q$ be two polynomials in $\mathbb{Z}[y]$ of degrees $p$ and $q \leqslant p$ and of bitsizes $\tau_P$ and $\tau_Q$, respectively.*

- *The coefficients of $\mathrm{Sres}_i(P, Q)$ have bitsize in $\widetilde{O}(p\tau_Q + q\tau_P)$.*
- *Any subresultant $\mathrm{Sres}_i(P, Q)$ as well as the sequence of principal subresultant coefficients $\mathrm{sres}_i(P, Q)$ can be computed in $\widetilde{O}(p)$ arithmetic operations, and $\widetilde{O}_B(p(p\tau_Q + q\tau_P))$ bit operations.*
- *In $\mathbb{Z}[y]$, the gcd of $P$ and $Q$ has bitsize $O(\min(p + \tau_P, q + \tau_Q))$ and it can be computed in $\widetilde{O}(p)$ arithmetic operations, and $\widetilde{O}_B(p(p\tau_Q + q\tau_P))$ bit operations. The gcd-free part of $P$ with respect to $Q$ has bitsize $O(p + \tau_P)$ and it can be computed in the same complexities.*

We also state the following complexity on the computation of the gcd and gcd-free parts of bivariate polynomials, whose proof is a minor refinement of one in [MSW15].

**Lemma 5.** *Given $P$ and $Q$ in $\mathbb{Z}[x, y]$ of maximum degree $d$ and maximum bitsize $\tau$, their gcd and the gcd-free parts can be computed in $\widetilde{O}_B(d^5 + d^4\tau)$ bit operations in the worst case.*

### 2.1.2 Lucky primes for gcd computations

We use three notions of lucky primes. We recall here the definition of lucky primes for gcds and we later introduce the definition of lucky primes for algebraic systems (Definition 15) and for triangular decompositions (Definition 11). Let $A$ and $B$ be polynomials in $\mathbb{Z}[x]$.

**Definition 6** ([Yap00, §4.4])**.** *A prime number $\mu$ is* **lucky for the gcd** *of $A$ and $B$ if*

- *$\phi_\mu(\mathrm{Lc}(A) \cdot \mathrm{Lc}(B)) \neq 0$, and*

[5] The complexity of computing the sequence of principal subresultant coefficients is stated in [vzGG13, §. 11.2] only for univariate polynomials, however, one can use the binary segmentation technique described in [Rei97, §8] to generalize the latter to multivariate polynomials.

- $\gcd(A, B)$ *has the same degree as* $\gcd(A_\mu, B_\mu)$.

**Lemma 7** ([Yap00, Lemmas 4.11 and 4.12])**.** *A prime number is lucky for the gcd of $A$ and $B$ if and only if it divides the leading coefficient of neither $A$, nor $B$, nor $\mathrm{Sres}_d(A, B)$ where $d$ is the degree of $\gcd(A, B)$. When $\mu$ is lucky for the gcd of $A$ and $B$, then $\phi_\mu(\gcd(A, B)) = \gcd(A_\mu, B_\mu)$ (up to a non-null factor in $\mathbb{F}_\mu$).*

### 2.1.3 Multiplicities

We define the two notions of multiplicities. The second one is important for the Isotop algorithm.

**Definition 8.** *Let $I$ be an ideal of $\mathbb{D}[x, y]$ and denote by $\mathbb{F}$ the algebraic closure of $\mathbb{D}$. To each zero $(\alpha, \beta)$ of $I$ corresponds a local ring $(\mathbb{F}[x, y]/I)_{(\alpha,\beta)}$ obtained by localizing the ring $\mathbb{F}[x, y]/I$ at the maximal ideal $\langle x - \alpha, y - \beta\rangle$. When this local ring is finite dimensional as $\mathbb{F}$-vector space, this dimension is called the* **multiplicity of $(\alpha, \beta)$ as a zero of I** *and is noted $\mathrm{mult}((\alpha, \beta), I)$[CLO05, §4.2].*

*We call the fiber of a point $p = (\alpha, \beta)$ the vertical line of equation $x = \alpha$. The* **multiplicity of p in its fiber** *with respect to a system of polynomials $\{P, Q\}$ in $\mathbb{F}[x, y]$ is the multiplicity of $\beta$ in the univariate polynomial $\gcd(P(\alpha, y), Q(\alpha, y))$. (This multiplicity is zero if $P$ or $Q$ does not vanish at $p$.)*

## 2.2 Complexity

There are numerous alternatives for solving algebraic systems. Typically, isolating boxes of the solutions can be computed either directly from the input system using numerical methods (such as subdivision or homotopy) or indirectly by first computing intermediate symbolic representations such as triangular sets, Gröbner bases, or rational parameterizations. However, only little work analyzes the bit complexity of isolating the solutions without any restriction, in particular for non-generic or non-radical systems. We present in this section our contributions to the problem of solving systems of *bivariate* polynomials with integer coefficients and we focus on the bit complexity of these methods in the RAM model. We focus in particular on the worst-case bit complexity in a deterministic setting and on the expected bit complexity in a probabilistic Las Vegas setting. Recall that, in Las Vegas algorithms, the sequence and number of operations are probabilistic but the output is deterministic (or, in other words, the number of operations is a random variable but the output is always correct). On the other hand, in Monte Carlo algorithms, the output is only correct with some probability. We consider throughout this section input polynomials of total degree at most $d$ with integer coefficients of bitsize at most $\tau$.

### 2.2.1 Previous work

A classical approach for solving a system of polynomials with a finite number of solutions is to compute a rational parameterization of its solutions. A rational parameterization is a representation of the (complex) solutions by a set of univariate polynomials and associated rational one-to-one mappings that send the roots of the univariate polynomials to the solutions of the system. With such a representation, many queries on the system can be transformed into queries on univariate polynomials, which eases the computations. For instance, isolating the solutions of the system can be done by isolating the roots of the univariate polynomials of the rational parameterization and by computing the image of the resulting intervals through the associated mappings. Similarly, querying whether a polynomial $P$ vanishes at the solutions of the system can be done by substituting in $P$ the variables by their images in each of the one-to-one mappings and by testing whether this resulting univariate polynomial vanishes at the roots of the associated univariate polynomial in the rational parameterization.

The core of the algorithms that compute such rational parametrizations (see for example [ABRW96, BSS03, DET09, GLS01, GVEK96, Rou99] and references therein) is the computation of a so-called *linear separating form* for the solutions, that is, a linear combination of the coordinates that takes different values when evaluated at different solutions of the system. Then, a shear of the coordinate system using such a linear form ensures that the system is in generic position, in the sense that no two solutions are vertically aligned. Since a linear form chosen randomly in a sufficiently large finite set is separating with probability close to one, probabilistic Monte Carlo algorithms can avoid this computation by considering a random linear form. Computing deterministically a separating linear form $x + ay$ without constraint on the size of $a$ is also trivial (using upper bounds and separation bounds on roots of polynomials). However, in order not to impact the bit complexity for computing the rational parameterizations and their bitsizes, the values of $a$ have to be small, i.e., polynomial in $d$ and $\tau$. Surprisingly, computing a

certified linear separating form with $a$ small, or even to check whether an arbitrary (e.g. random) linear form is separating, is a bottleneck in the computation of rational parameterizations, even for bivariate systems, as discussed below.

For arbitrary multivariate systems, Rouillier [Rou99] gives an algorithm for deterministically computing a separating form, which computes the number of solutions of a system with the rank of the Hermite quadratic form of the quotient algebra. The complexity of this computation dominates the one that follows for computing the rational representation. Considering the special case of systems of two bivariate polynomials of total degree bounded by $d$ with integer coefficients of bitsize bounded by $\tau$, another approach, based on a triangular decomposition, has been presented by Gonzalez-Vega and El Kahoui [GVEK96] for computing a separating linear form together with a rational parameterization of the solutions. The best-known bit complexity of this approach, analyzed by Diochnos et al. [DET09, Lemma 16 & Theorem 19][6], shows a bit complexity in $\widetilde{O}_B(d^{10} + d^9\tau)$ for computing a separating form and a bit complexity in $\widetilde{O}_B(d^7 + d^6\tau)$ for computing the corresponding rational parameterization. The computation of a separating linear form was still the bottleneck in the computation of the rational parameterization. An algorithm using modular arithmetic was then introduced by Bouzidi et al. [BLPR15*] reducing the complexity to $\widetilde{O}_B(d^8 + d^7\tau)$. This algorithm was later simplified and improved by transforming the problem into the computation of a separating form for the critical points of a curve, which improved the bit complexity to $\widetilde{O}_B(d^7 + d^6\tau)$ in the worst case and to $\widetilde{O}_B(d^5 + d^4\tau)$ in a probabilistic Las Vegas setting [BLM+14*]. Bouzidi et al. [BLPR15*] also showed that, given such a separating linear form, an alternative rational parameterization called Rational Univariate Representation (RUR) [Rou99] can be computed using $\widetilde{O}_B(d^7 + d^6\tau)$ bit operations. For the first time, the worst-case bit complexities of computing linear separating forms and rational parameterizations (even RURs) of bivariate systems were both in the same class of complexity $\widetilde{O}_B(d^7 + d^6\tau)$ and, also for the first time, the expected bit complexity for computing linear separating forms, in a Las Vegas setting, was in a smaller class of complexity, $\widetilde{O}_B(d^5 + d^4\tau)$.

For computing a RUR in a Monte Carlo setting, the first step of computing a separating linear form is trivial since a random form is separating with some known probability. Doing so, Mehrabi and Schost [MS16] then compute a RUR of the radical of the input system with probability of success at least $1-1/2^p$ in a bit complexity that is the sum of $\widetilde{O}_B(d^{2+\varepsilon}(d^2 + d\tau + dp + p^2))$, for $\varepsilon > 0$ arbitrarily small, and of the complexity of computing a random prime smaller than $M = (2^p d\tau)^{O(1)}$. This result is remarkable since the whole complexity is $\widetilde{O}_B(d^{4+\varepsilon} + d^{3+\varepsilon}\tau)$ when $p$ is a constant, which almost matches the worst-case upper and lower bounds $\widetilde{O}(d^4 + d^3\tau)$ and $\widetilde{\Omega}(d^4)$ on the size of the output (see Corollary 28 and [MS16, Main results]). However the drawback of this approach is that, so far, neither the separating form nor the computed RUR can be checked for correctness with a better bit complexity than those we present in this section, that is $\widetilde{O}_B(d^5 + d^4\tau)$ expected and $\widetilde{O}_B(d^6 + d^5\tau)$ in the worst case.

Recently, Kobel and Sagraloff [KS15] presented an algorithm of worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ for computing isolating boxes of the solutions of bivariate systems. Their approach is based on resultant computations, projecting the solutions on the $x$ and $y$-axes, thus defining a grid of candidate solutions. Then, approximate evaluations combined with adaptive evaluation bounds enable to identify the solutions from the grid. This method does not need the knowledge of a separating form, but once the solutions are isolated with enough precision, such a separating form can be computed in $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations [KS15]. This approach for computing separating linear forms has the best known worst-case complexity. However, it would be surprising that computing a separating form with such complexity would require to first isolate the solutions of the system, since separating forms are precisely instrumental for solving systems in parameterization-based approaches. Our contribution indeed demonstrates that separating linear forms and rational parameterizations (including RURs) can be directly computed with this $\widetilde{O}_B(d^6 + d^5\tau)$ state-of-the-art worst-case bit complexity, and that the solutions of the system can be isolated from the RUR in the same worst-case complexity.

Note furthermore that, for computing isolating boxes of the solutions, no complexity better than $\widetilde{O}_B(d^6 + d^5\tau)$ is known even if a RUR is given and even in a probabilistic setting. Indeed, all known algorithms require to isolate the roots of a univariate polynomial of degree at most $d^2$ and bitsize $\widetilde{O}(d^2 + d\tau)$, for which no complexity better than $\widetilde{O}_B(d^6 + d^5\tau)$ is known, even in a probabilistic setting, and this has not been improved for years [Pan02]. Hence, following the above discussion, in a Monte Carlo setting, computing isolating boxes of the solutions is the bottleneck of the whole process of solving

[6] The overall bit complexity stated in [DET09, Theorem 19] is $\widetilde{O}_B(d^{12} + d^{10}\tau^2)$ because it includes the isolation of the solutions of the system. Note, however, that the complexity of the isolation phase, and thus of the whole algorithm, decreases to $\widetilde{O}_B(d^{10} + d^9\tau)$ using Pan [Pan02] results on the complexity of isolating the real roots of a univariate polynomial.

bivariate systems, and this is also the case in a Las Vegas setting with the results we present in this section.

### 2.2.2 Triangular decomposition

We first recall the classical subresultant-based algorithm for computing the triangular decomposition of a zero-dimensional bivariate system $\{P, Q\}$. This decomposition appears, for instance, for solving bivariate systems [LMMRS11] or for the computation of the topology of curves [GVEK96]. This triangular decomposition algorithm will be used in our algorithm for computing a RUR decomposition of $\{P, Q\}$.

We then present a straightforward variation on this algorithm, which only computes the degree of this triangular decomposition (see Definition 10). This variation decreases the expected bit complexity of the algorithm and it is critical for our Las Vegas algorithm for computing a separating linear form.

We then present another variation on the triangular decomposition algorithm, which computes a luckiness certificate for this triangular decomposition. A luckiness certificate of $\{P, Q\}$ is an integer such that if a prime $\mu$ does not divide it, then $\mu$ is lucky for the triangular decomposition of $\{P, Q\}$ that is, the degree of the decomposition is preserved by the reduction modulo $\mu$ and the decomposition commutes with the reduction modulo $\mu$ (see Definition 11). Our deterministic algorithms for the separating form and the RUR computations will both use this luckiness certificate.

We finally state that the worst-case bit complexities of these three algorithms are in $\widetilde{O}_B(d^6 + d^5\tau)$ and that the expected bit complexity of the one for computing the degree of the triangular decomposition is in $\widetilde{O}_B(d^5 + d^4\tau)$ (Proposition 14). The worst-case complexity is obtained by considering amortized bounds on the degrees and bitsizes of factors of the resultant and it improves by a factor $d$ the state-of-the-art complexity for computing the triangular decomposition [DET09, Proof of Theorem 19]. Besides of being of independent interest, these improvements are critical for the complexity analysis of the following algorithms.

#### Triangular decomposition via subresultants

The idea is based on the specialization lemma of subresultants (Lemma 2) which states that, after specialization at $x = \alpha$, the first (with respect to increasing $i$) non-zero subresultant $\mathrm{Sres}_i(P, Q)(\alpha, y)$ (Definition 1) is of degree $i$ and is equal to the gcd of $P(\alpha, y)$ and $Q(\alpha, y)$. This induces a decomposition into triangular subsystems $\{A_i(x), \mathrm{Sres}_i(P, Q)(x, y)\}$ where a solution $\alpha$ of $A_i(x) = 0$ is such that the system $\{P(\alpha, y), Q(\alpha, y)\}$ admits exactly $i$ roots (counted with multiplicity), which are exactly those of $\mathrm{Sres}_i(P, Q)(\alpha, y)$. Furthermore, these triangular subsystems are regular chains, i.e., the leading coefficient of the bivariate polynomial (seen in $y$) is coprime with the univariate polynomial. We recall in Algorithm 1 how this decomposition is computed. Note that this algorithm performs $\widetilde{O}(d^4)$ arithmetic operations. Indeed, the computation of the subresultant sequence has complexity $\widetilde{O}(d^4)$ and there are at most $d$ gcd computations each of complexity $\widetilde{O}(d^2)$ (see e.g. [BLPR15*, Lemma 15] for details). The next lemma summarizes the main properties of this triangular decomposition.

**Lemma 9** ([GVEK96, LMMRS11]). *Algorithm 1 computes a triangular decomposition $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$ such that*

- *the set of distinct solutions of $\{P, Q\}$ is the disjoint union of the sets of distinct solutions of the $\{A_i(x), B_i(x, y)\}$, $i \in \mathcal{I}$,*
- *$\prod_{i\in\mathcal{I}} A_i$ is squarefree,*
- *for any root $\alpha$ of $A_i$, $B_i(\alpha, y)$ is of degree $i$ and equals $\gcd(P(\alpha, y), Q(\alpha, y))$ (up to a constant factor),*
- *$A_i$ is coprime with $\mathrm{Lc}_y(B_i) = \mathrm{sres}_i(P, Q)$.*

#### Degree of a triangular decomposition

**Definition 10.** *The degree of a triangular decomposition $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$ of is the sum of the degrees of these systems, that is*

$$\sum_{i\in\mathcal{I}} \mathrm{d}_x(A_i(x))\ \mathrm{d}_y(B_i(x, y))$$

*where $\mathrm{d}_x$ refers to the degree of the polynomial with respect to $x$ and similarly for $y$. For simplicity, we refer to the degree of the triangular decomposition of $\{P, Q\}$ as to the degree of the triangular decomposition computed by Algorithm 1 on $\{P, Q\}$.*

**Algorithm 1** Triangular decomposition [GVEK96, LMMRS11]

**Input:** $P, Q$ in $\mathbb{D}[x, y]$ coprime such that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime.
**Output:** Triangular decomposition $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$ such that the set of solutions of $\{P, Q\}$ is the disjoint union of the sets of solutions of $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$.
1: If needed, exchange $P$ and $Q$ so that $\mathrm{d}_y(Q) \leqslant \mathrm{d}_y(P)$.
2: Compute the subresultant sequence of $P$ and $Q$ with respect to $y$: $B_i = \mathrm{Sres}_i(P, Q)$.
3: $G_0 =$ squarefree part$(\mathrm{Res}(P, Q))$ and $\mathcal{T} = \emptyset$
4: **for** $i = 1$ **to** $\mathrm{d}_y(Q)$ **do**
5: $\quad G_i = \gcd(G_{i-1}, \mathrm{sres}_i(P, Q))$
6: $\quad A_i = G_{i-1}/G_i$
7: $\quad$ if $\mathrm{d}_x(A_i) > 0$, add $(A_i, B_i)$ to $\mathcal{T}$
8: **return** $\mathcal{T} = \{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$

**Algorithm 1'** Degree of the triangular decomposition

**Input:** $P, Q$ in $\mathbb{D}[x, y]$ coprime such that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime.
**Output:** The degree of the triangular decomposition of $\{P, Q\}$.
1: If needed, exchange $P$ and $Q$ so that $\mathrm{d}_y(Q) \leqslant \mathrm{d}_y(P)$.
2: Compute $(\mathrm{sres}_i(P, Q))_{i=0,\ldots,\mathrm{d}_y(Q)}$ the sequence of principal subresultant coefficients of $P$ and $Q$ with respect to $y$.
3: $G_0 =$ squarefree part$(\mathrm{Res}(P, Q))$
4: **for** $i = 1$ **to** $\mathrm{d}_y(Q)$ **do**
5: $\quad G_i = \gcd(G_{i-1}, \mathrm{sres}_i(P, Q))$
6: **return** $\sum_{i\in\mathcal{I}}(\mathrm{d}_x(G_{i-1}) - \mathrm{d}_x(G_i))\, i$

For the notation in the algorithms, we refer the reader to Section 2.1. Algorithm 1', a straightforward variant of Algorithm 1, computes the degree of triangular decomposition of $\{P, Q\}$. The difference with Algorithm 1 is that we do not compute (in Line 2) the whole subresultant sequence but only the sequence of their principal coefficients. In other words, we do not compute the bivariate polynomials $B_i(x, y)$ of the triangular decomposition but only their leading terms (seen as polynomials in $y$). Furthermore, we do not compute the univariate polynomials $A_i(x)$ of the decomposition but only their degrees. This simplification does not modify the worst-case bit complexity of the algorithm but it decreases its expected bit complexity (see Proposition 14). This simplification is thus not needed in the deterministic version of our algorithm for computing a separating linear form but it is needed in our randomized version (see Section 2.2.3).

### Lucky primes for a triangular decomposition

We now define the lucky primes for the triangular decomposition of Algorithm 1 and introduce Algorithm 2 that computes a luckiness certificate i.e. an integer that is divisible by all the unlucky primes.

**Definition 11.** *A prime $\mu$ is* **lucky for the triangular decomposition** *of Algorithm 1 applied to $P$ and $Q$ if the decomposition commutes with the morphism $\phi_\mu$ (reduction modulo $\mu$) and its degree is invariant through $\phi_\mu$. More precisely, with $\{(A_i, B_i)\}_{i\in\mathcal{I}} =$ Algorithm 1(P, Q) and $\{(A_i^\mu, B_i^\mu)\}_{i\in\mathcal{I}^\mu} =$ Algorithm 1$(\phi_\mu(\mathrm{P}), \phi_\mu(\mathrm{Q}))$, $\mu$ is lucky if $\mathcal{I} = \mathcal{I}^\mu$, $\phi_\mu(A_i) = A_i^\mu$, $\phi_\mu(B_i) = B_i^\mu$ for every $i \in \mathcal{I}$ and the two triangular decompositions have the same degree. Note also that $(P, Q)$ and $(\phi_\mu(P), \phi_\mu(Q))$ are required to satisfy the hypotheses of Algorithm 1.*

**Lemma 12** ([BLM+16*, Lemma 12]). *[Correctness of Algorithm 2] The integer $\Pi$ output by Algorithm 2 is a luckiness certificate of $\{P, Q\}$, that is, if $\mu$ does not divide $\Pi$, then it is lucky for the triangular decomposition of $\{P, Q\}$.*

### Amortized complexity analysis

For the analysis of Algorithms 1, 1' and 2, we first prove amortized bounds on the degree and bitsize of the factors $G_i$ of the resultant in the triangular decomposition. The proof is using the multiplicativity of the Mahler measure of polynomials and Mignotte's bound.

**Algorithm 2** Luckiness certificate

**Input:** $P, Q$ in $\mathbb{Z}[x, y]$ coprime such that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime.

**Output:** A luckiness certificate of $\{P, Q\}$, that is, an integer $\Pi$ such that if $\mu$ does not divide $\Pi$, then $\mu$ is lucky for the triangular decomposition of $\{P, Q\}$ according to Definition 11.

1: If needed, exchange $P$ and $Q$ so that $\mathrm{d}_y(Q) \leqslant \mathrm{d}_y(P)$.
2: Compute $(\mathrm{sres}_i(P, Q))_{i=0,\ldots,\mathrm{d}_y(Q)}$ the sequence of principal subresultant coefficients of $P$ and $Q$ with respect to $y$.
3: $G_0 = \text{squarefree part}(\mathrm{Res}(P, Q))$
4: $SG_0 = \mathrm{sres}_{x,k}(\mathrm{Res}(P, Q), \frac{\partial \mathrm{Res}(P,Q)}{\partial x})$ the first non-null principal subresultant coefficient (for $k$ increasing).
5: **for** $i = 1$ **to** $\mathrm{d}_y(Q)$ **do**
6: $\quad SG_i = \mathrm{sres}_{x,k}(G_{i-1}, \mathrm{sres}_i(P, Q))$ the first non-null principal subresultant coefficient (for $k$ increasing).
7: $\quad G_i = \gcd(G_{i-1}, \mathrm{sres}_i(P, Q))$
8: $\Pi = \mathrm{Lc}_x(\mathrm{Lc}_y(P)) \cdot \mathrm{Lc}_x(\mathrm{Lc}_y(Q)) \cdot \mathrm{Res}_x(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q)) \cdot \mathrm{d}_x(\mathrm{Res}(P, Q)) \cdot \prod_{i=0}^{\mathrm{d}_y(Q)} SG_i \cdot \mathrm{Lc}_x(\mathrm{sres}_i(P, Q))$
9: **return** $\Pi$

**Proposition 13** ([BLM+16*, Proposition 15]). *For $i = 0, \ldots, \mathrm{d}_y(Q) - 1$, let $d_i$ and $\tau_i$ be the degree and bitsize of the polynomial $G_i$ in the triangular decomposition of $P$ and $Q$ computed in Algorithm 1. We have:*

- $d_i \leqslant \frac{d^2}{i+1}$ *and* $\tau_i = \widetilde{O}(\frac{d^2+d\tau}{i+1})$,
- $\sum_{i=0}^{\mathrm{d}_y(Q)-1} d_i \leqslant d^2$ *and* $\sum_{i=0}^{\mathrm{d}_y(Q)-1} \tau_i = \widetilde{O}(d^2 + d\tau)$.

**Proposition 14** ([BLM+16*, Proposition 16]). *If $P, Q$ in $\mathbb{Z}[x, y]$ have degree at most $d$ and bitsize at most $\tau$, Algorithms 1, 1' and 2 perform $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case. Algorithm 1' performs $\widetilde{O}_B(d^5 + d^4\tau)$ bit operations on average. The integer $\Pi$ output by Algorithm 2 has bitsize $\widetilde{O}(d^4 + d^3\tau)$.*

### 2.2.3 Separating linear form

We present a new algorithm for computing separating linear forms for a bivariate system $\{P, Q\}$. We actually present two algorithms, a deterministic one of worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ and a probabilistic Las Vegas variant of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ (Theorems 22 and 23).

Our approach is based on the algorithms presented in [BLPR15*] and [BLM+14*] while improving the worst-case bit complexity by one order of magnitude. We briefly recall the essence of these algorithms. The first step of the algorithm presented in [BLPR15*] is to compute the number of distinct solutions and a so-called lucky prime for the system. Such a lucky prime is, roughly speaking, a prime such that the system has the same number of distinct solutions as its image modulo $\mu$ (see Definition 15). In a second step, all polynomials and computations are considered modulo $\mu$. The algorithm then considers iteratively a candidate separating form $x + ay$ with an integer $a$ incrementing from 0. The algorithm computes the number of distinct solutions after projection along the direction of the line $x + ay = 0$ and stops when a value $a$ is found such that the number of distinct projected solutions equals that of the system. The worst-case bit complexity of this algorithm is in $\widetilde{O}_B(d^8 + d^7\tau)$.

The main additional ideas introduced in [BLM+14*] are that it is sufficient to compute a separating form for the system $\{H, \frac{\partial H}{\partial y}\}$ of critical points of a curve $H$ associated to the input system $\{P, Q\}$ and that the number of critical points can easily be computed as the difference between the degrees of the triangular decompositions of the systems $\{H, (\frac{\partial H}{\partial y})^2\}$ and $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 10). This improves the worst-case bit complexity of the algorithm to $\widetilde{O}_B(d^7 + d^6\tau)$.

We now show how these algorithms can be again improved by one order of magnitude in the worst case. The main ideas of these improvements are as follows. First, we show how our improvement on the complexity analysis of triangular decompositions presented in Section 2.2.2 improves the complexity of the computation of the number of solutions of $\{H, \frac{\partial H}{\partial y}\}$.

Second, we present a new algorithm for computing a lucky prime for the system $\{H, \frac{\partial H}{\partial y}\}$ using the luckiness certificates for triangular decompositions presented in Section 2.2.2. More precisely, we

compute a lucky prime for $\{H, \frac{\partial H}{\partial y}\}$ by computing a prime $\mu$ that, essentially, does not divide the product of the luckiness certificates of the two systems $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$. By definition of the luckiness certificates, the degrees of the triangular decompositions of these two systems are the same over $\mathbb{Z}$ and $\mathbb{F}_\mu = \mathbb{Z}/\mu\mathbb{Z}$. The difference of these degrees, which is the number of solutions of $\{H, \frac{\partial H}{\partial y}\}$, is thus also the same over $\mathbb{Z}$ and $\mathbb{F}_\mu$, which essentially yields that $\mu$ is lucky for $\{H, \frac{\partial H}{\partial y}\}$.

The last ingredient of our algorithm is to show how, given the number of solutions and a lucky prime for the system $\{H, \frac{\partial H}{\partial y}\}$, the bit complexity of the algorithm presented in [BLPR15*] for computing a separating linear form for $\{H, \frac{\partial H}{\partial y}\}$ can be improved from $\widetilde{O}_B(d^8 + d^7\tau)$ to $\widetilde{O}_B(d^6 + d^5\tau)$ by using multipoint evaluation and changing the organization of the computations.

We wrap up these results, which prove that we can compute a separating linear form for the input system $\{P, Q\}$ in $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case (Theorem 22).

Finally, we show that our deterministic algorithm can be modified in a straightforward manner into a probabilistic Las Vegas algorithm of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ (Theorem 23). This is done by choosing randomly a linear form $x + ay$ and a prime $\mu$ for the system $\{H, \frac{\partial H}{\partial y}\}$, until the number of distinct solutions of $\{H, \frac{\partial H}{\partial y}\}$ is equal to the number of distinct solutions of that system modulo $\mu$ and after projection along the direction of the line $x + ay = 0$.

This new algorithm is similar to one presented in [BLM+14*] and it has the same expected bit complexity, while its worst-case counterpart is improved by a factor $d$. Furthermore, this new algorithm is simpler because, in particular, (i) we choose random values for $a$ and $\mu$ independently instead of first computing a lucky prime and only then a separating form and (ii) we avoid the explicit computation of the constant in the asymptotic upper bound on the number of unlucky primes. Point (i) makes the presentation of the algorithm substantially simpler and (ii) avoids the unpleasant computation of an explicit bound and, as a consequence, also avoids computing random primes in unnecessary large sets.

### Notation and definitions

Given the two input polynomials $P$ and $Q$, we consider the "generic" change of variables $x = t - sy$, and define the "sheared" polynomials $P(t - sy, y)$, $Q(t - sy, y)$, and their resultant with respect to $y$,

$$R(t, s) = \mathrm{Res}(P(t - sy, y), Q(t - sy, y)).$$

We introduce the following notation for the leading coefficients of these polynomials;

$$L_P(s) = \mathrm{Lc}_y(P(t - sy, y)), \quad L_Q(s) = \mathrm{Lc}_y(Q(t - sy, y)). \tag{2.1}$$

Note that these polynomials do not depend on $t$.

**Definition 15** ([BLPR15*, Definition 8])**.** *A prime number $\mu$ is said to be* **lucky for a zero-dimensional system** $\{P, Q\}$ *if $\{P, Q\}$ and $\{\phi_\mu(P), \phi_\mu(Q)\}$ have the same number of distinct solutions (in their respective algebraic closures), $\mu > 2d^4$ and $\phi_\mu(L_P(s))\ \phi_\mu(L_Q(s)) \not\equiv 0$.*

Note that we consider $\mu$ in $\Omega(d^4)$ in Definition 15 because, in Algorithm 5, we want to ensure that there exists, for the system $\{\phi_\mu(P), \phi_\mu(Q)\}$ (resp. $\{P, Q\}$), a separating form $x + ay$ with $a$ in $\mathbb{F}_\mu$ (resp. $0 \leqslant a < \mu$ in $\mathbb{Z}$). The constant 2 in the bound $2d^4$ is an overestimate, which simplifies some proofs in [BLPR15*].

**Definition 16.** *Let $H$ be a polynomial in $\mathbb{Z}[x, y]$. A* **separating form for the curve** *defined by $H$ is a separating form for the system $\{H, \frac{\partial H}{\partial y}\}$ of critical points of the curve.*

Remark that shearing the critical points of a curve (with respect to $x$) is not the same as taking the critical points of a sheared curve. In particular, given a separating form $x + ay$ for a curve, it is possible that the shearing $(x, y) \mapsto (x' = x + ay, y)$ does not shear the curve in a generic position in the sense of Gonzalez-Vega et al. [GVEK96], that is the critical points (with respect to $x'$) of the sheared curve may be vertically aligned.

### From a system to a curve

We present here Proposition 17, which states that it is essentially equivalent from an asymptotic bit complexity point of view to compute a separating linear form for a system $\{P, Q\}$ and to compute a separating linear form for the critical points of a curve $H$. According to Definition 16, we refer to the latter as a separating linear form for $H$.

**Algorithm 3** Number of critical points of $H$

**Input:** $H$ in $\mathbb{Z}[x,y]$ squarefree such that $\mathrm{Lc}_y(H)$ is in $\mathbb{Z}$.
**Output:** The number of critical points of $H$.

1: **return** Algo 1'$(H, (\frac{\partial H}{\partial y})^2)$ - Algo 1'$(H, \frac{\partial H}{\partial y})$

The critical points of a curve of equation $H$ are the solutions of the system $\{H, \frac{\partial H}{\partial y}\}$, thus computing a separating linear form for a curve amounts, by definition, to computing a separating linear form for a system of two equations. Conversely, a separating linear form for the curve $PQ$ is also separating for the system $\{P,Q\}$ since any solution of $\{P,Q\}$ is also solution of $PQ$ and of $\frac{\partial PQ}{\partial y} = P\frac{\partial Q}{\partial y} + \frac{\partial P}{\partial y}Q$. However, it may happen that the curve $PQ$ admits no separating linear form even if $\{P,Q\}$ admits one. Indeed, $\{P,Q\}$ can be zero-dimensional while $PQ$ is not squarefree (and such that the infinitely many critical points cannot be separated by a linear form). Nevertheless, if $P$ and $Q$ are coprime and squarefree, then $PQ$ is squarefree and thus it has finitely many singular points. Still the curve $H = PQ$ may contain vertical lines, and thus infinitely many critical points, but this issue can easily be handled by shearing the coordinate system.

**Proposition 17** ([BLM+16*, Proposition 22]). *Let $P$ and $Q$ be two coprime polynomials in $\mathbb{Z}[x,y]$ of maximum degree $d$ and maximum bitsize $\tau$. We can compute a shearing of the coordinate system from $(x,y)$ to $(t = x+\alpha y, y)$, with $\alpha$ an integer in $[0,2d]$, and a squarefree polynomial $H$ in $\mathbb{Z}[t,y]$ of degree at most $2d$, bitsize $\widetilde{O}(d+\tau)$, with $\mathrm{Lc}_y(H)$ in $\mathbb{Z}$, so that any separating linear form for the zero-dimensional system $\{H, \frac{\partial H}{\partial y}\}$ is also separating for $\{P,Q\}$ after being sheared back. The worst-case complexity of this computation is in $\widetilde{O}_B(d^5 + d^4\tau)$.*

We consider an arbitrary curve defined by $H$ in $\mathbb{Z}[x,y]$ of degree $d$ and bitsize $\tau$, squarefree and with a constant leading coefficient in $y$. In particular, the polynomial $H$ defined in Proposition 17 satisfies these two last conditions, which yield that the curve has a finite number of critical points. We show in the remaining of this section that computing (i) the number of the critical points of $H$, (ii) a lucky prime for the system of critical points $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 15), and finally (iii) a separating form for the curve $H$ (Definition 16) can be done with a bit complexity in $\widetilde{O}_B(d^6 + d^5\tau)$.

Algorithm 3 computes the number of critical points of a curve $H$ as the difference between the degrees of the triangular decompositions of the systems $\{H, (\frac{\partial H}{\partial y})^2\}$ and $\{H, \frac{\partial H}{\partial y}\}$. The complexity of Algorithm 3 follows directly from Proposition 14.

**Proposition 18** ([BLM+16*, Lemma 23]). *If $H$ in $\mathbb{Z}[x,y]$ has degree $d$ and bitsize $\tau$, Algorithm 3 computes the number of distinct critical points of $H$ in $\widetilde{O}_B(d^6+d^5\tau)$ bit operations in the worst case and $\widetilde{O}_B(d^5 + d^4\tau)$ bit operations on average.*

We now introduce Algorithm 4 computing a lucky prime for the system of critical points. Let $\Pi =$ Algo 2$(H, \frac{\partial H}{\partial y}) \cdot$ Algo 2$(H, (\frac{\partial H}{\partial y})^2)$ be the product of the luckiness certificates output by Algo 2 for the triangular decompositions of $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$. Lemma 19 shows that we can easily check using a divisibility test on $\Pi$ whether a prime number is lucky for the system $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 15).

**Lemma 19** ([BLM+16*, Lemma 24]). *Let $\mu$ be a prime such that $\mu > 2d^4$ and $\phi_\mu(L_H(s))\ \phi_\mu(L_{\frac{\partial H}{\partial y}}(s)) \not\equiv 0$. If $\mu$ does not divide $\Pi$ then $\mu$ is lucky for the system $\{H, \frac{\partial H}{\partial y}\}$.*

Algorithm 4 finds such a lucky prime by an iterative application of this divisibility test. To keep the complexity in the desired bound, the primes to be tested are grouped and a remainder tree is used for the computation of the reduction of $\Pi$ modulo all the primes in a group. In the following, $L_H(s)$ and $L_{\frac{\partial H}{\partial y}}(s)$ are defined similarly as in Equation (2.1).

**Proposition 20** ([BLM+16*, Proposition 25]). *Given $H$ in $\mathbb{Z}[x,y]$ of degree $d$ and bitsize $\tau$, Algorithm 4 computes a lucky prime of bitsize $O(\log(d\tau))$ for the system $\{H, \frac{\partial H}{\partial y}\}$ using $\widetilde{O}_B(d^4 + d^3\tau)$ bit operations.*

Algorithms 3 and 4 compute the number of distinct (complex) critical points of a curve and a lucky prime $\mu$ for the system of critical points. With this information, Algorithm 5 computes a separating form for the curve. More precisely, Algorithm 5 computes a separating linear form for a curve by considering iteratively linear forms $x+ay$, where $a$ is an integer incrementing from 0 and by computing the degree of the squarefree part of the reduction modulo $\mu$ of $R(t,a)$ until this degree is equal to the (known) number of distinct critical points of the curve and such that $\phi_\mu(L_H(a))\ \phi_\mu(L_{\frac{\partial H}{\partial y}}(a)) \neq 0$.

**Algorithm 4** Lucky prime for $\{H, \frac{\partial H}{\partial y}\}$

**Input:** $H$ in $\mathbb{Z}[x,y]$ of degree $d$ and bitsize $\tau$, squarefree such that $\mathrm{Lc}_y(H)$ is in $\mathbb{Z}$ and $\Pi$ = Algo 2$(H, \frac{\partial H}{\partial y})$ · Algo 2$(H, (\frac{\partial H}{\partial y})^2)$.
**Output:** A lucky prime $\mu$ for the system $\{H, \frac{\partial H}{\partial y}\}$.

Compute $L_H(s)$ and $L_{\frac{\partial H}{\partial y}}(s)$.
$m = 2d^4$
**while** true **do**
  Compute the set $B$ of the first $d^4 + d^3\tau$ primes $> m$.
  **for all** $\mu$ in $B$ **do**
    Compute the reduction mod. $\mu$ of $\Pi$, $L_H$, $L_{\frac{\partial H}{\partial y}}$.
    **if** $\phi_\mu(\Pi)\ \phi_\mu(L_H(s))\ \phi_\mu(L_{\frac{\partial H}{\partial y}}(s)) \not\equiv 0$ **then**
      **return** $\mu$
  $m$ = the largest prime in $B$

**Algorithm 5** Separating form of a curve

**Input:** $H$ in $\mathbb{Z}[x,y]$ of degree $d$ and bitsize $\tau$, squarefree and such that $\mathrm{Lc}_y(H)$ is in $\mathbb{Z}$.
**Output:** A separating linear form $x + ay$ of the curve $H$, with $a < 2d^4$.

Compute $N$ = Algorithm 3$(H)$, the number of distinct (complex) critical points of $H$.
Compute $\Pi$ = Algo 2$(H, \frac{\partial H}{\partial y})$ · Algo 2$(H, (\frac{\partial H}{\partial y})^2)$, the product of the luckiness certificates output by Algo 2 for the triangular decompositions of $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$.
Compute $\mu$ =Algorithm 4$(H, \Pi)$, a lucky prime for $\{H, \frac{\partial H}{\partial y}\}$.
Compute $H(t - sy, y)$ and $\frac{\partial H}{\partial y}(t - sy, y)$.
Compute $\Upsilon_\mu(s)$ the reduction modulo $\mu$ of $L_H(s) \cdot L_{\frac{\partial H}{\partial y}}(s)$.
Compute the resultant $R_\mu(t,s)$ of the reductions modulo $\mu$ of $H(t - sy, y)$ and $\frac{\partial H}{\partial y}(t - sy, y)$.
Compute $R_\mu(t,a)$ for all $a$ in $\{0, \ldots, 2d^4\}$ using multipoint evaluation.
$a = 0$
**repeat**
  Compute the degree $N_a$ of the squarefree part of $R_\mu(t,a)$.
  $a = a + 1$
**until** $\Upsilon_\mu(a) \neq 0$ (in $\mathbb{F}_\mu$) and $N_a = N$
**return** The linear form $x + ay$.

**Proposition 21** ([BLM+16*, Proposition 26]). *Given $H$ in $\mathbb{Z}[x,y]$ of degree $d$ and bitsize $\tau$, Algorithm 5 computes, with a worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$, an integer $a$ in $[0, 2d^4 - 2d]$ such that the linear form $x + ay$ is separating for the system $\{H, \frac{\partial H}{\partial y}\}$ of critical points of the curve $H = 0$.*

From a worst-case complexity point of view, the knowledge of the number $N$ of (distinct) complex critical points of the input curve in Algorithm 5 is not mandatory since one could instead compute the number of solutions $N_a$ of $R_\mu(t,a)$ for all integers $a$ smaller than $2d^4$ and output a value of $a$ that maximizes $N_a$. However, knowing $N$, the algorithm can stop as soon as a value of $a$ is found such that $N_a = N$, which improves the expected complexity of the algorithm in a Las Vegas setting, as discussed below.

### Separating linear form of a system

Propositions 17 and 21 directly yield the following theorem where the separating form is obtained by shearing back the separating form output by Algorithm 5.

**Theorem 22** ([BLM+16*, Proposition 28]). *Let $P, Q$ in $\mathbb{Z}[x,y]$ be of total degree at most $d$ and maximum bitsize $\tau$. A separating linear form $x + by$ for $\{P, Q\}$ with an integer $b$ in $[0, 2d^4]$ can be computed using $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case. Furthermore, $b$ is such that the leading coefficients of $P(t - by, y)$ and $Q(t - by, y)$ in $y$ are in $\mathbb{Z}$.*

### Las Vegas algorithm

**Algorithm 5'** Separating form of a curve – Las Vegas version

**Input:** $H$ in $\mathbb{Z}[x,y]$ of degree $d$ and bitsize $\tau$, squarefree and such that $\mathrm{Lc}_y(H)$ is in $\mathbb{Z}$.
**Output:** A separating linear form $x+ay$ of the curve $H$, with $a < 2d^4$.

1: Compute $N =$ Algorithm 3$(H)$, the number of distinct (complex) critical points of $H$.
2: Compute $H(t-sy,y)$, $\frac{\partial H}{\partial y}(t-sy,y)$, and $\Upsilon(s) = L_H(s) \cdot L_{\frac{\partial H}{\partial y}}(s)$.
3: $M = 2d^4$
4: **repeat**
5: $\quad M = 2M$
6: $\quad$ Choose uniformly at random an integer $a$ in $[0, 2d^4 - 2d]$ and a prime $\mu$ in $(2d^4, M)$.
7: $\quad$ Compute $\Upsilon_\mu(a) = \phi_\mu(\Upsilon)(a)$.
8: $\quad$ Compute $\phi_\mu(H(t-ay,y))$, $\phi_\mu(\frac{\partial H}{\partial y}(t-ay,y))$ and their resultant $R_{\mu,a}(t)$ with respect to $y$.
9: $\quad$ Compute the degree[7] $N_a$ of the squarefree part of $R_{\mu,a}(t)$.
10: **until** $\Upsilon_\mu(a) \neq 0$ (in $\mathbb{F}_\mu$) and $N_a = N$.
11: **return** The linear form $x+ay$.

We now show that the algorithm presented above for computing a separating linear form can easily be transformed into an efficient Las Vegas algorithm. Our Las Vegas algorithm is obtained from our deterministic version by only modifying Algorithm 5 into a randomized version, Algorithm 5'. The main difference between these two versions is that, in Algorithm 5', we choose randomly a candidate separating linear form $x+ay$ and a candidate lucky prime $\mu$ for $\{H, \frac{\partial H}{\partial y}\}$ (Definition 15) until the degree $N_a$ of the squarefree part of $R_\mu(t,a)$ is equal to the known number of solutions $N$. If $a$ and $\mu$ are chosen randomly in sufficiently large sets, the probability that $x+ay$ is separating and that $\mu$ is lucky is larger than a positive constant, which implies that the expected number of such choices is a constant.

This modification yields a major simplification: since we do not compute anymore a lucky prime in a deterministic way, we do not need Algorithm 4 (Lucky prime), which again implies that Algorithm 2 (Luckiness certificate) is not needed. Furthermore, note that, in Algorithm 5', we do not need anymore to use multipoint evaluation for evaluating $R_\mu(t,s)$ at $a$ since the expected number of choices of $a$ is a constant. Note finally that we choose the candidate lucky prime $\mu$ in increasingly larger sets. The reason is that, if we wanted to compute a unique set for which a random prime would be lucky with probability at least some constant, we would need an explicit upper bound (without $\widetilde{O}$ notation) on the number of unlucky primes and such a computation is highly unappealing.

**Theorem 23** ([BLM+16*, Theorem 29]). *Let $P, Q$ in $\mathbb{Z}[x,y]$ be of total degree at most $d$ and maximum bitsize $\tau$. A separating linear form $x+by$ for $\{P,Q\}$ with an integer $b$ in $[0, 2d^4]$ can be computed with $\widetilde{O}_B(d^5+d^4\tau)$ bit operations on average. Furthermore, $b$ is such that the leading coefficients of $P(t-by,y)$ and $Q(t-by,y)$ in $y$ are in $\mathbb{Z}$.*

### 2.2.4 RUR decomposition

We present an algorithm for computing a rational parameterization of the solutions of a bivariate system $\{P,Q\}$. We actually present two algorithms, a deterministic one of worst-case bit complexity $\widetilde{O}_B(d^6+d^5\tau)$ and a probabilistic Las Vegas variant of expected bit complexity $\widetilde{O}_B(d^5+d^4\tau)$ (Theorems 32 and 33). We consider here that a separating form $x+ay$ has been computed for $\{P,Q\}$ as shown in Section 2.2.3.

Recall that the two algorithms with best known bit complexity for computing rational parameterizations of the solutions are those by Gonzalez-Vega and El Kahoui [GVEK96] and by Bouzidi et al. [BLPR15*], both of complexity $\widetilde{O}_B(d^7+d^6\tau)$. The former algorithm first shears the input polynomials according to the separating form (to ensure that no two solutions are vertically aligned) and then computes a parameterization of the solutions of every system of the triangular decomposition of the sheared system; the multiplicities of the solutions of $\{P,Q\}$ are thus not preserved. The latter algorithm computes a single RUR of $\{P,Q\}$, which preserves the multiplicities of the solutions (see Proposition 25). In this latter approach, the input bivariate polynomials are formally sheared using an additional variable that parameterizes a generic linear form, and the resultant of these (trivariate) polynomials is computed. The polynomials of the RUR are then expressed (and computed) as combinations of this resultant and its partial derivatives, specialized at the value $a$ associated with the given linear separating form.

Here, we combine in essence these two approaches and compute a RUR for every system of the triangular decomposition of the sheared input system. However, in order to obtain the claimed complexities,

we do not compute a RUR of every triangular system using the approach of [BLPR15*]. Instead, Algorithm 6 works as follows. First, we compute the triangular decomposition of the sheared input system as in [GVEK96]. We then observe that the radical ideals of the triangular systems can easily be obtained due to the generic position enforced by the shear. Using the simple structure of these radical ideals, we derive formulas for their RURs (Lemma 29). For complexity issues, we do not use these formulas to directly compute RURs over $\mathbb{Q}$ but we use instead a multi-modular approach. For that purpose, we derive tight bounds on the size of the RUR coefficients and the luckiness certificate of the triangular decomposition introduced in Section 2.2.2 to select the primes.

Finally, we show how our deterministic algorithm can be transformed into a probabilistic Las Vegas one of expected bit complexity $\widetilde{O}_B(d^5+d^4\tau)$. In order to obtain this complexity, we cannot compute the triangular decomposition as described above. Instead, we show that we can only compute the coefficients of these triangular systems that are needed for obtaining their radicals, within the targeted bit complexity. We also choose randomly the primes in the multi-modular computation described above. This can be done in a Las Vegas setting because we show that we can choose good primes with sufficiently high probability and that we can check whether the primes are good within the targeted complexity.

### RUR definition and properties

**Definition 24** ([Rou99, Definition 3.3]). *Let $I \subset \mathbb{Q}[x,y]$ be a zero-dimensional ideal, $V(I) = \{\sigma \in \mathbb{C}^2, v(\sigma) = 0, \forall v \in I\}$ its associated variety, and let $(x,y) \mapsto x + ay$ be a linear form with $a$ in $\mathbb{Q}$. The* RUR*-candidate of $I$ associated to $x+ay$ (or simply, to $a$), denoted $\mathrm{RUR}_{I,a}$, is the following set of four univariate polynomials in $\mathbb{C}[t]$*

$$\begin{aligned} f_{I,a}(t) &= \prod_{\sigma \in V(I)} (t - x(\sigma) - ay(\sigma))^{\mu_I(\sigma)} \\ f_{I,a,v}(t) &= \sum_{\sigma \in V(I)} \mu_I(\sigma) v(\sigma) \prod_{\varsigma \in V(I), \varsigma \neq \sigma} (t - x(\varsigma) - ay(\varsigma)), \quad \text{for } v \in \{1, x, y\} \end{aligned} \tag{2.2}$$

*where, for $\sigma$ in $V(I)$, $\mu_I(\sigma)$ denotes the multiplicity of $\sigma$ in $I$. If $(x,y) \mapsto x+ay$ is injective on $V(I)$, we say that the linear form $x+ay$ separates $V(I)$ (or is separating for $I$) and $\mathrm{RUR}_{I,a}$ is called a* RUR *(the* RUR *of $I$ associated to $a$).*

The following proposition states fundamental properties of RURs, which are all straightforward from the definition except for the fact that the RUR polynomials have rational coefficients [Rou99, Theorem 3.1].

**Proposition 25** ([Rou99, Theorem 3.1]). *If $I \subset \mathbb{Q}[x,y]$ is a zero-dimensional ideal and $a$ in $\mathbb{Q}$, the four polynomials of the* RUR*-candidate $\mathrm{RUR}_{I,a}$ have rational coefficients. Furthermore, if $x+ay$ separates $V(I)$, the following mapping between $V(I)$ and $V(f_{I,a}) = \{\gamma \in \mathbb{C}, f_{I,a}(\gamma) = 0\}$*

$$\begin{array}{ccc} V(I) & \to & V(f_{I,a}) \\ (\alpha, \beta) & \mapsto & \alpha + a\beta \\ \left(\dfrac{f_{I,a,x}}{f_{I,a,1}}(\gamma), \dfrac{f_{I,a,y}}{f_{I,a,1}}(\gamma)\right) & \leftarrow\!\shortmid & \gamma \end{array}$$

*is a bijection, which preserves the real roots and the multiplicities.*

Next, we define a RUR decomposition of an ideal.

**Definition 26.** *Let $I \subset \mathbb{Q}[x,y]$ be a zero-dimensional ideal, $V(I) = \{\sigma \in \mathbb{C}^2, v(\sigma) = 0, \forall v \in I\}$ its associated variety, and let $(x,y) \mapsto x+ay$ be a linear form with $a$ in $\mathbb{Q}$. A RUR-candidate decomposition of $I$ is a sequence of RUR-candidates, associated to $x+ay$, of ideals $I_i \supseteq I$, $i \in \mathcal{I}$ such that $V(I)$ is the disjoint union of the varieties $V(I_i)$, $i \in \mathcal{I}$. If $x+ay$ separates $V(I_i)$ for all $i \in \mathcal{I}$, the RUR-candidate decomposition is a RUR decomposition of $I$.*

### RUR bitsize

The RUR algorithm presented in [BLPR15*] is based on a formal shear using an additional variable that parameterizes a generic linear form, and the resultant of (trivariate) polynomials. The polynomials of the RUR are then expressed (and computed) as combinations of this resultant and its partial derivatives, specialized at the value associated with the given linear separating form. Even if this approach is not

**Algorithm 6** RUR decomposition

**Input:** $P, Q$ coprime in $\mathbb{Z}[x, y]$ of degree at most $d$ and bitsize at most $\tau$.
**Output:** RUR decomposition of $\{P, Q\}$ of total bitsize $\widetilde{O}(d^4 + d^3\tau)$.

1: Compute a separating form $x + ay$ for $\{P, Q\}$ with $a \in \mathbb{Z}$ of bitsize $O(\log d)$ such that the leading coefficients of $P(t - ay, y)$ and $Q(t - ay, y)$ with respect to $y$ are coprime (see Theorem 22).
2: Compute $\widetilde{P}(t, y) = P(t - ay, y)$ and $\widetilde{Q}(t, y) = Q(t - ay, y)$, and let $\tilde{d}$ and $\tilde{\tau}$ be their maximum degree and bitsize.
3: Compute $\{T_i\}_{i \in \mathcal{I}}$ = Algorithm 1$(\widetilde{P}, \widetilde{Q})$.
Recall that $T_i = \{A_i(t), B_i(t, y)\}$ with $B_i(t, y) = \operatorname{sres}_i(\widetilde{P}, \widetilde{Q})(t)\, y^i + \operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})(t)\, y^{i-1} + \cdots$
Let $\widehat{T}_i = \langle A_i, i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q})\, y + \operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})\rangle$ be the radical ideal of $T_i$.
4: Let $K = \lceil C(\tilde{d}^2 + \tilde{d}\tilde{\tau}) \log^k(\tilde{d}\tilde{\tau})\rceil$ be an integer that bounds from above the bitsize of the coefficients of the RURs of the systems $\widehat{T}_i$ (see Proposition 27 and subsequent discussion) and let $\Pi$ = Algorithm 2$(\widetilde{P}, \widetilde{Q})$. Compute the set $\mathcal{L}$ of the $2K$ first prime numbers that are larger than $\tilde{d}$ and that do not divide $\Pi$. Let $\Pi_{\mathcal{L}}$ be the product of all primes in $\mathcal{L}$.
5: **for all** $i$ in $\mathcal{I}$ **do**
6: **for all** $\mu$ in $\mathcal{L}$ **do**
7: Compute $\phi_\mu(\widehat{T}_i)$ by reducing modulo $\mu$ the polynomials $A_i$, $\operatorname{sres}_i(\widetilde{P}, \widetilde{Q})$ and $\operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$.
8: Compute $\mathrm{RUR}_i^\mu$ the RUR in $\mathbb{F}_\mu$ of $\phi_\mu(\widehat{T}_i)$ associated to the separating form $(t, y) \mapsto t$ (see Lemma 29).
9: Lift $\{\mathrm{RUR}_i^\mu\}_{\mu \in \mathcal{L}}$ to $\mathrm{RUR}_i^{\Pi_{\mathcal{L}}}$ in $\mathbb{Z}/\Pi_{\mathcal{L}}\mathbb{Z}$ using the Chinese Remainder Algorithm.
10: Compute $\mathrm{RUR}_i^{\mathbb{Q}}$, the RUR in $\mathbb{Q}$ of $\widehat{T}_i$ associated to the separating form $(t, y) \mapsto t$, with a rational reconstruction from $\mathrm{RUR}_i^{\Pi_{\mathcal{L}}}$.
11: **return** the image of $\mathrm{RUR}_i^{\mathbb{Q}}$, $i \in \mathcal{I}$, through the reverse shearing from $(t, y)$ to $(x, y)$ (see Lemma 31).

optimal for computing the RUR (since it computes a trivariate resultant), it enables to derive tight bounds on its bitsize which is critical in view of the lifting step of modular algorithms. The following proposition recalls an upper bound on the bitsize of a RUR of an ideal containing two coprime polynomials $P$ and $Q$, that is a RUR parameterizing a subset of the solutions of the system $\{P, Q\}$.

**Proposition 27** ([BLPR15*, Proposition 28])**.** *Let $P$ and $Q$ in $\mathbb{Z}[x, y]$ be two coprime polynomials of total degree at most $d$ and maximum bitsize $\tau$, let $a$ be a rational of bitsize $\tau_a$, and let $J$ be any ideal of $\mathbb{Z}[x, y]$ containing $P$ and $Q$. The polynomials of the* RUR*-candidate of $J$ associated to $a$ have degree at most $d^2$ and bitsize in $\widetilde{O}(d^2\tau_a + d\tau)$.*

Note that according to Theorem 22, a separating form $x + ay$ can be computed with an integer $a$ of bitsize $O(\log d)$ and the bound in Proposition 27 becomes $\widetilde{O}(d^2 + d\tau)$. In addition, even if Proposition 27 only states an asymptotic upper bound, an explicit upper bound $C(d^2 + d\tau)\log^k(d\tau)$ with $C, k \in \mathbb{Z}$ can be obtained from straightforward, although unappealing, computations following the proof of that proposition. Indeed, this proof is based on Hadamard's inequality and Mignotte's lemma, which both state explicit bounds. Such an explicit upper bound will be used in the lifting step of Algorithm 6.

Proposition 27 applies to the RURs of our RUR decomposition and yields the following bound on the total bitsize of any RUR decomposition which is used in Algorithm 6.

**Corollary 28** ([BLM+16*, Corollary 38])**.** *Let $P$ and $Q$ in $\mathbb{Z}[x, y]$ be two coprime polynomials of total degree at most $d$ and maximum bitsize $\tau$, and let $a$ be a rational of bitsize $\tau_a$. The sum of the bitsizes of all coefficients of any RUR-candidate decomposition of $\langle P, Q\rangle$, associated to $x + ay$, is in $\widetilde{O}(d^4\tau_a + d^3\tau)$.*

### Decomposition algorithm

Algorithm 6 computes a RUR decomposition of a zero-dimensional system $\{P, Q\}$, by first computing a separating form $x + ay$ as shown in Section 2.2.3 (Line 1). We then use this separating form to shear the system in generic position (Line 2) and compute the radical of a triangular decomposition of this system (Line 3). Then, using a multimodular approach, we compute RURs of each of the resulting radical systems (Lines 4–10) and return these RURs after a shear back (Line 11).

The next lemma states formulas for the RURs of the radical ideals $\widehat{T}_i$ computed in Line 8.

**Lemma 29** ([BLM+16*, Lemma 40]). *Let $\mathbb{F}$ be a field, $A, B_0, B_1$ be three polynomials in $\mathbb{F}[t]$ and let $I = \langle A(t), B_1(t)\,y + B_0(t)\rangle$ be an ideal such that $A$ is squarefree and coprime with $B_1$. The linear form $(t, y) \mapsto t$ is separating for that ideal and its associated RUR is given by*[8]

$$f_I = \tfrac{A}{\mathrm{Lc}(A)} \qquad f_{I,1} = f_I' \qquad f_{I,t} = t\,f_{I,1} \text{ rem } A \qquad f_{I,y} = -B_0\,U\,f_{I,1} \text{ rem } A$$

*where $U \in \mathbb{F}[t]$ is the inverse of $B_1$ modulo $A$, defined by Bézout's identity $UB_1 + VA = 1$ and where $f$ rem $g$ denotes the remainder of the Euclidean division of $f$ by $g$.*

Even if the bitsize of the RUR of $\widehat{T}_i$ is known to be in $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau}) = \widetilde{O}(d^2 + d\tau)$ (Proposition 27), the naive computation of these RURs using the above formulas over the rationals would suffer from large intermediate bitsizes.[9] To overcome this difficulty, we use in Algorithm 6 a classical multimodular technique, which consists in first computing the polynomials modulo a set of primes whose product is larger than the bitsize of the output coefficients, then lifting the result using the Chinese Remainder Algorithm and finally performing a rational reconstruction. However, to output a correct result, this technique requires that, for any selected prime $\mu$, the formulas of Lemma 29 commute with the reduction modulo $\mu$. We show in Lemma 30 how to satisfy this requirement using the luckiness certificate output by Algorithm 2. This lemma is instrumental for the proof of correctness of Algorithm 6.

**Lemma 30** ([BLM+16*, Lemma 41]). *Let $\mu > i$ be a prime that does not divide $\Pi$ (defined in Line 4 of Algorithm 6). The ideals $\widehat{T}_i$ and $\phi_\mu(\widehat{T}_i)$ satisfy the hypotheses of Lemma 29. In particular, the linear form $(t, y) \mapsto t$ is separating for both ideals. For this linear form, the RUR of $\phi_\mu(\widehat{T}_i)$ is equal to the reduction modulo $\mu$ of the RUR of $\widehat{T}_i$.*

Since the RURs are computed with respect to the coordinates $(t, y)$ in Line 8, a reverse shearing is computed in Line 10 using the following formulas.

**Lemma 31** ([BLM+16*, Lemma 42]). *Let $\{f_I, f_{I,1}, f_{I,t}, f_{I,y}\}$ be the RUR[8] of an ideal $I$ in $\mathbb{Q}[t, y]$ associated to the separating linear form $(t, y) \mapsto t$. Let $J$ in $\mathbb{Q}[x, y]$ be the image of $I$ through the mapping $(t, y) \mapsto (x = t - ay, y)$. The linear form $(x, y) \mapsto x + ay$ is separating for $J$ and its associated RUR is given by*

$$f_{J,a} = f_I, \qquad f_{J,a,1} = f_{I,1}, \qquad f_{J,a,x} = f_{I,t} - a\,f_{I,y}, \qquad f_{J,a,y} = f_{I,y}.$$

We finally state the complexity of Algorithm 6.

**Theorem 32** ([BLM+16*, Theorem 45]). *Let $P, Q$ in $\mathbb{Z}[x, y]$ be of total degree at most $d$ and maximum bitsize $\tau$. Algorithm 6 computes, with $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case, a RUR decomposition of $\{P, Q\}$ of total bitsize $\widetilde{O}(d^4 + d^3\tau)$.*

### Las Vegas algorithm

We show here that the algorithm presented above for computing a RUR decomposition can be transformed into an efficient Las Vegas algorithm with a better complexity.

**Theorem 33** ([BLM+16*, Theorem 46]). *Let $P, Q$ in $\mathbb{Z}[x, y]$ be of total degree at most $d$ and maximum bitsize $\tau$. Algorithm 6' computes, with $\widetilde{O}_B(d^5 + d^4\tau)$ bit operations on average, a RUR decomposition of $\{P, Q\}$ of total bitsize $\widetilde{O}(d^4 + d^3\tau)$.*

Algorithm 6', our Las Vegas version of Algorithm 6, is obtained from the latter with only three modifications. First, in Line 2, we use the Las Vegas version of our algorithm for computing a separating linear form for $\{P, Q\}$, described in Section 2.2.3.

Second, in Line 3, we modify the way we compute the radicals $\widehat{T}_i$ of the ideals $T_i$ output by Algorithm 1$(\widetilde{P}, \widetilde{Q})$. We still use the formula $\widehat{T}_i = \langle A_i, i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})\,y + \mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})\rangle$ for computing these radical ideals, but instead of computing the $T_i$ with Algorithm 1, we show that the subresultant coefficients $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ and $\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$ can be computed more efficiently.

[8] We omit in the subscript of the polynomials of the RUR the reference to the parameter, 0, of the separating form $(t, y) \mapsto t + 0y$.

[9] More precisely, the computation of the RURs using the formulas of Lemma 29 over the rationals would require $\widetilde{O}_B(d^8 + d^7\tau)$ bit operations for each triangular system and $\widetilde{O}_B(d^9 + d^8\tau)$ for all of them. This bit complexity corresponds roughly to the cost of multiplications and divisions involving the inverse of $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ rem $A_i$, which is a polynomial of degree $O(d^2)$ and bitsize in $\widetilde{O}(d^4 + d^3\tau)$.

**Algorithm 6'** RUR decomposition – Las Vegas version

**Input:** $P, Q$ coprime in $\mathbb{Z}[x, y]$ of degree at most $d$ and bitsize at most $\tau$.
**Output:** RUR decomposition of $\{P, Q\}$ of total bitsize $\widetilde{O}(d^4 + d^3\tau)$.

1: Compute a separating form $x + ay$ for $\{P, Q\}$ with $a \in \mathbb{Z}$ of bitsize $O(\log d)$ such that the leading coefficients of $P(t - ay, y)$ and $Q(t - ay, y)$ with respect to $y$ are coprime (see Theorem 23).
2: Compute $\widetilde{P}(t, y) = P(t - ay, y)$ and $\widetilde{Q}(t, y) = Q(t - ay, y)$, and let $\tilde{d}$ and $\tilde{\tau}$ be their maximum degree and bitsize.
3: Compute the coefficients $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})(t)$ of subresultant sequence of $\widetilde{P}$ and $\widetilde{Q}$ with respect to $y$ and, for $i$ such that $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}) \not\equiv 0$, compute $\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})(t)$ (see Corollary 35).
Compute the polynomials $A_i(t)$, $i \in \mathcal{I}$, of the triangular decomposition of $\widetilde{P}$ and $\widetilde{Q}$ following Algorithm 1.
Let $\widehat{T}_i = \langle A_i, i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})\, y + \mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})\rangle$, $i \in \mathcal{I}$, be the radicals of the ideals output by Algorithm 1$(\widetilde{P}, \widetilde{Q})$.
4: Let $K = \lceil C(\tilde{d}^2 + \tilde{d}\tilde{\tau}) \log^k(\tilde{d}\tilde{\tau})\rceil$ be an integer that bounds from above the bitsize of the coefficients of the RURs of the systems $\widehat{T}_i$ (see Proposition 27 and subsequent discussion). Let $U = 8K$ and $\mathcal{L} = \emptyset$.
5: **repeat**
6: Double $U$, choose uniformly at random $8K$ primes in $[1, U]$, and let $\mathcal{P}$ be the resulting set.
7: For all $i \in \mathcal{I}$, $\mu \in \mathcal{P}$, reduce $A_i$ and $i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ modulo $\mu$ (using remainder trees).
8: Add in $\mathcal{L}$ the $\mu \in \mathcal{P}$ such that, $\forall i$, $\phi_\mu(A_i)$ is squarefree and coprime with $\phi_\mu(i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}))$.
9: **until** $\mathcal{L}$ contains at least $2K$ distinct primes.
10: **return** The image of $\mathrm{RUR}_i^{\mathbb{Q}}$, the RUR of $\widehat{T}_i$, $i \in \mathcal{I}$, through the reverse shearing from $(t, y)$ to $(x, y)$, as in Algorithm 6, Lines 5-11.

Third, we modify the way we compute in Algorithm 6, Line 4, a set $\mathcal{L}$ of $2K$ prime numbers $\mu > \tilde{d}$ that do not divide $\Pi =$ Algorithm 2$(\widetilde{P}, \widetilde{Q})$. Here, in Line 9, we weaken the constraints on these primes and we avoid, in particular, computing $\Pi$.

A key step of Algorithm 6' is the computation of the coefficients $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ and the computation of $\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$ when $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}) \not\equiv 0$. We show that all these coefficients can be computed in $\widetilde{O}_B(d^4\tau)$ bit complexity in Corollary 35. This result generalizes [vzGG13, Corollary 11.18] to the case where one wants to compute the $k$ terms of greater degrees in the sequence of remainders in the Euclidean algorithm.

Given two polynomials $P, Q \in \mathbb{F}[y]$ such that $\deg(P) \geqslant \deg(Q)$, we denote by $r_j$ and $q_j$ the polynomials appearing in the Euclidean algorithm such that $r_0 = P, r_1 = Q$ and $r_{i-1} = q_i r_i + r_{i+1}$. For any polynomial $P \in \mathbb{F}[y]$ and any integer $n$, we denote by $P_{|n}$ the coefficient of its term of degree $\deg(P) - n$, if any, and 0 otherwise. It follows that $r_{i|j}$ denotes the coefficient of the term of $r_i$ of degree $\deg(r_i) - j$.

**Theorem 34** ([BLM+16*, Theorem 51]). *Let $k$ be an integer and $P, Q \in \mathbb{F}[y]$ be two polynomials with $d = \deg(P) \geqslant \deg(Q)$. We can compute, for all $0 \leqslant j \leqslant k$ and for all the remainders $r_i$ appearing in the Euclidean algorithm, the coefficients $r_{i|j}$ in $O(k^2 d + M(d)\log d)$ arithmetic operations, where $M(d)$ is the complexity of the multiplication of degree $d$ polynomials.*

We can now state the corollary that we use in the analysis of Algorithm 6'.

**Corollary 35** ([BLM+16*, Corollary 52]). *Let $P, Q \in \mathbb{Z}[x, y]$ be of degree at most $d$ with coefficients of bitsize at most $\tau$. We can compute in $\widetilde{O}_B(d^4\tau)$ bit operations in the worst case the sequence of all subresultant coefficients $\mathrm{sres}_i(P, Q)$ and, for $i$ such that $\mathrm{sres}_i(P, Q) \not\equiv 0$, the coefficients $\mathrm{sres}_{i,i-1}(P, Q)$.*

### 2.2.5 Computing isolating boxes from a RUR decomposition

We introduce Algorithm 7 computing boxes isolating the complex solutions from a RUR. By definition, the RUR of an ideal $I$ defines a mapping between the roots of a univariate polynomial and the solutions of $I$. A RUR is hence naturally designed to compute isolating boxes using univariate isolation and interval evaluation.

let $L$ be an arbitrary positive integer. We define $\tilde{x} \in \mathbb{Q} + i\,\mathbb{Q}$ to be a $L$-bit approximation of $x$ if $\tilde{x}$ is of the form $\tilde{x} = (m_{\Re} + i\, m_{\Im}) \cdot 2^{-L-2}$, with $m_{\Re}, m_{\Im} \in \mathbb{Z}$, and $|x - \tilde{x}| < 2^{-L}$. Notice that an $L$-bit

**Algorithm 7** Isolating boxes for the solutions of $P = Q = 0$

**Input:** $P, Q$ coprime in $\mathbb{Z}[x, y]$ of degree at most $d$ and bitsize at most $\tau$ and $(\mathrm{RUR}_i)_{i\in\mathcal{I}} = (f_i, f_{i,1}, f_{i,x}, f_{i,y})_{i\in\mathcal{I}}$ the RUR decomposition of $\{P, Q\}$ as computed computed by Algorithm 6 or 6'.
**Output:** Isolating boxes for all solutions of $P = Q = 0$.

1: $f = \prod_{i\in\mathcal{I}} f_i$
2: Compute isolating disks $D_\gamma \subset \mathbb{C}$ for all complex roots $\gamma$ of $f$.
3: $S = \{(\gamma, \gamma') \mid \gamma \text{ and } \gamma' \text{ distinct roots of } f\}$
4: $L = 1$
5: **repeat**
6: $\quad L = 2L$
7: $\quad$ **for** $i \in \mathcal{I}$ **do**
8: $\quad\quad$ For all roots $\gamma$ of $f_i$, compute $L$-bit approximations $\tilde{\sigma}_{\gamma,x}$ and $\tilde{\sigma}_{\gamma,y}$ of $\sigma_{\gamma,x} = \frac{f_{i,x}(\gamma)}{f_{i,1}(\gamma)}$ and $\sigma_{\gamma,y} = \frac{f_{i,y}(\gamma)}{f_{i,1}(\gamma)}$, respectively .
9: **until** for all pairs $(\gamma, \gamma') \in S$, $|\tilde{\sigma}_{\gamma,x} - \tilde{\sigma}_{\gamma',x}| > 2^{-L+2}$ or $|\tilde{\sigma}_{\gamma,y} - \tilde{\sigma}_{\gamma',y}| > 2^{-L+2}$
10: **return** $\{B(\tilde{\sigma}_{\gamma,x}) \times B(\tilde{\sigma}_{\gamma,y}) \mid \gamma \text{ root of } f\}$

approximation $\tilde{x} = (m_\Re + i\, m_\Im) \cdot 2^{-L-2}$ of some point $x \in \mathbb{C}$ naturally defines a box

$$B(\tilde{x}) = \frac{[m_\Re - 4, m_\Re + 4]}{2^{L+2}} + i \cdot \frac{[m_\Im - 4, m_\Im + 4]}{2^{L+2}} \subset \mathbb{C} \tag{2.3}$$

of width $2^{-L+1}$ in $\mathbb{C}$ that contains $x$. We extend the definition of an $L$-bit approximation $\tilde{x}$ of a point $x \in \mathbb{C}$ to that of an $L$-bit approximation $(\tilde{x}, \tilde{y})$ of a point $(x, y) \in \mathbb{C}^2$ by requiring that both $\tilde{x}$ and $\tilde{y}$ are $L$-bit approximations of $x$ and $y$, respectively.

Algorithm 7 computes disjoint isolating boxes for the solutions $\sigma \in \mathbb{C}^2$ of a zero-dimensional system $P = Q = 0$. More specifically, for a given $L$, we first compute $L$-bit approximations $\tilde{\sigma}_{i,j}$ of the solutions $\sigma_{i,j} = (x_{i,j}, y_{i,j})$, $1 \leqslant j \leqslant d_i = \deg f_i$, of each factor $\mathrm{RUR}_i = (f_i, f_{i,1}, f_{i,x}, f_{i,y})$ in the RUR decomposition $(\mathrm{RUR}_i)_{i\leqslant d}$ of $\{P, Q\}$ as computed by Algorithm 6 or 6'. This is achieved by first computing sufficiently small isolating disks for the roots $\gamma_{i,j}$ of the univariate polynomial $f_i \in \mathbb{Z}[x]$ in $\mathrm{RUR}_i$, and then evaluating the fractions $\frac{f_{i,x}}{f_{i,1}}$ and $\frac{f_{i,y}}{f_{i,1}}$ at the roots $\gamma_{i,j}$ to an absolute error less than $2^{-L}$. From the corresponding $L$-bit approximations $\tilde{x}_{i,j}$ and $\tilde{y}_{i,j}$, we can then derive boxes $B_{i,j} = B(\tilde{\sigma}_{i,j}) = B(\tilde{x}_{i,j}) \times B(\tilde{y}_{i,j}) \subset \mathbb{C}^2$ of width $2^{-L+1}$ containing all solutions of $\mathrm{RUR}_i$. If, for all $i$ and $j$, the boxes $B_{i,j}$ do not overlap, then they are already isolating for the solutions of $P = Q = 0$. Otherwise, we have to increase $L$ until the boxes do not overlap.

Using amortized bounds for the roots of polynomials and fast multipoint evaluation, the following theorem analyzes the complexity of the isolation of a system $\{P, Q\}$ from a RUR decomposition.

**Theorem 36** ([BLM+16*, Theorem 61]). *Let $P, Q \in \mathbb{Z}[x, y]$ be coprime polynomials of degree at most $d$ with integer coefficients of bitsize at most $\tau$. Algorithm 7 computes isolating boxes for all complex solutions of $P = Q = 0$ using $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations.*

In order to isolate only the real solutions of $P = Q = 0$, it suffices to iterate in Algorithm 7 over the real roots of $f$ since the separating form $(x, y) \mapsto x + ay$ is a one-to-one mapping between the real solutions of $P = Q = 0$ and the real roots of $f$ (see Proposition 25). Note that, in this case, it is preferable to consider a dedicated real root isolation method in Step 2 for computing the real roots of $f$.

## 2.3 Practical and efficient RUR decomposition algorithm for ISOTOP

As explained in Section 3.3, the ISOTOP algorithm is designed to output the topology of a curve in its original coordinate system. This implies that the potential vertical asymptotes of the curve have to be explicitly processed. Section 2.3.1 details our contribution to extend the triangular decomposition in the presence of asymptotes [LPR17*]. The other key feature of ISOTOP is the way it computes the local topology at critical points with the multiplicity in fibers. Section 2.3.2 details our method to isolate the critical points of the curve using a triangular decomposition and RURs that computes multiplicities in fibers [BLPR11*, Bou14].

### 2.3.1 Triangular decomposition in presence of asymptotes

The classical triangular decomposition of two coprime polynomials $P(x,y)$ and $Q(x,y)$ via subresultants, Algorithm 1, assumes that the leading coefficients $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime in $\mathbb{Q}[x]$ in order to apply the specialization property of subresultants (Lemma 2). In Isotop, we want to apply this algorithm for the critical points of a curve defined by a polynomial $H(x,y)$, that is on the system $\{H, \frac{\partial H}{\partial y}\}$. In the case where the curve has vertical asymptotes, that is the leading coefficient $\mathrm{Lc}_y(H)$ is not a constant, this assumption is not satisfied. We thus extend the triangular decomposition in a recursive way, taking care of degenerate cases and without increasing its complexity.

#### Previous work and contribution

Let two $P = \sum_{i=0}^{p} a_i(x)y^i$ and $Q = \sum_{i=0}^{q} b_i(x)y^i$ be two polynomials in $\mathbb{Z}[x,y]$. The values $\alpha$ such that $a_p(\alpha)$ and $b_q(\alpha)$ both vanish are exactly the $x$-coordinates of the common vertical asymptotes of the curves defined by $P$ and $Q$, which we refer to as the common vertical asymptotes of the polynomials, for simplicity. In the general case when $P$ and $Q$ (may) admit common vertical asymptotes, the natural solution for computing a (full) triangular decomposition is to first use Algorithm 1 to compute the triangular decomposition of the solutions of $\{P,Q\}$ that do not lie on common vertical asymptotes (this can be done by removing from the resultant of $P$ and $Q$ the solutions corresponding to these asymptotes, i.e., $\gcd(a_p, b_q)$). Then, the triangular decomposition algorithm is called recursively on $P$ and $Q$ reduced modulo $\gcd(a_p, b_q)$. The drawback of this approach is that the number of recursive calls may be linear in the minimum of the degrees in $x$ and $y$ of the input polynomials (it may happen that only one vertical asymptote is "handled" at each recursive call) and that the bitsize of the coefficients of the reduction of $P$ and $Q$ increases at each recursive call.

Li et al. [LMMRS11] proposed a simple variation on this natural algorithm where, instead of considering $P$ and $Q$ modulo $\gcd(a_p, b_q)$ at the first recursive call (and similarly for the other calls), they simply remove the leading terms $a_p y^p$ and $b_q y^q$ of $P$ and $Q$. However, they did not provide a complexity analysis of their algorithm.

Here, we present and analyze a variation on this algorithm. First, we solve some issues in Li et al.'s algorithm in which, during the recursion, the reduced versions of $P$ and $Q$ may not define a zero-dimensional system (and also that they may be both univariate). Second, we carefully arrange our computations in a way that is critical for the analysis of our complexity bounds. In particular, (i) we only compute the principal subresultant sequence (instead of the full polynomial sequence) in order to compute only the relevant subresultant polynomials and (ii) in the recursion, we only compute the decomposition above the roots that define asymptotes for all the preceding polynomials.

In our modified algorithm, the number of recursive calls may still be linear in $d$ but we show that the complexity of the overall recursive algorithm is the same as the complexity of the non-recursive algorithm (with no vertical asymptotes), that is $\widetilde{O}(d^4)$ for the arithmetic complexity and $\widetilde{O}_B(d^6 + d^5\tau)$ for the bit complexity. More precisely, we prove an arithmetic complexity in $\widetilde{O}(d_x d_y^3 + d_x^2 d_y^2)$ and a bit complexity in $\widetilde{O}_B(d_x^3 d_y^3 + (d_x^2 d_y^3 + d_x d_y^4)\tau)$ in the worst case where $d_x$ and $d_y$ bound the degrees of $P$ and $Q$ in $x$ and $y$, respectively. We also prove that the total bitsize of the decomposition is in $\widetilde{O}((d_x^2 d_y^3 + d_x d_y^4)\tau)$. This implies in particular that, unless improving this upper bound, there is not much room for improving the bit complexity of the computation of the triangular decomposition. This also shows that, when there is a disparity between degrees $d_x$ and $d_y$, the ordering of variables in the triangular decomposition impacts the complexity of the algorithm and of the output.

It is worthwhile to mention that in the general context of solving systems, one standard approach is to shear the coordinate system, that is to apply a change of coordinates of the form $(x,y) \mapsto (x + ay, y)$, and to compute a triangular decomposition of the sheared system. We have seen, in Section 2.2, that this approach gives the best complexity bounds for solving bivariate systems. On the hand, this approach does not solve the given problem of computing a triangular decomposition of the input system since it computes a triangular decomposition of another system. We also show that our improved decomposition algorithm naturally preserves multiplicities in fibers (Definition 8) which are important invariants for Isotop.

**Lemma 37.** *The multiplicity of any solution in the triangular systems output by Algorithm 9 is its multiplicity in its fiber with respect to the system* $\{P,Q\}$.

**Algorithm 8: Triangular decomposition of $\{P,Q\}$ away from their common vertical asymptotes and such that $A$ vanishes.** Algorithm 8 takes as input $P, Q \in \mathbb{Z}[x,y]$ and a univariate polynomial

**Algorithm 8** Triangular decomposition away from asymptotes

**Input:** $P, Q$ in $\mathbb{Z}[x, y]$ and $A$ squarefree in $\mathbb{Z}[x]$ such that system $\{P, Q, A\}$ is zero-dimensional.
**Output:** A set of regular triangular systems, each of the form $\{U(x), V(x, y)\}$ with coefficients in $\mathbb{Z}$, whose solutions are those of $\{P, Q, A\}$ that do not lie on a common vertical asymptote of $P$ and $Q$.

**if** $A \equiv 0$ **then**
  $R(x) = \overline{\mathrm{Res}_y(P, Q)}$, $B(x) = \overline{\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$, $F = R/B$
**else**
  $R(x) = \mathrm{Res}_y(P, Q)$, $B(x) = \gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$, $F = \frac{\gcd(R,A)}{\gcd(B,A)}$
**if** neither $P$ nor $Q$ is in $\mathbb{Z}[x]$ **then**
  If needed, exchange $P$ and $Q$ so that $\mathrm{d}_y(Q) \leqslant \mathrm{d}_y(P)$
  Compute $\{\mathrm{sres}_{y,i}(P, Q)\}_{i=0,\ldots,\mathrm{d}_y(Q)}$, the principal subresultant sequence of $P$ and $Q$ w.r.t. $y$
  $G_0 = F$, $\mathcal{TD} = \emptyset$
  **for** $i = 1$ **to** $\mathrm{d}_y(Q)$ **do**
    $G_i = \gcd(G_{i-1}, \mathrm{sres}_{y,i}(P, Q))$
    $F_i = G_{i-1}/G_i$
    **if** $\mathrm{d}_x(F_i) > 0$ **then**
      Compute $\mathrm{Sres}_{y,i}(P, Q)$
      $\mathcal{TD} = \mathcal{TD} \cup \{F_i, \mathrm{Sres}_{y,i}(P, Q)\}$
  **return** $\mathcal{TD}$
**else if** $P$ and $Q$ are in $\mathbb{Z}[x]$ **then**
  **return** $\emptyset$
**else** ▷ Assume wlog that $P$ is in $\mathbb{Z}[x]$ (and $Q$ is not)
  **return** $\{F, Q\}$

**Algorithm 9** Complete triangular decomposition

**Input:** $P, Q$ in $\mathbb{Z}[x, y]$ defining a zero-dimensional system.
**Output:** A set of regular triangular systems, each of the form $\{U(x), V(x, y)\}$ with coefficients in $\mathbb{Z}$, whose sets of solutions are disjoint and are exactly those of $\{P, Q\}$. The multiplicity of any solution in these triangular systems is the multiplicity of the solution in its fiber with respect to the system $\{P, Q\}$ (see Definition 8).

$A = 0$, $B = B_{new} = \overline{\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$, $\mathcal{TD} = \emptyset$
**repeat**
  **if** $\mathrm{d}_x(A) \neq 0$ **then**
    $\mathcal{TD} = \mathcal{TD} \cup$ Algorithm 8$(P, Q, A)$
  $P = P - \mathrm{Lc}_y(P) y^{\deg_y(P)}$, $Q = Q - \mathrm{Lc}_y(Q) y^{\deg_y(Q)}$
  $B = B_{new}$
  $B_{new} = \gcd(B, \mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$
  $A = \frac{B}{B_{new}}$
**until** $\deg(B) = 0$
**return** $\mathcal{TD}$

$A \in \mathbb{Z}[x]$ such that system $\{P, Q, A\}$ is zero dimensional and it computes a set of triangular systems whose solutions are the solutions of $\{P, Q, A\}$ that do not lie on a common vertical asymptote of the curves defined by $P$ and $Q$. Considering the calls to Algorithm 8 made by Algorithm 9, Algorithm 8 will first run with $A \equiv 0$ and compute a triangular decomposition of the solutions away from the common vertical asymptotes of $P$ and $Q$; then Algorithm 8 will be called with $A$ encoding a subset of these common vertical asymptotes and two polynomials that coincide with $P$ and $Q$ on these asymptotes. Algorithm 8 is essentially Algorithm 1 in the case where $\{P, Q\}$ is zero dimensional, $P$ and $Q$ do not have any common vertical asymptote, and $A \equiv 0$.

The projection onto the $x$-axis of the solutions of system $\{P, Q\}$ that do not lie on a common vertical asymptote of the curves defined by $P$ and $Q$ are exactly the roots of the resultant of $P$ and $Q$ with respect to $y$ divided by the gcd of the leading coefficients of $P$ and $Q$ with respect to $y$. We actually consider the squarefree parts of these polynomials, $\overline{\mathrm{Res}_y(P, Q)}$ and $\overline{\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$, which is critical for our property on the multiplicity of the solutions in their fibers (Lemma 37). In order to restrict the set of

solutions of $\{P, Q\}$ that do not lie on a common vertical asymptote to those where $A$ vanishes, we consider the gcd of $\frac{\overline{\mathrm{Res}_y(P,Q)}}{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))}$ with $A$. However, this does not work when $\mathrm{Res}_y(P, Q) \equiv 0$, that is when $\{P, Q\}$ is not zero dimensional (and in generic position). We thus consider instead $F = \frac{\overline{\mathrm{Res}_y(P,Q)}}{\overline{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))}}$ when $A \equiv 0$ and, otherwise, $F = \frac{\overline{\gcd(\mathrm{Res}_y(P,Q),A)}}{\overline{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q),A)}}$, which is equal to $\frac{\gcd(\mathrm{Res}_y(P,Q),A)}{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q),A)}$ since $A$ is squarefree. Then, the roots of $F$ are the projections (on $x$) of the solutions of $\{P, Q, A\}$ that are not on the common vertical asymptotes of $P$ and $Q$.

Algorithm 8 decomposes $F$ into factors according to the specialization Lemma 2. Recall that $\mathrm{sres}_{y,i}(P, Q)$ denotes the coefficient of the monomial of degree $i$ in $y$ of $\mathrm{Sres}_{y,i}(P, Q)$, the $i$-th polynomial subresultant of $P$ and $Q$ with respect to $y$. Polynomial $F$ is decomposed into factors $F_i$, $i = 1, 2, \ldots$, such that for any root $\alpha$ of $F_i$, $\mathrm{sres}_{y,i}(P, Q)(\alpha)$ is the first (for $i$ increasing) non-vanishing coefficient. The algorithm then returns the set of non-trivial triangular systems $\{F_i, \mathrm{Sres}_{y,i}(P, Q)\}$ whose solutions are, by Lemma 2, those of $\{P, Q, A\}$ that are not on the common vertical asymptotes of $P$ and $Q$. The triangular systems are regular by construction.

**Algorithm 9: Complete triangular decomposition of $\{P, Q\}$**

Algorithm 9 takes as input a zero-dimensional system $\{P, Q\}$ in $\mathbb{Z}[x, y]$ and computes a set of regular triangular systems whose solutions are those of $\{P, Q\}$. Algorithm 9 calls Algorithm 8 recursively, first with the input polynomials $P_1 = P$, $Q_1 = Q$ and $A_1 \equiv 0$, and then, for $h \geqslant 2$, with $P_h = P_{h-1} - \mathrm{Lc}_y(P_{h-1})y^{\deg_y(P_{h-1})}$, $Q_h = Q_{h-1} - \mathrm{Lc}_y(Q_{h-1})y^{\deg_y(Q_{h-1})}$ and $A_h \in \mathbb{Z}[x]$ that vanishes exactly on the common vertical asymptotes of $P_1, Q_1, \ldots, P_{h-1}, Q_{h-1}$ that are not common vertical asymptotes of $P_h$ and $Q_h$.

### 2.3.2 RUR decomposition algorithm for Isotop

The first step of the Isotop algorithm for isolating the critical points of a curve is to compute a complete triangular decomposition as explained in Section 2.3.1. The multiplicities in these triangular systems are the multiplicities in fibers we want for computing the local topology at critical points. In order to keep these multiplicities we compute RURs of each triangular system. The RUR computation includes the computation of a separating form. To simplify the computations in practice, it is desirable to identify when the form $x$ is separating, that when the triangular system is in generic position. To do so we turn the genericity test introduced in [Dia09] into a procedure to split a triangular system in its generic and non-generic parts. Next, we show that the particular structure of the triangular systems improves the classical RUR-candidate algorithm of [Rou99]. Finally, we show how the RUR-candidate can be checked to be a RUR, i.e. that the chosen linear form is indeed separating. We give below more details about these algorithms and refer to [BLPR11*, Bou14] for a more in depth description.

#### Splitting a triangular system in generic and non-generic parts

Each triangular system obtained via our decomposition has the form

$$S_k = \begin{cases} F_k(x) \\ \mathrm{Sres}_k(x, y) = \mathrm{sres}_{k,k}(x)y^k + \mathrm{sres}_{k,k-1}(x)y^{k-1} + \ldots \end{cases}$$

such that $\gcd(F_k(x), \mathrm{sres}_k(x)) = 1$ and $F_k$ is squarefree. This system is in generic position if and only if, above any root $\alpha$ of $F_k$, the bivariate polynomial $\mathrm{Sres}_k(\alpha, y)$ has only one root, that is it factorizes as $\mathrm{sres}_{k,k}(\alpha)(y - \beta)^k = \mathrm{sres}_{k,k}(\alpha)(y + \frac{\mathrm{sres}_{k,k-1}(\alpha)}{k\,\mathrm{sres}_{k,k}(\alpha)})^k$. The idea of the genericity check in [Dia09, Theorem 3.3.8] is to expand this expression and test by divisibility that all coefficients match those of $\mathrm{Sres}_k(\alpha, y)$. If we replace the divisibility test by a gcd computation, we obtain the factor of $F_k$ that encodes generic solutions and the other factor (the gcd free part) encodes the non-generic ones.

#### Computing a RUR-candidate of a triangular system

A specific property of the decomposition obtained above is that $\gcd(F_k(x), \mathrm{sres}_k(x)) = 1$. By Bézout identity, there exists $U, V \in \mathbb{Q}[x]$ such that $UF_k + V\mathrm{sres}_k = 1$. Thus, since $\mathrm{Sres}_k(x, y) = \mathrm{sres}_k(P, Q)y^k + R_k(x, y)$ (with $R_k$ of degree less than $k$ in $y$), the system $S_k$ is equivalent to $\tilde{S}_k : \begin{cases} F_k(x) \\ y^k + \tilde{R}_k(x, y) \end{cases}$ where

$\tilde{R}_k$ is the reduction of $R_k$ modulo $F_k$; hence $\tilde{R}_k$ has degree in $y$ strictly less than $k$ and strictly less than that of $F_k$ in $x$. The resulting system thus is a (reduced) lexicographic Gröbner basis [BPR06].

At this stage, we can thus use a standard Rational Univariate Representation (RUR) algorithm for solving this system (see e.g. [Rou99]). The particular structure of this system yields improvements to the algorithm of Rouillier [Rou99] for computing a RUR. The critical properties of our input system $\tilde{S}_k$ is that (i) it is a Gröbner basis *for the lexicographic order*, which implies that one of the polynomials is univariate, and (ii) there are only two polynomials in the basis. Most of the RUR computation are performed using linear algebra in the quotient algebra $\mathbb{Q}[x,y]/\tilde{S}_k$. Property (ii) implies that a basis of this algebra is simply $\{x^i y^j\}$, for $0 \leqslant i < \text{degree}(F_k)$ and $0 \leqslant j < k$. One important step of the algorithm is to compute the product of all pairs of elements of that basis, reduced modulo the system $\tilde{S}_k$; this step is substantially simplified by the structure of the basis and by using Property (i). The complexity of this step, and actually of the whole RUR computation, is reduced from $O(D^3)$ to $O(D^2)$ where $D$ is the size of the algebra basis [Bou14, §5.3.2].

### Checking separability

For a generic triangular system, the linear form $x$ is separating and a RUR is computed with this form. For a non-generic system, we try the forms $y$ and then $x + ay$ with $a$ an integer increasing from 1 and compute the corresponding RUR-candidate. The first step is to split the RUR according to multiplicities. Since the RUR respect multiplicities of the input system, the computed multiplicities are the multiplicities in fibers encoded in the input triangular systems. The multiplicities being encoded in the univariate polynomial of the extension, splitting a RUR just means factoring this polynomial. We then check whether the roots of the RUR $(f, g_x, g_y, g)$ are indeed roots of the original system $(P, Q)$ with the correct multiplicities. Checking whether the roots are correct is done by substituting the rational coordinates $x = \frac{g_x(t)}{g(t)}$ and $y = \frac{g_y(t)}{g(t)}$ in $P$ and $Q$ and checking that $f$ divides its numerator. A multiplicity $m$ of a root $(\alpha, \beta)$ corresponding to a root $t$ of $f$ can be verified similarly by substituting the rational coordinates in the $i$-th derivative of $P$ and $Q$ with respect to $y$ (for $i$ from 1 to $m-1$), and checking divisibility by $f$.

### Modular methods

The binary size of the RUR polynomials is reasonable in the sense that their coefficients have more or less the same size as those in the resultant (see Proposition 27). However, the size of the coefficients of the subresultants and lexicographic Gröbner bases in the intermediate computations may be quite large. A standard approach for avoiding such intermediate growth is to use modular computations and the Chinese Remainder Theorem [BPR06]. The difficulty being to design a Las-Vegas algorithm rather than a Monte-Carlo algorithm. We showed that an approach similar to what we exposed in Section 2.2 can be applied here. It is intersting to note that our Las-Vegas algorithm checks the correctness of the triangular decomposition and the separability at the same time [Bou14, §5.4.4].

# Chapter 3

# ISOTOP algorithm: topology of algebraic curves

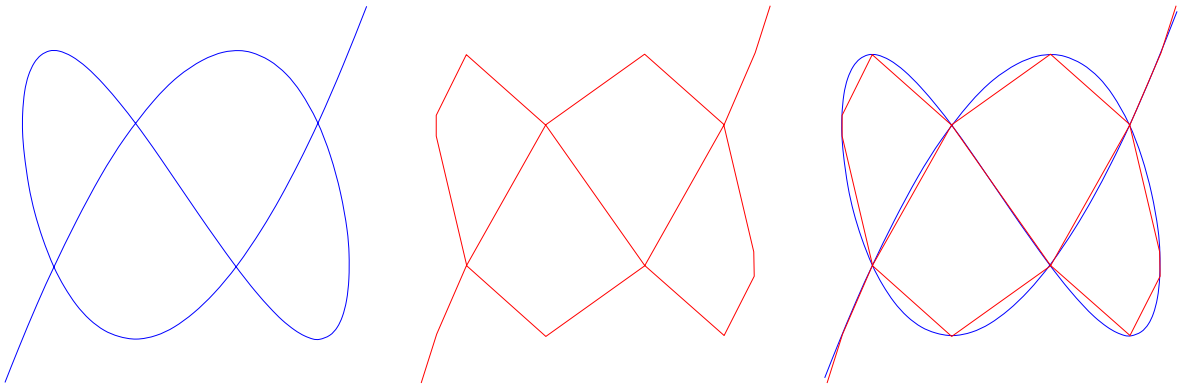
A detailed description of my ISOTOP algorithm was published in [CLP+10*]. In this chapter, I recall the algorithm and its original contributions. I also present the improvements that came afterwards and in particular from my work on bivariate systems [BLPR11*, LPR17*] already detailed in Chapter 2. An interactive visualization is available via a webserver (`https://isotop.gamble.loria.fr/`). The experiments performed on an extensive database of curves show that ISOTOP outperforms other software for most classes of examples [BLPR15*].

On this topic, I co-advised the PhD of Luis Peñaranda with Sylvain Lazard. I also collaborated with two postdocs: Jinsan Cheng and Elias Tsigaridas on the first version of our algorithm and software. The webserver of ISOTOP was developed together with the Inria engineer Benjamin Dexheimer.

After introducing the problem (Section 3.1) and previous work (Section 3.2), I detail the contributions of ISOTOP: the rectangle decomposition of the plane without shearing (Section 3.3.1); the use of RUR representations of the critical points (Section 3.3.2); the computation and the use of multiplicities in fibers (Sections 3.3.3 and 3.3.4); the parallel algorithm using multimodular computations (Section 3.3.5).

## 3.1 Introduction

Let $\mathcal{C}$ be a real algebraic plane curve defined in a Cartesian coordinate system by a bivariate polynomial $f$ with rational coefficients, *i.e.*, $\mathcal{C} = \{(x, y) \in \mathbb{R}^2 \mid f(x, y) = 0\}$ with $f \in \mathbb{Q}[x, y]$. We consider the problem of computing the topology of $\mathcal{C}$ with additional geometric information associated to the given coordinate system. By computing the topology of $\mathcal{C}$, we mean to compute an arrangement of polylines $\mathcal{G}$,



(a) (b) (c)

Figure 3.1: (a) Curve of equation $16\,x^5 - 20\,x^3 + 5\,x - 4\,y^3 + 3\,y = 0$ plotted in MAPLE, (b) its isotopic graph computed by ISOTOP, and (c) their overlay.

that is topologically equivalent to $\mathcal{C}$ (see Figure 3.1). Note that this arrangement of polylines $\mathcal{G}$ is often identified to a graph embedded in the plane, where the vertices can be located at infinity and the edges are straight line segments. We first define formally what we mean by topologically equivalent, before discussing the additional geometric information we consider and their relevance.

Two curves $\mathcal{C}$ and $\mathcal{G}$ of the Euclidean plane are said to be (ambient) isotopic if there exists a continuous map $F : \mathbb{R}^2 \times [0,1] \longrightarrow \mathbb{R}^2$, such that $F_t = F(\cdot, t)$ is a homeomorphism for any $t \in [0,1]$, $F_0$ is the identity of $\mathbb{R}^2$ and $F_1(\mathcal{C}) = \mathcal{G}$. This notion formalizes the idea that one can deform one curve to the other by a deformation of the whole plane. Isotopy is stronger than homeomorphy, for instance, (i) two nested loops and (ii) two non-nested loops are not isotopic.[10]

We now discuss the relevance of adding geometric information to the graph $\mathcal{G}$. From the topological point of view, the graph $\mathcal{G}$ must contain vertices that correspond to self-intersections and isolated points of the curve. However, in order to avoid separating such relevant points from other singularities (*e.g.*, cusps), all *singular points* of $\mathcal{C}$, that is, points at which the tangent is not well defined, are chosen to be vertices of the graph.

While singular points are needed for computing the topology of a curve, the extreme points of a curve are also very important for representing its geometry. Precisely, the *extreme points* of $\mathcal{C}$ for a particular direction, say the direction of the $x$-axis, are the non-singular points of $\mathcal{C}$ at which the tangent line is vertical (*i.e.*, parallel to the $y$-axis); the extreme points in the direction of the $x$-axis are called *$x$-extreme*. These extreme points are crucial for various applications and, in particular, for computing arrangements of curves by a standard sweep-line approach [EK08b]. Of course, one can theoretically compute an arrangement of algebraic curves by computing the topology of their product. However, this approach is obviously highly inefficient compared to computing the topology of each input curve and, only then, computing the arrangement of all the curves with a sweep-line algorithm. Note that the $x$-extreme and singular points of $\mathcal{C}$ form together the *$x$-critical points* of the curve (the $x$-coordinates of these points are exactly the positions of a vertical sweep line at which there may be a change in the number of intersection points with $\mathcal{C}$).

It is thus useful to require that all the $x$-critical points of $\mathcal{C}$ are vertices of the graph we want to compute. Unsurprisingly, almost all methods for computing the topology of a curve compute the *critical points* of the curve and associate corresponding vertices in the graph. We refer to [AMW08, BSGY08] for subdivision methods that avoid the computation of non-singular critical points. However, it should be stressed that almost all methods do not necessarily compute the critical points for the specified $x$-direction. Indeed, when the curve is not in *generic position*, that is, if two $x$-critical points have the same $x$-coordinate or if the curve admits a vertical asymptote, most algorithms shear the curve so that the resulting curve is in generic position. As we will see in the sequel, the analysis of a curve in generic position is significantly simpler. Shearing the curve is, however, an issue for several reasons. First, determining whether a curve is in generic position is not a trivial task and it is time consuming [GVEK96, SF90]. Second, if one wants to compute arrangements of algebraic curves with a sweep-line approach, the extreme points of all the curves have to be computed for the same direction. Finally, if the coordinate system is sheared, the polynomial of the initial curve is transformed into a dense polynomial, which slows down, in practice, the computation of the critical points.

## 3.2 Previous work

There have been many papers addressing the problem of computing the topology of algebraic plane curves (or closely related problems) defined by a bivariate polynomial with rational coefficients [ACM84, AM88, CR88, Sak91, Fen92, GVEK96, Hon96, Bro01, GVN02, MC02, Str06, BPR06, MPS+06, EKW07, DET07, AMW08, BEKS13, KS15, DDR+18]. Most of the algorithms assume generic position for the input curve. As mentioned above, this is without loss of generality since we can always shear a curve into generic position [SF90, GVEK96] but this has a substantial negative impact on the computation time. All these algorithms perform the following phases:

1. Project the $x$-critical points of the curve on the $x$-axis via a resultant and isolate the real roots of this univariate polynomial in $x$. This gives the $x$-coordinates of all the $x$-critical points.

2. For each such critical value $x_i$, compute the intersection points between the curve $\mathcal{C}$ and the vertical line $x = x_i$.

[10] Note that in two dimensions, the notion of ambient isotopy is equivalent to the notion of ambient homeomorphy, that is, to the existence of an orientation preserving homeomorphism of $\mathbb{R}^2$ that maps $\mathcal{C}$ onto $\mathcal{G}$ [Bir75, Thm. 4.4 p.161].

3. Through each of these points, determine the number of branches of $\mathcal{C}$ coming from the left and going to the right.

4. Connect all these points appropriately.

The main difficulty in all these algorithms is to compute efficiently all the critical points in Phase 2 because the $x$-critical values in Phase 1 are, a priori, non-rational thus computing the corresponding $y$-coordinates in Phase 2 amounts, in general, to solving a univariate polynomial with non-rational coefficients and at least one multiple root (corresponding to the critical point). To this end, most algorithms [AM88, CR88, BPR06, Fen92, GVEK96, GVN02, Hon96, MPS+06, Sak91, ACM84, Str06] rely heavily on computations with real algebraic numbers, Sturm sequences or subresultant sequences. We now detail a few contributions we believe significant either from a practical point of view or from a complexity point of view.

An approach using a variant of subresultant sequences for computing the critical points in Phase 2 was proposed by Hong [Hon96]. It reduces to sign computations of algebraic numbers to apply a Sturm based subdivision isolation algorithm in the fibers, it is in practice sped up by interval computation in floating point arithmetic. He computed this way ($xy$-parallel) boxes with rational endpoints and separating the critical points. Counting the branches in Phase 3 can then be done by intersecting the boundary of the boxes with the curve which only involves univariate polynomials with rational coefficients. This approach, see also [ACM84, MC02, Bro01] and the software package Cad2D[11], does not assume that the curve is in generic position.

González-Vega and Necula [GVN02] use a triangular decomposition via subresultants in generic position which allows them to express the $y$-coordinate of the each critical point as a rational function of its $x$-coordinate. They implemented their algorithm in MAPLE with symbolic methods modulo the fact that the algebraic coefficients of the polynomials in Phase 2 are approximated in fixed-precision arithmetic. The algorithm takes as a parameter the initial precision for the floating-point arithmetic and recursively increases the precision. This approach is however not certified in the case where the curve is not in generic position because the algorithm checks for the equality of pairs of polynomials whose coefficients are evaluated (incorrect results have been reported in [SW05]). Note that there exists one variant of González-Vega and Necula algorithm that handles, without shearing, curves that are not in generic position [MPS+06]. This approach, however, requires substantial additional time-consuming symbolic computations such as computing Sturm sequences.

Seidel and Wolpert [SW05] presented an alternate approach for computing the critical points avoiding most costly algebraic computations but to the expense of computing several projections of the critical points. They project, in Phase 1, the $x$-critical points on both $x$ and $y$-axes and also on a third random axis. After isolating the roots on each axis, they can recover ($xy$-parallel) boxes with rational endpoints that contain each exactly one critical point. From there, all computations only involve rational numbers but they, however, still need to compute Sturm-Habicht sequences for refining the boxes containing the singular points until each box interests only the branches of the curve incident to the singular point. Their approach assumes that the curve is in generic position by a pre-processing phase in which the curve is sheared if needed. They also present a MAPLE implementation, INSULATE, which is an implementation of a certified algorithm for curves in arbitrary positions. Note that their implementation does not report $x$-extreme points in the original system when the curved is sheared.

Eigenwillig et al. [EKW07] (see also [Ker06]) presented a variant of González-Vega and Necula approach, in which the roots of the polynomials with non-rational coefficients are efficiently isolated using an implementation of a variant of an interval-based Descartes algorithm [EKK+05]. This variant, as [GVN02], does not assume that the curve is in generic position but detects such configurations online. More precisely, if the bit-stream Descartes algorithm is, in a sense, unlucky then, rather than refining down to a separation bound (*e.g.* [BFM+01, LPY04]), the algorithm shears the input curve and starts again. Note that this approach still computes Sturm-Habicht sequences for determining the polynomials appearing in Phase 2 and the multiplicity of its multiple root. Also, if the curve is sheared to a $(x', y')$ coordinate system, they compute extreme points both for the $x'$-direction and the direction corresponding to the $x$-axis. This approach has been implemented in C++ and is an implementation of a certified algorithm that handles curves that are not necessarily in generic positions and that reports $x$-extreme points for the original coordinate system.

Note finally that another approach that avoids expensive algebraic computations is to compute the critical or singular points using subdivision methods [AMW08, BSGY08]. The major drawback of these

[11] https://www.usna.edu/Users/cs/wcbrown/qepcad/B/user/cad2d.html

methods is that, in order to certify the results, the subdivision has, in general, to reach a separation bound (certification can also be achieved by solving algebraic systems by other means). It follows that, if no certification is required, these methods are very fast in practice, however, they can become very slow on difficult instances, if certification is required. It is worth noticing that for the restricted case of smooth curves (which is not our focus in this chapter), subdivision methods combined with interval arithmetic are efficient [Sny92, PV04].

Berberich et al. [BEKS13] propose a symbolic-numeric algorithm for computing arrangements of curves that improves in practice the approach of Eigenwillig et al. [EK08a]. The algorithm is based on a new bivariate solver and a new algorithm for the topology of a single curve. The symbolic computations are reduced to gcds and resultants only and these operations are done in a multimodular setting that enables parallelization on the GPU. For the lifting phase, a filtering is processed with a single modular computation using a formula of Teissier relating multiplicities in fibers to multiplicities in two systems, see Lemma 38. This filtering is often successful and in case of failure, a more symbolic method is used. We performed experiments with this software in [Bou14].

The complexity of the problem of computing the topology of a curve defined by a polynomial of degree $d$ and bitsize $\tau$ has been greatly improved other the years. With $n = \max(d, \tau)$, the bound improved from $\widetilde{O}(n^{30})$ [AM88] to $\widetilde{O}(n^{16})$ [GVEK96], $\widetilde{O}(n^{12})$ [DET07], $\widetilde{O}(n^{10})$ [KS12] and finally to $\widetilde{O}(d^6 + d^5\tau)$ [KS15] ($\widetilde{O}$ denotes that the poly-logarithmic factors are omitted). The best bounds were first found with algorithms performing a shearing of the coordinate system, but it has been shown recently that the $\widetilde{O}(d^6 + d^5\tau)$ bound is also reachable without such a shearing [DDR+18]. This complexity is also the one of Phase 1, that is isolating the resultant encoding critical points which is of degree $\Theta(d^2)$ and bitsize $\Theta(d\tau)$. This implies that further improvements for the complexity of computing the topology of an algebraic curve via this projection and lifting approach must improve the fondamental problem of univariate isolation.

## 3.3 Contributions of Isotop

In this section, we explain in more details the Isotop algorithm and how we improved it over time. Even if we modified significantly some steps, the overall structure of the algorithm remained as follows:

- Isolate $x$-critical points, further classify them as singular or $x$-extreme.
- Compute multiplicities in fibers for critical points.
- Compute local topology at critical points.
- Handle vertical asymptotes and vertical lines.
- Connect using a sweep-line algorithm.

The computations performed by Isotop on the curve defined by $-4\,x^2y^2 + y^4 + 24\,x^3 - 6\,xy^2 + x^2 = 0$ is illustrated on Figure 3.2.

### 3.3.1 Handling non-generic curves in the original coordinate system

We already motivated the importance of being able to report the topology of a curve in its original coordinate system. The difficulties induced by this constraint are the processing of degenerate cases such as the presence of vertical lines included in the curve, vertical asymptotes and the fact that several critical points could be on the same vertical $x$-fiber.

The vertical lines have as $x$-coordinates the roots of the gcd of the coefficients of the polynomial $f(x, y)$ defining the curve seen as a univariate polynomial in $y$. The curve without these lines is first analyzed and the vertical lines are added back. Checking whether a vertical line passes through a critical point (of the curve without these lines) is simplified due to the RUR encoding of these points, as we discuss in the next section.

The $x$-coordinates of vertical asymptotes are the roots of the leading coefficient of the polynomial $f(x, y)$ considered as a polynomial in $y$. To deal with an asymptote $x = \alpha$, the idea is, informally, to isolate the point $(\alpha, \infty)$ in a box $[a, b] \times (]-\infty, -M] \cup [M, \infty[)$ whose vertical sides do not intersect the curve $\mathcal{C}$. Moreover, we want that every branch that intersects a horizontal side of the box is a branch going to $\pm\infty$ with this asymptote box (see Figure 3.3(a)). We thus use a bound on the $y$-coordinates of the $y$-critical points and refinement of the asymptote box for branch counting.

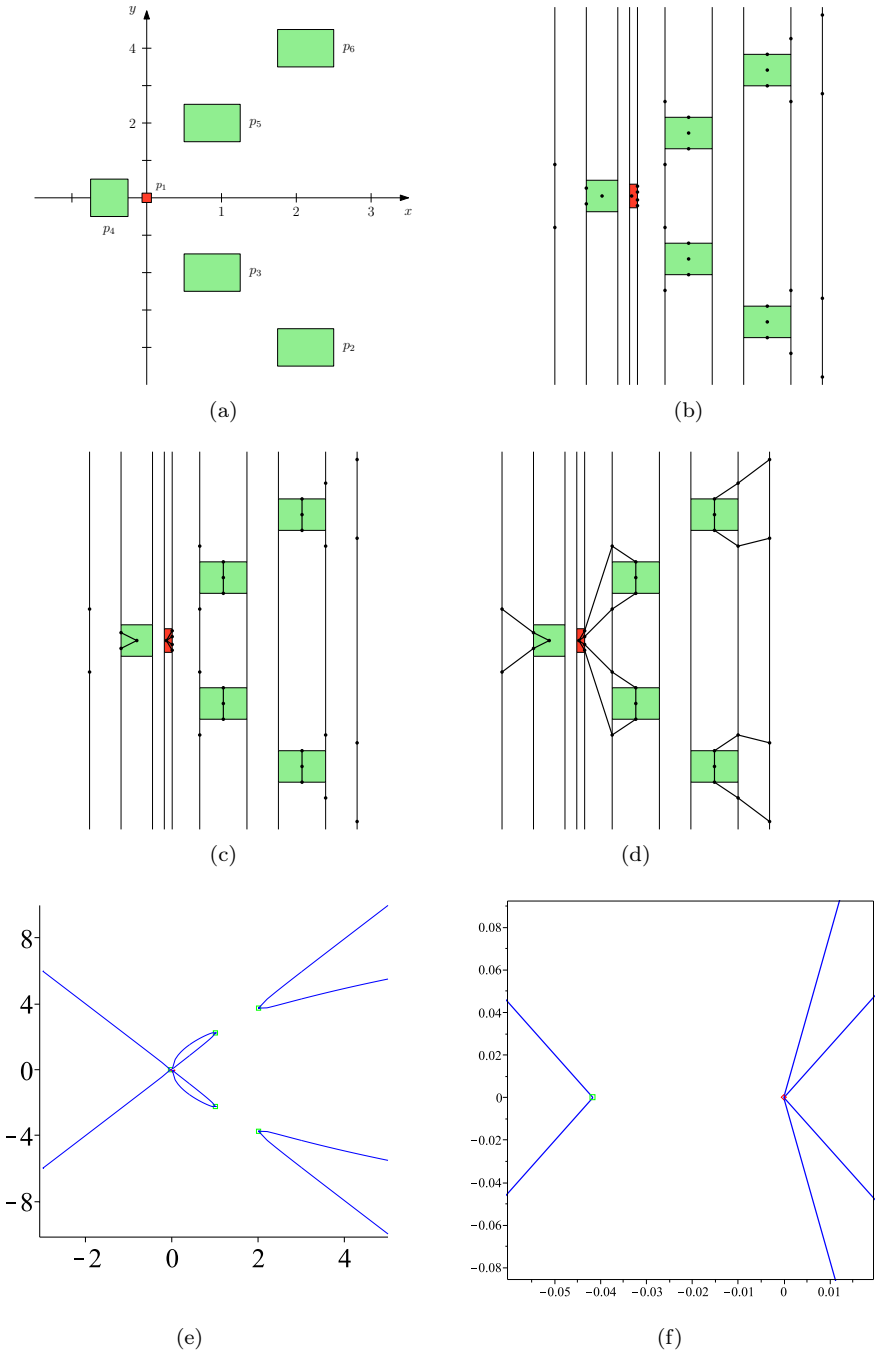



Figure 3.2: (a) Critical boxes of the curve $f(x,y) = -4\,x^2y^2 + y^4 + 24\,x^3 - 6\,xy^2 + x^2 = 0$. The small square at the origin contains the only singular point $p_1$, solution of the system $\{f = \frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = 0\}$. Each other rectangles contains one of the $x$-extreme points $p_2, \ldots, p_6$, solutions of the system $\{f = \frac{\partial f}{\partial y} = 0\}$. (b) Rectangular decomposition of the plane induced by the critical boxes. (c) Topology inside the critical boxes. (d) Graph isotopic to the curve. (e) Graph drawn with refinement. (f) Detail of the graph near the origin.

In the current version of Isotop before computing a RUR of the critical points, a triangular decomposition of the system encoding these points is performed. We presented in [LPR17*] an improvement of the classical triangular decomposition to take into account the presence of asymptotes, this approach is detailed in Section 2.3.1.

### 3.3.2 Rational Univariate Parametrizations (RUR)

In our algorithm, we need to represent solutions of zero-dimensional ideals depending on two variables by boxes containing them. We use the so-called Rational Univariate Representation (RUR) of the roots, which can be viewed as a univariate equivalent to the studied ideal. The key feature of this RUR is the ability to isolate solutions in easily refinable boxes and to compute multiplicities (Definition 24 and Proposition 2.2).

Given a zero-dimensional ideal $I$ in $\mathbb{Q}[x,y]$, for instance $I$ is generated by $f$ and $\frac{\partial f}{\partial y}$ for the critical points, a Rational Univariate Representation of the solutions $\mathrm{V}(I)$ is given by $F(t)=0, x=\frac{G_x(t)}{G_0(t)}, y=\frac{G_y(t)}{G_0(t)}$, where $F, G_0, G_x, G_y$ are univariate polynomials in $\mathbb{Q}[t]$ (where $t$ is a new variable). All these univariate polynomials, and thus the RUR, are uniquely defined with respect to a given polynomial $\gamma \in \mathbb{Q}[x,y]$ which is injective on $V(I)$; $\gamma$ is called the *separating polynomial* of the RUR. The RUR defines a bijection between the (complex and real) roots of the ideal $I$ and those of $F$. Furthermore, this bijection preserves the multiplicities and the real roots. Computing a box for a solution of the system is done by isolating the corresponding root of the univariate polynomial $F$ and evaluating the coordinate functions with interval arithmetic. To refine a box, one just needs to refine the corresponding root of $F$ and evaluate the coordinates again.

An important issue for representing the critical points by a RUR is to find a separating polynomial. The computation of the RUR is easier when this polynomial is as simple as possible and thus a linear form $x+ay$ with $a$ a small integer is the best one can expect. The problem of finding a separating form for a RUR is then similar to the one of finding a shearing to put the curve in generic position. The reader may then wonder what is the advantage of the RUR compared to shearing. First, the RUR still gives the coordinates of the solutions in the original coordinates, it somehow shears back internally. Second, with the current version of Isotop performing a triangular decomposition before computing RURs, we apply a test introduced in [Dia09] to check whether a given triangular system has the polynomial $x$ as separating form. We indeed go one step further and decompose each triangular system in its generic part and non-generic part (that is where $x$ is not separating). This processing dramatically improves the running times in cases where the curve has critical points with different multiplicities in fibers. We detail in Section 2.3.2 our RUR decomposition algorithm specialized for Isotop.

Another advantage of a RUR is the ability of evaluating the sign of another polynomial at solutions of a system with the additional possibility of splitting the RUR in two to encode the solutions where the constraint vanishes or not. This process is described and analyzed in [BLPR15*]. It is applied several times in the algorithm, for instance to distinguish singular from extreme point, to handle vertical lines and asymptotes, to compute multiplicities in fibers, ...

### 3.3.3 Multiplicities in fibers

Knowing the multiplicity in fiber of a critical point is a key point to isolate it from other intersections of the curve in the same fiber and computing the number of branches of the curve on the left and right, that is computing its "local" topology.

We now recall the notion of multiplicity of the roots of an ideal, then we state two lemmas using this notion for studying the local topology at critical points. Geometrically, the notion of multiplicity of intersection of two regular curves is intuitive. If the intersection is transverse, the multiplicity is one; otherwise, it is greater than one and it measures the level of degeneracy of the tangential contact between the curves. Defining the multiplicity of the intersection of two curves at a point that is singular for one of them (or possibly both) is more involved and an abstract and general concept of multiplicity in an ideal is needed. We refer to [CLO05, §4.2] for a formal definition of the intersection multiplicity via the dimension of local rings. Let $f, g \in \mathbb{Q}[x,y]$ be such that the associated curves intersect at an isolated point $\mathbf{p}=(\alpha,\beta)$, we denote by $\mathrm{mult}(\mathbf{p},\{f,g\})$ the **intersection multiplicity of the two curves at this point**. We call a **fiber** a vertical line of equation $x=\alpha$. For a point $\mathbf{p}=(\alpha,\beta)$ on the curve $\mathcal{C}_f$, we call the multiplicity of $\beta$ in the univariate polynomial $f(\alpha,y)$ the **multiplicity of p in its fiber** and denote it as $\mathrm{mult}(\beta, f(\alpha,y))$.

The next lemma, due to Teissier [Tei73], relates the multiplicity of a point in a fiber with the multiplicity in the critical ideal. We will use it to deduce the multiplicity in the fiber knowing multiplicity in the ideal. In the following, we use the notation $f_x = \frac{\partial f}{\partial x}$ and $f_{x^k} = \frac{\partial^k f}{\partial^k x}$.

**Lemma 38** ([Tei73][BR90, Lemma D.3.4 p.314]). *For an $x$-critical point $\mathbf{p} = (\alpha, \beta)$ of $f$ one has*

$$\text{mult}(\beta, f(\alpha, y)) = \text{mult}(\mathbf{p}, \{f, f_y\}) - \text{mult}(\mathbf{p}, \{f_x, f_y\}) + 1. \tag{3.1}$$

To compute the local topology of the curve at a singular point, we aim at isolating the singular point in a box so that the intersection of its border and the curve determines the topology. Indeed for a small enough box, the topology is given by the connection of the singular point with all the intersections on the border. So the box shall avoid parts of the curve not connected to the singular point. Knowing the multiplicity of the singular point in the fiber enables to isolate the singular point from other crossings of the curve in this fiber. Requiring in addition that intersections with the curve only occur on the left or right sides of the box leads to the following lemma.

**Lemma 39** ([SW05]). *Let $\mathbf{p} = (\alpha, \beta)$ be a real singular point of the curve $\mathcal{C}_f$ of multiplicity $k$ in its fiber. Let $B$ be a box satisfying (i) $B$ contains $\mathbf{p}$ and no other $x$-critical point, (ii) the function $f_{y^k}$ does not vanish on $B$, and (iii) the curve $\mathcal{C}_f$ crosses the border of $B$ only on the left or the right sides. Then the topology of the curve in $B$ is given by connecting the singular point with all the intersections on the border.*

The proof of Lemma 39 is based on a recursive application of the mean value theorem stating that the roots of the derivative of a polynomial $P$ lie between those of $P$. We now detail, in chronological order, the different methods we experimented in Isotop for computing the multiplicities in fibers.

**Method using Teissier formula [CLP+08*].** Our first attempt to compute multiplicities in fibers was to use Teissier formula via the computation of RURs of the systems $\{f, f_y\}$ and $\{f_x/\gcd(f_x, f_y), f_y/\gcd(f_x, f_y)\}$. This had the disadvantage of solving an additional system.

**Method using saturation [CLP+10*].** For singular points, we use the definition of univariate multiplicity, namely the smallest integer $k$ such that the $k^{th}$ derivative no longer vanishes. Let $I_{s,k}$ be the system of singular points with in addition the equations $f_{y^i} = 0$ for $i$ from 2 to $k$. We solve, for $k$ increasing from 2, the systems $I_{s,k}$ until it has no solutions. At each step, a singular point which was a solution of $I_{s,k-1}$ but is no longer solution of $I_{s,k}$ has its multiplicity in fiber equal to $k$. In practice, as $k$ increases, the systems have less and less solutions and hence tend to be easier to solve. Note also that the number of systems to solve is the highest multiplicity of the singular points of the curve.

**Method using a triangular decomposition and RURs [BLPR11*].** Finally, the current version of Isotop uses a triangular decomposition via subresultants before computing RURs as detailed in Section 2.3.2. More precisely the subresultant sequence of $f$ and $f_y$ is computed and due to the specialization property the triangular system associated to the subresultant of order $k$ encodes the solutions with $x$-coordinate $\alpha$ such that the subresultant $\text{Sres}_k(f, f_y)(\alpha, y) = \gcd(f(\alpha, y), f_y(\alpha, y))$ is of degree $k$ in $y$. The multiplicity of a critical point $(\alpha, \beta)$ in its fiber is thus the multiplicity in this triangular system plus one. In the generic case, that is when $x$ is a separating form, the multiplicities are then all $k+1$ for this triangular system. As explained above in Section 3.3.2, when this triangular system is not generic, it is split in its generic and non-generic parts. For the non-generic part, the RUR still preserves the multiplicities in the system and thus computes the multiplicities in fibers.

### 3.3.4 Sweep-line connection algorithm

We have to stress that our approach does not compute a Cylindrical Algebraic Decomposition and instead is based on a rectangular decomposition of the plane. For simplicity, all the boxes computed above are called critical boxes and the points at infinity on vertical asymptotes are also called critical. First we compute, with a sweep-line algorithm, the vertical rectangular decomposition obtained by extending the vertical sides of the critical boxes either to infinity or to the first encountered critical box, see Figure 3.3(a). On each of the edges of the decomposition, we isolate the intersections with $\mathcal{C}$. We create vertices in the graph corresponding to these intersection points and to the critical points. For describing the arcs connecting these vertices in the graph, we assimilate, for simplicity, the points and the graph

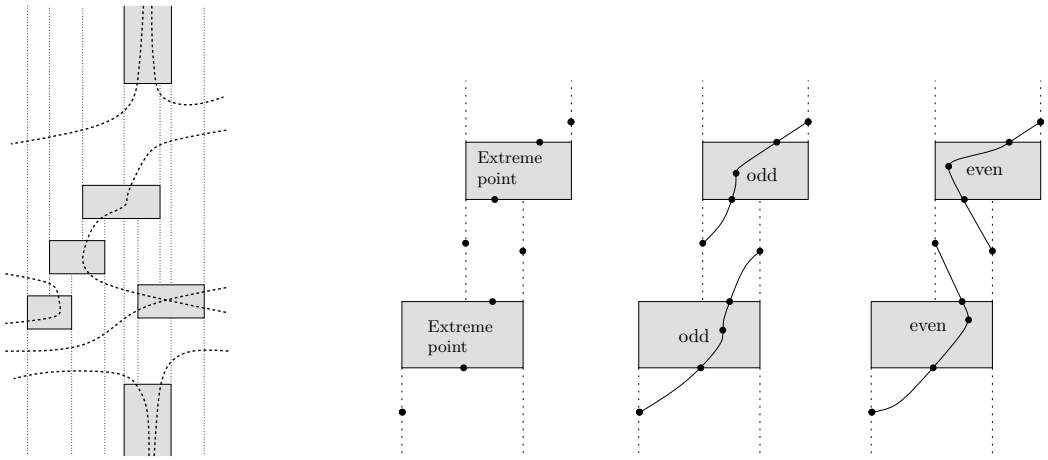



Figure 3.3: (a) Example of rectangle decomposition of the plane induced by the isolating boxes (critical and asymptotes). There are boxes for the asymptote, the singular point, and the three extreme points, two of them with even multiplicities and one with odd multiplicity. (b) Possible connections involving extreme points depending on their multiplicities.

vertices. In each critical box the topology is simple once sufficient refinements ensure the conditions of Lemma 39: the critical point is connected to each of the intersection points on the boundary of the box.

There are several approaches to do the connections in the other rectangles of the decomposition. The usual and conceptually simplest is to refine boxes of extreme points so as to avoid top and bottom crossings; then, the number of left and right crossings in rectangles always match and the connection is one-to-one. Unlike comparable algorithms, we do not require that $\mathcal{C}$ intersects the boundary of $B$ on its vertical sides. This is important in practice because, since the curve has a vertical tangent at an $x$-extreme point, refining until the curve intersects the vertical side is time consuming. Since we allow top/bottom crossings for efficiency, the straightforward one-to-one connection method in rectangles does not apply. Another approach (see [AMW08, Sak91]) is to compute the sign of the slope of the tangent to the curve at the top/bottom crossings (this yields whether the top/bottom crossing should be connected to vertex to the right or to the left of its rectangle). We however want to avoid such additional computations.

For computing the connections in the non-critical rectangles of the decomposition, we use the multiplicities in fibers of the extreme points and a greedy algorithm. The geometric meaning of the parity of this multiplicity is the following: if it is even, the curve makes a U-turn at the extreme point, else it is odd and the curve is $x$-monotone in the neighborhood of the extreme point. Still, there are some difficulties for connecting the vertices, as illustrated on Figure 3.3(b): on the left is the information we may have on the crossings for two extreme points with $x$-overlapping boxes; the second and third drawings are two possible connections *in the middle rectangle* for different parities of the multiplicities. To distinguish between these two situations we compute the connections in rectangles starting from the top such that the connections in a rectangle below a critical box are computed once the connections in all the rectangles above the box are done.

### 3.3.5 Multimodular computations and parallelization

Our method for isolating the critical points of the curve is to first compute a triangular decomposition before computing RURs. We proved in [BLPR15*, Prop. 28] and [BLM+16*, Cor. 36] that the total bitsize of the RURs of the decomposition is in $\widetilde{O}(d^4 + d^3\tau)$. Comparatively, the total bitsize of the polynomials in the triangular decomposition is in $\widetilde{O}(d^5+d^4\tau)$ in the worst case. To avoid this intermediate coefficient swell, we use computations modulo a set of small prime numbers and only lift over $\mathbb{Q}$ the final RURs but not the triangular decomposition. In addition, such computations are well suited to be parallelized and experiments with our multi-thread implementation are reported in [Bou14].

# Chapter 4

# Reliable numeric algorithms for the topology of plane projections of smooth curves

I have discussed in Chapter 3 the problem of computing the topology of a real algebraic plane curve given by an implicit polynomial equation via symbolic methods. In parallel to this work, I have investigated certified numerical approaches that apply to a larger class of smooth curves. In [IMP16*, IMP15a*, IMP18*], I proposed a solution for the projection in the plane of generic smooth curves in 3D. Such projections are generically singular but the singularities are only nodes or ordinary cusps, see Figure 1.2. I generalized this work to curves in higher dimensions in [KLMP21*]. The remaining of this section is mainly based on this latter work.

On this topic, I co-advised the PhD of George Krait with Sylvain Lazard and Guillaume Moroz. I also collaborated with a postdoc, Rémi Imbach, and an engineer, Mohamed Eissa, via the ANR SingCAST I coordinated with Guillaume Moroz.

I present a square and regular system that encodes the singularities of the plane projection of a $C^\infty$ smooth curve in $\mathbb{R}^n$ (Section 4.3). This approach does not use elimination theory to compute the equation of the projected curve and it is not restricted to the algebraic case: it applies to the larger class of $C^\infty$ smooth curves. Being square and regular, this system can thus be solved with state-of-the-art certified numerical methods based on interval arithmetic or certified homotopy. However it encodes the singularities of the plane projection only if some assumptions, defined in Section 4.2, are satisfied. My second main result is that those assumptions are satisfied generically, which I prove using transversality theory. I also present semi-algorithms that checks whether a given curve satisfies my assumptions, that is, an algorithm that stops if and only if the assumptions are satisfied. The combination of these results provides a method that is both numerical and certified for isolating the singularities of the plane projection of a generic curve (Section 4.4). For the case of curves in $\mathbb{R}^3$, I go one step further and design a data structure for encoding a topologically correct and geometrically accurate representation of the projected curve that enables location queries (Section 4.5). Finally, I present several illustrative experiments for a high degree algebraic curve and an application in robotics (Section 4.6).

## 4.1 Introduction

### 4.1.1 Context

For the problem of drawing plane curves with the correct topology, one of the main challenges is to isolate the singular points efficiently and correctly. Our aim here is to do so with certified numerical methods and we show that this could be achieved for the specific class of plane curves that are projections of $C^\infty$ smooth curves in higher dimension.

Although this class of curves seems specific, it appears naturally in visualization and robotic applications and the curves from this class often contain singularities. When visualizing a curve given by $n-1$ implicit equations in $n$ dimensions for instance, we compute its projection in 2D to display it on a screen. This class of curves also appears in robotic applications. For instance given a robot with two degrees of freedom that moves in the plane, the set of points it can reach is bounded by a curve. In this case,

computing the correct topology of this curve is often needed for deciding if a specific position is reachable. This curve is usually the projection of a smooth curve embedded in a space of higher dimension, and it often contains singular points.

By certified algorithms, we refer to algorithms that always output mathematically correct results in a given model of computation; for instance, randomized Las-Vegas algorithms are (usually) certified, but randomized Monte-Carlo algorithms are not; numerical methods that may miss solutions or output spurious solutions are not certified. Recall that the singular points of a plane curve, defined by the equation $f(x, y) = 0$, are the solutions of the system defined by $f(x, y) = \frac{\partial f}{\partial x}(x, y) = \frac{\partial f}{\partial y}(x, y) = 0$; it should be stressed that this system is over-determined, i.e., it has more equations than variables, which prevents the use of certified numerical methods such as interval Newton methods [MKC09]. The latter system can be translated into a square system using combinations of its equations with first derivatives [Ded06], and non-regular solutions can be handled through deflation systems [OWM83, LVZ06], but the resulting systems are usually still overdetermined. On the other hand, symbolic methods can solve such over-determined systems but they are restricted to algebraic systems and their complexity is high with respect to the degree of the equations.

### 4.1.2 State of the art

The problem of isolating the singularities of a plane curve is a special case of the problem of isolating the solutions of a zero-dimensional system in $\mathbb{R}^2$. We give a concise summary of the state of the art of certified methods for these two problems, organized in two main classes.

*Symbolic methods.* Symbolic methods are widely used for solving in a certified way zero-dimensional algebraic systems. Such classical methods are based on Gröbner bases, resultant theory and univariate representations (see e.g., [CLO92, BPR06]). In this context, methods dedicated to the bivariate case have also been designed as discussed in Chapter 2. Compared to numerical methods, these methods are adapted to over-constrained or non-regular systems. On the other hand, they suffer several drawbacks. They are not adaptive in the sense that solving in a small region is not easier than solving for all solutions. They are limited to algebraic systems and their complexity is high with respect to the degree of the system.

*Certified numerical methods.* When a zero-dimensional system is square and regular, its solutions can be isolated in a certified way using interval-arithmetic subdivision methods [Neu90, MKC09] or homotopy approaches with certified path tracking (see [BL13] and references therein). However, these methods do not directly work for isolating the singularities of a plane curve given by the equation $f(x, y) = 0$ because the system $f(x, y) = \frac{\partial f}{\partial x}(x, y) = \frac{\partial f}{\partial y}(x, y) = 0$ that encodes the singularities is neither square nor regular. On the other hand, the curve may not be given by its implicit equation and computing this representation may not be required nor desirable. There are only few contributions designing certified numerical algorithms, even with additional restrictions on the curve and its singularities. When the curve is algebraic, Burr et al. [BCGY12] use separation bounds to isolate the singularities via a subdivision algorithm but the worst-case values of such bounds make it inefficient in practice. Lien et al. [LSVY14, LSVY20] study the special case of a singular curve that is a union of non-singular ones such that the singularities are only transverse intersections between them. They propose a subdivision algorithm using the Moore-Kioustelidis interval test for isolating the square and regular system defined by two curves. No implementation is available but such a subdivision scheme in two dimensions is expected to be efficient. In the case where the plane curve is defined as a projection, only two contributions present certified numerical approaches for isolating the singularities. Delanoue and Lagrange [DL14] consider the topology of the set of critical values of a mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$. The graph of such a function is a surface in $\mathbb{R}^4$, the critical points with respect to the projection on the image space is the silhouette curve whose projection is the set of critical values. Cusps and nodes singularities are described in order to be handled by interval analysis based scheme and a topological encoding of the plane curve of critical values is proposed. Plantinga and Vegter [PV06] study one-parameter families of silhouettes of surfaces in $\mathbb{R}^3$ that model a moving point of view of a surface. They compute a topologically correct approximation of the silhouette in $\mathbb{R}^3$ and the values of the parameter at which it becomes singular, but they do not certify its projection.

## 4.2 Generic assumptions

For a positive integer $n$, a closed (resp. open) $n$-box is the Cartesian product of $n$ closed (resp. open) intervals. Assume that $n \geqslant 3$ and let $B$ be an open $n$-box and $\overline{B}$ be the topological closure of $B$


Figure 4.1: Left: At an $A_1$ singularity, two branches of the curve intersect transversally. Right: At an $A_{2k+1}$ singularity with $k > 1$, the tangent lines of the two branches at the intersection point coincide.

with respect to the usual topology in $\mathbb{R}^n$. Let $C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$ denote the set of smooth functions (i.e., differentiable infinitely many times) from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$. Consider the function $P = (P_1, \ldots, P_{n-1}) \in C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$. We denote by $\mathfrak{C}$ (resp. $\overline{\mathfrak{C}}$) the solution set of the system $\{P_1(x) = \cdots = P_{n-1}(x) = 0\}$, with $x = (x_1, \ldots, x_n) \in B$ (resp. with $x \in \overline{B}$). Also, consider the projection $\pi_{\mathfrak{C}}$ (resp. $\pi_{\overline{\mathfrak{C}}}$) from $\mathfrak{C}$ (resp. $\overline{\mathfrak{C}}$) to the $(x_1, x_2)$-plane. Unless otherwise stated, the plane projection of a point $x \in \mathbb{R}^n$ is $(x_1, x_2)$.

A singularity of a plane curve is of type $A_k$ if the curve is locally defined at the origin by the 0-set of the function $x^2 - y^{k+1}$. As important special cases, $A_1$ is called a node singularity and $A_2$ is called an ordinary cusp singularity, see Figure 4.1.

If $\overline{\mathfrak{C}}$ is a smooth curve, define $\mathfrak{L}_{\mathrm{c}}$ (resp. $\overline{\mathfrak{L}_{\mathrm{c}}}$) as the set of points $q$ in $\mathfrak{C}$ (resp. $\overline{\mathfrak{C}}$) where the tangent line, denoted by $T_q\mathfrak{C}$, (resp. $T_q\overline{\mathfrak{C}}$) is orthogonal to the $(x_1, x_2)$-plane. We also define the set $\mathfrak{L}_{\mathrm{n}}$ (resp. $\overline{\mathfrak{L}_{\mathrm{n}}}$) to be the set of points $q$ in $\mathfrak{C}$ (resp. $\overline{\mathfrak{C}}$) such that the cardinality of the pre-image of $\pi_{\mathfrak{C}}(q)$ under $\pi_{\mathfrak{C}}$ (resp. $\pi_{\overline{\mathfrak{C}}}$) is at least two without counting multiplicities. We will see that, under some generic assumption, $\mathfrak{L}_{\mathrm{c}}$ (resp. $\mathfrak{L}_{\mathrm{n}}$) is the set of points in $\mathfrak{C}$ that project to cusps (resp. nodes), which explains the subscript c (resp. n).

Our goal is to design a numerical algorithm for isolating the singularities that appear in the plane projection $\pi_{\mathfrak{C}}(\mathfrak{C})$ of a curve $\mathfrak{C}$ in $\mathbb{R}^n$. Numerical algorithms usually cannot handle degenerate cases, that is, not any type of singularities can be handled. We require that only node singularities can appear in the projection (Assumption $\mathcal{A}_5$). Our other assumptions on $\mathfrak{C}$ are, roughly speaking, that it is smooth ($\mathcal{A}_1$), that its projection only has a discrete set of singularities ($\mathcal{A}_4$), and that at most two points of $\mathfrak{C}$ project on each singularity ($\mathcal{A}_3$); see Figure 4.2.

$\mathcal{A}_1$ – For all $q \in \overline{\mathfrak{C}}$, $\operatorname{rank}(J_P(q)) = n - 1$, where $J_P$ stands for the Jacobian of $P$. In particular, $\overline{\mathfrak{C}}$ is a smooth curve.[12]

$\mathcal{A}_2$ – The set $\overline{\mathfrak{L}_{\mathrm{c}}}$ is discrete and does not intersect the boundary of $B$.

$\mathcal{A}_3$ – For all points $p = (\alpha, \beta) \in \pi_{\overline{\mathfrak{C}}}(\overline{\mathfrak{C}})$, the pre-image of $p$ under $\pi_{\overline{\mathfrak{C}}}$ consists of at most two points in $\overline{B}$ counted with multiplicities in the system $\{P(x) = 0 \in \mathbb{R}^{n-1}, x_1 - \alpha = x_2 - \beta = 0\}$.

$\mathcal{A}_4$ – The set $\overline{\mathfrak{L}_{\mathrm{n}}}$ is discrete and does not intersect the boundary of $B$.

$\mathcal{A}_5$ – The singular points of $\pi_{\mathfrak{C}}(\mathfrak{C})$ are only nodes.

Our first contribution is to show the genericity of these assumptions.

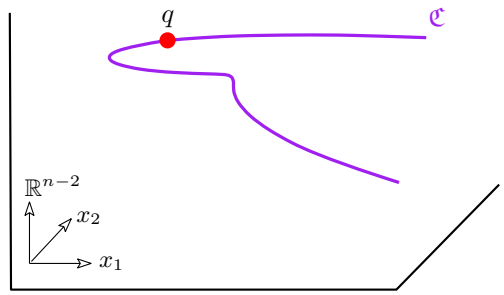
**Theorem 40** ([KLMP21*, §7]). *For a curve defined by $P \in C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$, Assumptions $\mathcal{A}_{1-5}$ are generic.*

Roughly speaking, a property is generic when it holds for most functions and it remains true by perturbation. To give a precise definition, we work with the set of smooth functions $C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$ with the weak (or compact-open) topology [Dem00, §3.9.2], that is convergence is understood as uniform on compact subsets and for any derivative. A subset of $C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$ is called residual if it contains the intersection of a countable family of dense open subsets. A property is generic in $C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$ if it is satisfied by a residual subset.

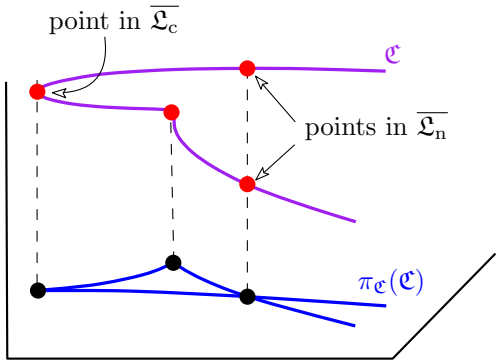
The key to prove the genericity of our assumptions is Thom's Transversality Theorem [Dem00, Theorem 3.9.4]. The idea of the proofs of genericity of our assumptions is to express each assumption as a system of equations in the *jet space*, which is an encoding of the Taylor expansions of the function $P \in C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$.

We also consider a weaker version of Assumption $\mathcal{A}_5$ in which ordinary cusps can also appear in the projection:

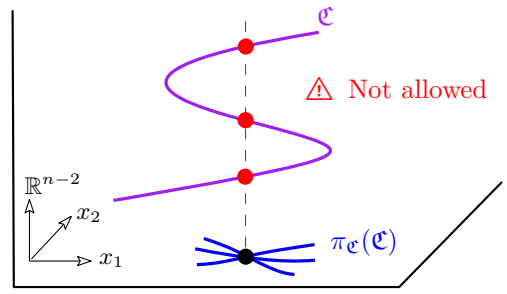
[12] Note that the converse is not true as the vertical (double) line defined by $x_1^2 = x_2 = 0$ in $\mathbb{R}^3$ is smooth but the rank of its Jacobian is never full.



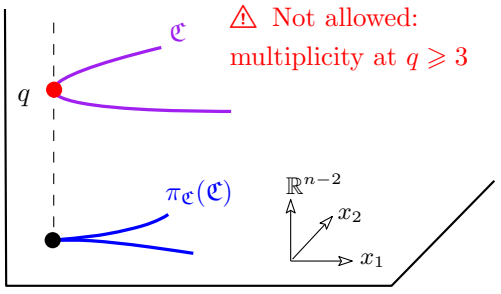

(a) Assumption $\mathcal{A}_1$: $\overline{\mathfrak{C}}$ is a smooth; the rank of the Jacobian of $P$ at $q$ is full.



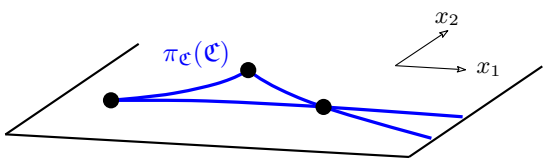

(b) Assumptions $\mathcal{A}_2$ and $\mathcal{A}_4$: the sets $\overline{\mathfrak{L}_c}$ and $\overline{\mathfrak{L}_n}$ are finite and do not intersect the boundary of $B$.



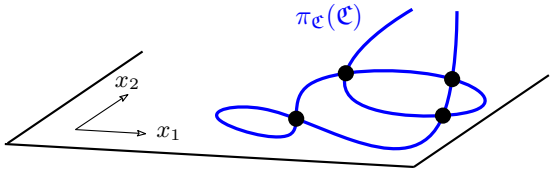

(c) Assumption $\mathcal{A}_3$: No three points (counted with multiplicity) have the same projection.




(d) Assumption $\mathcal{A}_3$: No three points (counted with multiplicity) have the same projection.




(e) Assumption $\mathcal{A}_5^-$: points in $\pi_{\mathfrak{C}}(\mathfrak{C})$ are either smooth, nodes or ordinary cusps.




(f) Assumption $\mathcal{A}_5$: points in $\pi_{\mathfrak{C}}(\mathfrak{C})$ are only smooth or nodes.

Figure 4.2: Illustration of the assumptions.

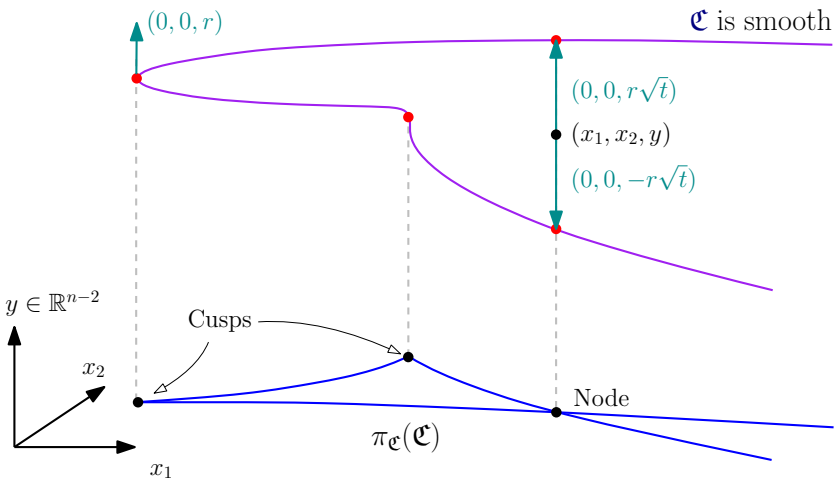



Figure 4.3: Illustration of a node and cusps in the plane projection of a smooth curve.

$\mathcal{A}_5^-$ – The singular points of $\pi_{\mathfrak{C}}(\mathfrak{C})$ are only ordinary cusps or nodes.

**Definition 41.** *Assumptions $\mathcal{A}_{1-5}$ are called the* strong assumptions *and Assumptions $\mathcal{A}_{1-4}$ and $\mathcal{A}_5^-$ are called the* weak assumptions.

Our motivation for also considering these weak assumptions is dual. First, our certified algorithm for isolating the singularities of the projection of curves satisfying the strong assumptions also works, to some extent, if only the weak assumptions hold: namely, it outputs a *superset* of the isolating boxes of the singularities, see Section 4.4. Second, we want to address the case of curves that are the silhouettes of smooth surfaces in $\mathbb{R}^n$ (the silhouette being the set of points on the surface where the tangent plane projects on the plane of projection in a line or a point). Such curves naturally appear in parametric systems since they partition the parametric space with respect to the number of solutions of the system. We conjecture that our weak assumptions are satisfied by silhouette curves of generic surfaces but we were not able to prove it completely (see [KLMP21*, §7] for details).

## 4.3 Ball system encoding of singularities

Our goal is, under the weak assumptions, to encode the singularities of the projected curve $\pi_{\mathfrak{C}}(\mathfrak{C})$ by a square and regular system so that it is solvable with certified numerical methods.

By Assumption $\mathcal{A}_5^-$, the singularities of the projected curve $\pi_{\mathfrak{C}}(\mathfrak{C})$ are only nodes and cusps. Intuitively, a node appears when two points of $\mathfrak{C}$ project to the same point and a cusp appears when projecting a point with a tangent line orthogonal to the projection plane (see Figure 4.3).

The idea to encode the nodes is to design a system whose variables are the coordinates of two different points in $\mathbb{R}^n$ constrained to be on $\mathfrak{C}$ and so that they have the same plane projection. To encode a cusp, we design a system whose variables are the coordinates of one point in $\mathbb{R}^n$ constrained to be on $\mathfrak{C}$ with a tangent orthogonal to the projection plane. Furthermore, in order to apply certified numerical methods we need systems that are square and regular. To solve this issue and to gather the two systems into a single one, we first parametrize two different points of $\mathfrak{C}$ with the same projection by $(x_1, x_2, y + r\sqrt{t})$ and $(x_1, x_2, y - r\sqrt{t})$, with $x_1, x_2, t \in \mathbb{R}$, $y, r$ in $\mathbb{R}^{n-2}$ and $\|r\| = 1$. Then, given any function $f$ from $\mathbb{R}^n$ to $\mathbb{R}$ so that $f = 0$ is one of the $n-1$ hypersurfaces that define $\mathfrak{C}$, we introduce in two smooth functions $S \cdot f$ and $D \cdot f$.

**Definition 42.** *Let $x_1, x_2, t$ be variables in $\mathbb{R}$ and $y, r$ in $\mathbb{R}^{n-2}$. For a smooth function $f : \overline{B} \subset \mathbb{R}^n \to \mathbb{R}$, we define the functions $S \cdot f$ and $D \cdot f$ from $\mathbb{R}^{2n-1}$ to $\mathbb{R}$.*

$$S \cdot f(x_1, x_2, y, r, t) = \begin{cases} \frac{1}{2}[f(x_1, x_2, y + r\sqrt{t}) + f(x_1, x_2, y - r\sqrt{t})], & \text{for } t > 0 \\ f(x_1, x_2, y), & \text{for } t = 0 \end{cases}$$

*and*

$$D \cdot f(x_1, x_2, y, r, t) = \begin{cases} \frac{1}{2\sqrt{t}}[f(x_1, x_2, y + r\sqrt{t}) - f(x_1, x_2, y - r\sqrt{t})], & \text{for } t > 0 \\ \nabla f(x_1, x_2, y) \cdot (0, 0, r), & \text{for } t = 0. \end{cases}$$

When $t > 0$, $S \cdot f$ and $D \cdot f$ return, roughly speaking, the arithmetic mean and difference of $f$ at the above two points, hence they both vanish if and only if the two points are on the hypersurface $f = 0$. When $t = 0$, the two points coincide and $S \cdot f$ and $D \cdot f$ return, roughly speaking, $f$ evaluated at this point and the gradient of $f$ (at that point) scalar the "vertical" vector $(0, 0, r)$; hence, they both vanish if and only if the point is on the hypersurface $f = 0$ and its tangent hyperplane is normal to the plane of projection. It follows that given a curve defined by $P_1 = \cdots = P_{n-1} = 0$, the solutions of the so-called Ball system of all $S \cdot P_i = D \cdot P_i = 0$ is the set of points on the curve that project to nodes and cusps.

**Theorem 43** ([KLMP21*, Theorem 11]). *Consider $P = (P_1, \ldots, P_{n-1}) \in C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$ that satisfies Assumptions $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$. Then, $X = (x_1, x_2, y, r, t) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{n-2} \times \mathbb{R}^{n-2} \times \mathbb{R}$ is a solution of the* Ball system

$$\text{Ball}(P) = \begin{cases} S \cdot P_1(X) = \cdots = S \cdot P_{n-1}(X) = 0 \\ D \cdot P_1(X) = \cdots = D \cdot P_{n-1}(X) = 0 \\ \|r\|^2 - 1 = 0 \end{cases} \tag{4.1}$$

*if and only if $(x_1, x_2)$ is a singular point of $\pi_{\mathfrak{C}}(\mathfrak{C})$.*

Note also that we consider $\sqrt{t}$ instead of $t$ in the parameterization $(x_1, x_2, y \pm r\sqrt{t})$. This is critical for ensuring the regularity of the Ball system when $t = 0$, because this ensures that the linear term of the Taylor expansion of $D \cdot f$ does not vanish. A case analysis of the local shape of the curve at nodes and cusps yields the following regularity theorem.

**Theorem 44** ([KLMP21*, Theorem 27]). *Let $P \in C^\infty(\mathbb{R}^n, \mathbb{R}^{n-1})$ that satisfies Assumptions $\mathcal{A}_1$, $\mathcal{A}_2$,$\mathcal{A}_3$ and $\mathcal{A}_4$, then $P$ satisfies Assumption $\mathcal{A}_5^-$ if and only if* Ball$(P)$ *is regular in $B_{\text{Ball}}$.*

## 4.4 Semi-algorithms to check assumptions and isolate singularities

We present Semi-algorithm 3 that checks the weak assumptions (Definition 41) and, if it terminates, outputs a superset of isolating boxes of the singularities of the plane projection $\pi_{\mathfrak{C}}(\mathfrak{C})$ of $\mathfrak{C}$. We also present Semi-algorithm 4 that checks the strong assumptions and, if it terminates, outputs a set of isolating boxes of the singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$.

The main idea of these semi-algorithms comes from Theorems 43 and 44: Theorem 43 states that, under Assumptions $\mathcal{A}_{1-4}$, the singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$ are the plane projections of the solutions of Ball$(P)$. Theorem 44 states that, under the further Assumption $\mathcal{A}_5^-$, Ball$(P)$ is regular, so we can use certified numerical methods such as interval Newton methods [MKC09] to solve Ball$(P)$. In addtion, in order to verify these assumptions, we use subdivision approaches based on interval arithmetic in a semi-algorithm framework.

### 4.4.1 Interval arithmetic

We recall this standard tool that enables certifications of numerical computations following the notation and definitions by Lin and Yap [LY11]. For a subset $A \subset \mathbb{R}^k$, the set $IA$ is the set of all closed $k$-boxes that are contained in $A$. For the positive integer $m$ and a function $f : A \to \mathbb{R}^m$, the function $\square f : IA \to I\mathbb{R}^m$ is called an inclusion of $f$ if the set $f(\mathfrak{B}) = \{f(x) \mid x \in \mathfrak{B}\}$ is contained in $\square f(\mathfrak{B})$, for all $\mathfrak{B} \in IA$. An inclusion $\square f$ of $f$ is called a box function, if for any descending sequence of closed $k$-boxes $\mathfrak{B}_1 \supset \mathfrak{B}_2 \supset \ldots$ that converges to a point $q \in \mathbb{R}^k$, the sequence $\square f(\mathfrak{B}_1) \supset \square f(\mathfrak{B}_2) \supset \ldots$ converges to $f(q)$. In the rest of this section, we assume that we are given a box function $\square f$ for any function $f$ we consider. The command *subdivide* is applied to a closed $k$-box $\overline{\mathfrak{B}}$, and it returns the set of boxes obtained by bisecting $\overline{\mathfrak{B}}$ in all dimensions.

An interval matrix $\square M$ is a matrix whose coefficients are intervals. It can also be seen as the set of all matrices whose $(i, j)$-th coefficients belong to the $(i, j)$-th interval. The rank of an interval matrix $\square M$, denoted by rank$(\square M)$, is the minimum of the ranks of all the matrices in this set.

### 4.4.2 Alternative assumptions

Since our semi-algorithms mostly work with the Ball system, we reformulate, using Theorems 43 and 44, some of our assumptions in the Ball system space
$B_{\text{Ball}} = \{(x_1, x_2, y, r, t) \mid t \geqslant 0, (x_1, x_2, y + r\sqrt{t}), (x_1, x_2, y - r\sqrt{t}) \in B, \|r\|^2 = 1\}$.

$\aleph_1$ - All solutions of Ball($P$) in $\overline{B}_{\text{Ball}}$ are regular.

$\aleph_2$ - For every solution $X = (x_1, x_2, y, r, t)$ of Ball($P$) in $\overline{B}_{\text{Ball}}$, the points $q_1 = (x_1, x_2, y + r\sqrt{t})$ or $q_2 = (x_1, x_2, y - r\sqrt{t})$ are not in the boundary of $B$.

$\aleph_3$ - No two distinct solutions of Ball($P$) in $\overline{B}_{\text{Ball}}$ have the same plane projection.

The next lemma shows the relation between these new assumptions and those of Section 4.2.

**Lemma 45.** *Let $B$ be an open $n$-box and $P$ be a smooth function from $\overline{B}$ to $\mathbb{R}^{n-1}$ that satisfies Assumption $\mathcal{A}_1$ in $\overline{B}$. Then, Assumptions $\aleph_1$, $\aleph_2$ and $\aleph_3$ are satisfied if and only if Assumptions $\mathcal{A}_2$, $\mathcal{A}_3$ , $\mathcal{A}_4$ and $\mathcal{A}_5^-$ are satisfied in $\overline{B}$.*

### 4.4.3 Checking Assumption $\mathcal{A}_1$ via Semi-algorithm 1: Smoothness of $\mathfrak{C}$

Semi-algorithm 1 is a classical subdivision based algorithm.

**Semi-algorithm 1** Checking Assumption $\mathcal{A}_1$

**Input:** An open $n$-box $B$ and a function $P$ from $\overline{B}$ to $\mathbb{R}^{n-1}$.
**Termination:** If and only if $P$ satisfies Assumption $\mathcal{A}_1$ in $\overline{B}$.
**Output:** True.

```
1: L := {B̄}
2: while L ≠ ∅ do
3:     𝔅 := pop(L)
4:     if 0 ∈ □P(𝔅) and rank(□J_P(𝔅)) < n − 1 then
5:         Subdivide 𝔅 and add its children to L.
6: return True.
```

### 4.4.4 Isolating Ball($P$) solutions via Semi-algorithm 2

Semi-algorithm 2 checks Assumptions $\aleph_1$ and $\aleph_2$ and isolates the solutions of Ball($P$). It outputs a finite set of pairwise disjoint boxes in $\overline{B}_{\text{Ball}}$ such that every box contains at most one solution of Ball($P$) and the union of these boxes contains all solutions of Ball($P$) in $\overline{B}_{\text{Ball}}$. Notice that, by the definition of box functions, for a closed $(2n-1)$-box $\overline{\mathfrak{U}}$, if $0 \notin \square\text{Ball}(P)(\overline{\mathfrak{U}})$, then $\overline{\mathfrak{U}}$ does not contain a solution of Ball($P$), whereas the condition $0 \in \square\text{Ball}(P)(\overline{\mathfrak{U}})$ does not necessarily imply that a solution is in $\overline{\mathfrak{U}}$. This is why the set we are going to find might have unnecessary boxes.

In Step (7), the function $f_{\text{Ball}}$ maps a Ball system solution $X = (x_1, x_2, y, r, t)$ to the two points $(x_1, x_2, y \pm r\sqrt{t})$ in $\mathbb{R}^n$. Steps (8) and (12) are not detailed because they are standard in subdivision algorithms to handle the issue of solutions on or near box boundaries and ensuring that solution boxes are pairwise disjoint. We only sketch below how these steps are done and refer to [Sta95, §5.9.1] [Kea97, XY19] for details. In Step (8), an existence test is performed by evaluating an interval Newton operator on an $\epsilon$-inflation of the box $\mathfrak{U}$. The inflated box $\epsilon$-inflation($\mathfrak{U}$) is certified to contain a solution if its image by the interval Newton operator is contained in the interior of $\epsilon$-inflation($\mathfrak{U}$). When the existence test is positive, the solution may be on the boundary or even outside $\mathfrak{U}$, but still in the interior of $\epsilon$-inflation($\mathfrak{U}$). The side effect is that the same solution may be reported several times when it is on or near a boundary of the subdivision. This issue is then solved in Step (12) as follows. When two boxes in the set *Solutions* intersect, they must report the same solution, and in addition, this solution is in the intersection of the two boxes. In Step (12), we thus compute intersections between boxes and replace intersecting ones by their intersection box. The boxes in the output set *Solutions* are thus pairwise disjoint.

**Semi-algorithm 2** Isolating the solutions of Ball($P$) (under Assumption $\mathcal{A}_1$)

**Input:** An open $n$-box $B$, a function $P$ from $\overline{B}$ to $\mathbb{R}^{n-1}$ such that $P$ satisfies Assumption $\mathcal{A}_1$ in $\overline{B}$ and a $(2n-1)$-open box $\mathfrak{U}_0$ that contains $\overline{B}_{\text{Ball}}$.
**Termination:** If and only if Ball($P$) satisfies Assumptions $\aleph_1$ and $\aleph_2$ in $\overline{B}_{\text{Ball}}$.
**Output:** A list of pairwise disjoint isolating boxes of the solutions of Ball($P$) in $\mathfrak{U}_0$ such that their images by $f_{\text{Ball}}$ lies in $B \times B$.

*Solutions* $= \emptyset$.
$L := \{\mathfrak{U}_0\}$.
**while** $L \neq \emptyset$ **do**
  $\mathfrak{U} := pop(L)$.
  **if** $0 \notin \square\text{Ball}(P)(\overline{\mathfrak{U}})$ or $(\square f_{\text{Ball}}(\overline{\mathfrak{U}})) \cap (\overline{B} \times \overline{B}) = \emptyset$ **then**
    Do nothing ($\mathfrak{U}$ is simply removed from $L$).
  **else if** $\text{rank}(\square J_{\text{Ball}(P)}(\overline{\mathfrak{U}})) = 2n-1$ and $\square f_{\text{Ball}}(\epsilon\text{-inflation}(\overline{\mathfrak{U}}))$[13] $\subset B \times B$ **then**
    **if** $\epsilon$-inflation($\mathfrak{U}$) contains a solution of Ball($P$) **then**
      Add $\epsilon$-inflation($\mathfrak{U}$) to *Solutions*.
  **else**
    Subdivide $\mathfrak{U}$ and add its children to $L$.
Remove duplicates in *Solutions*.
**return** *Solutions*

### 4.4.5 Singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$, Semi-algorithm 3 and 4

For an open $n$-box $B$ and a smooth function $P$ from $\overline{B}$ to $\mathbb{R}^{n-1}$, Semi-algorithm 3 stops if and only if $P$ satisfies the weak assumptions in $\overline{B}$ and then it returns a set of isolating boxes of all the singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$, plus possibly other spurious disjoint boxes.

To identify the possible cusp points in the set $U$ returned by Semi-algorithm 3, one may wish to solve independently the Ball system with the additional constraint $t = 0$. Unfortunately, in this case we have an over-determined system and thus we cannot certify its solutions numerically. In the special case of a silhouette curve in $\mathbb{R}^3$, we were able to identify cusp points with numerical algorithms [IMP16*, Lemmas 9 & 10], but we leave as a conjecture its generalization for $n > 3$.

On the other hand, for curves that satisfy the strong assumptions, $\mathcal{A}_5$ ensures that there are no cusps in the projection, which is equivalent to Ball($P$) having no solutions on the hyperplane $t = 0$. Hence, if Assumptions $\mathcal{A}_{1-5}$ hold, we can refine the boxes output by Semi-Algorithm 3 until no box intersects $t = 0$. Boxes in the half-space $t < 0$ correspond to imaginary points in $\mathbb{C}^n$. Then the boxes satisfying $t > 0$ are all the isolating boxes of the nodes of $\pi_{\mathfrak{C}}(\mathfrak{C})$ and are reported by Semi-algorithm 4

**Semi-algorithm 3** Checking the weak assumptions and computing a superset of the singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$

**Input:** An open $n$-box $B$ and a smooth function $P$ from $\overline{B}$ to $\mathbb{R}^{n-1}$.
**Termination:** If and only if $P$ satisfies Assumptions $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$ and $\mathcal{A}_5^-$ in $\overline{B}$.
**Output:** $N$, a list of certified node singularities: a list of boxes in $\mathbb{R}^{2n-1}$ whose projections in $\mathbb{R}^2$ contain each a single node of $\pi_{\mathfrak{C}}(\mathfrak{C})$.
$U$, a list of uncertified singularities: a list of boxes in $\mathbb{R}^{2n-1}$ whose projections in $\mathbb{R}^2$ contain each at most one node or one cusp of $\pi_{\mathfrak{C}}(\mathfrak{C})$.
The union of all these projected 2D boxes contains all the singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$.

Check Assumption $\mathcal{A}_1$ (Semi-algorithm 1).
Compute a closed $(2n-1)$-box $\overline{\mathfrak{U}}_0$ that contains $\overline{B}_{\text{Ball}}$.
$L :=$ output of Semi-algorithm 2.
Keep refining all boxes $\overline{\mathfrak{U}} \in L$ until no triplets of boxes overlap in projection. This ensures Assumption $\aleph_3$.
$N :=$ boxes of $L$ that lie in the halfspace $t > 0$.
$U :=$ boxes of $L$ that intersect the hyperplane $t = 0$.
**return** $N$ and $U$.

Semi-algorithm 4 stops if and only if $P$ satisfies the strong assumptions in $\overline{B}$ and then it returns a set of isolating boxes of all the singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$.

[13] For a box $\mathfrak{U}$ and $\epsilon > 0$, $\epsilon$-inflation($\mathfrak{U}$) is the box that has the same center as $\mathfrak{U}$ and its width is that of $\mathfrak{U}$ multiplied by $(1+\epsilon)$. The box $\epsilon$-inflation($\mathfrak{U}$) thus contains $\mathfrak{U}$, the exact value of $\epsilon$ is not important for the algorithm and it is usually set to 0.1 in subdivision algorithms [Rum10].

**Semi-algorithm 4** Checking Assumptions $\mathcal{A}_{1-5}$ and isolating the singularities of $\pi_{\mathfrak{C}}(\mathfrak{C})$

**Input:** An open $n$-box $B$ and a smooth function $P$ from $\overline{B}$ to $\mathbb{R}^{n-1}$.
**Termination:** If and only if $P$ satisfies Assumptions $\mathcal{A}_{1-5}$ in $\overline{B}$.
**Output:** A list of boxes in $\mathbb{R}^{2n-1}$ whose projections in $\mathbb{R}^2$ are isolating boxes of the singularities $\pi_{\mathfrak{C}}(\mathfrak{C})$ (all singularities are in some boxes and each box contains a unique singularity).

1: $N, U :=$ output of Semi-Algorithm 3.
2: **for** all $\overline{\mathfrak{U}} \in U$ **do**
3: Keep refining $\overline{\mathfrak{U}}$ until it does not intersect the hyperplane $t = 0$ and discard it if it lies in the half-space $t < 0$.
4: **return** $N \cup U$.

## 4.5 Topology encoding via combinatorial maps

In [IMP18*], we proposed a data structure for performing reliable location with respect to the projection of a smooth curve in $\mathbb{R}^3$. We restricted our study to the three dimensional case but we believe it can be extended along the same lines to higher dimensions. The encoding of the topology and the geometry we propose can thus be seen as a reliable tool to validate a robot configuration, to check if the clearance with respect to special configurations is large enough, or to check whether during a motion the robot passes through a singularity or not.

For the geometric approximation of the space curve $\mathfrak{C}_3$ and thus its projection $\pi(\mathfrak{C}_3)$, we compute a sequence of 3-dimensional boxes enclosing all the connected components of $\mathfrak{C}_3$. We use a reliable numerical solver to find intersections of the curve with the boundary of the input box and at least one point on each connected component of $\mathfrak{C}_3$ using a critical point method. We then use these points as starting points for a tracking algorithm which is an adapted version of the one in [MGGJ13]. Figure 4.4 illustrates the geometric enclosure of the silhouette of the torus and its projection.

We want our topological encoding to be able to answer location queries. The curve $\pi(\mathfrak{C}_3)$ decomposes the box of interest $\mathbf{B}_0$ in connected components of dimension two that we call faces. For a point $p$ in $\mathbf{B}_0$ but not on $\pi(\mathfrak{C}_3)$, one wants to find the face to which $p$ belongs. Such queries can be answered by considering a combinatorial encoding of the embedding of the projected curve, *i.e.* the subdivision of $\mathbf{B}_0$ induced by the curve $\pi(\mathfrak{C}_3)$. We use Combinatorial Maps and their extensions to the non-connected case, the eXtended Planar Maps [Köt02], to encode embeddings of plane curves. In a first step, we focus on singularities and isolate the singular points of $\pi(\mathfrak{C}_3)$ in boxes of width as small as desired as explained in Section 4.4. Then the second step is to compute the topology in these isolating boxes. The last step uses the tracking of the space curve to connect its smooth branches to the isolating boxes of the singular and critical points and thus compute a combinatorial map for each connected component. The construction of the extended planar map encoding the embedding of $\pi(\mathfrak{C}_3)$ is incremental on the connected components, Figure 4.5 illustrates the combinatorial maps of the curve of Figure 4.4. The point location algorithm is based on a vertical ray shooting principle but eventually needs to compute intersections of the space curve $\mathfrak{C}_3$ with planes.

## 4.6 Experiments

Based on the semi-algorithms presented in Section 4.4, our PhD student, George Krait, developed the software *Isolating_singularities*[14] written in Python. Given a curve $\mathfrak{C}$ in $\mathbb{R}^n$, our software finds the singularities of its plane projection. In addition, the software provides a tool to visualize the projected curve and its singularities. Our software is based on interval arithmetic, interval evaluations of analytic functions and an interval solver. We use the following libraries, Python-FLINT [Joh12] and Ibexsolve [Nin15] for these tasks.

The first example is a sparse but reasonably-high-degree algebraic equations in $\mathbb{R}^4$. It should be stressed that, up to our knowledge, this example is out of reach by algebraic methods since the bivariate equation defining the 2D projection has a very high degree.

The second example revisits a classical problem in robotics design and analysis. This problem consists of determining the configurations in which a robot is allowed to move without damaging it. This example also emphasizes the ability of our software to process non-algebraic curves.

[14] The software is accessible at https://github.com/gkrait/Isolating_singularities.

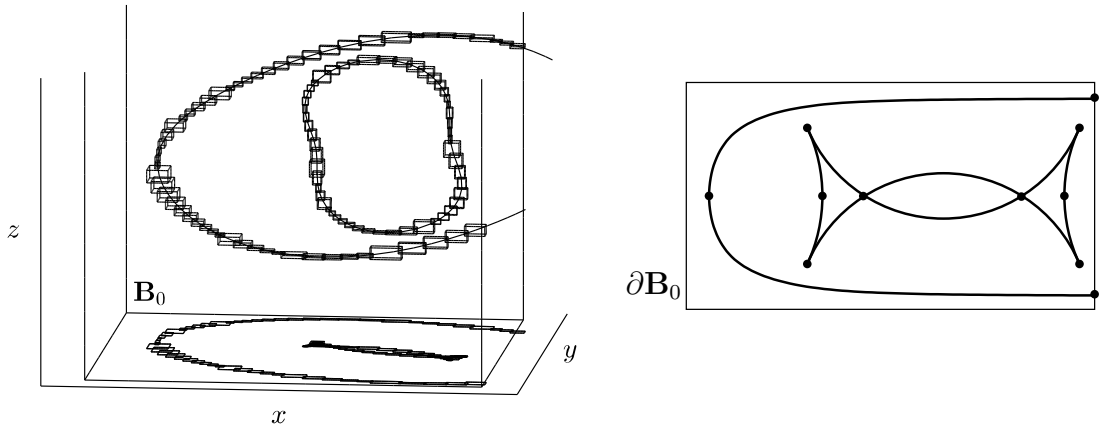



Figure 4.4: Left: a $\delta$-approximation of $\mathfrak{C}_3$ in $\mathbf{B}_0 \times \mathbb{R}$, and a $\delta$-approximation of $\pi(\mathfrak{C}_3)$ in $\mathbf{B}_0$. Right: an embedding of the graph $G$ for the apparent contour of a torus. Black circles represent the points associated to the vertices of $G$, and thick curves the curves associated with its edges. Thin lines represent the boundary of $\mathbf{B}_0$.

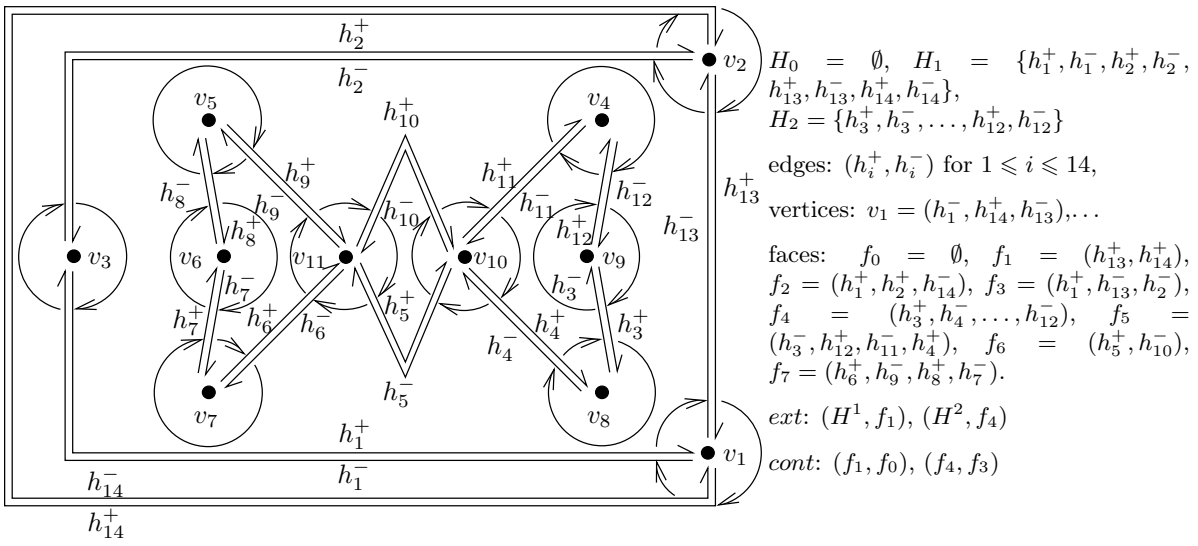



Figure 4.5: An XPMap encoding the topology of the apparent contour of the torus in $\mathbf{B}_0$. Each edge is represented by a pair of half-edges which is a cycle of the permutation $\alpha$. Curved arrows around vertices represent cycles of the permutation $\varphi$.

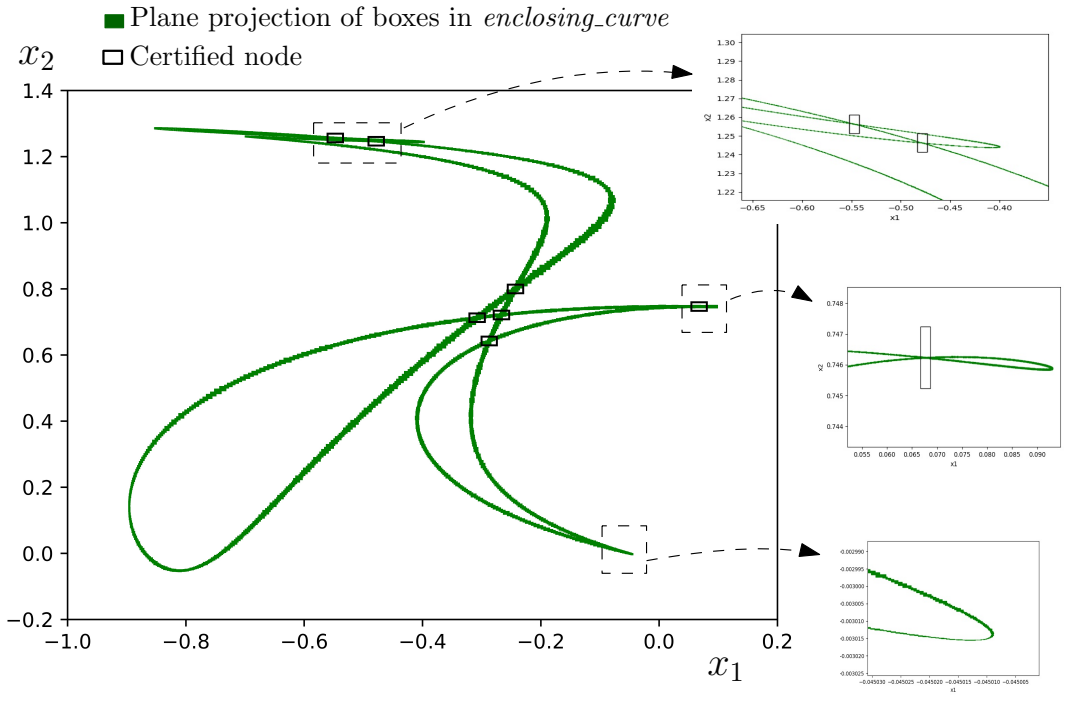



Figure 4.6: High degree algebraic curve in $\mathbb{R}^4$ generating 7 nodes.

### 4.6.1 High degree algebraic curve in $\mathbb{R}^4$

The goal of this experiment is to emphasize the genericity of the assumptions and the efficiency of our software in the sparse polynomial case. The curve $\mathfrak{C}$ in defined in the 4-box $B = (-1, 0.2) \times (-0.2, 1.4) \times (-10, 10)^2$ and is the zero set of three polynomials of degrees 17, 15 and 13, respectively that have a unique monomial of highest degree (which is monic) and 9 other random monomials of degrees at most 2 with integer coefficients in $(-25, 25)$.

$$\begin{aligned} P = [&x_1^{17} - 14x_1^2 - 7x_1x_3 - 7x_2^2 - 22x_2x_4 - x_3x_4 - 19x_4^2 + 8x_1 - 14x_3 + 9,\\ &x_2^{15} + 8x_1x_3 - 14x_1x_4 - 15x_2^2 + 16x_2x_3 + 8x_2x_4 + 2x_3^2 + 13x_4^2 + 11x_1 + 11x_2,\\ &x_3^{13} + 17x_1^2 - 15x_1x_2 + 4x_1x_3 - 20x_1x_4 + 2x_2^2 - 10x_2x_3 + 4x_2x_4 + 20x_4^2 - 23x_2]. \end{aligned}$$

Figure 4.6 illustrates the 7 nodes of the projection of $\mathfrak{C}$, the running time of our software was 5 seconds.

Note that since $P$ is polynomial, the implicit equation of $\pi_{\mathfrak{C}}(\mathfrak{C})$ can be computed using elimination theory and its singularities can then be isolated using algebraic solvers. However, the implicit equation we obtained for $\pi_{\mathfrak{C}}(\mathfrak{C})$ is defined by an irreducible bivariate polynomial of degree 442 with 51074 monomials. Isolating the singularities of such a high-degree polynomial is then a real challenge.

### 4.6.2 Parallel singularities of the RRRRR robot

Caro et al. [CCG+12] studied the *generalized aspects* of two degrees-of-freedom parallel robots, that are components of the configuration space where a robot can be moved without damaging it. These generalized aspects are bounded by the curve of *parallel singularities*. They proposed certified interval methods to compute inner approximations of these generalized aspects by staying away from the curve of parallel singularities. Our contribution can be seen as a continuation of this work where we attack the problem from another point of view: we focus instead on certifying the curve of parallel singularities and it projection.

We apply our software on the RRRRR robot that consists of five links $AP'$, $P'P$, $PP''$, $P''B$ and $BA$ connected by five joints $A, P', P, P''$ and $B$, see Figure 4.7. The links' lengths are respectively $8, 5, 8, 5$ and 9. The angles $BAP'$ and $ABP''$ are the commands $q_1, q_2$ respectively that vary in the interval $(-\pi, \pi)$. The pose variables $x_1, x_2$ are the coordinates of the point $P$ and vary in the interval $[-15, 15]$. The equations of the configuration space are:

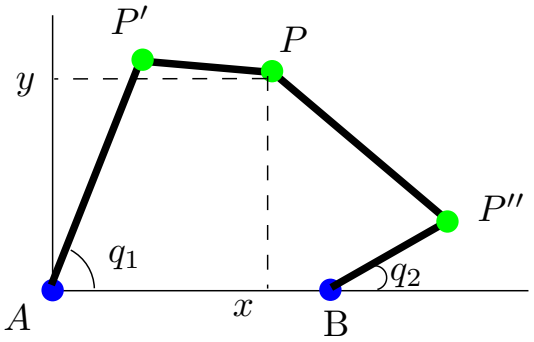


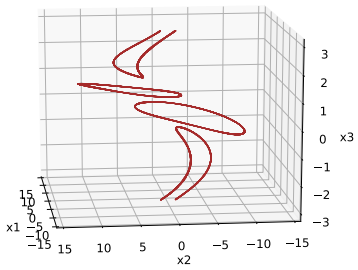

Figure 4.7: $\underline{R}RRR\underline{R}$ robot.

$$\begin{cases} (x_1 - 8\cos(q_1))^2 + (x_2 - 8\sin(q_1))^2 - 25 = 0, \\ (x_1 - 9 - 5\cos(q_2))^2 + (x_2 - 5\sin(q_2))^2 - 64 = 0. \end{cases} \tag{4.2}$$

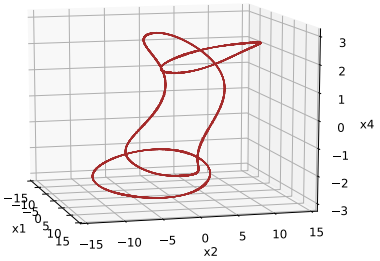
Thus, the configuration space is a surface in the four-dimensional vector space represented by the variables $x_1, x_2, q_1$ and $q_2$. The parallel singularities are defined as the silhouette of the configuration space with respect to the projection on the command space $(q_1, q_2)$, the additional equation is thus the Jacobian of $P$ with respect to the pose variables $(x_1, x_2)$. In other words, the function $P$ that define the curve of parallel singularities $S_{parallel}$ is:

$$\begin{aligned} P = &[(x_1 - 8\cos(q_1))^2 + (x_2 - 8\sin(q_1))^2 - 25, \\ &(x_1 - 9 - 5\cos(q_2))^2 + (x_2 - 5\sin(q_2))^2 - 64, \\ &(2x_1 - 16\cos(q_1))(2x_2 - 10\sin(q_2)) - (2x_2 - 16\sin(q_1))(2x_1 - 10\cos(q_2) - 18)]. \end{aligned} \tag{4.3}$$

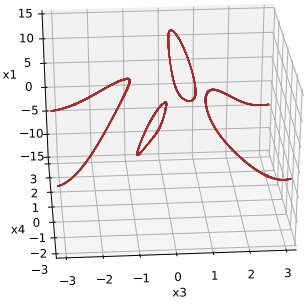
Figures 4.8 illustrates the projections of the parallel singularities on different variable sub-spaces. Figure 4.9 shows the plane projection of $S_{parallel}$ onto the pose space $(x_1, x_2)$ (brown boxes) in addition to the boxes certified to enclose the nodes (black boxes).



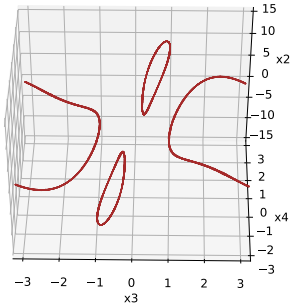

(a) The projection of $S_{parallel}$ on $\mathbb{R}^3$ with respect to $(x_1, x_2, q_1)$.



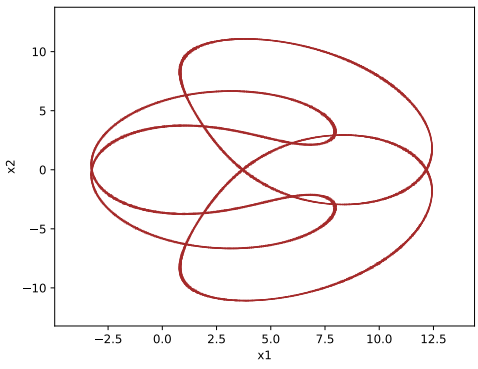

(b) The projection of $S_{parallel}$ on $\mathbb{R}^3$ with respect to $(x_1, x_2, q_2)$.



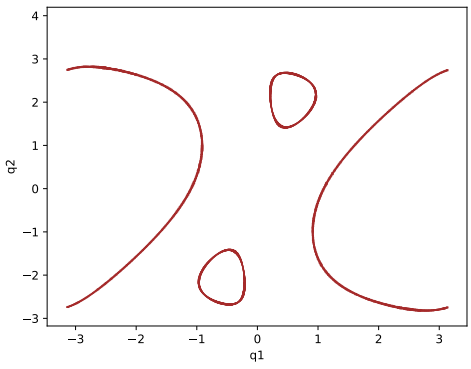

(c) The projection of $S_{parallel}$ on $\mathbb{R}^3$ with respect to $(q_1, q_2, x_1)$.




(d) The projection of $S_{parallel}$ on $\mathbb{R}^3$ with respect to $(q_1, q_2, x_2)$.




(e) The plane projection of $S_{parallel}$ on $\mathbb{R}^3$ with respect to $(x_1, x_2)$.




(f) The plane projection of $S_{parallel}$ on $\mathbb{R}^3$ with respect to $(q_1, q_2)$.

Figure 4.8: Illustration of the projections of $S_{parallel}$ on different variable sub-spaces.

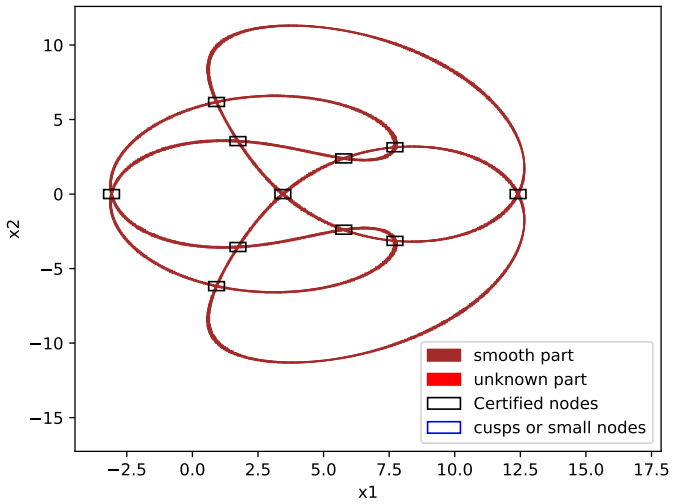



Figure 4.9: Certified nodes, reported by our software, of the plane projection of the curve of parallel singularities $S_{parallel}$.

# Chapter 5

# Research project

My objects of research are singular manifolds, that is, in low dimensions, curves and surfaces with self-intersections. Such objects either appear explicitly by parameterizations in computer aided design or implicitly as the solutions of a system of equations modeling a constrained physical or mathematical problem. The smooth part of a manifold encodes the typical behavior of a physical system whereas its singular loci highlight a particular behavior. Depending on the application, the singular loci must be identified either to avoid them (in robotics design this is where the robot may break) or to use them for a global understanding of the problem (crossing a singularity by moving a parameter changes the global behavior) [JCB+18, MZ19]. For a *reliable* description of the manifold, one can distinguish two types of guarantees:

- topological: all the components and all the singularities (isolated points or self-intersections) must be reported;
- geometrical: the manifold must be approximated, within a given distance, by a piecewise parametric (often linear) one.

Such a description is necessary for efficiently computing a good visualization since different methods are applied to smooth and singular parts. In addition, the topological information is computed only once and different levels of detail are obtained by refining only the geometric approximation.

When the manifold is defined by polynomial equations, symbolic algebraic methods apply and enable the description of singular loci with certification (for surfaces see e.g. [BKS10, DMR12, BDRH+13, FGT15] and references therein). These methods can handle any types of singularities and their complexity in terms of global parameters (degree of the polynomials and bitsize of the coefficients) is well understood. However their high complexities limit their usage in practice.

Even though symbolic methods provide certification for singular loci and thus the computation of the topology, numerical computations are necessary for the geometric approximation of the manifold. In addition, numerical methods are not restricted to the polynomial setting, trigonometric or exponential functions can be dealt with. Another advantage of numerical methods is the fact that their complexity adapts to the local complexity of the manifold restricted to the studied domain, this makes them efficient in practice. On the other hand, the certification of numerical methods a priori only holds for non-singular manifolds [BT06].

It is worth mentioning that current software tools do not allow correct visualisation of singular surfaces, see e.g. Figure 5.1. Following are several tracks I will explore symbolic methods, numerical methods or a combination of both and their application to visualization in engineering and mathematics. As in my previous work, this program applies my mathematical background together with my experience for the case of curves to the design and analysis of algorithms and the development of visualization software for surfaces.

**Algorithmic and complexity for singular surfaces**

- The certification of the topology of smooth surfaces can be achieved with subdivision algorithms together with interval arithmetic tools. The basic idea is that, numerically, a variable can never be certified to be zero (or any specific value) but can be certified not to take the value zero (by bounding its values with an interval and refining this interval). In this setting, a singular case is identified by a zero test and thus is not doable numerically. The challenge is thus to intertwine further

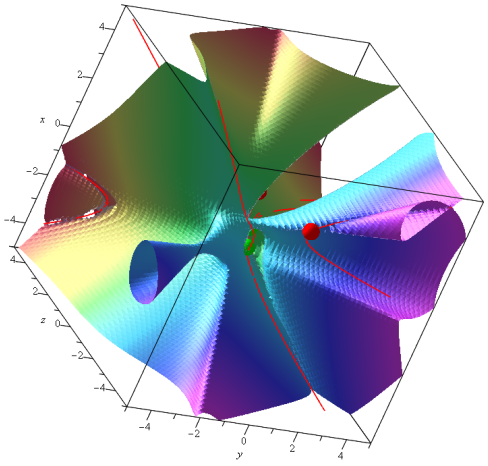


Figure 5.1: A non-certified visualization via the Maple command `implicitplot` of a singular surface in $\mathbb{R}^3$ given as the projection of a smooth surface in $\mathbb{R}^4$, it contains auto-intersections (red curve), triple points (green) and cross-caps (red) where a curve of auto-intersections vanishes.

symbolic/numeric approaches to take advantage of the certification of the symbolic ones and the efficiency of the numerical ones. My approach is to study classes of manifolds with restricted types of singularities to which a specific analysis is conducted to "desingularize" them, i.e. a symbolic preprocessing is applied to convert a zero test into a stable numerical non-zero test.

I am already working on a numerical algorithm for the topology of singular surfaces that are projections of smooth ones in higher dimension [DMP19*]. I am exploring the domain of singularity theory to better understand what are the expected generic singularities. Note that the singular locus of a surface can be quite complicated and is in general a 3D singular curve itself, see e.g. Figure 5.1.

- Subdivision algorithms, like the Marching Cube, divide the space into a grid and evaluate a function at all grid points. For the specific case of a polynomial function, evaluating it at several points simultaneously is faster than sequentially using a fast Fourier tranform. The goal is to introduce a combination of multi-evaluation via the FFT and interval arithmetic to achieve both efficiency and certification. This is challenging due to over-estimations intrinsic to interval arithmetic. Preliminary results obtained in the univariate case will be generalized to the multivariate case.

- Voronoi diagrams encode nearest neighbor information, they are classical for point sets and have a linear structure. For lines in 3D, the Voronoi diagram has a non-linear structure and includes patches of quadrics [ELLSED09]. I contributed to the classification of degenerate cases of configurations of 3 lines [EGL+09*]. One goal is to better understand the complexity of the Voronoi diagram of $n$ lines and more generally of a polyhedron, surprisingly the best known bounds are $\Omega(n^2)$ and $O(n^{3+\epsilon})$. Another goal is the design of an algorithm for its computation.

- Numerical algorithms are desirable since one can observe that in practice their efficiency correlates with the local complexity of the object in the region of interest. The challenge is thus to better understand this correlation and find the geometric parameters to quantify the complexity. Recent results have appeared in this direction for analyzing subdivision algorithms for smooth manifolds either in the worst-case [BGT17] or in a probabilist setting [CETC19]. A long-term difficult goal is to extend these results to our numerical algorithms computing the topology of singular curves and surfaces. In particular combining the ideas of continuous amortization for subdivision algorithms [Bur16] and of conditioning for numerical computation [CB13] seems promising.

**Applications and software**

- A visualization based on reliable computation is important to help mathematicians develop new intuitions regarding complex objects that are currently difficult to picture such as Seifert surfaces

associated to knots [vWC06] or 3-manifolds. In particular, I will continue my collaboration with F. Rouillier and his Ouragan Inria team at the *Institut Mathématique de Jussieu*. In low dimensional topology and geometry, mathematicians of this team are eager to experiment with numerical or symbolic computations that we can propose.

For researchers in robotics, a certified visualization is a valuable tool to easily see the effect of some parameters and avoid singular situations [JCB+18, MZ19]. My goal is to provide software prototypes of our theoretical work [DMP19*, KLMP21*] to this community.

- An efficient reliable visualization software for surfaces is much more difficult to achieve than for curves and several solutions will be tested. Our SubdivisionSolver software is already a first building block that will be combined together with curve tracking and possibly the surface mesher of the CGAL library [JAYB15].

- Isotop, my software tool for the visualization of 2D curves is already mature. Still some improvements are expected by testing which algorithmic design choices, suggested by my last theoretical advances [BLM+16*, LPR17*], are practically efficient. At least as important is to improve the diffusion and impact. I am working on a transfert to MapleSoft that will ensure diffusion to a large audience.

# Chapter 6

# Publications

**Practice in publications.** Alphabetical order of authors is the practice in both the computational geometry and computer algebra communities. Conference papers usually do not contain all proofs and thus are not always really reviewed for correctness. It is the practice to publish journal versions containing all proofs and additional material.

## 6.1 International journals


[CP05b*] Frédéric Cazals and Marc Pouget. Estimating Differential Quantities using Polynomial fitting of Osculating Jets. ***Computer Aided Geometric Design***, 22(2):121–146, 2005. hal

[CP05a*] Frédéric Cazals and Marc Pouget. Differential topology and geometry of smooth embedded surfaces: selected topics. *International Journal of Computational Geometry and Applications*, Vol. 15, No. 5:511–536, 2005. hal

[CFPR06*] Frédéric Cazals, Jean-Charles Faugère, Marc Pouget, and Fabrice Rouillier. The implicit structure of ridges of a smooth parametric surface. ***Computer Aided Geometric Design***, 23(7):582–598, 2006. hal

[CP08*] Frédéric Cazals and Marc Pouget. Jet fitting 3: A Generic C++ Package for Estimating the Differential Properties on Sampled Surfaces via Polynomial Fitting. ***ACM Transactions on Mathematical Software***, 35(3):1–24, 2008. hal

[CLP+10*] Jinsan Cheng, Sylvain Lazard, Luis Mariano Peñaranda, Marc Pouget, Fabrice Rouillier, and Elias Tsigaridas. On the topology of real algebraic plane curves. *Mathematics in Computer Science*, 4(1):113–137, 2010. hal

[BLPR15*] Yacine Bouzidi, Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Separating linear forms and Rational Univariate Representations of bivariate systems. ***Journal of Symbolic Computation***, 68(0):84–119, May 2015. hal

[BLM+16*] Yacine Bouzidi, Sylvain Lazard, Guillaume Moroz, Marc Pouget, Fabrice Rouillier, and Michael Sagraloff. Solving bivariate systems using Rational Univariate Representations. ***Journal of Complexity***, 37:34–75, 2016. hal

[GLMP16*] Marc Glisse, Sylvain Lazard, Julien Michel, and Marc Pouget. Silhouette of a random polytope. ***Journal of Computational Geometry***, 7(1):86–99, 2016. hal

[IMP16*] Rémi Imbach, Guillaume Moroz, and Marc Pouget. A certified numerical algorithm for the topology of resultant and discriminant curves. ***Journal of Symbolic Computation***, 80, Part 2:285–306, 2016. hal

[LPR17*] Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Bivariate triangular decompositions in the presence of asymptotes. ***Journal of Symbolic Computation***, 82:123–133, 2017. hal

[IMP18*] Rémi Imbach, Guillaume Moroz, and Marc Pouget. Reliable Location with Respect to the Projection of a Smooth Space Curve. *Reliable Computing*, 26:13–55, 2018. hal

[IPY20a*] Rémi Imbach, Marc Pouget, and Chee Yap. Clustering Complex Zeros of Triangular Systems of Polynomials. *Mathematics in Computer Science*, June 2020. hal

[KLMP21*] George Krait, Sylvain Lazard, Guillaume Moroz, and Marc Pouget. Certified numerical algorithm for isolating the singularities of the plane projection of generic smooth space curves. *Journal of Computational and Applied Mathematics*, 2021. hal

## 6.2 Reviewed international conferences

[CP03b*] Frédéric Cazals and Marc Pouget. Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets. In L. Kobbelt, P. Schröder, and H. Hoppe, editors, *Eurographics Symposium on Geometry Processing*, pages 177–187, Aachen, Germany, June 2003. Eurographics. hal

[BLL+08*] David Bremner, Jonathan Lenchner, Giuseppe Liotta, Christophe Paul, Marc Pouget, Svetlana Stolpner, and Stephen Wismath. A Note on $\alpha$-Drawable k-Trees. In *CCCG'08: Canadian Conference on Computational Geometry*, pages 23–27, Canada, September 2008. hal

[CLP+09*] Jinsan Cheng, Sylvain Lazard, Luis Mariano Peñaranda, Marc Pouget, Fabrice Rouillier, and Elias Tsigaridas. On the topology of planar algebraic curves. In John Hershberger and Efi Fogel, editors, ***25th annual symposium on Computational geometry - SCG 2009***, pages 361–370, Aarhus, Denmark, June 2009. ACM. hal

[BLPR13b*] Yacine Bouzidi, Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Rational Univariate Representations of Bivariate Systems and Applications. In ***ISSAC - 38th International Symposium on Symbolic and Algebraic Computation***, pages 109–116, Boston, United States, June 2013. hal

[BLPR13c*] Yacine Bouzidi, Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Separating Linear Forms for Bivariate Systems. In ***ISSAC - 38th International Symposium on Symbolic and Algebraic Computation***, pages 117–124, Boston, United States, June 2013. hal

[BLM+14*] Yacine Bouzidi, Sylvain Lazard, Guillaume Moroz, Marc Pouget, and Fabrice Rouillier. Improved algorithm for computing separating linear forms for bivariate systems. In ***ISSAC - 39th International Symposium on Symbolic and Algebraic Computation***, Kobe, Japan, July 2014. hal

[IMP15a*] Rémi Imbach, Guillaume Moroz, and Marc Pouget. Numeric and Certified Isolation of the Singularities of the Projection of a Smooth Space Curve. In *Proceedings of the 6th International Conferences on Mathematical Aspects of Computer and Information Sciences*, Berlin, Germany, October 2015. Springer LNCS. hal

[IPY19*] Rémi Imbach, Marc Pouget, and Chee Yap. Clustering Complex Zeros of Triangular System of Polynomials. In: CASC - 21st International Workshop on Computer Algebra in Scientific Computing. LNCS, Springer, 2019. hal

[DMP19*] Sény Diatta, Guillaume Moroz, and Marc Pouget. Reliable Computation of the Singularities of the Projection in $\mathbb{R}^3$ of a Generic Surface of $\mathbb{R}^4$. In *MACIS 2019 - Mathematical Aspects of Computer and Information Sciences*, Gebze-Istanbul, Turkey, November 2019. hal

## 6.3 Books chapters

[CFPR08*] Frédéric Cazals, Jean-Charles Faugère, Marc Pouget, and Fabrice Rouillier. Ridges and Umbilics of Polynomial Parametric Surfaces. In B. Juttler and R. Piene, editors, *Geometric Modeling and Algebraic Geometry*, pages 141–159. Springer, 2008. hal

## 6.4 Other international publications (non-selective conferences, software documentation)

[CP07a*] Frédéric Cazals and Marc Pouget. Approximation of Ridges and Umbilics on Triangulated Surface Meshes. In CGAL Editorial Board, editor, *CGAL User and Reference Manual 3.3 edition*. CGAL Editorial Board, 2007. hal

[CP07b*] Frédéric Cazals and Marc Pouget. Estimation of Local Differential Properties. In CGAL Editorial Board, editor, *CGAL User and Reference Manual 3.3 edition*. CGAL Editorial Board, 2007. hal

[CLP+08*] Jinsan Cheng, Sylvain Lazard, Luis Mariano Peñaranda, Marc Pouget, Fabrice Rouillier, and Elias Tsigaridas. On The Topology of Planar Algebraic Curves. In *24th European Workshop on Computational Geometry - EuroCG 2008*, pages 213–216, Nancy, France, March 2008. hal

[EGL+09*] Hazel Everett, Christian Gillot, Daniel Lazard, Sylvain Lazard, and Marc Pouget. The Voronoi diagram of three arbitrary lines in R3. In *25th European Workshop on Computational Geometry - EuroCG'09*, pages 297–300, Bruxelles, Belgium, March 2009. hal

[BLPR11*] Yacine Bouzidi, Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. New bivariate system solver and topology of algebraic curves. In *27th European Workshop on Computational Geometry - EuroCG 2011*, Morschach, Switzerland, March 2011. hal

[KLMP19*] George Krait, Sylvain Lazard, Guillaume Moroz, and Marc Pouget. Numerical Algorithm for the Topology of Singular Plane Curves. In *35th European Workshop on Computational Geometry, EuroCG'19*, Utrecht, Netherlands, March 2019. hal

## 6.5 Other publications

[Pou05*] Marc Pouget. *Geometry of surfaces: from the estimation of local differential quantities to the robust extraction of global differential features*. PhD Thesis, Université Nice Sophia Antipolis, December 2005. tel

[CP05c*] Frédéric Cazals and Marc Pouget. Topology driven algorithms for ridge extraction on meshes. Research Report RR-5526, INRIA, 2005. hal

# Bibliography


[ABRW96] M.-E Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Multiplicities and idempotents for zerodimensional systems. In *Algorithms in Algebraic Geometry and Applications*, volume 143 of *Progress in Mathematics*, pages 1–20. Birkhäuser, 1996.

[ACM84] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition ii: An adjacency algorithm for the plane. *SIAM J. Comput.*, 13(4):878–889, 1984.

[AM88] D. Arnon and S. McCallum. A polynomial time algorithm for the topological type of a real algebraic curve. *J. Symbolic Computation*, 5:213–236, 1988.

[AMW08] L. Alberti, B. Mourrain, and J. Wintz. Topology and arrangement computation of semi-algebraic planar curves. *Comput. Aided Geom. Des.*, 25(8):631–651, 2008.

[BCGY12] M. Burr, S. W. Choi, B. Galehouse, and C. K. Yap. Complete subdivision algorithms ii: Isotopic meshing of singular algebraic curves. *Journal of Symbolic Computation*, 47(2):131 – 152, 2012.

[BDRH+13] Gian Mario Besana, Sandra Di Rocco, Jonathan D Hauenstein, Andrew J Sommese, and Charles W Wampler. Cell decomposition of almost smooth real algebraic surfaces. *Numerical Algorithms*, 63(4):645–678, 2013.

[BEKS13] Eric Berberich, Pavel Emeliyanenko, Alexander Kobel, and Michael Sagraloff. Exact symbolic-numeric computation of planar algebraic curves. *Theor. Comput. Sci.*, 491:1–32, 2013.

[BFM+01] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. In *Proc. 9th Annual European Symposium on Algorithms*, volume 2161 of *LNCS*, pages 254–265. Springer-Verlag, 2001.

[BGT17] Michael A. Burr, Shuhong Gao, and Elias Tsigaridas. The complexity of an adaptive subdivision method for approximating real curves. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 61–68, New York, NY, USA, 2017. ACM.

[Bir75] J. Birman. *Braids, Links, and Mapping Class Groups.* Princeton University Press, 1975.

[BKS10] Eric Berberich, Michael Kerber, and Michael Sagraloff. An efficient algorithm for the stratification and triangulation of an algebraic surface. *Comput. Geom. Theory Appl.*, 43(3):257–278, 2010.

[BL13] C. Beltrán and A. Leykin. Robust certified numerical homotopy tracking. *Foundations of Computational Mathematics*, pages 1–43, 2013.

[Bou14] Y. Bouzidi. *Solving bivariate algebraic systems and topology of plane curves.* Theses, Université de Lorraine, March 2014.

[BPR06] S. Basu, R. Pollack, and M.-R. Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics.* Springer-Verlag, 2nd edition, 2006.

[BR90] R. Benedetti and J.J. Risler. *Real Algebraic and Semi-algebraic Sets, Actualites Mathematiques.* Hermann, 1990.

[Bro01] C. W. Brown. Improved projection for cylindrical algebraic decomposition. *J. Symb. Comput.*, 32(5):447–465, 2001.

[BSGY08] M. Burr, S.W.Choi, B. Galehouse, and C. Yap. Complete subdivision algorithms, ii: Isotopic meshing of singular algebraic curves. In *International Symposium on Symbolic and Algebraic Computation Symposium - ISSAC*, 2008.

[BSS03] A. Bostan, B. Salvy, and É. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. *Applicable Algebra in Engineering, Communication and Computing*, 14(4):239–272, 2003.

[BT06] Jean-Daniel Boissonnat and Monique Teillaud, editors. *Effective Computational Geometry for Curves and Surfaces*. Springer-Verlag, Mathematics and Visualization, 2006.

[Bur16] Michael A. Burr. Continuous amortization and extensions: With applications to bisection-based root isolation. *Journal of Symbolic Computation*, 77:78 – 126, 2016.

[CB13] Felipe Cucker and Peter Bürgisser. *Condition : The Geometry of Numerical Algorithms*. Springer, 2013.

[CCG+12] Stéphane Caro, Damien Chablat, Alexandre Goldsztejn, Daisuke Ishii, and Christophe Jermann. A Branch and Prune Algorithm for the Computation of Generalized Aspects of Parallel Robots. In *18th international conference on Principles and Practice of Constraint Programming*, LNCS, pages 867–882, Québec City, Canada, 2012.

[CETC19] Felipe Cucker, Alperen A. Ergür, and Josue Tonelli-Cueto. Plantinga-vegter algorithm takes average polynomial time. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '19, pages 114–121, New York, NY, USA, 2019. ACM.

[CLO92] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms an introduction to computational algebraic geometry and commutative algebra*. Undergraduate texts in mathematics. Springer-Verlag New York-Berlin-Paris, 1992.

[CLO05] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Number 185 in Graduate Texts in Mathematics. Springer, New York, 2nd edition, 2005.

[CR88] M. Coste and M. F. Roy. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *J. Symb. Comput.*, 5(1/2):121–129, 1988.

[DDR+18] Daouda Niang Diatta, Sény Diatta, Fabrice Rouillier, Marie-Françoise Roy, and Michael Sagraloff. Bounds for polynomials on algebraic numbers and application to curve topology, 2018.

[Ded06] J.P. Dedieu. *Points fixes, zéros et la méthode de Newton*. Mathématiques et Applications. Springer, 2006.

[Dem00] M. Demazure. *Bifurcations and catastrophes: geometry of solutions to nonlinear problems*. Universitext. Springer, Berlin, New York, 2000. École polytechnique.

[DET07] D. I. Diochnos, I. Z. Emiris, and E. P. Tsigaridas. On the complexity of real solving bivariate systems. In C. W. Brown, editor, *Proc. Int. Symp. Symbolic and Algebraic Computation*, pages 127–134, Waterloo, Canada, 2007.

[DET09] Dimitrios I. Diochnos, Ioannis Z. Emiris, and Elias P. Tsigaridas. On the asymptotic and practical complexity of solving bivariate systems over the reals. *J. Symb. Comput.*, 44(7):818–835, 2009.

[Dia09] Daouda Niang Diatta. *Calcul effectif de la topologie de courbes et surfaces algébriques réelles*. Theses, Université de Limoges, September 2009.

[DL14] N. Delanoue and S. Lagrange. A numerical approach to compute the topology of the apparent contour of a smooth mapping from $R^2$ to $R^2$. *Journal of Computational and Applied Mathematics*, 271:267–284, 2014.

[DMR12] Daouda Niang Diatta, Bernard Mourrain, and Olivier Ruatta. On the isotopic meshing of an algebraic implicit surface. *Journal of Symbolic Computation*, 47(8):903 – 925, 2012.

[EK08a] Arno Eigenwillig and Michael Kerber. Exact and efficient 2d-arrangements of arbitrary algebraic curves. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms - SODA*, pages 122–131, San Francisco, USA, January 2008. ACM-SIAM, ACM/SIAM.

[EK08b] A. Eigenwilling and M. Kerber. Exact and efficient 2d-arrangements of arbitrary algebraic curves. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA08)*, pages 122–131, San Francisco, USA, January 2008. ACM-SIAM, ACM/SIAM.

[EKK+05] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. A Descartes Algorithm for Polynomials with Bit-Stream Coefficients. In V. Ganzha, E. Mayr, and E. Vorozhtsov, editors, *CASC*, volume 3718 of *LNCS*, pages 138–149. Springer, 2005.

[EKW07] A. Eigenwillig, M. Kerber, and N. Wolpert. Fast and exact geometric analysis of real algebraic plane curves. In *Proc. Int. Symp. Symbolic and Algebraic Computation - ISSAC*, pages 151–158, 2007.

[ELLSED09] Hazel Everett, Daniel Lazard, Sylvain Lazard, and Mohab Safey El Din. The voronoi diagram of three lines. *Discrete & Computational Geometry*, 42(1):94–130, Jul 2009.

[Fen92] H. Feng. *Decomposition and Computation of the Topology of Plane Real Algebraic Curves*. Ph.d. thesis, The Royal Institute of Technology, Stockholm, 1992.

[FGT15] E. Fortuna, P. Gianni, and B.M. Trager. Computation of topological invariants for real projective surfaces with isolated singularities. *Journal of Symbolic Computation*, 68, Part 2:131 – 166, 2015. Effective Methods in Algebraic Geometry.

[GLS01] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner free alternative for solving polynomial systems. *J. of Complexity*, 17(1):154–211, 2001.

[GVEK96] L. González-Vega and M. El Kahoui. An improved upper complexity bound for the topology computation of a real algebraic plane curve. *J. Complexity*, 12(4):527–544, 1996.

[GVN02] L. González-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Computer Aided Geometric Design*, 19(9), 2002.

[Hon96] H. Hong. An efficient method for analyzing the topology of plane real algebraic curves. *Mathematics and Computers in Simulation*, 42(4–6):571–582, 1996.

[JAYB15] Clément Jamin, Pierre Alliez, Mariette Yvinec, and Jean-Daniel Boissonnat. Cgalmesh: A generic framework for delaunay mesh generation. *ACM Trans. Math. Softw.*, 41(4):23:1–23:24, October 2015.

[JCB+18] Ranjan Jha, Damien Chablat, Luc Baron, Fabrice Rouillier, and Guillaume Moroz. Workspace, joint space and singularities of a family of delta-like robot. *Mechanism and Machine Theory*, 127:73 – 95, 2018.

[Joh12] Fredrik Johansson. Python-flint, 2012. http://fredrikj.net/python-flint/.

[Kah03] M. El Kahoui. An elementary approach to subresultants theory. *J. Symb. Comput.*, 35(3):281–292, 2003.

[Kea97] R. B. Kearfott. Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear systems. *SIAM Journal on Scientific Computing*, 18(2):574–594, 1997.

[Ker06] M. Kerber. Analysis of real algebraic plane curves. Master's thesis, MPII, 2006.

[Köt02] Ullrich Köthe. Xpmaps and topological segmentation-a unified approach to finite topologies in the plane. In *International Conference on Discrete Geometry for Computer Imagery*, pages 22–33. Springer, 2002.

[KS12] Michael Kerber and Michael Sagraloff. A worst-case bound for topology computation of algebraic curves. *Journal of Symbolic Computation*, 47(3):239 – 258, 2012.

[KS15] Alexander Kobel and Michael Sagraloff. On the complexity of computing with planar algebraic curves. *J. Complex.*, 31(2):206–236, 2015.

[LMMRS11] X. Li, M. Moreno Maza, R. Rasheed, and É. Schost. The modpn library: Bringing fast polynomial arithmetic into maple. *Journal of Symbolic Computation*, 46(7):841 – 858, 2011.

[LPY04] C. Li, S. Pion, and C. Yap. Recent progress in exact geometric computation. *J. of Logic and Algebraic Programming*, 64(1):85–111, 2004. Special issue on"Practical Development of Exact Real Number Computation".

[LR01] T. Lickteig and M-F. Roy. Sylvester-Habicht Sequences and Fast Cauchy Index Computation. *J. Symb. Comput.*, 31(3):315–341, 2001.

[LSVY14] Jyh-Ming Lien, Vikram Sharma, Gert Vegter, and Chee Yap. Isotopic arrangement of simple curves: An exact numerical approach based on subdivision. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 277–282, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[LSVY20] Jyh-Ming Lien, Vikram Sharma, Gert Vegter, and Chee Yap. Isotopic Arrangement of Simple Curves: an Exact Numerical Approach based on Subdivision. *arXiv e-prints*, page arXiv:2009.00811, September 2020.

[LVZ06] A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theoretical Computer Science*, 359(13):111 – 122, 2006.

[LY11] L. Lin and C. Yap. Adaptive isotopic approximation of nonsingular curves: the parameterizability and nonlocal isotopy approach. *Discrete & Computational Geometry*, 45(4):760–795, 2011.

[MC02] S. McCallum and G. E. Collins. Local box adjacency algorithms for cylindrical algebraic decompositions. *J. Symb. Comput.*, 33(3):321–342, 2002.

[MGGJ13] B. Martin, A. Goldsztejn, L. Granvilliers, and C. Jermann. Certified parallelotope continuation for one-manifolds. *SIAM J. Numerical Analysis*, 51(6):3373–3401, 2013.

[MKC09] Ramon E Moore, R Baker Kearfott, and Michael J Cloud. *Introduction to interval analysis*. Siam, 2009.

[MPS+06] B. Mourrain, S. Pion, S. Schmitt, J.-P. Técourt, E. P. Tsigaridas, and N. Wolpert. Algebraic issues in Computational Geometry. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, Mathematics and Visualization, chapter 3. Springer, 2006.

[MS16] Esmaeil Mehrabi and Eric Schost. A softly optimal monte carlo algorithm for solving bivariate polynomial systems over the integers. *Journal of Complexity*, 34:78–128, 2016.

[MSW15] K. Mehlhorn, M. Sagraloff, and P. Wang. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. *J. Symb. Comput.*, 66(0):34–69, 2015.

[MZ19] Andreas Muüller and Dimiter Zlatanov, editors. *Singular Configurations of Mechanisms and Manipulators*. Springer, 2019.

[Neu90] A. Neumaier. *Interval methods for systems of equations*. Cambridge University Press, 1990.

[Nin15] Jordan Ninin. Global Optimization based on Contractor Programming: an Overview of the IBEX library. In *MACIS*, Berlin, Germany, November 2015.

[OWM83] T. Ojika, S. Watanabe, and T. Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations. *Journal of Mathematical Analysis and Applications*, 96(2):463 – 479, 1983.

[Pan02] V. Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *J. Symb. Comput.*, 33(5):701–733, 2002.

[PV04] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In *SGP '04: Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 245–254, 2004.

[PV06] Simon Plantinga and Gert Vegter. Computing contour generators of evolving implicit surfaces. *ACM Trans. Graph.*, 25(4):1243–1280, 2006.

[Rei97] Daniel Reischert. Asymptotically fast computation of subresultants. In *Proceedings of the 1997 international symposium on Symbolic and algebraic computation*, ISSAC '97, pages 233–240, New York, NY, USA, 1997. ACM.

[Rou99] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *J. of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[Rum10] Siegfried M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 5 2010.

[Sak91] T. Sakkalis. The topological configuration of a real algebraic curve. *Bull. Austrl. Math. Soc.*, 43:37–50, 1991.

[SF90] T. Sakkalis and R. Farouki. Singular points of algebraic curves. *J. Symb. Comput.*, 9(4):405–421, 1990.

[Sny92] John M. Snyder. Interval analysis for computer graphics. *SIGGRAPH Comput. Graph.*, 26(2):121–130, July 1992.

[Sta95] V. Stahl. *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations*. PhD thesis, Johannes Kepler University, Linz, Austria, 1995.

[Str06] A. Strzebonski. Cylindrical algebraic decomposition using validated numerics. *J. Symb. Comput.*, 41(9):1021–1038, 2006.

[SW05] R. Seidel and N. Wolpert. On the exact computation of the topology of real algebraic curves. In *Proc 21st ACM Symposium on Computational Geometry*, pages 107–115, 2005.

[Tei73] B. Teissier. Cycles évanescents, sections planes et conditions de Whitney. (french). In *Singularités à Cargèse (Rencontre Singularités Géom. Anal., Inst. Études Sci., Cargèse, 1972)*, number 7–8 in Asterisque, pages 285–362. Soc. Math. France, Paris, 1973.

[vWC06] Jarke J. van Wijk and Arjeh M. Cohen. Visualization of seifert surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):485–496, July 2006.

[vzGG13] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge Univ. Press, Cambridge, U.K., 3rd edition, 2013.

[XY19] Juan Xu and Chee Yap. Effective subdivision algorithm for isolating zeros of real systems of equations, with complexity analysis. In *Proceedings of the 2019 International Symposium on Symbolic and Algebraic Computation*, ISSAC '19, pages 355–362, New York, NY, USA, 2019. ACM.

[Yap00] C.K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, Oxford-New York, 2000.