HAL
open science

Exact Algebraic and Geometric Computations for Parametric Curves

Christina Katsamaki

To cite this version:

Christina Katsamaki. Exact Algebraic and Geometric Computations for Parametric Curves. Computational Geometry [cs.CG]. Sorbonne Université, 2023. English. NNT : 2023SORUS206 . tel-04167904v2

HAL Id: tel-04167904

https://theses.hal.science/tel-04167904v2

Submitted on 8 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

École doctorale de sciences mathématiques de Paris centre

Thèse de doctorat

Discipline : Mathématiques

présentée par

Christina Katsamaki

Exact Algebraic and Geometric Computations for Parametric Curves

---

dirigée par Fabrice Rouillier et Elias Tsigaridas

Soutenue le 21 juin 2023 devant le jury composé de :

Mme Marianne Akian	Inria Saclay-Ile-de-France et CMAP, Ecole polytechnique	présidente
M. Laureano Gonzalez-Vega	CUNEF Universidad	rapporteur
Mme Evelyne Hubert	Centre Inria d'Université Côte d'Azur	rapporteure
Mme Ariane Mézard	Sorbonne Université	examinatrice
M. Marc Pouget	Université de Lorraine, CNRS, Inria, LORIA Nancy	examinateur
M. Fabrice Rouillier	Centre Inria de Sorbonne Université	directeur
M. Elias Tsigaridas	Centre Inria de Sorbonne Université	co-directeur

Institut de mathématiques de Jussieu-
Paris Rive gauche. UMR 7586.
Boîte courrier 247
4 place Jussieu
75 252 Paris Cedex 05

Université de Paris.
École doctorale de sciences
mathématiques de Paris centre.
Boîte courrier 290
4 place Jussieu
75 252 Paris Cedex 05

Acknowledgements

These few lines are to offer a glimpse of my gratitude to all who contributed, in math and/or beyond, to the completion of this thesis.

First, to my advisors, Elias and Fabrice, who not only guided me through the scientific labyrinth of this thesis, but also made the journey enjoyable. I am sincerely thankful for the countless lessons learned, all your ideas and time that you generously devoted, and, math aside, the support and compassion you have shown.

To Evelyne Hubert and Laureano Gonzalez-Vega for their time and effort in reviewing my work, as well as for their kind words. I appreciate their valuable feedback. To Marianne Akian, Ariane Mézard and Marc Pouget for accepting to be part of the jury of this thesis. I extend special thanks to Marc for his valuable suggestion that contributed to a result in this thesis.

To Michael Sagraloff for generously dedicating his time to discuss and read my work, and for sharing his insightful ideas.

To Josué, for being this unique person to work with (and to live nearby!). To Mattias, for always boosting my confidence and morale. A thought also to Grégoire; it was a pleasure to have you around. To Charles, to whom I wish all the luck with his PhD. To Zafeirakis and Tolis; I am delighted to have had the opportunity to collaborate with you as co-authors.

To my tutors, Jean-Claude Bajard and Marc Chardin for diligently monitoring the progress of my thesis throughout these years. A special note of appreciation to Jean-Claude for his eagerness to assist me during the entire process.

To Ariela Briani and FSMP that co-funded my PhD and to all the staff of IMJ-PRG and Inria that concretely made this possible. Notably to Evariste, Sakina and Laurence.

My days in Jussieu all these years would not have been the same if it was not for the *couloir des doctorants*. Your camaraderie, support, and shared experiences (in the cantine, at the coffee breaks, as well as in the office, in parks, in quais or in bars) have made my days at Jussieu truly memorable and enjoyable.

To all the co-bureaux I have had throughout these years; Jean Michel who has always been my favorite, Thibaut who very quickly became a friend, Franzi who became a friend

a bit after, Raphael who was always pleasant to be around, to Thomas and Benoit for your frequent visits to our office (and our couch). To the new recruits: Camille, Joaquin, and Xenia; thank you for keeping the office alive. To the girls next door; Anna (Florio), Anna, Perla, Chiara, to the girls of the other next door; Mahya and Grace, and to Eva. Especially to Mahya and Grace who welcomed me so warmly and formed the cutest team to be around. To all the rest of the doctorants (or not still) as they come to my mind; to Mathieu and Vadim, to Raphaël and Jacques that accompanied me throughout the last leg of submitting this thesis, to Christophe, Sylvain, Maxime, Arnaud, Nelson, Thiago, Joao, Odylo, Lucas, Mattias, Pietro, Germain, Francesco, Haowen, Juan Felipe. Also to the colleagues from the neighboring tower, Jorge, Rémi and Georgy, that was fun being around you at the conferences.

To my companions these parisian years (and wherever the next years will be); to Alex, Antonis, Zoi, Nikiforos for all the aperos or the long nights, the coffees or the dinners, or just for being there. To Meropi and Christianna, and to Sofy and Spyros, who never really left. Sofy, whether in Paris or in Athens (or even in Argos), I'm glad having you as a friend.

To some of my favorite people in Greece; Vagia, Vassiliki, Eleni, Pipos. I am grateful for having grown up with you, and growing up a bit more day by day. For all our summers and winters and springs, that cannot fully make up for the months we now spent apart. To Eirini, Raf, Stavros, Christos, Moumis, Bilo that I also love and miss. To Polina, that is always in some part of the world, yet by my side, and to my long-distance (for now, I hope) friend Dimitris.

To my mom, my dad, and my sister, thank you for doing everything within your power to support me. I would not have been able to do this without you.

And lastly, to my special person, Jean Michel. For all the reasons of the world, from the ones that I could write in these acknowledgements, to all the ones that have no place here, I am extremely happy having you in my life.

Résumé

Cette thèse propose des algorithmes pour résoudre des problèmes de géométrie computationnelle non linéaire liés aux courbes paramétriques. Elle se concentre spécifiquement sur le calcul de la topologie des courbes dans $\mathbb{R}^n$ et de l'enveloppe convexe des courbes dans $\mathbb{R}^2$ et $\mathbb{R}^3$, sans avoir recours à l'implicitation. Les algorithmes effectuent des calculs exacts avec des nombres réels grâce à des bornes de séparation et à l'arithmétique des intervalles.

Pour le calcul de la topologie, l'algorithme proposé fonctionne pour des courbes de n'importe quelle dimension et calcule un graphe abstrait qui est isotopique à la courbe dans l'espace de plongement. La complexité binaire est analysée et trouvée pour être linéaire dans la dimension de l'espace ambiant.

Pour le calcul de l'enveloppe convexe, la thèse présente des algorithmes pour les courbes planes et les courbes spatiales. L'enveloppe convexe est un ensemble semi-algébrique et une représentation exacte de sa frontière est obtenue par une combinaison de segments de droite et d'arcs de la courbe pour le cas en 2D, et de triangles et de patchs de surface pour le cas en 3D. La description de la frontière est calculée pour chaque cas ainsi que des estimations de la complexité binaire. Le calcul se réduit à la résolution univariée et bivariée et à l'isolement des racines d'un polynôme univarié avec des coefficients dans une extension de corps multiple.

Pour fournir des bornes supérieures asymptotiques pour le problème de l'enveloppe convexe, la thèse fournit des bornes de complexité binaire pour l'isolement des racines d'un polynôme $F \in L[Y]$, où L est une extension algébrique multiple de $\mathbb{Q}$. Des bornes amorties sur la séparation des racines de F sont utilisées et deux solutions algorithmiques sont présentées; une formelle, qui est basée sur la représentation rationnelle univariée, et une solution numérique, qui est également certifiée.

Mots-clés

Courbe paramétrique, topologie, enveloppe convexe, système polynomial, isolation de racines, corps d'extension, complexité binaire.

Abstract

This thesis proposes algorithms for solving non-linear computational geometry problems related to parametric curves. Specifically, it focuses on computing the topology of curves in $\mathbb{R}^n$ and the convex hull of curves in $\mathbb{R}^2$ and $\mathbb{R}^3$, without resorting to implicitization. The algorithms perform exact computations with real algebraic numbers through separation bounds and interval arithmetic.

For the topology computation, the proposed algorithm works for curves in any dimension and computes an abstract graph that is isotopic to the curve in the embedding space. The bit-complexity is analyzed and found to be linear in the dimension of the ambient space.

For the convex hull computation, the thesis presents algorithms for plane and space curves. The convex hull is a semi-algebraic set and an exact representation of its boundary is obtained through a combination of line segments and arcs of the curve for the 2D case, and of triangles and surface patches for the 3D case. The boundary description is computed for each case along with bit-complexity estimates. The computation reduces to univariate and bivariate solving and to isolating roots of a univariate polynomial with coefficients in a multiple algebraic field extension.

To provide asymptotic upper bounds for the convex hull problem, the thesis provides bit-complexity bounds for the root isolation of a polynomial $F \in L[Y]$, where L is a multiple algebraic extension of $\mathbb{Q}$. Aggregate bounds for the separation of the roots of F are employed and two algorithmic solutions are presented; a formal one based on Rational Univariate Representations and a numerical one that is also certified.

Keywords

Parametric curve, topology, convex hull, polynomial system, root isolation, extension field, bit-complexity

Contents

1 Introduction **11**

1.1 Related work and Contribution 16

1.1.1 Root isolation in a multiple field extension 16

1.1.2 Topology of parametric curves 18

1.1.3 Convex hull of plane and space parametric curves 20

1.2 Organization of the thesis 23

2 Preliminaries **25**

2.1 Notation 25

2.2 Computations with polynomials 26

2.2.1 Basic operations 26

2.2.2 Evaluation of polynomials 28

2.3 Greatest Common Divisor, Resultant and Subresultants 32

2.3.1 Greatest Common Divisor 32

2.3.2 Resultant 34

2.3.3 Subresultant sequences 37

2.4 Univariate root isolation 38

2.4.1 Bounds on univariate polynomials 38

2.4.2 Root isolation 42

2.5 Zero-dimensional polynomial systems 43

2.5.1 Rational Univariate Representation 46

2.6 Modular algorithms 49

Polynomial System Solving

3 Isolating Roots in a Multiple Field Extension **53**

3.1 Introduction 53

3.2 Prerequisite: Multipoint evaluation at a Cartesian product in $\mathbb{C}^n$ 56

3.3 Amortized bounds for polynomials in a multiple field extension 57

3.4 Solving in a multiple field extension 64

3.5 Computing with a unique field extension 70
3.5.1 Operations in the quotient algebra $\mathcal{A}$ 71
3.5.2 Lucky primes for the RUR computation 74
3.5.3 Separating Linear Form: Las-Vegas algorithm 75
3.5.4 Computing an RUR 78
3.5.5 From multiple to simple field extension 81
3.6 Application: Sum Of Square roots of integers problem 84

Algorithms on Parametric Curves

4 Topology of Parametric Curves 87
4.1 Introduction 87
4.2 Algebraic tools 91
4.3 Rational curves 93
4.3.1 Proper parametrization 94
4.3.2 Normal parametrization 95
4.4 Special points on the curve 96
4.4.1 Computation and Complexity 98
4.5 `PTOPO`: Topology and Complexity 103
4.6 Isotopic embedding for plane and space curves 108
4.7 Multiplicities and characterization of singular points 112
4.8 Implementation and Examples 115

5 Convex Hull of Parametric Curves 117
5.1 Introduction 117
5.2 Background on rational curves 124
5.3 SUPPORT: a predicate for parametric curves in $\mathbb{R}^n$ 125
5.4 Convex hull of parametric curves in $\mathbb{R}^2$ 125
5.4.1 Algorithm 126
5.4.2 Complexity analysis 129
5.5 Convex hull of parametric curves in $\mathbb{R}^3$ 132
5.5.1 Algorithm 134
5.5.2 Complexity analysis 139
5.6 Implementation and Examples 142

Bibliography 146

Chapter 1

Introduction

Since its development, computational geometry focused on solving geometric and combinatorial problems involving linear objects such as points, line segments, polygons and polyhedra [PS85, Ede04, dBCvKO08]. However, curved objects play a crucial role in real-world applications and in science and engineering as well; in addition, these traditional methods are inadequate for addressing the complex, algebraic nature of contemporary challenges in fields such as geometrical modeling [JP08, CRE01], computer-aided design [Sal06, PLH+05], robot motion planning [PASM17, STM19], reverse engineering [VMC97] and learning theory [BBL+17, OA21].

Linearization is a commonly used method to effectively address problems involving non-linear geometric objects. For, example, one approach is to discretize the objects into meshes [HDPS11, DLLP08a, DLLP08b, DLLP08c]. However, this method often results in a considerable loss of accuracy, which may cause issues depending on the specific application. Furthermore, meshing can lead to a combinatorial explosion, which can decrease efficiency compared to directly solving the problems on the non-linear geometric objects themselves.

To effectively tackle problems with non-linear objects without resorting to linear approximations, we must turn to real and complex algebraic geometry as the natural framework for analyzing these objects [Buc88, BT06, Tei06]. In recent years, there has been a surge in research towards algorithmic algebraic geometry, which aims to transform the insightful concepts of unconstructive algebraic geometry into practical algorithms for geometric reasoning [BT06]. Since non-linear objects are often described as the (real) zero set of polynomials, finding the roots of polynomial systems is of particular interest and it can provide valuable information on their structure and properties [CCC+05, CLO05, Stu02]. Advancements in computation facilitated the development of effective algorithms for problems related to Voronoi diagrams [ELLED07, ETMT06], robot-motion planning [Can88, Mou93, Buc87], curve topology [DDR+22, BT06], geometric modelling [Emi05], geometric optimization [HDLP22, Baj88, Las01] and many others.

Parametric curves

The focus of this thesis is on computational geometry problems related to rational parametric curves. A rational curve in $\mathbb{R}^n$ is parametrized by

$$\begin{aligned} \phi: \quad & \mathbb{R} \to \mathbb{R}^n \\ & t \mapsto \big(\phi_1(t), \ldots, \phi_n(t)\big) = \left(\frac{p_1(t)}{q_1(t)}, \ldots, \frac{p_n(t)}{q_n(t)}\right), \end{aligned}$$

where p_i, q_i are univariate polynomials. We call $\phi(t)$ a *parametrization* of the curve. Parametric curves have long been an important subject in computational algebra and geometry [SW99] and continue to receive attention in current research [Sed86a, CKPU11, BLY19, SWPD08, RSS13, SWPD08]. Efficient algorithms for working with parametric curves are of particular interest because parametric representations are frequently used in different fields such as computer modeling and computer-aided geometric design [FGS10], control theory [Kur12], machine learning [SMC20] or chemical engineering [CKLS18].

The parametric representation is to be juxtaposed to the implicit representation of a curve, where it given as the zero set of several polynomials. The implicit representation is more general since every rational parametric curve can be implicitly represented, but the converse is not true [SAG84]. To work with parametric curves, one common approach is to convert them to implicit form using implicitization algorithms that have been extensively studied, such as those described in [SC95, BLY19] and the references therein. However, it is also important to be able to manipulate parametric curves directly, without converting them to implicit form. This allows for instance, for easier visualization and identification of points on the curve, especially for high-dimensional curves.

In what follows, we will introduce the two geometrical problems that are addressed in this thesis.

Topology. Computing the topology of a parametric curve in $\mathbb{R}^n$ refers to the computation of an abstract graph that is *isotopic* [BT06, p. 184] to the curve in the embedding space. The vertices of the embedded graph correspond to points on the curve and they are connected with straight line segments that can be continuously deformed into the curve without any topological changes (Fig. 1.1). The graph must include vertices that correspond to self-intersections, cusps and other important points on the curve, such as extreme points with respect to the coordinate directions. All these points are essential for precisely capturing the curve's geometry and producing a certified visualization of plane and space curves.

The computation of the topology of an implicit plane curve has been extensively studied ([DDR+22, KS15] and references therein), but in higher dimensions, where the curve is the zero locus of several polynomials, there does not exist general methods (see [CJP+21] for a

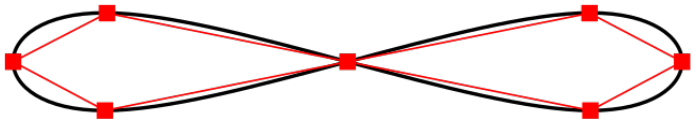


Figure 1.1: A plane curve (in black) and a graph isotopic to it (in red).

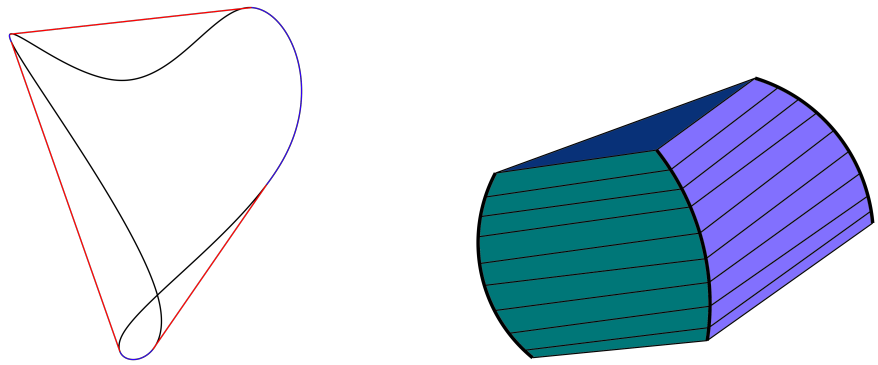
recent result on space curves). In contrast, working directly with the parametrization of the curve allows for a general algorithm for any dimension but on the same time presents unique challenges. Arcs of a parametric curve are described by the corresponding intervals of the parameter t and, for instance, choosing an appropriate parameter interval that includes all the important topological features, such as singular and extreme points, is not always straightforward [ADT10]. Additionally, visualizing the curve using symbolic computational tools may result in missing points and branches, which need to be addressed [AR07, Sen02].

Convex hull. For any subset of $\mathbb{R}^n$, where $n \geq 1$, its convex hull is defined as the smallest convex set that contains it, or in other words, as the intersection of its supporting halfspaces [Brø83, Thm.4.5]. We focus on parametric curves in $\mathbb{R}^2$ and in $\mathbb{R}^3$; in particular, we consider the problem of computing an exact description of the boundary of the convex hull. Note that the boundary is a semi-algebraic set (since it is the boundary of the intersecting halfspaces that define the convex hull) and thus it cannot be described by polynomial equations.

For plane curves, the boundary of the convex hull consists of a combination of smooth arcs of the curve and line segments joining two points on the curve (Fig. 5.1a). For space curves, it is a combination of triangles and parts of a ruled surface that is also developable (Fig. 5.1b), meaning that it can be flattened on the plane in a way that the length of any curve drawn on the surface is preserved [SF00].

Convex hull computations are fundamental in computational geometry with direct applications in motion planning [ZST11, YLLF11], computer vision [dFT90], geometric modeling systems [EMP10, GVNPD$^+$04]; to mention few of them. Convex hull of non-linear objects arise naturally in optimization [Las09, BPT12] and learning theory [CS01b, SNW11] because they are useful, among other things, in optimizing a linear function over a nonlinear object.

This thesis tackles these two problems: computing the topology of parametric curves in general dimension, and computing the convex hull of plane and space parametric curves. The algebraic formulation of these problems involves polynomial systems and finding their solutions is a pivotal issue for the effectiveness of our algorithms. We design algorithms that are certified to produce accurate numerical, geometrical, and combinatorial results. We also take into consideration the structure of the input to ensure that our solutions are tailored to the specific problem at hand. To this scope, we use state-of-the-art techniques for



(a) (b)

Figure 1.2: (a) A plane curve and its convex hull; the boundary consists of the red line segments and the blue arcs of the curve. (b) A curve in $\mathbb{R}^3$ (in bold black) and its convex hull; the visible part of the boundary consists of the blue triangle and the ruled surface patches in purple and green.

computing resultants and solving polynomial systems and equations, but we also develop our own algebraic tools that are customized to the special structure of the problems. In particular, we put efforts into designing an efficient algorithm for isolating the roots of a univariate polynomial that has coefficients within a multiple algebraic field extension.

Before explaining in detail our contributions (see Sec. 1.1), since we aim for methods that always return the correct result for all types of input, we give a synopsis on how exact computations can be performed.

Exact computations

The fact that computers cannot represent all real numbers and can only provide approximations can lead to significant errors in computational geometry [KMP+08]. These errors may result in incorrect decisions, crashes, and other algorithm failures. Therefore, algorithms in computational geometry must take measures to ensure the correctness of geometric results and avoid any inconsistencies.

One common source of error is evaluating the sign of an expression with real numbers using naive computer arithmetic. This evaluation can be wrong due to numerical errors, especially when the expression is zero or close to zero, where even a small error can produce the wrong sign. To guarantee correct results in computations with real numbers, one approach is to perform exact computation; for example by relying on separation bounds and/or interval arithmetic. Under the exact computation paradigm computations are done with numbers of arbitrary precision and the algorithm never makes errors in its decisions during computation [YE94].

In the exact computation paradigm, we can compute exactly using algebraic numbers.

An algebraic number is the root of a univariate polynomial with integer coefficients. One way to represent algebraic numbers is using Thom encoding [CR88]. Here, we represent an algebraic number $\alpha \in \mathbb{C}$ by the *isolating interval representation*. When $\alpha \in \mathbb{R}$ (resp. $\mathbb{C}$), its isolating interval representation includes a square-free polynomial which vanishes at α and a rational interval (resp. Cartesian products of rational intervals) containing α and no other root of this polynomial [Yap99]. For example $\sqrt{2}$ is represented as $(X^2 - 2, [1, 2])$.

The endpoints of the isolating intervals in the representation of algebraic numbers are rationals or multiprecision numbers. Then, computing exactly with algebraic numbers is done by providing algorithms for the different operations (such as sign computation or comparison) using interval arithmetic, combined with separation bounds, e.g. [BFM$^+$01]. As an illustration, if an algebraic expression's sign needs to be evaluated, interval approximations of its value are computed iteratively. If the interval includes zero, then the sign cannot be decided and the computation is retried with greater precision. This iteration continues until the resulting interval either does not include zero or has width smaller than a separation bound. This bound is a minimum estimate of the smallest non-zero value (in absolute terms) that the expression can attain. One drawback of this approach is that it can be slow when the expression to be evaluated is equal to zero.

Efficiency

Obtaining efficient implementations in the exact computation paradigm is challenging, especially when dealing with curved objects. Curved objects trigger algebraic operations such as solving systems of polynomials, which makes certifying results more involved. Of central interest is determining the *computational complexity* of our algorithms. For this reason, we describe briefly the computational model considered in this thesis.

Computational model. We consider the Turing machine model of computation, which is essentially a multitape Turing machine in which binary strings encode the input objects. The main complexity measures in this model are the number of steps of a computation and also the space occupied on the tape for the computation. The latter is determined by size of representation of the objects involved in the computation, i.e., the bitsize (cf. next paragraph). The Turing machine model can be contrasted to the real Random Access Machine (real RAM) model in which the size of representation is not a measure of complexity; the complexity of an algorithm is then given by summing up the number of steps needed to execute the algorithm, and computations with real numbers are exact and of unit cost. We refer to [Yap99, BCSS98, Pap07, AB09] for more details on computational models.

Bitsize. The bitsize of an integer p is the number of bits needed to represent it, that is $\lfloor \log p \rfloor + 1$ (log denotes the logarithm of base 2). If p is a rational number, then its

bitsize is the maximum bitsize of its numerator and denominator. We define the bitsize of a polynomial with integer or rational coefficients as the maximum bitsize of its coefficients.

We use the $\mathcal{O}$ (big-O) notation for asymptotic upper bounds. For the complexity analysis of algorithms, we differentiate between the *arithmetic* and the *bit*-complexity. The arithmetic complexity is complexity in the real RAM model and is equal to the number of arithmetic operations, assuming that each operation has unitary cost, regardless the size of the operands. When the operands are integers or rationals, the bit-complexity takes into account the growth of the coefficients in the operations, by considering the number of bit-operations performed. For asymptotic upper bounds in the bit-complexity model, we use the notation $\mathcal{O}_B$. We use $\widetilde{\mathcal{O}}$, resp. $\widetilde{\mathcal{O}}_B$, to ignore (poly-) logarithmic factors.

For deterministic algorithms, the complexities are worst-case complexities. There are also two types of randomized algorithms [MR95] considered in this thesis. A *Las-Vegas* algorithm is a randomized algorithm that always produces the correct result or it informs about the failure. The run time of a Las Vegas algorithm is probabilistic and it differs depending on the input. We refer to it as *expected* run time. Contrastingly, a *Monte-Carlo* algorithm is a randomized algorithm that has deterministic run time but produces output that may be incorrect with a certain (typically small) probability. It is possible to turn a Monte Carlo algorithm into a Las Vegas algorithm by additionally providing a way to test the algorithm.

1.1 Related work and Contribution

This thesis makes contributions in three directions: root isolation of a polynomial with coefficients in a multiple field extension, topology computation of parametric curves, and computation of the convex hull of plane and space parametric curves.

In the sequel, we discuss each of the problems addressed in this thesis and provide a thorough summary of our contribution to them. For an extended bibliography review, we refer the reader to the corresponding chapters.

1.1.1 Root isolation in a multiple field extension

In Ch. 3, we consider the following system of polynomials with integer coefficients:

$$\begin{aligned} F_1(X_1) &= 0\,, \\ &\vdots \\ F_n(X_n) &= 0\,, \\ F(X_1,\ldots,X_n,Y) &= 0\,. \end{aligned} \tag{1.1}$$

Let d be the maximum of the degrees of these polynomials and τ the maximum bit-size of coefficients. Solving the previous system can be seen as isolating the roots of $F(a_1, \ldots, a_n, Y) \in L[Y]$ where $L = \mathbb{Q}(a_1, \ldots, a_n)$ is an extension of $\mathbb{Q}$ and a_i, for $i = 1, \ldots, n$, have minimal polynomials $F_i(X_i)$, respectively. "Solving" a polynomial system usually means computing isolating boxes for its roots, i.e., disjoint boxes in $\mathbb{C}^n$ such that each one of them contains one root and their union covers all roots. There also exist other representations of the roots, as for instance the Rational Univariate Representation (RUR) [Rou99] or the representation through triangular systems (cf. Sec. 2.5 for a short review). Notably, the RUR establishes a bijection between the roots of a zero-dimensional system and the roots of a univariate polynomial. The coordinates of the roots of the system are all expressed as rational polynomial functions evaluated at the roots of the same univariate polynomial.

An efficient algorithm specifically designed for isolating the roots of the system in Eq. (1.1), under no assumptions, lacks from literature. However, partial results do exist [DDR$^+$22, KS12, ST19, MSW15, KS15, Rum77, JK97]. Notably, in the simple extension case, the state-of-the-art result is from Diatta et al. [DDR$^+$22]. They provide precise amortized separation bounds for the polynomial, i.e., bounds on the minimum distances between the roots, and bit-complexity estimates for isolating the roots. In a multiple extension, the only complete algorithmic result concerns the case where F does not have multiple roots [ST12]. Algorithms for solving zero-dimensional polynomial systems of $n+1$ equations in $n+1$ variables could also be employed to solve the system Eq. (1.1), but it would be excessive as we are not utilizing the particular structure of our problem. The state-of-the-art algorithm for zero-dimensional square systems is given by Brand and Sagraloff [BS16] and it is a Las Vegas algorithm. Employing this algorithm to isolate the roots of the system in Eq. (1.1) has expected complexity in $\widetilde{\mathcal{O}}_B\left((n+1)^{n(\omega+1)+2} d^{(\omega+2)n+1}\tau\right)$, where ω denotes the exponent in the complexity of matrix multiplication.

Contribution. We generalize the results of [DDR$^+$22] concerning the geometry of the roots of F, and notably their separation. We provide amortized bounds taking into consideration the multiplicities of the roots. These bounds can be exploited for designing efficient root isolation algorithms but are also of independent interest; for example we apply the aggregate bounds of F on the 'Sum of Square Roots of Integers' Problem. This problem can be formulated as the solution of smallest magnitude of a system in the form of Eq. (1.1). Our approach is consistent with prior results using separation bounds [BFMS00, MS00], and, even more, the proven bounds are aggregate.

For the root isolation of the system of Eq. (1.1), we design an algorithm that is based on two ingredients: the amortized separation bounds for F and the univariate root isolation algorithm of [MSW15] for polynomials with complex coefficients. The general idea is that we isolate the roots of the n univariate polynomials $F_1, \ldots, F_n$ and then, for every root

in $\mathbb{C}^n$ we approximate the coefficients of F up to a certain precision (that is determined by the separation bounds). After that, F is a univariate polynomial whose roots can be isolated using the root isolation algorithm of [MSW15].

However, certain technical concerns need to be addressed. The root isolation algorithm of [MSW15] for univariate polynomials, demands the number of distinct roots of the polynomial to be known. This is because it uses this number in a termination criterion, that the stops the computation when the correct number of roots has been found. This means that, in our case, we should compute the number of distinct roots of $F(\mathbf{x}, Y)$ for every root $\mathbf{x} \in \mathbb{C}^n$ of $F_1 = \ldots = F_n = 0$. We find that this operation dominates the total bit-complexity of the algorithm, justifying the fact that the case where all the roots are simple was only treated in literature so far [ST12].

To find the number of distinct roots we follow two approaches; a numerical (yet certified) and a formal one. The first one, uses numerical approximations of every $\mathbf{x} \in \mathbb{C}^n$ that is a root of $F_1 = \cdots = F_n = 0$, to determine the degree of the gcd of $F(\mathbf{x}, Y)$ and $\frac{\partial F}{\partial Y}(\mathbf{x}, Y)$ through subresultant computations. This leads to a deterministic algorithm with worst case complexity in $\widetilde{\mathcal{O}}_B\left(n^2 d^{3n+3}\tau + n^3 d^{2n+4}\tau\right)$ (Thm. 3.7(ii), Rem. 3.8).

The second approach is to compute a Rational Univariate Representation (RUR) of the roots $\mathbf{x}$ of $F_1 = \ldots = F_n$ (Thm. 3.35). We employ the algorithm of [Rou99] that uses the traces of the multiplication matrices in $\mathbb{Q}[X_1, \ldots, X_n]/\langle F_1, \ldots, F_n\rangle$ and can be fast due to their structure. Then, we use the RUR to transform the system of Eq. (1.1) into a bivariate system in triangular form. The number of distinct roots of the original system is then found through the bivariate system using gcd computations. We assemble everything into a Las-Vegas algorithm for the root isolation that is of expected complexity

$$\widetilde{\mathcal{O}}_B(n(n + 2^n)d^{3n-1}(d + \tau) + n^{10}),$$

when $n \geq 6$ and $\widetilde{\mathcal{O}}_B(d^{2n+5}(d + \tau))$ otherwise (Rem.3.38).

1.1.2 Topology of parametric curves

In Ch. 4, we consider a curve in $\mathbb{C}^n$, parametrized by

$$\begin{aligned} \phi: \quad & \mathbb{C} \dashrightarrow \mathbb{C}^n \\ & t \mapsto \big(\phi_1(t), \ldots, \phi_n(t)\big) = \Big(\frac{p_1(t)}{q_1(t)}, \ldots, \frac{p_n(t)}{q_n(t)}\Big), \end{aligned} \tag{1.2}$$

where p_i, q_i are polynomials in $\mathbb{Z}[t]$ with maximum degree d and bitsize of coefficients at most τ. The focus of this chapter is on computing the topology of the real trace, $\mathcal{C}$, of the curve, that is the Zariski closure of the image of ϕ intersected with $\mathbb{R}^n$. Our objective is to develop a general algorithm that can be applied to curves of any dimension. Computing the topology, as also discussed in pg. 12, essentially suggests the computation of an abstract

graph that, when embedded in $\mathbb{R}^n$, is isotopic to the curve. For the graph's vertices, one has to compute the singular points on the curve, and other important points, such as the extreme ones (with respect to the coordinate directions).

The computation of singular points in parametric curves is a fundamental element of any algorithm that aims to compute the topology. For the parametrization ϕ (Eq. (1.2)), the following system of bivariate polynomials

$$h_i(s,t) = \frac{p_i(s)q_i(t) - q_i(s)p_i(t)}{s-t}, \quad \text{for } i \in [n]\,, \tag{1.3}$$

is often considered to compute the singular points, since, for instance, the self-intersections of the curve correspond to points in $\mathbb{R}^n$ given by two parameter values s and t, with $\phi(s) = \phi(t)$. Numerous research works find the singular points using univariate resultant computations involving these polynomials [AB89, vdEY97, Par02, PD07, GRS02, BPD19, RSV09, ADT10]. Other methods also exist, for example by using the syzygies of the ideal generated by the polynomials that give the parametrization [CKPU11], or the μ-basis of the parametrization ([JSC18] and references therein).

On the topology computation, a general algorithm for any dimension lacks from literature. Alcázar and Díaz-Toca [ADT10] study the topology of real plane and space parametric curves without implicitizing. For the computation of the singularities they use a resultant based approach of the system in Eq. (1.3). The case of space curves is reduced to the plane one through a birational [SWPD08, Def. 2.37] projection of the curve on the plane. For the isotopic graph construction they use a sweep-line approach, which is common for implicit curves.

Contribution. We design an algorithm, `PTOPO` (Alg. 6), that applies directly to rational parametric curves of any dimension and is exact and complete, in the sense that there are no assumptions on the input, as for instance in [ADT10] where they require the absence of axis-parallel asymptotes. However, a preprocessing step ensures some good properties of the parametrization, that is properness [PD06] (Lem. 4.6) and abscence of singularities at infinity (Lem. 4.7).

To compute the singular points, we isolate the roots of the polynomial system of Eq. (1.3); it is an *over-determined bivariate system.* We introduce an algorithm for isolating the roots of over-determined bivariate polynomial systems using the Rational Univariate Representation (RUR) [Rou99, BLPR15, BLM$^+$16, BLPR13]. It achieves a worst case and expected bit complexity that matches the bounds for bivariate systems (Thm. 4.16). By leveraging the symmetry of the system and using nearly optimal algorithms for computations with real algebraic numbers [DDR$^+$22, BLM$^+$16, DET09, PT17], we then are able to find the parameters corresponding to the interesting points on the curve that will comprise the graph's vertices.

For the connection of the graph's vertices, the novelty of PTOPO is that it computes only in the parameter space and it does not use a sweep-line algorithm to construct the isotopic graph (in contrast to [ADT10]). In this way, we circumvent the need to carry out operations such as univariate root isolation in a field extension or polynomial evaluation at an algebraic number, that would result in an increase of complexity.

PTOPO computes the abstract graph in

$$\widetilde{\mathcal{O}}_B(nd^6 + nd^5\tau + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau)$$

bit-operations in the worst case (Thm. 4.24). We also provide a Las Vegas variant with expected complexity in

$$\widetilde{\mathcal{O}}_B(d^6 + d^5(n + \tau) + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau).$$

If $n = \mathcal{O}(1)$, the aforementioned bounds become $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$. For plane and space curves, our bound improves the previously known bound due to [ADT10] by a factor of N^{10}. In particular, for $n = 2$, our bound matches the record bound, $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$ or $\widetilde{\mathcal{O}}_B(N^6)$, for computing the topology of implicit plane curves [DDR+22, KS15].

Additionally, for plane and space curves, we can compute an isotopic embedding of the graph in box that contains all the the topological features of the curve (Lem. 4.20) in the same bit-complexities (Thm. 4.23, Cor. 4.26, Thm. 4.28).

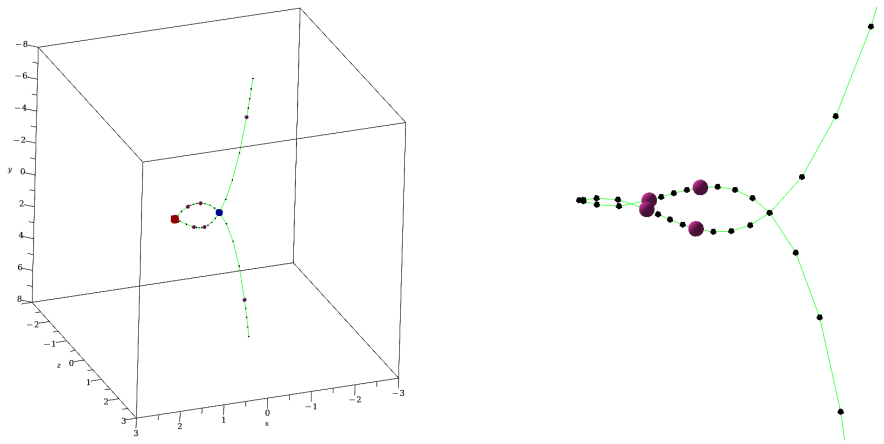
We provide a certified implementation[1] of PTOPO in MAPLE that computes the topology of plane and space curves and visualizes them. Our package is built upon the real root isolation routines of MAPLE's RootFinding library and the SLV package [DET09]. A typical output of PTOPO for a space parametric curve is shown in Fig. 1.3. Notice that in Fig. 1.3(a) the box in which the curve is plotted is large enough to contain all the topological features, including the branches that go to infinity.

1.1.3 Convex hull of plane and space parametric curves

In Ch. 5, we consider plane and space curves parametrized by ϕ as in Eq. (1.2), when $n = 2$ and 3 respectively. The polynomials of the parametrization are of degree at most d and bitsize τ. Our objective is to compute an exact boundary description of the convex hull of $\phi(I)$, where $I \subseteq \mathbb{R}$ such that $\phi(I)$ is compact.

A complete algorithmic framework for the exact boundary description is missing from literature. There is a series of works for the problem of computing the convex hull of curved arcs, given in parametric form, on the plane [SJVW87, DS90, BK91, JL05] that are mostly combinatorial and rely on expensive oracles. Ranestad and Sturmfels [RS09] compute the algebraic boundary of the convex hull of a space parametric curve; however,

[1] https://gitlab.inria.fr/ckatsama/ptopo



(a) (b)

Figure 1.3: (a) The visualisation of PTOPO for the curve parametrized by $\phi(t) = \left(-\frac{3t^4-1}{(t^2+1)^2}, -\frac{t^3(23t^4-8)}{(t^2+1)^2}, -\frac{t^3(29t^4-10)}{(t^2+1)^8}\right)$. The blue square point is a multiple point (for $t = \pm 0.767963$), the red square is a cusp ($t = 0$) and the purple spheres are extreme points. (b) A close-up of the curve.

this description does not give information on the exact boundary structure. The convex hull of parametric curves in $\mathbb{R}^n$ can be expressed as the projection of a spectrahedron, the feasible region of a semidefinite program [Hen11]. While this representation of the convex hull is suitable for optimization, it fails to provide any insights into the facial structure of the boundary.

Previous research, such as that of [Joh04, EKH01] for plane curves and [SEJK04] for 3D curves, has also presented a problem formulation approach that involves solving polynomial systems. Specifically, the roots of the considered polynomial systems include the segments (for plane curves) and triangles (for space curves) that lie on the boundary of the convex hull. For space curves, the boundary's surface patches can be traced by an implicit plane curve; generically, a point on this curve corresponds to a segment connecting two points on the space curve. However, some aspects of this approach are not easily implementable, and there is no analysis of the computational complexity. Therefore, we seek to bridge the gap between a theoretical method and one that can be effectively implemented in a working system.

Contribution. We design an exact and complete algorithm for the convex hull computation of plane and space parametric curves, defined over a parameter interval $I \subset \mathbb{R}$, such that $\phi(I)$ is compact.

The boundary description that we compute, consists, in the case of plane curves, of a sequence of parametric arcs, described by the corresponding parameter intervals, and of

line segments that connect two points on the curve. All the parameters are represented by algebraic numbers and thus a numerical approximation of arbitrary precision can be obtained.

For space curves, we describe the faces of the boundary of the convex hull using a doubly connected edge list (DCEL) [dBCvKO08, Ch.2.2], that is often used to encode the boundary information of convex polyhedra. The vertices of the faces that are triangles are described by the corresponding parameters that are algebraic numbers. A surface patch is described by an arc of an implicit plane curve that traces the ruled patch.

A basic ingredient of our algorithms, is the Support predicate (Sec. 5.3) that can be applied to parametric curves of any dimension and checks if a hyperplane is supporting for the curve or not. The test that the predicate performs can be formulated as a test of sign-invariance of a univariate polynomial. It should be noted that creating such a predicate for implicit curves is not an easy task.

The algorithm for plane curves, first computes the line segments. The line segments are solutions of some polynomial systems. However, these systems contain some extra roots that do not correspond to segments on the boundary of the convex hull. We split the curve at the endpoints corresponding to all the solutions of the systems (by splitting the parameter interval); so the curve is segmented to possibly more arcs than needed because of the extra roots of the polynomial system. Computing these extra segments is unavoidable, since they are all roots of the same polynomial system. However, this does not affect the correctness of the algorithm. Every arc of the decomposition has the property that it is either entirely on the boundary of the convex hull, or it is contained in its interior. Then, for every branch we can use the Support predicate to check if it is on the boundary or not. At the last step, connecting the arcs can be done using some combinatorial criteria since the sets of segments are already computed.

For the convex hull of space parametric curves our algorithm follows the same line as the 2D algorithm. The triangle facets are obtained through solving some polynomial systems. Then, the surface patches of the curve, are traced through an implicit plane curve, the so-called bisecant curve (Eq. (5.8)). We are able to find the surface patches on the boundary by a suitable decomposition of this implicit curve (using a specially tailored Cylindrical Algebraic Decomposition) and applications of the Support predicate. Connecting the surface patches with triangles can be realised, since the triangles are already computed, by using again some combinatorial criteria.

We analyse the bit-complexity of the algorithms. In the quest of minimizing the bit-complexity, we manage to express it by means of the complexity of isolating the roots of systems of the form of Eq. (1.1), instead of square systems of the same dimension (Thm. 5.6 and Thm. 5.15). Using the results of Ch. 3 for the bit-complexity of solving systems of this form, we show that the bit-complexity of the algorithm for plane curves is in $\widetilde{\mathcal{O}}_B(d^{10} + d^9\tau)$ and in $\widetilde{\mathcal{O}}_B(d^{13} + d^{12}\tau)$ for space curves. If alternatively, we employ the

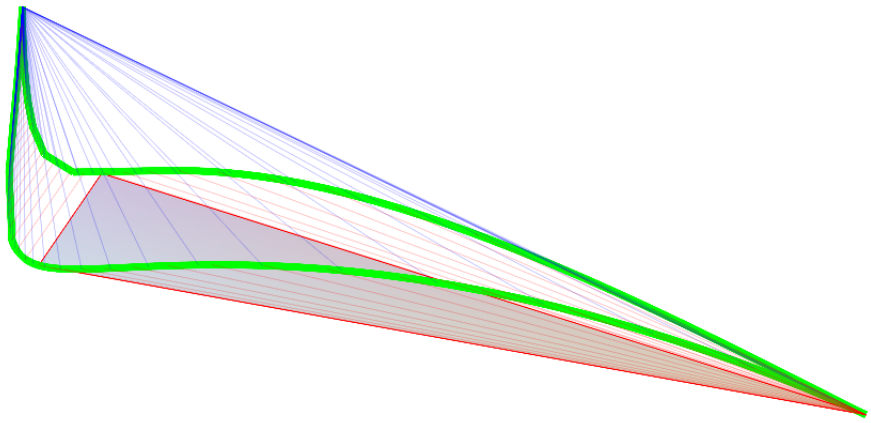


Figure 1.4: The boundary of the convex hull of the curve parametrized by Eq. (1.4) consists of one triangle and four ruled (developable) surface patches.

Las-Vegas algorithm of [BS16] for solving square zero-dimensional polynomial systems, the bit-complexities would be

$$\widetilde{\mathcal{O}}_B\left(d^{2\omega+5}(d+\tau)\right) \approx \widetilde{\mathcal{O}}_B\left(d^{9.75}(d+\tau)\right)$$

for plane curves, and

$$\widetilde{\mathcal{O}}_B\left(d^{3\omega+7}(d+\tau)\right) \approx \widetilde{\mathcal{O}}_B\left(d^{14.12}(d+\tau)\right)$$

for 3D curves, where $\omega \approx 2.372873$ denotes the exponent in the complexity of matrix multiplication. This highlights the advantage of utilizing the geometry of a problem to create a formulation that involves system of special structure. A preliminary version of our algorithm is implemented in Maple; as an example, the convex hull of the curve parametrized by

$$\phi(t) = \left(-\frac{8\left(42t^2+13t-42\right)^3}{614125\left(t^2+1\right)^3}, \frac{(t-13)^3\left(13t+1\right)^3}{614125\left(t^2+1\right)^3}, \frac{(t-13)^3\left(8783t^3+2223t^2-2379t+10989\right)}{2456500\left(t^2+1\right)^3}\right) \quad (1.4)$$

is shown in Fig. (1.4).

1.2 Organization of the thesis

Chapter 2 of this thesis provides an overview of the algebraic geometry concepts that form the foundation of our research. Although this chapter does not present any novel contributions, it is essential for understanding the subsequent chapters.

The remaining chapters of the thesis focus on our original contributions, which are divided into two parts: *Polynomial system solving* and *Algorithms on parametric curves*.

Polynomial system solving

- **Chapter 3** contains our contribution to solving polynomial equations with coefficients in a multiple field extension. Part of this chapter will appear in
  - *Isolating roots in a multiple field extension.*
    C.K., Fabrice Rouillier.
    ISSAC '23: Proceedings of the 48th International Symposium on Symbolic and Algebraic Computation.

Algorithms on parametric curves

- **Chapter 4** contains our contribution to the computation of the topology of parametric curves in $\mathbb{R}^n$. The contributions of this chapter are a joint work with Fabrice Rouillier, Elias Tsigaridas and Zafeirakis Zafeirakopoulos and they have appeared in
  - *On the geometry and the topology of parametric curves.*
    C.K., Fabrice Rouillier, Elias Tsigaridas, Zafeirakis Zafeirakopoulos.
    ISSAC '20: Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation (10.1145/3373207.3404062) [KRTZ20a].
  - *PTOPO: Computing the geometry and the topology of parametric curves.*
    C.K., Fabrice Rouillier, Elias Tsigaridas, Zafeirakis Zafeirakopoulos.
    Journal of Symbolic Computation (10.1016/j.jsc.2022.08.012) [KRTZ23].
- **Chapter 5** contains our contribution to the computation of the convex hull of parametric curves in $\mathbb{R}^2$ and in $\mathbb{R}^3$.

Chapter 2

Preliminaries

The purpose of this chapter is to introduce some basic notions of computational algebra and the computational tools employed regularly throughout this thesis. Most of the proofs are omitted, since they can be found in classical textbooks of computer algebra, as for example [BPR06, vzGG13, Yap99]. We chose to include some proofs that are instrumental for the understanding of the reader or for whom we were not able to find a reference that is appropriate for our setting.

The chapter is organised as follows. In Sec. 2.1 we present our notation. Then, we review some algorithms for basic operations involving polynomials (Sec. 2.2), gcd, resultant and subresultant computations (Sec. 2.3), root isolation of univariate polynomials (Sec. 2.4) and general polynomial systems (Sec. 2.5). Lastly, Sec. 2.6 provides a general description of modular algorithms. For the most part, the polynomials will be considered with integer coefficients whose size will be taken into account to offer realistic estimates for the complexity of the different algorithms; the so-called *bit-complexity*.

2.1 Notation

We introduce some notation and terminology used throughout this thesis.

Vectors. Throughout the thesis, vectors are denoted by boldface symbols. Let $n \in \mathbb{N}$. For a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{C}^n$, we denote by $\mathbf{x}_{-i}$ the vector $(x_1, \ldots, x_{i-1}, x_{i+1}, x_n) \in \mathbb{C}^{n-1}$, $i \in [n]$. We use the abbreviation $[n]$ for the set $\{1, \ldots, n\}$, for a positive integer n.

We call absolute L-bit approximation of a real number x, a rational number $\tilde{x}$ such that $|x - \tilde{x}| < 2^{-L}$.

Polynomials and Ideals. For a polynomial $f = \sum_{i=1}^{d} a_i X^i \in \mathbb{C}[X]$ we denote by $\mathrm{lc}(f)$ its leading coefficient and by $\mathrm{tc}(f)$ its tail coefficient (the coefficient of its non-zero term of the lowest degree). The k-th derivative of f is denoted by $f^{(k)}$. When f has integer coefficients, we denote its bitsize by $\mathcal{L}(f)$ and we say that it is of *size* (d, τ). when its

degree is at most d and has bitsize τ. Similarly, a multivariate polynomial with integer coefficients is of *size* (d, τ) when its total degree is at most d and has bitsize τ.

Let $I = \langle f_1, \ldots, f_k \rangle$ be a polynomial ideal in $\mathbb{C}[X_1, \ldots, X_n]$, $k \in \mathbb{N}$. We denote by $V_{\mathbb{C}}(I)$ or $V_{\mathbb{C}}(f_1, \ldots, f_k)$ the complex variety defined by I. For a $\mathbf{x} \in V_{\mathbb{C}}(I)$, we denote by $\mu_I(\mathbf{x})$ its multiplicity as root of I. When I is generated by one polynomial $f \in \mathbb{C}[X]$, we write for simplicity $\mu_f(x)$ to denote the multiplicity of x as root of $I = \langle f \rangle$. The ideal $\langle f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_k \rangle$, for $i \in \{1, \ldots, k\}$ is denoted by $I \setminus f_i$.

2.2 Computations with polynomials

We consider polynomials with integer coefficients and we recall some elementary results on computing with these polynomials. In particular, we review addition, multiplication and division of polynomials and evaluation at rational numbers. We present bit-complexity results for these operations, using state-of the-art algorithms.

2.2.1 Basic operations

The bit-complexity of adding two integers of bitsize τ is $\mathcal{O}_B(\tau)$. The bit-complexity of multiplying two integers of bitsize τ depends strongly on the algorithm used: $\mathcal{O}_B(\tau^2)$ when the multiplication is done naively, $\mathcal{O}_B(\tau^{\log(3)})$ when Karatsuba's method is used, $\mathcal{O}_B(\tau \log \tau \log(\log(\tau)))$ using Fast Fourier Transform (FFT) and $\mathcal{O}(\tau \log \tau)$ using the algorithm of Harvey and van Der Hoeven [HvdH21]. We refer to [vzGG13, §8] for relative information. For the results presented in this subsection we refer to [BPR06, §8] and [vzGG13] for a detailed exposition.

Given two multivariate polynomials with integer coefficients, their addition amounts to adding the coefficients of the corresponding monomials. It is described in [BPR06, Alg. 8.9]. For the addition of two n-variate polynomials of total degree $\leq d$, we have to take into account that each one has $\binom{d+n}{n}$ monomials [BPR06, Lem. 8.10], and so we have to perform $\binom{d+n}{n} \leq (d+1)^n$ additions.

Theorem 2.1 (Addition). *Let $f, g \in \mathbb{Z}[X_1, \ldots, X_n]$ be n-variate polynomials of size (d, τ). Their sum $f + g$ has bitsize at most $\tau + 1$ and it can be computed in $\mathcal{O}_B(d^n \tau)$ bit-operations.*

For the multiplication of two univariate polynomials we use Schönhage and Strassen's multiplication algorithm [vzGG13, §8.3]. For a more recent result, we refer again to [HvdH21].

Theorem 2.2 (Univariate Multiplication). [vzGG13, Cor.8. 27] *Let $f, g \in \mathbb{Z}[X]$ univariate polynomials of size (d, τ). Their product $f \cdot g$ can be computed in $\widetilde{\mathcal{O}}(d)$ arithmetic operations and in $\widetilde{\mathcal{O}}_B(d\tau)$ bit-operations.*

For the multiplication of two polynomials in $\mathbb{Z}[X_1, \ldots, X_n]$, one can use the method of *binary segmentation*. Binary segmentation is used to map a multivariate polynomial $f \in \mathbb{Z}[X_1, \ldots, X_n]$ directly into an integer, by substituting the variables $X_1, \ldots, X_n$ by integers $2^{\nu_1}, \ldots, 2^{\nu_n}$, where $\nu_1, \ldots, \nu_n$ are as small as possible so that $f(X_1, \ldots, X_n)$ can then be uniquely recovered from its integer value $f(2^{\nu_1}, \ldots, 2^{\nu_n})$.

Klose's multivariate binary segmentation [Klo95] is based on the following power-2 substitution homomorphism:

Lemma 2.3. [Klo95, Lem.2.1] *Let $n, \tau, d \in \mathbb{N}$, with $n \geq 2$. Let also $\nu = (\nu_1, \ldots, \nu_n)$ with $\nu_i := (\tau + 1)(d^{i-1} + d^{i-2} + \cdots + 1)$, for $i \in [n]$. Then, the substitution homomorphism*

$$\begin{aligned} \phi_\nu : \quad & \mathbb{Z}[X_1, \ldots, X_n] \to \mathbb{Z} \\ & f \mapsto f(2^{\nu_1}, \ldots, 2^{\nu_n}) \end{aligned}$$

is injective on the set $P_{d,\tau} = \{f \in \mathbb{Z}[X_1, \ldots, X_n] : \mathcal{L}(f) \leq \tau, \deg(f) \leq d\}$. For $f \in P_{d,\tau}$ we have that $\mathcal{L}(\phi_\nu(f)) \leq \nu_n\, d + \tau + 1$.

Using the mapping ϕ_ν, one can compute the image of f under it and then perform operations in $\mathbb{Z}$, instead of $\mathbb{Z}[X_1, \ldots, X_n]$. The result, can be recovered by computing the inverse image with respect to ϕ_ν. This is described in [Klo95, Alg. Inverse Binseg] and uses integer division with remainder. So, for the multiplication of two multivariate polynomials using multivariate binary segmentation we have the following:

Theorem 2.4 (Multivariate multiplication). [Klo95, Thm. 3.1] *Let $f, g \in \mathbb{Z}[X_1, \ldots, X_n]$ be n-variate polynomials of size (d, τ). Their product $f \cdot g$ is a polynomial of size $(\mathcal{O}(d), \mathcal{O}(\tau + n \log d))$ and it can be computed using binary segmentation in $\widetilde{\mathcal{O}}_B((2d)^n(n + \tau))$ bit-operations.*

An alternative to binary segmentation is *Kronecker substitution* which reduces the multiplication to multivariate polynomials to the multiplication of univariate ones. It leads to comparable bit-complexity estimates. We refer the interested reader to [vzGG13, §8.4], [vzGG13, Ex. 8.38] and [Pan94]. For a probabilistic approach to Kronecker substitution we refer to [Arn16].

Division with remainder is possible for any two elements in a Euclidean domain. This means that, since $\mathbb{Z}[X]$ is not Euclidean domain, we cannot always divide with remainder in $\mathbb{Z}[X]$. For instance, division of x^2+1 by $2x$ gives a trivially zero quotient and remainder equal to the dividend $x^2 + 1$. For polynomials in $K[X]$, where K is a field, division with remainder by an arbitrary nonzero polynomial is possible [BPR06, Alg. 8.5].

However, for two polynomials $f, g \in \mathbb{Z}[X]$, division with remainder in $\mathbb{Z}[X]$ can still be done as long as $\mathrm{lc}(g) = -1$ or 1, that is, g is invertible (see [vzGG13, Alg. 2.5]). In general, to ensure that division of $f(X)$ by $g(X)$ can be performed in $\mathbb{Z}[X]$, or any integral

domain, we can perform the so-called *pseudo-division*. It amounts to multiplying $f(X)$ by $\mathrm{lc}(g)^{\deg(f)-\deg(g)+1}$ (see [vzGG13, Alg. 6.61]).

Theorem 2.5 (Univariate pseudo-division). [vzGG13, Thm. 9.6] *Let $f, g \in \mathbb{Z}[X]$ be of sizes (d, τ). The pseudo-division of f by g can be performed in $\widetilde{\mathcal{O}}_B(d^2\tau)$ bit-operations.*

Moreover, when the remainder of the division of f by g is known to be zero, we have the following:

Lemma 2.6 (Exact division). [vzGG13, Ex. 9.14] *Let $f, g \in \mathbb{Z}[X]$ such that f is of size (d, τ) and there exists $q \in \mathbb{Z}[X]$ such that $f = qg$. Then q can be computed in $\widetilde{\mathcal{O}}_B(d(d+\tau))$ bit-operations.*

Assume now that we have two multivariate polynomials $f, g \in \mathbb{Z}[X_1, \ldots, X_n]$, and we want to perform pseudodivision of f by g with respect to the variable X_n. Then, we can use binary segmentation to reduce computations to the univariate pseudo-division.

Theorem 2.7 (Multivariate pseudo-division). [Klo95, Thm. 3.2] *Let $f, g \in \mathbb{Z}[X_1, \ldots, X_n]$ of sizes (d, τ) and $n \geq 2$. The pseudo-division of f by g with respect to X_n can be performed in*

$$\begin{cases} \widetilde{\mathcal{O}}_B(d^{2n+1}(n+\tau))\,, & \text{if } d \geq 2 \text{ and } \deg_{X_n}(g) \geq 1\,, \\ \widetilde{\mathcal{O}}_B(d^{2n}(n+\tau))\,, & \text{if } d \geq 2 \text{ and } \deg_{X_n}(g) = 0\,, \\ \widetilde{\mathcal{O}}_B(d^2\tau)\,, & \text{if } d = 1 \end{cases}$$

bit-operations.

2.2.2 Evaluation of polynomials

Let

$$f(X) = a_d X^d + \ldots + a_1 X + a_0 \in \mathbb{Z}[X]$$

of size (d, τ). To evaluate f at a rational number a, one can use the Horner's rule to write the polynomial as

$$f(X) = a_0 + X(a_1 + X(a_2 + X(a_3 + \ldots + (a_{n-1} + a_n X) \cdots)\,. \tag{2.1}$$

Then, we start by evaluating the inner-most parenthesis and then we continue similarly by evaluating always the inner-most parenthesis. This algorithm is also described in [BPR06, Alg. 8.14].

Having expressed the polynomial in the form of Eq. (2.1), its evaluation only requires d additions and d multiplications. On the contrary, if the classic monomial form of f is used for its evaluation, then we have to perform d additions and $\mathcal{O}(d^2)$ multiplications. For the bit-complexity of the Horner's method, we have the following:

Lemma 2.8 (Evaluation using Horner's method). *Let $a \in \mathbb{Q}$ of bitsize σ and a univariate polynomial $f \in \mathbb{Z}[X]$ of size (d, τ). The evaluation of f at a, $f(a)$, can be computed using the Horner's method in $\widetilde{\mathcal{O}}_B(d(\tau + d\sigma))$ bit-operations and it has bitsize in $\mathcal{O}(\tau + d\sigma)$.*

Proof. The polynomials of degree one that are evaluated at each step, have increasing bitsizes that are at most τ, $\mathcal{O}(\tau + \sigma)$, $\mathcal{O}(\tau + 2\sigma)$, $\ldots$, $\mathcal{O}(\tau + (d-1)\sigma)$. To evaluate each polynomial, we perform one addition and one multiplication, thus we perform $\widetilde{\mathcal{O}}_B(\tau + \sigma)$, $\ldots$, $\widetilde{\mathcal{O}}_B(\tau + d\sigma)$ bit-operations respectively. Adding all the previous bounds, results to a total number of bit-operations in $\widetilde{\mathcal{O}}_B(d(\tau + d\sigma))$. □

However, one can do better by using a divide and conquer approach, known as Estrin's scheme. Assuming, for simplicity, that d is a power of two, we can write f as

$$f(X) = \sum_{i=0}^{d/2} a_i X^i + X^{d/2} \sum_{i=1}^{d/2} a_{i+d/2} X^i \,. \tag{2.2}$$

Thus, Eq. (2.2) shows that computing $f(a)$ amounts to evaluating $X^{d/2}$ and two $(d/2)$-degree polynomials of size (d, τ) and then to performing one multiplication and one addition. By repeating for the $(d/2)$-degree polynomials, we obtain a better bound on the bit-complexity of the evaluation.

Lemma 2.9 (Evaluation using Estrin's scheme). [BLPR15, Lem.6] *Let $a \in \mathbb{Q}$ of bitsize σ and a univariate polynomial $f \in \mathbb{Z}[X]$ of size (d, τ). The evaluation of f at a, $f(a)$, can be computed using Estrin's scheme in $\widetilde{\mathcal{O}}_B(d(\tau + \sigma))$ bit-operations and it has bitsize in $\mathcal{O}(\tau + d\sigma)$.*

Now, evaluating a multivariate polynomial at one point can be induced recursively to the evaluation of univariate polynomials.

Lemma 2.10 (Evaluation of multivariate polynomials). *Let $\mathbf{a} \in \mathbb{Q}^n$ of bitsize σ and a polynomial $f \in \mathbb{Z}[\mathbf{X}]$ of size (d, τ). The evaluation of f at $\mathbf{a}$ can be computed in $\widetilde{\mathcal{O}}_B(nd^n(\tau + \sigma))$ bit-operations and $f(\mathbf{a})$ has bitsize in $\widetilde{\mathcal{O}}(nd\sigma + \tau)$.*

Proof. We write $f(\mathbf{X}) = f_d(\mathbf{X}_{-n})X_n^d + \cdots + f_0(\mathbf{X}_{-n})$. We denote by $E(n)$ the cost of the evaluation of f and by $E(n-1)$ the cost of the evaluation of the $(n-1)$-variate polynomials $f_0, \ldots, f_d$. Let $L(n-1)$ be the bitsize of $f_i(\mathbf{a}_{-n})$. Supposing that we evaluate $f_0, \ldots, f_d$ and then, using Lem. 2.9, we evaluate the resulting univariate polynomial, we have the relation

$$E(n) = (d+1) \cdot E(n-1) + d(L(n-1) + \sigma).$$

Then, we induce the evaluation of each f_i, for $i = 0, \ldots, d$, to the evaluation of $d+1$ $(n-2)$-variate polynomials and a univariate one, and we continue similarly. Again, for the

k-variate polynomials that appear in this recursive process, we denote by $E(k)$ the cost their evaluation and by $L(k)$ the bitsize of the result after the evaluation. Thus, we have that in the end:

$$E(n) = (d+1)^{n-1}E(1) + \sum_{j=1}^{n-1} d^j L(n-j) + \sigma \sum_{j=1}^{n-1} d^j \ \in \widetilde{\mathcal{O}}_B(nd^n(\tau+\sigma)),$$

since $E(1) = \widetilde{\mathcal{O}}_B(d(\tau+\sigma))$, $L(1) = \tau + d\sigma$ and $L(k) = L(k-1) + d\sigma$, for $k = 2, \ldots, n-1$ (Lem. 2.9). The bitsize of $f(\mathbf{a})$, i.e., $L(n)$, is in $\widetilde{\mathcal{O}}(nd\sigma + \tau)$. □

Now, say that we want to evaluate a univariate polynomial $f \in \mathbb{Z}[X]$ of degree d at some numbers $a_1, \ldots, a_d \in \mathbb{Q}$, the straight-forward way would be to use the algorithms of Lem. 2.8 or Lem. 2.9 d times. However, there is a faster way to do this, using *fast multipoint evaluation* ([vzGG13, §10.1], [MB72]). Using recursion, this method reduces the evaluation of a univariate polynomial at d points to successive polynomial multiplications and divisions, while the recursion trees are balanced with respect to degree.

To describe the algorithm, we now assume for simplicity that $d = 2^k$. We define the polynomials

$$P_{i,j} := \prod_{l=1}^{2^i} (X - a_{2^i j + l - 1}) \ \text{ for } 0 \le i \le k \text{ and } 1 \le j \le 2^{k-i}\,.$$

Each $P_{i,j}$ is a subproduct with 2^i factors. We compute them in a recursive way, by building a *subproduct tree* in bottom-up fashion, using the relations:

$$P_{0,j} = (x - a_j) \text{ and } P_{i+1,j} = P_{i,2j-1} \cdot P_{i,2j}\,.$$

The subproduct tree is illustrated in Fig. 2.1.

Then, we build the *remainder tree*, in a top-bottom fashion. We define the polynomials

$$r_{i,j} := f(X) \mod P_{i,j} \ \text{ for } 0 \le i \le k \text{ and } 1 \le j \le 2^{k-i}\,.$$

Then, we set $r_{k,1} = f(X)$ and we compute them using the relation

$$r_{i,j} = r_{i+1,\lceil j/2 \rceil}(X) \mod P_{i,j}\,.$$

The evaluation of f at a_j is exactly the remainder

$$r_{0,j} = f(X) \mod P_{0,j} = f(X) \mod (X - a_j) = f(a_j)\,.$$

The remainder tree is illustrated in Fig. 2.2. For the bit-complexity of the algorithm we have the following result:

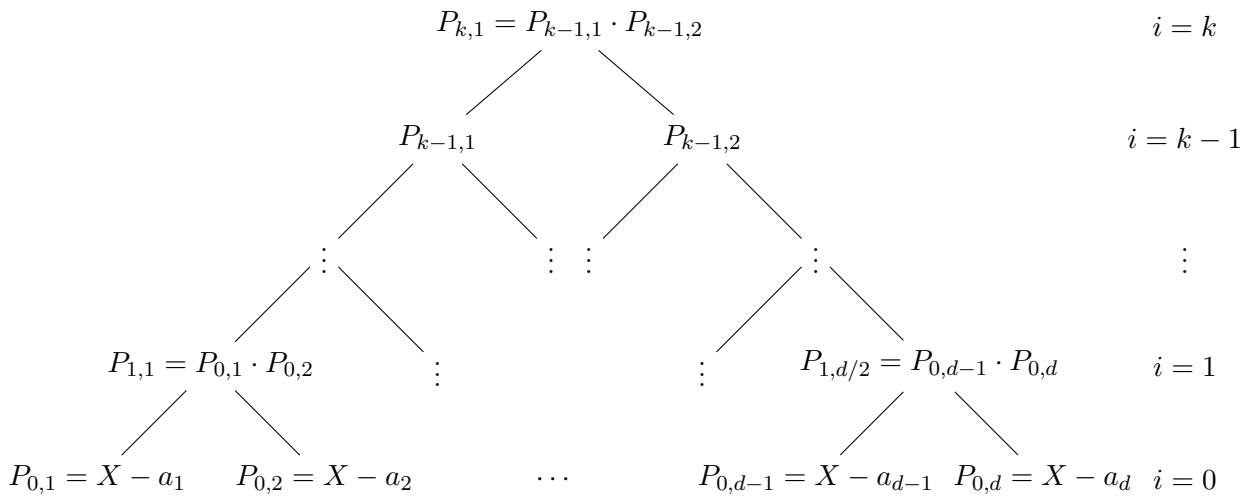


Figure 2.1: The subproduct tree of fast multipoint evaluation of f at $a_1, \ldots, a_d$.

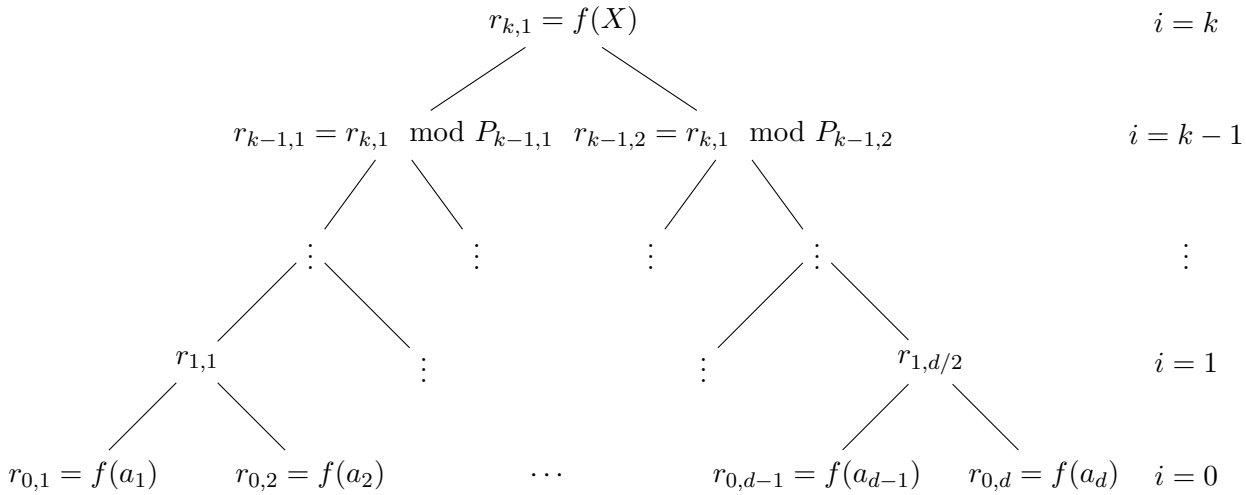


Figure 2.2: The remainder tree of fast multipoint evaluation of f at $a_1, \ldots, a_d$.

Theorem 2.11 (Multipoint evaluation in $\mathbb{Z}$). [KS13, Thm.9] *Let $f \in \mathbb{Z}[X]$ be a polynomial of degree d and of bitsize τ and let $a_1, \ldots, a_d \in \mathbb{Q}$ of bitsize σ. Then, $f(a_1), \ldots, f(a_d)$ can be computed using fast multipoint evaluation in*

$$\widetilde{\mathcal{O}}_B\left(d(\tau + d\sigma)\right)$$

bit-operations.

In particular, when the polynomial's coefficients are complex numbers, we can have another formulation of the previous theorem, that specifies that precision in which the coefficients of the polynomial need to be approximated to obtain the evaluations at a desired precision. Note that the existence of an oracle providing these approximations is assumed.

Theorem 2.12 (Multipoint evaluation in $\mathbb{C}$). [KS13, Thm.9] *Let $f \in \mathbb{C}[X]$ be a polynomial of degree d with bitsize at most τ and let $a_1, \ldots, a_d \in \mathbb{C}$ be complex points with absolute values bounded by 2^Γ, where $\Gamma \geq 1$. Then, approximate multipoint evaluation up to a precision of 2^{-L} for some integer $L \geq 0$, that is, computing $\tilde{f}_i$ such that $|\tilde{f}_i - f(a_i)| \leq 2^{-L}$ for all i, can be done in*

$$\widetilde{\mathcal{O}}_B\left(d(L + \tau + d\Gamma)\right)$$

bit-operations. The precision demand on f and the points a_i is bounded by $L + \widetilde{\mathcal{O}}(\tau + d\Gamma)$ bits.

Remark 2.13 (Multivariate multipoint evaluation). *The multipoint evaluation of a univariate polynomial is not an immediate extension of the univariate case, unless if the evaluation points belong in a set of a certain structure (for example tensor-grids [vdHS12]). Other partial results concern only the bivariate case (when $n = 2$) [NRS20, vdHL21] or evaluation over $\mathbb{Z}/r\mathbb{Z}$ [KU11, KU08]. For a general result, we refer the reader to [vdHL23] for recent advances on this matter.*

2.3 Greatest Common Divisor, Resultant and Subresultants

This section reviews the relevant concepts of the greatest common divisor of two univariate polynomials, their resultant, and subresultant sequences. For the most of this section we consider polynomials with integer coefficients. We provide definitions and properties of these concepts, along with bit-complexity results.

2.3.1 Greatest Common Divisor

For simplicity, we first consider two polynomials in $\mathbb{Q}[X]$. However, $\mathbb{Q}$ can be replaced by any field. A greatest common divisor (gcd) of f and g in $\mathbb{Q}[X]$, denoted $\gcd(f, g)$, is a

polynomial $h \in \mathbb{Q}[X]$ such that h is a divisor of both f and g, and any divisor of both f and g is a divisor of h. A gcd always exists in a Euclidean domain and is unique up to the multiplication with an invertible element of $\mathbb{Q}$ (see [vzGG13, §3.4]).

A gcd does not always exit in an arbitrary ring. But when the polynomials have coefficients in a Unique Factorization Domain (UFD), for example $\mathbb{Z}$, still their gcd can be computed.

Take f, g polynomials now belonging in $\mathbb{Z}[X]$. Then f and g are expressed uniquely as

$$f = \mathrm{cont}(f) \cdot \mathrm{pp}(f)\,,$$
$$g = \mathrm{cont}(g) \cdot \mathrm{pp}(g)\,,$$

where $\mathrm{cont}(\cdot) \in \mathbb{Z}$ is the content and $\mathrm{pp}(\cdot) \in \mathbb{Z}[X]$ is the primitive part. Then, using Gauss' Lemma, we have that if $\tilde{h}$ is a gcd of $\mathrm{pp}(f)$ and $\mathrm{pp}(g)$ in $\mathbb{Q}[X]$, then

$$\gcd(\mathrm{cont}(f), \mathrm{cont}(g)) \cdot \tilde{h} \in \mathbb{Z}[X]$$

is a gcd of f and g in $\mathbb{Z}[X]$.

Computing the gcd of f, g in $\mathbb{Q}[X]$ (or an arbitrary Euclidean domain) can be done using the *Euclidean algorithm*, that is as follows:

Let $r_0 = f$ and $r_1 = g$. For $i > 1$ the algorithm computes the remainders

$$r_{i+1} = r_{i-1} \mod r_i\,,$$

and returns the first non-zero r_k.

The sequence of remainders $\{r_0, \ldots, r_k\}$ in the Euclidean Algorithm is called *Euclidean remainder sequence* of f and g. The last element r_k of the sequence is a gcd of f and g in $\mathbb{Q}[X]$ [BPR06, Prop.1.16].

The Euclidean algorithm performs $\mathcal{O}(d^2)$ arithmetic operations in $\mathbb{Q}$ [vzGG13, Thm. 3.11]. A faster algorithm is obtained with the *half gcd* approach [vzGG13, §11]: Essentially the idea is to compute only the quotients of the divisions in the Euclidean algorithm and not the entire remainder sequence. Computing the quotients and the remainder that gives the gcd can be done in $\mathcal{O}(d)$ operations in $\mathbb{Q}$ [vzGG13, Cor. 11.9]. However, computing the entire remainder sequence is not possible at the same cost: *any algorithm computing it requires at least $\mathcal{O}(d^2)$ operations, since the output size of the sequence is quadratic in the size of the input that is in $\mathcal{O}(d)$.*

Theorem 2.14. [BPR06, Cor. 11.14] *Let $f, g \in \mathbb{Z}[X]$ of sizes (d, τ). A gcd of f and g can be computed in $\widetilde{\mathcal{O}}_B(d^2\tau)$ bit-operations.*

The size of the gcd of two polynomials with integer coefficients can be deduced from

the following bound:

Theorem 2.15 (Mignotte's bound). [BPR06, Cor. 10.12] *Let $f \in \mathbb{Z}[X]$ of size (d, τ) and let $h \in \mathbb{Z}[X]$ be a polynomial that divides f in $\mathbb{Z}[X]$. Then h has bitsize in $\mathcal{O}(d + \tau)$.*

We end this subsection with a lemma that we will use to establish a connection between resultants and greatest common divisors (see Rem. 2.25).

Lemma 2.16. [BPR06, Prop. 1.17] *Let $f, g \in \mathbb{Z}[X]$ and h a gcd of f and g. Then, there exist $A, B \in \mathbb{Q}[X]$ with $\deg(A) < \deg(g)$ and $\deg(B) < \deg(f)$ such that:*

$$Af + bg = h\,.$$

2.3.2 Resultant

We consider two non-zero polynomials $f, g \in \mathbb{Z}[X]$ of degrees $p, q \in \mathbb{N}^*$ respectively, expressed as

$$\begin{aligned} f(X) &= a_p X^p + a_{p-1} X^{p-1} + \cdots + a_0\,, \\ g(X) &= b_q X^q + b_{q-1} X^{q-1} + \cdots + b_0\,. \end{aligned} \tag{2.3}$$

Definition 2.17 (Sylvester matrix). *Let $f, g \in \mathbb{Z}[X]$ as in Eq. (2.3). The Sylvester matrix of f and g is the $(p+q)$-square matrix*

$$\mathtt{Syl}(f,g) = \underbrace{\begin{bmatrix} a_p & \cdots & a_0 & \\ & \ddots & & \ddots \\ & a_p & \cdots & a_0 \\ b_q & \cdots & b_0 & \\ & \ddots & & \ddots \\ & b_q & \cdots & b_0 \end{bmatrix}}_{p+q} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} p \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} q \end{matrix}\,.$$

Definition 2.18 (Resultant). *The resultant of f and g, denoted by $\mathtt{res}(f,g)$, is the determinant of the Sylvester matrix $\mathtt{Syl}(f,g)$. When the dependence on the variable X needs to be emphasized, we write $\mathtt{res}_X(f,g)$ instead of $\mathtt{res}(f,g)$.*

The resultant $\mathtt{res}(f,g)$ [JA06] is an integer polynomial in the coefficients of f and g [CLO15, §3.6, Prop.3], i.e.,

$$\mathtt{res}(f,g) \in \mathbb{Z}[a_p, \ldots, a_0, b_q, \ldots, b_0].$$

This follows from the formula for the determinant of a matrix and from the fact that f and g have positive degree.

When f and g do not have both positive degree, the resultant is defined as follows:

$$\begin{aligned} \texttt{res}(a_0, g) &= a_0^q \text{ when } a_0 \neq 0\,, q > 0\,, \\ \texttt{res}(f, b_0) &= b_0^p \text{ when } b_0 \neq 0\,, p > 0\,, \\ \texttt{res}(a_0, b_0) &= 1 \text{ when } a_0, b_0 \neq 0\,. \end{aligned}$$

The resultant can be used to answer the question whether two polynomials $f, g \in \mathbb{Z}[X]$ have a *common factor*, that is when there exists a polynomial in $\mathbb{Q}[X]$ of positive degree that divides both f and g. More precisely, we have the following lemma:

Lemma 2.19. [CLO15, §3.6, Lem. 1] *Let $f, g \in \mathbb{Z}[X]$ be polynomials of degrees $p, q \in \mathbb{N}^*$, respectively. Then, f and g have a common factor in $\mathbb{Q}[X]$ if and only if there exist $A, B \in \mathbb{Q}[X]$ such that:*

(i) A and B are not both zero.

(ii) A has degree at most $p-1$ and B at most $q-1$.

(iii) $Af + Bg = 0$.

So, to see if f and g have a common factor in $\mathbb{Q}[X]$, one could compute their gcd using the Euclidean Algorithm. However this would require divisions in $\mathbb{Q}$, that can be avoided using the following property of the resultants:

Theorem 2.20 (Common factor property). [CLO15, §3.6, Prop. 3] *Let non-zero $f, g \in \mathbb{Z}[X]$. We have that $\texttt{res}(f, g) = 0$ if and only if f and g have a non-trivial common factor in $\mathbb{Q}[X]$.*

Remark 2.21. *Thm. 2.20 suggests that f and g have a common root in $\mathbb{C}$ if and only if their resultant is zero.*

The next property of resultants, offers a link between resultants and elimination.

Theorem 2.22 (Elimination property). [CLO15, §3.6, Prop. 5] *Let $f, g \in \mathbb{Z}[X]$. There exist polynomials A, B in the variable X such that $\texttt{res}(f, g) = Af + Bg$. The coefficients of A and B are integer polynomials in the coefficients of f and g.*

This property can be demonstrated with an example of how resultants can be used to eliminate variables from systems of multivariate polynomials. The resultant of two univariate polynomials is a polynomial with no variables. In multiple variables, this generalizes to the resultant being a polynomial with one variable less.

Let $f(X, Y) = X^2 + 2XY + 1$ and $g(X, Y) = Y^2 - 2X$. We compute the resultant with respect to X and we obtain

$$\texttt{res}_X(f, g) = Y^4 + 4Y^3 + 4 \in \mathbb{Z}[Y].$$

We can deduce from Thm. 2.22 that $\texttt{res}_X(f,g)$ is in the first elimination ideal of f and g. Also, the roots of $\texttt{res}_X(f,g) \in \mathbb{Z}[Y]$ are the values of Y for whom f and g share a non-trivial common factor in $\mathbb{C}[X]$. So, this means that by solving $Y^4 + 4Y^3 + 4 = 0$ we can find the Y-coordinates of the solutions of $f = g = 0$. However, not all the roots of the resultant with respect to X may extend to solution of the system. In particular, we have the following:

Lemma 2.23. [CLO15, §3.6, Cor.7] *Let non-zero $f, g \in \mathbb{Z}[X,Y]$ that have degrees p and q in X respectively. Let also $y_0 \in \mathbb{C}$ such that:*

(i) $f(X, y_0) \in \mathbb{C}[X]$ has degree p and $g(X, y_0) \in \mathbb{C}[X]$ has degree q.

(ii) $\texttt{res}_X(f,g)(y_0) = 0$.

Then, there exists $x_0 \in \mathbb{C}$ such that $f(x_0, y_0) = g(x_0, y_0) = 0$.

The previous lemma is a special case of the specialization property for the evaluation homomorphism, as demonstrated also in the sequel.

Theorem 2.24 (Specialization property). [BPR06, Prop. 8.74] *Let $\phi : D \to D'$ be a ring homomorphism, and let also $\phi : D[X] \to D'[X]$ the induced homomorphism. If $\deg(\phi(f)) = \deg(f)$ and $\deg(\phi(g)) = \deg(g)$, then*

$$\texttt{res}_X(\phi(f), \phi(g)) = \phi(\texttt{res}_X(f,g)).$$

In fact, let $y_0 \in \mathbb{C}$ and the ring homomorphism

$$\begin{aligned} \phi_{y_0} : \quad & \mathbb{Z}[Y] \to \mathbb{C} \\ & Y \mapsto y_0 \end{aligned}$$

and let also $\phi_{y_0} : \quad \mathbb{Z}[Y][X] \to \mathbb{C}[X]$ the induced homomorphism. Let $f, g \in \mathbb{Z}[Y][X]$ of degrees p and q in X respectively such that $\deg(\phi_{y_0}(f)) = p$ and $\deg(\phi_{y_0}(g)) = q$. Then, from Thm. 2.24 we have that

$$\begin{aligned} \texttt{res}_X(\phi_{y_0}(f), \phi_{y_0}(g)) =& \phi_{y_0}(\texttt{res}_X(f,g)) \Leftrightarrow \\ \Leftrightarrow \texttt{res}_X(f(X,y_0), g(X,y_0)) =& \texttt{res}_X(f(X,Y), g(X,Y))(y_0). \end{aligned}$$

If $\texttt{res}_X(f(X,Y), g(X,Y))(y_0) = 0$ then $\texttt{res}_X(f(X,y_0), g(X,y_0))$ is also zero and thus $f(X,y_0), g(X,y_0)$ have a common root (Rem. 2.21).

The next remark exhibits a connection between the resultant and the gcd.

Remark 2.25 (Resultant and gcd). *Let $f, g \in \mathbb{Z}[X]$ with $\texttt{res}(f,g) \neq 0$. Then, this means that f and g do not have a common factor in $\mathbb{Q}[X]$ (Thm. 2.20), so their* gcd *is equal*

to one. From Lem. 2.16, we have that there exist $A, B \in \mathbb{Q}[X]$ such that $Af + Bg = 1$. From Thm. 2.22 we have that there exist $\tilde{A}, \tilde{B} \in \mathbb{Z}[X]$ such that $\tilde{A}f + \tilde{B}g = \mathtt{res}(f, g)$. Thus it is evident that the coefficients of A, B have $\mathtt{res}(f, g)$ as denominator.

Finally, we give some useful expressions of the resultant. Let $\xi_1, \ldots, \xi_p$ be the roots of f and $\eta_1, \ldots, \eta_q$ the roots of g in $\mathbb{C}$ (counted with multiplicities). Then we can express the resultant as a function of the roots in $\mathbb{C}$ follows [BPR06, Thm. 4.14]:

$$\mathtt{res}_X(f, g) = a_p^q b_q^p \prod_{i=1}^{p} \prod_{j=1}^{q} (\xi_i - \eta_j) = a_p^q \prod_{i=1}^{p} g(\xi_i) = (-1)^{p\,q} b_q^p \prod_{j=1}^{q} f(\eta_j)\,.$$

2.3.3 Subresultant sequences

Before defining the subresultants, we start by introducing the concept of a polynomial determinant of a matrix. We refer the reader to [Kah03] for a detailed review of subresultants theory.

Definition 2.26 (Polynomial Determinant). *Let M be a $m \times n$ matrix with $m \leq n$ and M_i be the square submatrix of M consisting of the first $m - 1$ columns and the i-th column of M, for $i = m, \ldots, n$. Then, the polynomial determinant of M is the polynomial defined as*

$$\det(M_m)X^{n-m} + \det(M_{m+1})X^{n-(m+1)} + \cdots + \det(M_n).$$

Definition 2.27 (Sylvester submatrix). *Let $f, g \in \mathbb{Z}[X]$ as in Eq. (2.3) with $p > q$. For $i = 1, \ldots, \min(p, q-1)$, we define the i-th Sylvester matrix $\mathtt{Syl}_i(f, g)$ to be the $(p + q - 2i) \times (p + q - i)$ matrix obtained from $\mathtt{Syl}(f, g)$ by deleting the i last rows of the ones corresponding to the coefficients of f, the i last rows of the ones corresponding to the coefficients of g, and the i last columns.*

Definition 2.28 (Subresultants). *For $i = 0, \ldots, \min(q, p-1)$, the i-the polynomial subresultant of f and g, denoted by $\mathtt{Sres}_{X,i}(f, g)$ is the polynomial determinant of $\mathtt{Syl}_i(P, Q)$. The coefficient of degree i of the polynomial $\mathtt{Sres}_{X,i}(f, g)$, denoted by $\mathtt{sres}_{X,i}(f, g)$, is called the i-th principal subresultant coefficient of f and g.*

In the previous definitions we demand that $p > q$. When $q = p$ the q-th Sylvester matrix is not defined. Following [LPR17] we extend the definition when $p = q$ and we consider $\mathtt{Sres}_{X,q}(f, g) = g$.

Note that $\mathtt{Sres}_{X,0}(f, g) = \mathtt{sres}_{X,0}(f, g)$ is the resultant of f and g with respect to X. By the definition of subresultants it is evident that $\deg(\mathtt{Sres}_{X,i}(f, g)) \leq i$. If $\deg(\mathtt{Sres}_{X,i}(f, g)) = i$ then the subresultant is called *non-defective*. Otherwise, if the degree of $\mathtt{Sres}_{X,i}(f, g))$ is $k < i$, then it is called *defective* of degree k.

The following lemma establishes that the gcd can also be derived from the subresultant sequence.

Lemma 2.29. [BPR06, Cor. 8.58] *The last non-zero signed subresultant of f and g is nondefective and a greatest common divisor of f and g.*

Additionally, there is a close relationship between the subresultant polynomials and the polynomials in the (signed) remainder sequence.

Lemma 2.30. [BPR06, Cor. 8.59] *When for all $j = p, \ldots, 0$ we have that $\mathtt{Sres}_j(f, g)$ is non-defective, then the signed subresultant polynomials are proportional up to a square to the polynomials in the signed remainder sequence.*

To conclude this section, we provide a complexity result for computing the subresultant and principal subresultant coefficients of two (multivariate) polynomials with integer coefficients as it appears in [LPR17, Lem. 4]. This result can be used to determine the complexity of computing the resultant and the gcd of two polynomials.

Theorem 2.31. [BPR06, Prop.8.46],[vzGG13, §11.2] *Let f and g in $\mathbb{Z}[X_1, \ldots, X_n][Y]$, where n is fixed, with coefficients of bitsize τ and degrees in Y bounded by d_Y and the degrees in the other variables are bounded by d.*

(i) The coefficients of $\mathtt{Sres}_{Y,i}(f, g)$ have bitsize in $\widetilde{\mathcal{O}}(d_Y \tau)$.

(ii) The degree in X_j of $\mathtt{Sres}_{Y,i}(f, g)$ is at most $2d(d_Y - i)$.

(iii) For any $i \in \{0, \ldots, d_Y\}$, the polynomials $\mathtt{Sres}_{Y,i}(f, g)$ can be computed in $\widetilde{\mathcal{O}}_B(d^n d_Y^{n+2} \tau)$ bit-operations. The sequence of principal subresultant coefficients $\mathtt{sres}_{Y,i}(f, g)$ can be computed in the same bit-complexity.

2.4 Univariate root isolation

The task of computing the real or complex roots of a univariate polynomial is a well-researched problem in mathematics. When seeking a validated method to compute and represent all roots, it is typical to present the solutions as a collection of separate isolating intervals or boxes, with each interval or box containing exactly one root. This is why the term "(real) root isolation" is frequently used in the literature. The complexity of root isolation algorithms, often relies on certain quantitative results that pertain to the geometric characteristics of the roots, which will be covered in the first section. The second section will focus on the topic of root isolation.

2.4.1 Bounds on univariate polynomials

We begin by establishing notation and presenting definitions for the general case of a polynomial with coefficients in the complex field. Let a univariate polynomial $f = \sum_{i=0}^{d} a_i X^i \in \mathbb{C}[X]$, with leading coefficient $\mathrm{lc}(f) = a_d \neq 0$.

Definition 2.32 (Norms of a polynomial). *Let* $f = \sum_{i=0}^{d} a_i X^i \in \mathbb{C}[X]$.

- *Its* ℓ_1*-norm, or length, is* $\|f\|_1 := \sum_{i=0}^{d} |a_i|$.
- *Its* ℓ_2*-norm, is* $\|f\|_2 := \sqrt{\sum_{i=0}^{d} |a_i|^2}$.
- *Its* ℓ_∞*-norm, is* $\|f\|_\infty := \max_{i\in\{0,\dots,d\}} |a_i|$.

It holds that

$$|f(X)| \leq \|f\|_1 \cdot \max(1, |x|)^d \,.$$

We denote by $V_{\mathbb{C}}(f)$ the complex variety defined by $\langle f \rangle$; in our case it is a finite set of complex numbers. For an $x \in V_{\mathbb{C}}(f)$, we denote by $\mu_f(x)$ its multiplicity as a root of f. It is well known that

$$f(x) = f'(x) = \cdots = f^{(\mu_f(x)-1)}(x) = 0 \text{ and } f^{(\mu_f(x))}(x) \neq 0 \,.$$

We can factorize f in $\mathbb{C}[X]$ as

$$f(X) = \mathrm{lc}(f) \prod_{x \in V_{\mathbb{C}}(f)} (X - x)^{\mu_f(x)} \,,$$

and its *square-free* part is then

$$f^*(X) := \prod_{x \in V_{\mathbb{C}}(f)} (X - x) \,.$$

The *Mahler measure* of f is

$$\mathcal{M}(f) := |\mathrm{lc}(f)| \prod_{x \in V_{\mathbb{C}}(f)} \max(1, |x|)^{\mu_f(x)}.$$

The Mahler measure is multiplicative, i.e., for two polynomials $f, g \in \mathbb{C}[X]$, it holds that

$$\mathcal{M}(f \cdot g) = \mathcal{M}(f) \cdot \mathcal{M}(g) \,.$$

The following inequality bounds the Mahler measure of f by means of its ℓ_1 and ℓ_2 norms [BPR06, Prop.10.8 and Prop.10.9]:

$$2^{-d}\|f\|_1 \leq \mathcal{M}(f) \leq \|f\|_2. \tag{2.4}$$

In particular, if $f \in \mathbb{Z}[X]$ and it is of size (d, τ), the previous inequality becomes

$$2^{-d}\|f\|_1 \leq \mathcal{M}(f) \leq \|f\|_2 \leq 2^{\tau + \log(d+1)}. \tag{2.5}$$

Let a root $x \in V_{\mathbb{C}}(f)$. The *local separation of f at x* is

$$\operatorname{sep}(x, f) := \min_{y \in V_{\mathbb{C}}(f), y \neq x} |y - x|.$$

The *separation* of f is

$$\operatorname{sep}(f) := \min_{x \in V_{\mathbb{C}}(f)} \operatorname{sep}(x, f).$$

Following [DDR+22, Def. 3 and Prop. 1], we introduce the definition of the *generalized discriminant* of a polynomial $f \in \mathbb{C}[X]$, in analogy with the definition of the discriminant.

Definition 2.33 (Discriminant). *Let $f \in \mathbb{C}[X]$ of degree d. The discriminant* $\operatorname{Disc}(f)$ *of f is the element of $\mathbb{C}$ such that*

$$\operatorname{lc}(f)\operatorname{Disc}(f) = \mathtt{sres}_{X,0}(f, f') = (-1)^{d(d-1)/2}\mathtt{res}(f, f')\,.$$

Definition 2.34 (Generalized discriminant). *Let $f \in \mathbb{C}[X]$ of degree d. The generalized discriminant* $\operatorname{GDisc}(f)$ *of f is the element of $\mathbb{C}$ such that*

$$\operatorname{lc}(f) \cdot \operatorname{GDisc}(f) = \operatorname{tc}\left(\mathtt{res}_X\left(f, \sum_{k=1}^{d} U^{k-1} f^{[k]}\right)\right),$$

where $f^{[k]} := \frac{f^{(k)}}{k!}$.

The generalized discriminant of f can also be expressed as [DDR+22, Prop. 1]

$$\operatorname{GDisc}(f) = \operatorname{lc}(f)^{d-2} \prod_{x \in V_{\mathbb{C}}(f)} f^{[\mu_f(x)]}(x)^{\mu_f(x)}\,.$$

When f has only simple roots, i.e., with multiplicity 1, then we have that

$$\operatorname{GDisc}(f) = (-1)^{d(d-1)/2} \cdot \operatorname{Disc}(f)\,.$$

We also define

$$\operatorname{lGDisc}(f) := \sum_{x \in V_{\mathbb{C}}(f)} \mu_f(x) |\log(|f^{[\mu_f(x)]}(x)|)| \text{ and}$$
$$\operatorname{lsep}(f) := \sum_{x \in V_{\mathbb{C}}(f)} \mu_f(x) |\log(\operatorname{sep}(x, f))|\,.$$

The next proposition gives a bound on $\operatorname{lsep}(f)$ that depends on $d, \log \mathcal{M}(f)$ and $\operatorname{lGDisc}(f)$.

Proposition 2.35. [DDR+22, Prop. 2] *For a polynomial $f \in \mathbb{C}[X]$ of degree d with*

$|\mathrm{lc}(f)| \geq 1$, *it holds that*

$$\mathrm{lsep}(f) \in \mathcal{O}(d^2 + d \log \mathcal{M}(f) + \mathrm{lGDisc}(f)).$$

When the polynomial f has integer coefficients, with maximum bitsize τ, we have the following bounds:

$$\|f\|_1 \leq (d+1) \cdot 2^\tau \,,$$
$$\mathcal{M}(f) \leq \sqrt{d+1} \cdot 2^\tau \,.$$

Now, we want to transform the bound for $\mathrm{lsep}(f)$ of Prop. 2.35 into a bound depending on d and τ. We have that

$$\log \mathcal{M}(f) \in \mathcal{O}(\tau + \log d) \,. \tag{2.6}$$

Also, from [DDR+22, Prop. 4], we have that $\mathrm{lGDisc}(f) \in \widetilde{\mathcal{O}}(d\tau + d^2)$. Therefore, we conclude to the following:

Proposition 2.36. *For a polynomial $f \in \mathbb{Z}[X]$ of size (d, τ), it holds that*

$$\mathrm{lsep}(f) \in \widetilde{\mathcal{O}}(d\tau + d^2).$$

Prop. 2.35 and 2.36 give a lower bound on the sum of distances between consecutive roots of a polynomial. A more general result, is the well known *DMM (Davenport-Mahler-Mignotte) bound* that we include here as it appears in [EMT20] (see also [EP17]).

Theorem 2.37 (DMM bound). *Let $f \in \mathbb{C}[X]$ of degree d and $\gamma_1, \ldots, \gamma_d$ its complex roots in ascending magnitude order. Let also Ω be any set of couples of indices (i, j), with $i, j \in [d]$, $i \neq j$ and $|\Omega| = k$. Then*

$$2^k \mathcal{M}(f)^k \geq \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \geq 2^{k-d(d-1)/2} \mathcal{M}(f)^{1-d-k} \sqrt{\mathrm{Disc}(f^*)} \,,$$

where f^* is the square-free part of f. When $f \in \mathbb{Z}[X]$ with $\mathcal{L}(f) = \tau$ and $k \leq d$, we have that

$$d^{d/2} 2^{2d\tau} \geq \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \geq d^{-d} 2^{-d^2 - 6d\tau} \,.$$

The DMM bound shows that the separation of a polynomial is asymptotically equal to all the distances between consecutive roots. This indicates that not all roots of the polynomial can be "too close" to each other. The following classical bound will also be useful.

Lemma 2.38 (Cauchy bound). [BPR06, Cor.10.4] *When $f \in \mathbb{Z}[X]$ and it is of size (d, τ), then for any root γ of f it holds that $|\gamma| \leq 2^{\tau + \log d + 1}$.*

2.4.2 Root isolation

Given a univariate polynomial $f \in \mathbb{C}[X]$ with possibly multiple roots, the goal is to compute isolating discs for its roots, i.e., disjoint disks in the complex plane such that each disk contains exactly one root and the union of all disks covers all the roots (see Fig. 2.3). Multiple references are available that address the problem of root isolation and offer algorithms for solving it [Yap99, KS11, YS11, MSW15].

In this thesis, to isolate the roots of a univariate polynomial with coefficients in $\mathbb{C}$ we use the algorithm of Mehlhorn et al. [MSW15]. The algorithm requires that the number k of distinct roots is known. It first computes an approximate factorization of the polynomial using Pan's algorithm [Pan02]. Assuming that the polynomial is of degree d, from the approximate factorization one obtains approximations $\tilde{z}_1, \ldots, \tilde{z}_d$ of the roots. Then the roots are grouped into k clusters based on geometric vicinity. Every cluster is enclosed by a disc and contains the same number of approximations as roots counted with multiplicity. The next proposition summarizes their result.

Proposition 2.39. [MSW15, Thm. 3], [DDR+22, Prop. 10] *Let $f(x) \in \mathbb{C}[\mathbf{x}]$ of degree $d \geq 2$, for whom it holds that $1/4 \leq \mathrm{lc}(f) \leq 1$. We assume that the number of distinct roots of f is known. We can compute isolating discs for all $x \in V_{\mathbb{C}}(f)$, as well as their multiplicities, in*

$$\widetilde{\mathcal{O}}_B\big(d^3 + d^2 \log \mathcal{M}(f) + \mathrm{lGDisc}(f)\big)$$

bit-operations. As input, we need an oracle giving an absolute L-bit approximation of the coefficients of f with L bounded by

$$\widetilde{\mathcal{O}}\big(d \log \mathcal{M}(f) + \mathrm{lsep}(f) + \mathrm{lGDisc}(f)\big)\,.$$

The previous result states that the coefficients of f are provided by oracles. That is, on input L, the oracle essentially returns binary fraction approximations $\tilde{a}_i$ of the coefficients

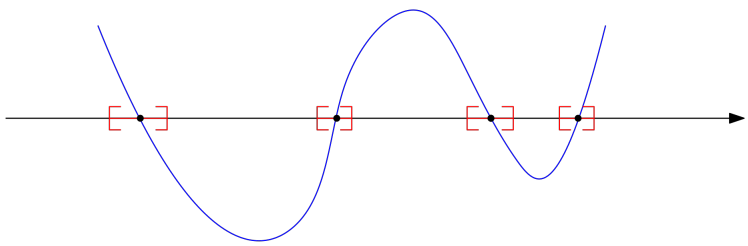


Figure 2.3: The graph of univariate polynomial (in blue) and the isolating intervals of its real roots (in red).

a_i such that

$$\|f - \sum_{i=0}^{d} \tilde{a}_i X^i\|_1 \leq 2^{-L} \|f\|_1 \,.$$

In the special case where the polynomial has integer coefficients we have the following:

Proposition 2.40. [MSW15, Thm. 5] *Let $f(x) \in \mathbb{Z}[\mathbf{x}]$ of degree $d \geq 2$ and bitsize at most τ. We can compute isolating discs for all $x \in V_{\mathbb{C}}(f)$, as well as their multiplicities, in*

$$\widetilde{\mathcal{O}}_B\left(d^3 + d^2\tau\right)$$

bit-operations. For a given positive integer L, we can further refine the isolating disks to a size of less than 2^{-L} with a number of bit operations bounded by

$$\widetilde{\mathcal{O}}_B\left(d^3 + d^2\tau + dL\right) \,.$$

Notice that in the previous theorem, the number of distinct roots is not needed. This is because we can compute the squarefree part f^* of f in $\widetilde{\mathcal{O}}_B(d^2\tau)$. Then, f^* has bitsize in $\mathcal{O}(d+\tau)$ and its degree gives the number of distinct roots of f. The coefficients of f can then be scaled so that $\mathrm{lc}(f)$ will belong in the interval $[1/4, 1]$.

2.5 Zero-dimensional polynomial systems

Let $f_1, \ldots, f_k \in \mathbb{Q}[X_1, \ldots, X_n]$. We denote by $V_{\mathbb{C}}(f_1, \ldots, f_k)$ the set of zeros of $f_1, \ldots, f_k$ in $\mathbb{C}^n$, that is the solution set in $\mathbb{C}^n$ of the system

$$\begin{aligned} f_1(X_1, \ldots, X_n) &= 0\,, \\ f_2(X_1, \ldots, X_n) &= 0\,, \\ &\vdots \\ f_k(X_1, \ldots, X_n) &= 0\,, \end{aligned} \tag{2.7}$$

in $\mathbb{C}^n$.

The system is *zero-dimensional* if it has a finite number of solutions in $\mathbb{C}^n$, i.e., if $V_{\mathbb{C}}(f_1, \ldots, f_k)$ is a non-empty finite set. In particular, we have the following:

Theorem 2.41. [BPR06, Thm.4.86] *Let $f_1, \ldots, f_k \in \mathbb{Q}[X_1, \ldots, X_n]$. The vector space $A = \mathbb{Q}[X_1, \ldots, X_n]/\langle f_1, \ldots, f_k\rangle$ is of finite dimension if and only if the system $\{f_1 = \cdots = f_k = 0\}$ is zero-dimensional. The number of elements of $V_{\mathbb{C}}(f_1, \ldots, f_k)$ is less than or equal to the dimension of A as a $\mathbb{Q}$-vector space.*

We denote by $\bar{A}$ the quotient ring $\mathbb{C}[X_1, \ldots, X_n]/\langle f_1, \ldots, f_k\rangle$, where $\langle f_1, \ldots, f_k\rangle$ in now considered an ideal in $\mathbb{C}[X_1, \ldots, X_n]$. Since A is a $\mathbb{Q}$-vector space and $\mathbb{C}$ is a field containing $\mathbb{Q}$, we have that $\bar{A}$ is a $\mathbb{C}$-vector space. Moreover, we have the following:

Lemma 2.42. [BPR06, Lem. 4.87 and Lem. 4.88] *It holds that $A \subset \bar{A}$. Moreover, A is a finite dimensional vector space of dimension m over $\mathbb{Q}$ if and only if $\bar{A}$ is a finite dimensional vector space of dimension m over $\mathbb{C}$.*

For the quotient ring $\bar{A}$, its localisation at $\mathbf{x} \in V_{\mathbb{C}}(I)$ is denoted by $\bar{A}_{\mathbf{x}}$. The local rings $\bar{A}_{\mathbf{x}}$ are also $\mathbb{C}$-vector spaces. Then, we have the following result:

Theorem 2.43. [BPR06, Thm. 4.95] *It holds that*

$$\bar{A} = \prod_{\mathbf{x} \in V_C(I)} \bar{A}_{\mathbf{x}} \,.$$

Now, we are able to define the multiplicity of a root of a system of polynomial equations.

Definition 2.44 (Multiplicity). *Let $\mathbf{x} \in V_{\mathbb{C}}(f_1, \ldots, f_k)$. The multiplicity of $\mathbf{x}$ is the dimension of the local ring $\bar{A}_{\mathbf{x}}$ as a $\mathbb{C}$-vector space.*

From Thm. 2.43 we have that the dimension of $\bar{A}$, and consequentially of A, is equal to the sum of multiplicities of all $\mathbf{x} \in V_{\mathbb{C}}(I)$. In other words, it is equal to the number of roots of I counted with multiplicity. We also note that for the system of Eq. (2.7) to be zero-dimensional, k has to be equal to n. Now, supposing that $f_1, \ldots, f_n$ are of degree d, the total number of roots, from Bézout's theorem [BPR06, Thm. 4.108], is at most d^n.

On computing the roots of the zero-dimensional system of Eq. (2.7) there is a great number of approaches, essentially aiming at a representation of the roots either in a formal way or a numerical one. For a comprehensive overview we refer the reader to [CCC+05, Stu02, CLO05].

For the rest of this section we present briefly how resultants can be used to obtain projections of the roots and how roots can be represented through triangular systems. Another formal representaion of the roots can be obtained through the *Rational Univariate Representation* (RUR), reducing zero-dimensional polynomial systems to the study of one single univariate polynomial. This will be the subject of the next subsection.

Projections via resultants

When $n = 2$ we saw in Sec. 2.3.2 how resultants can be used to get the projections of the coordinates of $V_{\mathbb{C}}(f_1, f_2)$. In an analogous manner to univariate resultants, resultants of several multivariate polynomials also exist [CLO05, §3.2] and they can be used to obtain projections of coordinates of the roots of the zero-dimensional system in Eq. (2.7) (for $k = n$).

Roughly speaking, given $n + 1$ polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$, their resultant is generically an irreducible polynomial in the coefficients of the polynomials and it can be computed using determinants (at least in many cases). In particular, it is a factor of the determinant

of a Sylvester-type matrix (the Macaulay matrix) [EP05, CLO05]. The vanishing of the resultant provides a necessary and sufficient condition for the existence of roots of the homogenized system in $\mathbb{P}^n$.

Multivariate resultants can be employed to find the projections of roots of the system in Eq. (2.7) on the coordinate axes. It is the so-called *hidden variable* method [CLO05, §3.5]. Essentially the polynomials are considered in $n-1$ variables, say $X_1, \ldots, X_{n-1}$, with coefficients in $\mathbb{Q}[X_n]$. So, X_n is the hidden variable. Then we compute the resultant of $f_1, \ldots, f_n$, which is a polynomial in $\mathbb{Q}[X_n]$. Among its roots are the X_n-coordinates of the solutions of the system. However, not all the roots of the resultant may extend to roots of the system [BS16, Lem. 1], but this can be ensured. A drawback of this approach is that it solely provides the separate coordinates of the solution points without indicating their correspondence with each other.

Another closely-related method using multivariate resultants, is the *u-resultant* ([CLO05, §3.5],[Can88]), according to which the process of solving the system is reduced to multivariable factorization. The u-resultant method has significant drawbacks. Primarily, it requires calculating symbolic determinants of large size. Moreover, performing multivariable factorization is hard; even if we can calculate the determinant, it may be challenging to apply the method in practice. In contrast, the hidden variable method has less computational complexity since it requires computing resultants of fewer equations and variables than the u-resultant approach.

Triangular systems

One way to represent algebraically the solutions of a zero-dimensional polynomial system, is through one of several *regular triangular systems* [Laz92, LPR17, ALM99, CGY09]. A triangular system with n variables is of the form

$$\begin{aligned} g_1(X_1) &= 0\,, \\ g_2(X_1, X_2) &= 0\,, \\ &\vdots \\ g_n(X_1, \ldots, X_n) &= 0\,, \end{aligned}$$

where $g_i \in \mathbb{Q}[X_1, \ldots, X_i]$. Moreover, we call the system regular when the leading coefficient of g_i in $\mathbb{Q}[X_1, \ldots, X_{i-1}][X_i]$, that is when it is viewed as a univariate polynomial in the variable X_i, does not have a common zero with $g_1, \ldots, g_{i-1}$.

This representation can be obtained by a Gröbner basis. It can be combined with numerical methods for obtaining isolating boxes of the roots [CGY09]; for every triangular set one starts by finding isolating boxes for the first univariate polynomial, and then substitutes approximations of the roots to the second polynomial and so on so forth.

So, by successive substitutions one solves only univariate polynomials. However, since after the first solutions the coefficients are no longer integers, successive substitutions are highly unstable. For a non-numerical solution, one must perform computations modulo a triangular set [LMP09, DSM+05]. One can also compute a univariate representation of the roots of the triangular set using the algorithm of [Laz92] in a number of arithmetic operations that is polynomial in d^n ([Laz92, Prop. 7],[Lak91]) (where d is an upper bound on the degree of the polynomials), thus nearly optimal since the solutions are at most d^n (see also [Ben19] for the sparse case).

2.5.1 Rational Univariate Representation

In this subsection, we consider a zero-dimensional ideal $I \in \mathbb{Q}[X_1, \ldots, X_n]$. A Rational Univariate Representation (RUR) of $V_{\mathbb{C}}(I)$ is a representation of the roots of I using only one variable. It is obtained by expressing every coordinate of the roots as a rational function evaluated at the zeros of the same univariate polynomial; the degree of the latter polynomial is equal to the number of solutions of the system, counting multiplicities.

We recall basic definitions and the algorithm for the RUR computation of I, from the multiplication tensor of the quotient algebra $\mathbb{Q}[X_1, \ldots, X_n]/I$, as presented in [Rou99]. We start with the definition of a separating element of $V_{\mathbb{C}}(I)$ which is necessary for the RUR computation, whose definition also follows.

Definition 2.45 (Separating form). [Rou99, Def. 2.1] *Let $I \subseteq \mathbb{Q}[X_1, \ldots, X_n]$ be a zero dimensional ideal and $t \in \mathbb{Q}[X_1, \ldots, X_n]$. We say that t is separating for $V_{\mathbb{C}}(I)$ if the map $\mathbf{x} \mapsto t(\mathbf{x})$ is injective on $V_{\mathbb{C}}(I)$.*

Finding a *separating linear form* (SLF for short) is always possible, and in particular we have the following result:

Lemma 2.46. [Rou99, Lem. 2.1] *If $\#V_{\mathbb{C}}(I) = D$, then the family of linear forms*

$$\left\{ X_1 + iX_2 + \cdots + i^{n-1}X_n\,, \quad i = 1, \ldots, (n-1)\,\frac{D(D-1)}{2} \right\}$$

contains at least one linear form that is separating for $V_{\mathbb{C}}(I)$.

In the sequel, for any $v \in \mathbb{Q}[\mathbf{X}]$ and $\sigma \in \mathbb{C}^n$, we denote by $v(\sigma)$ the evaluation of the polynomial function v at σ.

Definition 2.47 (Rational Univariate Representation). [Rou99, Def.3.3] *Let $I \in \mathbb{Q}[X_1, \ldots, X_n]$ be a zero-dimensional ideal. Let $t = \mathbf{a} \cdot \mathbf{X}$ be a linear form in $\mathbb{Q}[X_1, \ldots, X_n]$*

and $\mathbf{a} = (1, a_2, \ldots, a_n) \in \mathbb{Q}^n$. *We define the polynomials:*

$$f_{I,\mathbf{a}}(T) = \prod_{\sigma \in V_{\mathbb{C}}(I)} (T - X_1(\sigma) - a_2 X_2(\sigma) - \ldots a_n X_n(\sigma))^{\mu_I(\sigma)},$$

$$f_{I,\mathbf{a},v}(T) = \sum_{\sigma \in V_{\mathbb{C}}(I)} \mu_I(\sigma) v(\sigma) \prod_{s \in V(I) \setminus \{\sigma\}} (T - X_1(s) - a_2 X_2(s) - \ldots a_n X_n(s)),$$

for $v \in \{1, X_1, \ldots, X_n\}$. *The* $n+2$*-tuple* $\{f_{I,\mathbf{a}}, f_{I,\mathbf{a},1}, f_{I,\mathbf{a},X_1} \ldots, f_{I,\mathbf{a},X_n}\}$ *is called RUR candidate of* I *associated to* $t = \mathbf{a} \cdot \mathbf{X}$. *If* t *is separating for* $V_{\mathbb{C}}(I)$ *then it is called RUR of* I *associated to* t.

It is proven [Rou99, Thm. 3.1] that the polynomials $\{f_{I,\mathbf{a}}, f_{I,\mathbf{a},1}, f_{I,\mathbf{a},X_1} \ldots, f_{I,\mathbf{a},X_n}\}$ are in $\mathbb{Q}[T]$ and that, if the linear form $t = \mathbf{a} \cdot \mathbf{X}$ is separating, the coordinates of the roots of I are expressed as follows:

$$X_1 = \frac{f_{I,\mathbf{a},X_1}(T)}{f_{I,\mathbf{a},1}(T)}, \ldots, X_n = \frac{f_{I,\mathbf{a},X_n}(T)}{f_{I,\mathbf{a},1}(T)} \quad \text{for } T \text{ such that } f_{I,\mathbf{a}}(T) = 0\,.$$

Now, given a linear form

$$t = \mathbf{a} \cdot \mathbf{X} = \sum_{i=1}^{n} a_i X_i \in \mathbb{Q}[X_1, .., X_n],$$

we describe how the RUR candidate associated to it can be computed.

Notation. Let $D = \dim(\mathbb{Q}[X_1, \ldots, X_n]/I)$. For $P \in A$, where A is an F-algebra of finite dimension, where F is a field, we denote by M_P^A be the multiplication matrix by P in A, by $E_P^A(T)$ the (unitary) *characteristic polynomial* of M_P and by $\mathrm{Tr}^A(P)$ its trace. The superscript can be omitted when $A = \frac{\mathbb{Q}[X_1,..,X_n]}{I}$. For a linear form $t = \mathbf{a} \cdot X \in \frac{\mathbb{Q}[X_1,..,X_n]}{I}$, we use a slightly different notation and we let $E_{\mathbf{a}}(T)$ be the characteristic polynomial of M_t. Let $E_{\mathbf{a}}^*(T)$ be its squarefree part, i.e., $E_{\mathbf{a}}^*(T) = \sum_{i=0}^{d'} b_i T^{d'-i}$, where d' is its the degree. Let $H_j(T) = \sum_{i=0}^{j-i} b_i T^{j-i}$ be the j-th Horner polynomial associated to $E_{\mathbf{a}}^*(T)$. For all $v \in \mathbb{Q}[X_1, \ldots, X_n]$, let

$$h_{\mathbf{a},v}(T) = \sum_{i=0}^{d'} \mathrm{Tr}(vt^i) H_{d'-i-1}(T)\,.$$

As shown in [Rou99], we have that

$$f_{I,\mathbf{a}}(T) = E_{\mathbf{a}}(T)\,,$$
$$f_{I,\mathbf{a},v}(T) = h_{\mathbf{a},v}(T) \text{ for } v \in \{1, X_1, \ldots, X_n\}\,.$$

So, given a linear form $t = \mathbf{a} \cdot \mathbf{X}$ (not necessarily separating), finding a RUR candidate

amounts to computing:

1. The characteristic polynomial $E_{\mathbf{a}}(T)$ of the multiplication by $t = \mathbf{a} \cdot \mathbf{X}$ in $\frac{\mathbb{Q}[X_1,\ldots,X_n]}{I}$ and its squarefree part. From the latter, we deduce the Horner polynomials associated to $E_{\mathbf{a}}(T)$.

2. The traces $\mathrm{Tr}(vt^i)$, for $v \in \{1, X_1, \ldots, X_n\}$.

The traces $\mathrm{Tr}(t^i)$ are the Newton sums of $E_{\mathbf{a}}(T)$ and therefore suffice for computing $E_{\mathbf{a}}(T) = \sum_{i=0}^{D} \alpha_i T^{D-i}$, by solving the triangular system

$$(D-i)b_i = \sum_{j=0}^{i-1} \alpha_{i-j} \mathrm{Tr}(t^j)\,, \quad i = 0, \ldots, D\,. \tag{2.8}$$

Alg. 1 summarizes the computation of an RUR candidate. Given the required traces, the number of arithmetic operations is given in the lemma that follows.

Lemma 2.48. [Rou99, Prop.4.1] *Let $t = \mathbf{a} \cdot \mathbf{X}$ a linear form. Given $\mathrm{Tr}(t^i)$ and $\mathrm{Tr}(X_j t^i)$ for $j = 1, \ldots, n$ and $i = 0, \ldots, D$, we compute a RUR candidate of I in $\mathcal{O}(nD^2)$ arithmetic operations.*

Proof. We need $\mathcal{O}(D^2)$ arithmetic operations for the resolution of the triangular system of Eq. (2.8) to compute $E_{\mathbf{a}}(T)$. Then, in $\mathcal{O}(D^2)$ arithmetic operations we compute the squarefree part of $E_{\mathbf{a}}(T)$, in $\mathcal{O}(D^2)$ the Horner polynomials H_j for $j = 0, \ldots, d'-1$, where d' is the degree of $E^*_{\mathbf{a}}(T)$, and in $\mathcal{O}(nD^2)$ we compute $f_{I,\mathbf{a},v}(T)$ for $v \in \{1, X_1, \ldots, X_n\}$. □

Algorithm 1: `RUR candidate computation`

Input: I, linear form $t = \mathbf{a} \cdot \mathbf{X}$
Output: RUR candidate of I

1 Compute $\mathrm{Tr}(t^i), \mathrm{Tr}(X_j t^i)$ for $i \in [D], j \in [n]$, where $D = \#V_{\mathbb{C}}(I)$
2 Solve the system $(D-i)b_i = \sum_{j=0}^{i-1} b_{i-j}\mathrm{Tr}(t^j)\,, \quad i = 0, \ldots, D$
3 Set $f_{I,\mathbf{a}}(T) = \sum_{i=0}^{D} b_i T^i$
4 Compute the Horner polynomials $H_j(T) = \sum_{i=0}^{j-i} b_i T^{j-i}$ associated to $f_{I,\mathbf{a}}(T)$
5 Let d' is the degree of the squarefree part of $f_{I,\mathbf{a}}(T)$
6 Set $f_{I,\mathbf{a},v}(T) = \sum_{i=0}^{d'-1} \mathrm{Tr}(vt^i) H_{d'-i-1}(T)$ for $v \in \{1, X_1, \ldots, X_n\}$
7 RETURN $\{f_{I,\mathbf{a}}, f_{I,\mathbf{a},1}, f_{I,\mathbf{a},X_1}, \ldots, f_{I,\mathbf{a},X_n}\}$

We saw that the RUR computation of an ideal I, decomposes to the computation of a separating element for $V_{\mathbb{C}}(I)$ and to the computation of a RUR candidate. One can compute the separating linear form in advance, or take a random linear form $t \in \mathbb{Q}[\mathbf{X}]$, and then check if it is separating for $V_{\mathbb{C}}(I)$ or not given the RUR candidate associated to t. We refer the reader to [Rou99, Rou07, BLPR15, BLM+15, MST17] for more details on the RUR computation algorithms.

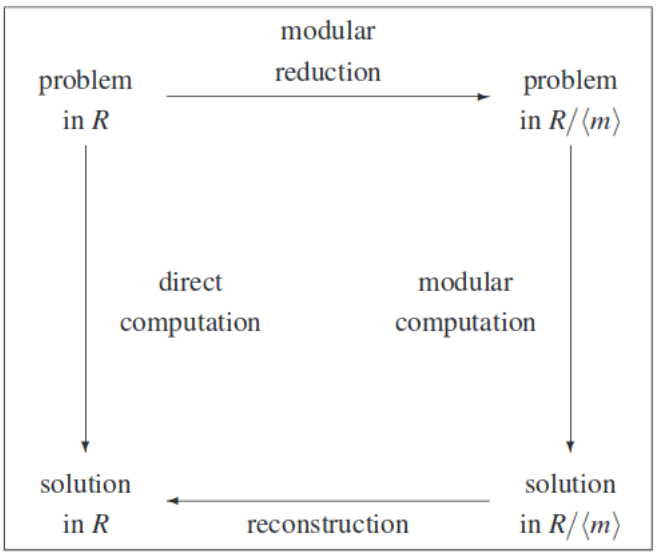


Figure 2.4: General scheme for big prime modular algorithms, where R is a Euclidean domain (Figure take from [vzGG13, Fig. 5.1]).

2.6 Modular algorithms

In this section, we briefly review the useful concept in computer algebra of *modular algorithms* ([vzGG13, §5],[Yap99, §4]). We will explore two different versions of modular algorithms, where instead of solving problem over $\mathbb{Z}$ (or in general over a Euclidean domain), a problem can be solved modulo a big prime number *(big prime version)* or several small primes *(small primes version)*. Then, under certain conditions the solution in $\mathbb{Z}$ can be reconstructed from the modular images. In the small primes version, a basic ingredient of the modular approach is the application of the Chinese Remainder Theorem [vzGG13, Cor. 5.3]; we solve the problem modulo a set of primes $p_1, \ldots, p_r$, then from the Chinese Remainder Theorem, we can obtain the solution to $\mathbb{Z}_p$, where $p = \prod_{i=1}^{r} p_i$, and from that we reconstruct the solution to $\mathbb{Z}$. The general scheme of the big prime and small primes version is summarized in Fig. 2.4 and Fig. 2.5 respectively.

One advantage of modular computations is that they can be used to avoid *intermediate coefficient swell*; during the computations of an algorithm, the coefficients of the intermediate results may become a lot bigger than the size of the output. But if we perform the computations modulo a certain number, the size of coefficients is controlled.

Let $\varphi_p : \mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ the reduction modulo p morphism. There are two main concerns in the design of modular algorithms; First, a chosen prime p has to be such that the solution over $\mathbb{Z}_p$ is the reduction modulo p of the solution over $\mathbb{Z}$. Not all prime numbers have this property. We call a prime number *lucky* for an algorithm, if the algorithm's computations commute with the morphism φ_p and *unlucky* otherwise. The second concern is that an upper bound on the size of the integers in the output has to be determined. This is so

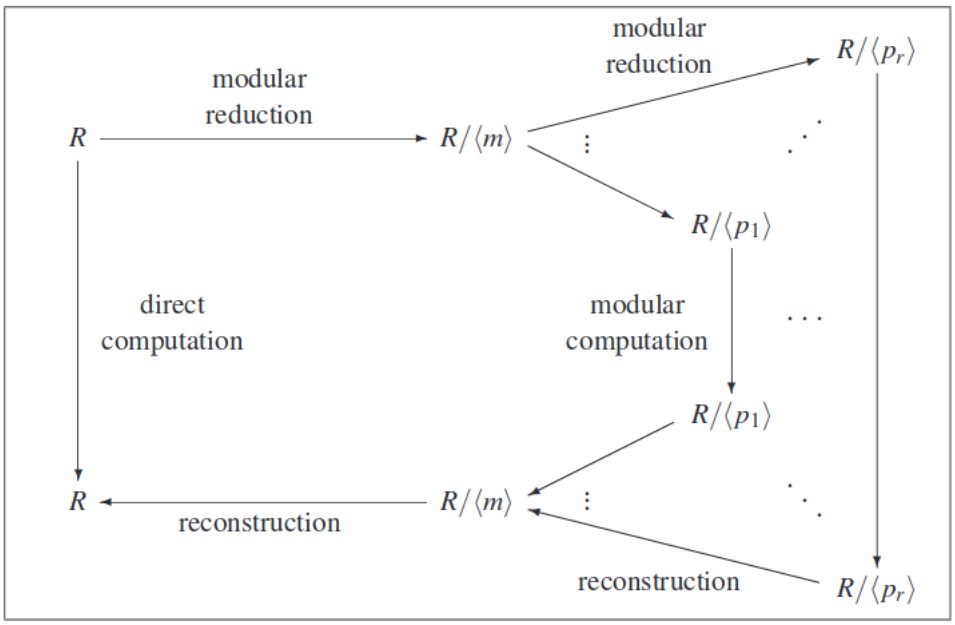


Figure 2.5: General scheme for small primes modular algorithms, where R is a Euclidean domain (Figure take from [vzGG13, Fig. 5.2]).

as to ensure that the result can be recovered from its modular image; in the big prime version, one must take a prime p larger that this bound and in the small primes version, the product of the chosen primes has to be larger than this bound.

In what follows, we showcase the basic ingredients of modular algorithms using the example of modular gcd computation.

Modular gcd computation. A classical application of modular algorithms is on the gcd computation of two polynomials $f, g \in \mathbb{Z}[X]$. From the Mignotte bound (Thm. 2.15), we have an upper bound on the coefficient size of the gcd; if f and g have size (d, τ), then the bitsize of the gcd is in $\widetilde{\mathcal{O}}(d + \tau)$. Nevertheless, during the computations of the traditional Euclidean algorithm, the size of the coefficients seems to be exponential in the size of the gcd (see for example [vzGG13, Example 6.1] and [vzGG13, Thm. 6.52]). Modular algorithms can assist in overcoming this issue.

Lucky primes for the gcd computation as defined as follows.

Definition 2.49. [Yap99, §4.4] *Let $f, g \in \mathbb{Z}[X]$. A prime number p is lucky for the gcd of f and g if $\varphi_p(\mathrm{lc}(f) \cdot \mathrm{lc}(g)) \neq 0$ and $\gcd(f, g)$ has the same degree as $\gcd(\varphi_p(f), \varphi_p(g))$.*

The problem with this definition is that it is not helpful for computing the gcd modulo a prime number, since the definition of the lucky prime assumes that the gcd is already known. The following proposition is useful:

Proposition 2.50. [Yap99, Lemmata 4.11 and 4.12] *The lucky primes for the gcd of f and g are the ones that do not divide neither $\mathrm{lc}(f)$, nor $\mathrm{lc}(g)$, nor the d-th principal subresultant*

coefficient of f and g, where d is the degree of gcd. The product of all the unlucky primes is

$$\Pi \leq N^{2D+2}$$

where $N = \max(\|f\|_2, \|g\|_2)$ and D is the maximum of the degrees of f and g.

As the previous proposition reveals, the number of unlucky primes is finite. So, if we choose a random prime there is high probability of success. We can check if the result is correct and, otherwise, repeat for another random prime. This method, combined with fast techniques for polynomial and integer arithmetic [vzGG13, §11], leads to an efficient modular Las-Vegas algorithm.

Theorem 2.51. [vzGG13, Cor. 11.14] *Let $f, g \in \mathbb{Z}[X]$ of degree d and bitsize τ. Then, their gcd can be computed with a Las-Vegas algorithm with expected bit-complexity $\widetilde{\mathcal{O}}_B(d^2 + d\tau)$.*

First Part

Polynomial System Solving

Chapter 3

Isolating Roots in a Multiple Field Extension

We address univariate root isolation when the polynomial's coefficients are in a multiple field extension. We consider a polynomial $F \in L[Y]$, where L is a multiple algebraic extension of $\mathbb{Q}$. We provide aggregate bounds for F and algorithmic and bit-complexity results for the problem of isolating its roots.

For the latter problem we follow a common approach based on univariate root isolation algorithms. For the particular case where F does not have multiple roots, we achieve a bit-complexity in $\widetilde{\mathcal{O}}_B(nd^{2n+2}(d+n\tau))$, where d is the total degree and τ is the bitsize of the involved polynomials. In the general case we need to enhance our algorithm with a preprocessing step that determines the number of distinct roots of F.

On realising this step, we follow two approaches. The first, is certified numerical method that has bit-complexity in $\widetilde{\mathcal{O}}_B(n^2d^{3n+3}\tau+n^3d^{2n+4}\tau)$. It essentially uses sufficiently good approximations of the roots of the minimal polynomials of L. The second one, is a formal approach that transforms the multiple extension to a simple extension through the computation of a Rational Univariate Representation of the roots of the system defined by the minimal polynomials of L. It leads to a Las-Vegas algorithm of expected complexity in $\widetilde{\mathcal{O}}_B(n(n+2^n)d^{3n-1}(d+\tau)+n^{10})$ for the values of n that are greater than 6.

3.1 Introduction

We consider the problem of isolating the (complex) roots of a univariate polynomial over a multiple algebraic field extension –the coefficients of the polynomial are multivariate polynomial functions evaluated at algebraic numbers–. Solving in a field extension is a common problem in computational mathematics; for example it arises in the topology computation of plane curves [DDR$^+$22, KS12], in the convex hull computation of parametric curves (Ch. 5), or it can be seen as a sub-problem in the resolution of triangular

systems [CM12, XY02] and regular chains [BCLM10].

For $n \geq 2$, we consider $F_1 \in \mathbb{Z}[X_1], \ldots, F_n \in \mathbb{Z}[X_n]$ univariate polynomials of degree at most M and bitsize Λ and $F \in \mathbb{Z}[X_1, \ldots, X_n, Y]$ of total degree at most d and bitsize τ. We want to isolate the roots of the system

$$\left\{ \begin{matrix} F_1(X_1) = 0\,, \ldots, F_n(X_n) = 0\,, \\ F(X_1, \ldots, X_n, Y) = 0\,. \end{matrix} \right\} \tag{3.1}$$

In theory, we can solve the system as follows: first, we isolate the roots of all the univariate polynomials $F_1, \ldots, F_n$. Then, for every root $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{C}^n$ of $\{F_1 = \cdots = F_n = 0\}$ we employ Pan's algorithm [Pan02] for the approximate factorization of the univariate $F(\mathbf{x}, Y)$, with the worst case precision; the approximate factorization algorithm returns as many root approximations as the degree of $F(\mathbf{x}, Y)$ in Y is, and the fact that we used the worst case precision allows to determine which root approximations correspond to the same root. This method leads to good worst-case bit-complexity estimates (Rem. 3.10). Nevertheless, this is at the price of requiring always to perform computations using the maximum precision and it cannot lead to a practical algorithm. For example, if $n = 2$, $d = 10$, and $\tau \in \mathcal{O}(1)$, then we have to work with $> 10^4$ bits in all of our computations. Our goal is to introduce an adaptive algorithm, depending on the multiplicities of the roots and on their pairwise distances, so we will follow a different approach.

Isolating the roots of the system in Eq. (3.1) has not been treated in the literature currently, but only in a simplified setting (e.g. [Rum77, JK97]). In [ST12] they consider the same problem when F does not have multiple roots; it is a generalization of a prior work for a simple algebraic extension [ST19, JK97]. They propose three methods. The first one computes the minimal polynomial of the system and uses multivariate resultants. The second one is based on Sturm's algorithm and the third one on solving directly the polynomial using univariate root isolation algorithms, similarly to ours. The last method is the most efficient with a bit-complexity of $\widetilde{\mathcal{O}}_B(n^3 N^{2n+3})$, where N is a bound on the size of the input polynomials, without any assumptions on the input. In [DDR+22], it is the first time the general problem, in the simple extension setting, is addressed in literature [KS12, ST19, MSW15, KS15]. The authors provide precise amortized separation bounds for the polynomial and complexity bounds for isolating the roots. We provide further details on their method in the sequel, since we share many ideas. We could also solve the system in Eq. (3.1) by applying a general algorithm for zero-dimensional square systems of an expected complexity in $\widetilde{\mathcal{O}}_B\left((n+1)^{n(\omega+1)+2} N^{(\omega+2)n+2}\right)$, where ω denotes the exponent in the complexity of matrix multiplication [BS16]. However, such a method would not exploit the special structure of the system.

Our approach and contribution

We generalize the results of [DDR$^+$22] for any $n > 1$. By following closely their techniques, we are able to provide amortized bounds on the separation of F. On solving the system of Eq. (3.1), the idea is to approximate the coefficients of F for every $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{C}^n$ that is root of $F_1 = \ldots F_n = 0$, up to a certain precision, so that to isolate the roots of $F(\mathbf{x}, Y)$, it suffices to isolate the roots of its approximation. The amortized bounds that we prove in Cor. 3.4 and Cor. 3.6, quantify the required precision. To find the roots of the approximation of $F(\mathbf{x}, Y)$ we can now use algorithms for univariate root isolation. Particularly, we employ the algorithm of [MSW15] that builds upon the algorithm of approximate factorization of a polynomial of Pan [Pan02]; if a univariate polynomial is of degree d, the approximate factorization algorithm returns d root approximations. Then, the approximations must be clustered in a way so that each cluster corresponds to a root, and it contains as many root approximations as the multiplicity of the corresponding root. In [MSW15] they run Pan's algorithm multiple times with increasing precision. For a stopping criterion, they require the number of distinct roots of the polynomial. Therefore, in our case, we should also compute the number of distinct roots of $F(\mathbf{x}, Y)$ for every root $\mathbf{x} \in \mathbb{C}^n$ of $F_1 = \ldots = F_n = 0$. This dominates the total bit-complexity.

In Sec. 3.4, we compute the number of distinct roots of $F(\mathbf{x}, Y)$ using a numerical approach (Lem. 3.13). We compute the principal subresultant coefficients of F and $\frac{\partial F}{\partial Y}$ with respect to Y. Then, for every root $\mathbf{x}$ of $F_1 = \ldots = F_n = 0$, we approximate the principal subresultant coefficients up to the necessary precision so that we can determine their sign correctly. The index of the first non-zero subresultant coefficient gives the degree of the gcd of $F(\mathbf{x}, Y)$ and $\frac{\partial F(\mathbf{x},Y)}{\partial Y}$, and thus the number of distinct roots. The total bit-complexity of solving the system of Eq. (3.1) then is described in Thm. 3.7(ii). In Rem. 3.8, we give for simplicity the bound for the case when all the polynomials are of have degree at most d and bitsize τ, which is $\widetilde{\mathcal{O}}_B\left(n^2 d^{3n+3}\tau + n^3 d^{2n+4}\tau\right)$. On the contrary, when the number of distinct roots of $F(\mathbf{x}, Y)$ is known for every $\mathbf{x}$, or when F is does not have multiple roots, we can isolate the roots of the system in $\widetilde{\mathcal{O}}_B(nd^{2n+2}(d+n\tau))$ bit operations. This is to be juxtaposed with the result of [ST12]; it improves it by a factor of n.

In Sec. 3.5 we follow an alternative, formal, approach on the number of distinct roots computation. We find a Rational Univariate Representation (RUR) [Rou99] of the roots $\mathbf{x}$ of $F_1 = \ldots = F_n$ (Thm. 3.35) We use the algorithm of [Rou99] and we analyse its bit-complexity for this specific case, which is of independent interest. In particular, we are able to compute the RUR fast since the multiplication matrices in $\mathbb{Q}[\mathbf{X}]/\langle F_1, \ldots, F_n\rangle$ are sparse and multiplying by a variable in $\mathbb{Q}[\mathbf{X}]/\langle F_1, \ldots, F_n\rangle$ requires a linear number of arithmetic operations with respect to the dimension of the matrix. The RUR allows then to transform the problem of solving in a multiple extension to solving in a simple extension. So, we end up with a bivariate system in triangular form. We use this system to find the number of distinct roots through gcd computations. In this case, our algorithm

for the root isolation is a Las-Vegas algorithm of expected complexity in

$$\widetilde{\mathcal{O}}_B(n^3 d^{3n} + n^2 d^{3n-1}\tau + n^2 d^{2n+2}\tau + nd^{2n+5} + d^{2n+4}\tau)$$

when $n \geq 6$ and $\widetilde{\mathcal{O}}_B(d^{2n+5}(d+\tau))$ otherwise (Rem. 3.8).

In Sec. 3.6 we apply the amortized bounds for F on the 'Sum of Square Roots of Integers' Problem. Comparing the length of two paths in the Euclidean Travel Salesman Problem (TSP), relates also to this problem. It has been already studied through the separation bound computation of the associated polynomial system [BFMS00, MS00]. Our approach matches the latter results, and, even more, the proven bounds are aggregate.

Outline. In Sec. 3.2 we prove a lemma on multipoint evaluation that will be useful in the sequel. In Sec. 3.3 we prove amortized bounds for a polynomial in a multiple field extension and in Sec. 3.4 we analyze the bit-complexity of solving the system of Eq. (3.1). In Sec. 3.5, we present the alternative Las Vegas algorithm to solve the system using a RUR. In Sec. 3.6 we apply the bounds of Sec. 3.3 to the 'Sum of Square Roots of Integers' problem.

3.2 Prerequisite: Multipoint evaluation at a Cartesian product in $\mathbb{C}^n$

If we want to evaluate a univariate polynomial $f \in \mathbb{C}[X]$ of degree d at some numbers $a_1, \ldots, a_D \in \mathbb{C}$, $D \in \mathbb{N}$, we can use *fast multipoint evaluation* [KS13] (see also Sec. 2.2.2). When $D \leq d$ the bit-complexity of the multipoint evaluation is given in [KS13, Thm.9] (see also Lem. 2.12). When $D > d$, we have to repeat multipoint-evaluation $\lceil \frac{D}{d} \rceil$ times.

Now, we want to evaluate a multivariate polynomial $f \in \mathbb{C}[\mathbf{X}]$ at $\mathbf{a}_1, \ldots, \mathbf{a}_D \in \mathbb{C}^n$. As discussed in [vL20], multipoint evaluation in the multivariate case is not an elementary extension of the univariate case, unless the evaluation points have good properties. In particular, when the evaluation points belong in a set of the form $S_1 \times \cdots \times S_n$, it is advantageous to perform multipoint evaluation at each coordinate one by one. The advantage comes from the fact that the number of different values in each coordinate is $|S_i|$, whereas the evaluation points are in total $\prod_{i=1}^n |S_i|$.

Proposition 3.1. *Let $f \in \mathbb{C}[\mathbf{X}]$ be a polynomial of degree d in each variable with absolute value of coefficients at most 2^τ and $S = S_1 \times \cdots \times S_n \subset \mathbb{C}^n$ be a set of complex points with absolute values bounded by 2^Γ, where $\Gamma \geq 1$, and $|S_i| \leq M$. Then, approximate multipoint evaluation at S up to a precision of 2^{-L} for some integer $L \geq 0$, that is, computing $\tilde{f}$ such that $|\tilde{f} - f(\mathbf{a})| \leq 2^{-L}$ for all $\mathbf{a} \in S$, can be done in*

$$\widetilde{\mathcal{O}}_B\left(\max(M,d)^{n-1}\left\lceil \frac{M}{d} \right\rceil (ndL + n^2 d\tau + n^3\, d^2\, \Gamma)\right)$$

bit-operations. The precision demand on f and the points $\mathbf{a} \in S$ is bounded by $L + \widetilde{\mathcal{O}}\left(n\tau + n^2 d\Gamma\right)$ bits.

Proof. For any $k = 1, \ldots, n-1$, we can write f as a polynomial in the variables $X_{k+1}, \ldots, X_n$ as

$$f(\mathbf{X}) = \sum_{\substack{e_i \in \{0,\ldots,d\},\\ i=k+1,\ldots,n}} f_{e_{k+1},\ldots,e_n}(X_1, \ldots, X_k)\ X_{k+1}^{e_{k+1}} \cdots X_n^{e_n}.$$

Then, $|f_{e_{k+1},\ldots,e_n}(X_1, \ldots, X_k)| \leq d^k \cdot 2^\tau \cdot 2^{kd\Gamma}$. In particular, for $k = n-1$ and $\mathbf{a}_{-n} \in S_1 \times \ldots S_{n-1}$, $f(\mathbf{a}_{-n}, X_n) = \sum_{i=0}^{d} f_i(\mathbf{a}_{-n})\, X_n^i$, with $|f_i(\mathbf{a}_{-n})| \leq 2^{\widetilde{\mathcal{O}}(\tau+(n-1)d\Gamma)}$. Evaluating $f(\mathbf{a}_{-n}, X_n)$ at S_n with precision L can be done in $\widetilde{\mathcal{O}}_B(d(L+\tau+(n-1)d\Gamma))$ with a precision demand on $f_i(\mathbf{a}_{-n})$ and on the points $a_n \in S_n$ in $L + \widetilde{\mathcal{O}}(\tau + (n-1)\, d\,\Gamma)$ bits.

Recursively, for any $k \in \{1, \ldots, n-1\}$ and $\mathbf{a}^{k-1} := (a_1, \ldots, a_{k-1}) \in S_1 \times \cdots \times S_{k-1}$, we need to evaluate the polynomials $f_{e_{k+1},\ldots,e_n}(\mathbf{a}^{k-1}, X_k)$ at S_k with precision $L + (n-k)\tau + d\Gamma \sum_{i=k}^{n-1} i$. Since the polynomial has coefficients with absolute value bounded by $\tau + (k-1)d\Gamma$, the required precision on the coefficients and on the points in S_k is in $L + \widetilde{\mathcal{O}}((n-k+1)\tau + \sum_{i=k-1}^{n-1} id\Gamma)$. For a polynomial $f_{e_{k+1},\ldots,e_n}$ this requires at most $M^{k-1} \cdot \lceil \frac{M}{d} \rceil$ multipoint evaluations of cost $\widetilde{\mathcal{O}}_B(d(L + n\tau + n^2 d\Gamma))$ each one. For a fixed k there are at most $(d+1)^{n-k}$ polynomials $f_{e_{k+1},\ldots,e_n}$ to be evaluated, so this yields a complexity in $\widetilde{\mathcal{O}}_B(d^{n-k} M^{k-1} \lceil \frac{M}{d} \rceil d(L + n\tau + n^2 d\Gamma))$ bit-operation. By summing for all $k = 1, \ldots, n-1$ we obtain a total bit-complexity in

$$\widetilde{\mathcal{O}}_B\left(\max(M,d)^{n-1} \left\lceil \frac{M}{d} \right\rceil nd(L + n\tau + n^2 d\Gamma)\right),$$

to compute all the evaluations with an error bounded by 2^{-L} and a required precision of all the coordinates of the points in S bounded by $L + \widetilde{\mathcal{O}}(n\tau + n^2 d\Gamma)$ bits. □

Notice that in Prop. 3.1, the existence of an oracle providing the necessary approximations is assumed.

3.3 Amortized bounds for polynomials in a multiple field extension

Let $F_1 \in \mathbb{Z}[X_1], \ldots, F_n \in \mathbb{Z}[X_n]$ be univariate polynomials of size (M, Λ) and $F \in \mathbb{Z}[\mathbf{X}, Y]$ of size (d, τ). We consider the ideals

$$\begin{aligned} \mathcal{I} &= \langle F_1, \ldots, F_n \rangle \subset \mathbb{Z}[X_1, \ldots, X_n] \quad \text{and} \\ \mathcal{J} &= \langle F_1, \ldots, F_n, F \rangle \subset \mathbb{Z}[X_1, \ldots, X_n, Y]. \end{aligned} \tag{3.2}$$

For $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, let $F_{\mathbf{x}}(Y) := F(\mathbf{x}, Y)$. We prove aggregate separation bounds for the roots of F in $(\mathbb{Z}[X_1, \ldots, X_n])[Y]$. We follow closely [DDR+22], where they treat the simple

extension case, and we generalize their results to the n-variate field extension. We use Lem. 3.3 and Lem. 3.5, which are generalizations of Prop. 3 and Prop. 14 of [DDR$^+$22] respectively, as building blocks for our proofs. Lem. 3.3 gives upper and lower bounds on the product of the evaluations of n-variate polynomials at all points in $V_{\mathbb{C}}(\mathcal{I})$ and in Lem. 3.5 the evaluation is of a set of $n+1$-variate polynomials at all points in $V_{\mathbb{C}}(\mathcal{J})$.

First, due to the special structure of the ideals $\mathcal{I}$ and $\mathcal{J}$, we have the following result on the multiplicities of the roots of the corresponding varieties. It will be used in the proof of Lem. 3.3.

Lemma 3.2. [IPY21, Prop.3], [ZFX09] *Let $\mathcal{I}$ and $\mathcal{J}$ be the ideals of Eq.* (3.2)*. For any $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ and any $i \in [n]$ it holds that $\mu_{\mathcal{I}}(\mathbf{x}) = \mu_{F_i}(x_i) \cdot \mu_{\mathcal{I}\setminus F_i}(\mathbf{x}_{-i})$. Moreover, for any $(\mathbf{x}, y) \in V_{\mathbb{C}}(\mathcal{J})$, it holds that $\mu_{\mathcal{J}}(\mathbf{x}, y) = \mu_{\mathcal{I}}(\mathbf{x}) \cdot \mu_{F_{\mathbf{x}}}(y)$.*

Lemma 3.3. *Let $\mathcal{I}$ and $\mathcal{J}$ be the ideals of Eq.* (3.2) *and $G_1, \ldots, G_m \in \mathbb{Z}[X_1, \ldots, X_n]$ of sizes (δ, σ).*

(i) Let $A \subseteq V_{\mathbb{C}}(\mathcal{I})$ such that for every $\mathbf{x} \in A$, there exists an index $i(\mathbf{x}) \in [m]$ such that $G_{i(\mathbf{x})}(\mathbf{x}) \neq 0$. Then,

$$\sum_{\mathbf{x} \in A} \mu_{\mathcal{I}}(\mathbf{x}) \log(|G_{i(\mathbf{x})}(\mathbf{x})|) \in \widetilde{\mathcal{O}}\left(M^n(n+\sigma) + n\delta M^{n-1}\Lambda\right) .$$

(ii) If for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ there exists an index $i \in [m]$ with $G_{i(\mathbf{x})}(\mathbf{x}) \neq 0$, then we denote by $i(\mathbf{x})$ the smallest such index. In this case,

$$\sum_{\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x}) \big| \log(|G_{i(\mathbf{x})}(\mathbf{x})|) \big| \in \widetilde{\mathcal{O}}\left(M^n(n+\sigma) + n\delta M^{n-1}\Lambda\right) .$$

Proof. (i) For any $\mathbf{x} = (x_1, \ldots, x_n) \in A$,

$$|G_{i(\mathbf{x})}(\mathbf{x})| \leq \binom{\delta+n}{n} 2^{\sigma} \prod_{j=1}^{n} \max\{1, |x_j|\}^{\delta},$$

since the number of monomials in $\mathbb{Z}[X_1, \ldots, X_n]$ of degree less than or equal to δ is $\binom{\delta+n}{n}$, the absolute value of every coefficient is $\leq 2^{\sigma}$ and every x_j is of degree at most δ. Therefore,

$$\prod_{\mathbf{x} \in A} |G_{i(\mathbf{x})}(\mathbf{x})|^{\mu_{\mathcal{I}}(\mathbf{x})} \leq \prod_{\mathbf{x} \in A} \left(\binom{\delta+n}{n} 2^{\sigma} \prod_{j=1}^{n} \max\{1, |x_j|\}^{\delta} \right)^{\mu_{\mathcal{I}}(\mathbf{x})} . \tag{3.3}$$

Since $\sum_{\mathbf{x}\in A}\mu_{\mathcal{I}}(\mathbf{x}) \leq M^n$ and $\binom{\delta+n}{n} \in \mathcal{O}((\delta+n)^n)$, we have:

$$\prod_{\mathbf{x}\in A}\left(\binom{\delta+n}{n}2^{\sigma}\right)^{\mu_{\mathcal{I}}(\mathbf{x})} \in 2^{\widetilde{\mathcal{O}}(M^n(n+\sigma))}. \tag{3.4}$$

For $j \in [n]$ it holds that

$$\begin{aligned}\prod_{\mathbf{x}\in A}\max\{1,|x_j|\}^{\delta\,\mu_{\mathcal{I}}(\mathbf{x})} &= \prod_{\mathbf{x}\in A}\left(\max\{1,|x_j|\}^{\mu_{F_j}(x_j)}\right)^{\delta\,\mu_{\mathcal{I}\setminus F_j}(\mathbf{x}_{-j})}\\ &= \prod_{x_j|\mathbf{x}\in A}\left(\max\{1,|x_j|\}^{\mu_{F_j}(x_j)}\right)^{\delta\,\sum_{\mathbf{x}_{-j}|\mathbf{x}\in A}\mu_{\mathcal{I}\setminus F_j}(\mathbf{x}_{-j})}\\ &\leq \prod_{x_j|\mathbf{x}\in A}\left(\max\{1,|x_j|\}^{\mu_{F_j}(x_j)}\right)^{\delta\,M^{n-1}} \leq \mathcal{M}(F_j)^{\delta\,M^{n-1}},\end{aligned}$$

where the first equality follows from Lem. 3.2 and the first inequality from the fact that $\sum_{\mathbf{x}_{-j}|\mathbf{x}\in A}\mu_{\mathcal{I}\setminus F_j}(\mathbf{x}_{-j}) \leq M^{n-1}$. Note that the last inequality is true since the coefficients of F_j are in $\mathbb{Z}$ and so the absolute value of the leading coefficient of F_j is greater or equal to 1. We have that $\mathcal{M}(F_j) \in 2^{\mathcal{O}(\Lambda+\log M)}$, following Eq. (2.5). Therefore,

$$\prod_{\mathbf{x}\in A}\max\{1,|x_j|\}^{\delta\mu_{\mathcal{I}}(\mathbf{x})} \in 2^{\widetilde{\mathcal{O}}(\delta M^{n-1}\Lambda)}. \tag{3.5}$$

From the equations (3.3), (3.4) and (3.5), we conclude.

(ii) Let $A = \{\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I}) : \ |G_{i(\mathbf{x})}(\mathbf{x})| \geq 1\}$. Then, we can write:

$$\begin{aligned}\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\mu_{\mathcal{I}}(\mathbf{x})\big|\log(|G_{i(\mathbf{x})}(\mathbf{x})|)\big| &= 2\sum_{\mathbf{x}\in A}\mu_{\mathcal{I}}(\mathbf{x})\log(|G_{i(\mathbf{x})}(\mathbf{x})|)-\\ &\quad-\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\mu_{\mathcal{I}}(\mathbf{x})\log(|G_{i(\mathbf{x})}(\mathbf{x})|).\end{aligned} \tag{3.6}$$

Using (i) of this lemma, we obtain an upper bound for the first term of the previous sum. Thus, we only need to compute a lower bound for $\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\mu_{\mathcal{I}}(\mathbf{x})\log(|G_{i(\mathbf{x})}(\mathbf{x})|)$. Let $G(\mathbf{X},U) = G_1(\mathbf{X})+G_2(\mathbf{X})U+\cdots+G_m(\mathbf{X})U^{m-1}$. We consider $\mathtt{res}(F_1,\ldots,F_n,G)$, the multivariate resultant where we eliminate $\mathbf{X}$. Using the Poisson formula [CLO05, Thm. 3.4] we can write

$$\begin{aligned}\mathtt{res}_Y(F_1,\ldots,F_n,G) = \mathtt{res}(\mathrm{lc}(F_1)X_1^{\deg(F_1)},\ldots,\mathrm{lc}(F_n)X_n^{\deg(F_n)})^{\mathcal{O}(\delta)}\cdot\\ \cdot\prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}G(\mathbf{x},U)^{\mu_{\mathcal{I}}(\mathbf{x})} =\end{aligned}$$

$$= \left(\mathtt{res}(X_1^{\deg(F_1)}, \dots, X_n^{\deg(F_n)}) \prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j} \deg(F_k)} \right)^{\mathcal{O}(\delta)} \cdot$$

$$\cdot \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} G(\mathbf{x}, U)^{\mu_{\mathcal{I}}(\mathbf{x})} =$$

$$= \left(\prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j} \deg(F_k)} \right)^{\mathcal{O}(\delta)} \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} G(\mathbf{x}, U)^{\mu_{\mathcal{I}}(\mathbf{x})},$$

which is a polynomial in $\mathbb{Z}[U]$; it is not identically zero, since by the hypothesis, for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, $G(\mathbf{x}, U)$ is not identically zero. The absolute value of the constant term of $\mathtt{res}(F_1, \dots, F_n, G)$ is

$$\left| \prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j} \deg(F_k)} \right|^{\mathcal{O}(\delta)} \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} |G_{i(\mathbf{x})}(\mathbf{x})|^{\mu_{\mathcal{I}}(\mathbf{x})} \geq 1. \tag{3.7}$$

Since $\left| \prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j} \deg(F_k)} \right| \in 2^{\mathcal{O}(n\Lambda M^{n-1})}$, it follows from Eq. (3.7) that

$$\prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} |G_{i(\mathbf{x})}(\mathbf{x})|^{\mu_{\mathcal{I}}(\mathbf{x})} \in 2^{-\mathcal{O}(n\delta\Lambda M^{n-1})}. \tag{3.8}$$

So, by applying part (i) of the lemma and Eq. (3.8) to Eq. (3.6), we conclude.

□

The following corollary, provides an amortized bound on the sum of the logarithms (bitsize) of the Mahler measures of the polynomials $F_{\mathbf{x}}(Y)$, for all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ (counting multiplicities).

Corollary 3.4 (Amortized Mahler measure). *Let $\mathcal{I}$ and $\mathcal{J}$ be the ideals of Eq.* (3.2)*. Then,*

$$\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x}) \log \mathcal{M}(F_{\mathbf{x}}) \in \widetilde{\mathcal{O}}\left(M^n(n+\tau+d) + nM^{n-1}d\Lambda \right).$$

Proof. We write $F(\mathbf{X}, Y) = f_d(\mathbf{X})Y^d + \cdots + f_0(\mathbf{X})$. For any $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, following Eq. (2.5), it holds

$$2^{-d}\|F_{\mathbf{x}}\|_1 \leq \mathcal{M}(F_{\mathbf{x}}) \leq \|F_{\mathbf{x}}\|_2, \tag{3.9}$$

since the degree of any $F_{\mathbf{x}}(Y)$ is $\leq d$. Let

$$|f_{M(\mathbf{x})}(\mathbf{x})| := \max_{j\in\{0,\dots d\}} |f_j(\mathbf{x})|,$$

$$|f_{m(\mathbf{x})}(\mathbf{x})| := \min_{j\in\{0,\dots d\}|f_j(\mathbf{x})\neq 0} |f_j(\mathbf{x})|\,.$$

Now, Eq.(3.9) gives

$$2^{-d}\,|f_{m(\mathbf{x})}(\mathbf{x})| \leq \mathcal{M}(F_{\mathbf{x}}) \leq \sqrt{d+1}\,|f_{M(\mathbf{x})}(\mathbf{x})|\,.$$

If we consider for all $\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})$ (counting multiplicities), then

$$\begin{aligned} 2^{-dM^n}\prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} |f_{m(\mathbf{x})}(\mathbf{x})|^{\mu_{\mathcal{I}}(\mathbf{x})} &\leq \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mathcal{M}(F_{\mathbf{x}})^{\mu_{\mathcal{I}}(\mathbf{x})} \leq \\ &\leq \sqrt{d+1}^{M^n} \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} |f_{M(\mathbf{x})}(\mathbf{x})|^{\mu_{\mathcal{I}}(\mathbf{x})}\,. \end{aligned} \tag{3.10}$$

We can bound the products on each side of the inequality in Eq.(3.10) by Lem. 3.3. This concludes the proof. □

The following lemma is an analog of Lem. 3.3, but in the case where we evaluate over $V_{\mathbb{C}}(\mathcal{J})$.

Lemma 3.5. *Let $\mathcal{I}$ and $\mathcal{J}$ be the ideals of Eq. (3.2) and $G_1,\dots,G_m\in\mathbb{Z}[X_1,\dots,X_n,Y]$ of sizes (δ,σ).*

(i) Let $A\subseteq V_{\mathbb{C}}(\mathcal{J})$ such that for every $(\mathbf{x},y)\in A$, there exists an index $i(\mathbf{x},y)\in[m]$ such that $G_{i(\mathbf{x},y)}(\mathbf{x},y)\neq 0$. Then,

$$\begin{aligned} &\sum_{(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})} \mu_{\mathcal{J}}(\mathbf{x},y)\log|G_{i(\mathbf{x},y)}(\mathbf{x},y)| \\ &\qquad\qquad \in \widetilde{\mathcal{O}}\left(M^n(d(n+\sigma)+\delta(n+\tau+d))+n\delta d M^{n-1}\Lambda\right). \end{aligned}$$

(ii) Supposing that for every $(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})$ there exists an index $i\in[m]$ with $G_i(\mathbf{x},y)\neq 0$, we denote by $i(\mathbf{x},y)$ the smallest such index. Then,

$$\begin{aligned} &\sum_{(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})} \mu(\mathbf{x},y)\big|\log(|G_{i(\mathbf{x},y)}(\mathbf{x},y)|)\big| \\ &\qquad\qquad \in \widetilde{\mathcal{O}}\left(M^n(d(n+\sigma)+\delta(n+\tau+d))+n\delta d M^{n-1}\Lambda\right). \end{aligned}$$

Proof. (i) For any $(\mathbf{x},y)\in A$,

$$|G_{i(\mathbf{x},y)}(\mathbf{x},y)| \leq \binom{\delta+n+1}{n+1} 2^{\sigma}\prod_{i=1}^{n}\max\{1,|x_i|\}^{\delta}\max\{1,|y|\}^{\delta}\,,$$

since the number of monomials in $\mathbb{Z}[X_1,\dots,X_n,Y]$ of degree less than or equal to δ

is $\binom{\delta+n+1}{n+1}$. We have that:

$$\prod_{(\mathbf{x},y)\in A}\left(\binom{\delta+n+1}{n+1}2^{\sigma}\right)^{\mu_{\mathcal{J}}(\mathbf{x},y)} \in 2^{\widetilde{\mathcal{O}}(M^n d(n+\sigma))}, \tag{3.11}$$

since $\sum_{(\mathbf{x},y)\in A}\mu_{\mathcal{J}}(\mathbf{x},y) \leq M^n d$. For $j=1,\ldots,n$:

$$\begin{aligned}
\prod_{(\mathbf{x},y)\in A}\max\{1,|x_j|\}^{\delta\mu_{\mathcal{J}}(\mathbf{x},y)} &\leq \prod_{(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})}\max\{1,|x_j|\}^{\delta\mu_{\mathcal{J}}(\mathbf{x},y)} \\
&= \prod_{(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})}\max\{1,|x_j|\}^{\delta\mu_{F_j}(x_j)\mu_{\mathcal{I}_{-j}}(\mathbf{x}_{-j})\mu_{F_{\mathbf{x}}}(y)} \\
&\leq \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\max\{1,|x_j|\}^{\delta\mu_{F_j}(x_j)\mu_{\mathcal{I}_{-j}}(\mathbf{x}_{-j})d} \\
&= \prod_{x_j\in V_{\mathbb{C}}(F_j)}\max\{1,|x_j|\}^{\delta\mu_{F_j}(x_j)\sum_{\mathbf{x}_{-j}|\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\mu_{\mathcal{I}\setminus F_j}(\mathbf{x}_{-j})d} \\
&\leq \prod_{x_j\in V_{\mathbb{C}}(F_j)}\max\{1,|x_j|\}^{\delta\mu_{F_j}(x_j)M^{n-1}d} \leq \mathcal{M}(F_j)^{\delta M^{n-1}d},
\end{aligned}$$

where the first equality follows from Lem. 3.2 and the third inequality from the fact that $\sum_{\mathbf{x}_{-j}|\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\mu_{\mathcal{I}\setminus F_j}(\mathbf{x}_{-j}) \leq M^{n-1}$. Note that the last inequality is true since the coefficients of F_j are in $\mathbb{Z}$. Since $\mathcal{M}(F_j) \in 2^{\mathcal{O}(\Lambda+\log M)}$, we have that

$$\prod_{(\mathbf{x},y)\in A}\max\{1,|x_j|\}^{\delta\mu_{\mathcal{J}}(\mathbf{x},y)} \in 2^{\widetilde{\mathcal{O}}(\delta M^{n-1}d\Lambda)}. \tag{3.12}$$

Lastly, we have that

$$\begin{aligned}
&\prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}|\mathrm{lc}(F_{\mathbf{x}}(Y)|^{\delta\mu_{\mathcal{I}}(\mathbf{x})}\cdot\prod_{(\mathbf{x},y)\in A}\max\{1,|y|\}^{\delta\mu_{\mathcal{J}}(\mathbf{x},y)} \leq \\
&\leq \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}|\mathrm{lc}(F_{\mathbf{x}}(Y)|^{\delta\mu_{\mathcal{I}}(\mathbf{x})}\left(\prod_{y\in V_{\mathbb{C}}(F_{\mathbf{x}})}\max\{1,|y|\}^{\mu_{F_{\mathbf{x}}}(y)}\right)^{\delta\mu_{\mathcal{I}}(\mathbf{x})} \leq \\
&\leq \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\mathcal{M}(F_{\mathbf{x}})^{\delta\mu_{\mathcal{I}}(\mathbf{x})} \in 2^{\widetilde{\mathcal{O}}(M^n\delta(n+\tau+d)+M^{n-1}\delta nd\Lambda)},
\end{aligned}$$

which follows from Cor.3.4. Also, from Lem. 3.3 we can bound the size of the factor $\prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}|\mathrm{lc}(F_{\mathbf{x}}(Y)|^{\delta\mu_{\mathcal{I}}(\mathbf{x})}$ on the left-hand side of the previous equation, and thus, we have that

$$\prod_{(\mathbf{x},y)\in A}\max\{1,|y|\}^{\delta\mu_{\mathcal{J}}(\mathbf{x},y)} \in 2^{\widetilde{\mathcal{O}}(M^n\delta(n+\tau+d)+M^{n-1}\delta nd\Lambda)}. \tag{3.13}$$

By putting together Eq. (3.11), Eq. (3.12) and Eq. (3.13) we can conclude.

(ii) Let $A = \{(\mathbf{x}, y) | |G_{i(\mathbf{x},y)}(\mathbf{x}, y)| \geq 1\}$. As in the proof of Lem. 3.3, we will first find a lower bound for

$$\sum_{(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})} \mu_{\mathcal{I}}(\mathbf{x})\mu_{F_{\mathbf{x}}}(y)\log(|G_{i(\mathbf{x},y)}(\mathbf{x}, y)|)\,.$$

Let $G(\mathbf{X}, Y, U) := G_1(\mathbf{X}, Y)+G_2(\mathbf{X}, Y)U+\cdots+G_m(\mathbf{X}, Y)U^{m-1}$. Let also $Q(\mathbf{X}, U) := \texttt{res}(G(\mathbf{X}, Y, U), F(\mathbf{X}, Y))$, be the resultant where we eliminate Y. Without loss of generality, we assume that the leading coefficient of $F(X_1, \ldots, X_n, Y)$ when considered as a polynomial in $\mathbb{Z}[X_1, \ldots, X_n][Y]$, is not canceled for any root of $\mathcal{I}$ (in the case where the leading coefficient is cancelled for some roots, F is replaced by a polynomial of smaller degree). So, the resultant is non the zero polynomial.

We consider $\texttt{res}(Q, F_1, \ldots, F_n)$, which is now the resultant where we eliminate $\mathbf{X}$. Using the Poisson formula [CLO05, §3] we can write

$$\begin{aligned}
&\texttt{res}(Q(\mathbf{X}, U), F_1(X_1), \ldots, F_n(X_n)) = \\
&= \left(\prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j}\deg(F_k)}\right)^{\mathcal{O}(d\delta)} \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} Q(\mathbf{x}, U)^{\mu(\mathbf{x})} = \\
&= \left(\prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j}\deg(F_k)}\right)^{\mathcal{O}(d\delta)} \prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} f_d(\mathbf{x})^{\mathcal{O}(\delta)\mu_{\mathcal{I}}(\mathbf{x})}\cdot \\
&\qquad \cdot \prod_{y|(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})} G(\mathbf{x}, y, U)^{\mu_{\mathcal{I}}(\mathbf{x})\mu(y)}
\end{aligned}$$

The absolute value of the constant term of $\texttt{res}(Q(\mathbf{X}, U), F_1(X_1), \ldots, F_n(X_n)) \in \mathbb{Z}[U]$ is:

$$\begin{aligned}
&\left|\prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j}\deg(F_k)}\right|^{\mathcal{O}(d\delta)} \prod_{(\mathbf{x})\in V_{\mathbb{C}}(\mathcal{I})} |f_d(\mathbf{x})|^{\mathcal{O}(\delta)\mu_{\mathcal{I}}(\mathbf{x})}\cdot \\
&\qquad \cdot \prod_{(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})} |G_{i(\mathbf{x},y)}(\mathbf{x}, y)|^{\mu_{\mathcal{I}}(\mathbf{x})\mu_{F_{\mathbf{x}}}(y)} \geq 1.
\end{aligned}$$

We have that $\left|\prod_{j=1}^{n} \mathrm{lc}(F_j)^{\prod_{k\in[n],k\neq j}\deg(F_k)}\right| \in 2^{\mathcal{O}(n\Lambda M^{n-1})}$.

it follows that

$$\prod_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} |G_{i(\mathbf{x})}(\mathbf{x})|^{\mu_{\mathcal{I}}(\mathbf{x})} \in 2^{-\mathcal{O}(n\delta\Lambda M^{n-1})}. \tag{3.14}$$

So, by combining the first part of the lemma and Eq.(3.14), we conclude.

□

Corollary 3.6 (Amortized bound on lGDisc **and** lsep**).** *Let $\mathcal{I}$ and $\mathcal{J}$ be the ideals of Eq.* (3.2)*. Then,*

$$\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x})\,\mathrm{lGDisc}(F_{\mathbf{x}}) \in \widetilde{\mathcal{O}}\left(dM^n(n+\tau+d)+nd^2M^{n-1}\Lambda)\right)$$

and

$$\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x})\,\mathrm{lsep}(F_{\mathbf{x}}) \in \widetilde{\mathcal{O}}\left(dM^n(n+\tau+d)+nd^2M^{n-1}\Lambda)\right)\ .$$

Proof. We can write

$$\begin{aligned}
&\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x})\mathrm{lGDisc}(F_{\mathbf{x}}) = \\
= &\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x}) \sum_{y\in V_{\mathbb{C}}(F_{\mathbf{x}})} \mu_{F_{\mathbf{x}}}(y)|\log(|F_{\mathbf{x}}^{[\mu_{F_{\mathbf{x}}}(y)]}(y)|)| = \\
&= \sum_{(\mathbf{x},y)\in V_{\mathbb{C}}(\mathcal{J})} \mu_{\mathcal{J}}(\mathbf{x},y)|\log(|F_{\mathbf{x}}^{[\mu_{F_{\mathbf{x}}}(y)]}(y)|)|
\end{aligned}$$

and then apply Lem. 3.5 for the family of polynomials $F_{\mathbf{x}}^{[k]}$, for $k=0,\dots,d$. These polynomials are of size $(d,\widetilde{\mathcal{O}}(\tau))$, therefore the first part follows. The second part is an immediate consequence of Prop. 2.35, Cor. 3.4 and the first part of this corollary. □

3.4 Solving in a multiple field extension

In this section, we study the complexity of isolating the roots of the system in Eq. (3.1). We first solve the univariate polynomials of the system and then, for every $\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})$, we will isolate the roots of $F_{\mathbf{x}}$. Following [DDR+22], we employ the univariate root isolation algorithm of Prop. 2.39. The main result of this section is summarized in the theorem that follows.

Theorem 3.7. *(i) If the number of distinct roots of $F_{\mathbf{x}}(Y)$ for every $\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})$ is known, then we compute isolating discs for all the roots and the corresponding multiplicities in*

$$\widetilde{\mathcal{O}}_B\Big(nM^{n+1}d(n+\tau+d)+nM^nd^2(n\Lambda+d^n(n+\tau+d+\Lambda))+n^2d^{n+3}M^{n-1}\Lambda\Big)\,.$$

(ii) If the number of distinct roots is not known, then we compute isolating discs for all the roots, together with the corresponding multiplicities in

$$\widetilde{\mathcal{O}}_B\Bigg(\max(M,d^2)^{n-1} \left\lceil \frac{M}{d^2} \right\rceil ((nM^n+n^2)d^5(\tau+n)+n^2\, d^6\, \Lambda(M^{n-1}+n))+ \\ +nM^n d^{n+2}(n+\tau+d+\Lambda)+n^2 d^{n+3} M^{n-1}\Lambda \Bigg) .$$

Remark 3.8. *When $M = d$ and $\Lambda = \tau$, the bit-complexity bounds of the previous theorem become*

$$\widetilde{\mathcal{O}}_B(nd^{2n+2}(d+n\tau)) ,$$

when the number of distinct roots of $F_{\mathbf{x}}(Y)$ is known for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ (or when F is squarefree) and

$$\widetilde{\mathcal{O}}_B\left(n^2 d^{3n+3}\tau + n^3 d^{2n+4}\tau\right) ,$$

otherwise.

Remark 3.9 (Number of distinct roots, numerically vs formally). *Determining the number of distinct roots of every $F_{\mathbf{x}}$ in Thm. 3.7 dominates the total complexity. When $n = 1$, a formal method involving univariate gcd computations can be used to find this number; the initial system is triangular and it can be efficiently decomposed into regular triangular systems, for whom the number of distinct roots over every $\mathbf{x}$ is constant [DDR+22, LPR17]. However, when $n > 1$ the system is not triangular and decomposing it to a set of regular triangular systems (cf. extended gcd computation [DSM+05]) as for the case $n = 1$, and thus loosing the original shape of the system, would require isolating roots of a triangular system in n variables. The latter problem is substantially more demanding than isolating the roots of n univariate polynomials. For this reason, we will follow a numerical approach to find the number of distinct roots of every $F_{\mathbf{x}}$.*

In the remark that follows, we compute the complexity of isolating the roots of the system of Eq. (3.1) using the algorithm of approximate factorization of Pan [Pan02] with the maximal precision; instead of requiring the number of distinct roots of a univariate polynomial, if we approximate its roots with precision up to the separation bound, then the root approximations that have pairwise distances smaller that the separation bound, will correspond to the same root. So, this is a method that avoids computing the number of distinct roots of the univariate polynomials $F_{\mathbf{x}}$. Nevertheless, it is a theoretical approach which brings about practical limitations in contrast to our adaptive method.

Remark 3.10 (Pan's algorithm with maximal precision). *On isolating the roots of $F_{\mathbf{x}}$ for every $V_{\mathbb{C}}(\mathcal{I})$, instead of employing the algorithm of Prop. 2.39, that requires knowing the number of distinct roots, we can use Pan's algorithm of approximate factorization with precision up to the separation bound of the roots of the initial system of Eq. (3.1). Then, for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, we approximate $F(\mathbf{x}, Y)$ up to $\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x}) \operatorname{lsep}(F_{\mathbf{x}}) \in \widetilde{\mathcal{O}}(M^n d(n+$*

$\tau + d) + nd^2M^{n-1}\Lambda))$ bits. From Prop. 3.1 this is done in

$$\widetilde{\mathcal{O}}_B\left(\max(M,d)^{n-1}\left\lceil\frac{M}{d}\right\rceil(nM^nd^2(n+\tau+d)+n^2d^3M^{n-1}\Lambda+n^2d\tau+n^3\,d^2\tau)\right)$$

since there are d polynomials to evaluate. Then, for all $F_{\mathbf{x}}(Y)$, Pan's algorithm runs in

$$\widetilde{\mathcal{O}}_B\left(M^{2n}d^2(n+\tau+d)+nd^3M^{2n-1}\Lambda\right),$$

and returns isolating intervals for all the roots.

Before presenting the proof of Thm. 3.7, we need some intermediate results. The next lemma, gives upper and lower bounds on the evaluation of a polynomial over an algebraic number. For simplicity, we ignore the logarithmic factors.

Lemma 3.11. *For every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ and a polynomial $b(\mathbf{X}) \in \mathbb{Z}[\mathbf{X}]$ of size (δ,σ), it holds that*

$$2^{-\widetilde{\mathcal{O}}(M^n(\sigma+n)+nM^{n-1}\delta\Lambda)} \leq |b(\mathbf{x})| \leq 2^{\widetilde{\mathcal{O}}(n\delta\Lambda+\sigma)}.$$

Proof. For the upper bound, we have that

$$|b(\mathbf{x})| \leq \binom{\delta+n}{n}2^{\sigma}\prod_{i\in[n]}\max(1,|x_i|)^{\delta} \leq \binom{\delta+n}{n}2^{\sigma}2^{(\Lambda+\log M+1)\delta n},$$

since, for $i \in [n]$, $\max(1,|x_i|) \leq 2^{\Lambda+\log M+1}$ from the Cauchy bound [BPR06, Cor.10.4]. For the lower bound, we follow the technique of u-resultant and consider the system:

$$F_0(\mathbf{X},Y) = F_1(X_1) = \cdots = F_n(X_n) = 0, \tag{3.15}$$

where $F_0(\mathbf{X},Y) = Y - b(\mathbf{X})$ and Y is a new variable. We consider the resultant of the previous system that eliminates $\mathbf{X}$. Then,

$$R(Y) := \mathtt{res}_{\mathbf{X}}(F_0,\ldots,F_n) = \mathrm{lc}(R)\prod_{\mathbf{a}\in V_{\mathbb{C}}(\mathcal{I})}(Y-b(\mathbf{a}))^{\mu_{\mathcal{I}}(\mathbf{a})} \in \mathbb{Z}[Y].$$

We will find an upper bound on the bitsize of R. To this scope, we follow the proof of the DMM bound in [EMT20] (see also the proof of the sparse resultant's height bound in [MST17]). For $i = 0,\ldots,n$, let Q_i be the Newton polytope of F_i. We denote by $\#Q_i$ the number of lattice points in the closed polytope Q_i and by M_i the mixed volume of all these polytopes except from Q_i. The resultant R is a univariate polynomial in Y, with coefficients homogeneous polynomials in the coefficients of the polynomials of the system in Eq. (3.15):

$$R(Y) = \ldots + \rho_k Y^k \mathbf{c}_{0,k}^{M_0-k}\mathbf{c}_{1,k}^{M_1}\cdots\mathbf{c}_{n,k}^{M_n} + \ldots,$$

where $\rho_k \in \mathbb{Z}$, $\mathbf{c}_{i,k}^{M_i}$ is a monomial in the coefficients of F_i with total degree M_i, for $i \in [n]$, and $\mathbf{c}_{0,k}^{M_0-k}$ is a monomial in the coefficients of F_0 of total degree $M_0 - k$. It holds that

$$\begin{aligned}
|\mathbf{c}_{1,k}^{M_1} \cdots \mathbf{c}_{n,k}^{M_n}| &\leq \prod_{i=1}^{n} \|F_i\|_\infty^{M_i} \leq 2^{nM^{n-1}\delta\Lambda}\,, \\
|\mathbf{c}_{0,k}^{M_0-k}| &\leq \|F_0\|_\infty^{M_0-k} \leq 2^{\sigma(M^n-k)}\,, \\
\rho_k &\leq \prod_{i=0}^{n} (\#Q_i)^{M_i} \leq ((\delta+1)^n+1)^{M^n}\,(M+1)^{nM^{n-1}\delta}.
\end{aligned}$$

So, $\|R\|_\infty \leq 2^{nM^{n-1}\delta(\Lambda+\log M+1))+M^n(\sigma+n\log\delta+n+1)}$. Since $R(Y)$ is a polynomial with integer coefficients, from the Cauchy bound, we have that the absolute value of any of its roots is $\geq \|R\|_\infty^{-1}$. □

Lemma 3.12. *For all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, we compute the degree of $F_{\mathbf{x}}(Y)$ in a bit-complexity in*

$$\widetilde{\mathcal{O}}_B\left(\max(M,d)^{n-1}\left\lceil\frac{M}{d}\right\rceil(nd^2(M^n(\tau+n)+n\tau)+n^2d^3\Lambda(M^{n-1}+n))\right).$$

Proof. To determine which is the first non-zero coefficient of $F_{\mathbf{x}}(Y) = \sum_{i=0}^{d} f_i(\mathbf{x})Y^i$, it suffices to approximate its coefficients up to L bits, where $L \in \widetilde{\mathcal{O}}(M^n(\tau+n)+nM^{n-1}d\Lambda)$ (Lem. 3.11). From Prop. 3.1, this can be done, for all $f_i(X)$ and all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, using multipoint evaluation, in

$$\widetilde{\mathcal{O}}_B\left(\max(M,d)^{n-1}\left\lceil\frac{M}{d}\right\rceil(nd^2L+n^2d^2\tau+n^3\,d^3\Lambda)\right) \tag{3.16}$$

bit-operations. This requires approximations of every $\mathbf{x}$ to bit-accuracy at most $\widetilde{\mathcal{O}}(L+nd+$ $+n^2d\Lambda)$, which is done for all $\mathbf{x} \in \mathcal{V}_{\mathbb{C}}(\mathcal{I})$ in $\widetilde{\mathcal{O}}_B(M^3+M^2\Lambda+ML+nMd+n^2dM\Lambda)$ [MSW15, Thm.5]. The total bit-complexity is dominated by the one in Eq. (3.16). We substitute the upper bound for L to conclude. □

Lemma 3.13. *For all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, we compute the number of distinct complex roots of $F_{\mathbf{x}}(Y)$ in a bit-complexity in*

$$\widetilde{\mathcal{O}}_B\left(\max(M,d^2)^{n-1}\left\lceil\frac{M}{d^2}\right\rceil(nd^5(\tau+n)(M^n+n)+n^2d^6\Lambda(M^{n-1}+n))\right).$$

Proof. We define the polynomials $F_\ell(\mathbf{X},Y) := \sum_{i=0}^{\ell} f_i(\mathbf{X})Y^i$, for $\ell = 0,\ldots,d$, which are truncated versions of $F(\mathbf{X},Y)$. Since the degree of $F_{\mathbf{x}}$ is not the same for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, we have to repeat the following steps for every $\ell = 0,\ldots,d$:

(1) We compute the principal subresultant coefficients of $F_\ell(\mathbf{X},Y)$, $\frac{\partial F_\ell}{\partial Y}(\mathbf{X},Y)$ with respect to Y. The j-th principal subresultant coefficient is a polynomial in $\mathbb{Z}[\mathbf{X}]$, denoted by $\mathtt{sres}_j(\mathbf{X})$, of total degree $\mathcal{O}(\ell(\ell-j))$ and bitsize $\widetilde{\mathcal{O}}((\tau+n)(\ell-j))$

[BPR06, Prop.8.72]. The computation of all principal subresultant coefficients is done in $\widetilde{\mathcal{O}}_B\left(\ell^{2n+2}\tau\right)$ [LPR17, Lem.4].

(2) The index of the first non-zero $\mathtt{sres}_j(\mathbf{x})$ gives the degree of the $\gcd(F_\ell(\mathbf{x},Y), \frac{\partial F_\ell}{\partial Y}(\mathbf{x},Y))$. So, for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, we approximate $\mathtt{sres}_j(\mathbf{x})$, for $j = 0,\ldots,\ell$, up to L bits, with $L \in \widetilde{\mathcal{O}}(M^n\ell(\tau+n) + nM^{n-1}\ell^2\Lambda)$ (Lem. 3.11), so as to determine if it zero or not. From Prop. 3.1, this can be done, for all $j \in \{0,\ldots,\ell\}$ and all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, using multi-point evaluation, in

$$\widetilde{\mathcal{O}}_B\left(\max(M,\ell^2)^{n-1}\left\lceil\frac{M}{\ell^2}\right\rceil(n\ell^4(\tau+n)(M^n+n) + n^2\ell^5\Lambda(M^{n-1}+n))\right).$$

Repeating the previous steps for all $\ell \in \{0,\ldots,d\}$, yields a total bit-complexity in

$$\widetilde{\mathcal{O}}_B\left(\max(M,d^2)^{n-1}\left\lceil\frac{M}{d^2}\right\rceil(nd^5(\tau+n)(M^n+n) + n^2d^6\Lambda(M^{n-1}+n))\right). \tag{3.17}$$

As we can see in the proof of Prop. 3.1, to compute these evaluations, we need to approximate each $\mathbf{x}$ to bit accuracy at most

$$\widetilde{\mathcal{O}}\left(M^n d(\tau+n) + n^2d^2M\Lambda + nd(\tau+n)\right).$$

This costs for all $\mathbf{x} \in \mathcal{V}_{\mathbb{C}}(\mathcal{I})$, $\widetilde{\mathcal{O}}_B(n\,M\,(M^n d(\tau+n) + n^2d^2M\Lambda + nd(\tau+n)))$ [MSW15, Thm.5], which does not overcome the bit-complexity of Eq. (3.17). □

Lemma 3.14. *For every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ let $\rho_{\mathbf{x}}$ be a positive integer. We compute $\rho_{\mathbf{x}}$-approximations of $F_{\mathbf{x}}(Y)$ for all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ in*

$$\widetilde{\mathcal{O}}_B\left(n\left(M^3 + M^2\Lambda + M\max_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\rho_{\mathbf{x}}\right) + d^{n+1}\left(M^n(\tau+d\Lambda) + \sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})}\rho_{\mathbf{x}}\right)\right).$$

Proof. For a $\rho_{\mathbf{x}}$-approximation of $F_{\mathbf{x}}(Y)$, it suffices to consider an $L_{\mathbf{x}}$-approximation of $\mathbf{x}$, where $L_{\mathbf{x}} \in \widetilde{\mathcal{O}}(\rho_{\mathbf{x}} + \tau + d\Lambda)$. This follows from [BS17, Lem.1], since the coefficients of $F(\mathbf{x},Y)$ are polynomials in $\mathbb{Z}[X_1,\ldots,X_n]$ of size (d,τ), and $\max(1,\|\mathbf{x}\|_\infty) \in 2^{\mathcal{O}(\Lambda)}$. To get the desired approximation of each $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, we compute isolating discs of the roots of each F_i, for $i = 1,\ldots,n$, of size less than $2^{-L_{\max}}$, where $L_{\max} = \max_{x\in V_{\mathbb{C}}(\mathcal{I})} L_{\mathbf{x}}$. This costs [MSW15, Thm.5]

$$\widetilde{\mathcal{O}}_B\left(n(M^3 + M^2\Lambda + ML_{\max})\right). \tag{3.18}$$

For a given $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$, to compute the $\rho_{\mathbf{x}}$-approximation of $F(\mathbf{x},Y)$, we evaluate its coefficients at the $L_{\mathbf{x}}$-approximation of $\mathbf{x}$. This costs $\widetilde{\mathcal{O}}_B(nd^{n+1}(\tau+L_{\mathbf{x}}))$: When we regard $F(\mathbf{X},Y)$ as a polynomial in Y, it has at most $d+1$ coefficients which are polynomials in $\mathbb{Z}[\mathbf{X}]$ of size (d,τ). Using [BLPR15, Lem.6], we evaluate each one of them in $\widetilde{\mathcal{O}}_B(d^n(\tau+L_{\mathbf{x}}))$, and then we multiply the latter bound by d. To obtain the final cost, we sum the latter

bound for all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ and add the cost in Eq. (3.18) and use the fact that $L_{\max} \in \widetilde{\mathcal{O}}(\max_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \rho_{\mathbf{x}} + \tau + d\Lambda)$. □

Now, by putting everything together, we can prove our main theorem.

Proof of Thm. 3.7. (i) For any $\mathbf{x} = (x_1, \ldots, x_n) \in V_{\mathbb{C}}(\mathcal{I})$, we compute a $\tau_{\mathbf{x}}$ such that $2^{-\tau_{\mathbf{x}}-2} \leq \mathrm{lc}(F_{\mathbf{x}}) \leq 2^{-\tau_{\mathbf{x}}}$. Then, the polynomial $2^{-\tau_{\mathbf{x}}} F_{\mathbf{x}}(Y)$ satisfies the conditions of Prop. 2.39, which we then use to isolate its roots (it has the same roots as $F_{\mathbf{x}}(Y)$). Repeating the arguments as in the proof of [DDR+22, Prop. 19] we have that $\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \tau_{\mathbf{x}} \in \mathcal{O}(\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \rho_{\mathbf{x}})$ and that its computation does not affect the total complexity.

Let $\tilde{F}_{\mathbf{x}}(Y) := 2^{\tau_{\mathbf{x}}} F_{\mathbf{x}}(Y)$. From Prop. 2.39, we can solve every $\tilde{F}_{\mathbf{x}}(Y)$ in

$$\widetilde{\mathcal{O}}_B(d(d^2 + d \log \mathcal{M}(\tilde{F}_{\mathbf{x}}) + \mathrm{lGDisc}(\tilde{F}_{\mathbf{x}})). \tag{3.19}$$

For that, we require an approximation of the coefficients of $\tilde{F}_{\mathbf{x}}(Y)$ to an absolute precision bounded by

$$\rho_{\mathbf{x}} \in \widetilde{\mathcal{O}}(d \log \mathcal{M}(\tilde{F}_{\mathbf{x}}) + \mathrm{lsep}(\tilde{F}_{\mathbf{x}}) + \mathrm{lGDisc}(\tilde{F}_{\mathbf{x}})).$$

Using Lem. 3.14, this is done for all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ in

$$\begin{aligned} \widetilde{\mathcal{O}}_B\big(n(M^3 + M\Lambda(M+d) + M \max_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \rho_{\mathbf{x}}) + \\ + nd^{n+1}M^n(\tau + d\Lambda) + nd^{n+1} \sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \rho_{\mathbf{x}}\big)\,. \end{aligned} \tag{3.20}$$

From Cor. 3.4 and Cor. 3.6, we have that

$$\sum_{\mathbf{x}\in V_{\mathbb{C}}(\mathcal{I})} \mu_{\mathcal{I}}(\mathbf{x})\rho_{\mathbf{x}} \in \widetilde{\mathcal{O}}(M^n d(n+\tau+d) + nd^2 M^{n-1}\Lambda))$$

Therefore, Eq. (3.20) becomes

$$\begin{aligned} \widetilde{\mathcal{O}}_B\big(n(M^3 + M^2\Lambda) + nM^{n+1}d(n+\tau+d) \\ + nM^n d^2(n\Lambda + d^n(n+\tau+d+\Lambda)) + n^2 d^{n+3} M^{n-1}\Lambda\big)\,. \end{aligned} \tag{3.21}$$

By summing Eq. (3.19) for all $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ yields

$$\widetilde{\mathcal{O}}_B\big(M^n d^2(n+\tau+d) + nd^3 M^{n-1}\Lambda\big)\,. \tag{3.22}$$

We add the bounds in Eq. (3.21) and Eq. (3.22) to conclude (the bound in Eq. (3.21) dominates).

(ii) When the number of distict roots of $F_{\mathbf{x}}(Y)$ for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ is not already known, then we have to compute it using Lem. 3.13. We add the cost of this computation to the bound of part (i) of the theorem.

□

3.5 Computing with a unique field extension

In this section, we follow an alternative approach to isolate the roots of the system in Eq. (3.1). We compute a Rational Univariate Representation (RUR) (cf. Sec. 2.5.1) of $\mathcal{I}$, that essentially is defined by a univariate polynomial with coefficients in $\mathbb{Q}[T]$ such that there exists a bijection between its roots and the roots of $\mathcal{I}$ that preserves the multiplicities and the real roots. Then, we use this univariate representation of the roots of the system

$$F_1(X_1) = \cdots = F_n(X_n) = 0\,,$$

to transform the the system of Eq. (3.1) to a bivariate triangular system. In this way, we reduce the original problem to the one of isolating the roots of a univariate polynomial with coefficients in a simple extension field.

In particular, to accelerate the RUR computation, we will find a RUR of the radical ideal $\sqrt{\mathcal{I}}$ and not of $\mathcal{I}$. Note that for a general ideal $I \subset \mathbb{Z}[\mathbf{X}]$, computing $\sqrt{I}$ is not easy, even if we already know a Gröbner basis, and the multiplicities of the roots are not preserved. However, in our case, it is not hard to compute $\sqrt{\mathcal{I}}$ (due to the structure of $\mathcal{I}$) and we can obtain the multiplicities of the roots at a reasonable cost (see Rem. 3.15).

In the sequel, we denote $\sqrt{\mathcal{I}}$ by $\mathcal{I}^*$ and the quotient algebra $\frac{\mathbb{Q}[\mathbf{X}]}{\mathcal{I}^*}$ associated to $\mathcal{I}^*$ by $\mathcal{A}$. In Sec. 2.5.1, it was shown that the computation of the RUR for an ideal can be broken down into two parts: finding a separating linear form (SLF) for the ideal and computing the RUR candidate linked to that linear form. In this section, we present a Las Vegas algorithm for computing the RUR of $\mathcal{I}^*$ using modular arithmetic. Firstly, in Sec. 3.5.1, we determine the arithmetic complexity of computing an RUR candidate and we also provide the bitsize of the RUR polynomials. Secondly, in Sec. 3.5.2, we revisit the principles of reducing the RUR computation modulo a prime number, which serves as the basis for the Las Vegas algorithms in the following two sections; Sec. 3.5.3 for the SLF computation and Sec. 3.5.4 for the RUR computation. Finally, in Sec. 3.5.5, we utilize the RUR of $\mathcal{I}^*$ to solve the original system.

3.5.1 Operations in the quotient algebra $\mathcal{A}$

In Sec. 2.5.1, we described how to compute an RUR candidate for general ideals in $\mathbb{Q}[\mathbf{X}]$. Given a linear form $t = \mathbf{a} \cdot \mathbf{X}$, computing an RUR candidate for an ideal requires

$$\mathrm{Tr}(t^i) \quad \text{and} \quad \mathrm{Tr}(X_j t^i) \quad \text{for} \quad j = 1, \ldots, n \quad \text{and} \quad i = 0, \ldots, D\,,$$

where $\mathrm{Tr}(t^i)$ and $\mathrm{Tr}(X_j t^i)$ are the traces of the multiplication matrix M_{t^i} and $M_{X_j t^i}$ respectively and D is the dimension of the associated quotient algebra (Lem. 2.48). Here, we compute the traces when the considered ideal is the radical ideal $\mathcal{I}^*$.

Remark 3.15 (Radical Ideal). *Since $\mathcal{I}$ is generated by univariate polynomials only, we can easily compute its radical $\mathcal{I}^*$, which is not the case in for a general zero-dimensional ideal in $\mathbb{Q}[\mathbf{X}]$ (cf. [GPB+08, § 4.5]). In particular, we have that $\{F_1(X_1), ..., F_n(X_n)\}$ is a Gröbner basis of $\mathcal{I}$ for any monomial ordering and the square-free parts of $F_1, \ldots, F_n$ can be computed in $\widetilde{\mathcal{O}}_B(nM^2\Lambda)$ bit-operations. When one takes the square-free parts, multiplicities are no longer preserved. However, in our case the multiplicity of a root $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$ is $\mu_{\mathcal{I}}(\mathbf{x}) = \prod_{i=1}^n \mu_{F_i}(x_i)$, and thus it can be easily recovered from the multiplicities of the roots of the univariate polynomials F_i.*

Let $F_1^*, \ldots, F_n^*$ be the square-free parts of $F_1, \ldots, F_n$. Then,

$$\mathcal{I}^* = \langle F_1^*, \ldots, F_n^* \rangle\,.$$

Therefore, for a linear form $t \in \mathbb{Q}[\mathbf{X}]$, the (monic) characteristic polynomial of the multiplication matrix by t in $\mathcal{A}$ and its minimal polynomial coincide. When t is also separating for $V_{\mathbb{C}}(\mathcal{I}^*)$, then the minimal polynomial is square-free.

A monomial basis of $\mathcal{A}$ is $\mathcal{B} := \{\prod_{i=1}^n X_i^{e_i}, e_i < M\}$ and we will denote it by

$$\mathcal{B} = \{w_1, .., w_D\}\,,$$

where $D \leq M^n$ and $w_1 = 1$. For an element $P \in \mathcal{A}$ we denote by $\overline{P}$ be the representative of the class of P in $\mathcal{A}$ expressed in $\mathcal{B}$, and by $\overline{P}[i]$ its i-th coordinate.

Given a linear form $t = \mathbf{a} \cdot \mathbf{X}$, for the computation of the traces $\mathrm{Tr}(t^i)$ and $\mathrm{Tr}(X_j t^i)$, we use the following remark which essentially suggests the precomputation of the multiplicative tensor

$$\mathcal{T} := \{\overline{w_i w_j}, 1 \leq i, j \leq D\}$$

and the vector of traces

$$\mathrm{Tr}_1 = [\mathrm{Tr}(w_1), \ldots, \mathrm{Tr}(w_D)]\,.$$

For a polynomial $P \in \mathbb{Q}[\mathbf{X}]$, with $\overline{P} = \sum_{i=1}^D a_1 w_i$, the trace of the multiplication matrix

M_P is computed as

$$\mathrm{Tr}(P) = \sum_{i=1}^{D} \overline{Pw_i}[i] = \sum_{j=1}^{D}\sum_{i=1}^{D} a_j \overline{w_j w_i}[i]\,.$$

Remark 3.16 (Linearity of the Trace map). *For any $P \in \mathbb{Q}[\mathbf{X}]$, $\mathrm{Tr}(P) = \vec{P} \cdot Tr_1$, where $\vec{P}$ is the expression of P as a vector in the base $\mathcal{B}$.*

Lemma 3.17 (Multiplication by one variable). *Let $P \in \mathcal{A}$. Then, for any $i = 1, \ldots, n$, computing $\overline{X_i P}$ can be done in $\mathcal{O}(D)$ arithmetic operations in $\mathbb{Q}$.*

Proof. Consider $B_1 = \{X_1^{M-1} \cdot \prod_{i=2}^{n} X_i^{e_i}, e_i < M\} \subset \mathcal{B}$ and $B_2 = \mathcal{B} \setminus B_1$. We can write P as

$$P = \sum_{m_\alpha \in X_1^{-(M-1)} B_1} a_\alpha X_1^{M-1} m_\alpha + \sum_{m_\beta \in B_2} a_\beta m_\beta$$

Then,

$$X_1 P = \sum_{m_\alpha \in X_1^{-(M-1)} B_1} a_\alpha X_1^{M} m_\alpha + \sum_{m_\beta \in B_2} a_\beta X_1 m_\beta.$$

Notice that $X_1 m_\beta \in \mathcal{B}$ so that $Q := \sum_{m_\beta \in B_2} a_\beta X_1 m_\beta$ is irreducible modulo I, i.e., $\overline{Q} = Q$, and thus

$$\overline{X_1 P} = \sum_{m_\alpha \in X_1^{-(M-1)} B_1} a_\alpha \frac{\mathrm{tail}(F_1)}{\mathrm{lc}(F_1)} m_\alpha + Q$$

where $\mathrm{tail}(F_1) = F_1 - \mathrm{lc}(F_1) X_1^M$. Note also that the supports of the polynomials $a_\alpha \frac{\mathrm{tail}(F_1)}{\mathrm{lc}(F_1)} m_\alpha$ are all disjoint so that computing $\sum_{m_\alpha \in X_1^{-(M-1)} B_1} a_\alpha \frac{\mathrm{tail}(F_1)}{\mathrm{lc}(F_1)} m_\alpha$ is a simple concatenation that can be done in $\mathcal{O}(D)$ arithmetic operations in $\mathbb{Q}$. The addition with $\overline{Q}$ can be done in $\mathcal{O}(D)$ arithmetic operations in $\mathbb{Q}$ which proves the lemma. □

Remark 3.18 (Size of the multiplicative tensor). *The size of the multiplicative tensor varies between D and D^2 in the general case (where D is the dimension of the quotient algebra) and plays an important role in the resulting complexity. Here, $\mathcal{T}$ has cardinality $\#\mathcal{T} \in \mathcal{O}((2M)^n) = \mathcal{O}(2^n D)$. This simplifies its computation, which is described in the next lemma.*

Lemma 3.19 (Traces computation). *(i) The multiplicative tensor $\mathcal{T}$ can be computed in $\mathcal{O}(2^n D^2)$ arithmetic operations in $\mathbb{Q}$.*

(ii) Given the multiplicative tensor $\mathcal{T}$, the vector $\mathrm{Tr}_1 = [\mathrm{Tr}(w_1), \ldots, \mathrm{Tr}(w_D))]$ can be computed in $\mathcal{O}(D^2)$ arithmetic operations in $\mathbb{Q}$.

(iii) Given $\mathcal{T}, \mathrm{Tr}_1$ and a linear form $t = \mathbf{a} \cdot \mathbf{X} \in \mathbb{Q}[\mathbf{X}]$, we compute $\mathrm{Tr}(t^i)$ and $\mathrm{Tr}(X_j t^i)$ in $\mathcal{O}(nD^2)$ arithmetic operations in $\mathbb{Q}$.

Proof. (i) As previously seen, $\mathcal{T}$ has less than $2^n D$ elements and each element $w_i w_j$, when $i \cdot j \neq 1$, can be written as $w_i w_j = X_k m$ for some $k \in [n]$ and $m \in \mathcal{T}$. Thus, $\mathcal{T}$ can be computed in $\mathcal{O}(2^n D^2)$ arithmetic operations in $\mathbb{Q}$, using Lem. 3.17.

(ii) We compute $\mathrm{Tr}_1 = [\mathrm{Tr}(w_1), \ldots, \mathrm{Tr}(w_D)]$ in $\mathcal{O}(D^2)$ arithmetic operations, since $\mathrm{Tr}(w_i) = \sum_{j=1}^{D} \overline{w_i w_j}[j]$.

(iii) We compute $1, t, t^{d^n}$ in $\mathcal{O}(nD^2)$ arithmetic operations: Lem. 3.17 implies that we can compute $\overline{tP}$ for a polynomial $P \in \mathcal{A}$ in $\mathcal{O}(nD)$ arithmetic operations. So, we compute $\overline{t^i}$ in $\mathcal{O}(nD^2)$ by setting P as $t, t^2, \ldots, t^{D-1}$. Then, $\mathrm{Tr}(1)$, $\mathrm{Tr}(t)$,..., $\mathrm{Tr}(t^D)$ are computed in $\mathcal{O}(D^2)$ arithmetic operations in $\mathbb{Q}$, since $\mathrm{Tr}(t^i) = \overrightarrow{t^i} \cdot \mathrm{Tr}_1$ (Rem. 3.16). Then similarly, we compute $\overline{X_j t^i}$ in $\mathcal{O}(nD^2)$ arithmetic operations in $\mathbb{Q}$ using Lem. 3.17, since $\overline{t^i}$ is known. The traces $\mathrm{Tr}(X_j t^i)$ are computed for all i, j in $\mathcal{O}(nD^2)$.

□

Given a linear form $t = \mathbf{a} \cdot \mathbf{X}$ and the corresponding traces $\mathrm{Tr}(t^i)$ and $\mathrm{Tr}(X_j t^i)$, in Lem. 3.19 we showed that the number of arithmetic operations needed to compute the RUR candidate for a general ideal is $\mathcal{O}(nD^2)$ (using Alg. 1). Combining this with the previous result, we obtain a bound on the total number of arithmetic operations.

Corollary 3.20. *Given a linear form $t = \mathbf{a} \cdot \mathbf{X} \in \mathbb{Q}[\mathbf{X}]$ we can compute a RUR candidate of $\mathcal{I}^*$ in $\mathcal{O}((n + 2^n)D^2)$ arithmetic operations in $\mathbb{Q}$.*

Remark 3.21. *It should be noted that for a general ideal in $\mathbb{Q}[\mathbf{X}]$, unlike the previous corollary, the arithmetic complexity of computing a RUR candidate given a linear form is in $\mathcal{O}(D^3 + nD^2)$ [Rou99, Prop. 4.1], where D is the dimension of the associated quotient algebra, or $\mathcal{O}(\#\mathcal{T} D^{3/2} + nD^2)$ using a Baby-Step/Giant-Step algorithm [Rou07, §3].*

Bitsizes of the RUR polynomials

Now, we provide an upper bound on the degree and the bitsize of the polynomials of the RUR. For a linear form $t = \mathbf{a} \cdot \mathbf{X} = X_1 + a_2 X_2 + \cdots + a_n X_n$, let $\tau_{\mathbf{a}}$ be the bitsize of $\mathbf{a}$. According to Brand and Sagraloff [BS16, Thm. 6], one can take a separating linear form t with $\tau_{\mathbf{a}} = \mathcal{O}(n \log M)$. In particular, we have the following:

Lemma 3.22. [BS16, Thm. 6] *There exists a separating linear form $t = \mathbf{a} \cdot \mathbf{X}$ for $V_{\mathbb{C}}(\mathcal{I}^*)$ with non-negative integer coefficients bounded by $M^{4n \log n}$.*

Then, for the sizes of the RUR polynomials, by the DMM bound [EMT20] and as it was also demonstrated in Lem. 3.11, we have the following:

Proposition 3.23. [MST17, Thm. 2.2] *Given a separating linear form $t = \mathbf{a} \cdot \mathbf{X}$, such that the bitsize of $\mathbf{a} = (1, a_2, \ldots, a_n)$ is in $\mathcal{O}(n \log n \log M)$, then the polynomials*

$$f_{\mathcal{I}^*,\mathbf{a}}, f_{\mathcal{I}^*,\mathbf{a},1}, f_{\mathcal{I}^*,\mathbf{a},X_1}, \ldots, f_{\mathcal{I}^*,\mathbf{a},X_n}$$

have degrees at most M^n and bitsize in $\widetilde{\mathcal{O}}(nM^n + nM^{n-1}\Lambda)$.

3.5.2 Lucky primes for the RUR computation

For complexity purposes, we will not compute the RUR over $\mathbb{Q}$ directly, but we will use a (multi-)modular approach instead. For a prime number p we denote by $\varphi_p : \mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ the reduction modulo p morphism and by I_p the reduction modulo p of an ideal $I \subset \mathbb{Z}[\mathbf{X}]$. Let also $\mathcal{A}_p(I_p) := (\mathbb{Z}/p\mathbb{Z}[\mathbf{X}])/I_p$. For the rest, we ue again the notation introduced in pg. 47.

The idea of the modular approach, is to find suitable primes, to compute the RUR modulo these primes and then to lift the result to $\mathbb{Q}$. This is to avoid intermediate coefficient swell that occurs in the computation of the traces. The total bit-complexity will be then derived by using bounds on the size of the RUR coefficients and of the prime numbers involved.

However, not all prime numbers are appropriate for this. We call a prime number *lucky* for a computation, if the algorithm computations commute with the morphism φ_p.

Definition 3.24 ($\mathcal{G}$-compatible prime). [Rou99, Def. 6.1] *Let $I \subset \mathbb{Q}[\mathbf{X}]$ be a zero-dimensional ideal and $\mathcal{G}$ a Gröbner basis. A prime number p is said to be $\mathcal{G}$-compatible if it greater than the dimension of $\mathbb{Q}[\mathbf{X}]/I$ and it does not divide any of the leading coefficients of the polynomials in $\mathcal{G}$.*

Since $F_1^*, \ldots, F_n^*$ form already a Gröbner basis of $\mathcal{I}^*$, we have the following immediate result:

Corollary 3.25. *For the Gröbner basis $\mathcal{G} = \{F_1^*, \ldots, F_n^*\}$ of $\mathcal{I}^*$, a prime number $p \in \mathbb{Z}$ is $\mathcal{G}$-compatible if it is greater than $dim(\mathcal{A}) \leq M^n$ and if it does not divide any of the leading coefficients of $F_1^*, \ldots, F_n^*$.*

According to the following theorem, a $\mathcal{G}$-compatible prime number is lucky for the computation of the multiplication matrix M_P by any polynomial P in $\mathbb{Z}/p\mathbb{Z}[\mathbf{X}]$, for the computation of the trace $\mathrm{Tr}(P)$ of M_P and of its characteristic polynomial E_P.

Theorem 3.26. [Rou99, Cor. 6.1] *Let $\mathcal{G}$ be a Gröbner basis of a zero-dimensional ideal $I \subset \mathbb{Z}[\mathbf{X}]$ and p a $\mathcal{G}$-compatible prime number. Then, for any polynomial $P \in \mathbb{Z}[\mathbf{X}]$, we have that*

1. $\phi_p(M_P) = M_{\phi_p(P)}^{\mathcal{A}_p(I_p)}$,
2. $\phi_p(\mathrm{Tr}(P)) = \mathrm{Tr}^{\mathcal{A}_p(I_p)}(\phi_p(P))$,

3. $\phi_p(E_P) = E_{\phi_p(P)}^{\mathcal{A}_p(I_p)}$.

In order to find a separating linear form for $\mathcal{I}^*$ using modular computations, we will use the following result:

Lemma 3.27. [Rou99, Prop. 6.1] *Let $\mathcal{G}$ be a Gröbner basis of a zero-dimensional ideal $I \subset \mathbb{Z}[X_1, \ldots, X_n]$ and p a $\mathcal{G}$-compatible prime number. If p is greater that* $\dim(\mathbb{Q}[\mathbf{X}]/I)$ *and $\#V_{\mathbb{C}}(I) = \#V_{\mathbb{C}}(\phi_p(I))$, then every polynomial $t \in \mathbb{Z}/p\mathbb{Z}[\mathbf{X}]$ that separates $V_{\mathbb{C}}(\phi_p(I))$, separates also $V_{\mathbb{C}}(I)$.*

In the corollary that follows, we adapt the previous lemma in our setting, to define the lucky primes for the separating linear form computation.

Corollary 3.28. *A prime number $p \in \mathbb{Z}$ is lucky for the computation of a separating linear form for $V_{\mathbb{C}}(\mathcal{I}^*)$ if it is greater that $dim(\mathcal{A}) \leq M^n$, if it does not divide any of the leading coefficients of $F_1^*, \ldots, F_n^*$ and if $\#V_{\mathbb{C}}(\mathcal{I}_p^*) = \#V_{\mathbb{C}}(\mathcal{I}^*)$. Then, any separating linear form for $V_{\mathbb{C}}(\mathcal{I}_p^*)$ is also a separating linear form for $V_{\mathbb{C}}(\mathcal{I}^*)$.*

In the next lemma we compute an upper bound on the number of unlucky primes for the separating linear form computation for $V_{\mathbb{C}}(\mathcal{I}^*)$, by following [BLPR15, Prop. 13].

Lemma 3.29 (Unlucky primes). *The number of unlucky primes for the computation of a separating linear form for $V_{\mathbb{C}}(\mathcal{I}^*)$ is in $\widetilde{\mathcal{O}}(nM^{2n} + M^{2n-1}\Lambda + n\Lambda)$.*

Proof. A prime number p is unlucky if it smaller that M^n or if it divides $\mathrm{lc}(F_1^*) \cdots \mathrm{lc}(F_n^*)$ or if $\#V_{\mathbb{C}}(\mathcal{I}^*) \neq \#V_{\mathbb{C}}(\mathcal{I}_p^*)$.

There are $\mathcal{O}(\frac{M^n}{n \ln M})$ prime numbers smaller than M^n. Let $\Pi = \mathrm{lc}(F_1^*) \cdots \mathrm{lc}(F_n^*)$. The bitsize of Π at most $n\Lambda$ and the number of prime divisors of Π is bounded by its bitsize. Let $t = \mathbf{a} \cdot \mathbf{X}$ a linear form that separates $V_{\mathbb{C}}(\mathcal{I}^*)$. We have that $\#V_{\mathbb{C}}(\mathcal{I}^*) \neq \#V_{\mathbb{C}}(\mathcal{I}_p^*)$ if and only if the degree of $\gcd(E_{\mathbf{a}}, E'_{\mathbf{a}})$ changes after taking the reduction modulo p. Let (d', τ') the size of the polynomials. Then the number of such primes is $\widetilde{\mathcal{O}}(d'\tau')$ (Prop. 2.50). Thus, here it is $\widetilde{\mathcal{O}}(nM^{2n} + M^{2n-1}\Lambda)$ (Prop. 3.23). □

3.5.3 Separating Linear Form: Las-Vegas algorithm

We present a Las-Vegas algorithm to compute a separating linear form for $V_{\mathbb{C}}(\mathcal{I}^*)$. The algorithm chooses iteratively a random candidate separating linear form and a random candidate lucky prime p. At every iteration, it computes the reduction modulo p of $\mathcal{I}^*$ and then the corresponding characteristic polynomial of the multiplication by the linear form in $\mathcal{A}_p(\mathcal{I}_p^*)$. The iterations stop when p is $\mathcal{G}$-compatible and the characteristic polynomial is square-free. Since $\mathcal{I}^*$ is radical, if the computed characteristic polynomial is square-free, then the chosen prime is lucky for the computation of an SLF and the chosen linear form is separating for $V_{\mathbb{C}}(\mathcal{I}_p^*)$, and thus for $V_{\mathbb{C}}(\mathcal{I}^*)$ too. The algorithm is described in pseudocode

in Alg. 2. It should be noted that, as in [BLM+15, Alg. 5'], we select the candidate lucky prime from increasingly larger sets. This is in order to avoid computing an explicit upper bound on the number of unlucky primes, which would be necessary to find an interval that contains a number of primes that is a constant multiple of the number of unlucky primes. The correctness of the algorithm is proven in detail in the lemma that follows.

Algorithm 2: `Separating linear form - Las Vegas`

Input: The radical ideal $\mathcal{I}^*$

Output: Separating linear form $t = \mathbf{a} \cdot \mathbf{X}$ for $\mathcal{I}^*$

1 $\Pi = \mathrm{lc}(F_1^*) \cdots \mathrm{lc}(F_n^*)$
2 $B = M^{2n}$
3 **repeat**
4 $\quad B = 2B$
5 $\quad$ Choose $(a_2, \ldots, a_n) \in \mathbb{Z}^{n-1}$ uniformly at random from the set $\{1, \ldots, M^{2n}\}^{n-1}$
6 $\quad$ Set $\mathbf{a} = (1, a_2, \ldots, a_n)$ and $t = \mathbf{a} \cdot \mathbf{X}$
7 $\quad$ Choose a prime p uniformly at random from the interval (M^{2n}, B)
8 $\quad$ Compute $\mathcal{I}_p^*$, the reduction modulo p of $\mathcal{I}^*$
9 $\quad$ Compute $\mathrm{Tr}^{\mathcal{A}_p(\mathcal{I}_p^*)}(t^i), \mathrm{Tr}^{\mathcal{A}_p(\mathcal{I}_p^*)}(X_j t^i)$ for $i \in [D], j \in [n]$
10 $\quad$ Solve the system $(D-i)b_i = \sum_{j=0}^{i-1} b_{i-j} \mathrm{Tr}^{\mathcal{A}_p(\mathcal{I}_p^*)}(t^j), \quad i = 0, \ldots, D$
11 $\quad$ Set $f_{\mathcal{I}_p^*, \mathbf{a}}(T) = \sum_{i=0}^{D} b_i T^i$
12 **until** $p \nmid \Pi$ *and* $f_{\mathcal{I}_p^*, \mathbf{a}}(T)$ *is squarefree*;
13 RETURN $t = \mathbf{a} \cdot \mathbf{X}$

Lemma 3.30 (Correctness). *Algorithm 2 computes a separating linear form for $\mathcal{I}^*$ of bitsize in $\mathcal{O}(n \log M)$.*

Proof. First, we show that if the loop in the lines 3-12 terminates, the exit condition guarantees that the prime p is lucky for the SLF computation; it is greater that M^n by definition, it does not divide $\mathrm{lc}(F_1^*), \ldots, \mathrm{lc}(F_n^*)$ and the fact that $f_{\mathcal{I}_p^*, \mathbf{a}}(T)$ is squarefree of degree D, means that $\#V_{\mathbb{C}}(\mathcal{I}^*) = \#V_{\mathbb{C}}(\mathcal{I}_p^*)$ (Cor. 3.28). Also, $f_{\mathcal{I}_p^*, \mathbf{a}}(T)$ being squarefree of degree D, means that the linear form $t = \mathbf{a} \cdot \mathbf{X}$ is separating for $V_{\mathbb{C}}(\mathcal{I}_p^*)$. Then, it is also separating for $V_{\mathbb{C}}(\mathcal{I}^*)$.

Moreover, there exist $\mathbf{a}$ and p such that the loop in the lines 3-12 terminates; $\mathbf{a}$ is chosen from a set of $M^{2n(n-1)}$ elements and the directions such that two points in $V_{\mathbb{C}}(\mathcal{I}_p^*)$ overlap after projecting them are $\binom{M^n}{2} \in \mathcal{O}(M^{2n})$. So, there exists a choice for $\mathbf{a}$ such that $t = \mathbf{a} \cdot \mathbf{X}$ is separating for $V_{\mathbb{C}}(\mathcal{I}_p^*)$. The set in which we choose a prime number increases in every iteration. Therefore, we will be able to find a lucky prime. □

Before computing the bit-complexity of Alg. 2, we need to find the expected number of iterations of the loop in lines 3-12. Following [BLM+15, Lem. 31], we have the following.

Lemma 3.31. *The expected number iterations of the loop in Algorithm 2 is in $\mathcal{O}(n \log nM\Lambda)$.*

Proof. Let K be an upper bound on the number of unlucky primes which is in $\mathcal{O}(nM^{2n} + M^{2n-1}\Lambda + n\Lambda)$ (Lem. 3.29). At every iteration, the algorithm chooses a prime number from the interval (M^{2n}, B), where B starts from $2M^{2n}$ and doubles at each iteration. We fix the value B with whom the algorithm terminates. At the last iteration, the algorithm chooses a prime number from an interval containing the sub-interval $(B/2, B)$, and thus from a set of at least $\frac{B/2}{2\ln B/2}$ primes [vzGG13, Thm. 18.7].

Now, we consider two cases. In the first case, $\frac{B/2}{2\ln B/2} \leq 2K$. Then, since B doubles at each iteration, the total number of iterations is at most $\log(B)$, which is in $\mathcal{O}(\log K) = \mathcal{O}(n \log nM\Lambda)$

In the second case, $\frac{B/2}{2\ln B/2} > 2K$. For the choice of $\mathbf{a}$ and p at the last iteration of the loop, we have that:

$$\Pr(p \text{ is lucky and } \mathbf{a} \cdot X \text{ is separating for } V_{\mathbb{C}}(\mathcal{I}^*)) =$$
$$\underbrace{\Pr(p \text{ is lucky })}_{P_1} \cdot \underbrace{\Pr(\mathbf{a} \cdot X \text{ is separating for } V_{\mathbb{C}}(\mathcal{I}^*) | p \text{ is lucky })}_{P_2}.$$

The probability P_1 is $\geq \frac{1}{2}$ since p is chosen uniformly at random from a set containing at least $\frac{B/2}{2\ln B/2} > 2K$ primes. For P_2, we have that if p is lucky, therefore $V_{\mathbb{C}}(\mathcal{I}_p^*)$ has the same cardinality as $V_{\mathbb{C}}(\mathcal{I}^*)$. There are $\mathcal{O}(M^n)$ roots and thus $\binom{\mathcal{O}(M^n)}{2} \in \mathcal{O}(M^{2n})$ directions such that two solutions overlap after projecting them onto the separating linear form t. Since we choose $\mathbf{a}$ from a set of cardinality $(2M^{2n})^{n-1}$, we have that

$$P_2 \geq 1 - \frac{M^{2n}}{(2M^{2n})^{n-1}} \geq \frac{1}{2},$$

since $n > 1$. Thus, we exit the loop with probability $\geq 1/4$ each time, i.e., the expected number or iterations in this case is 4. So, in total we have that the expected number of iterations is $\mathcal{O}(n \log nM\Lambda)$. □

The bit-complexity of Alg. 2 can be proven now.

Theorem 3.32. *Given the radical ideal $\mathcal{I}^*$, Algorithm 2 computes a separating linear form for $V_{\mathbb{C}}(\mathcal{I}^*)$, which is of bitsize in $\mathcal{O}(n \log M)$, by performing*

$$\widetilde{\mathcal{O}}_B(n^2(n + 2^n)M^{2n} + n^2 M\Lambda + n^{10})$$

bit-operations in expected case.

Proof. The correctness follows from Lem. 3.30 and the bitsize of t from the choice of $\mathbf{a}$ (line 5). For the bit-complexity, we first consider one iteration of the loop in lines 3-12. The random vector $\mathbf{a}$ can be chosen in $\mathcal{O}_B(n^2 \log M)$, since each coordinate can be chosen in $\mathcal{O}_B(n \log M)$ bit-operations. The interval (M^{2n}, B) contains $(B/2, B)$, so there are at least $\frac{B/2}{2\ln B/2}$ primes in it [vzGG13, Thm. 18.7]. The probability that we choose a prime

number is at least $\frac{1}{4\ln B/2}$, so the prime is found after $\mathcal{O}(4\ln B/2)$ iterations is the expected case. Checking that a number p is prime is done in $\widetilde{\mathcal{O}}_B(\log^{7.5} B)$ [AKS02, Thm. 5.3]. So, a prime number p is found in expected bit-complexity $\widetilde{\mathcal{O}}_B(\log^9 B)$ and it has bitsize in $\mathcal{O}(\log B)$.

Computing the reduction modulo p of $\mathcal{I}^*$, amounts to reducing modulo p the coefficients of $F_1^*, \ldots, F_n^*$. The coefficients are $\mathcal{O}(nM)$ in total and each one has bitsize in $\widetilde{\mathcal{O}}(M+\Lambda)$. So, they are all reduced modulo p in $\widetilde{\mathcal{O}}_B(nM(M+\Lambda+\log B))$ bit-operations [vzGG13, Thm. 9.8].

The rest of the computations inside the loop is done in $\widetilde{\mathcal{O}}_B((n+2^n)M^{2n}\log B)$ bit-operations (Cor. 3.20). At the end of every iteration, we check if the computed characteristic polynomial is square-free in $\widetilde{\mathcal{O}}_B(M^{2n}\log B)$. The divisibility test of Π by p is done in $\widetilde{\mathcal{O}}_B(M\Lambda)$.

Thus, in every iteration, there are performed

$$\widetilde{\mathcal{O}}_B((n+2^n)M^{2n}\log B + nM\Lambda + \log^9 B)$$

bit-operations in expected case. At the last iteration $B = 2^j M^{2n}$, with $j \in \mathcal{O}(n\log nM\Lambda)$. So, the expected bit-complexity of the last iteration is in

$$\widetilde{\mathcal{O}}_B((n+2^n)M^{2n}n + nM\Lambda + n^9)$$

and the expected bit-complexity of all the iterations is in

$$\widetilde{\mathcal{O}}_B((n+2^n)M^{2n}n^2 + n^2M\Lambda + n^{10})\,.$$

□

3.5.4 Computing an RUR

In this subsection, Algorithm 3 computes a RUR of $\mathcal{I}^*$. First, a separating linear form for $V_{\mathbb{C}}(\mathcal{I}^*)$ is found by employing the Las Vegas algorithm of the previous section. For the rest, we use a multimodular approach. We first compute a set $\mathcal{S}$ of lucky primes whose product $\Pi_{\mathcal{S}}$ is greater than the output coefficients. Then, we compute the RUR modulo these primes and we lift the result to $\mathbb{Z}_{\Pi_{\mathcal{S}}}$ using the Chinese Remainder Theorem. The last step is to reconstruct the RUR to $\mathbb{Q}$ from the modular image in $\mathbb{Z}_{\Pi_{\mathcal{S}}}$. In the lemma that follows we prove that the solution to the rational reconstruction from $\mathbb{Z}_{\Pi_{\mathcal{S}}}$ to $\mathbb{Q}$ is indeed the RUR of $\mathcal{I}^*$.

Before proving the correctness of Alg. 3, we need the following general lemma.

Lemma 3.33. *Let ideal $I \subset \mathbb{Q}[X_1,\ldots,X_n]$. Given a linear form $t = \mathbf{a}\cdot\mathbf{X} \in \mathbb{Q}[\mathbf{X}]$ and $E_{\mathbf{a}} \in \mathbb{Q}[T]$ the characteristic polynomial of the multiplication matrix by t in $\mathbb{Q}[\mathbf{X}]/I$, we*

have that

$$f_{I,\mathbf{a},v}(T)\gcd(E_{\mathbf{a}},E'_{\mathbf{a}}) = \sum_{i=0}^{D} \mathrm{Tr}(vt^i)H_{D-i-1}(T)\,,$$

where D is the degree of $E_{\mathbf{a}}$ and $v \in \{1, X_1, \ldots, X_n\}$.

Proof. Let $E_{\mathbf{a}}(T) = \sum_{i=0}^{D} b_i T^i$. Following the proof of [Rou99, Thm. 3.1], we have that

$$\frac{f_{I,\mathbf{a},v}(T)\gcd(E_{\mathbf{a}},E'_{\mathbf{a}})}{f_{I,\mathbf{a}}(T)\gcd(E_{\mathbf{a}},E'_{\mathbf{a}})} = \sum_{i\geq 0} \frac{\mathrm{Tr}(vt^i)}{T^{i+1}}\,.$$

By multiplying both sides with $f_{I,\mathbf{a}}(T)\gcd(E_{\mathbf{a}}(T),E'_{\mathbf{a}}(T)) = E_{\mathbf{a}}(T)$, and given that $f_{I,\mathbf{a},v}(T)\gcd(E_{\mathbf{a}}(T),E'_{\mathbf{a}}(T))$ is a polynomial, we conclude that

$$f_{I,\mathbf{a},v}(T)\gcd(E_{\mathbf{a}}(T),E'_{\mathbf{a}}(T)) = \sum_{i=0}^{D-1}\sum_{j=0}^{D-i-1} \mathrm{Tr}(vt^i)b_jT^{D-i-j-1} = \sum_{i=0}^{D} \mathrm{Tr}(vt^i)H_{D-i-1}(T)\,.$$

□

Algorithm 3: `RUR - Las Vegas`

Input: The radical ideal $\mathcal{I}^*$

Output: A separating linear form $t = \mathbf{a}\cdot\mathbf{X}$ for $V_{\mathbb{C}}(\mathcal{I}^*)$ and the RUR of $\mathcal{I}^*$ associated to t

$\Pi = \mathrm{lc}(F_1^*)\cdot\cdots\cdot\mathrm{lc}(F_n^*)$
Compute an SLF $t = \mathbf{a}\cdot\mathbf{X}$ for $V_{\mathbb{C}}(\mathcal{I}^*)$ using Alg. 2
$B = 2(M^n(n+\mathcal{L}(a)) + nM^{n-1}(\Lambda + \log M)) \in \widetilde{\mathcal{O}}(nM^n + nM^{n-1}\Lambda)$
Compute a set $\mathcal{S}$ of $2B$ primes that are larger than M^{2n} and that do not divide Π
Let $\Pi_{\mathcal{S}}$ be the product of all the primes in $\mathcal{S}$
for $p \in \mathcal{S}$ **do**
  Compute $\mathcal{I}_p^*$, the reduction modulo p of $\mathcal{I}^*$
  Compute $\mathrm{Tr}^{\mathcal{A}_p(\mathcal{I}_p^*)}(t^i), \mathrm{Tr}^{\mathcal{A}_p(\mathcal{I}_p^*)}(X_jt^i)$ for $i \in [D], j \in [n]$
  Solve the system $(D-i)b_i = \sum_{j=0}^{i-1} b_{i-j}\mathrm{Tr}^{\mathcal{A}_p(\mathcal{I}_p^*)}(t^j)\,, \quad i = 0,\ldots,D$
  Set $f_{\mathcal{I}^*,\mathbf{a}}(T) = \sum_{i=0}^{D} b_iT^i$
  Compute the Horner polynomials $H_j(T) = \sum_{i=0}^{j-i} b_iT^{j-i}$ associated to $f_{\mathcal{I}^*,\mathbf{a}}(T)$
  Set $f_{\mathcal{I}^*,\mathbf{a},v} = \sum_{i=0}^{D} \mathrm{Tr}^{\mathcal{A}_p(\mathcal{I}_p^*)}(vt^i)H_{D-i-1}(T)$ for $v \in \{1, X_1, \ldots, X_n\}$
  $\mathrm{RUR}_p \leftarrow \{f_{\mathcal{I}^*,\mathbf{a}}, f_{\mathcal{I}^*,\mathbf{a},1}, f_{\mathcal{I}^*,\mathbf{a},X_1}, \ldots, f_{\mathcal{I}^*,\mathbf{a},X_n}\}$
end
Lift $\{\mathrm{RUR}_p\}_{p\in\mathcal{S}}$ to the RUR in $\mathbb{Z}_{\Pi_{\mathcal{S}}}$ using the Chinese Remainder Theorem
Reconstruct the RUR $\{f_{\mathcal{I}^*,\mathbf{a}}, f_{\mathcal{I}^*,\mathbf{a},1}, f_{\mathcal{I}^*,\mathbf{a},X_1}, \ldots, f_{\mathcal{I}^*,\mathbf{a},X_n}\}$ in $\mathbb{Q}$ from the RUR in $\mathbb{Z}_{\Pi_{\mathcal{S}}}$
RETURN $t = \mathbf{a}\cdot\mathbf{X}$, $\{f_{\mathcal{I}^*,\mathbf{a}}, f_{\mathcal{I}^*,\mathbf{a},1}, f_{\mathcal{I}^*,\mathbf{a},X_1}, \ldots, f_{\mathcal{I}^*,\mathbf{a},X_n}\}$

Lemma 3.34 (Correctness). *Algorithm 3 computes a RUR for $\mathcal{I}^*$. The polynomials*

$$f_{\mathcal{I}^*,\mathbf{a}}, f_{\mathcal{I}^*,\mathbf{a},1}, f_{\mathcal{I}^*,\mathbf{a},X_1}, \dots, f_{\mathcal{I}^*,\mathbf{a},X_n}$$

have degrees at most M^n and bitsize in $\widetilde{\mathcal{O}}(nM^n + nM^{n-1}\Lambda)$.

Proof. Correctness of line 2 comes from Lem. 3.30. Then B is a bound on the bitsize of the RUR polynomials, as computed in the proof of Lem. 3.11. In line 4, the primes in $\mathcal{S}$ are all lucky for the computation of the traces and of the characteristic polynomial, since they are greater than M^n and they do not divide Π. We note that we chose the primes to be greater than M^{2n} so that the separating linear form remains the same for every $p \in \mathcal{S}$.

Now, for a $p \in \mathcal{S}$, the characteristic polynomial of the multiplication by t in $\mathbb{Z}/p\mathbb{Z}[\mathbf{X}]$, is not necessarily squarefree, since p may not be lucky for the SLF. However, we set $f_{\mathcal{I}^*,\mathbf{a}}$ to be the characteristic polynomial and not its squarefree part, in contrast to the RUR candidate computation (Alg. 1). Likewise, in the computation of $f_{\mathcal{I}^*,\mathbf{a},v}$ we consider the Horner polynomials associated to $f_{\mathcal{I}^*,\mathbf{a}}$ and not to the squarefree part of the characteristic polynomial (line 10). So, the computed polynomials are all multiplied by $\gcd(f_{\mathcal{I}^*,\mathbf{a}}(T), f'_{\mathcal{I}^*,\mathbf{a}}(T))$ (Lem. 3.33). This is to guarantee that the polynomials have the correct degree and that $\{\mathrm{RUR}_p\}_{p\in\mathcal{S}}$ contains the images of the RUR modulo all the primes in $\mathcal{S}$.

For the correctness of line 16, we have to ensure that there is a solution to the rational number reconstruction problem of the coefficients of the RUR polynomials from their modular images.

Let c a coefficient in $\mathbb{Z}_{\Pi_\mathcal{S}}$. We want to find $\tilde{r}, \tilde{t} \in \mathbb{Z}$ such that for a $k \in \{1, \dots, \Pi_\mathcal{S}\}$, we have that

$$\gcd(t, \Pi_\mathcal{S}) = 1 \ , \ r^s - 1 = c \mod \Pi_\mathcal{S} \ , \ |r| < k \text{ and } 0 < s \leq \frac{\Pi_\mathcal{S}}{k} \ , \tag{3.23}$$

where s^{-1} is the inverse of s in $\mathbb{Z}_{\Pi_\mathcal{S}}$. From [vzGG13, Thm. 5.26] there is at most one solution to the Eq. (3.23) with $r < k/2$. We will show that there is exactly one solution to the Eq. (3.23) that is the inverse modular image of the corresponding coefficient in the RUR.

Let $\tilde{r}/\tilde{s} \in \mathbb{Q}$ be a coefficient of the RUR in $\mathbb{Q}$, with $\gcd(\tilde{r}, \tilde{s}) = 1$ and $\tilde{s} > 0$. Since B is an upper bound on the bitsize of the RUR coefficients, we have that $|\tilde{r}|, \tilde{s} \leq 2^B$. Now, $\Pi_\mathcal{S}$ is the product of $2B > 2$ primes, so it is greater than $2^{2B-1} \cdot 2^2 = 2^{2B+1}$. Then, $0 < \tilde{s} < 2^B < \frac{\Pi_\mathcal{S}}{2^{B+1}}$. We also have that $\gcd(\tilde{s}, \Pi_\mathcal{S}) = 1$, since $\Pi_\mathcal{S}$ is a product of lucky primes for the RUR computation by its definition and thus the reduction of $\tilde{r}\tilde{s}^{-1} \mod \Pi_\mathcal{S}$ is well defined, i.e. $\tilde{s} \not\equiv 0 \mod \Pi_\mathcal{S}$. So, for $k = 2^{B+1}$ we have that $\tilde{r}, \tilde{s}$ is the unique solution to the Eq. (3.23) with $|\tilde{r}| < k/2$.

Finally, Prop. 3.23 and Thm. 3.32 provide the bitsize of the RUR polynomials. □

Theorem 3.35. *Given the radical ideal $\mathcal{I}^*$, Algorithm 3 computes a separating linear form $t = \mathbf{a} \cdot \mathbf{X}$ for $V_{\mathbb{C}}(\mathcal{I}^*)$ such that the bitsize of $\mathbf{a} = (1, a_2, \ldots, a_n)$ is in $\mathcal{O}(n \log M)$, and the RUR for $\mathcal{I}^*$ associated to t in*

$$\widetilde{\mathcal{O}}_B(n(n + 2^n)M^{3n-1}(M + \Lambda) + n^{10})$$

bit-operations in the expected case. The polynomials

$$f_{\mathcal{I}^*,\mathbf{a}}, f_{\mathcal{I}^*,\mathbf{a},1}, f_{\mathcal{I}^*,\mathbf{a},X_1}, \ldots, f_{\mathcal{I}^*,\mathbf{a},X_n}$$

have degrees at most M^n and bitsize in $\widetilde{\mathcal{O}}(nM^n + nM^{n-1}\Lambda)$.

Proof. Correctness comes from the previous lemma. For the bit-complexity analysis, the SLF is computed in expected bit complexity $\widetilde{\mathcal{O}}_B(n^2(n+2^n)M^{2n}+n^2M\Lambda+n^{10})$ (Thm. 3.32). Then, in line 4 we compute the first $2B + \lceil \log \Pi \rceil$ primes that are larger than M^{2n} in $\widetilde{\mathcal{O}}_B(B + n\Lambda)$ [vzGG13, Thm. 18.10(ii)], since $\log \Pi \in \mathcal{O}(n\Lambda)$. Among them, there are at least $2B$ primes that do not divide Π, since its prime divisors are at most $n\Lambda$. We compute the reduction of the primes modulo Π and we keep $2B$ of them that do not reduce to zero. Since each prime has bitsize in $\mathcal{O}(\log B)$ [vzGG13, Thm. 18.10(ii)], all the reductions modulo Π can be done in $\widetilde{\mathcal{O}}_B(B)$ [vzGG13, Thm. 9.8]. Since all the primes in $\mathcal{S}$ have bitsize in $\widetilde{\mathcal{O}}(\log B)$, the bitsize of $\Pi_{\mathcal{S}}$ is in $\widetilde{\mathcal{O}}_B(B)$.

In the loop, for all $p \in \mathcal{S}$ we compute the modular images of $F_1^*, \ldots, F_n^*$ in $\widetilde{\mathcal{O}}_B(nM(M+\Lambda + B))$ bit-operations; the coefficients are $\mathcal{O}(nM)$ in total and each one has bitsize in $\widetilde{\mathcal{O}}(M + \Lambda)$. So, they are all reduced modulo all the primes p in $\widetilde{\mathcal{O}}_B(nM(M + \Lambda + \log \Pi_{\mathcal{S}}))$ bit-operations [vzGG13, Thm. 9.8]. Then, the rest of the computations in the each iteration of loop requires $\mathcal{O}((n + 2^n)M^{2n})$ arithmetic operations. Thus, the total cost of the loop is $\widetilde{\mathcal{O}}_B((n + 2^n)M^{2n}B)$ bit-operations.

Lifting $\{\mathrm{RUR}_p\}_{p\in\mathcal{S}}$ to $\mathbb{Z}_{\Pi_{\mathcal{S}}}$ amounts in lifting every coefficient to $\mathbb{Z}_{\Pi_{\mathcal{S}}}$. There are $\mathcal{O}(nM^n)$ coefficients and each lifting costs $\widetilde{\mathcal{O}}_B(\Pi_{\mathcal{S}}) = \widetilde{\mathcal{O}}_B(B)$, thus in total $\widetilde{\mathcal{O}}_B(nM^nB)$ bit-operations. Then, the rational reconstruction of a coefficient in $\mathbb{Z}_{\Pi_{\mathcal{S}}}$ is done in $\widetilde{\mathcal{O}}_B(\Pi_{\mathcal{S}}) = \widetilde{\mathcal{O}}(B)$ [WP03, Cor. 5.1]. In total this step costs $\widetilde{\mathcal{O}}_B(nM^nB)$ bit-operations.

Since $B \in \widetilde{\mathcal{O}}(nM^n + nM^{n-1}\Lambda)$ the result follows. □

3.5.5 From multiple to simple field extension

In this subsection, we utilize the RUR of $\mathcal{I}^*$ computed by Thm. 3.35, to transform the system of Eq. (3.1) into a bivariate system in triangular form. Then, by decomposing the latter bivariate system into regular triangular systems, we can deduce the number of distinct roots of $F_{\mathbf{x}}$ for every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I})$. Knowing this number, we can then isolate the roots of the system of Eq. (3.1) as described in the proof of Thm. 3.7(i).

Let $\{f_{\mathcal{I}^*,\mathbf{a}}, f_{\mathcal{I}^*,\mathbf{a},1}, f_{\mathcal{I}^*,\mathbf{a},X_1}, \ldots, f_{\mathcal{I}^*,\mathbf{a},X_n}\}$ be the RUR of $\mathcal{I}^*$ associated to the separating

linear form $t = \mathbf{a} \cdot \mathbf{X}$. We plug-in of the RUR in the system of Eq. (3.1), and then we have the bivariate system

$$\begin{aligned} f_{\mathcal{I}^*,\mathbf{a}}(T) &= 0\,, \\ \tilde{F}(T,Y) := f_{\mathcal{I}^*,1}(T)^d F\left(\frac{f_{\mathcal{I}^*,\mathbf{a},X_1}(T)}{f_{\mathcal{I}^*,1}(T)}, \ldots, \frac{f_{\mathcal{I}^*,\mathbf{a},X_n}(T)}{f_{\mathcal{I}^*,1}(T)}, Y\right) &= 0\,. \end{aligned} \tag{3.24}$$

In order to compute this, we employ the Las Vegas algorithm of [HL21, Cor. 3.6] for multivariate modular composition. This has expected complexity in

$$\widetilde{\mathcal{O}}_B(\max(d,M)^{(1+\varepsilon)n}(ndM^n + ndM^{n-1}\Lambda + \tau))\,, \tag{3.25}$$

for any $\varepsilon > 0$, since the output bitsize is $\widetilde{\mathcal{O}}(ndM^n + ndM^{n-1}\Lambda + \tau)$.

We now demonstrate the procedure to determine the number of distinct roots of $\tilde{F}(t,Y)$ for every $t \in V_{\mathbb{C}}(f_{\mathcal{I}^*,\mathbf{a}})$ following [DDR+22]. We write $\tilde{F}(T,Y) = \sum_{\ell=0}^{d} \tilde{f}_\ell(T)Y^\ell$. We denote by $\tilde{F}^\ell(T,Y)$ the truncation of $\tilde{F}$, containing only the terms of degree $\leq \ell$ with respect to Y, for $\ell = 0,\ldots,d$. First, we compute the polynomials $R_\ell \in \mathbb{Q}[T]$, $\ell = 0,\ldots,d$, such that

$$\tilde{F}(t,Y) = \tilde{F}^\ell(t,Y) \Leftrightarrow R_\ell(t) = 0\,.$$

They are defined as follows:

$$\begin{aligned} R_{\leq d}(T) &:= f_{\mathcal{I}^*,\mathbf{a}}(T)\,, \\ R_{\leq \ell-1}(T) &:= \gcd(R_{\leq \ell}(T), \tilde{f}_\ell(T)) \text{ for } \ell = 1,\ldots d \text{ and} \\ R_{\ell-1}(T) &:= \frac{R_{\leq \ell}(T)}{R_{\leq \ell-1}(T)}\,. \end{aligned}$$

Therefore, we can decompose the system of Eq. (3.24) into the regular systems

$$\begin{aligned} R_\ell(T) &= 0\,, \\ \tilde{F}^\ell(T,Y) &= 0\,, \end{aligned}$$

for $\ell = 1,\ldots,d$. Then, for every ℓ, we compute the principal subresultant coefficients of $\tilde{F}^\ell(T,Y)$ and $\frac{\partial \tilde{F}^\ell}{\partial Y}(T,Y)$, which we denote by $r_{\ell,k}(T)$, $k = 0,\ldots,\ell$. We define the following polynomials in $\mathbb{Q}[T]$:

$$\begin{aligned} \Delta^\ell_{\geq 0}(T) &:= R_\ell(T)\,, \\ \Delta^\ell_{\geq k+1}(T) &:= \gcd(\Delta^\ell_{\geq k}(T), r_{\ell,k}(T)) \text{ for } k = 0,\ldots \ell-1 \text{ and} \\ \Delta^\ell_k(T) &:= \frac{\Delta^\ell_{\geq k}(T)}{\Delta^\ell_{\geq k+1}(T)}\,. \end{aligned}$$

We have that

$$\deg_Y(\gcd(\tilde{F}^\ell(t,Y), \frac{\partial \tilde{F}^\ell}{\partial Y}(t,Y))) = k \Leftrightarrow \Delta_k^\ell(t) = 0\,,$$

meaning that $\deg_Y(\tilde{F}(t,Y)) = \ell$ and $\tilde{F}^\ell(t,Y)$ has $d-k$ distinct roots in $\mathbb{C}$ if and only if $\Delta_k^\ell(t) = 0$. We also have that

$$f_{\mathcal{I}^*,\mathbf{a}}(T) = \mathrm{cont}(f_{\mathcal{I}^*,\mathbf{a}}) \prod_{\ell,k} \Delta_k^\ell(T)\,.$$

Using this decomposition, we can then find the number of distinct roots of $\tilde{F}(t,Y)$ for every $t \in V_{\mathbb{C}}(f_{\mathcal{I},\mathbf{a}})$. In the lemma that follows we analyse the bit-complexity.

Lemma 3.36. *For every $t \in V_{\mathbb{C}}(f_{\mathcal{I}^*,\mathbf{a}})$, we determine the number of distinct roots of $\tilde{F}(t,Y)$ in expected bit-complexity*

$$\widetilde{\mathcal{O}}_B(nd^6M^{2n} + nd^6M^{2n-1}\Lambda + d^5M^n\tau)\,,$$

with a Las Vegas algorithm.

Proof. To compute all the polynomials $R_{\leq \ell-1}$ we need to perform d gcd computations. The polynomials $\tilde{f}_\ell$ have bitsize in $\widetilde{\mathcal{O}}(ndM^n + ndM^{n-1}\Lambda + \tau)$ and degree $\mathcal{O}(dM^n)$. So, we do the gcd computations in an expected complexity $\widetilde{\mathcal{O}}_B(nd^2M^{2n} + nd^2M^{2n-1}\Lambda + dM^n\tau)$ [KRTZ20a, Lem.2.2]. Then, to compute $R_{\ell-1}(T)$ we perform exact divisions of $R_{\leq \ell}(T)$ with $R_{\leq \ell-1}(T)$. These polynomials have bitsize in $\widetilde{\mathcal{O}}(nM^n + nM^{n-1}\Lambda)$ and degree at most M^n, so this is done for all ℓ in $\widetilde{\mathcal{O}}_B(nM^{2n} + nM^{2n-1}\Lambda)$ [vzGG13, Ex.10.21].

Now, we take $\ell = d$. We have that $\tilde{F}^d(T,Y)$ is of degree d in Y, of degree $\mathcal{O}(dM^n)$ in T, and of bitsize in $\widetilde{\mathcal{O}}(ndM^n + ndM^{n-1}\Lambda + \tau)$. The degree of $r_{d,k}(T)$ is in $\mathcal{O}(d^2M^n)$ and its bitsize is in $\widetilde{\mathcal{O}}(nd^2M^n + nd^2M^{n-1}\Lambda + d\tau)$; they are computed for all k in $\widetilde{\mathcal{O}}_B(d^4M^n(ndM^n + ndM^{n-1}\Lambda + \tau))$ [LPR17, Lem.4]. In order to compute $\Delta_{\geq k}^d(T)$ for all k, we need to perform d gcd computations. We do so in an expected complexity $\widetilde{\mathcal{O}}_B(nd^5M^{2n} + nd^5M^{2n-1}\Lambda + d^4M^n\tau)$ [KRTZ20a, Lem.2.2]. In the end, to compute $\Delta_k^d(T)$ we perform exact divisions of $\Delta_{\geq k}^d(T)$ with $\Delta_{\geq k+1}(T)$. This is done again for all k in $\widetilde{\mathcal{O}}_B(nd^5M^{2n} + nd^5M^{2n-1}\Lambda + d^4M^n\tau)$[vzGG13, Ex.10.21]. We multiply the latter bound by d conclude. □

In the next theorem, we put together the previous results to prove the bit-complexity of a Las-Vegas algorithm to isolate the roots of the system in Eq. (3.1).

Theorem 3.37. *There is a Las Vegas algorithm that computes isolating discs for all the roots of the system in Eq. (3.1), together with the corresponding multiplicities in expected*

complexity

$$\widetilde{\mathcal{O}}_B\big(n(n+2^n)M^{3n-1}(M+\Lambda)+n^{10}+nd^6M^{2n}+nd^6M^{2n-1}\Lambda+d^5M^n\tau+ \\ nM^{n+1}d(n+\tau)+nM^nd^2(n\Lambda+d^n(n+\tau+d+\Lambda))+n^2d^{n+3}M^{n-1}\Lambda\big)\,.$$

Proof. The proof essentially follows from the combination of Thm. 3.35 for the RUR computation, the cost of computing the system of Eq. (3.24), which is given in Eq. (3.25), Lem. 3.36 for the complexity to determine the number of distinct roots, and from Thm. 3.7(i) for the root isolation. Note that there always exists a choice of $\varepsilon > 0$ so that the bit-complexity in Eq. (3.25) gets dominated. Also, we have to add another step that finds the correspondence of each $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I}^*)$ with a root of $f_{\mathcal{I}^*,\mathbf{a}}$. This can be done by evaluating the separating linear form $\mathbf{a}\cdot\mathbf{X}$ at suitable approximations $\tilde{\mathbf{x}}$ of every $\mathbf{x} \in V_{\mathbb{C}}(\mathcal{I}^*)$, so that $\mathbf{a}\cdot\tilde{\mathbf{x}}$ belongs to the isolating interval of the root $\mathbf{a}\cdot\mathbf{x}$ of $f_{\mathcal{I}^*,\mathbf{a}}$. To perform these evaluations, we work as in the proof of Lem. 3.14, where we employ Prop. 3.1 for $L \in \widetilde{\mathcal{O}}(nM^{2n}+nM^{2n-1}\Lambda)$. This costs $\widetilde{\mathcal{O}}_B(n^2M^{3n}+n^2M^{3n-1}\Lambda)$, since the polynomial that we evaluate is now linear and has bitsize in $\widetilde{\mathcal{O}}(n)$. So, the bit-complexity of this operation does not overcome the total one, which is

$$\widetilde{\mathcal{O}}_B\big(n(n+2^n)M^{3n-1}(M+\Lambda)+n^{10}+nd^6M^{2n}+nd^6M^{2n-1}\Lambda+d^5M^n\tau+ \\ nM^{n+1}d(n+\tau+d)+nM^nd^2(n\Lambda+d^n(n+\tau+d+\Lambda))+n^2d^{n+3}M^{n-1}\Lambda\big)\,.$$

After a simplification, we arrive to the desired bound. □

Remark 3.38. *When $M = d$ and $\Lambda = \tau$, the bit-complexity bound of the previous theorem becomes*

$$\widetilde{\mathcal{O}}_B\big(n(n+2^n)d^{3n-1}(d+\tau)+nd^{2n+5}(d+\tau)+n^{10}\big)\,,$$

which simplifies to

$$\begin{cases}\widetilde{\mathcal{O}}_B\big(n(n+2^n)d^{3n-1}(d+\tau)+n^{10}\big) & \text{when } n \geq 6\,,\\ \widetilde{\mathcal{O}}_B\big(d^{2n+5}(d+\tau)\big) & \text{otherwise.}\end{cases}$$

3.6 Application: Sum Of Square roots of integers problem

We consider the problem of determining the minimum non-zero difference between two sums of square roots of integers. It appears as Problem 33 in 'The Open Problems Project' and was originally addressed by Joseph O'Rouke [DMO33].

Let $a_i, b_i \in \mathbb{Z}^{\geq 0}$ for $i = 1, \ldots, n$ of bitsize τ. We want to decide if $\sum_{i=1}^n \sqrt{a_i}$ is less than, equal to, or greater than $\sum_{i=1}^n \sqrt{b_i}$. This problem is also related to the Euclidean Travel Salesman Problem (TSP): Given a set of points in the plane with integer coordinates and

$L \in \mathbb{N}$, decide if there exists a circuit passing through all these points and having total length (with respect to the Euclidean distance) at most L. The length of the path is a sum of square-roots of integers.

Comparing $\sum_{i=1}^{n} \sqrt{a_i}$ with $\sum_{i=1}^{n} \sqrt{b_i}$ in the real-RAM model, can be done trivially. However, in the bit-complexity setting, one has to determine the number of bits that is sufficient to obtain a correct result. We by $r(n,\tau)$ denote the minimum positive value of $\left|\sum_{i=1}^{n} \sqrt{a_i} - \sum_{i=1}^{n} \sqrt{b_i}\right|$. Lower bounds on $r(n,\tau)$, and in turn upper bounds on $-\log r(n,\tau)$, give upper bounds on the precision needed to compare $\sum_{i=1}^{n} \sqrt{a_i}$ with $\sum_{i=1}^{n} \sqrt{b_i}$. In particular, if $-\log r(n,\tau)$ is bounded above by a polynomial in k and n, then the sign of $\sum_{i=1}^{n} \sqrt{a_i} - \sum_{i=1}^{n} \sqrt{b_i}$ can be computed in polynomial time. Nevertheless, existing upper bounds on $-\log r(n,\tau)$ are exponential. In [BFMS00, MS00] they prove that $-\log r(n,\tau) \in \widetilde{\mathcal{O}}(\tau 2^{2n})$, through studying separation bounds.

Here, we apply the results of Sec. 3.3 to derive bounds that, however, remain exponential in n. Nonetheless, the same bounds apply to the sum of all the roots of the associated system that has as root the two quantities that we have to compare. We consider the system

$$\left\{\begin{array}{c} F_i(X_i) := X_i^2 - a_i = 0\,, i \in [n] \\ G_i(Y_i) := Y_i^2 - b_i = 0\,, i \in [n] \\ H(\mathbf{X},\mathbf{Y},Z) := (Z - X_1 - \cdots - X_n)(Z - Y_1 - \cdots - Y_n) = 0 \end{array}\right\}$$

Let $\mathcal{K} = \langle F_1, \ldots, F_n, G_1, \ldots, G_n \rangle$. From Cor.3.6 we have that

$$\sum_{(\mathbf{x},\mathbf{y}) \in V_{\mathbb{C}}(\mathcal{K})} \mu_{\mathcal{K}}(\mathbf{x},\mathbf{y}) \mathrm{lsep}(H(\mathbf{x},\mathbf{y},\cdot)) \in \widetilde{\mathcal{O}}(n 2^{2n} \tau)\,,$$

or equivalently, since all the multiplicities are equal to one,

$$\sum_{(\mathbf{x},\mathbf{y}) \in V_{\mathbb{C}}(\mathcal{K})} \left|\log\left|\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i\right|\right| \in \widetilde{\mathcal{O}}(n 2^{2n} \tau)\,. \tag{3.26}$$

We see that, as Eq. (3.26) shows, not only $-\log r(n,\tau)$ is in $\widetilde{\mathcal{O}}(n\tau 2^{2n})$, but also the sum of the differences associated to all the 2^{2n} roots of the system $\{F_1 = \cdots = F_n = G_1 = \cdots = G_n = 0\}$.

Second Part

Algorithms on Parametric Curves

Chapter 4

Topology of Parametric Curves

We consider the problem of computing the topology and describing the geometry of a parametric curve in $\mathbb{R}^n$. We present an algorithm, `PTOPO`, that constructs an abstract graph that is isotopic to the curve in the embedding space. Our method exploits the benefits of the parametric representation and does not resort to implicitization.

Most importantly, we perform all computations in the parameter space and not in the implicit space. When the parametrization involves polynomials of degree at most d and maximum bitsize of coefficients τ, then the worst case bit complexity of `PTOPO` is $\widetilde{\mathcal{O}}_B(nd^6 + nd^5\tau + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau)$. This bound matches the current record bound $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$ for the problem of computing the topology of a plane algebraic curve given in implicit form. For plane and space curves, if $N = \max\{d, \tau\}$, the complexity of `PTOPO` becomes $\widetilde{\mathcal{O}}_B(N^6)$, which improves the state-of-the-art result, due to Alcázar and Díaz-Toca [ADT10], by a factor of N^{10}. In the same time complexity, we obtain a graph whose straight-line embedding is isotopic to the curve. For curves of general dimension, we can also distinguish between ordinary and non-ordinary real singularities and determine their multiplicities in the same expected complexity of `PTOPO` by employing the algorithm of Blasco and Pérez-Díaz [BPD19]. We have implemented `PTOPO` in MAPLE for the case of plane and space curves. Our experiments illustrate its practical nature.

4.1 Introduction

Parametric curves constitute a classical and important topic in computational algebra and geometry [SW99] that constantly receives attention, e.g., [Sed86a, CKPU11, BLY19, SWPD08]. The motivation behind the continuous interest in efficient algorithms for computing with parametric curves emanates, among others reasons, by the frequent presence of parametric representations in computer modeling and computer aided geometric design, e.g., [FGS10].

We focus on computing the topology of a real parametric curve, that is, the computation

of an abstract graph that is isotopic [BT06, p. 184] to the curve in the embedding space. We design an algorithm, `PTOPO`, that applies directly to rational parametric curves of any dimension and is complete, in the sense that there are no assumptions on the input. We consider different characteristics of the parametrization, like properness and normality, before computing the singularities and other interesting points on the curve. These points are necessary for representing the geometry of the curve, as well as for producing a certified visualization of plane and space curves.

Previous work. A common strategy when dealing with parametric curves is implicitization. There has been a lot of research effort, e.g., [SC95, BLY19] and the references therein, in designing algorithms to compute the implicit equations describing the curve. However, it is also important to manipulate parametric curves directly, without converting them to implicit form. For example, in the parametric form it is easier to visualize the curve and to find points on it. The advantages of the latter become more significant for curves of high dimension.

The study of the topology of a real parametric curve is a topic that has not received much attention in the literature, in contrast to its implicit counterpart [DDR$^+$22, KS15]. The computation of the topology requires special treatment, since for instance it is not always easy to choose a parameter interval such that when we plot the curve over it, we include all the important topological features (like singular and extreme points) [ADT10]. Moreover, while visualizing the curve using symbolic computational tools, the problem of missing points and branches may arise [AR07, Sen02]. [ADT10] study the topology of real parametric curves without implicitizing. They work directly with the parametrization and address both plane and space real rational curves. Our algorithm to compute the topology is to be juxtaposed to their work. We also refer to [CFGVN14] and [AMW08] for other approaches based on computations by values and subdivision, respectively.

To compute the topology of a curve it is essential to detect its singularities. This is an important and well studied problem [ADT10, RSV09, KS15] of independent interest. To identify the singularities, we can first compute the implicit representation and then apply classical approaches [Wal78, Ful69]. Alternatively, we can compute the singularities using directly the parametrization. For instance, there are necessary and sufficient conditions to identify cusps and inflection points using determinants, e.g., [LC97, MC92].

On computing the singularities of a parametric curve, a line of work related to our approach, does so by means of a univariate resultant [AB89, vdEY97, Par02, PD07, GRS02]. We can use the Taylor resultant [AB89] and the D-resultant [vdEY97] of two polynomials in $K[t]$, to find singularities of plane curves parametrized by polynomials, where K is a field of characteristic zero in the first case and of arbitrary characteristic in the latter, without resorting to the implicit form. Park [Par02] extends previous results to curves parametrized by polynomials in affine n-space. The generalization of the D-resultant for a

pair of rational functions and its application to the study of rational plane curves, is due to [GRS02]. In [PD07, BPD19] they present a method for computing the singularities of plane curves using a univariate resultant and characterizing the singularities using its factorization. Notably, Rubio et al. [RSV09] work on rational parametric curves in affine n-space; they use generalized resultants to find the parameters of the singular points. Moreover, they characterize the singularities and compute their multiplicities.

Cox et al. [CKPU11] use the syzygies of the ideal generated by the polynomials that give the parametrization to compute the singularities and their structure. There are state-of-the-art approaches that exploit this idea and relate the problem of computing the singularities with the notion of the μ-basis of the parametrization, e.g., [JSC18] and references therein. In [CS01a], they reveal the connection between the implicitization Bézout matrix and the singularities of a parametric curve. Busé and D'Andrea [BD12] present a complete factorization of the invariant factors of resultant matrices built from birational parametrizations of rational plane curves in terms of the singular points of the curve and their multiplicity graph. Let us also mention the important work on matrix methods [Bus14, BLB10] for representing the implicit form of parametric curves, that is suitable for numerical computations. In [BGI16], the authors use the projection from the rational normal curve to the curve and exploit secant varieties.

Overview of our approach and our contributions. We introduce `PTOPO`, a complete, exact, and efficient algorithm (Alg. 6) for computing the geometric properties and the topology of rational parametric curves in $\mathbb{R}^n$. Unlike other algorithms, e.g. [ADT10], it makes no assumptions on the input curves, such as the absence of axis-parallel asymptotes, and is applicable to any dimension. Nevertheless, it does not identify knots for space curves nor it can be used for determining the equivalence of two knots.

If the (proper) parametrization of the curve consists of polynomials of degree d and bitsize τ, then `PTOPO` outputs a graph isotopic [BT06, p.184], [ACDT20] to the curve in the embedding space, by performing

$$\widetilde{\mathcal{O}}_B(nd^6 + nd^5\tau + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau)$$

bit operations in the worst case (Thm. 4.24), assuming no singularities at infinity. We also provide a Las Vegas variant with expected complexity

$$\widetilde{\mathcal{O}}_B(d^6 + d^5(n + \tau) + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau).$$

If $n = \mathcal{O}(1)$, the bounds become $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$. The vertices of the output graph correspond to special points on the curve, in whose neighborhood the topology is not trivial, given by their parameter values. Each edge of the graph is associated with two parameter values and corresponds to a unique smooth parametric arc. For an embed-

ding isotopic to the curve, we map every edge of the abstract graph to the corresponding parametric arc.

For plane and space curves, our bound improves the previously known bound due to [ADT10] by a factor of $\widetilde{\mathcal{O}}_B(N^{10})$. The latter algorithm [ADT10] performs some computations in the implicit space. On the contrary, `PTOPO` is a fundamentally different approach since we work exclusively in the parameter space and we do not use a sweep-line algorithm to construct the isotopic graph. We handle only the parameters that give important points on the curve, and thus, we avoid performing operations such as univariate root isolation in a field extension or evaluation of a polynomial at an algebraic number.

Computing singular points is an essential part of `PTOPO` (Lem. 4.18). We chose not to exploit recent methods, e.g., [BPD19], for this task because this would require introducing new variables (to employ the T-resultant). We employ older techniques, e.g., [RSV09, PD07, ADT10], that rely on a bivariate polynomial system, Eq. (4.2). We take advantage of this system's symmetry and of nearly optimal algorithms for bivariate system solving and for computations with real algebraic numbers [DDR+22, BLM+16, DET09, PT17]. In particular, we introduce an algorithm for isolating the roots of over-determined bivariate polynomial systems by exploiting the Rational Univariate Representation (RUR) [BLPR15, BLM+16, BLPR13] that has worst case and expected bit complexity that matches the bounds for square systems (Thm. 4.16). These are key steps for obtaining the complexity bounds of Thm. 4.23 and Thm. 4.24.

Moreover, our bound matches the current state-of-the-art complexity bound, $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$ or $\widetilde{\mathcal{O}}_B(N^6)$, for computing the topology of implicit plane curves [DDR+22, KS15]. However, if we want to visualize the graph in 2D or 3D, then we have to compute a characteristic box (Lem. 4.20) that contains all the the topological features of the curve and the intersections of the curve with its boundary. This can be done in the same bit-complexity (Thm. 4.23).

A preprocessing step of `PTOPO` consists of finding a proper reparametrization of the curve (if it is not proper). We present explicit bit complexity bounds (Lem. 4.6) for the algorithm of [PD06] to compute a proper parametrization. Another preprocessing step is to ensure that there are no singularities at infinity. Lem. 4.7 handles this task and provides explicit complexity estimates.

Additionally, we consider the case where the embedding of the abstract graph has straight line edges and not parametric arcs; in particular for plane curves, we show that the straight line embedding of the abstract graph in $\mathbb{R}^2$ is already isotopic to the curve (Cor. 4.26). For space curves, the procedure supported by Thm. 4.28 adds a few extra vertices to the abstract graph, so that the straight line embedding in $\mathbb{R}^3$ is isotopic to the curve. The extra number of vertices serves in resolving situations where self-crossings occur when continuously deforming the graph to the curve. In Thm. 4.29 we also prove that for curves of any dimension, we can compute the multiplicities and the characterization

of singular points in the same bit-complexity as computing the points (in a Las Vegas setting). For that, we use the method by [BPD19], which does not require any further computations apart from solving the system that gives the parameters of the singular points (cf. [ADT10]).

Last but not least, we provide a certified implementation[1] of `PTOPO` in MAPLE. The implementation computes the topology of plane and space curves and visualizes them. If the input consists of rational polynomials, our algorithm and the implementation is certified, since, first of all, the algorithm always outputs the correct geometric and topological result. This is because we perform exact computations with real algebraic numbers based on arithmetic over the rationals. Moreover, no assumption that cannot be verified (for example by another algorithm) is made on the input.

Outline. The next section presents some useful results needed for our proofs. In Sec. 4.3 we give the basic background on rational curves in affine n-space. We characterize the parametrization by means of injectivity and surjectivity and describe a reparametrization algorithm. In Sec. 4.4 we present the algorithm to compute the singular, extreme points (in the coordinate directions), and isolated points on the curve. In Sec. 4.5 we describe our main algorithm, `PTOPO`, that constructs a graph isotopic to the curve in the embedding space and its complexity. In Sec. 4.6 we expatiate on the isotopic embedding for plane and space curves. In Sec. 4.7, we study the multiplicities and the character of real singular points for curves of arbitrary dimension. Finally, in Sec. 4.8 we give examples and experimental results.

4.2 Algebraic tools

We present some useful results, needed for our analysis.

Lemma 4.1. *Let $A = \sum_{i=0}^{m} a_i X^i, B = \sum_{i=0}^{n} b_i X^i \in \mathbb{Z}[X]$ of degrees m and n and of bit-sizes τ and σ respectively. Let $\alpha_1, \ldots, \alpha_m$ be the complex roots of A, counting multiplicities. Then, for any $\kappa = 1, \ldots, m$ it holds that*

$$2^{-m\sigma - n\tau - (m+n)\log(m+n)} < |B(\alpha_\kappa)| < 2^{m\sigma + n\tau + (m+n)\log(m+n)}.$$

Proof. Following [ST19], let $R = \mathtt{res}_X(A(X), Y - B(X)) \in \mathbb{Z}[Y]$. Using the Poisson's formula for the resultant we can write $R(Y) = a_m^n \prod_{\kappa=1}^{m}(Y - B(\alpha_\kappa))$. The maximum bitsize of the coefficients of $R(Y)$ is at most $m\sigma + n\tau + (m+n)\log(m+n)$. We observe that the roots of $R(Y)$ are $B(\alpha_\kappa)$ for $\kappa = 1, \ldots, m$. Therefore, using Cauchy's bound we

[1] `https://gitlab.inria.fr/ckatsama/ptopo`

deduce that

$$2^{-m\sigma-n\tau-(m+n)\log(m+n)} < |B(\alpha_\kappa)| < 2^{m\sigma+n\tau+(m+n)\log(m+n)}.$$

□

Lemmata 4.2 and 4.3 restate known results on the gcd computation of various univariate and bivariate polynomials.

Lemma 4.2. *Let $f_1(X), \dots, f_n(X) \in \mathbb{Z}[X]$ of sizes (d, τ). We can compute their* gcd*, which is of size $(d, \widetilde{\mathcal{O}}(d+\tau))$, in worst case complexity $\widetilde{\mathcal{O}}_B(n(d^3+d^2\tau))$, with a Monte Carlo algorithm in $\widetilde{\mathcal{O}}_B(d^2+d\tau)$, or with a Las Vegas algorithm in $\widetilde{\mathcal{O}}_B(n(d^2+d\tau))$.*

Proof. These are known results [vzGG13]. We repeat the arguments adapted to our notation.

Worst case: We compute g by performing n consecutive gcd computations, that is

$$\gcd(f_1, \gcd(f_2, \gcd(\cdots, \gcd(f_{n-1}, f_n))).$$

Since each gcd computation costs $\widetilde{\mathcal{O}}_B(d^3+d^2\tau)$ [BLPR15, Lem.4], the result for this case follows.

Monte Carlo: We perform one gcd computation by allowing randomization. If we choose integers $a_3, \dots, a_n$ independently at random from the set $\{1, \dots, Kd\}$, where $K = \mathcal{O}(1)$, we get that $\gcd(f_1, \dots, f_n) = \gcd(f_1, f_2+a_3f_3+\cdots+a_nf_n)$ in $\mathbb{Z}[x]$, with probability $\geq 1/2$ [vzGG13, Thm. 6.46]. This actually computes the monic gcd in $\mathbb{Q}$. To compute the gcd in $\mathbb{Z}$ we need to multiply with the gcd of the leading coefficients of $f_1, f_2+a_3f_3+\cdots+a_nf_n$ and then take the primitive part of the resulting polynomial. This is sufficient since the leading coefficient of the gcd in $\mathbb{Z}[X]$ divides the leading coefficients of the two polynomials. Also, by [vzGG13, Cor. 6.10] the monic gcd of two polynomials in $\mathbb{Q}[X]$ is equal to their gcd in $\mathbb{Z}[X]$ divided by their leading coefficient. The gcd of the two leading coefficients of $f_1, f_2+a_3f_3+\cdots+a_nf_n$ is an integer of bitsize $\widetilde{\mathcal{O}}(\tau)$, therefore this does not pollute the total complexity.

We compute $g^* = \gcd(f_1, f_2+a_3f_3+\cdots+a_nf_n)$. Notice that the polynomial $f_2+a_3f_3+\cdots+a_nf_n$ is asymptotically of size (d, τ). So, it takes $\widetilde{\mathcal{O}}_B(d^2+d\tau)$ to find g^*, using the probabilistic algorithm in [Sch88].

Las Vegas: We can reduce the probability of failure in the Monte Carlo variant of the gcd computation to zero, by performing n exact divisions. In particular, we check if g^* divides $f_3, \dots, f_n$. Using [vzGG13, Ex.10.21], the bit complexity of these operations is in total $\widetilde{\mathcal{O}}_B(n(d^2+d\tau))$. □

Lemma 4.3. *Let $f_1(X,Y), \dots, f_n(X,Y) \in \mathbb{Z}[X,Y]$ of bidegrees (d,d) and $\mathcal{L}(f_i) = \tau$. We can compute their* gcd*, which is of bitsize $\widetilde{\mathcal{O}}(d+\tau)$, in worst case complexity $\widetilde{\mathcal{O}}_B(n(d^5+$*

$d^4\tau$)), with a Monte Carlo algorithm in $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$, or with a Las Vegas algorithm in $\widetilde{\mathcal{O}}_B(n(d^3 + d^2\tau))$.

Proof. The straightforward approach is to perform n consecutive gcd computations, that is

$$\gcd(f_1, \gcd(f_2, \gcd(\cdots, \gcd(f_{n-1}, f_n))).$$

To accelerate the practical complexity we sort f_i in increasing order with respect to their degree. Each gcd computation costs $\widetilde{\mathcal{O}}_B(d^5 + d^4\tau)$ [BLM+16, Lem. 5], so the total worst case cost is $\widetilde{\mathcal{O}}_B(nd^5 + nd^4\tau)$.

Alternatively, we consider the operation $\gcd(f_1, \sum_{k=2}^{n} a_k f_k)$, where a_k are random integers, following [vzGG13, Thm. 6.46]. The expected cost of this gcd is $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$. To see this, notice that we can perform a bivariate gcd in expected time $\widetilde{\mathcal{O}}(d^2)$ [vzGG13, Cor. 11.12], over a finite field with enough elements, and the bitsize of the result is $\widetilde{\mathcal{O}}(d+\tau)$ [Mah62].

Then, for a Las Vegas algorithm, using exact division, we test if the resulting polynomial divides all f_i, for $2 \le i \le n$. This costs $\widetilde{\mathcal{O}}_B(n(d^3 + d^2\tau))$, by adapting [vzGG13, Ex.10.21] to the bivariate case. □

4.3 Rational curves

Following [ADT10] closely, we introduce basic notions for rational curves. Let $\widetilde{\mathcal{C}}$ be an algebraic curve over $\mathbb{C}^n$, parametrized by the map

$$\begin{aligned} \phi : \quad & \mathbb{C} \dashrightarrow \widetilde{\mathcal{C}} \\ & t \mapsto \big(\phi_1(t), \ldots, \phi_n(t)\big) = \Big(\frac{p_1(t)}{q_1(t)}, \ldots, \frac{p_n(t)}{q_n(t)}\Big), \end{aligned} \tag{4.1}$$

where $p_i, q_i \in \mathbb{Z}[t]$ are of size (d, τ) for $i \in [n]$, and $\widetilde{\mathcal{C}}$ is the Zariski closure of $\mathrm{Im}(\phi)$. We call $\phi(t)$ a *parametrization* of $\widetilde{\mathcal{C}}$.

We study the real trace of $\tilde{\mathcal{C}}$, that is $\mathcal{C} := \tilde{\mathcal{C}} \cap \mathbb{R}^n$. A parametrization ϕ is chatacterized by means of *properness* (Sec. 4.3.1) and *normality* (Sec. 4.3.2). To ensure these properties, one can reparametrize the curve, i.e., apply a rational change of parameter to the given parametrization. We refer to [SWPD08, Ch. 6] for more details on reparametrization.

Without loss of generality, we assume that no coordinate of the parametrization ϕ is constant; otherwise we could embed $\tilde{\mathcal{C}}$ in a lower dimensional space. We consider that ϕ is in *reduced form*, i.e., $\gcd(p_i(t), q_i(t)) = 1$, for all $i \in [n]$. The point at infinity, $\mathbf{p}_\infty$, is the point on $\mathcal{C}$ we obtain for $t \to \pm\infty$ (if it exists). For a parametrization ϕ, we consider the

following system of bivariate polynomials:

$$h_i(s,t) = \frac{p_i(s)q_i(t) - q_i(s)p_i(t)}{s-t}, \qquad \text{for } i \in [n]. \tag{4.2}$$

Remark 4.4. *For every $i \in [n]$ $h_i(s,t)$ is a polynomial since (s,s) is a root of the numerator for every s. Also, $h_i(t,t) = \phi_i'(t)q_i^2(t)$ [GRY02, Lem. 1.7].*

4.3.1 Proper parametrization

A parametrization is proper if $\phi(t)$ is injective for almost all points on $\widetilde{\mathcal{C}}$. In other words, almost every point on $\widetilde{\mathcal{C}}$ is the image of exactly one parameter value (real or complex). For other equivalent definitions of properness, we refer the reader to [SWPD08, Ch. 4], [RSV09]. As stated in [ADT10, Thm. 1], a parametrization is proper if and only if $\deg(\gcd(h_1(s,t),\ldots,h_n(s,t))) = 0$. This leads to an algorithm for checking properness. By applying Lem. 4.3 we get the following:

Lemma 4.5. *There is an algorithm that checks if a parametrization ϕ is proper in worst-case bit complexity $\widetilde{\mathcal{O}}_B(n(d^5 + d^4\tau))$ and in expected bit complexity $\widetilde{\mathcal{O}}_B(n(d^3 + d^2\tau))$.*

Proof. We construct the polynomials $h_i(s,t)$ for all $i \in [n]$ in $\mathcal{O}_B(nd^2\tau)$. Then, we need to check if $\deg(\gcd(h_1(s,t),\ldots,h_n(s,t))) = 0$ [ADT10, Thm. 1]. For the gcd computation, we employ Lem. 4.3 and the result follows. □

If ϕ is a not a proper parametrization, then there always exists a parametrization $\psi \in \mathbb{Z}(t)^n$ and $R(t) \in \mathbb{Z}(t)$ such that $\psi(R(t)) = \phi(t)$ and ψ is proper [SWPD08, Thm. 7.6]. There are various algorithms for obtaining a proper parametrization, e.g., [Sed86a, GRS02, SWPD08, PD06, GC92]. We consider the algorithm in [PD06] for its simplicity; its pseudo-code is in Alg. 4.

Lemma 4.6. *Consider a non-proper parametrization of a curve $\mathcal{C}$, consisting of univariate polynomials of size (d,τ). Alg. 4 computes a proper parametrization of $\mathcal{C}$, involving polynomials of degree at most d and bitsize $\mathcal{O}(d^2 + d\tau)$, in $\widetilde{\mathcal{O}}_B(n(d^5 + d^4\tau))$, in the worst case.*

Proof. The correctness of the algorithm is proved in [PD06]. We analyze its complexity. The algorithm first computes the bivariate polynomials $H_i(s,t) = p_i(s)q_i(t) - p_i(t)q_i(s)$ for $i = 1,\ldots,n$. They have bi-degree at most (d,d) and bitsize at most $2\tau + 1$. Then, we compute their gcd, which we denote by H, in $\widetilde{\mathcal{O}}_B(n(d^5+d^4\tau))$ (Lem. 4.3). By [Mah62] and [BPR06, Prop. 10.12] we have that $\mathcal{L}(H) = \mathcal{O}(d+\tau)$. If we write $H = C_m(t)s^m + \cdots + C_0(t)$, it also holds that $\mathcal{L}(C_j) = \mathcal{O}(d+\tau)$, $j = 1,\ldots,m$.

If the degree of H is one, then the parametrization is already proper and we have nothing to do. Otherwise, we consider H as a univariate polynomial in s and we find two

Algorithm 4: `Make_Proper`(ϕ)

Input: A parametrization $\phi \in \mathbb{Z}(t)^n$ as in Eq. (4.1)
Output: A proper parametrization $\psi = (\psi_1, \ldots, \psi_n) \in \mathbb{Z}(t)^n$

1 **for** $i \in [n]$ **do** $H_i(s,t) \leftarrow p_i(s)q_i(t) - p_i(t)q_i(s) \in \mathbb{Z}[s,t]$;
2 $H \leftarrow \gcd(H_1, \ldots, H_n) = C_m(t)s^m + \cdots + C_0(t) \in (\mathbb{Z}[t])[s]$
3 **if** $m = 1$ **then** RETURN $\phi(t)$;
4 Find $k, l \in [m]$ such that:
$\deg(\gcd(C_k(t), C_l(t))) = 0$ and $\frac{C_k(t)}{C_l(t)} \notin \mathbb{Q}$
5 $R(t) \leftarrow \frac{C_k(t)}{C_l(t)}$
6 $r \leftarrow \deg(R) = \max\{\deg(C_k), \deg(C_l)\}$
7 $G \leftarrow s\, C_l(t) - C_k(t)$
8 **for** $i \in [n]$ **do**
9 $\quad F_i \leftarrow xq_i(t) - p_i(t)$
10 $\quad L_i(s,x) \leftarrow \mathtt{res}_t(F_i(t,x), G(t,s)) = (\tilde{q}_i(s)x - \tilde{p}_i(s))^r$
11 **end**
12 RETURN $\psi(t) = \left(\frac{\tilde{p}_1(t)}{\tilde{q}_1(t)}, \ldots, \frac{\tilde{p}_n(t)}{\tilde{q}_n(t)}\right)$

of its coefficients that are relatively prime, using exact division. The complexity of this operation is $m^2 \times \widetilde{\mathcal{O}}_B(d^2 + d\tau) = \widetilde{\mathcal{O}}_B(d^4 + d^3\tau)$ [vzGG13, Ex. 10.21].

Subsequently, we perform n resultant computations to get $L_1, \ldots L_n$, as defined in Alg. 4. From these we obtain the rational functions of the new parametrization. We focus on the computation of L_1. The same arguments hold for all L_i. The bi-degree of $L_1(s,x)$ is (d,d) [BPR06, Prop. 8.49] and $\mathcal{L}(L_1) = \mathcal{O}(d^2 + d\tau)$ [BPR06, Prop. 8.50]; the latter dictates the bitsize of the new parametrization.

To compute L_1, we consider F_1 and G as univariate polynomials in t and we apply a fast algorithm for computing the univariate resultant based on subresultants [LR01]; it performs $\widetilde{\mathcal{O}}(d)$ operations. Each operation consists of multiplying bivariate polynomials of bi-degree (d,d) and bitsize $\mathcal{O}(d^2 + d\tau)$ so it costs $\widetilde{\mathcal{O}}_B(d^4 + d^3\tau)$. We compute the resultant in $\widetilde{\mathcal{O}}_B(d^5 + d^4\tau)$. We multiply the latter bound by n to conclude the proof. □

4.3.2 Normal parametrization

Normality of the parametrization concerns the surjectivity of the map ϕ. The parametrization $\phi(t)$ is $\mathbb{R}$-normal if for all points $\mathbf{p}$ on $\mathcal{C}$ there exists $t_0 \in \mathbb{R}$ such that $\phi(t_0) = \mathbf{p}$. When the parametrization is not $\mathbb{R}$-normal, the points that are not in the image of ϕ for $t \in \mathbb{R}$ are $\mathbf{p}_\infty$ (if it exists) and the isolated points that we obtain for complex values of t [AR07, Prop. 4.2]. An $\mathbb{R}$-normal reparametrization does not always exist. We refer to [SWPD08, Sec. 7.3] for further details. However, if $\mathbf{p}_\infty$ exists, then we reparametrize the curve to avoid possible singularities at infinity. The point $\mathbf{p}_\infty$ exists if $\deg(p_i) \leq \deg(q_i)$, for all $i \in [n]$.

Lemma 4.7. *If $\mathbf{p}_\infty$ exists, then we can reparametrize the curve using a linear rational function to ensure that $\mathbf{p}_\infty$ is not a singular point, using a Las Vegas algorithm in expected time $\widetilde{\mathcal{O}}_B(n(d^2+d\tau))$. The new parametrization involves polynomials of size $(d, \widetilde{\mathcal{O}}(d+\tau))$.*

Proof. The point at infinity depends on the parametrization. So, for this proof, let us denote the point at infinity of ϕ by $\mathbf{p}_\infty^\phi$. This point is obtained for $t \to \infty$.

The reparametrization consists of choosing $t_0 \in \mathbb{R}$ and applying the map $r : t \mapsto \frac{t_0\, t+1}{t-t_0}$ to ϕ, to obtain a new parametrization, $\psi = \phi \circ r$. The point at infinity of the new parametrization is $\mathbf{p}_\infty^\psi = \phi(t_0)$. We need to ensure that $\mathbf{p}_\infty^\psi = \phi(t_0)$ is not singular. There are $\leq d^2$ singular points, so we choose t_0 uniformly at random from the set $\{1, \ldots, Kd^2\}$ where $K \geq 2$. Then, with probability $\geq 1/2$, $\phi(t_0)$ is not singular and $\mathbf{p}_\infty^\psi$ is also not singular. The bound on the possible values of t_0 implies that the bitsize of t_0 is $\mathcal{O}(\lg(d))$.

We compute the new parametrization, ψ, in $\widetilde{\mathcal{O}}_B(n(d^2+d\tau))$ using multipoint evaluation and interpolation, by exploiting the fact that the polynomials in ψ have degrees at most d and bitsize $\widetilde{\mathcal{O}}(d+\tau)$.

For a Las Vegas algorithm we need to check if $\phi(t_0)$ is a cusp or a multiple point. For the former, we evaluate ϕ' at t_0 (see Rem. 4.4). This costs $\widetilde{\mathcal{O}}_B(nd\tau)$ [BLPR13, Lem. 3]. For the latter, we check if $\deg(\gcd(\phi_1(t_0)q_1(t) - p_1(t), \ldots, \phi_1(t_0)q_1(t) - p_1(t))) = 0$ in $\widetilde{\mathcal{O}}_B(n(d^2+d\tau))$ (Lem. 4.2). If $\phi'(t_0)$ is not the zero vector and the degree of the gcd is zero, then $\phi(t_0)$ is not singular. □

Remark 4.8. *Since the reparametrizing function in the previous lemma is linear, it does not affect properness [SWPD08, Thm. 6.3].*

4.4 Special points on the curve

We consider a parametrization ϕ of $\mathcal{C}$ as in Eq. (4.1), such that ϕ is proper and there are no singularities at infinity. We highlight the necessity of these assumptions when needed. We detect the *parameters* that generate the *special points* of $\mathcal{C}$, namely the singular, the isolated, and the extreme points (in the coordinate directions). We identify the values of the parameter for which ϕ is not defined, namely the poles (see Def. 4.9). In presence of poles, $\mathcal{C}$ consists of multiple components.

Definition 4.9. *The parameters for which $\phi(t)$ is not defined are the poles of ϕ. The sets of poles over the complex and the reals are:*

$$\mathtt{T}_P^{\mathbb{C}} = \{t \in \mathbb{C} : \prod_{i\in[n]} q_i(t) = 0\} \text{ and } \mathtt{T}_P^{\mathbb{R}} = \mathtt{T}_P^{\mathbb{C}} \cap \mathbb{R}, \text{ respectively.}$$

We consider the solution set S of the system of Eq.(4.2) over $\mathbb{C}^2$:

$$S = \{(t,s) \in \mathbb{C}^2 : h_i(t,s) = 0 \text{ for all } i \in [n]\}.$$

Remark 4.10. *Notice that when ϕ is in reduced form, if $(s,t) \in S$ and $(s,t) \in (\mathbb{C} \setminus \mathtt{T}_P^{\mathbb{C}}) \times \mathbb{C}$, then also $t \notin \mathtt{T}_P^{\mathbb{C}}$ [RSV09, (in the proof of) Lem. 9].*

Next, we present some well-known results [RSV09, SWPD08] that we adapt to our notation.

Singular points. Quoting [MC92], "Algebraically, singular points are points on the curve, in whose neighborhood the curve cannot be represented as an one-to-one and C^∞ bijective map with an open interval on the real line". Geometrically, singularities correspond to shape features that are known as cusps and self-intersections of smooth branches. *Cusps* are points on the curve where the tangent vector is the zero vector. This is a necessary and sufficient condition when the parametrization is proper [MC92]. Self-intersections are *multiple* points, i.e., points on $\mathcal{C}$ with more than one preimages.

Lemma 4.11. *The set of parameters corresponding to real cusps is*

$$\mathtt{T}_C = \left\{ t \in \mathbb{R} \setminus \mathtt{T}_P^{\mathbb{R}} : (t,t) \in S \right\}.$$

The set of parameters corresponding to real multiple points is

$$\mathtt{T}_M = \{ t \in \mathbb{R} \setminus \mathtt{T}_P^{\mathbb{R}} : \exists s \neq t, s \in \mathbb{R} \text{ such that } (t,s) \in S \}.$$

Proof. The description of $\mathtt{T}_C$ is an immediate consequence of Rem. 4.4. It states that $h_i(t,t) = \phi_i'(t) q_i^2(t)$, for $i \in [n]$.

Now let $\mathbf{p} = \phi(t)$ be a multiple point on $\mathcal{C}$. Then, there is $s \in \mathbb{R} \setminus \mathtt{T}_P^{\mathbb{R}}$ with $\phi(t) = \phi(s) \Rightarrow h_i(t,s) = 0$ for all $i \in [n]$ and so $t \in \mathtt{T}_M$. Conversely, let $t \in \mathtt{T}_M$ and $s \neq t$, $s \in \mathbb{R}$ such that $h_i(t,s) = 0$ for all $i \in [n]$. From [RSV09, (in the proof of) Lem. 9], when ϕ is in reduced form, if $(t,s) \in S$ and $(t,s) \in (\mathbb{R} \setminus \mathtt{T}_P^{\mathbb{R}}) \times \mathbb{R}$, then also $s \notin \mathtt{T}_P^{\mathbb{R}}$. So, $h_i(t,s) = 0 \Leftrightarrow \frac{p_i(t)}{q_i(t)} = \frac{p_i(s)}{q_i(s)}$ for all $i \in [n]$, and thus $\mathbf{p} = \phi(t) = \phi(s)$ is a real multiple point. □

Notice that $\mathtt{T}_C$ and $\mathtt{T}_M$ are not necessarily disjoint, for we may have both cusps and smooth branches that intersect at the same point.

Isolated points. An isolated point on a real curve can only occur for complex values of the parameter. The point at infinity is not isolated because it is the limit of a sequence of real points. So, additional care is needed in order to avoid cases where the point at infinity is obtained also for complex values of the parameter.

Lemma 4.12. *The set of parameters generating isolated points of $\mathcal{C}$ is*

$$\mathtt{T}_I = \{ t \in \mathbb{C} \setminus (\mathbb{R} \cup \mathtt{T}_P^{\mathbb{C}}) : (t,\bar{t}) \in S \text{ and } \nexists s \in \mathbb{R} \text{ s.t. } (t,s) \in S \text{ and } \phi(t) \neq \lim_{s \to \infty} \phi(s) \}.$$

Proof. Let $\mathbf{p} = \phi(t) \in \mathbb{R}^n$ be an isolated point, where $t \in \mathbb{C} \setminus (\mathbb{R} \cup \mathtt{T}_P^{\mathbb{C}})$. Notice that $\mathbf{p}$ is also a multiple point, since it holds that $\phi_i(t) = \overline{\phi_i(t)} = \phi_i(\bar{t})$ for $i \in [n]$. Thus, $h_i(t, \bar{t}) = 0$ for all $i \in [n]$ and $(t, \bar{t}) \in S$. Moreover, since $\mathbf{p}$ is isolated, there are no real branches through $\mathbf{p}$ and there does not exist $s \in \mathbb{R}$ such that $\phi(t) = \phi(s) \Rightarrow h_i(t, s) = 0$, for all $i \in [n]$. So, $t \in \mathtt{T}_I$.

Conversely, let $(t, \bar{t}) \in S$ with $t \in \mathbb{C} \setminus \mathbb{R} \cup \mathtt{T}_P^{\mathbb{C}}$. Since ϕ is in reduced form, we have that $\bar{t} \notin P^{\mathbb{C}}$[RSV09, (in the proof of) Lem. 9], therefore $h_i(t, \bar{t}) = 0$, for all $i \in [n]$, implies that $\phi(t) = \phi(\bar{t}) = \overline{\phi(t)} \in \mathbb{R}^n$. Since there does not exist $s \in \mathbb{R}$ with $\phi(t) = \phi(s)$, $\mathbf{p}$ is an isolated point on $\mathcal{C}$. □

Extreme points. Consider a vector $\vec{\delta}$ and a point on $\mathcal{C}$ whose normal vector is parallel to $\vec{\delta}$. If the point is not singular, then it is an extreme point of $\mathcal{C}$ with respect to $\vec{\delta}$. We compute the extreme points with respect to the direction of each coordinate axis. Rem. 4.4 leads to the following lemma:

Lemma 4.13. *The set of parameters generating extreme points in the coordinate directions is*

$$\mathtt{T}_E = \left\{ t \in \mathbb{R} \setminus \mathtt{T}_P^{\mathbb{R}} : \prod_{i \in [n]} h_i(t, t) = 0 \text{ and } t \notin \mathtt{T}_C \cup \mathtt{T}_M \right\}.$$

4.4.1 Computation and Complexity

From Lemmata 4.11, 4.12, and 4.13, it follows that given a proper parametrization ϕ without singular points at infinity, we can easily find the poles and the set of parameters generating cusps, multiple, extreme, and isolated points. We do so by solving an overdetermined bivariate polynomial system and univariate polynomial equations. Then, we classify the parameters that appear in the solutions, by exploiting the fact the system is symmetric. For sake of completeness, we describe the procedure in Alg. 5.

To compute the Rational Univariate Representation (RUR) [Rou99] of an overdetermined bivariate system (Thm. 4.16), we employ Lem. 4.14 and Prop. 4.15, which adapt the techniques used in [BLM+16] to our setting.

Lemma 4.14. *Let $f, g, h_1, \ldots, h_n \in \mathbb{Z}[X, Y]$ with degrees bounded by δ and bitsize of coefficients bounded by L. Computing a common separating element in the form $X + \alpha Y, \alpha \in \mathbb{Z}$ for the $n+1$ systems of bivariate polynomial equations $\{f = g = 0\}$, $\{f = h_i = 0\}$, $i = 1 \ldots n$ needs $\widetilde{\mathcal{O}}_B(n(\delta^6 + \delta^5 L))$ bit operations in the worst case, and $\widetilde{\mathcal{O}}_B(n(\delta^5 + \delta^4 L))$ in the expected case with a Las Vegas Algorithm. Moreover, the bitsize of α does not exceed* $\log(2n\delta^4)$.

Proof. A straightforward strategy consists of simultaneously running Algorithm 5 (worst case) or Algorithm 5' (Las Vegas) from [BLM+16] on all the systems. The only modifications needed are that the values of α to be considered are less than $2n\delta^4$ (twice a bound

Algorithm 5: `Special_Points`(ϕ)

Input: Proper parametrization $\phi \in \mathbb{Z}(t)^n$ without singularity at infinity, as in Eq. (4.1)

Output: Real poles and parameters that give real cusps, multiple, isolated and extreme points with respect to the direction the coordinate axes.

```
/* The subroutines SOLVE_R and SOLVE_C return the solution set of a
   univariate polynomial or a system of polynomials over the real and
   complex numbers resp.                                            */
```

Compute polynomials $h_1(s,t), \ldots, h_n(s,t)$
$\mathtt{T}_P^{\mathbb{R}} \leftarrow \bigcup_{i\in n} \mathtt{SOLVE_R}(q_i(t)=0)$
$\mathtt{T}_P^{\mathbb{C}} \leftarrow \bigcup_{i\in [n]} \mathtt{SOLVE_C}(q_i(t)=0)$
$S \leftarrow \mathtt{SOLVE_C}(h_1(s,t)=0, \ldots, h_n(s,t)=0)$
$\mathtt{T}_C, \mathtt{T}_M, \mathtt{T}_I, W \leftarrow \emptyset$
for $(s,t) \in S$ **do**
  if $s = t$ *and* $s \in \mathbb{R} \setminus \mathtt{T}_P^{\mathbb{R}}$ **then**
    $\mathtt{T}_C \leftarrow \mathtt{T}_C \cup \{t\}$
  end
  else if $s \neq t$ **then**
    if $s \in \mathbb{R} \setminus \mathtt{T}_P^{\mathbb{R}}$ **then**
      if $t \in \mathbb{R}$ **then**
        $\mathtt{T}_M \leftarrow \mathtt{T}_M \cup \{t\}$
      end
      else
        $W \leftarrow W \cup \{t\}$
      end
    end
    else if $s = \bar{t}$ *and* $s \notin \mathtt{T}_P^{\mathbb{C}}$ **then**
      $\mathtt{T}_I \leftarrow \mathtt{T}_I \cup \{t\}$
    end
  end
end
$\mathtt{T}_I \leftarrow \mathtt{T}_I \setminus W$
```
/* Extreme points                                                   */
```
$\mathtt{T}_E \leftarrow \bigcup_{i\in n} \mathtt{SOLVE_R}(h_i(t,t)=0)$
$\mathtt{T}_E \leftarrow \mathtt{T}_E \setminus (T_E \cap (\mathtt{T}_C \cup \mathtt{T}_M))$

on the total number of solutions of all the systems) and that the exit test is valid if and only if it is valid for all the systems. □

Proposition 4.15. *Let $f, g \in \mathbb{Z}[X,Y]$ with degrees bounded by δ and coefficients' bitsizes bounded by L. We can compute a rational parametrization $\{h(T), X = \frac{h_X(T)}{h_1(T)}, Y = \frac{h_Y(T)}{h_1(T)}\}$ of f, g with $h, h_1, h_X, h_Y \in \mathbb{Z}[T]$ with degrees less than δ^2 and coefficients' bitsizes in $\widetilde{\mathcal{O}}(\delta(L+\delta))$, in $\widetilde{\mathcal{O}}_B(\delta^5(L+\delta))$ bit operations in the worst case and $\widetilde{\mathcal{O}}_B(\delta^4(L+\delta))$ expected bit operations with a Las Vegas Algorithm.*

Proof. Algorithms 6 and 6' from [BLM+16] compute an RUR decomposition of $f = g = 0$ in $\widetilde{\mathcal{O}}_B(\delta^5(L+\delta))$ bit operations in the worst case and $\widetilde{\mathcal{O}}_B(\delta^4(L+\delta))$ expected bit operations with a Las Vegas Algorithm respectively. They provide $s \leqslant \delta$ parametrizations in the form $\{h_i(T), \frac{h_{i,X}(T)}{h_{i,1}(T)}, \frac{h_{i,Y}(T)}{h_{i,1}(T)}\}$, where $i = 1, \ldots, s$, with the following properties:

- $\prod_{i=1}^{s} h_i$ is a polynomial of degree at most δ^2 with coefficients of bitsize $\widetilde{\mathcal{O}}(\delta L + \delta^2)$.
- The degrees of $h_{i,1}(T), h_{X,1}(T)$ and $h_{Y,1}(T)$ are less then the degree of h_i.
- The coefficients' bitsizes of $h_{i,1}(T), h_{X,1}(T)$ and $h_{Y,1}(T)$ are in $\widetilde{\mathcal{O}}_B(\delta L + \delta^2)$.

Also,

$$\prod_{i=1}^{s} h_i, \frac{\sum_{n=1}^{n} h_{j,X} \prod_{i\neq j} h_i}{\sum_{n=1}^{n} h_{j,1} \prod_{i\neq j} h_i}, \frac{\sum_{n=1}^{n} h_{j,Y} \prod_{i\neq j} h_i}{\sum_{n=1}^{n} h_{j,1} \prod_{i\neq j} h_i}$$

is a rational parametrization of the system $\{f = g = 0\}$, defined by polynomials of degree less than δ^2 with coefficients of bitsizes $\widetilde{\mathcal{O}}(\delta(L + \delta))$ and can be computed from the RUR decomposition performing $\mathcal{O}(s)$ multiplications of polynomials of degree at most δ^2 with coefficients of bitsize $\widetilde{\mathcal{O}}(\delta(L + \delta))$, which requires $\widetilde{\mathcal{O}}_B(\delta^4(L + \delta))$ bit operations. □

Theorem 4.16. *There exists an algorithm that computes the RUR and the isolating boxes of the roots of the system $\{h_1(s,t) = \cdots = h_n(s,t) = 0\}$ with worst-case bit complexity $\widetilde{\mathcal{O}}_B(n(d^6 + d^5\tau))$. There is also a Las Vegas variant with expected complexity $\widetilde{\mathcal{O}}_B(d^6 + nd^5 + d^5\tau + nd^4\tau)$.*

Proof. Assume that we know a common separating linear element $\ell(s,t) = \ell_0+\ell_1 s+\ell_2 t$ that separates the roots of the $n-1$ systems of bivariate polynomial equations $\{h_1 = h_2 = 0\}$, $\{h_1 = h_i = 0\}$, for $3 \leq i \leq n$. We can compute ℓ with $\widetilde{\mathcal{O}}_B(n(d^6 + d^5\tau))$ bit operations in the worst case and with $\widetilde{\mathcal{O}}_B(n(d^5 + d^4\tau))$ expected bit operations with a Las Vegas algorithm (Lem. 4.14).

We denote an RUR for $\{h_1 = h_2 = 0\}$ with respect to ℓ by $\{r(T), \frac{r_s(T)}{r_I(T)}, \frac{r_t(T)}{r_I(T)}\}$. In addition, for $i = 3 \ldots n$, let $\{r_i(T), \frac{r_{i,s}(T)}{r_{i,I}(T)}, \frac{r_{i,t}(T)}{r_{i,I}(T)}\}$ be the RUR of $\{h_1 = h_i = 0\}$, also with respect to ℓ. We compute these representations for all $i = 3 \ldots n$ with $\widetilde{\mathcal{O}}_B(n(d^6+d^5\tau))$ bit operations in the worst case, and with $\widetilde{\mathcal{O}}_B(n(d^5+d^4\tau))$ in expected case with a Las Vegas algorithm (Lem. 4.15).

Then, for the system $\{h_1 = h_2 = \ldots = h_n = 0\}$ we can define a rational parametrization $\{\chi(T), \frac{r_s(T)}{r_I(T)}, \frac{r_t(T)}{r_I(T)}\}$, where

$$\begin{aligned}\chi(T) = \gcd(\quad & r(T), r_3(T), \ldots, r_n(T),\\ & r_s(T)r_{3,I}(T) - r_{3,s}(T)r_I(T), r_t(T)r_{3,I}(T) - r_{3,t}(T)r_I(T),\\ & \quad\vdots\\ & r_s(T)r_{n,I}(T) - r_{n,s}(T)r_I(T), r_t(T)r_{n,I}(T) - r_{n,t}(T)r_I(T)).\end{aligned}$$

So to compute such a parametrization, we still need to compute the gcd of $3n-5$ univariate polynomials of degrees at most d^2 and coefficients of bitsizes in $\widetilde{\mathcal{O}}(d\tau)$ which needs $\widetilde{\mathcal{O}}_B(n(d^6+d^4\tau))$ bit operations in the worst case. Isolating the roots of such a parametrization requires $\widetilde{\mathcal{O}}_B(d^6+d^5\tau)$ according to Alg. 7 from [BLM+16]. □

Remark 4.17 (RUR and isolating interval representation). *If we use Thm.4.16 to solve the over-determined bivariate system of the h_i polynomials of Eq. (4.2), then we obtain in the output an RUR for the roots, which is as follows: There is a polynomial $\chi(T)\in\mathbb{Z}[T]$ of size $(\mathcal{O}(d^2),\widetilde{\mathcal{O}}(d^2+d\tau))$ and a mapping:*

$$\begin{aligned} V(\chi) &\to V(h_1,\dots,h_n)\\ T &\mapsto \left(\frac{r_s(T)}{r_I(T)},\frac{r_t(T)}{r_I(T)}\right), \end{aligned} \tag{4.3}$$

that defines an one-to-one correspondence between the roots of χ and those of the system. The polynomials r_s, r_t, and r_I are in $\mathbb{Z}[T]$ and have also size $(\mathcal{O}(d^2),\widetilde{\mathcal{O}}(d^2+d\tau))$.

Taking into account the cost to compute this parametrization of the solutions (Thm.4.16), we can also compute the resultant of $\{h_1,h_2\}$ with respect to s or t at no extra cost. Notice that both resultants are the same polynomial, since the system is symmetric. Let $R_s(t)=\mathtt{res}_s(h_1,h_2)$. It is of size $(\mathcal{O}(d^2),\mathcal{O}(d^2+d\tau))$ [BPR06, Prop. 8.46].

Under the same bit complexity, we can sufficiently refine the isolating boxes of the solutions of the bivariate system (computed in Thm.4.16), so that every root $(\frac{r_s(\xi)}{r_I(\xi)},\frac{r_t(\xi)}{r_I(\xi)})$, where $\chi(\xi)=0$, has a representation as a pair of algebraic numbers in isolating interval representation:

$$((R_s, I_{1,\xi}\times I_{2,\xi}),(R_s, J_{1,\xi}\times J_{2,\xi})). \tag{4.4}$$

Both coordinates in the latter representation, are algebraic numbers which are roots of the same polynomial. Moreover, $I_{2,\xi}, J_{2,\xi}$ are empty sets when the corresponding algebraic number is real. Therefore, we can immediately distinguish between real and complex parameters. At the same time, we associate to each isolating box of a root of R_s the algebraic numbers $\rho=(\chi, I_\rho\times J_\rho)$ for which it holds that $\frac{r_s(\rho)}{r_I(\rho)}$ projects inside this isolating box. We can interchange between the two representations in constant time and this will simplify our computations in the sequel.

Lemma 4.18. *Let $\mathcal{C}$ be a curve with a proper parametrization $\phi(t)$ as in Eq. (4.1), that has no singularities at infinity. We compute the real poles of ϕ and the parameters corresponding to singular, extreme (in the coordinate directions), and isolated points of $\mathcal{C}$ in worst-case bit complexity*

$$\widetilde{\mathcal{O}}_B(nd^6+nd^5\tau+d^4(n^2+n\tau)+d^3(n^2\tau+n^3)+n^3d^2\tau),$$

and using a Las Vegas algorithm in expected bit complexity

$$\widetilde{\mathcal{O}}_B(d^6 + d^5(n+\tau) + d^4(n^2+n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau).$$

Proof. The proof is an immediate consequence of the following:

• *We compute all $h_i \in \mathbb{Z}[s,t]$ in $\widetilde{\mathcal{O}}_B(nd^2\tau)$*: To construct each h_i we perform d^2 multiplications of numbers of bitsize τ; the cost for this is $\widetilde{\mathcal{O}}_B(d^2\tau)$. The bi-degree of each is at most (d,d) and $\mathcal{L}(h_i) \leq 2\tau + 1 = \mathcal{O}(\tau)$.

• *The real poles of ϕ are computed in $\widetilde{\mathcal{O}}_B(n^2(d^4+d^3\tau))$:* To find the poles of ϕ, we isolate the real roots of each polynomial $q_i(t)$, for $i \in [n]$. This costs $\widetilde{\mathcal{O}}_B(n(d^3+d^2\tau))$ [PT17]. Then we sort the roots in $\widetilde{\mathcal{O}}_B(n\,d\,n(d^3+d^2\tau)) = \widetilde{\mathcal{O}}_B(n^2(d^4+d^3\tau))$.

• *The parameters corresponding to cusps, multiple and isolated points of $\mathcal{C}$ are computed in $\widetilde{\mathcal{O}}_B(n(d^6+d^5\tau))$:*

We solve the bivariate system of Eq. (4.2) in $\widetilde{\mathcal{O}}_B(n(d^6+d^5\tau))$ or in expected time $\widetilde{\mathcal{O}}_B(d^6 + nd^5 + d^5\tau + nd^4\tau)$ (Thm. 4.16). Then we have a parametrization of the solutions of the bivariate system of Eq. (4.2) of the form of Eq. (4.3) and in the same time of the form of Eq. (4.4) (see Rem. 4.17). Some solutions $(s,t) \in S$ may not correspond to points on the curve, since s,t can be poles of ϕ. Notice that from Rem. 4.10, s and t are either both poles or neither of them is a pole. We compute $g_s = \gcd(R_s, Q)$, where $Q(t) = \prod_{i\in[n]} q_i(t)$, and the gcd-free part of R_s with respect to Q. This is done in $\widetilde{\mathcal{O}}_B(\max\{n,d\}\,(nd^3\tau + nd^2\tau^2))$ [BLPR15, Lem. 5].

Every root of R_s^* is an algebraic number of the form $(R_s, I_{1,\xi} \times I_{2,\xi})$, for some ξ that is root of χ. We can easily determine if it corresponds to a cusp, a multiple or an isolated point; when real (i.e., $I_{2,\xi} = \emptyset$) it corresponds to a cusp of $\mathcal{C}$ if and only if $((R_s, I_{1,\xi}), (R_s, I_{1,\xi}))$ is in S. Otherwise, it corresponds to a multiple point. When it is complex (i.e., $I_{2,\xi} \neq \emptyset$), it corresponds to an isolated pont of $\mathcal{C}$ if and only if $((R_s, I_{1,\xi} \times I_{2,\xi}), (R_s, I_{1,\xi} \times (-I_{2,\xi}))) \in S$ and there is no root in S of the form $((R_s, I_{1,\xi} \times I_{2,\xi}), (R_s, J_{1,\xi'}))$.

• *The parameters corresponding to extreme points of $\mathcal{C}$ with respect to the direction of each coordinate axis are computed in $\widetilde{\mathcal{O}}_B(d^4n\tau + d^3(n^2\tau + n^3) + d^2n^3\tau)$:*

For all $i \in [n]$, $h_i(t,t)$ is a univariate polynomial of size $(\mathcal{O}(d), \mathcal{O}(\tau))$. Then, $H(t) = \prod_{i\in[n]} h_i(t,t)$ is of size $(\mathcal{O}(nd), \widetilde{\mathcal{O}}(n\tau))$. The parameters that correspond to the extreme points are among the roots of $H(t)$. To make sure that poles and parameters that give singular points are excluded, we compute $\gcd(H, Q \cdot R_s)$, where $Q(t) = \prod_{i\in[n]} q_i(t)$, and the gcd-free part of H with respect to $Q \cdot R_s$, say H^*. Since $Q \cdot R_s$ is a polynomial of size $(d^2 + nd, (d+n)\tau)$, the computation of the gcd and the gcd-free part costs $\widetilde{\mathcal{O}}_B(n(d^4\tau + nd^3\tau + n^2d^2\tau))$ [BLPR15, Lem. 5]. Then, $H = \gcd(H, Q \cdot R_s)H^*$, and the real roots of H^* give the parameters that correspond to the extreme points. We isolate the real roots of H^* in $\widetilde{\mathcal{O}}_B(n^3(d^3+d^2\tau))$, since it is a polynomial of size $(\mathcal{O}(nd), \widetilde{\mathcal{O}}(n(d+\tau)))$. □

4.5 PTOPO: Topology and Complexity

We present PTOPO, an algorithm to construct an abstract graph G that is isotopic [BT06, p.184] to $\mathcal{C}$ when we embed it in $\mathbb{R}^n$. We emphasize that, currently, we do not identify knots in the case of space curves. The embedding consists of a graph whose vertices are points on the curve given by their parameter values. The edges are smooth parametric arcs that we can continuously deform to branches of $\mathcal{C}$ without any topological changes. We need to specify a bounding box in $\mathbb{R}^n$ inside which the constructed graph results in an isotopic embedding to $\mathcal{C}$. We comment at the end of the section on the case where an arbitrary box is provided at the input. We determine a bounding box in $\mathbb{R}^n$, which we call *characteristic*, that captures all the topological information of $\mathcal{C}$.

Definition 4.19. *A characteristic box of $\mathcal{C}$ is a box enclosing a subset of $\mathbb{R}^n$ that intersects all components of $\mathcal{C}$ and contains all its singular, extreme (in the coordinate directions), and isolated points.*

Let $\mathcal{B}_\mathcal{C}$ be a characteristic box of $\mathcal{C}$. If $\mathcal{C}$ is bounded, then $\mathcal{C} \subset \mathcal{B}_\mathcal{C}$. If $\mathcal{C}$ is unbounded, then the branches of $\mathcal{C}$ that extend to infinity intersect the boundary of $\mathcal{B}_\mathcal{C}$. A branch of the curve extends to infinity if for $t \to t_0$, it holds $||\phi(t)|| > M$, for any $M > 0$, where $t_0 \in \mathbb{R} \cup \{\infty\}$. Lem. 4.20 computes a characteristic box using the degree and bitsize of the polynomials in the parametrization of Eq. (4.1).

Lemma 4.20. *Let $\mathcal{C}$ be a curve with a parametrization as in Eq. (4.1). For $b = 15d^2(\tau + \log d) = \mathcal{O}(d^2\tau)$, $\mathcal{B}_\mathcal{C} = [-2^b, 2^b]^n$ is a characteristic box of $\mathcal{C}$.*

Proof. We estimate the maximum and minimum values of ϕ_i, $i \in [n]$, when we evaluate it at the parameter values that correspond to special points and also at each pole that is not a root of q_i.

Let t_0 be a parameter that corresponds to a cusp or an extreme point with respect to the i-th direction. Then, it is a root of $\phi_i'(t)$. Let $N(t) = p_i'(t)q_i(t) - p_i(t)q_i'(t)$ the numerator of $\phi_i'(t)$. Then $N(t_0) = 0$. The degree of $N(t)$ is $\leq 2d - 1$ and $\mathcal{L}(N) \leq 2^{2\tau + \log d + 1}$. From Lem. 4.1 we conclude that $|p_i(t_0)| \leq 2^{4d\tau + d\log(d) + (3d-1)\log(3d-1) + d - \tau}$. Analogously, it holds that $|q_i(t_0)| \geq 2^{-4d\tau - d\log(d) - (3d-1)\log(3d-1) - d + \tau}$. Therefore,

$$|\phi_i(t_0)| \leq 2^{2(4d\tau + d\log(d) + (3d-1)\log(3d-1) + d - \tau)}.$$

Now, let (t_1, t_2) be two parameters corresponding to a multiple point of $\mathcal{C}$, i.e., (t_1, t_2) is a root of the bivariate system in Eq. (4.2). Take any $j, k \in [n]$ with $j \neq k$ and let $R(t) = \mathtt{res}_s(h_j, h_k)$. It holds that $R(t_1) = 0$. The degree of R is $\leq 2d^2$ and $\mathcal{L}(R) \leq 2d(\tau + \log(d) + \log(d+1) + 1)$ [BPR06, Prop. 8.29]. By applying Lem. 4.1, we deduce that

$$|\phi_i(t_1)| \leq 2^{4d^2(\tau + \log(d) + \log(d+1) + 1) + 4d^2\tau + (2d^2 + d)\log(2d^2 + d)}.$$

Let t_3 be a pole of ϕ with $q_j(t_3) = 0$, for some $j \neq i$. If $\phi_i(t_3)$ is defined, applying Lem. 4.1 gives

$$|\phi_i(t_3)| \leq 2^{4d\tau + 4d \log 2d}.$$

To conclude, we take the maximum of the three bounds. However, to simplify notation, we slightly overestimate the latter bound. □

The vertices of the embedded graph must include the singular and the isolated points of $\mathcal{C}$. Additionally, to rigorously visualize the geometry of $\mathcal{C}$, we consider as vertices the extreme points of $\mathcal{C}$, with respect to all coordinate directions, as well as the intersections of $\mathcal{C}$ with the boundary of the bounding box. We label the vertices of G using the corresponding parameter values generating these points, and we connect them accordingly. Alg. 6 presents the pseudo-code of `PTOPO` and here we give some more details on the various steps.

We construct G as follows: First, we compute the poles and the sets $\mathtt{T}_C, \mathtt{T}_M, \mathtt{T}_E$, and $\mathtt{T}_I$ of parameters corresponding to "special points". Then, we compute the characteristic box of $\mathcal{C}$, say $\mathcal{B}_\mathcal{C}$. We compute the set $\mathtt{T}_B$ of parameters corresponding to the intersections of $\mathcal{C}$ with the boundary of $\mathcal{B}_\mathcal{C}$ (if any). Lem. 4.21 describes this procedure and its complexity.

Lemma 4.21. *Let $\mathcal{B} = [l_1, r_1] \times \cdots \times [l_n, r_n]$ in $\mathbb{R}^n$ and $\mathcal{L}(l_i) = \mathcal{L}(r_i) = \sigma$, for $i \in [n]$. We can find the parameters that give the intersection points of ϕ with the boundary of $\mathcal{B}$ in $\widetilde{\mathcal{O}}_B(n^2d^3 + n^2d^2(\tau + \sigma))$.*

Proof. For each $i \in [n]$ the polynomials $q_i(t)l_i - p_i(t) = 0$ and $q_i(t)r_i - p_i(t) = 0$ are of size $(\mathcal{O}(d), \mathcal{O}(\tau + \sigma))$. So, we compute isolating intervals for all their real solutions in $\widetilde{\mathcal{O}}_B(d^2(\tau + \sigma))$ [Pan02]. For any root t_0 of each of these polynomials, since ϕ is in reduced form (by assumption), we have $t_0 \notin \mathtt{T}_P^{\mathbb{R}}$. We check if $\phi_j(t_0) \in [l_j, r_j]$, $j \in [n] \setminus i$. This requires 3 sign evaluations of univariate polynomials of size $(d, \tau + \sigma)$ at all roots of a polynomial of size $(d, \tau + \sigma)$. The bit complexity of performing these operations for all the roots is $\widetilde{\mathcal{O}}_B(d^3 + d^2(\tau + \sigma))$ [ST19, Prop. 6]. Since we repeat this procedure $n - 1$ times for every $i \in [n]$, the total cost is $\widetilde{\mathcal{O}}_B(n^2d^3 + n^2d^2(\tau + \sigma))$. □

We partition $\mathtt{T}_C \cup \mathtt{T}_M \cup \mathtt{T}_E \cup \mathtt{T}_I \cup \mathtt{T}_B$ into groups of parameters that correspond to the same point on $\mathcal{C}$. For each group, we add a vertex to G if and only if the corresponding point is strictly inside the bounding box $\mathcal{B}$; for the characteristic box it is strictly inside by construction.

Lemma 4.22. *The graph G has $\kappa = \mathcal{O}(d^2 + nd)$ vertices, which can be computed using $\mathcal{O}(d^2 + nd)$ arithmetic operations.*

Proof. Since $\mathtt{T}_B \cap \mathtt{T}_M = \emptyset$ and $\mathtt{T}_E \cap \mathtt{T}_M = \emptyset$, to each parameter in $\mathtt{T}_B$ and $\mathtt{T}_E$ corresponds a unique point on $\mathcal{C}$. So for every $t \in \mathtt{T}_B \cup \mathtt{T}_E$ we add a vertex to G, labeled by the respective parameter. Next, we group the parameters in $\mathtt{T}_C \cup \mathtt{T}_M \cup \mathtt{T}_I$ that give the same point on $\mathcal{C}$ and we add for each group a vertex at G labeled by the corresponding parameter values.

Grouping the parameters is done as follows: For every $t \in \mathtt{T}_C \cup \mathtt{T}_M$ we add a vertex to G labeled by the set $\{s \in \mathbb{R} : (s,t) \in S\} \cup \{t\}$ and for every $t \in \mathtt{T}_I$ we add a vertex to G labeled by the set $\{s \in \mathbb{C} : (s,t) \in S\} \cup \{t\}$. We compute these sets simply by reading the elements of S.

It holds that $\mathtt{T}_B = \mathcal{O}(nd)$, $\mathtt{T}_E = \mathcal{O}(nd)$ and $|S| = \mathcal{O}(d^2)$. Since for each vertex, we can find the parameters that give the same point in constant time, the result follows. □

We denote by $v_1, \ldots, v_\kappa$ the vertices (with distinct labels) of G and by $\lambda(v_1), \ldots, \lambda(v_\kappa)$ their label sets (i.e., the parameters that correspond to each vertex). Let $\mathtt{T}$ be the *sorted* list of parameters in $\mathtt{T}_C \cup \mathtt{T}_M \cup \mathtt{T}_E \cup \mathtt{T}_B$ (notice that we exclude the parameters of the isolated points). If for two consecutive elements $t_1 < t_2$ in $\mathtt{T}$, there exists a pole $s \in \mathtt{T}_P^{\mathbb{R}}$ such that $t_1 < s < t_2$, then we split $\mathtt{T}$ into two lists: $\mathtt{T}_1$ containing the elements $\leq t_1$ and $\mathtt{T}_2$ containing the elements $\geq t_2$. We continue recursively for $\mathtt{T}_1$ and $\mathtt{T}_2$, until there are no poles between any two elements of the resulting list. This procedure partitions $\mathtt{T}$ into $\mathtt{T}_1, \ldots, \mathtt{T}_\ell$.

To add edges to G, we consider each $\mathtt{T}_i$ with more than one element, where $i \in [\ell]$, independently. For any consecutive elements $t_1 < t_2$ in $\mathtt{T}_i$, with $t_1 \in \lambda(v_{i,1})$ and $t_2 \in \lambda(v_{i,2})$, we add the edge $\{v_{i,1}, v_{i,2}\}$. To avoid multiple edges, we make the convention that we add an edge between $v_{i,j}$, $j = 1, 2$, and an (artificial) intermediate point corresponding to a parameter in (t_1, t_2). If $\mathbf{p}_\infty$ exists, we add an edge to the graph connecting the vertices corresponding to the last element of $\mathtt{T}_\ell$ and the first element of the $\mathtt{T}_1$.

Algorithm 6: $\mathtt{PTOPO}(\phi)$ (Inside the characteristic box)

Input: A proper parametrization $\phi \in \mathbb{Z}(t)^n$ without singular points at infinity.
Output: Abstract graph G

Compute real poles $\mathtt{T}_P^{\mathbb{R}}$.
Compute parameters of 'special points' $\mathtt{T}_C, \mathtt{T}_M, \mathtt{T}_E, \mathtt{T}_I$.
/* Characteristic box */
$b \leftarrow 15d^2(\tau + \log d)$, $\mathcal{B}_\mathcal{C} \leftarrow [-2^b, 2^b]^n$
$\mathtt{T}_B \leftarrow$ parameters that give to intersections of $\mathcal{C}$ with $\mathcal{B}_\mathcal{C}$
Construct the set of vertices of G using Lem.4.22
Sort the list of all the parameters $\mathtt{T} = [\mathtt{T}_C, \mathtt{T}_M, \mathtt{T}_E, \mathtt{T}_B]$.
Let $T_1, \ldots, T_\ell$ the sublists of $\mathtt{T}$ when split at parameters in $\mathtt{T}_P^{\mathbb{R}}$
for *every list* $\mathtt{T}_i = [t_{i,1}, \ldots, t_{i,k_i}]$ **do**
  for $j = 1, \ldots, k_i - 1$ **do**
    Add the edge $\{t_{i,j}, t_{i,j+1}\}$ to the graph
  end
end
if $\mathbf{p}_\infty$ *exists* **then**
  Add the edge $\{t_{1,1}, t_{\ell,k_\ell}\}$ to the graph
end

Theorem 4.23 (PTOPO inside the characteristic box). *Consider a proper parametrization ϕ of curve $\mathcal{C}$ involving polynomials of degree d and bitsize τ, as in Eq.(4.1), that has no singularities at infinity. Alg. 6 outputs a graph G that, if embedded in $\mathbb{R}^n$, is isotopic to $\mathcal{C}$, within the characteristic box $\mathcal{B}_{\mathcal{C}}$. It has worst case complexity*

$$\widetilde{\mathcal{O}}_B(nd^6 + nd^5\tau + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau),$$

while its expected complexity is

$$\widetilde{\mathcal{O}}_B(d^6 + d^5(n + \tau) + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau),$$

If $n = \mathcal{O}(1)$, then the bounds become $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$.

Proof. We count on the fact that ϕ is continuous in $\mathbb{R}\setminus\mathtt{T}_P^{\mathbb{R}}$. Thus, for each real interval $[s, t]$ with $[s, t]\cap\mathtt{T}_P^{\mathbb{R}} = \emptyset$, there is a parametric arc connecting the points $\phi(s)$ and $\phi(t)$. Since for any (sorted) list $\mathtt{T}_i$, for $i \in [\ell]$, the interval defined by the minimum and maximum value of its elements has empty intersection with $\mathtt{T}_P^{\mathbb{R}}$, then for any $s, t \in \mathtt{T}_i$ there exists a parametric arc connecting $\phi(s)$ and $\phi(t)$ and it is entirely contained in $\mathcal{B}_{\mathcal{C}}$. If $\mathbf{p}_\infty$ exists, then $\mathbf{p}_\infty$ is inside $\mathcal{B}_{\mathcal{C}}$. Let $t_{1,1}, t_{\ell,k_\ell}$ be the first element of the first list and the last element of the last list. There is a parametric arc connecting $\phi(t_{1,1})$ with $\mathbf{p}_\infty$ and $\mathbf{p}_\infty$ with $\phi(t_{\ell,k_\ell})$. So we add the edge $\{t_{1,1}, t_{\ell,\kappa_\ell}\}$ to G. Then, every edge of G is embedded to a unique smooth parametric arc and the embedding of G can be trivially continuously deformed to $\mathcal{C}$.

For the complexity analysis, we know from Lem.4.18 that steps 1-2 can be performed in wost-case bit complexity

$$\widetilde{\mathcal{O}}_B(nd^6 + nd^5\tau + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau),$$

and in expected bit complexity

$$\widetilde{\mathcal{O}}_B(d^6 + d^5(n + \tau) + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau),$$

using a Las Vegas algorithm. From Lemmata 4.20, 4.21, and 4.22 steps 4-5 cost $\widetilde{\mathcal{O}}_B(n^2(d^3\tau))$.

To perform steps 6-7 we must sort all the parameters in $\mathtt{T}\cup\mathtt{T}_P^{\mathbb{R}}$, i.e., we sort $\mathcal{O}(d^2 + nd)$ algebraic numbers. The parameters that correspond to cusps and extreme points with respect to the i-th coordinate direction can be expressed as roots of $\prod_{i\in[n]} h_i(t, t)$, which is of size $(nd, n\tau)$. The poles are roots of $\prod_{i\in[n]} q_i(t)$, which has size $(nd, n\tau)$. The parameters that correspond to multiple points are roots of R_s which has size $(d^2, d\tau)$. At last, parameters in $\mathtt{T}_B$ are roots of a polynomial of size $(d, d^2\tau)$.

We can consider all these algebraic numbers together as roots of a single univariate polynomial (the product of all the corresponding polynomials). It has degree $\mathcal{O}(d^2 + nd)$ and bitsize $\widetilde{\mathcal{O}}(d^2\tau + n\tau)$. Hence, its separation bound is in $\widetilde{\mathcal{O}}(d^4\tau + nd^3\tau + n^2d\tau)$ and the

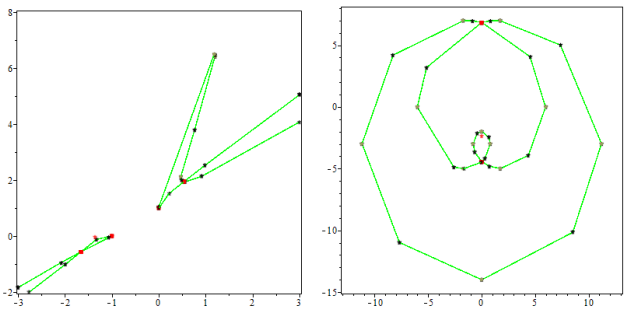


Figure 4.1: *The left figure is the output of* PTOPO *for the parametric curve* $(\frac{3\,t^2+3\,t+1}{t^6-2\,t^4-3\,t-1}, \frac{(t^4-2\,t+2)t^2}{t^6-2\,t^4-3\,t-1})$, *while the right figure is the output for the curve* $(\frac{6\,t^8-756\,t^6+3456\,t^5-31104\,t^3+61236\,t^2-39366}{t^8+36\,t^6+486\,t^4+2916\,t^2+6561}, \frac{-18(6\,t^6-16\,t^5-126\,t^4+864\,t^3-1134\,t^2-1296\,t+4374)t}{t^8+36\,t^6+486\,t^4+2916\,t^2+6561})$. *Multiple points are indicated by red squares and isolated points by red stars.*

same bound also holds for sum of the bitsizes of these numbers. Since we know that the bitsize of a number is in the range $[1, \widetilde{\mathcal{O}}(d^4\tau + nd^3\tau + n^2d\tau)]$, we can use the counting sort algorithm to sort the numbers according to their bitsize in $\widetilde{\mathcal{O}}_B(d^4\tau + nd^3\tau + n^2d\tau)$. Then, we sort numbers of the same bitsize using radix sort, in a bit-complexity bounded by the total sum of bitsizes, that is $\widetilde{\mathcal{O}}_B(d^4\tau + nd^3\tau + n^2d\tau)$. □

If we are interested only in the topology of $\mathcal{C}$, i.e., the abstract graph G, and not in visualizing the curve within $\mathcal{B}_\mathcal{C}$, then the intersection points of $\mathcal{C}$ with $\mathcal{B}_\mathcal{C}$ are not important any more. We sketch a procedure to construct the abstract graph by avoiding their computation:

Assume that we have not computed the points on $\mathcal{C} \cap \mathcal{B}_\mathcal{C}$. We split again the sorted list $\mathtt{T} = [\mathtt{T}_C, \mathtt{T}_M, \mathtt{T}_E]$ at the real poles, and we add an artificial parameter at the beginning and at the end of each sublist. The rest of the procedure remains unaltered.

To verify the correctness of this approach, it suffices to prove that the graph that we obtain by this procedure, is isomorphic to the graph G. It is immediate to see that the latter holds, possibly up to the dissolution of the vertices corresponding to the first and last artificial vertices. Adding these artificial parameters does not affect the overall complexity, since we do not perform any algebraic operations. Therefore, the bit complexity of the algorithm is determined by the complexity of computing the parameters of the special points (Lem.4.18), and so we have the following theorem:

Theorem 4.24 (PTOPO and an abstract graph). *Consider a proper parametrization ϕ of curve $\mathcal{C}$ involving polynomials of degree d and bitsize τ, as in Eq. (4.1), that has no singularities at infinity. Alg. 6 outputs a graph G that, if we embed it in $\mathbb{R}^n$, then it is isotopic to $\mathcal{C}$. It has worst case complexity*

$$\widetilde{\mathcal{O}}_B(nd^6 + nd^5\tau + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3d^2\tau),$$

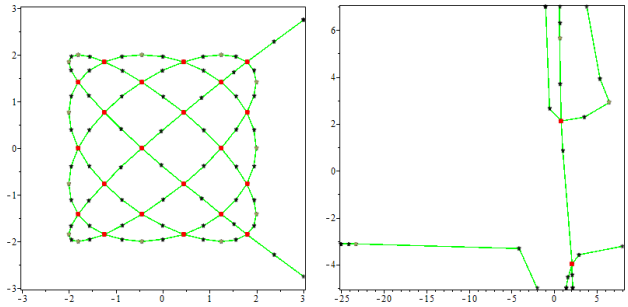


Figure 4.2: *The left figure is the output of* PTOPO *for the parametric curve* $(t^8 - 8\,t^6 + 20\,t^4 - 16\,t^2 + 2, t^7 - 7\,t^5 + 14\,t^3 - 7\,t)$ *while the right figure is the output for the curve* $(\frac{37\,t^3-23\,t^2+87\,t+44}{29\,t^3+98\,t^2-23\,t+10}, \frac{-61\,t^3-8\,t^2-29\,t+95}{11\,t^3-49\,t^2-47\,t+40})$. *Multiple points are indicated by red squares.*

while its expected complexity is

$$\widetilde{\mathcal{O}}_B(d^6 + d^5(n+\tau) + d^4(n^2+n\tau) + d^3(n^2\tau + n^3) + n^3 d^2 \tau),$$

If $n = \mathcal{O}(1)$, *then bounds become* $\widetilde{\mathcal{O}}_B(N^6)$, *where* $N = \max\{d, \tau\}$.

Remark 4.25. *If we are given a box* $\mathcal{B} \subset \mathbb{R}^n$ *at the input, we slightly modify* `PTOPO`, *as follows: We discard the parameter values in* $\mathtt{T}_C \cup \mathtt{T}_M \cup \mathtt{T}_E \cup \mathtt{T}_I$ *that correspond to points not contained in* $\mathcal{B}$. *The set of* G*'s vertices is constructed similarly. To connect the vertices, we follow the same method with a minor modification: For any consecutive elements* $t_1 < t_2$ *in a list* $\mathtt{T}_i$ *with more than two elements, such that* $t_1 \in \lambda(v_{i,1})$ *and* $t_2 \in \lambda(v_{i,2})$, *we add the edge* $\{v_{i,1}, v_{i,2}\}$ *if and only if* $\phi(t_1)$, $\phi(t_2)$ *are not both on the boundary of* $\mathcal{B}$; *or in other words* t_1 *and* t_2 *are not both in* $\mathtt{T}_B$.

4.6 Isotopic embedding for plane and space curves

In this section we elaborate on the isotopic embedding of the output graph G of Thm. 4.24 for the case of plane and space parametric curves $\mathcal{C}$. We embed every edge of the abstract graph G in the corresponding parametric arc by sampling many parameter values in the associated parametric interval and then connecting the corresponding points accordingly, in $\mathbb{R}^2$ or $\mathbb{R}^3$. The larger the number of sampled parameters, the more likely it is for the embedding to be isotopic to $\mathcal{C}$. However, we might need a prohibitive large number of points to sample; their number is related to the distance between two branches of the curve. We show that by introducing a few additional points, we can replace the parametric arcs of the embedded graph with straight line segments and count on it being isotopic to $\mathcal{C}$. Following [ACDT20] closely, if $X, Y \subset \mathbb{R}^n$ are one-dimensional, then being isotopic implies that *one of them can be deformed into the other without removing or introducing self-intersections.*

For plane curves, there is no need to take intermediate points on each parametric arc. We consider the embedding of the abstract graph G in $\mathbb{R}^2$ as a straight-line graph $\tilde{G}$, i.e., with straight lines for edges, whose vertices are mapped to the corresponding points of

the curve. The vertices of $\tilde{G}$ are all the singular and extreme points with respect to the x- and y- directions. Therefore, the edges of $\tilde{G}$ correspond to smooth and monotonous parametric arcs and so they cannot intersect but at their endpoints. The embedding $\tilde{G}$ is then trivially continuously deformed to $\mathcal{C}$. The above discussion summarizes as follows:

Corollary 4.26 (`PTOPO` and isotopic embedding for plane curves). *Consider a proper parametrization ϕ of a curve $\mathcal{C}$ in $\mathbb{R}^2$ involving polynomials of degree d and bitsize τ, as in Eq. (4.1), that has no singularities at infinity. Alg. 6 computes an abstract graph whose straight-line embedding in $\mathbb{R}^2$ is isotopic to $\mathcal{C}$ in worst case complexity $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$.*

For space curves, a straight-line embedding of G is not guaranteed to be isotopic to $\mathcal{C}$ for knots may be present. To overcome this issue, we need to segment some edges of G into two or more edges. To find the extra vertices that we need to add to the graph, we follow a common approach [ADT10, ACDT20, Kah08, DMR08, CJL13] that projects the space curve to a plane one. For a projection defined by the map $\pi : \mathcal{C} \to \mathbb{R}^2$, we write $\tilde{\mathcal{C}} = \pi(\mathcal{C})$. We will ensure in the sequel that the following two conditions are satisfied:

(C1) $\mathcal{C}$ has no asymptotes parallel to the direction of the projection.

(C2) The map π is birational [SWPD08, Def. 2.37].

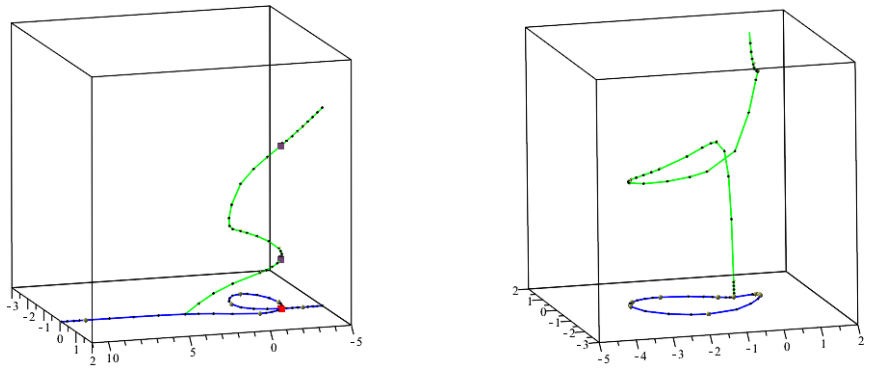
The first condition is to ensure that the point at infinity $\mathbf{p}_\infty$ of $\mathcal{C}$ exists if and only if the point at infinity of $\tilde{\mathcal{C}}$ exists and is equal to $\pi(\mathbf{p}_\infty)$ [ADT10, Lem. 10]; see Fig. 4.3b for an instance where this condition is violated. The second condition ensures that only a finite number of points on $\tilde{\mathcal{C}}$ have more than one point as a preimage. We call these points *apparent singularities* [DMR08]; see Fig. 4.3a. Thus, with this condition we avoid the "bad" cases where two branches of $\mathcal{C}$ project to the same branch of $\tilde{\mathcal{C}}$.

Lemma 4.27. *Consider a proper parametrization ϕ of curve $\mathcal{C}$ in $\mathbb{R}^3$ involving polynomials of degree d and bitsize τ, as in Eq. (4.1), that has no singularities at infinity. We compute a map $\pi : \mathcal{C} \to \mathbb{R}^2$ satisfying conditions (C1), (C2) in worst case complexity $\widetilde{\mathcal{O}}_B(d^5 + d^4\tau)$ and using a Las Vegas algorithm in expected complexity $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$.*

Proof. By [Wal78, Thm. 6.5, pg. 146], any space curve can be birationally projected to a plane curve. We choose an integer a uniformly at random from the set $\{1, \dots Kd^2\}$, where $K = \mathcal{O}(1)$; we explain later in the proof about the size of this set. We define the mapping:

$$\begin{aligned} \pi : \quad & \mathbb{C}^3 \to \mathbb{C}^2 \\ & (x, y, z) \mapsto (x, y + az) \end{aligned} \tag{4.5}$$

It will be useful in the sequel to regard the application of $\pi(\cdot)$ to $\phi(t)$ as being performed in two steps:



(a) (b)

Figure 4.3: *(a) Graph of the curve parametrized by* $\phi(t) = \left(\frac{7t^4-22t^3+55t^2+94t-87}{56t^4+62t^2-97t+73}, \frac{88t^5+4t^4-83t^3+10t^2-62t-82}{56t^4+62t^2-97t+73}, -\frac{95t^5-4t^4+83t^3-10t^2+62t+82}{56t^4+73}\right)$ *(green) and its orthogonal projection on the xy-plane (blue). In red, a double point in the projected curve with two points in its preimage, i.e., apparent singularities (purple). (b) Graph of the curve parametrized by* $\phi(t) = \left(\frac{-7t^4+22t^3-55t^2-94t+87}{-56t^4-62t^2+97t-73}, \frac{-4t^4+83t^3-10t^2+62t+82}{-56t^4-62t^2+97t-73}, \frac{t^7-4t^4+83t^3-10t^2+62t+82}{-56t^4-73}\right)$ *(green) and its orthogonal projection on the xy-plane (blue). The space curve does not have a point at infinity, whereas the plane curve has.*

1. Change of orthogonal basis of $\mathbb{R}^3$: Let $\left\{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\-a\end{pmatrix}, \begin{pmatrix}0\\a\\1\end{pmatrix}\right\}$ be an orthogonal basis of $\mathbb{R}^3$ and
$$A = \begin{pmatrix}1 & 0 & 0\\0 & 1 & a\\0 & -a & 1\end{pmatrix}.$$
In the new basis the curve is parametrized by:
$$(\phi_1(t), \phi_2(t), \phi_3(t)) \cdot A = (\phi_1(t), \phi_2(t) + a\phi_3(t), -a\phi_2(t) + \phi_3(t))$$

2. Orthogonal projection onto the first two coordinates: This yields the plane curve parametrized by $(\phi_1(t), \phi_2(t) + a\phi_3(t))$.

For a given choice of a, we check if conditions (C1) and (C2) are satisfied:

For (C1): The direction of the projection is defined by the vector $(0, a, 1)$. So, $\mathcal{C}$ has no asymptotes parallel to $(0, a, 1)$ if and only if the curve with parametrization $\phi(t) \cdot A$ has no asymptotes parallel to $(0, a, 1) \cdot A^{-1} = (0, 0, 1)$. We can check if this is the case by employing [ADT10, Lem. 9]; this is no more costly that solving a univariate polynomial of size $(\mathcal{O}(d), \widetilde{\mathcal{O}}(\tau))$, which costs $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$ worst case.

Morover, there are $\mathcal{O}(d)$ bad values of a for whom (C1) does not hold: for any asymptote

of $\mathcal{C}$, there is a unique value of a that maps it to an asymptote parallel to $(0,0,1)$ in the new basis. The asymptotes of $\mathcal{C}$ are $\mathcal{O}(d)$ since they occur at the poles of $\phi(t)$ and at the branches that extend to infinity.

For (C2): Since $\phi(t)$ is proper, $\pi(\cdot)$ is birational if and only if $\pi(\phi(t))$ is also proper. We check the properness of this parametrization of $\tilde{\mathcal{C}}$ in $\widetilde{\mathcal{O}}_B(d^3+d^2\tau)$ expected time, using Lem. 4.5.

To find the values of a that result in a 'bad' map: Let $\tilde{h_1}(s,t), \tilde{h_2}(s,t)$ be the polynomials of Eq. (4.1) associated to $\pi(\phi(t))$. The parametrization $\pi(\phi(t))$ is proper if and only if $\gcd(\tilde{h_1}(s,t), \tilde{h_2}(s,t)) = 1$. If $\gcd(\tilde{h_1}(s,t), \tilde{h_2}(s,t)) \neq 1$ then, by letting $R(s) = \mathtt{res}_t(\tilde{h_1}(s,t), \tilde{h_2}(s,t))$, we have that $R(s) = 0$. Notice that $R(s)$ is not always identically zero (e.g., for $\tilde{h_1}(s,t) = t+s, \tilde{h_2}(s,t) = t+s-1$ we get $R(s) = 1$). We consider $R(s)$ as a polynomial in $\mathbb{Z}(a)[s]$:

$$R(s) = c_{d^2}(a)s^{d^2} + \cdots + c_1(a)s + c_0(a),$$

where $c_i \in \mathbb{Z}[a]$ is of size $(d, \widetilde{\mathcal{O}}(d\tau))$ for $0 \leq i \leq d^2$. The bad values of $a \in \mathbb{R}$ satisfy then the equation:

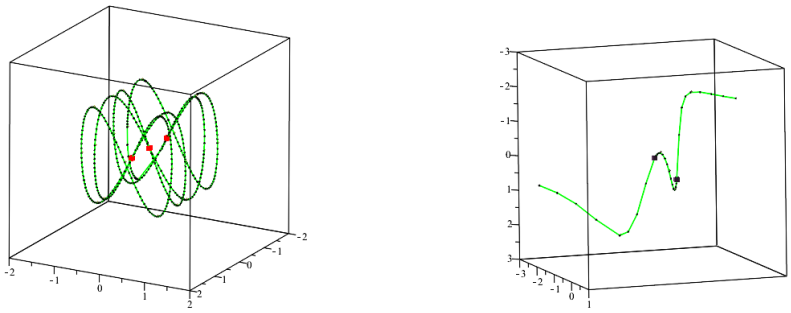
$$c_{d^2}^2(a) + \cdots + c_1^2(a) + c_0^2(a) = 0.$$

The polynomial has degree $\mathcal{O}(d^2)$ and so there are $\mathcal{O}(d^2)$ bad values to avoid. This points to the worst case complexity $\widetilde{\mathcal{O}}_B(d^5 + d^4\tau)$. □

Given a map π computed through Lem. 4.27, we find the parameters that give the real multiple points of $\tilde{\mathcal{C}}$ and its extreme points with respect to the two coordinate axes. We add the corresponding vertices to G and we obtain an augmented graph, say G'. The straight-line embedding of G' in $\mathbb{R}^2$ is isotopic to $\tilde{\mathcal{C}}$ by Cor. 4.26, possibly up to the isolated points [ADT10, Thm. 13 and Lem. 15]. Then, by lifting this embedding to the corresponding straight-line graph in $\mathbb{R}^3$, we obtain a graph isotopic to $\mathcal{C}$ [ADT10, Thm. 13 and Thm. 14]. The following theorem summarizes the previous discussion and states its complexity:

Theorem 4.28 (`PTOPO` and isotopic embedding for 3D space curves). *Consider a proper parametrization ϕ of curve $\mathcal{C}$ in $\mathbb{R}^3$ involving polynomials of degree d and bitsize τ, as in Eq. (4.1), that has no singularities at infinity. There is an algorithm that computes an abstract graph whose straight-line embedding in $\mathbb{R}^3$ is isotopic to $\mathcal{C}$ in worst case complexity $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$.*

Proof. Given a projection map $\pi(\cdot)$ such that (C1) and (C2) hold, correctness follows from the previous discussion. From Lem. 4.27, we find such a map in expected complexity $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$ and $\widetilde{\mathcal{O}}_B(d^5 + d^4\tau)$. Using Alg. 5 for $\pi(\phi(t))$ we find the parameters of the extreme (in the coordinate directions) and real multiple points of $\tilde{\mathcal{C}}$, say $\tilde{\mathtt{T}}$, in $\widetilde{\mathcal{O}}_B(d^6+d^5\tau)$ (Lem. 4.18). Then, we employ Alg. 6 for $\phi(t)$, by augmenting the list of parameters that are treated with $\tilde{\mathtt{T}}$. The last step dominates the complexity and is not affected by the addition



(a) (b)

Figure 4.4: *Output of* `PTOPO` *for (a) a Lissajous curve parametrized by* $\left(\frac{(-t^2-1)}{t^2+1}, \frac{-8t(t^2-1)(t^4-6t^2+1)}{t^8+4t^6+6t^4+4t^2+1}, \frac{-(16(t^4-6t^2+1))(t^2-1)t(t^8-28t^6+70t^4-28t^2+1)}{(t^16+8t^14+28t^12+56t^10+70t^8+56t^6+28t^4+8t^2+1)}\right)$. *The multiple points are indicated by the red squares. (b) a space curve parametrized by* $\left(\frac{-2t^2}{t^2+1}, \frac{-2t(3t^5-19t^4+8t^3-9t^2-t+2}{(t^2+1)(10t^4-6t^3+3t^2+1)}, t\right)$. *The apparent singularities are indicated by the black squares.*

of extra parameters to the list since $|\tilde{\mathtt{T}}| = \mathcal{O}(d^2)$. The complexity result in Thm. 4.24 allows us to conclude. □

4.7 Multiplicities and characterization of singular points

We say that a singularity is ordinary if it is at the intersection of smooth branches only and the tangents to all branches are distinct [Wal78, p.54]. In all the other cases, we call it a *non-ordinary* singularity. The *character* of a singular point is either ordinary or non-ordinary. To determine the multiplicity and the character of each real singular point in $\mathbb{C}$ we follow the method presented in the series of papers [PD07, BPD17, BPD19]. They provide a complete characterization using resultant computations that applies to curves of any dimension. In the sequel, we present the basic ingredients of their approach and we estimate the bit complexity of the algorithm.

Let $n = 2$ and $H_i(s,t) = p_i(s)q_i(t) - p_i(t)q_i(s)$, for $i \in [2]$. Consider a point $\mathbf{p} \in \mathcal{C}$ given by the parameter values $\{s_1, \ldots, s_k\}$, $k \geq 1$; that is $\mathbf{p} = \phi(s_i)$, for all $i \in [k]$. The *fiber function* at $\mathbf{p}$ [BPD19, Def. 2] is

$$F_{\mathbf{p}}(t) = \gcd(H_1(s_j,t), H_2(s_j,t)),$$

for any $j \in [k]$. It is a univariate polynomial, the real roots of which are the parameter values that correspond to the point $\mathbf{p}$. When the parametrization of the curve is proper, for any point $\mathbf{p}$ on $\mathcal{C}$ other than $\mathbf{p}_\infty$ it holds that $\deg(F_{\mathbf{p}}(t)) = \mathrm{mult}_{\mathbf{p}}(\mathcal{C})$ [BPD19, Cor. 1]. Also, when the parameters in $\{s_1, \ldots, s_k\}$ are all real, k equals the *number of real branches* that go through $\mathbf{p}$.

To classify ordinary and non-ordinary singularities we proceed as follows: For a point $\mathbf{p} \in \mathcal{C}$ the *delta invariant* $\delta_{\mathbf{p}}$ is a nonnegative integer that measures the number of double points concentrated around $\mathbf{p}$. We can compute it by taking into account the multiplicities of $\mathbf{p}$ and its neighboring singularities. [BPD19] consider three different types of non-ordinary singularities and use the delta invariant to distinguish them. In particular:

1. If $k = \mathrm{mult}_{\mathbf{p}}(\mathcal{C})$ and $2\delta_{\mathbf{p}} = \mathrm{mult}_{\mathbf{p}}(\mathcal{C})(\mathrm{mult}_{\mathbf{p}}(\mathcal{C})-1)$, then $\mathbf{p}$ is an ordinary singularity.

2. If $k < \mathrm{mult}_{\mathbf{p}}(\mathcal{C})$ and $2\delta_{\mathbf{p}} = \mathrm{mult}_{\mathbf{p}}(\mathcal{C})(\mathrm{mult}_{\mathbf{p}}(\mathcal{C})-1)$, then $\mathbf{p}$ is a type I non-ordinary singularity.

3. If $k = \mathrm{mult}_{\mathbf{p}}(\mathcal{C})$ and $2\delta_{\mathbf{p}} > \mathrm{mult}_{\mathbf{p}}(\mathcal{C})(\mathrm{mult}_{\mathbf{p}}(\mathcal{C})-1)$, then $\mathbf{p}$ is a type II non-ordinary singularity.

4. If $k < \mathrm{mult}_{\mathbf{p}}(\mathcal{C})$ and $2\delta_{\mathbf{p}} > \mathrm{mult}_{\mathbf{p}}(\mathcal{C})(\mathrm{mult}_{\mathbf{p}}(\mathcal{C})-1)$, then $\mathbf{p}$ is a type III non-ordinary singularity.

[BPD19] compute the delta invariant, $\delta_{\mathbf{p}}$, using the formula

$$\delta_{\mathbf{p}} = \frac{1}{2}\sum_{j=1}^{k} \sum_{t \text{ s.t. } h_1(s_j,t)=h_2(s_j,t)=0} \mathrm{Int}_{h_1,h_2}(s_j,t), \tag{4.6}$$

where $\mathrm{Int}_{h_1,h_2}(\alpha,\beta)$ is the *intersection multiplicity* of the coprime polynomials $h_1(s,t)$ and $h_2(s,t)$ at a point (α,β). Using a well known result, e.g., [Ful84, 1.6] as stated in [BKM05, Prop. 5], we can compute the intersection multiplicities using resultant computations. Let $R(s) = \mathtt{res}_t(h_1(s,t), h_2(s,t))$. For a root α of $R(s)$, its multiplicity $\mu(a)$ is equal to the sum of the intersection multiplicities of solutions of the system of $\{h_1, h_2\}$ in the form (α, t), and so

$$\mu(\alpha) = \sum_{t \text{ s.t. } h_1(\alpha,t)=h_2(\alpha,t)=0} \mathrm{Int}_{h_1,h_2}(\alpha,t). \tag{4.7}$$

Therefore, from Eq. (4.6) and Eq. (4.7), we conclude that:

$$\delta_{\mathbf{p}} = \frac{1}{2}\sum_{j=1}^{k} \mu(s_j). \tag{4.8}$$

For space curves, we birationally project $\mathcal{C}$ to a plane curve [Wal78, Thm. 6.5, pg. 146]. The multiplicities of the singular points of $\mathcal{C}$ and their delta invariant are preserved under the birational map realizing the projection [BPD19, Prop. 1 and Cor. 4]. The pseudo code of the algorithm appears in Alg. 7.

Theorem 4.29. *Let $\mathcal{C}$ be a curve with a proper parametrization $\phi(t)$ as in Eq. (4.1), that has no singularities at infinity. Alg. 7 computes the singular points of $\mathcal{C}$, their multiplicity*

Algorithm 7: `CharacterizeSingularPoints`$(\phi, \mathcal{S})$

Input: Proper parametrization $\phi \in \mathbb{Z}(t)^n$ without singularity at infinity, as in Eq. (4.1)

Output: Multiplicities and characterization of points

```
S ← Special_Points(φ)
if n=2 then
    Compute polynomials H1(s,t), H2(s,t) for φ
end
else
    repeat
        Choose integers a3,...,an at random from {1,...,Kd^n}, where K = O(1).
        φ̃(t) ← (φ1(t), φ2(t) + a3φ3(t) + ··· + anφn(t))
    until φ̃(t) is proper;
    Compute polynomials H1(s,t), H2(s,t) for φ̃
end
for p ∈ S do
    M_p ← {s ∈ ℝ : φ(s) = p} // parameters that give the same point p
    k ← |M_p| // number of real branches that go through p
    Take s0 ∈ M_p
    mult_p(C) ← deg(gcd(H1(s0,t), H2(s0,t))) // multiplicity
    // compute delta invariant
    δ_p ← 0
    for sj ∈ M_p do
        μ(sj) ← multiplicity of sj as a root of res_t(H1(s,t)/(s−t), H2(s,t)/(s−t))
        δ_p ← δ_p + μ(sj)
    end
    δ_p ← δ_p/2
    if k = mult_p(C) and 2δ_p = mult_p(C)(mult_p(C) − 1) then
        return p is an ordinary singularity
    end
    else if k < mult_p(C) and 2δ_p = mult_p(C)(mult_p(C) − 1) then
        return p is a type I non-ordinary singularity
    end
    else if k = mult_p(C) and 2δ_p > mult_p(C)(mult_p(C) − 1) then
        return p is a type II non-ordinary singularity
    end
    else if k < mult_p(C) and 2δ_p > mult_p(C)(mult_p(C) − 1) then
        return p is a type III non-ordinary singularity
    end
end
```

and character (ordinary/non-ordinary) in

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

worst-case complexity when $n = 2$ and for $n > 2$ in expected complexity

$$\widetilde{\mathcal{O}}_B(d^6 + d^5(n+\tau) + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3 d^2\tau).$$

Proof. We compute the parameters that give the singular points of $\mathcal{C}$ using Alg. 5 in $\widetilde{\mathcal{O}}_B(d^6 + d^5(n+\tau) + d^4(n^2 + n\tau) + d^3(n^2\tau + n^3) + n^3 d^2\tau)$ when $n > 2$, which becomes worst-case when $n = 2$ (Lem. 4.18).

When $n > 2$, lines 5-8 compute a birational projection of $\mathcal{C}$ to a plane curve parametrized by $\tilde{\phi}(t)$ (note that the projection is birational if and only if $\tilde{\phi}(t)$ is proper since $\phi(t)$ is proper). The expected complexity of this is $\widetilde{\mathcal{O}}_B(d^3 + nd^2\tau)$ by slightly adapting the proof of Lem. 4.27.

We can group the parameter values that give singular points using Lem. 4.22 in $\mathcal{O}(d^2 + nd)$ arithmetic operations. We have that $\gcd(H_1(s,t), H_2(s,t)) = s - t$. For $h_1(s,t) = H_1(s,t)/(s-t)$, $h_2(s,t) = H_2(s,t)/(s-t)$, we compute a triangular decomposition of the system $\{h_1(s,t) = h_2(s,t) = 0\}$ which consists of the systems $\{(A_i(s), B_i(s,t))\}_{i\in\mathcal{I}}$. For any root α of A_i, $B_i(\alpha,t)$ is of degree i and equals $\gcd(h_1(\alpha,t), h_2(\alpha,t))$ (up to a constant factor). By [BLM+15, Prop. 16], the triangular decomposition is computed in $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$ worst case[2].

For a singular point $\mathbf{p}$ and $s_0 \in M_{\mathbf{p}}$: To compute the degree of $\gcd(H_1(s_0), H_2(s_0))$, since s_0 is a root of $\mathtt{res}_t(h_1(s,t), h_2(s,t))$, it suffices to find for which $i \in \mathcal{I}$ $A_i(s_0) = 0$. The latter is not immediate when s_0 is given in isolated interval representation as a root of $\mathtt{res}_t(h_1(s,t), h_2(s,t))$. However, we can isolate the roots of all A_i by taking care that the isolating invervals are small enough so that the isolating interval of s_0 intersects only one of them. Because they are roots of the same polynomial this cannot exceed the complexity of isolating the roots of $\mathtt{res}_t(h_1(s,t), h_2(s,t))$. This is can be done in $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$. In the same bit complexity, we have the multiplicity $\mu(s_j)$ of s_j as a root of $\mathtt{res}_t(h_1, h_2)$. □

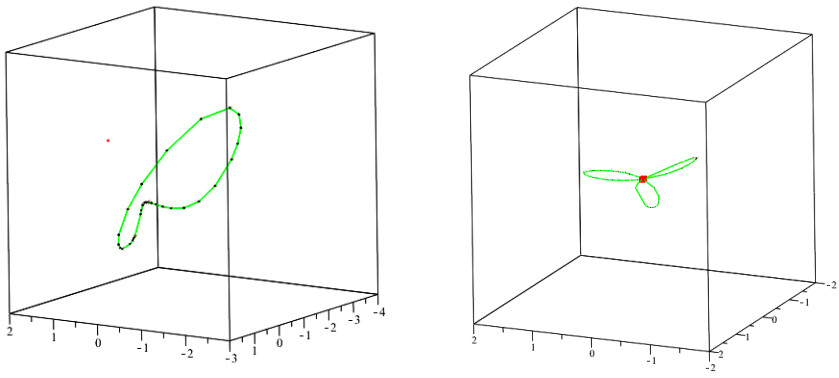
4.8 Implementation and Examples

`PTOPO` is implemented in MAPLE[3] for plane and 3D curves. A typical output appears in Figures 4.1, 4.2, 4.5a, and 4.5. Besides the visualization the software computes all the points of interest of curve (singular, extreme, etc) in isolating interval representation as well as in suitable floating point approximations.

We build upon the real root isolation routines of MAPLE's `RootFinding` library and the SLV package [DET09], to use a certified implementation of general purpose exact computations with one and two real algebraic numbers, like comparison and sign evaluations, as well as exact (bivariate) polynomial solving.

[2] In a Las Vegas setting this computation could be reduced to $\widetilde{\mathcal{O}}_B(d^4 + d^3\tau)$, but since it does not affect the total complexity we chose not to expand onto this.

[3] `https://gitlab.inria.fr/ckatsama/ptopo`



(a) (b)

Figure 4.5: *Output of* `PTOPO` *for a curve parametrized by (a)* $\left(\frac{-7t^4+22t^3-55t^2-94t+87}{-56t^4-62t^2+97t-73}, \frac{-4t^4*83t^3-10t^2-62t-82}{-56t^4-62t^2+97t-73}, \frac{-4t^4*83t^3-10t^2-62t-82}{-56t^4-62t^2+97t-73}\right)$ *(b)* $\left(\frac{-3t^2+1}{(t^2+1)^2}, \frac{(-3t^2+1)t}{(t^2+1)^2}, \frac{(-3t^2+1)t^3}{(t^2+1)^4}\right)$. *The red star in (a) corresponds to an isolated point, whereas the red square in (b) corresponds to a multiple point. Examples are taken from [ADT10].*

`PTOPO` computes the topology and visualizes parametric curves in two (and in the near future in three) dimensions. For a given parametric representation of a curve, `PTOPO` computes the special points on the curve, the characteristic box, the corresponding graph, and then it visualizes the curve (inside the box). The computation, in all examples from literature we tested, takes less than a second in a MacBook laptop, running MAPLE 2020. We refer the reader to the website of the software and to [KRTZ20b] for further details.

Chapter 5

Convex Hull of Parametric Curves

We consider the problem of the computing the boundary of the convex hull of rational parametric curves in $\mathbb{R}^2$ and in $\mathbb{R}^3$. The boundary of the convex hull is a semi-algebraic set and an exact representation, breaks it down to a combination of line segments and arcs of the curve for the 2D case, and of triangles and surface patches for the 3D case. We provide an algorithmic solution together with bit-complexity estimates. We show that its computation reduces to univariate and bivariate solving and to isolating roots of a univariate polynomial with coefficients in a multiple field extension. We express the bit-complexity with respect to the bit-complexity of the latter operation. By utilizing the results of Ch. 3, we offer asymptotic upper bounds on the bit-complexity.

5.1 Introduction

For any subset of $\mathbb{R}^n$, where $n \geq 1$, its convex hull is defined as the smallest convex set that contains it, or in more technical terms, as the intersection of its supporting halfspaces [Brø83, Thm. 4.5]. Convex hull computations are fundamental in computational geometry with direct applications in motion planning [ZST11, YLLF11], computer vision [dFT90] and geometric modeling systems [EMP10, GVNPD$^+$04]; to mention few of them. Convex hulls of linear objects (points, segments, polygons, polyhedra) are well-studied in literature [dBCvKO08]. We focus on non-linear convex hulls, and in particular on the convex hull of parametric curves in $\mathbb{R}^2$ and in $\mathbb{R}^3$. Convex hulls of non-linear objects arise naturally in optimization [Las09, BPT12] and learning theory [CS01b, SNW11] because they are useful, among other things, in optimizing a linear function over a nonlinear object.

Let $\mathcal{C}$ be an algebraic curve over $\mathbb{R}^n$, parametrized by

$$\begin{aligned} \phi: \quad & \mathbb{R} \dashrightarrow \mathbb{R}^n \\ & t \mapsto \big(\phi_1(t), \ldots, \phi_n(t)\big) = \Big(\frac{p_1(t)}{q_1(t)}, \ldots, \frac{p_n(t)}{q_n(t)}\Big), \end{aligned} \tag{5.1}$$

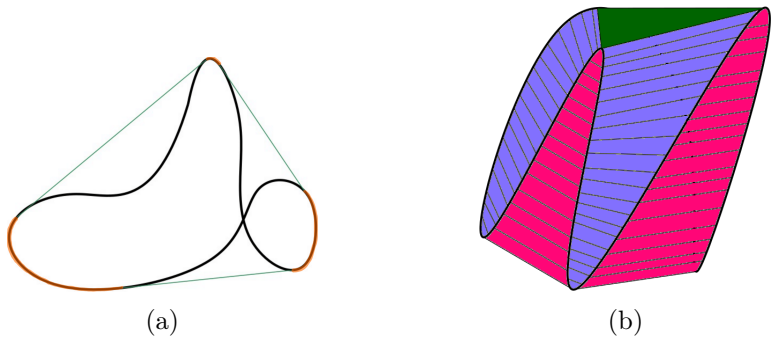


Figure 5.1: (a) A plane curve and its convex hull; the boundary consists of the green line segments and the orange arcs. (b) A curve in $\mathbb{R}^3$ and its convex hull; the visible part of the boundary consists of the green triangle and the ruled surfaces patches in purple and magenta.

where $p_i, q_i \in \mathbb{Z}[t]$ have degree at most d and bitsize τ and $\gcd(p_i(t), q_i(t)) = 1$, $i \in [n]$. Let $I \subseteq \mathbb{R}$. We denote by $\mathrm{conv}(\phi(I))$ the convex hull of $\phi(I)$. When $n = 2$, the boundary of the convex hull consists of a combination of smooth curved arcs and segments joining two points on the curve (Fig. 5.1a). When $n = 3$, it is a combination of triangles and ruled (developable) surface patches (Fig. 5.1b). We design algorithms for the boundary description in these cases and we study their bit-complexity.

Parametric curves stimulated our interest since the parametric representation has already been proven advantageous in the topology computation; for parametric curves in $\mathbb{R}^n$ the bit-complexity is linear in n, whereas for the implicit case the dependence in n seems to be exponential. In particular, for plane curves the bit-complexity of the implicit and parametric case coincides [KRTZ20a] but for space curves the bit-complexity gap is already of order $\mathcal{O}(d^2)$, where d is the degree of the polynomials involved [CJP+21]. In addition, parametric curves come up in various applications in control theory [Kur12], in machine learning [SMC20] or in chemical engineering [CKLS18].

Previous work. There is a series of works for the problem of computing the convex hull of curved arcs, given in parametric form, on the plane [SJVW87, DS90, BK91, JL05]. Nevertheless, in all these approaches, there are assumptions on the monotonicity and/or the total curvature of the arcs. The corresponding algorithms are combinatorial and are adaptations of algorithms for polygons. The resulting arithmetic complexity estimates are optimal with respect to the number of curved edges they consider but they rely on several oracles whose running time can be expensive. For instance, such oracles are considered for the computation of bitangents, that are lines tangent to the curve at two points.

Bitangents' computation is a problem of independent interest that appears in several geometric applications, e.g., on belt synchronous of automobile engines [CLL+20]. Lee and Kim [LK92] compute the common tangents to monotone curve segments. Parida

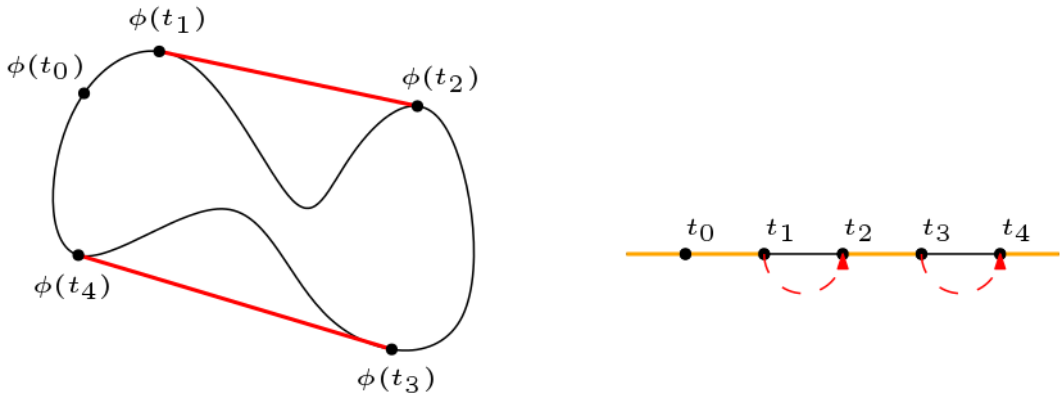


Figure 5.2: On the left, a smooth plane curve and the bitangents on the boundary of the convex hull (in red). On the right, the parameter intervals corresponding to arcs on the boundary of the convex hull are annotated in orange.

and Mudur [PM95] present a heuristic algorithm for computing common tangent lines of two parametric curves. Their algorithm is based on a rejection criterion that aims in excluding portions of the curves that do not admit common tangent lines. Johnstone [Joh01] reduces the problem of finding common tangents between a pair of parametric curves to the intersection of parametric curves in a dual space. Then, he applies his method for the convex hull computation of a smooth paramatric plane curve [Joh04]; the bitangents now correspond to multiple points of the dual curve. After computing the bitangents, he constructs the convex hull by walking along the curve, starting from a point on the convex hull and leaping, every time a bitangent is found, to the other end of the segment. Essentially, with this method we walk along the curve by walking along the parameter interval. It applies only to smooth curves; for example, for the curve of Fig. 5.2, starting from t_0, we walk along the parameter interval $(t_0, +\infty)$ and when we reach t_1 we leap to t_2. Then, we continue walking on the parameter interval in the same direction, that is on the interval $(t_2, +\infty)$. On the contrary, for the curve in Fig. 5.3 that has a self intersection, after leaping from the point $\phi(t_1)$ to $\phi(t_3)$, we must continue walking on the parameter interval $(-\infty, t_3)$ and not on $(t_3, +\infty)$.

A more general approach for plane parametric curves with possible self-intersections is proposed by Elber et al. [EKH01]. They express the parameter intervals that correspond to the arcs of the curve on the boundary of the convex hull, as the complement of the projection of the graph of a bivariate polynomial on one coordinate axis. Then, they sort the arcs with respect to their normal angles and connect them accordingly with segments whenever there is a gap. So, in some sense, the sorting compensates for the fact that now the curve has self-intersections and the connections of the arcs of the curve are not trivial. In practice, computing the arcs of the curve that are on the boundary of the convex hull and sorting them is not straightforward. For an exact algorithm and an analysis of its bit-complexity, a precise algebraic formulation of the problem is needed. Moreover, the

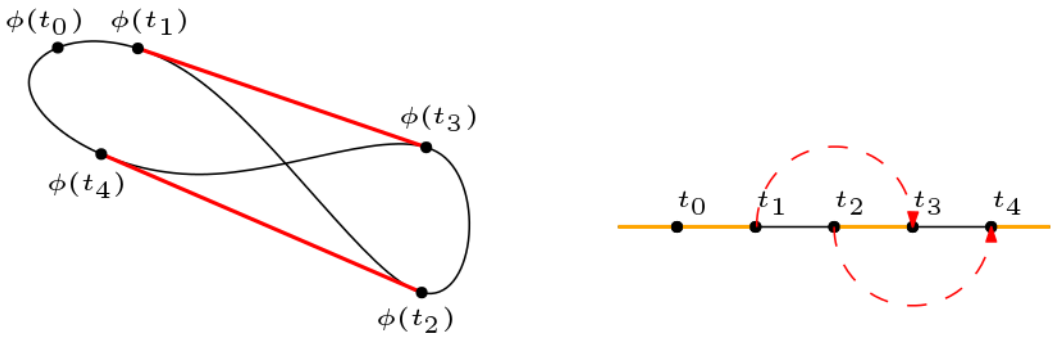


Figure 5.3: On the left, a non-smooth plane curve and the bitangents on the boundary of the convex hull (in red). On the right, the parameter intervals corresponding to arcs on the boundary of the convex hull are annotated in orange.

presence of cusps is not considered.

A general algorithm for the case of implicit curves in the plane is given by [KYP92], without bit-complexity estimates. Although the general idea of the algorithm is common for parametric and implicit plane curves, it is important to take advantage of the parametric representation when designing the necessary predicates. So, our algorithms for parametric curves are not to be juxtaposed to the ones for implicit curves.

For space curves, Sedykh provides a detailed study of the singularities of the convex hull's boundary [Sed86b]. Nevertheless, in our case, a complete classification is not necessary. Seong et al. [SEJK04] present an algorithm for the computation of the convex hull, that is extension of the work of [EKH01] for plane curves. They describe the boundary of the convex hull as a combination of developable surface patches and plane patches (triangles). The plane patches are found by solving systems of polynomial equations and the developable surface patches by tracing an implicit plane curve that expresses a bitangent condition. As with the algorithm for plane curves, cusps are not treated and there are no bit-complexity estimates. Ranestad and Sturmfels [RS09] compute the algebraic boundary of the convex hull of a space curve, that is an irreducible polynomial defining the so-called edge surface. However, this surface contains parts that are not on the boundary of the convex hull. They extend their work for general algebraic varieties in [RS11] using duality. An important result of Henrion [Hen11] states that the convex hull of parametric curves in $\mathbb{R}^n$ can be represented as the projection of a spectrahedron, the feasible region of a semidefinite program, at the price of introducing a certain number of lifting variables, that is polynomial in the dimension. Nevertheless, although this expression of the convex hull is appropriate for optimization, it does give information on the facial structure of the boundary. Vinzant [Vin11] builds off the works [RS09, RS11, Hen11] and describes the faces of the convex hull of parametrized curves using semidefinite programming, yet without providing much information about the connection of the different facets nor bit-complexity estimates. Her results hold for parametric curves of any dimension. The convex hull of

space curves of totally positive torsion is studied in [dIM22], who provide parametrizations of the upper and lower hulls.

Other work on non-linear higher dimensional objects, however not related to our approach, is the one of [CKLS18], that sample points on a smooth parametric curve and then compute inner polyhedral approximations that converge to the convex hull. Ierardi [HI93] exploits duality to find the convex hull of general dimensional "curved polyhedra" following a combinatorial method. We also refer to [BCD$^+$96] for the convex hull of two circles in $\mathbb{R}^3$ and to [GIHS01] for the convex hull of ellipsoids.

Contribution. We propose an algorithm for the convex hull computation of plane parametric curves, as well as one for parametric curves in $\mathbb{R}^3$. The algorithms apply to any curve, as long as it is properly reparametrized (see [PD06] for an algorithm and [KRTZ20a] for an analysis of its complexity), but require in the input a parameter interval $I \subset \mathbb{R}$, such that $\phi(I)$ is compact (Rem. 5.1). Throughout the chapter, we have assumed for simplicity that the sum of the bitsizes of the boundary points of I is a small constant. The algorithms share the basic idea and they both rely on our Support predicate (Sec. 5.3) that applies to any dimension and checks if a given hyperplane is a supporting hyperplane for the curve (see Sec. 5.2 for a definition). The latter problem is reduced to a condition of sign-invariance of a univariate polynomial. We remark that for implicit curves designing such a predicate is not straightforward.

The boundary of the convex hull of plane parametric curves comprises of arcs of the curve and line segments, whose types we classify in Def. 5.2. The first step of the algorithm is to compute all these line segments. This is realised by solving some polynomial systems. However, among the solutions we obtain segments that do not lie on a supporting line to the curve and thus, they are not on the boundary of $\mathrm{conv}(\phi(I))$ (see Fig. 5.4 for an example). It is not needed at this point to check if they lie on a supporting line. The second step consists of decomposing the curve at the endpoints of these segments; in this way, every arc of the decomposition is either entirely on the boundary of the convex hull, or it is contained in its interior. This is equivalent to decomposing the parameter interval I at the parameters that correspond to the segments' endpoints. At the third step, we start from a point on $\phi(I)$, we follow the branches of the curve one by one and we check for every branch if it is on the boundary of the convex hull by calling the Support predicate for an arbitrary tangent line at the interior of the branch. The last step, amounts to connecting the branches that are on the boundary accordingly with line segments. If the endpoint of a branch is smooth, then determining the segment is trivial, since it lies on the tangent line (and the other endpoint has been found at the first step). When the endpoint is not smooth (i.e., cusp or endpoint of $\phi(I)$), we check all the possible segments by employing the Support predicate of Sec. 5.3.

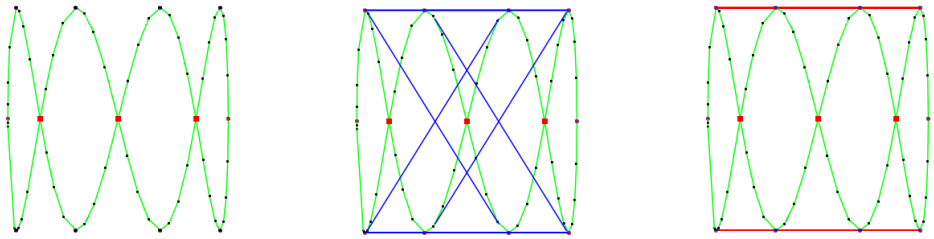
The boundary of the convex hull of curves in $\mathbb{R}^3$ consists of a combination of triangle

facets, whose types we classify in Def. 5.9, and of some surface patches (Def. 5.11). The surface patches are ruled (they are generated by line segments connecting two points on the curve) and also developable. They are part of the bisecant surface (Eq. (5.8)), which is traced through the bisecant curve (Eq. (5.7)), an implicit plane curve in the space of parameters. Finding the surface patches on the boundary is equivalent to determining certain branches of the bisecant curve. The algorithm follows the same line as in the 2D case. First, we compute the triples of parameters that correspond to triangle facets, through solving some polynomial systems. At the second step, we split the graph of the bisecant curve into branches by computing a special purpose Cylindrical Algebraic Decomposition (CAD) of I. The third step is to identify the surface patches that are on the boundary of the convex hull. They correspond to branches of the bisecant curve. The SUPPORT predicate is used again to find the branches of the bisecant curve that correspond to surface patches. At the last step, we connect the surface patches with triangle facets. Triangles with a smooth vertex that are on the boundary are revealed by the endpoints of the branches of the bisecant curve. We call the predicate SUPPORT for the planes on which there are triangles that do not have any smooth vertex. Then, the triangles that do not belong on a supporting plane are discarded.

Our algorithms are built upon the algorithms of [EKH01] for plane curves and of [SEJK04] for 3D curves. These approaches present challenges when it comes to implementation and lack analysis of their computational complexity. So, our goal is to bridge the gap between a theoretical approach and one that can be practically implemented in a working system. Moreover, linear facets (segments or triangles) with non-smooth vertices (cusps or endpoints of $\phi(I)$) require special treatment. This is done at the last step of our algorithm, where we check if the corresponding lines or planes are supporting for $\phi(I)$. When the facets have a smooth vertex, this is not necessary to do so, since the tangency condition at the smooth point defines the facet uniquely.

A bit-complexity analysis, up to our knowledge, has been missing from literature. Our main challenge was to minimize the bit-complexity by identifying the appropriate algebraic formulation that could eliminate expensive operations. By exploiting the problem's geometry, we show that treating the linear facets with non-smooth vertices at the last step amounts to isolating the roots of a zero-dimensional system of the form $\{F_1(X_1) = \cdots = F_N(X_N) = F(\mathbf{X}, Y)\}$. We denote the bit-complexity of this operation by $C(M, \Lambda, N)$ (Def. 5.4). We emphasize that the last polynomial might not be square-free, if we consider it as univariate polynomial in Y. Since there has not been extensive work on the complexity of isolating roots of systems of this form, we express the bit-complexity with respect to d, τ and $C(M, \Lambda, N)$. We employ algorithms for univariate and bivariate solving, and the resulting bit-complexities are in

$$\widetilde{\mathcal{O}}_B\left(d^7 + d^6\tau\right) + C\left(d, \tau, 2\right)$$



(a) (b) (c)

Figure 5.4: *(a) The graph of the curve parametrized by* $\left(\frac{-t^2+1}{t^2+1}, \frac{-8t(t^2-1)(t^4-6t^2+1)}{t^8+4t^6+6t^4+4t^2+1}\right)$*, (b) segments tangent to the curve at two points or more (in blue) and (c) the ones that are on the convex hull (in red).*

for 2D curves (Thm. 5.6), and in

$$\widetilde{\mathcal{O}}_B(d^{10} + d^9\tau) + C(d, d^6\tau, 1) + C(d, d+\tau, 3)$$

for 3D curves (Thm. 5.15). Using [DDR$^+$22, Prop.19] for the case where $N = 1$ and our results from Ch. 3, the previous bit-complexities become

$$\widetilde{\mathcal{O}}_B\left(d^{10} + d^9\tau\right)$$

for plane curves and

$$\widetilde{\mathcal{O}}_B\left(d^{13} + d^{12}\tau\right)$$

for space curves. Alternatively, one can employ the Las-Vegas algorithm of [BS16] for $N \times N$ zero-dimensional polynomial systems. Then, by denoting by $\omega \approx 2.372873$ the exponent in the complexity of matrix multiplication, the bit-complexities become $\widetilde{\mathcal{O}}_B(d^{2\omega+5}(d+\tau)) \approx \widetilde{\mathcal{O}}_B(d^{9.75}(d+\tau))$ for plane curves and $\widetilde{\mathcal{O}}_B(d^{3\omega+7}(d+\tau)) \approx \widetilde{\mathcal{O}}_B(d^{14.12}\ (d+\tau))$ for 3D curves. So, our results lead to lower bit-complexity estimates. A preliminary version of our algorithms is implemented in Maple; we give some examples in Sec. 5.6.

Outline. In Sec. 5.2 we give the basic background on rational curves in $\mathbb{R}^n$ and their convex hull. A general predicate for parametric curves in any dimension is given in Sec. 5.3, that decides if a hyperplane is supporting for the curve. In Sec. 5.4 we present the algorithm for plane curves (Sec. 5.4.1) and its bit-complexity (Sec. 5.4.2). Analogously, in Sec. 5.5 we present the algorithm for 3D curves (Sec. 5.5.1) and its bit-complexity (Sec. 5.5.2). Sec. 5.6 contains examples using our prototype implementation.

5.2 Background on rational curves

For the sake of generality, we introduce some basic notions for rational curves in $\mathbb{R}^n$ and their convex hull. Let $\mathcal{C}$ be an algebraic curve over $\mathbb{R}^n$, parametrized by ϕ in Eq. (5.1). Then, $\mathcal{C}$ is the Zariski closure of $\phi(\mathbb{R})$. We call $\phi(t)$ a *parametrization* of $\mathcal{C}$. We say that the parametrization is of size (d, τ) when the polynomials p_i, q_i have degree at most d and bitsize τ, $i \in [n]$. The parametrization is *proper*, if it is injective for almost all points on $\mathcal{C}$ [SWPD08, Ch. 4]. If it is not proper, reparametrization algorithms exist, e.g., [PD06] (see [KRTZ20a] for an analysis of its complexity). Without loss of generality, we assume that no component of the parametrization ϕ is constant. The *point at infinity*, $\mathbf{p}_\infty$, is the point on $\mathcal{C}$ we obtain for $t \to \pm\infty$ (if it exists). The *tangent vector* of the curve at a point $\mathbf{p} = \phi(t)$ is $\phi'(t)$.

Singular points on the curve correspond to shape features that are known as cusps and self-intersections of smooth branches. Self-intersections are *multiple* points, i.e., points on $\mathcal{C}$ with more than one preimages. The parameters that correspond to pairs of multiple points are obtained by solving the square polynomial system [KRTZ20a, Lem. 11]:

$$h_i(s,t) := \frac{p_i(s)q_i(t) - q_i(s)p_i(t)}{s-t}, \quad \text{for } i \in [n]. \tag{5.2}$$

Cusps are points on the curve where the tangent vector is the zero vector. This is a necessary and sufficient condition when the parametrization is proper [MC92]. It holds that $h_i(t,t) = \phi_i'(t)q_i^2(t)$, for $i \in [n]$. Therefore, a parameter that gives a cusp of $\mathcal{C}$ is a root of

$$H(t) := \sum_{i=1}^{n} h_i^2(t,t) \tag{5.3}$$

(see [KRTZ20a, Lem. 4.1]). *Poles* are parameters for whom ϕ is not well-defined, i.e., $t \in \mathbb{R}$ such that $\prod_{i\in[n]} q_i(t) = 0$.

We adopt the definitions of [Brø83] of a supporting halfspace and supporting hyperplane for sets in $\mathbb{R}^n$. Let C be a non-empty set in $\mathbb{R}^n$. A *supporting halfspace* of C is a closed halfspace K in $\mathbb{R}^n$ such that $C \subset K$ and $H \cap C \neq \emptyset$, where H denotes the bounding hyperplane of K. A *supporting hyperplane* of C is a hyperplane H in $\mathbb{R}^n$ which bounds a supporting halfspace. If C is not contained in H, then H is a *proper supporting hyperplane*. Then, the convex hull of a set in $\mathbb{R}^n$, is the intersection of its supporting halfspaces.

Remark 5.1. *Let $I \subset \mathbb{R}$. For the convex hull conv$(\phi(I))$ to be a compact subset of $\mathbb{R}^n$, I has to be either a union of closed intervals not containing any poles of ϕ or a union of such closed intervals and intervals of the form $(-\infty, a], [a', +\infty)$, for $a \leq a'$, $a, a' \in \mathbb{R}$, if $\mathbf{p}_\infty$ exists.*

5.3 SUPPORT: a predicate for parametric curves in $\mathbb{R}^n$

Our algorithms are based on the following predicate that holds for curves in any dimension: Given the implicit equation of a hyperplane $\langle \mathbf{a}, \mathbf{x} \rangle + c = 0$, where $\mathbf{a}, \mathbf{x} \in \mathbb{R}^n$ and $c \in \mathbb{R}$, decide if it is a supporting hyperplane of $\phi(I)$. By definition, this holds if the hyperplane bounds a supporting halfspace of $\phi(I)$. Then, the convex hull of the intersection points of $\phi(I)$ and the hyperplane is a face on the boundary of the convex hull of $\phi(I)$ (see Fig. 5.5 for an example in two dimensions).

Given the parametric representation of the curve, we can reduce the support test for the hyperplane in question to a condition of sign-invariance; for the hyperplane defined by $\langle \mathbf{a}, \mathbf{x} \rangle + c = 0$ is supporting for $\phi(I)$ if and only if $\langle \mathbf{a}, \phi(t) \rangle + c \in \mathbb{R}(t)$ has at least one root in I and is sign-invariant in I. Therefore, we just have to isolate the real roots of $\langle \mathbf{a}, \phi(t) \rangle + c$, since this allows to determine its sign over I.

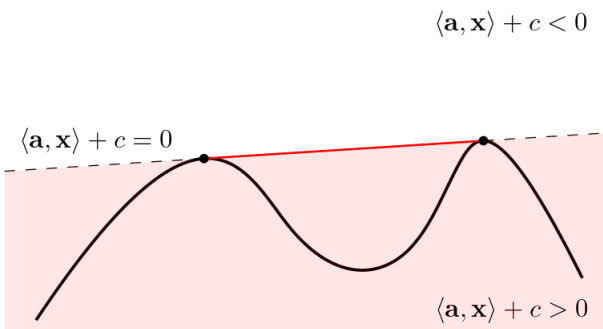


Figure 5.5: The line $\langle \mathbf{a}, \mathbf{x} \rangle + c = 0$ is supporting for $\phi(I)$; $\phi(I)$ lies entirely in the halfplane where $\langle \mathbf{a}, \mathbf{x} \rangle + c$ is positive. The line segment (in red) is on the boundary of the convex hull.

5.4 Convex hull of parametric curves in $\mathbb{R}^2$

Let $\mathcal{C}$ be a plane curve parametrized by $\phi(t) = (\phi_1(t), \phi_2(t)) = \left(\frac{p_1(t)}{q_1(t)}, \frac{p_2(t)}{q_2(t)}\right)$, defined as in Eq. (5.1).We further assume that the parametrization is *proper*. We consider $I \subseteq \mathbb{R}$, for whom $\phi(I)$ is a compact subset of $\mathbb{R}^2$ (see Rem. 5.1). We make the following simplifying assumptions:

Assumptions. *(i) The point at infinity, if it exists and it is in $\phi(I)$, is not endpoint of any segment. The reason is that, if this does not hold, we will not be able to obtain these segments as solutions of a polynomial system. We can reparametrize the curve to ensure that the point at infinity is not a segment's endpoint, using a Las Vegas algorithm in expected time $\widetilde{\mathcal{O}}_B(d^2 + d\tau)$ [KRTZ20a, Lem. 7].*

(ii) The real subset I has $k \in \mathcal{O}(1)$ boundary points of constant bitsize.

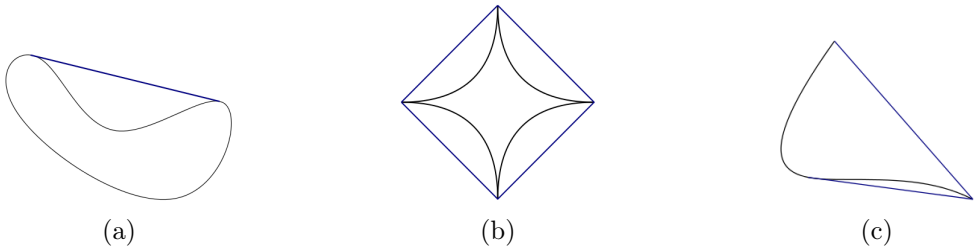


Figure 5.6: *Segments of type (a) I, (b) III and (c) IV and VI (in blue).*

We compute a *boundary description* of $\mathrm{conv}(\phi(I))$. Its facets are line segments joining two (or more) points on the curve. The boundary also consists of one-dimensional families of points on the curve, which are zero-dimensional faces; they assemble to smooth arcs of the curve (Fig. 5.1b). We will consider these arcs also as facets, by abuse of terminology.

In the next definition, we classify the line segments on the boundary of the convex hull in several types (see also Fig. 5.6). For the set $I \subset \mathbb{R}$, we denote by $\mathrm{bd}(I)$ its boundary.

Definition 5.2 (Types of line segments). *We distinguish the following types of line segments (I-VI) on the boundary of conv($\phi(I)$) according to their endpoints:*

I. bitangent segments: segments on the common tangent line of two or more smooth points of $\phi(I)$,

II. cusp-curve segments: segments connecting a cusp and a smooth point on the curve, lying on the latter point's tangent line,

III. cusp-cusp segments: segments whose endpoints are both cusps,

IV. endpoint-curve segments: segments connecting a point corresponding to a parameter in bd(I) and a smooth point on the curve, lying on the latter point's tangent line,

V. endpoint-cusp segments: segments connecting a point corresponding to a parameter in bd(I) and a cusp,

VI. endpoint-endpoint segments: segments connecting two points corresponding to parameters in bd(I).

5.4.1 Algorithm

Now, we develop the different steps of our algorithm that computes a boundary description of $\mathrm{conv}(\phi(I))$.

Step 1: Computing the segments. We compute all segments of types I to VI for $\phi(I)$ (Def. 5.2). They form a superset of the set of segments that are on the boundary of $\mathrm{conv}(\phi(I))$ (Fig. 5.4). Each segment has a type (I to VI) and it is determined by two parameter values, say a and b, such that $\phi(a)$ and $\phi(b)$ are the segment's endpoints respectively. We find the parameters that correspond to each type of segments by solving a polynomial system and we denote by $\mathcal{S}$ the set of all these pairs of parameters. We assume that for a given pair of parameters in $\mathcal{S}$ we can determine the type of the corresponding segment in constant time.

We consider the equations in $\mathbb{Z}(s,t)$

$$\begin{aligned} \langle \phi(s) - \phi(t), (-\phi_2'(s), \phi_1'(s)) \rangle &= 0\,, \\ \langle \phi(s) - \phi(t), (-\phi_2'(t), \phi_1'(t)) \rangle &= 0\,. \end{aligned} \tag{5.4}$$

We can easily verify that the pairs of parameters corresponding to segments of Types I to III satisfy these equations. We divide them by $(s-t)^2$, since it is always a factor and we let $H_1(s,t), H_2(s,t) \in \mathbb{Z}[s,t]$ be their numerators respectively. We now have to compute the isolated roots of the system $\{H_1(s,t) = H_2(s,t) = 0\}$ over $\mathbb{R}^2$. This system has as roots also the tuple of parameters that correspond to multiple points of the curve and the tuples of poles. The pairs that correspond to multiple points can be viewed as bitangents of length zero, and thus do not harm the correctness of the algorithm. As for the pairs of poles, they can easily be excluded, by checking if any parameter is also a root of $q_1(s) \cdot q_2(s)$.

For the segments of types IV and V, that have an endpoint of the form $\phi(a)$, with $a \in \mathrm{bd}(I)$, we isolate the roots of $H_1(s,a) \in \mathbb{Q}[s]$ and we consider all the pairs (s,a) such that s is a root of $H_1(s,a)$. For segments of type VI we just have to consider all pairs of parameters in $\mathrm{bd}(I)$.

In that way, we can construct the set $\mathcal{S}$, containing all the pairs of parameters.

Step 2: Decomposition of the curve. We subdivide I into a finite number of open intervals and points, where the points of the decomposition are the parameters of all pairs in $\mathcal{S}$. Let $R_t(s) = \mathtt{res}_t(H_1, H_2)$. The points of the decomposition are roots of the polynomial

$$P(s) := R_t(s) \prod_{a \in \mathrm{bd}(I)} (s-a) \cdot H_1(s,a)\,. \tag{5.5}$$

Every interval of the decomposition corresponds to a parametric arc that does not contain endpoints of any segment and thus, it is either contained in the boundary of the convex hull or in its interior.

Step 3: Computing the arcs. For every interval I_j of the decomposition of I computed in Step 2, we work as follows: We pick any rational $u \in I_j$; $\phi(u)$ is always a smooth point on $\mathcal{C}$ since the singular points are given by parameters that are among the points of

decomposition of I. We check if $\phi(u)$ is on the boundary of $\mathrm{conv}(\phi(I))$, which is the case if and only if there is a supporting line of $\phi(I)$ passing from $\phi(u)$ [Brø83, Thm.4.3]. But since $\phi(u)$ is smooth, the only possible supporting line is the tangent line to $\mathcal{C}$ at $\phi(u)$. So we call the predicate SUPPORT for the tangent line at $\phi(u)$. Its equation is

$$\langle (x,y) - \phi(u), (-\phi_2'(u), \phi_1'(u)\rangle = 0\,.$$

Let $\ell_u(\lambda) \in \mathbb{Q}[\lambda]$ be the numerator of the tangent line equation after substituting (x,y) with the parametrization $\phi(\lambda)$. We isolate the real roots of $\ell_u(\lambda)$ in order to determine its sign in I. If it is supporting to $\phi(I)$, then the parametric arc $\phi(I_j)$ is on the boundary of $\mathrm{conv}(\phi(I))$.

Step 4: Connecting the facets. Having determined the curved facets on the boundary at the previous step, we now describe how they connect with each other. In other words, we compute the segments, among the ones found in the first step, that are on the boundary of the convex hull. Two neighboring facets, intersect at a point of the curve. Since this point is on the boundary, there has to be a supporting line to the curve passing from it [Brø83, Thm.4.3]. If it is a smooth point, then the supporting line can only be the tangent line. When the point is not smooth, there is not a unique possible choice for the supporting line. So, for every arc of the curve that is on the boundary, we check each of its endpoints:

- If the endpoint is smooth, then the parametric arc is adjacent to the bitangent segment passing from this point.
- If the endpoint is not smooth but it belongs to a segment that has only one non-smooth point, then, the segment will be found with this procedure from the other one of its two adjacent curved facets.
- If the endpoint is not smooth and it belongs to a segment with two non-smooth endpoints, that is cusp-cusp (type III), endpoint-cusp (type IV) and endpoint-endpoint (type V) segment, we need to check every possible combination in order to find the ones that contribute to the boundary of the convex hull. Thus, for every such segment we check if it lies on a supporting line to the curve by calling the predicate SUPPORT.

  The parameters that correspond to cusp-cusp segments are also among the solutions of the system $\{H(s) = H(t) = 0\}$. The equation of the line passing through $\phi(s)$ and $\phi(t)$ is

  $$\langle \phi(s) - (x,y), (-(\phi_2(s) - \phi_2(t)), (\phi_1(s) - \phi_1(t))\rangle = 0\,.$$

  We substitute (x,y) with $\phi(\lambda)$ in the previous equation. Let $L(s,t,\lambda) \in \mathbb{Z}[s,t,\lambda]$ be the polynomial obtained after clearing the denominators . Calling the predicate

SUPPORT for all the lines connecting cusps, amounts to computing the isolated roots of

$$\begin{aligned} H(s) &= 0\,, \\ H(t) &= 0\,, \\ L(s,t,\lambda) &= 0\,. \end{aligned} \tag{5.6}$$

For the segments of type V, for all $a \in \mathrm{bd}(I)$, we compute the polynomial $L(a,t,\lambda) \in \mathbb{Q}[t,\lambda]$ and then solve the system $\{H(t) = L(a,t,\lambda) = 0\}$. For the endpoint-endpoint segments, we solve for λ the polynomials $L(a,b,\lambda) \in \mathbb{Q}[\lambda]$ for every $a, b \in \mathrm{bd}(I)$.

Output of the algorithm. The output is circular doubly linked list $\mathcal{L}$; this data structure consists of a sequence of records, each one corresponding to a facet of the convex hull. Every record contains the data required to describe the facet and a link to a previous and a next record. The data is a parameter interval, say $[t_1, t_2]$, together with a parametrization; the image of the interval over the parametrization is an arc of $\phi(I)$ or a segment on the boundary of $\mathrm{conv}(\phi(I))$. The link to the previous record points to the neighboring facet containing $\phi(t_1)$, and the link to the next record points to the neighboring facet containing $\phi(t_2)$. Note that the intervals that are contained in the data of each record are all bounded sub-intervals of I, except from at most one which may be of the form $(-\infty, a] \cup [b, +\infty)$ (Rem. 5.1).

Remark 5.3 (Degeneracies). *In a degenerate situation, several line segments on the boundary of the convex hull can lie on the same supporting line (e.g., Fig. 5.4). Consequently, each segment will be accounted for multiple times with different pairs of endpoints. However, this problem can be resolved during a post-processing stage, where the systems' solutions from the first step are analyzed to identify such situations.*

5.4.2 Complexity analysis

We introduce the following definition, which will serve in expressing the bit-complexity.

Definition 5.4. *We consider the zero-dimensional polynomial system*

$$\{F_1(X_1) = \cdots = F_N(X_N) = F(X_1, \ldots, X_N, Y) = 0\}$$

in $\mathbb{Z}[X_1, \ldots, X_N, Y]$. If all the polynomials have degree $\mathcal{O}(M)$ in each variable and bitsize in $\widetilde{\mathcal{O}}(\Lambda)$, then $C(M, \Lambda, N)$ is the bit-complexity of computing isolating intervals for the roots. $C(M, \Lambda, N)$ has the following properties:

1. *It is linear in Λ.*
2. *It is increasing with N.*

Step 1. We compute the polynomials H_1, H_2 in $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$: To construct the numerator of each H_i we perform multiplications of univariate polynomials in the variable s and in the variable t; this costs $\widetilde{\mathcal{O}}_B(d^2\tau)$ [vzGG13, Ch.8]. The bi-degree of the resulting expression is in $(\mathcal{O}(d), \mathcal{O}(d))$ and its bitsize in $\widetilde{\mathcal{O}}(\tau)$. Then, we divide by $(s-t)^2$ and this costs $\widetilde{\mathcal{O}}_B(d^3+d^2\tau)$ by adapting [vzGG13, Ex.10.21] to the bivariate case. The polynomials H_1 and H_2 are then of size $(\mathcal{O}(d), \widetilde{\mathcal{O}}(d+\tau))$.

Isolating the isolated roots of the system $\{H_1(s,t) = H_2(s,t) = 0\}$ costs $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$ [KS15]. If it is not zero-dimensional, then we have first to compute the gcd of H_1 and H_2 and their gcd-free parts. This is done in $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$. We can also compute the resultant of H_1 and H_2 with respect to s or t at no extra cost. Notice that both resultants are the same polynomial, since the system is symmetric. Let $R_t(s) = \mathtt{res}_t(H_1, H_2)$. It is of size $(\mathcal{O}(d^2), \mathcal{O}(d^2 + d\tau))$ [BPR06, Prop. 8.46]. For the segments of types IV-V, for every $a \in \mathrm{bd}(I)$, we construct the polynomial $H_a(t)$ by performing $\mathcal{O}(d)$ evaluations of polynomials of size $(\mathcal{O}(d), \widetilde{\mathcal{O}}(d+\tau))$ at a in $\widetilde{\mathcal{O}}_B(d(d+\tau))$ [BLPR15, Lem. 6]. Then we isolate its roots in $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$ [MSW15, Thm. 5].

Step 2. We decompose I at the endpoints of all the segments, which are roots of the polynomial P (Eq. (5.5)). Every endpoint is approximated by an isolating interval, so what it suffices, is that for two consecutive parameters in the sorted list, the respective isolating intervals do not overlap. The polynomial P is of size $(\mathcal{O}(d^2), \widetilde{\mathcal{O}}(d\tau))$ and we find isolating intervals of its roots in $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$ [MSW15, Thm. 5]. Every endpoint of an isolating interval is a rational of size $\widetilde{\mathcal{O}}(\sigma_i)$, and for all of them the bitsizes sum to $\widetilde{\mathcal{O}}(d^3\tau)$.

Step 3. We take a rational inside every interval of the decomposition, say u_i with $i = 1, \ldots, \mathcal{O}(d^2)$. The sum of their bitsizes is again in $\widetilde{\mathcal{O}}(d^3\tau)$. For every u_i, $\ell_{u_i}(\lambda) \in \mathbb{Q}[\lambda]$ is a polynomial of size $(d, \widetilde{\mathcal{O}}(d\sigma_i + \tau))$, so we isolate its roots in $\widetilde{\mathcal{O}}_B(d^3\sigma_i + d^2\tau)$ [MSW15, Thm. 5]. In the same bit-complexity we can decide if $\ell_{u_i}(\lambda)$ is non-negative in I. Summing for all i we get a total bit-complexity in $\widetilde{\mathcal{O}}_B(d^6\tau)$.

Step 4. To keep only the zero dimensional part of the solutions of the system in Eq. (5.6) we work as follows: Let $L(s,t,\lambda) = l_D(s,t)\lambda^D + \cdots + l_1(s,t)\lambda + l_0(s,t)$, where $D = \mathcal{O}(d)$. For $i = 1, \ldots, D$, we compute $R_i^s(s) = \mathtt{res}_t(l_i(s,t), H(t))$ and $R_i^t(t) = \mathtt{res}_s(l_i(s,t), H(s))$ in $\widetilde{\mathcal{O}}_B(d^5\tau)$ [LPR17, Lem. 4]. We compute the gcd of $R_0^s(s), \ldots, R_D^s(s)$ in $\widetilde{\mathcal{O}}_B(d^7 + d^6\tau)$ [KRTZ20a, Lem. 2] and then the gcd of the later with $H(s)$ in $\widetilde{\mathcal{O}}_B(d^3\tau)$ [BLM$^+$15, Lem. 4]. Let $\tilde{h}_s(s)$ the gcd-free part of $H(s)$. It has bitsize $\mathcal{O}(d+\tau)$ [BLM$^+$15, Lem. 4]. Analogously, we compute the gcd of $R_0^t(t), \ldots, R_D^t(t), H(t)$ and we let $\tilde{h}_t(t)$ be the gcd-free part of the last polynomial. Then, the system $\{\tilde{h}_s(s) = \tilde{h}_t(t) = L(s,t,\lambda) = 0\}$ is zero-dimensional and gives the isolated solutions of the system in Eq. (5.6). The bit-complexity of isolating its roots is $C(d, d+\tau, 2)$.

Examining the multiplicities of the roots, allows to find the pairs (s_0, t_0) for whom $L(s_0, t_0, \lambda)$ is sign-invariant in I, and therefore decide which segments are on the boundary of the convex hull. For the segments of type V, for all $a \in \mathrm{bd}(I)$, we compute the polynomial $L(a, t, \lambda)$ and then solve the system $\{H(t) = L(a, t, \lambda) = 0\}$. Since the number of boundary points is assumed to be in $\mathcal{O}(1)$, this cost is $C(d, \tau, 1)$. Step 4 requires also manipulating the output data structure; we perform $\mathcal{O}(d^2)$ updates, each one in constant time.

Corollary 5.5. *The bit-complexity of the algorithm is in $\widetilde{\mathcal{O}}_B(d^7 + d^6\tau) + C(d, d + \tau, 2)$.*

In the theorem that follows we summarize our result.

Theorem 5.6. *Let $\mathcal{C}$ be a curve with a proper parametrization $\phi(t)$ as in Eq. (5.1), of size (d, τ). Let $I \subset \mathbb{R}$ such that $\phi(I)$ is compact in $\mathbb{R}^2$. There is an algorithm that computes the convex hull of $\phi(I)$ in $\widetilde{\mathcal{O}}_B(d^7 + d^6\tau) + C(d, d + \tau, 2)$. The output of the algorithm is a circular doubly linked list, containing the parameter intervals and the corresponding parametrizations for each facet; the image of the interval over the parametrization is a parametric arc or a segment on the boundary of conv$(\phi(I))$. The circular doubly linked list has $N \in \mathcal{O}(d^2)$ elements.*

Proof. The correctness of the algorithm is based on the fact that the Support predicate, correctly detects the curve branches that are on the boundary of the convex hull (Step 3). On connecting the curve branches with segments, we refer to the selection criterion of the gift-wrapping algorithm for the set of points $\phi(I)$; given a point p on the boundary of the convex hull, the next point with whom it connects, is a point $q \in \phi(I)$ such that the line that passes from p and q is strictly supporting for the convex hull. The connection phase in Step 4 respects this criterion.

For the size of the output, we have that $N \in \mathcal{O}(d^2)$ by taking into account the degrees of the systems entailed in the computation of segments of the different types and Assumption 5.4(ii). □

Remark 5.7. *If the curve is smooth, without self-intersections, we have that $N \in \mathcal{O}(d)$. [EKH01]; this is since there are $\mathcal{O}(d)$ inflection points and poles. If there is a bitangent connecting two smooth points $\phi(t_1)$, $\phi(t_2)$ with $t_1 < t_2$ then in the interval $[t_1, t_2]$, there exists at least one parameter that corresponds to an inflection point or a pole. Moreover, since there are $\mathcal{O}(d)$ cusps, the number of Type II and III segments that are on the boundary of the convex hull is also in $\mathcal{O}(d)$. The segments of Type IV, V, VI are already in $\mathcal{O}(1)$ (Assumption 5.4(ii)).*

Corollary 5.8. *Using the bit-complexity results of Ch. 3 (see Rem. 3.8) for $C(d, d + \tau, 2)$, we have that the boundary of the convex hull of a plane parametric curve can be computed in $\widetilde{\mathcal{O}}_B(d^{10} + d^9\tau)$ bit-operations.*

5.5 Convex hull of parametric curves in $\mathbb{R}^3$

Let $\phi(t) = (\phi_1(t), \phi_2(t), \phi_3(t)) = \left(\frac{p_1(t)}{q_1(t)}, \frac{p_2(t)}{q_2(t)}, \frac{p_3(t)}{q_3(t)}\right)$, defined as in Eq. (5.1). Again, the ϕ is proper and the polynomials p_i, q_i have degree at most d and bitsize τ, $i = 1, 2$. We consider $I \subseteq \mathbb{R}$, for whom $\phi(I)$ is a compact subset of $\mathbb{R}^3$ (see Rem. 5.1). We make again the simplifying Assumptions 5.4 of Sec. 5.4.

Our goal is to compute a boundary description of $\mathrm{conv}(\phi(I))$. The facets of the convex hull are triangles with vertices points on the curve. In addition, it includes one-dimensional families of segments with endpoints on the curve, which are one-dimensional faces. These families assemble to two-dimensional surface patches, to which we refer from now on as facets, with abuse of terminology.

The next definition classifies the triangles on the boundary of the convex hull in several types (see Fig. 5.7).

Definition 5.9. *We distinguish the following types of triangles on the boundary of conv$(\phi(I))$ according to their vertices:*

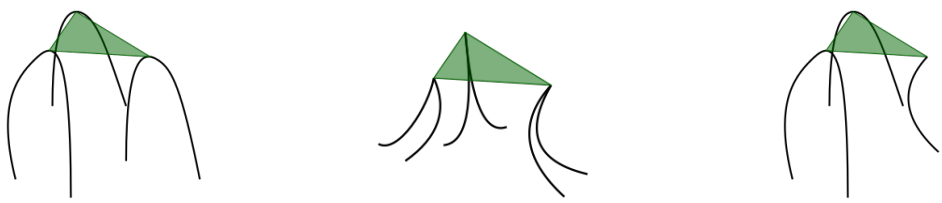
I. tangent triangles: there is at least one vertex that is a smooth point of $\mathcal{C}$ and the triangle lies on a tangent plane to $\mathcal{C}$ at this point. The other two vertices of the triangle are either smooth points, at which the plane is also tangent, or cusps,

II. cuspidal triangles: the three vertices of the triangle are cusps,

III. endpoint triangles: there is at least one vertex of the triangle that is given by a parameter in bd(I).

Remark 5.10. *If we consider all the possible combinations of the three vertices of a triangle (smooth point, cusp, endpoint), we see that there exist ten different configurations. Among them, three correspond to tangent triangles, one to cuspidal triangle and six to endpoint triangles. We divide them in the three categories of Def. 5.9 due to the following reasons:*

- *To define uniquely a plane that passes from a smooth point of $\mathcal{C}$ and is tangent to the curve at this point, we need to specify another point belonging on the plane. Therefore,*



(a) (b) (c)

Figure 5.7: In green (a) a tangent triangle, (b) a cuspidal triangle and (c) an endpoint triangle.

the plane in which the tangent triangle belongs is not affected by the smoothness (or not) of the two other vertices.

- *The cuspidal and endpoint triangles can be obtained as solutions of a polynomial system of special structure, which is simpler comparing to the system corresponding to tangent triangles.*

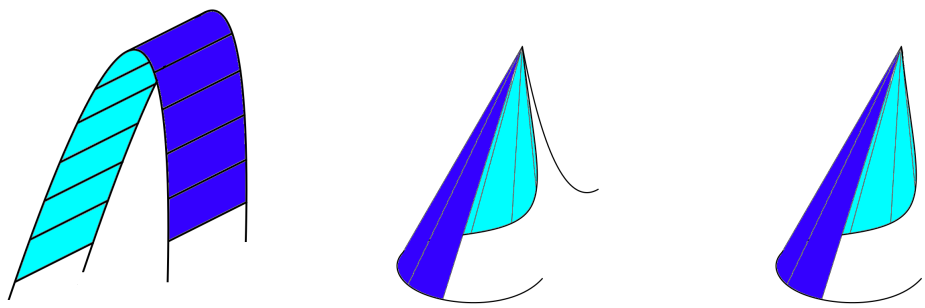
The one-dimensional families of segments on the boundary of the convex hull form developable surface patches, that can be also classified (see Fig. 5.8).

Definition 5.11. *We distinguish the following types of surface patches according to the vertices of the spanning segments:*

I. bitangent surface patches: The segments' endpoints are smooth points of $\mathcal{C}$ and the segment lies on the plane tangent to the curve at these two points,

II. cuspidal surface patches: All segments have a common endpoint that is a cusp,

III. endpoint surface patches: All segments have a common endpoint that is given by a parameter in bd(I).



(a) (b) (c)

Figure 5.8: (a) A bitangent surface patch, (b) a cuspidal surface patch and (c) an endpoint surface patch.

The pairs of parameters that give these type of segments, are points of a plane curve. Let $E(s) := \prod_{a \in \mathrm{bd}(I)}(s-a)$. We define the *bisecant curve* as the zero set of the equation

$$G(s,t) := \frac{\det(\phi(s)-\phi(t), \phi'(s), \phi'(t)) \cdot E(s) \cdot E(t)}{(s-t)^4} = 0\,. \tag{5.7}$$

In the previous equation, we divide by $(s-t)^4$ since, as it can be shown, it is a factor of the numerator. We multiplied by both $E(t)$ and $E(s)$ to preserve the symmetry. A point $(s,t) \in \mathbb{R}^2$ satisfying Eq. (5.7) is such that:

- $\phi(s) - \phi(t), \phi'(t), \phi'(s) \neq \mathbf{0}$ and there is a bitangent plane to the curve at $\phi(s)$ and $\phi(t)$, or
- $\phi(s) = \phi(t)$ and/or $\phi'(t) = \mathbf{0}$ and/or $\phi(s) = \mathbf{0}$, i.e., $\phi(t)$ is a multiple point and/or $\phi(t)$ is a cusp and/or $\phi(s)$ is a cusp, or
- $s \in \mathrm{bd}(I)$ or $t \in \mathrm{bd}(I)$.

Now, we consider all the segments $\overline{\phi(s)\phi(t)}$, with s and t satisfying $G(s,t) = 0$. This gives rise to the *bisecant surface* in $\mathbb{R}^3$:

$$\mathcal{B} = \left\{ T\,\phi(s) + (1-T)\,\phi(t) \,|\, T \in [0,1], G(s,t) = 0 \right\}. \tag{5.8}$$

The bisecant surface is *ruled*, since through every point of the surface there is a straight-line segment that lies on it, and it is also *developable* [Bla05, §5.1]. The surface patches of Def. 5.11 are part of this surface.

Only certain parts of the bisecant surface are on the boundary of the convex hull. We will use the bisecant curve to find and describe these parts.

5.5.1 Algorithm

We now describe the steps of our algorithm.

Step 1: Computing the triangles. We compute all triangles of types I to III for $\phi(I)$ (Def. 5.9). They form a superset of the set of triangles that are on the boundary of $\mathrm{conv}(\phi(I))$. Each triangle has a type (I to III) and is determined by three parameter values, say a, b and c, such that $\phi(a), \phi(b)$ and $\phi(c)$ are the triangle's vertices. We will find the parameters that correspond to each type of triangles by solving a polynomial system and we will denote by $\mathcal{S}$ the set of all such triples. We assume that for a given triple of parameters in $\mathcal{S}$ we can determine the type of the corresponding triangle in constant time.

To find the different types of triangles, it is easier to factorize G (Eq. (5.7)) and consider each type separately. We know that cusps of $\mathcal{C}$ are given by parameters that are among the roots of $H(s)$ (Eq. (5.3)). Therefore, in the case where there are cusps, the polynomial G can be factorised as $\hat{H}(s) \cdot \hat{H}(t) \cdot \hat{G}(s,t) \cdot E(s) \cdot E(t)$, where $\hat{H}$ corresponds to the parameters that give cusps. We consider the equations in $\mathbb{Z}(s,t,u)$:

$$\frac{\det(\phi(u) - \phi(t), \phi'(t), \phi'(s))}{(s-t)(t-u)^2} = 0\,, \tag{5.9}$$

$$\frac{\det(\phi'(s), \phi'(t), \phi'(u))}{(s-t)(t-u)(s-u)} = 0\,. \tag{5.10}$$

Let $(s,t) \in \mathbb{R}^2$ such that $\hat{G}(s,t) = 0$. Then, $\phi(s)$ and $\phi(t)$ are both smooth points, since we have excluded the factors that correspond to cusps. Let $\mathcal{P}$ be the common tangent

plane passing from $\phi(s)$ and $\phi(t)$. Then, Eq. (5.9) expresses the fact that the plane $\mathcal{P}$ intersects the curve at $\phi(u)$. Eq. (5.10) expresses the condition that the three tangent vectors, $\phi'(s), \phi'(t)$, and $\phi'(u)$ lie on the same plane and so $\mathcal{P}$ is also tangent to $\mathcal{C}$ at $\phi(u)$, if it is smooth. Let $H_1, H_2 \in \mathbb{Z}[s,t,u]$ be the numerators of Eq. (5.9) and Eq. (5.10) respectively. From the isolated roots of the system $\{\hat{G} = H_1 = H_2 = 0\}$ we can obtain the tangent triangles with three or two smooth vertices (when $\phi(u)$ is a cusp).

We consider the equations in $\mathbb{Z}(s,t,u)$:

$$\frac{\det(\phi(u)-\phi(s), \phi(u)-\phi(t), \phi'(u))}{(t-u)^2(s-u)^2(s-t)} = 0\,,$$
$$\frac{\det(\phi(u)-\phi(s), \phi(u)-\phi(t), \phi'(s))}{(t-u)(s-u)^2(s-t)^2} = 0\,.$$

Let $H_3, H_4 \in \mathbb{Z}[s,t,u]$ be their numerators respectively. For the tangent triangles with one smooth vertex and two cusps, we solve the system:

$$\begin{aligned} H(s) &= 0\,, \\ H(t) &= 0\,, \\ H_3(s,t,u) &= 0\,. \end{aligned} \tag{5.11}$$

The system has also as roots the parameters corresponding to cuspidal triangles, since $\phi(u)$ can be a cusp. For the endpoint triangles, for every $a \in \mathrm{bd}(I)$, to find the planes that go through it, we solve the system

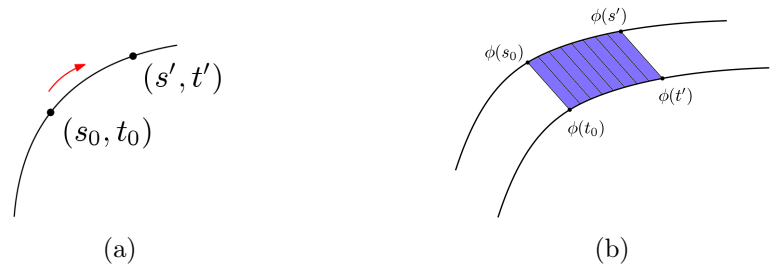
$$\begin{aligned} H_1(s,t,a) &= 0\,, \\ H_4(s,t,a) &= 0\,. \end{aligned} \tag{5.12}$$

For any $a, b \in \mathrm{bd}(I)$, to find the planes that go through $\phi(a), \phi(b)$, we solve the equation

$$H_4(s,a,b) = 0\,. \tag{5.13}$$

Step 2: Decomposition of the bisecant curve. We consider the graph of G and we compute a special purpose Cylindrical Algebraic Decomposition (CAD) of the s-axis. In a CAD we decompose the s-axis into a finite number of points and open intervals delimited by these points over which the graph of the bisecant curve G has a cylindrical structure, i.e., the number of branches of the graph of the curve is constant. We call the points defining the decomposition *special values*. Special fibers are the points on the curve above the special values. Regular fibers are the points on the curve above additional points between two special values.

In a CAD the special values are the projections of the s-critical points and the vertical asymptotes. We refine the decomposition by further subdividing the intervals at the



(a) (b)

Figure 5.9: (a) An arc of the graph of the bisecant curve G and (b) the surface patch that corresponds to the part of the arc between (s_0, t_0) and (s', t') (in purple).

projections of the t-critical points and at the parameters of all triples corresponding to the triangles computed in the previous step.

Let (s_0, t_0) be a point on the graph of G, such that the line segment connecting $\phi(s_0)$ and $\phi(t_0)$ is on the boundary of the convex hull and is on a bisecant surface patch. If we move along the arc of the graph of G around the point (s_0, t_0) and we take a neighboring point (s', t') that is sufficiently close, then, the segment $\overline{\phi(s')\phi(t')}$ is also on the boundary of the convex hull, part of the same bisecant surface patch (see Fig. 5.9). In particular, by moving along the arc in both directions starting from (s_0, t_0), we get bitangent segments that are on the bisecant surface patch, and thus, obtain a way to trace it. We continue until we find point (s'', t'') that corresponds to a segment that is on the intersection of the bisecant surface patch with another facet. This point can be such that:

- $\phi(s'')$ and $\phi(t'')$ are vertices of triangle (so, s'' and t'' are both special values of the decomposition), or
- (s'', t'') is a self-intersection point of G, i.e., there are two surface patches that intersect at the segment with endpoints $\phi(s'')$ and $\phi(t'')$. So, in this case s'' is an s-critical point and therefore a special value of the decomposition.

All the previous discussion suggests that an arc of the decomposition of G (that is obtained by the decomposition of the s-axis) corresponds to a surface patch that is either on the boundary of the convex hull or in its interior.

Step 3: Computing the bisecant surface patches. We want to find the arcs of G that correspond to the bisecant surface patches on the boundary of the convex hull. The following lemma gives a criterion. It is a direct consequence of the discussion in Step 2.

Lemma 5.12. *We consider a smooth and monotonous arc on the graph of G, with $A = (s_1, t_1)$ and $B = (s_2, t_2)$ its endpoints. If there is no point on the arc that corresponds to a bitangent segment belonging on a triangle, then, if any interior point on the arc corresponds to a bitangent segment on the boundary of the convex hull then the entire arc corresponds to a bitangent surface patch on the boundary of the convex hull.*

Over every open interval of the decomposition of the s-axis there are several arcs of the graph of G, corresponding to bitangent surface patches. From Lem. 5.12, in order to determine if a patch is on the boundary it suffices to take an interior point of the arc and check if the corresponding tangent plane is supporting for $\phi(I)$, using the SUPPORT predicate.

For every (open) interval I_j of the decomposition, we work as follows: We pick any rational $u \in I_j$ and we solve $G(u,t) = 0$. For every v such that $G(u,v) = 0$, we have that $\phi(u)$ and $\phi(v)$ are always smooth points on $\mathcal{C}$ since the singular points are given by parameters that are among the points of decomposition of the s-axis. We check if the segment connecting $\phi(u)$ and $\phi(v)$ is on the boundary of $\mathrm{conv}(\phi(I))$, which is the case if and only if there is a supporting plane of $\phi(I)$ passing from $\phi(u)$ and $\phi(v)$. But since both points are smooth, the only possible supporting plane is the tangent plane to $\mathcal{C}$ at $\phi(u)$ and $\phi(v)$. So we call the predicate SUPPORT for this plane. Its equation is

$$\det((x,y,z) - \phi(u), \phi'(u), \phi'(v)) = 0\,.$$

Let $\ell_{u,v}(\lambda) \in \mathbb{Q}[\lambda]$ be the numerator of the tangent line equation after substituting (x,y) with the parametrization $\phi(\lambda)$. We isolate the real roots of $\ell_{u,v}(\lambda)$ in order to determine its sign in I. If the plane is supporting to $\phi(I)$ then the arc of G over I_j, containing the point (u,v), corresponds to a bisecant surface patch that is on the boundary of $\mathrm{conv}(\phi(I))$.

Step 4: Connecting the facets. Having determined the bisecant surface patches on the boundary at the previous step, we now describe how they connect with each other. In other words, we compute the triangles, among the ones found in the first step, that are on the boundary of the convex hull. Two neighboring facets intersect at a line segment with endpoints that are points on the curve. Let $\phi(s)$ and $\phi(t)$ be the endpoints of such a segment. Since this segment is on the boundary, there has to be a supporting plane to the curve containing it. If at least one point is smooth, then the supporting plane can only be the plane tangent to the curve at the smooth point and passing from both. If none of the points is smooth, then the supporting plane is not uniquely defined.

So, for every arc of the graph of G corresponding to a bisecant surface patch on the boundary, we check each of its endpoints. Let (s,t) be one of them.

1. If at least one of $\phi(s)$ and $\phi(t)$ is a smooth point, say $\phi(s)$, then the plane that is tangent to $\mathcal{C}$ at $\phi(s)$ and passes from $\phi(t)$ is uniquely defined. This plane passes from a third point $\phi(u)$ on $\mathcal{C}$, that was found in Step 1. So, there is only one possible plane facet (if there are no degeneracies) that is neighboring, and this is the triangle with vertices $\phi(s)$, $\phi(t)$ and $\phi(u)$. In a degenerate situation, there will be more than one u, corresponding to the multiple interections of the plane with the curve, and the different triangles that belong on the same plane will be considered multiple times

(see Rem. 5.13).

2. If both $\phi(s)$ and $\phi(t)$ are cusps or endpoints of $\phi(I)$, then we need to check every possible combination in order to find the triangle containing this segment that is on boundary of the convex hull.

   For the cuspidal triangles, let $(s,t,u) \in \mathbb{R}^3$ be a triple of parameters corresponding to cusps. Then,

$$\langle (\phi(s) - \phi(u)) \times (\phi(s) - \phi(t)), \phi(s) - (x,y,z) \rangle = 0$$

   is the implicit equation of the plane in $\mathbb{R}^3$ that goes through $\phi(s), \phi(t)$ and $\phi(u)$. We substitute (x,y,z) with $\phi(\lambda)$ in the previous equation. Let $L(s,t,u,\lambda) \in \mathbb{Z}[s,t,u,\lambda]$ be the polynomial obtained after clearing the denominators. Calling the predicate Support for all these planes, amounts to isolating the roots of the system

$$\begin{aligned} H(s) &= 0\,, \\ H(t) &= 0\,, \\ H(u) &= 0\,, \\ L(s,t,u,\lambda) &= 0\,. \end{aligned} \tag{5.14}$$

   For the triangles with non-smooth vertices that involve an endpoint, we distinguish three cases for the system that we solve:

   - If one vertex is an endpoint and the other two cusps, the system is of the form $\{H(s) = H(t) = L(s,t,a,\lambda) = 0\}$, where $a \in \mathrm{bd}(I)$,
   - If two vertices are endpoints and the third one is a cusp, the system is of the form $\{H(s) = L(s,a,b,\lambda) = 0\}$, where $a, b \in \mathrm{bd}(I)$,
   - If all the vertices are endpoints, then we just have to solve the univariate equation $L(a,b,c,\lambda) = 0$, where $a, b, c \in \mathrm{bd}(I)$.

So, for a surface patch that corresponds to an arc of the graph of G with endpoints (s_1,t_1) and (s_2,t_2), the neighboring facets from each side can be determined.

Output of the Algorithm. We will use the half-edge data structure, or doubly connected edge list (DCEL), to represent the output [dBCvKO08, Ch.2.2]. This data structure is used to represent an embedding of a planar graph in the plane by maintaining the following records: vertices, edges and faces. By extension, it is widely used to describe the boundary of three-dimensional convex polyhedra and is also convenient to describe the boundary of the convex hull of $\phi(I)$. In our case, a face is a 2-dimensional facet of the convex hull, that can be either a triangle or a bisecant surface patch. An edge is intersec-

tion of two facets, i.e, a parametric arc or a segment connecting two points of the curve. A vertex is the intersection of two edges of the convex hull, i.e., a point on the curve.

The principle of this data structure is to 'decompose' every edge into two half-edges with opposite directions. All the records are associated to a half-edge and this allows to encode all the combinatorial information of the planar graph, or of the boundary of the convex hull in the present case. In particular:

- a vertex is associated to one (arbitrary) halfedge with the vertex as starting point,
- an edge is associated to one halfedge,
- a face is associated to some halfedge on its boundary,
- a halfedge is associated to the vertex that is its origin, to a "twin" halfedge, that is the halfedge with the opposite direction, and to the face that is incident to the edge on the left side, when we traverse it from the origin to its endpoint.

Although it is a combinatorial data structure, we can also store geometrical information on the records, by giving them extra attributes:

- For a vertex, since it a point on the curve, we store the corresponding parameter,
- For an edge, if it is a parametric arc we store the corresponding parameter interval. If it is a bitangent segment, it is described by the two parameters that correspond to the endpoints of the segment,
- A face can be either a triangle or a bisecant surface patch. In the first case, it is described by the parameters of the incident vertices. In the second case, it is described by the reference to its corresponding arc of the graph of G. An arc is described by its endpoints, its projection on the s-axis and its ordering with respect to other branches over the interval.

Remark 5.13 (Degeneracies). *In a degenerate situation, several triangles on the boundary of the convex hull can lie on the same supporting plane (e.g., Fig. 5.13b) or several surface patches can be included in a bigger surface patch. Consequently, each boundary element will be accounted for multiple times. However, this problem can be resolved during a post-processing stage, where the systems' solutions from the first step are analyzed to identify such situations.*

5.5.2 Complexity analysis

In this subsection, we analyse the bit-complexity of the algorithm step by step. Again, we use $C(M, \Lambda, N)$ (Def. 5.4) to express it.

Step 1. We construct the polynomials G, H_1, H_2, H_3, H_4 in $\widetilde{\mathcal{O}}_B(d^3\tau)$ [vzGG13, Ch. 8]. We factorize G as $\hat{H}(s) \cdot \hat{H}(t) \cdot \hat{G}(s,t) \cdot E(s) \cdot E(t)$ in $\widetilde{\mathcal{O}}_B(d^4 + d^3\tau)$ [DDR+22, Prop. 21], by considering $G(s,t)$ as a polynomial in s or in t and then finding the gcd of the coefficients.

To find the triangles with two or three smooth vertices we need to find the isolated roots of the system $\{\hat{G}(s,t) = H_1(s,t,u) = H_2(s,t,u) = 0\}$. In particular, we are only interested in the first two coordinates of the roots; if (a,b,c) is a solution to the system and $\phi(u)$ is smooth, then all the possible permutations of a,b,c are solutions as well. So, we can obtain the triples corresponding to tangent planes with smooth vertices just by the (s,t)-projections. We take the resultant $R_1(s,t) := \texttt{res}_u(H_1, H_2) \in \mathbb{Z}[s,t]$. It is a polynomial of degree $\mathcal{O}(d^2)$ in each variable and bitsize in $\widetilde{\mathcal{O}}(d\tau)$ [BPR06, Prop. 8.72]. It is computed in $\widetilde{\mathcal{O}}_B(d^7\tau)$. Then, we consider the system $\{\hat{G} = R_1 = 0\}$. We take the resultant $R_2(s) := \texttt{res}_t(\hat{G}, R_1) \in \mathbb{Z}[s]$. It is of size $\left(\mathcal{O}(d^3), \widetilde{\mathcal{O}}(d^2\tau))\right)$ [KS15, Cor. 5] and is computed in $\widetilde{\mathcal{O}}_B(d^8\tau)$ [KS15, Lem. 6]. At last, we isolate the roots of $\{R_2 = \hat{G} = 0\}$ in $\widetilde{\mathcal{O}}_B(d^9 + d^8\tau)$ [KS15] (if the system is not zero-dimensional we first compute the gcd and the gcd-free parts).

Tangent triangles with two smooth vertices can be found by computing the isolated roots of the system $\{\hat{G}(s,t) = H(u) = H_1(s,t,u) = 0\}$. In particular, we are only interested this time in the last two coordinates of the roots; if (a,b,c) is a solution to the system and $\phi(a), \phi(b)$ are smooth, then (b,a,c) is a solution as well. Thus, we can group the triples corresponding to the same triangle. The resultant $R_3(s,u) := \texttt{res}_t(\hat{G}, H_1) \in \mathbb{Z}[s,u]$ is of size $(\mathcal{O}(d^2), \widetilde{\mathcal{O}}(d\tau))$ and is computed in $\widetilde{\mathcal{O}}_B(d^7\tau)$. We isolate the roots of $\{H(u) = R_3(s,u) = 0\}$ in $\widetilde{\mathcal{O}}_B(d^8 + d^7\tau)$.

For the triangles with one smooth point and the cuspidal triangles, we find the isolated roots of the system of Eq. (5.11) in $\widetilde{\mathcal{O}}_B(d^7 + d^6\tau)$ in the same way as for the system in Eq. (5.6).

At last, for every $a \in \mathrm{bd}(I)$, we compute $H_1(s,t,a)$ and $H_4(s,t,a)$ by performing $\mathcal{O}(d^2)$ evaluations at a; this costs $\widetilde{\mathcal{O}}_B(d^3\tau)$ and the resulting polynomials have bitsize in $\widetilde{\mathcal{O}}(d+\tau)$ [BLPR15, Lem.7]. Then, we solve the bivariate systems of Eq. (5.12) in $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau)$. For all $a, b \in \mathrm{bd}(I)$, we compute $H_4(s,a,b)$ (Eq. (5.13)) in $\widetilde{\mathcal{O}}_B(d^3\tau)$ [BLPR15, Lem.7]. These polynomials have bitsize in $\widetilde{\mathcal{O}}(d+\tau)$ and so we isolate the roots in $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$ [MSW15, Thm. 5].

Step 2. The decomposition points of the s-axis are:

- s-projections of critical points of G, and thus roots of $\texttt{res}_t(G, \partial G/\partial t)$. The graph of G is symmetric with respect to the line $s = t$, so we do not consider also $\texttt{res}_t(G, \partial G/\partial s)$,

- The s-values where vertical asymptotes occur. These values are roots of the leading coefficient of $G(s,t)$, when considered as a polynomial in t,

- parameters corresponding to a triangle, and thus roots of

$$H(s) \cdot R_2(s) \prod_{a \in \mathrm{bd}(I)} \left((s-a) \cdot \mathtt{res}_t(H_1(s,t,a), H_4(s,t,a)) \prod_{b \in \mathrm{bd}(I)} H_4(s,a,b) \right).$$

The product of these polynomials is of size $(\mathcal{O}(d^3), \widetilde{\mathcal{O}}(d^2\tau))$. We isolate its roots in $\widetilde{\mathcal{O}}_B(d^9 + d^8\tau)$ [MSW15, Thm. 5]. Every endpoint of an isolating interval is a rational of size $\widetilde{\mathcal{O}}(\sigma_i)$, and for all of them the bitsizes sum to $\widetilde{\mathcal{O}}(d^5\tau)$.

Step 3. We take a rational point q_i inside every interval of the decomposition (e.g. the midpoint); it has bitsize $\widetilde{\mathcal{O}}(\sigma_i)$. We construct the polynomial $G(q_i, t)$ by performing $\mathcal{O}(d)$ evaluations of univariate polynomials at q_i. Then, calling the predicate SUPPORT for every intersection point on the graph of G, amounts to isolating the roots of the system

$$\begin{aligned} G(q_i, t) &= 0 \\ H_1(t, q_i, \lambda) &= 0 \end{aligned} \tag{5.15}$$

This costs $C(d, d\sigma_i, 1)$. Summing for all i, since C is linear in the bitsize, we obtain $C(d, d^6\tau, 1)$.

Step 4. To keep only the zero dimensional part of the solutions of the system in Eq. (5.14) we work as follows: Let $L(s,t,u,\lambda) = l_D(s,t,u)\lambda^D + \cdots + l_1(s,t,u)\lambda + l_0(s,t,u)$, where $D = \mathcal{O}(d)$. For $i = 1, \ldots, D$, we compute

$$\begin{aligned} R_i^s(s) &= \mathtt{res}_u(\mathtt{res}_t(l_i(s,t,u), H(t)), H(u)) \in \mathbb{Z}[s]\,, \\ R_i^t(t) &= \mathtt{res}_u(\mathtt{res}_s(l_i(s,t,u), H(s)), H(u)) \in \mathbb{Z}[t]\,, \\ R_i^u(u) &= \mathtt{res}_t(\mathtt{res}_s(l_i(s,t,u), H(s)), H(t)) \in \mathbb{Z}[u]\,. \end{aligned}$$

This is done in $\widetilde{\mathcal{O}}_B(d^8\tau)$ [KS15, Lem. 6]. by successive resultant computations. They are polynomials of size $\left(\mathcal{O}(d^3), \widetilde{\mathcal{O}}(d^2\tau)\right)$ [BPR06, Prop.8.72]. For $v = s, t, u$, we compute the gcd of $R_0^v(v), \ldots, R_D^v(v)$ in $\widetilde{\mathcal{O}}_B(d^{10} + d^9\tau)$ [KRTZ20a, Lem. 2] and then the gcd of the later with $H(v)$ in $\widetilde{\mathcal{O}}_B(d^6\tau)$ [BLM+15, Lem. 4]. Let $\tilde{h}_v(v)$ the gcd-free part of $H(v)$. It has bitsize $\mathcal{O}(d+\tau)$ [BLM+15, Lem. 4]. Then, the system $\{\tilde{h}_s(s) = \tilde{h}_t(t) = \tilde{h}_u(u) = L(s,t,u,\lambda) = 0\}$ is zero-dimensional and gives the isolated solutions of the system in Eq. (5.14). The bit-complexity of isolating its roots is $C(d, d+\tau, 3)$. Computing the isolated roots of the other systems in this step is dominated by the previous computations.

Corollary 5.14. *The bit-complexity of the algorithm is in* $\widetilde{\mathcal{O}}_B(d^{10} + d^9\tau) + C(d, d^6\tau, 1) + C(d, d+\tau, 3)$.

In the theorem that follows we summarize our result.

Theorem 5.15. *Let $\mathcal{C}$ be a curve in $\mathbb{R}^3$ with a proper parametrization $\phi(t)$ as in Eq. (5.1), of size (d, τ). Let $I \subset \mathbb{R}$ such that $\phi(I)$ is compact in $\mathbb{R}^3$. There is an algorithm that computes the boundary of the convex hull of $\phi(I)$ in*

$$\widetilde{\mathcal{O}}_B\left(d^{10} + d^9\tau\right) + C(d, d^6\tau, 1) + C(d, d+\tau, 3)\,.$$

The output of the algorithm is a doubly connected linked list describing the facets of conv$(\phi(I))$, which are $\mathcal{O}(d^3)$.

Proof. The correctness of the algorithm is based on Lem. 5.12 and the fact that the Support predicate correctly detects the bisecant surface patches that are on the boundary. On connecting the bisecant surface patches with triangles, we refer to the selection criterion of the gift-wrapping algorithm for the set of points $\phi(I)$. Given a segment that is on the boundary of the convex hull (and also on the boundary of a surface patch), the next point with whom it connects, is a point $q \in \phi(I)$ such that the plane that contains the segment and passes from q is strictly supporting for the convex hull. The connection phase in Step 4 respects this criterion. For the number of the facets on the boundary of the convex hull, we consider the degrees of the systems entailed in the computation of triangles and Assumption 5.4(ii).

□

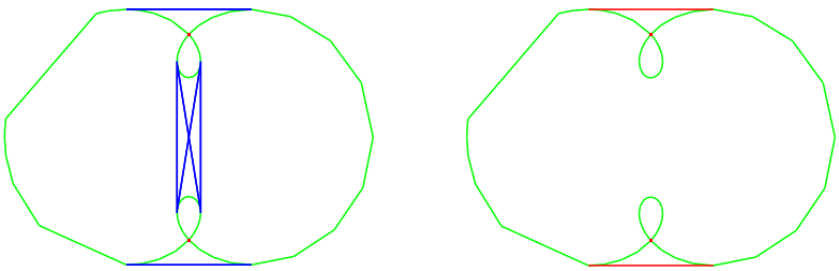
Corollary 5.16. *Using the bit-complexity results of Ch. 3 (see Rem. 3.8) for $C(d, d+\tau, 3)$ and [DDR+22] for $C(d, d^6\tau, 1)$, we have that the boundary of the convex hull of a space parametric curve can be computed in $\widetilde{\mathcal{O}}_B(d^{13} + d^{12}\tau)$ bit-operations.*

5.6 Implementation and Examples

In this section, we provide some examples using our prototype implementation in Maple. It is built upon the real root isolation routines of Maple's RootFinding library and the `PTOPO` package [KRTZ20b], to compute the topology and visualize parametric curves in two and three dimensions.

Example 5.17. We consider a curve in $\mathbb{R}^2$ parametrized by

$$\phi(t) = \left(-\frac{2\left(12t^2 + 7t - 12\right)\left(1679t^4 + 2688t^3 - 5074t^2 - 2688t + 1679\right)}{15625\left(t^2+1\right)^3}, \right.$$
$$\left. \frac{(t-7)(7t+1)\left(2929t^4 + 2688t^3 - 2574t^2 - 2688t + 2929\right)}{15625\left(t^2+1\right)^3} \right).$$



(a) (b)

Figure 5.10: (a) The curve of Example 5.17 (in green) and the segments computed at the first step of the algorithm (in blue). (b) The segments on the boundary of the convex hull (in red).

The parametrization is proper. We take $I = \mathbb{R}$. We compute the polynomials

$$\begin{aligned}
H_1(s,t) =& 128092t^4s^4 + 86016t^3s^4 + 43008t^4s^3 - 159912t^2s^4 + 292544t^3s^3 - 25000t^4s^2 - \\
& - 43008t\,s^4 + 43008t^2s^3 + 129024t^3s^2 + 34364s^4 - 187456t\,s^3 + 265264t^2s^2 - \\
& - 187456t^3s + 34364t^4 - 129024t\,s^2 - 43008t^2s + 43008t^3 - 25000s^2 + \\
& + 292544st - 159912t^2 - 43008s - 86016t + 128092\,, \\
H_2(s,t) =& - 128092t^4s^4 - 43008t^3s^4 - 86016t^4s^3 + 25000t^2s^4 - 292544t^3s^3 + 159912t^4s^2 - \\
& - 129024t^2s^3 - 43008t^3s^2 + 43008t^4s - 34364s^4 + 187456t\,s^3 - 265264t^2s^2 + \\
& + 187456t^3s - 34364t^4 - 43008s^3 + 43008t\,s^2 + 129024t^2s + 159912s^2 - \\
& - 292544st + 25000t^2 + 86016s + 43008t - 128092\,.
\end{aligned}$$

We isolate the real roots of the system $\{H_1 = H_2 = 0\}$; they correspond to the blue segments in Fig. 5.10a. Then $\mathbb{R}$ is decomposed at the s-projections of the roots as shown in Fig. 5.11. The segments on the boundary of the convex hull are shown in red in Fig. 5.10b and the parameter intervals corresponding to the arcs on the boundary of $\mathrm{conv}(\phi(\mathbb{R}))$ are annotated in orange in Fig. 5.11.

Figure 5.11: Decomposition of the parameter interval in Example 5.17.

Example 5.18. We consider a curve in $\mathbb{R}^3$ parametrized by

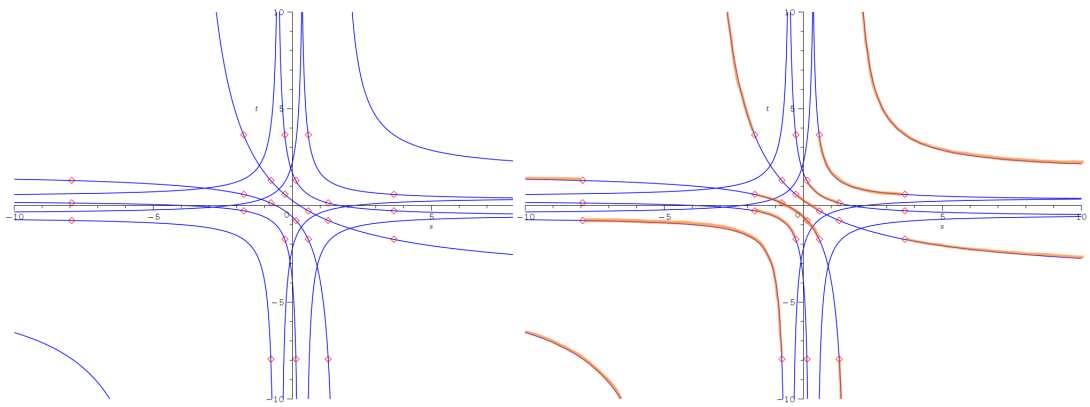
$$\phi(t) = \left(\frac{-(t+3)(3t-1)}{5(t^2+1)}, \frac{2(t-2)(2t+1)}{5(t^2+1)}, \frac{8(t-2)(2t+1)(t+3)(3t-1)(7t^2+2t-7)(t^2-14t-1)}{625(t^2+1)^4} \right).$$

The parametrization is proper. We take $I = \mathbb{R}$. We compute the bisecant curve, which is

defined by the polynomial

$$G(s,t) = 64\,(5s^2t^2 - s^2 + 12st - t^2 + 5)(17s^2t^2 + 62s^2t + 62st^2 - 17s^2 - 68st - 17t^2 - 62s - \\ - 62t + 17) \cdot (31s^2t^2 - 34s^2t - 34st^2 - 31s^2 - 124st - 31t^2 + 34s + 34t + 31)\,.$$

The curve has neither cusps nor endpoints. We compute the polynomials H_1, H_2 and $R_1 = \mathtt{res}_u(H_1, H_2)$ and we solve the system $\{G = R_1 = 0\}$. There are 96 real roots. For simplicity, in Fig. 5.12a we show only 24 of them on the graph of G; they correspond to the boundary segments of the triangle facets on the boundary of $\mathrm{conv}(\phi(\mathbb{R}))$. Then, for the second step of the algorithm, we decompose the s-axis at the s-projections of these roots and of the critical points of G and at the values where G has a vertical asymptote; there are 46 decomposition points in total. The branches of the bisecant curve that correspond to bitangent surface patches on the boundary are annotated in Fig. 5.12b. By sampling points on the graph of the bisecant curve, we construct the bisecant surface in Fig. 5.13a. At last, again by sampling but only on the annotated branches of the bisecant curve we construct the convex hull in Fig. 5.13b. It consists of 8 surface patches and triangle facets that assemble to two squares.



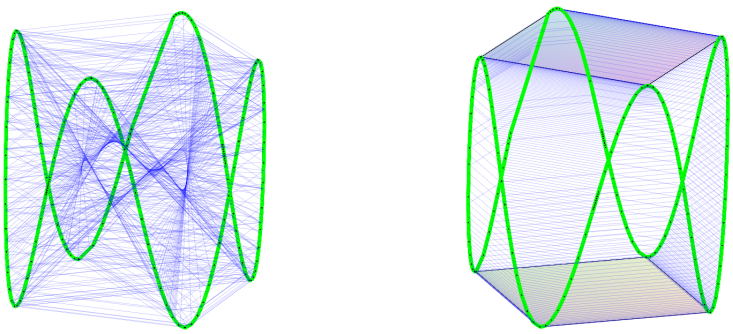
(a) (b)

Figure 5.12: (a) The graph of the bisecant curve of Example 5.18. (b) The arcs of the bisecant curve that correspond to bisecant surface patches are shown in orange.

Example 5.19. We consider the curve in $\mathbb{R}^3$ parametrized by

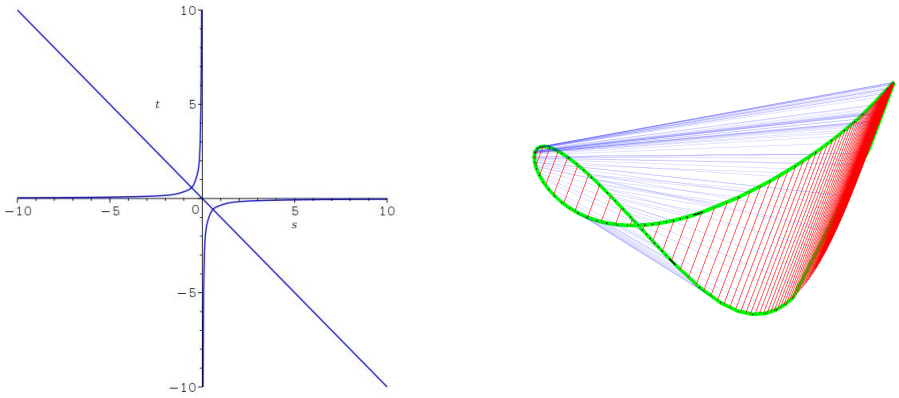
$$\phi(t) = \left(-\frac{3t^2 - 1}{t^2 + 1}, -\frac{t\,(3t^2 - 1)}{(t^2 + 1)^2}, -\frac{(t^6 - 1)(3t^2 - 1)}{(t^2 + 1)^4} \right)\,.$$

The parametrization is proper. We take $I = \mathbb{R}$. The bitangent curve is defined by the



(a) (b)

Figure 5.13: (a) The bisecant surface and (b) the convex hull of the curve of Example 5.18.



(a) (b)

Figure 5.14: (a) The bisecant curve and (b) the convex hull of the curve in Example 5.19

polynomial

$$\begin{aligned} G(s,t) = &-8(s+t)(63s^7t^7 + 81s^7t^5 - 33s^6t^6 + 81s^5t^7 + 45s^7t^3 - 159s^6t^4 - 201s^5t^5 - \\ &- 159s^4t^6 + 45s^3t^7 + 27s^7t + 13s^6t^2 + 195s^5t^3 + 295s^4t^4 + 195s^3t^5 + 13s^2t^6 + \\ &+ 27s\,t^7 + 11s^6 + 93s^5t + 211s^4t^2 + 375s^3t^3 + 211s^2t^4 + 93s\,t^5 + 11t^6 + 13s^4 - \\ &- 31s^3t - 169s^2t^2 - 31s\,t^3 + 13t^4 + 17s^2 + 31st + 17t^2 + 15)\,. \end{aligned}$$

The graph of G is shown in Fig. 5.14a. There are no triangle facets on the boundary of the convex hull. The convex hull consists of two surface patches (Fig. 5.14b); the red one corresponds to the factor $(s+t)$ of G, and the blue one corresponds to the other factor.

Bibliography

[AB89] S. S. Abhyankar and C. J. Bajaj. Automatic parameterization of rational curves and surfaces IV: Algebraic space curves. *ACM Trans. Graph.*, 8(4):325–334, 1989.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ACDT20] Juan Gerardo Alcázar, Jorge Caravantes, Gema María Díaz, and Elias Tsigaridas. Computing the topology of a plane or space hyperelliptic curve. *Computer Aided Geometric Design*, 78:101830, 2020.

[ADT10] Juan Gerardo Alcázar and Gema María Díaz-Toca. Topology of 2D and 3D rational curves. *CAGD*, 27(7):483 – 502, 2010.

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, 160, 09 2002.

[ALM99] Philippe Aubry, Daniel Lazard, and Marc Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1):105–124, 1999.

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

[AR07] Carlos Andradas and Tomás Recio. Plotting missing points and branches of real parametric curves. *Applicable Algebra in Engineering, Communication and Computing*, 18, 02 2007.

[Arn16] Arnold, Andrew. *Sparse Polynomial Interpolation and Testing*. PhD thesis, UWSpace, 2016.

[Baj88] Chanderjit Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3(2):177–191, Jun 1988.

[BBL+17] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[BCD+96] Jean-Daniel Boissonnat, André Cérézo, Olivier Devillers, Jacqueline Duquesne, and Mariette Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension d. *Computational Geometry*, 6(2):123 – 130, 1996.

[BCLM10] François Boulier, Changbo Chen, François Lemaire, and Marc Maza. Real root isolation of regular chains. *Proc. Asian Symposium on Computer Mathematics (ASCM)*, 09 2010.

[BCSS98] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.

[BD12] Laurent Busé and Carlos D'Andrea. Singular factors of rational plane curves. *J. Algebra*, 357:322–346, 2012.

[Ben19] Matías R. Bender. *Algorithms for sparse polynomial systems: Gröbner basis and resultants*. PhD thesis, Sorbonne Université, 2019.

[BFM+01] Christoph Burnikel, Stefan Funke, Kurt Mehlhorn, Stefan Schirra, and Susanne Schmitt. A separation bound for real algebraic expressions. *Algorithmica*, 55:14–28, 08 2001.

[BFMS00] Christoph Burnikel, Rudolf Fleischer, Kurt Mehlhorn, and Stefan Schirra. A strong and easily computable separation bound for arithmetic expressions involving radicals. *Algorithmica*, 27:87–99, 2000.

[BGI16] Alessandra Bernardi, Alessandro Gimigliano, and Monica Idà. Singularities of plane rational curves via projections. *J. Symb. Comput.*, 09 2016.

[BK91] Chandrajit Bajaj and Myung-Soo Kim. Convex hulls of objects bounded by algebraic curves. *Algorithmica*, 6:533–553, 06 1991.

[BKM05] Laurent Busé, Houssam Khalil, and Bernard Mourrain. Resultant-based methods for plane curves intersection problems. In Victor G. Ganzha, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 75–92, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Bla05] Paul A. Blaga. *Lectures on the Differential Geometry of Curves and Surfaces*. Napoca Press, Cluj-Napoca, Romania, 2005.

[BLB10] Laurent Busé and Thang Luu Ba. Matrix-based Implicit Representations of Rational Algebraic Curves and Applications. *CAGD*, 27(9):681–699, October 2010.

[BLM+15] Yacine Bouzidi, Sylvain Lazard, Guillaume Moroz, Marc Pouget, Fabrice Rouillier, and Michael Sagraloff. Improved algorithms for solving bivariate systems via rational univariate representations. *Research report, Inria*, 2015.

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

[BLPR13] Yacine Bouzidi, Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Rational univariate representations of bivariate systems and applications. In *Proc. 38th Int'l Symp. on Symbolic and Algebraic Computation*, ISSAC '13, pages 109–116, NY, USA, 2013. ACM.

[BLPR15] Yacine Bouzidi, Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Separating linear forms and rational univariate representations of bivariate systems. *J. Symb. Comput.*, pages 84–119, 2015.

[BLY19] Laurent Busé, Clément Laroche, and Fatmanur Yıldırım. Implicitizing rational curves by the method of moving quadrics. *Computer-Aided Design*, 114:101–111, 2019.

[BPD17] Angel Blasco and S. Pérez-Díaz. Resultants and singularities of parametric curves. *arXiv*, abs/1706.08430s, 2017.

[BPD19] Angel Blasco and Sonia Pérez-Díaz. An in depth analysis, via resultants, of the singularities of a parametric curve. *CAGD*, 68:22–47, 2019.

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

[BPT12] Grigoriy Blekherman, Pablo A. Parrilo, and Rekha R. Thomas. *Semidefinite Optimization and Convex Algebraic Geometry*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2012.

[Brø83] Arne Brøndsted. *An Introduction to Convex Polytopes*. Graduate Texts in Mathematics. Springer, 1983.

[BS16] Cornelius Brand and Michael Sagraloff. On the Complexity of Solving Zero-Dimensional Polynomial Systems via Projection. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation - ISSAC '16*, pages 151–158, Waterloo, ON, Canada, 2016. ACM Press.

[BS17] Ruben Becker and Michael Sagraloff. Counting solutions of a polynomial system locally and exactly. *ArXiv*, abs/1712.05487, 2017.

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

[Buc87] Bruno Buchberger. Applications of gröbner bases in non-linear computational geometry. In Rainer Janßen, editor, *Trends in Computer Algebra, International Symposium, Bad Neuenahr, Germany, May 19-21, 1987, Proceedings*, volume 296 of *Lecture Notes in Computer Science*, pages 52–80. Springer, 1987.

[Buc88] B. Buchberger. Algebraic methods for non-linear computational geometry (invited address). In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, SCG '88, page 81–82, New York, NY, USA, 1988. Association for Computing Machinery.

[Bus14] Laurent Busé. Implicit matrix representations of rational Bézier curves and surfaces. *Computer-Aided Design*, 46:14–24, 2014.

[Can88] John F. Canny. *The complexity of robot motion planning.* PhD thesis, MIT Press, 1988.

[CCC+05] Eduardo Cattani, David A. Cox, Guillaume Chèze, Alicia Dickenstein, Mohamed Elkadi, Ioannis Z. Emiris, André Galligo, Achim Kehrein, Martin Kreuzer, and Bernard Mourrain. *Solving Polynomial Equations: Foundations, Algorithms, and Applications (Algorithms and Computation in Mathematics).* Springer-Verlag, Berlin, Heidelberg, 2005.

[CFGVN14] Jorge Caravantes, Mario Fioravanti, Laureano Gonzalez-Vega, and Ioana Necula. Computing the topology of an arrangement of implicit and parametric curves given by values. In Vladimir P. Gerdt, Wolfram Koepf, Werner M. Seiler, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 59–73, Cham, 2014. Springer.

[CGY09] Jin-San Cheng, Xiao-Shan Gao, and Chee-Keng Yap. Complete numerical isolation of real roots in zero-dimensional triangular systems. *Journal of Symbolic Computation*, 44(7):768–785, 2009. International Symposium on Symbolic and Algebraic Computation.

[CJL13] Jin-San Cheng, Kai Jin, and Daniel Lazard. Certified rational parametric approximation of real algebraic space curves with local generic position method. *Journal of Symbolic Computation*, 58:18 – 40, 2013.

[CJP+21] Jin-San Cheng, Kai Jin, Marc Pouget, Junyi Wen, and Bingwei Zhang. An Improved Complexity Bound for Computing the Topology of a Real Algebraic Space Curve. working paper or preprint, December 2021.

[CKLS18] Daniel Ciripoi, Nidhi Kaihnsa, Andreas Löhne, and Bernd Sturmfels. Computing convex hulls of trajectories. *arXiv*, abs/1810.03547, 2018.

[CKPU11] David Cox, Andrew Kustin, Claudia Polini, and Bernd Ulrich. A study of singularities on rational curves via syzygies. *Memoirs of the American Mathematical Society*, 222, 02 2011.

[CLL+20] Franz Chouly, Jinan Loubani, Alexei Lozinski, Bochra Méjri, Kamil Merito, Sébastien PASSOS, and Angie Pineda. Computing bi-tangents for transmission belts. working paper or preprint, January 2020.

[CLO05] David Cox, John Little, and Donal O'Shea. Using algebraic geometry. Graduate Texts in Mathematics N.185. Springer-Verlag New York, 2005.

[CLO15] David Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Undergraduate texts in mathematics. Springer, 4rth ed edition, 2015.

[CM12] Changbo Chen and Marc Moreno Maza. Algorithms for computing triangular decomposition of polynomial systems. *Journal of Symbolic Computation*, 47(6):610–642, 2012. Advances in Mathematics Mechanization.

[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. *Journal of Symbolic Computation*, 5(1):121–129, 1988.

[CRE01] Elaine Cohen, Richard F. Riesenfeld, and Gershon Elber. *Geometric modeling with splines - an introduction*. A K Peters, 2001.

[CS01a] Eng-Wee Chionh and Thomas W. Sederberg. On the minors of the implicitization bézout matrix for a rational plane curve. *CAGD*, 18(1):21 – 36, 2001.

[CS01b] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *BULLETIN*, 39, 11 2001.

[dBCvKO08] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[DDR+22] Daouda Niang Diatta, Sény Diatta, Fabrice Rouillier, Marie-Françoise Roy, and Michael Sagraloff. Bounds for polynomials on algebraic numbers and

application to curve topology. *Discrete & Computational Geometry*, Feb 2022.

[DET09] Dimitris 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. (Special issue on ISSAC 2007).

[dFT90] Rui JP de Figueiredo and Hemant D Tagare. Curves and surfaces in computer vision. In *Curves and Surfaces in Computer Vision and Graphics*, volume 1251, pages 10–16. SPIE, 1990.

[dIM22] Jaume de Dios Pont, Paata Ivanisvili, and José Madrid. The convex hull of space curves with totally positive torsion. *arXiv, abs/2201.12932*, January 2022.

[DLLP08a] Laurent Dupont, Daniel Lazard, Sylvain Lazard, and Sylvain Petitjean. Near-Optimal Parameterization of the Intersection of Quadrics: I. The Generic Algorithm. *Journal of Symbolic Computation*, 43(3):168–191, 2008.

[DLLP08b] Laurent Dupont, Daniel Lazard, Sylvain Lazard, and Sylvain Petitjean. Near-Optimal Parameterization of the Intersection of Quadrics: II. A Classification of Pencils. *Journal of Symbolic Computation*, 43(3):192–215, 2008.

[DLLP08c] Laurent Dupont, Daniel Lazard, Sylvain Lazard, and Sylvain Petitjean. Near-Optimal Parameterization of the Intersection of Quadrics: III. Parameterizing Singular Intersections. *Journal of Symbolic Computation*, 43(3):216–232, 2008.

[DMO33] Erik D. Demaine, Joseph B. M. Mitchell, and Joseph O'Rourke. Problem 33. *The Open Problems Project*, 2007, (https://topp.openproblem.net/p33).

[DMR08] Daouda Niang Diatta, Bernard Mourrain, and Olivier Ruatta. On the computation of the topology of a non-reduced implicit space curve. In *Proceedings of the Twenty-First International Symposium on Symbolic and Algebraic Computation*, ISSAC '08, page 47–54, New York, NY, USA, 2008. Association for Computing Machinery.

[DS90] David P. Dobkin and Diane L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5:421–457, 1990.

[DSM+05] X. Dahan, É. Schost, M. Moreno Maza, W. Wu, and Y. Xie. On the complexity of the D5 principle. *SIGSAM Bull.*, 39(3):97–98, sep 2005.

[Ede04] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10. 01 2004.

[EKH01] Gershon Elber, Myung-Soo Kim, and Hee-Seok Heo. The convex hull of rational plane curves. *Graphical Models*, 63:151–162, 05 2001.

[ELLED07] Hazel Everett, Sylvain Lazard, Daniel Lazard, and Mohab Safey El Din. The Voronoi diagram of three lines. In *Proceedings of the Twenty-Third Annual Symposium on Computational Geometry*, SCG '07, page 255–264, New York, NY, USA, 2007. Association for Computing Machinery.

[Emi05] Ioannis Z. Emiris. *Toric resultants and applications to geometric modelling*, pages 269–300. Solving Polynomial Equations: Foundations, Algorithms, and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[EMP10] M. Elkadi, B. Mourrain, and R. Piene. *Algebraic Geometry and Geometric Modeling*. Mathematics and Visualization. Springer Berlin Heidelberg, 2010.

[EMT20] Ioannis Emiris, Bernard Mourrain, and Elias Tsigaridas. Separation bounds for polynomial systems. *Journal of Symbolic Computation*, 101:128–151, 2020.

[EP05] Ioannis Z. Emiris and Victor Y. Pan. Improved algorithms for computing determinants and resultants. *Journal of Complexity*, 21(1):43–71, 2005. Foundations of Computational Mathematics Conference 2002.

[EP17] Paula Escorcielo and Daniel Perrucci. On the Davenport–Mahler bound. *Journal of Complexity*, 41:72–81, 2017.

[ETMT06] Ioannis Emiris, Elias Tsigaridas, and George M. Tzoumas. The predicates for the Voronoi diagram of ellipses. *Proceedings of the Annual Symposium on Computational Geometry*, 2006:227–236, 01 2006.

[FGS10] Rida T. Farouki, Carlotta Giannelli, and Alessandra Sestini. Geometric design using space curves with rational rotation-minimizing frames. In Morten Dæhlen, Michael Floater, Tom Lyche, Jean-Louis Merrien, Knut Mørken, and Larry L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces*, pages 194–208. Springer, 2010.

[Ful69] W. Fulton. *Algebraic Curves. An Introduction to Algebraic Geometry*. Addison Wesley, 1969.

[Ful84] William Fulton. *Introduction to Intersection Theory in Algebraic Geometry*. Cbms Regional Conference Series in Mathematics. American Mathematical Society, 1984.

[GC92] Xiao-Shan Gao and Shang-Ching Chou. Implicitization of rational parametric equations. *J. Symb. Comput.*, 14(5):459 – 470, 1992.

[GIHS01] Nicola Geismann, Fachrichtung Informatik, Michael Hemmer, and Elmar Sch. The convex hull of ellipsoids. *Proceedings of the Annual Symposium on Computational Geometry*, 04 2001.

[GPB+08] Gert-Martin Greuel, Gerhard Pfister, Olaf Bachmann, Christoph Lossen, and Hans Schönemann. *A Singular introduction to commutative algebra*, volume 348. Springer, 2008.

[GRS02] Jaime Gutierrez, Rosario Rubio, and David Sevilla. On multivariate rational function decomposition. *J. Symb. Comput.*, 33(5):545 – 562, 2002.

[GRY02] Jaime Gutierrez, Rosario Rubio, and Jie-Tai Yu. D-resultant for rational functions. *Proc. American Mathematical Society*, 130, 08 2002.

[GVNPD+04] Laureano González-Vega, Ioana Necula, Sonia Pérez-Díaz, Juana Sendra, and Juan Sendra. Algebraic methods in computer aided geometric design: Theoretical and practical applications. *Geometric Computation*, 11, 03 2004.

[HDLP22] Petr Hruby, Timothy Duff, Anton Leykin, and Tomás Pajdla. Learning to solve hard minimal problems. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5522–5532. IEEE, 2022.

[HDPS11] Michael Hemmer, Laurent Dupont, Sylvain Petitjean, and Elmar Schömer. A Complete, Exact and Efficient Implementation for Computing the Edge-Adjacency Graph of an Arrangement of Quadrics. *Journal of Symbolic Computation*, 46(4):467–494, 2011.

[Hen11] Didier Henrion. Semidefinite representation of convex hulls of rational varieties. *Acta applicandae mathematicae*, 115(3):319–327, 2011.

[HI93] Chao-Kuei Hung and Doug Ierardi. Convex hulls of curved objects via duality: a general framework and an optimal 2-d algorithm. *Technical Report USC-CS-93-556, University of Southern California*, 1993.

[HL21] J. V. D. Hoeven and G. Lecerf. On the complexity exponent of polynomial system solving. *Foundations of Computational Mathematics*, 21:1–57, 2021.

[HvdH21] David Harvey and Joris van der Hoeven. Integer multiplication in time O(n log n). *Annals of Mathematics*, 2021.

[IPY21] Rémi Imbach, Marc Pouget, and Chee-Keng Yap. Clustering complex zeros of triangular systems of polynomials. *Mathematics in Computer Science*, 15:271–292, 2021.

[JA06] Jean-Pierre Jouanolou and François Apéry. *Elimination. Le cas d'une variable.* Hermann, Collection Méthodes, 11 2006.

[JK97] J. R. Johnson and Werner Krandick. Polynomial real root isolation using approximate arithmetic. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, ISSAC '97, page 225–232, New York, NY, USA, 1997. Association for Computing Machinery.

[JL05] Yan-Bin Jia and Huan Lin. On the convex hulls of parametric plane curves. *Technical report, www.cs.iastate.edu/jia*, 2005.

[Joh01] J. K. Johnstone. A parametric solution to common tangents. In *Proceedings International Conference on Shape Modeling and Applications*, pages 240–249, May 2001.

[Joh04] J. K. Johnstone. Giftwrapping a curve with the convex hull. In *Proceedings of the 42nd Annual Southeast Regional Conference*, ACM-SE 42, page 224–227, New York, NY, USA, 2004. Association for Computing Machinery.

[JP08] Bert Jüttler and Ragni Piene. *Geometric modeling and algebraic geometry.* Springer, 2008.

[JSC18] Xiaohong Jia, Xiaoran Shi, and Falai Chen. Survey on the theory and applications of μ-bases for rational curves and surfaces. *J. Comput. Appl. Math.*, 329:2–23, 2018.

[Kah03] M'hammed El Kahoui. An elementary approach to subresultants theory. *Journal of Symbolic Computation*, 35(3):281–292, 2003.

[Kah08] M'hammed El Kahoui. Topology of real algebraic space curves. *J. Symb. Comput.*, 43(4):235 – 258, 2008.

[Klo95] Jürgen Klose. Binary segmentation for multivariate polynomials. *J. Complexity*, 11(3):330–343, 1995.

[KMP+08] Lutz Kettner, Kurt Mehlhorn, Sylvain Pion, Stefan Schirra, and Chee Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61–78, 2008.

[KRTZ20a] Christina Katsamaki, Fabrice Rouillier, Elias Tsigaridas, and Zafeirakis Zafeirakopoulos. On the geometry and the topology of parametric curves.

In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, ISSAC '20, page 281–288, New York, NY, USA, 2020. Association for Computing Machinery.

[KRTZ20b] Christina Katsamaki, Fabrice Rouillier, Elias P. Tsigaridas, and Zafeirakis Zafeirakopoulos. PTOPO: A Maple package for the topology of parametric curves. *ACM Commun. Comput. Algebra*, 54(2):49–52, 2020.

[KRTZ23] Christina Katsamaki, Fabrice Rouillier, Elias Tsigaridas, and Zafeirakis Zafeirakopoulos. Ptopo: Computing the geometry and the topology of parametric curves. *Journal of Symbolic Computation*, 115:427–451, 2023.

[KS11] Michael Kerber and Michael Sagraloff. Efficient real root approximation. In *Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, ISSAC '11, page 209–216, New York, NY, USA, 2011. Association for Computing Machinery.

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

[KS13] Alexander Kobel and Michael Sagraloff. Fast approximate polynomial multipoint evaluation and applications. *arXiv*, abs/1304.8069, 2013.

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

[KU08] Kiran S. Kedlaya and Christopher Umans. Fast modular composition in any characteristic. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–155, 2008.

[KU11] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802, 2011.

[Kur12] Aleksei Kurbatskiĭ. Convex hulls of a curve in control theory. *Sbornik Mathematics - SB MATH*, 203:406–423, 03 2012.

[KYP92] David J Kriegman, Erliang Yeh, and Jean Ponce. Convex hulls of algebraic curves. In *Curves and Surfaces in Computer Vision and Graphics III*, volume 1830, pages 118–127. SPIE, 1992.

[Lak91] Y. N. Lakshman. *A Single Exponential Bound on the Complexity of Computing Gröbner Bases of Zero Dimensional Ideals*, pages 227–234. Birkhäuser Boston, Boston, MA, 1991.

[Las01] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[Las09] Jean B Lasserre. Convexity in semialgebraic geometry and polynomial optimization. *SIAM Journal on Optimization*, 19(4):1995–2014, 2009.

[Laz92] D. Lazard. Solving zero-dimensional algebraic systems. *Journal of Symbolic Computation*, 13(2):117–131, 1992.

[LC97] Yong-Ming Li and Robert J. Cripps. Identification of inflection points and cusps on rational curves. *CAGD*, 14(5):491 – 497, 1997.

[LK92] In-Kwon Lee and Myung-Soo Kim. Primitive geometric operations on planar algebraic curves with gaussian approximation. In Tosiyasu L. Kunii, editor, *Visual Computing*, pages 449–468, Tokyo, 1992. Springer Japan.

[LMP09] Xin Li, Marc Moreno Maza, and Wei Pan. Computations modulo regular chains. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, ISSAC '09, page 239–246, New York, NY, USA, 2009. Association for Computing Machinery.

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

[LR01] Thomas Lickteig and Marie-Françoise Roy. Sylvester–Habicht sequences and fast Cauchy index computation. *J. Symb. Comput.*, 31(3):315–341, March 2001.

[Mah62] Kurt Mahler. On some inequalities for polynomials in several variables. *J. London Mathematical Society*, 1(1):341–344, 1962.

[MB72] Robert T. Moenck and Allan Borodin. Fast modular transforms via division. In *Scandinavian Workshop on Algorithm Theory*, 1972.

[MC92] Dinesh Manocha and John F. Canny. Detecting cusps and inflection points in curves. *CAGD*, 9(1):1 – 24, 1992.

[Mou93] B. Mourrain. The 40 "generic" positions of a parallel robot. In *Proceedings of the 1993 International Symposium on Symbolic and Algebraic Computation*, ISSAC '93, page 173–182, New York, NY, USA, 1993. Association for Computing Machinery.

[MR95] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *SIGACT News*, 26:48–50, 1995.

[MS00] K. Mehlhorn and S. Schirra. *A Generalized and Improved Constructive Separation Bound for Real Algebraic Expressions.* Forschungsberichte des Max-Planck-Instituts für Informatik. MPI Informatik, Bibliothek & Dokumentation, 2000.

[MST17] Angelos Mantzaflaris, Eric Schost, and Elias Tsigaridas. Sparse rational univariate representation. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, page 301–308, New York, NY, USA, 2017. Association for Computing Machinery.

[MSW15] Kurt Mehlhorn, Michael Sagraloff, and Pengming Wang. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 66:34 – 69, 2015.

[NRS20] Vincent Neiger, Johan Rosenkilde, and Grigory Solomatov. Generic bivariate multi-point evaluation, interpolation and modular composition with precomputation. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, ISSAC '20, page 388–395, New York, NY, USA, 2020. Association for Computing Machinery.

[OA21] Barak Or and Idowu Oluwafemi Amos. Lengthnet: Length learning for planar euclidean curves. In *Smart Tools and Applications in Graphics*, 2021.

[Pan94] Victor Y. Pan. Simple multivariate polynomial multiplication. *Journal of Symbolic Computation*, 18(3):183–186, 1994.

[Pan02] Victor Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *Journal of Symbolic Computation*, 33(5):701–733, 2002.

[Pap07] Christos H. Papadimitriou. *Computational complexity.* Academic Internet Publ., 2007.

[Par02] Hyungju Park. Effective computation of singularities of parametric affine curves. *J. Pure and Applied Algebra*, 173:49–58, 08 2002.

[PASM17] Lucía Hilario Pérez, Marta Covadonga Mora Aguilar, Nicolás Montés Sánchez, and Antonio Falcó Montesinos. Path planning based on parametric curves. In Rastislav Róka, editor, *Advanced Path Planning for Mobile Entities*, chapter 7. IntechOpen, Rijeka, 2017.

[PD06] Sonia Pérez-Díaz. On the problem of proper reparametrization for rational curves and surfaces. *CAGD*, 23(4):307–323, 2006.

[PD07] Sonia Pérez-Díaz. Computation of the singularities of parametric plane curves. *J. Symb. Comput.*, 42(8):835 – 857, 2007.

[PLH+05] Helmut Pottmann, Stefan Leopoldseder, Michael Hofer, Tibor Steiner, and Wenping Wang. Industrial geometry: Recent advances and applications in cad. *Computer-Aided Design*, 37:751–766, 06 2005.

[PM95] Laxmi Parida and SP Mudur. Common tangents to planar parametric curves: a geometric solution. *Computer-Aided Design*, 27(1):41–47, 1995.

[PS85] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, Berlin, Heidelberg, 1985.

[PT17] Victor Pan and Elias Tsigaridas. Accelerated approximation of the complex roots and factors of a univariate polynomial. *Theor. Computer Science*, 681:138 – 145, 2017.

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

[Rou07] Fabrice Rouillier. *Algorithmes pour l'étude des solutions réelles des systèmes polynomiaux*. Habilitation à diriger des recherches, Université Pierre & Marie Curie - Paris 6, March 2007.

[RS09] Kristian Ranestad and Bernd Sturmfels. The convex hull of a space curve. *Advances in Geometry*, 12, 12 2009.

[RS11] Kristian Ranestad and Bernd Sturmfels. The convex hull of a variety. *Notions of Positivity and the Geometry of Polynomials*, pages 331–344, 2011.

[RSS13] Sonia L. Rueda, Juana Sendra, and J. Rafael Sendra. An algorithm to parametrize approximately space curves. *J. Symb. Comput.*, 56:80 – 106, 2013.

[RSV09] R. Rubio, J.M. Serradilla, and M.P. Vélez. Detecting real singularities of a space curve from a real rational parametrization. *J. Symb. Comput.*, 44(5):490 – 498, 2009.

[Rum77] Siegfried M. Rump. Real root isolation for algebraic polynomials. *SIGSAM Bull.*, 11(2):2–3, may 1977.

[SAG84] T.W Sederberg, D.C Anderson, and R.N Goldman. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics, and Image Processing*, 28(1):72 – 84, 1984.

[Sal06] David Salomon. *Curves and surfaces for computer graphics.* 01 2006.

[SC95] Thomas W. Sederberg and Falai Chen. Implicitization using moving curves and surfaces. In *Proc. of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 301–308, NY, USA, 1995.

[Sch88] A Schönhage. Probabilistic computation of integer polynomial gcds. *J. Algorithms*, 9(3):365 – 371, 1988.

[Sed86a] Thomas W. Sederberg. Improperly parametrized rational curves. *CAGD*, 3(1):67–75, May 1986.

[Sed86b] V. Sedykh. Structure of the convex hull of a space curve. *Journal of Mathematical Sciences*, 33:1140–1153, 05 1986.

[SEJK04] J.-K Seong, Gershon Elber, J. Johnstone, and Myung-Soo Kim. The convex hull of freeform surfaces. *Computing*, 72:171–183, 04 2004.

[Sen02] J. Rafael Sendra. Normal parametrizations of algebraic plane curves. *J. Symb. Comput.*, 33:863–885, 2002.

[SF00] Meng Sun and Eugene Fiume. A technique for constructing developable surfaces. *Graphics Interface '96*, 12 2000.

[SJVW87] Alejandro Schaffer and Christopher J Van Wyk. Convex hulls of piecewise-smooth jordan curves. *Journal of Algorithms*, 8:66–94, 03 1987.

[SMC20] Guillaume Staerman, Pavlo Mozharovskyi, and S. Clémençon. The area of the convex hull of sampled curves: a robust functional statistical depth measure. In *AISTATS*, 2020.

[SNW11] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning.* The MIT Press, 2011.

[ST12] Adam Strzeboński and Elias P. Tsigaridas. Univariate real root isolation in multiple extension fields. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, ISSAC '12, page 343–350, New York, NY, USA, 2012. Association for Computing Machinery.

[ST19] Adam Strzebonski and Elias Tsigaridas. Univariate real root isolation in an extension field and applications. *J. Symb. Comput.*, 92:31 – 51, 2019.

[STM19] Christian Scheiderer, Timo Thun, and Tobias Meisen. Bézier curve based continuous and smooth motion planning for self-learning industrial robots.

Procedia Manufacturing, 38:423–430, 2019. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[Stu02] Bernd Sturmfels. *Solving systems of polynomial equations.* Number 97. American Mathematical Soc., 2002.

[SW99] J. Rafael Sendra and Franz Winkler. Algorithms for rational real algebraic curves. *Fundam. Inf.*, 39(1,2):211–228, April 1999.

[SWPD08] J Rafael Sendra, Franz Winkler, and Sonia Pérez-Díaz. Rational algebraic curves. *Algorithms and Computation in Mathematics*, 22, 2008.

[Tei06] Monique Teillaud. From triangles to curves (invited talk). In *22nd European Workshop on Computational Geometry*, 2006.

[vdEY97] Arno van den Essen and Jie-Tai Yu. The D-resultant, singularities and the degree of unfaithfulness. *Proc. American Mathematical Society*, 125, 01 1997.

[vdHL21] Joris van der Hoeven and Grégoire Lecerf. Amortized bivariate multi-point evaluation. In *Proceedings of the 2021 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '21, page 179–185, New York, NY, USA, 2021. Association for Computing Machinery.

[vdHL23] Joris van der Hoeven and Grégoire Lecerf. Amortized multi-point evaluation of multivariate polynomials. *Journal of Complexity*, 74:101693, 2023.

[vdHS12] Joris van der Hoeven and Éric Schost. Multi-point evaluation in higher dimensions. *Applicable Algebra in Engineering, Communication and Computing*, 24:37 – 52, 2012.

[Vin11] Cynthia Leslie Vinzant. *Real algebraic geometry in convex optimization.* University of California, Berkeley, 2011.

[vL20] Joris van der Hoeven and Grégoire Lecerf. Fast multivariate multi-point evaluation revisited. *Journal of Complexity*, 56:101405, 2020.

[VMC97] Tamás Várady, Ralph R Martin, and Jordan Cox. Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, 29(4):255–268, 1997. Reverse Engineering of Geometric Models.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra.* Cambridge University Press, 3rd edition, 2013.

[Wal78] Robert J. Walker. *Algebraic curves*. Springer-Verlag, 1978.

[WP03] Xinmao Wang and Victor Y. Pan. Acceleration of euclidean algorithm and rational number reconstruction. *SIAM Journal on Computing*, 32(2):548–556, 2003.

[XY02] Bican Xia and Lu Yang. An algorithm for isolating the real solutions of semi-algebraic systems. *J. Symb. Comput.*, 34:461–477, 2002.

[Yap99] Chee Keng Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, Inc., USA, 1999.

[YE94] Chee Yap and Thomas E. The exact computation paradigm. *Computing in Euclidean Geometry*, 08 1994.

[YLLF11] Y Yang, Y.-C Liu, M.-Y Liu, and M.-Y Fu. A path planning algorithm based on convex hull for autonomous service robot. 31:54–58+63, 01 2011.

[YS11] Chee-Keng Yap and Michael Sagraloff. A simple but exact and efficient algorithm for complex root isolation. In *International Symposium on Symbolic and Algebraic Computation*, 2011.

[ZFX09] Zhihai Zhang, Tian Fang, and Bican Xia. Real solution isolation with multiplicity of zero-dimensional triangular systems. *Science China Information Sciences*, 54, 06 2009.

[ZST11] F Zhou, Baoye Song, and Guohui Tian. Bézier curve based smooth path planning for mobile robot. *Journal of Information and Computational Science*, 8:2441–2450, 12 2011.