Solving bivariate systems using Rational Univariate Representations

Yacine Bouzidi, Sylvain Lazard, Guillaume Moroz, Marc Pouget, Fabrice Rouillier, Michael Sagraloff

Solving bivariate systems using Rational Univariate Representations

Yacine Bouzidi[*†] Sylvain Lazard[*†] Guillaume Moroz[*†] Marc Pouget[*†]
Fabrice Rouillier[*‡] Michael Sagraloff[§]

July 5, 2016

Abstract

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

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

1 Introduction

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

[*]INRIA, France. `Firstname.Name@inria.fr`
[†]LORIA laboratory, Nancy, France.
[‡]Institut de Mathématiques de Jussieu, Paris, France.
[§]Max-Planck-Institut für Informatik, Saarbrücken, Germany.

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

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

For arbitrary multivariate systems, Rouillier [Rou99] gives an algorithm for deterministically computing a separating form, which computes the number of solutions of a system with the rank of the Hermite quadratic form of a quotient algebra. The complexity of this computation dominates the one that follows for computing the rational representation. Considering the special case of systems of two bivariate polynomials of total degree bounded by d with integer coefficients of bitsize bounded by τ, another approach, based on a triangular decomposition, has been presented by Gonzalez-Vega and El Kahoui [GVEK96] for computing a separating linear form together with a rational parameterization of the solutions. The best-known bit complexity of this approach, analyzed by Diochnos et al. [DET09, Lemma 16 & Theorem 19][1], shows a bit complexity in $\widetilde{O}_B(d^{10} + d^9\tau)$ for computing a separating form and a bit complexity in $\widetilde{O}_B(d^7 + d^6\tau)$ for computing the corresponding rational parameterization. The computation of a separating linear form was still the bottleneck in the computation of the rational parameterization. An algorithm using modular arithmetic was then introduced by Bouzidi et al. [BLPR15] reducing the complexity to $\widetilde{O}_B(d^8+d^7\tau)$. This algorithm was later simplified and improved by transforming the problem into the computation of a separating form for the critical points of a curve, which improved the bit complexity to $\widetilde{O}_B(d^7 + d^6\tau)$ in the worst case and to $\widetilde{O}_B(d^5 + d^4\tau)$ in a probabilistic Las Vegas setting [BLP+14]. Bouzidi et al. [BLPR15] also showed that, given such a separating linear form, an alternative rational parameterization called Rational Univariate Representation (RUR) [Rou99] can be computed using

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

$\widetilde{O}_B(d^7 + d^6\tau)$ bit operations. For the first time, the worst-case bit complexities of computing linear separating forms and rational parameterizations (even RURs) of bivariate systems were both in the same class of complexity $\widetilde{O}_B(d^7 + d^6\tau)$ and, also for the first time, the expected bit complexity for computing linear separating forms, in a Las Vegas setting, was in a smaller class of complexity, $\widetilde{O}_B(d^5 + d^4\tau)$.

For computing a RUR in a Monte Carlo setting, the first step of computing a separating linear form is trivial since a random form is separating with some known probability. Doing so, Mehrabi and Schost [MS15] then compute a RUR of the radical of the input system with probability of success at least $1 - 1/2^p$ in a bit complexity that is the sum of $\widetilde{O}_B(d^{2+\varepsilon}(d^2 + d\tau + dp + p^2))$, for $\varepsilon > 0$ arbitrarily small, and of the complexity of computing a random prime smaller than $M = (2^p d\tau)^{O(1)}$. This result is remarkable since the whole complexity is $\widetilde{O}_B(d^{4+\varepsilon} + d^{3+\varepsilon}\tau)$ when p is a constant, which almost matches the worst-case upper and lower bounds $\widetilde{O}(d^4 + d^3\tau)$ and $\widetilde{\Omega}(d^4)$ on the size of the output (see Corollary 36 and [MS15, Main results]). However the drawback of this approach is that, so far, neither the separating form nor the computed RUR can be checked for correctness with a better bit complexity than those we present in this paper, that is $\widetilde{O}_B(d^5 + d^4\tau)$ expected and $\widetilde{O}_B(d^6 + d^5\tau)$ in the worst case.

Very recently, Kobel and Sagraloff [KS15b] presented an algorithm of worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ for computing isolating boxes of the solutions of bivariate systems. Their approach is based on resultant computations, projecting the solutions on the x and y-axes, thus defining a grid of candidate solutions. Then, approximate evaluations combined with adaptive evaluation bounds enable to identify the solutions from the grid. This method does not need the knowledge of a separating form, but once the solutions are isolated with enough precision, such a separating form can be computed in $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations [KS15b]. This approach for computing separating linear forms has the best known worst-case complexity. However, it would be surprising that computing a separating form with such complexity would require to first isolate the solutions of the system, since separating forms are precisely instrumental for solving systems in parameterization-based approaches. The present work indeed demonstrates that separating linear forms and rational parameterizations (including RURs) can be directly computed with this $\widetilde{O}_B(d^6 + d^5\tau)$ state-of-the-art worst-case bit complexity, and that the solutions of the system can be isolated from the RUR in the same worst-case complexity.

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

Main results. Let P and Q be two coprime polynomials in $\mathbb{Z}[x, y]$ of degree bounded by d and bitsize bounded by τ. We present three algorithms, one for each of the main steps of solving a bivariate system $\{P, Q\}$ via RURs, that is, computing (i) a separating linear form $x + ay$ with $a \in \{0, \ldots, 2d^4\}$, (ii) a RUR decomposition of the system, and (iii) isolating boxes of the solutions. Each of these algorithms has worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ and we also present Las Vegas variants of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ of our two first algorithms (see Theorems 26, 27, 43, 44, 59). We do not present a Las Vegas variant of our last algorithm for computing isolating boxes but it should be noticed that the complexity of that subdivision-based algorithm actually depends on the distances between the solutions and thus its worst-case complexity is not always

reached.

These complexities are not proved to be optimal but it should be stressed that, even in the simpler setting where the ideal is known to be radical, the complexities for problem (i) match the best known worst-case and expected bit complexities for checking whether a linear form is separating (see the conclusion for details). This also holds for problem (ii) since computing a RUR decomposition *requires* computing a separating form. Similarly, for problem (iii), observe that our bound also matches the best known complexity for the isolation of the roots of a given polynomial of degree and bitsize comparable to those of the resultant of the input polynomials [MSW15, Theorem 5].

Our algorithm for computing a separating linear form is based on the one presented in [BLP+14], while improving its worst-case bit complexity by a factor d. Furthermore, its Las Vegas variant is simpler than the one presented in [BLP+14] and it has the same expected bit complexity. As mentioned above the worst-case complexity of this new algorithm also matches the recent one by Kobel and Sagraloff [KS15b]. Our algorithm for computing a RUR decomposition of $\{P, Q\}$ improves by a factor d the state-of-the-art worst-case bit complexity [BLPR15]. Furthermore, our Las Vegas variant is, up to our knowledge, the first Las Vegas algorithm whose expected complexity is asymptotically better than the worst-case complexity and, as a result, our Las Vegas algorithm improves the state-of-the-art expected bit complexity by a factor d^2. For the isolation problem from the RURs, we improve the state-of-the-art complexity by a factor d^2 [BLPR15, Proposition 35], while matching the resultant-based complexity presented by Kobel and Sagraloff [KS15b].

Last but not least, we present an amortized analysis of the classical triangular decomposition via subresultants of bivariate systems [GVEK96], proving that the decomposition can be computed in $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case (Proposition 16), which improves by a factor d the state-of-the-art analysis [DET09, Proof of Theorem 19]. This result, while instrumental for the worst-case complexity analyses of our algorithms, is also of independent interest.

We first present a detailed overview of our contributions in Section 2. Notation and preliminaries are then introduced in Section 3. We present in Section 4 our amortized analysis of the triangular decomposition and a related luckiness certificate which is a key feature of the following multi-modular algorithms. Sections 5, 6 and 7 present respectively our algorithms for computing separating forms, RUR decompositions and isolating boxes of the solutions.

2 Overview

We present in this section a detailed overview of our contributions and strategies. Recall that P and Q denote two coprime polynomials in $\mathbb{Z}[x, y]$ of total degree at most d and maximum bitsize τ.

2.1 Triangular decomposition and luckiness certificate

We first recall in Section 4.1 the classical subresultant-based algorithm for computing the triangular decomposition of a zero-dimensional bivariate system $\{P, Q\}$. This decomposition appears, for instance, for solving bivariate systems [LMMRS11] or for the computation of the topology of curves [GVEK96]. This triangular decomposition algorithm will be used in our algorithm for computing a RUR decomposition of $\{P, Q\}$.

We then present in Section 4.2 a straightforward variation on this algorithm, which only computes the degree of this triangular decomposition (see Definition 11). This variation decreases the expected bit complexity of the algorithm and it is critical for our Las Vegas algorithm for computing a separating linear form.

We then present in Section 4.3 another variation on the triangular decomposition algorithm, which computes a luckiness certificate for this triangular decomposition. A luckiness certificate of $\{P, Q\}$ is an integer such that if a prime μ does not divide it, then μ is lucky for the triangular decomposition of $\{P, Q\}$ that is, the degree of the decomposition is preserved by the reduction modulo μ and the decomposition commutes with the reduction modulo μ (see Definition 13). Our deterministic algorithms for the separating form and the RUR computations will both use this luckiness certificate.

In Section 4.4, we prove that the worst-case bit complexities of these three algorithms are in $\widetilde{O}_B(d^6 + d^5\tau)$ and that the expected bit complexity of the one for computing the degree of the triangular decomposition is in $\widetilde{O}_B(d^5 + d^4\tau)$ (Proposition 16). The worst-case complexity is obtained by considering amortized bounds on the degrees and bitsizes of factors of the resultant and it improves by a factor d the state-of-the-art complexity for computing the triangular decomposition [DET09, Proof of Theorem 19]. Besides of being of independent interest, these improvements are critical for the complexity analysis of the following algorithms.

2.2 Separating linear form

In Section 5, we present a new algorithm for computing separating linear forms for a bivariate system $\{P, Q\}$. We actually present two algorithms, a deterministic one of worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ and a probabilistic Las Vegas variant of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ (Theorems 26 and 27).

Our approach is based on the algorithms presented in [BLPR15] and [BLP+14] while improving the worst-case bit complexity by one order of magnitude. We briefly recall the essence of these algorithms. The first step of the algorithm presented in [BLPR15] is to compute the number of distinct solutions and a so-called lucky prime for the system. Such a lucky prime is, roughly speaking, a prime such that the system has the same number of distinct solutions as its image modulo μ (see Definition 18). In a second step, all polynomials and computations are considered modulo μ. The algorithm then considers iteratively a candidate separating form $x + ay$ with an integer a incrementing from 0. The algorithm computes the number of distinct solutions after projection along the direction of the line $x + ay = 0$ and stops when a value a is found such that the number of distinct projected solutions equals that of the system. The worst-case bit complexity of this algorithm is in $\widetilde{O}_B(d^8 + d^7\tau)$.

The main additional ideas introduced in [BLP+14] are that it is sufficient to compute a separating form for the system $\{H, \frac{\partial H}{\partial y}\}$ of critical points of a curve H associated to the input system $\{P, Q\}$ (see Section 5.2) and that the number of critical points can easily be computed as the difference between the degrees of the triangular decompositions of the systems $\{H, (\frac{\partial H}{\partial y})^2\}$ and $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 11). This improves the worst-case bit complexity of the algorithm to $\widetilde{O}_B(d^7 + d^6\tau)$.

In Section 5.3, we show how these algorithms can be again improved by one order of magnitude in the worst case. The main ideas of these improvements are as follows. First, in Section 5.3.1, we show how our improvement on the complexity analysis of triangular decompositions presented in Section 4.4 improves the complexity of the computation of the number of solutions of $\{H, \frac{\partial H}{\partial y}\}$.

In Section 5.3.2, we present a new algorithm for computing a lucky prime for the system $\{H, \frac{\partial H}{\partial y}\}$ using the luckiness certificates for triangular decompositions presented in Section 4.3. More precisely, we compute a lucky prime for $\{H, \frac{\partial H}{\partial y}\}$ by computing a prime μ that, essentially, does not divide the product of the luckiness certificates of the two systems $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$. By definition of the luckiness certificates, the degrees of the triangular decompositions of these two systems are the same over $\mathbb{Z}$ and $\mathbb{F}_\mu = \mathbb{Z}/\mu\mathbb{Z}$. The difference of these degrees, which is the number

of solutions of $\{H, \frac{\partial H}{\partial y}\}$, is thus also the same over $\mathbb{Z}$ and $\mathbb{F}_\mu$, which essentially yields that μ is lucky for $\{H, \frac{\partial H}{\partial y}\}$.

The last ingredient of our algorithm is to show, in Section 5.3.3, how, given the number of solutions and a lucky prime for the system $\{H, \frac{\partial H}{\partial y}\}$, the bit complexity of the algorithm presented in [BLPR15] for computing a separating linear form for $\{H, \frac{\partial H}{\partial y}\}$ can be improved from $\widetilde{O}_B(d^8+d^7\tau)$ to $\widetilde{O}_B(d^6+d^5\tau)$ by using multipoint evaluation and changing the organization of the computations.

In Section 5.4, we wrap up these results, which prove that we can compute a separating linear form for the input system $\{P, Q\}$ in $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case (Theorem 26).

Finally, we show in Section 5.5 that our deterministic algorithm can be modified in a straightforward manner into a probabilistic Las Vegas algorithm of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ (Theorem 27). This is done by choosing randomly a linear form $x + ay$ and a prime μ for the system $\{H, \frac{\partial H}{\partial y}\}$, until the number of distinct solutions of $\{H, \frac{\partial H}{\partial y}\}$ is equal to the number of distinct solutions of that system modulo μ and after projection along the direction of the line $x + ay = 0$.

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

2.3 Multimodular RUR decomposition

We present in Section 6 a new algorithm for computing a rational parameterization of the solutions of a bivariate system $\{P, Q\}$. As in Section 5, we actually present two algorithms, a deterministic one of worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ and a probabilistic Las Vegas variant of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ (Theorems 43 and 44). We consider here that a separating form $x + ay$ has been computed for $\{P, Q\}$ as shown in Section 5.

Recall that the two algorithms with best known bit complexity for computing rational parameterizations of the solutions are those by Gonzalez-Vega and El Kahoui [GVEK96] and by Bouzidi et al. [BLPR15], both of complexity $\widetilde{O}_B(d^7 + d^6\tau)$. The former algorithm first shears the input polynomials according to the separating form (to ensure that no two solutions are vertically aligned) and then computes a parameterization of the solutions of every system of the triangular decomposition of the sheared system; the multiplicities of the solutions of $\{P, Q\}$ are thus not preserved. The latter algorithm computes a single RUR of $\{P, Q\}$, which preserves the multiplicities of the solutions (see Proposition 33). In this latter approach, the input bivariate polynomials are formally sheared using an additional variable that parameterizes a generic linear form, and the resultant of these (trivariate) polynomials is computed. The polynomials of the RUR are then expressed (and computed) as combinations of this resultant and its partial derivatives, specialized at the value a associated with the given linear separating form.

Here, we combine in essence these two approaches and compute a RUR for every system of the triangular decomposition of the sheared input system. However, in order to obtain the claimed complexities, we do not compute a RUR of every triangular system using the approach of [BLPR15]. Instead, Algorithm 6 works as follows. First, we compute the triangular decomposition of the sheared input system as in [GVEK96]. We then show that the radical ideals of the triangular systems can easily be obtained (Lemma 37). Using the simple structure of these radical ideals, we

derive formulas for their RURs (Lemma 38). For complexity issues, we do not use these formulas to directly compute RURs over $\mathbb{Q}$ but we use instead a multi-modular approach. For that purpose, we use known bounds on the size of the RUR coefficients and the luckiness certificate of the triangular decomposition introduced in Section 4.3 to select the primes.

Finally, we show in Section 6.3 how our deterministic algorithm can be transformed into a probabilistic Las Vegas one of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$. In order to obtain this complexity, we cannot compute the triangular decomposition as described above. Instead, we show in Section 6.3.1 that we can only compute the coefficients of these triangular systems that are needed for obtaining their radicals, within the targeted bit complexity. We also choose randomly the primes in the multi-modular computation described above. This can be done in a Las Vegas setting because we show that we can choose good primes with sufficiently high probability and that we can check whether the primes are good within the targeted complexity.

2.4 Computing isolating boxes from a RUR decomposition

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

An algorithm with bit complexity $\widetilde{O}_B(d^8+d^7\tau)$ was presented in [BLPR15, §5.1 and Proposition 35] for the isolation of the real solutions of a system $\{P, Q\}$ of two bivariate polynomials of degree bounded by d and bitsize bounded by τ. Section 7.2 presents a modified algorithm that isolates all complex solutions. Using several amortized bounds for the roots of polynomials (Section 7.1), we show that Algorithm 7 applied to a RUR decomposition of a system $\{P, Q\}$, isolates all complex solutions in $\widetilde{O}_B(d^6 + d^5\tau)$ (Theorem 59).

3 Notation and preliminaries

We introduce notation and recall some classical material about subresultants, gcds, lucky primes for gcd computations, and multiplicities. Experienced readers can skip this classical material at first and later return to it for reference.

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

In this paper, we consider algorithms both in the worst case and the probabilistic Las Vegas setting. Recall that in Las Vegas algorithms the sequence and number of operations are probabilistic but the output is deterministic. The expected complexities of these algorithms refer to average number of bit operations that are performed for distributions of random variables considered in the process of the algorithms; these expected complexities hold without any assumption on the distribution of the input.

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

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

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

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

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

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

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

Definition 1. *([EK03, §3]). For $i = 0, \ldots, \min(q, p-1)$, the i-th polynomial subresultant of P and Q, denoted by $\mathrm{Sres}_{y,i}(P,Q) = \mathrm{sres}_{y,i}(P,Q)y^i + \mathrm{sres}_{y,i,i-1}(P,Q)y^{i-1} + \cdots + \mathrm{sres}_{y,i,0}(P,Q)$ is the polynomial determinant of $\mathrm{Syl}_{y,i}(P,Q)$.*

For practical consideration, when $q = p$, we define the q-th polynomial subresultant of P and Q as Q.[2] The polynomial $\mathrm{Sres}_i(P,Q)$ has degree at most i in y and it can be written as $\mathrm{sres}_{y,i}(P,Q)y^i + \mathrm{sres}_{y,i,i-1}(P,Q)y^{i-1} + \cdots + \mathrm{sres}_{y,i,0}(P,Q)$, where the coefficient of its monomial of degree i in y, $\mathrm{sres}_i(P,Q)$, is called the i-th *principal subresultant coefficient*. Unless specified otherwise, the subresultants are always considered with respect to the variable y and then, for simplicity, we do not explicitly refer to the variable in the notation. Note that $\mathrm{Sres}_0(P,Q) = \mathrm{sres}_0(P,Q)$ is the *resultant* of P and Q with respect to y, which we also denote by $\mathrm{Res}_y(P,Q)$.

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

Again, when the resultant is considered with respect to y, we omit the reference to the variable and denote it by $\mathrm{Res}(P,Q)$.

The matricial definition of subresultants implies the so-called *specialization property* of subresultants, that is $\phi(\mathrm{Sres}_i(P,Q)) = \mathrm{Sres}_i(\phi(P),\phi(Q))$ for any morphism ϕ between $\mathbb{D}$ and another unique factorization domain $\mathbb{D}'$ such that none of the leading coefficients of P and Q vanishes through ϕ. More generally, the equality holds up to a non-zero multiplicative constant in $\mathbb{D}'$ when only one of the leading coefficients vanishes [EK03, Lemmas 2.3, 3.1].

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

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

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

Proof. This property is a direct consequence of the specialization property of subresultants and of the gap structure theorem; see for instance [EK03, Lemmas 2.3, 3.1 and Cor. 5.1]. □

Note that this lemma is often stated with a stronger assumption, that is, that *none* of the leading coefficients $a_p(\alpha)$ and $b_q(\alpha)$ vanishes.

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

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

- *The coefficients of $\mathrm{Sres}_i(P,Q)$ have bitsize in $\widetilde{O}(d_y\tau)$.*
- *The degree in x_j of $\mathrm{Sres}_i(P,Q)$ is at most $2d(d_y-i)$.*
- *For any $i \in \{0,\ldots,\min(\deg_y(P),\deg_y(Q))\}$, the polynomial $\mathrm{Sres}_i(P,Q)$ can be computed in $\widetilde{O}(d^n d_y^{n+1})$ arithmetic operations and $\widetilde{O}_B(d^n d_y^{n+2}\tau)$ bit operations. These complexities also hold for the computation of the sequence of principal subresultant coefficients $\mathrm{sres}_i(P,Q)$.*[3]

In the univariate case, we need a refinement of the previous lemma in the case of two polynomials with different degrees and bitsizes. In addition, we often consider the gcd of two univariate polynomials P and Q and the gcd-free part of P with respect to Q, that is $\frac{P}{\gcd(P,Q)}$. Note that when $Q = P'$, the latter is the squarefree part of P. Since the gcd and gcd-free part can be computed via subresultants, we summarize all these complexity results in the following lemma. Since we do not

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

know a proper reference for these results in the case of different degrees and bitsizes, we provide a proof.

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

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

Proof. Using the well-known half-gcd approach, the algorithm in [LR01] computes any polynomial in the Sylvester-Habicht and cofactors sequence in a softly-linear number of arithmetic operations, and it exploits Hadamard's inequality on the Sylvester matrix to bound the size of the coefficients. The Sylvester-Habicht sequence is a signed variant of the subresultant sequence thus the same complexity bounds apply for both. The same approach is also used in [vzGG13, §11] to compute the sequence of principal subresultant coefficients.

When the two input polynomials have different degrees and bitsizes, Hadamard's inequality reads as $\widetilde{O}(p\tau_Q + q\tau_P)$ instead of simply $\widetilde{O}(d\tau)$ when both polynomials have degree bounded by d and bitsize bounded by τ. Using the Chinese Remainder Algorithm, the algorithms in [LR01] and in [vzGG13, §11] hence compute any subresultant polynomial as well as the sequence of principal subresultant coefficients in $\widetilde{O}_B(p(p\tau_Q + q\tau_P))$ bit operations instead of simply $\widetilde{O}(d^2\tau)$. One subresultant and a cofactor are, up to integer factors, the gcd and gcd-free part of P and Q ([BPR06, Prop. 10.14]). These polynomials in $\mathbb{Z}[y]$ are thus computed in $\widetilde{O}_B(p(p\tau_Q + q\tau_P))$ and have bitsize in $\widetilde{O}(p\tau_Q + q\tau_P)$. On the other hand, Mignotte's lemma (see e.g. [BPR06, Corollary 10.12]) gives the stated better bounds for the bitsize of the gcd and the gcd-free part. Thus, dividing the computed polynomials by the gcd of their coefficients, which can be done with $\widetilde{O}_B(p(p\tau_Q + q\tau_P))$ bit operations, yields the primitive parts of the gcd and gcd-free part in $\mathbb{Z}[y]$ (when input polynomials are not primitive, the gcd is obtained by multiplying this primitive gcd by the gcd of the contents of the input polynomials). □

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

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

Proof. [MSW15, Lemma 13] proves that G, the gcd of P and Q, can be computed in $\widetilde{O}_B(d^6 + d^5\tau)$ bit complexity. More precisely, they prove a complexity in $\widetilde{O}_B(d^5 + d^4\tau)$ plus that of computing the whole subresultant sequence of two bivariate polynomials of total degree $O(d)$ and bitsize $O(d+\tau)$ (that is of P and Q sheared so that their leading coefficients in y are in $\mathbb{Z}$). However, only the first non-zero subresultant is needed and the bit complexity of this computation is in $\widetilde{O}_B(d^5 + d^4\tau)$ by Lemma 3.

We now consider P and G, the gcd of P and Q, as polynomials in y with coefficients in $\mathbb{Z}[x]$. The gcd-free part of P with respect to Q is the quotient of the Euclidean division between P and G. This division can be run in $\mathbb{Z}[x][y]$. Indeed, since the leading coefficient of G divides that of P, it also divides the leading coefficient of each intermediate remainder $r_i = P - q_i G$ where q_i refers

to the i-th truncation (with respect to y) of the quotient of P by G. Moreover, since G divides P and by Mignotte's lemma (see e.g. [BPR06, Corollary 10.12]), the polynomials q_i have coefficients of degree in $O(d)$ in x and bitsize in $O(d+\tau)$, and so as for the intermediate remainders r_i. The Euclidean division can thus be done using $O(d^2)$ additions and multiplications and $O(d)$ exact divisions between polynomials in $\mathbb{Z}[x]$ of degree in $O(d)$ and bitsize in $O(d+\tau)$, which yields a bit complexity in $\widetilde{O}_B(d^4+d^3\tau)$.

Note that, alternatively, the gcd-free parts of P and Q could be obtained almost directly as i-th subresultant cofactors of P and Q (see [BPR06, Proposition 10.14 & Corollary 8.32 & §1.2]). □

Lucky primes for gcd computations. We use in this paper three notions of lucky primes. We recall here the definition of lucky primes for gcds and we later introduce the definition of lucky primes for algebraic systems (Definition 18) and for triangular decompositions (Definition 13). Let A and B be polynomials in $\mathbb{Z}[x]$.

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

- $\phi_\mu(\mathrm{Lc}(A)\cdot\mathrm{Lc}(B))\neq 0$, *and*
- $\gcd(A,B)$ *has the same degree as* $\gcd(A_\mu,B_\mu)$.

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

Multiplicities. We define the two notions of multiplicities that we use for the solutions of a system and show an inequality that they satisfy, which is used for the amortized complexity analysis of the triangular decomposition (Proposition 15).

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

Lemma 9. *Let (α,β) be a solution of the system $\{P,Q\}$. The multiplicity of (α,β) in the system $\{P,Q\}$ is larger than or equal to the multiplicity of β in the univariate polynomial* $\gcd(P(\alpha,y),Q(\alpha,y))$.

Proof. Let (α,β) be a solution of the system $\{P,Q\}$. We have the inclusion of ideals

$$\begin{aligned}\langle P(x,y),Q(x,y)\rangle &\subseteq \langle P(x,y),Q(x,y),x-\alpha,\gcd(P(\alpha,y),Q(\alpha,y))\rangle\\ &\subseteq \langle P(\alpha,y),Q(\alpha,y),x-\alpha,\gcd(P(\alpha,y),Q(\alpha,y))\rangle\\ &\subseteq \langle x-\alpha,\gcd(P(\alpha,y),Q(\alpha,y))\rangle.\end{aligned}$$

Indeed, the first and last inclusions are trivial and the second one follows from the fact that $P(x,y)\in\langle P(\alpha,y),x-\alpha\rangle$ since $P(x,y)$ can be written as $P(\alpha,y)+\sum_{i\geqslant 1}\frac{\partial^i P(\alpha,y)}{\partial x^i}(x-\alpha)^i$. This ideal inclusion implies that the multiplicity of (α,β) in $\langle P,Q\rangle$ is larger than or equal to its multiplicity in $\langle x-\alpha,\gcd(P(\alpha,y),Q(\alpha,y))\rangle$, which is equal to the multiplicity of β in $\gcd(P(\alpha,y),Q(\alpha,y))$ since $x-\alpha$ is squarefree. □

4 Triangular decomposition and luckiness certificate

This section presents an improved complexity analysis of the classical triangular decomposition via subresultants and two variants of this algorithm that we will need in the following sections. The improvement comes from new amortized bounds on the degree and bitsize of factors of the resultant, which we prove in Proposition 15. Besides of being of independent interest, this improvement is critical for the complexity analysis of our two variants of this algorithm.

In Section 4.1, we recall Algorithm 1, the classical algorithm for computing the triangular decomposition of a zero-dimensional bivariate system $\{P, Q\}$. In Section 4.2, we present Algorithm 1', a variant that only computes the "degree" of the triangular decomposition. This algorithm is identical to the one we presented in [BLP+14, Algorithm 3], however, we improve here its complexity analysis by a factor d in the worst case. In Section 4.3, we present Algorithm 2, another variant that computes a luckiness certificate (see Definition 13) for the triangular decomposition. Finally, in Section 4.4, we present an amortized complexity analysis of these algorithms.

4.1 Triangular decomposition via subresultants

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

Lemma 10 ([GVEK96, LMMRS11])**.** *Algorithm 1 computes a triangular decomposition* $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$ *such that*

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

4.2 Degree of a triangular decomposition

Definition 11. *The degree of a triangular decomposition* $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$ *of is the sum of the degrees of these systems, that is*

$$\sum_{i\in\mathcal{I}} \deg_x(A_i(x))\ \deg_y(B_i(x, y))$$

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

Algorithm 1 Triangular decomposition [GVEK96, LMMRS11]

Input: P, Q in $\mathbb{D}[x, y]$ coprime such that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime.
Output: Triangular decomposition $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$ such that the set of solutions of $\{P, Q\}$ is the disjoint union of the sets of solutions of $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$.

1: If needed, exchange P and Q so that $\deg_y(Q) \leqslant \deg_y(P)$.
2: Compute the subresultant sequence of P and Q with respect to y: $B_i = \mathrm{Sres}_i(P, Q)$.
3: $G_0 =$ squarefree part$(\mathrm{Res}(P, Q))$ and $\mathcal{T} = \emptyset$
4: **for** $i = 1$ **to** $\deg_y(Q)$ **do**
5: $\quad G_i = \gcd(G_{i-1}, \mathrm{sres}_i(P, Q))$
6: $\quad A_i = G_{i-1}/G_i$
7: $\quad$ if $\deg_x(A_i) > 0$, add (A_i, B_i) to $\mathcal{T}$
8: **return** $\mathcal{T} = \{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$

Algorithm 1' Degree of the triangular decomposition

Input: P, Q in $\mathbb{D}[x, y]$ coprime such that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime.
Output: The degree of the triangular decomposition of $\{P, Q\}$.

1: If needed, exchange P and Q so that $\deg_y(Q) \leqslant \deg_y(P)$.
2: Compute $(\mathrm{sres}_i(P, Q))_{i=0,\ldots,\deg_y(Q)}$ the sequence of principal subresultant coefficients of P and Q with respect to y.
3: $G_0 =$ squarefree part$(\mathrm{Res}(P, Q))$
4: **for** $i = 1$ **to** $\deg_y(Q)$ **do**
5: $\quad G_i = \gcd(G_{i-1}, \mathrm{sres}_i(P, Q))$
6: **return** $\sum_{i\in\mathcal{I}}(\deg_x(G_{i-1}) - \deg_x(G_i))\, i$

Algorithm 1', a straightforward variant of Algorithm 1, computes the degree of triangular decomposition of $\{P, Q\}$. The difference with Algorithm 1 is that we do not compute (in Line 2) the whole subresultant sequence but only the sequence of their principal coefficients. In other words, we do not compute the bivariate polynomials $B_i(x, y)$ of the triangular decomposition but only their leading terms (seen as polynomials in y). Furthermore, we do not compute the univariate polynomials $A_i(x)$ of the decomposition but only their degrees. This simplification does not modify the worst-case bit complexity of the algorithm but it decreases its expected bit complexity (see Proposition 16 and its proof). This simplification is thus not needed in the deterministic version of our algorithm for computing a separating linear form but it is needed in our randomized version (see Section 5.5).

Lemma 12 (Correctness of Algorithm 1'). *Algorithm 1' computes the degree of the triangular decomposition of* $\{P, Q\}$.

Proof. Let $\{(A_i(x), B_i(x, y))\}_{i\in\mathcal{I}}$ denote the triangular decomposition of $\{P, Q\}$. By Lemma 10, $B_i(x, y)$ is of degree i in y. On the other hand, $A_i(x)$ is defined in Algorithm 1 Line 6 as G_{i-1}/G_i thus its degree is $\deg_x(G_{i-1}) - \deg_x(G_i)$. It follows that the degree of the triangular decomposition is $\sum_{i\in\mathcal{I}}(\deg_x(G_{i-1}) - \deg_x(G_i))\, i$. □

4.3 Lucky primes for a triangular decomposition

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

Algorithm 2 Luckiness certificate

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

Output: A luckiness certificate of $\{P, Q\}$, that is, an integer Π such that if μ does not divide Π, then μ is lucky for the triangular decomposition of $\{P, Q\}$ according to Definition 13.

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

Definition 13. *A prime μ is* **lucky for the triangular decomposition** *of Algorithm 1 applied to P and Q if the decomposition commutes with the morphism ϕ_μ and its degree is invariant through ϕ_μ. More precisely, with $\{(A_i, B_i)\}_{i\in\mathcal{I}} = \text{Algorithm 1}(\mathrm{P}, \mathrm{Q})$ and $\{(A_i^\mu, B_i^\mu)\}_{i\in\mathcal{I}^\mu} = \text{Algorithm 1}(\phi_\mu(\mathrm{P}), \phi_\mu(\mathrm{Q}))$, μ is lucky if $\mathcal{I} = \mathcal{I}^\mu$, $\phi_\mu(A_i) = A_i^\mu$, $\phi_\mu(B_i) = B_i^\mu$ for every $i \in \mathcal{I}$ and the two triangular decompositions have the same degree. Note also that (P, Q) and $(\phi_\mu(P), \phi_\mu(Q))$ are required to satisfy the hypotheses of Algorithm 1.*

Lemma 14 (Correctness of Algorithm 2)**.** *The integer Π output by Algorithm 2 is a luckiness certificate of $\{P, Q\}$, that is, if μ does not divide Π, then it is lucky for the triangular decomposition of $\{P, Q\}$.*

Proof. For convenience, we simply denote by ϕ the morphism ϕ_μ that performs a reduction modulo μ. Let P and Q be two coprime polynomials in $\mathbb{Z}[x, y]$ such that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime. Let G_i, A_i, B_i be the polynomials computed in Algorithm 1 on the input P and Q, and G_i^μ, A_i^μ, B_i^μ be the polynomials computed in Algorithm 1 on the input $\phi(P)$ and $\phi(Q)$.

We first prove that $\phi(P)$ and $\phi(Q)$ satisfy the conditions of Algorithm 1, that is that they are coprime and that their leading coefficients are coprime. Observe first that $\phi(\mathrm{Lc}_y(P)) = \mathrm{Lc}_y(\phi(P))$ since μ does not divide $\mathrm{Lc}_x(\mathrm{Lc}_y(P))$, and similarly for Q. Furthermore, since μ does not divide the leading coefficients of $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$, their resultant and ϕ commute. Hence, $\phi(\mathrm{Res}_x(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))) = \mathrm{Res}_x(\mathrm{Lc}_y(\phi(P)), \mathrm{Lc}_y(\phi(Q)))$ and, since the left-hand-side term is non-zero by assumption, so is the right-hand side, which means that that the leading coefficients of $\phi(P)$ and $\phi(Q)$ are coprime. Furthermore, since μ does not divide $\mathrm{sres}_0(P, Q) = \mathrm{Res}(P, Q)$, we also have that $\mathrm{Res}(\phi(P), \phi(Q)) \not\equiv 0$. We have proved that $\phi(P)$ and $\phi(Q)$ have a non-zero resultant and that their leading coefficients are coprime, which implies that $\phi(P)$ and $\phi(Q)$ are coprime. Hence, they satisfy the conditions of Algorithm 1.

We now prove that $\phi(A_i) = A_i^\mu$, $\phi(B_i) = B_i^\mu$ and $\deg_x(A_i) = \deg_x(A_i^\mu)$ for all $i > 0$. Since μ divides neither the leading coefficients of P nor Q, the specialization property of the subresultant polynomials writes as $\phi(B_i) = \phi(\mathrm{Sres}_i(P, Q)) = \mathrm{Sres}_i(\phi(P), \phi(Q)) = B_i^\mu$. We now show by induction on $i \geqslant 0$ that $\phi(G_i) = G_i^\mu$ and $\deg_x(G_i) = \deg_x(G_i^\mu)$, which implies that $\phi(A_i) = A_i^\mu$ and $\deg_x(A_i) = \deg_x(A_i^\mu)$ for $i > 0$ since $A_i = G_{i-1}/G_i$.

Case $i = 0$. μ is lucky for the gcd of $\mathrm{sres}_0(P,Q)$ and $\frac{\partial \mathrm{sres}_0(P,Q)}{\partial x}$ by Lemma 7. Indeed, first, μ does not divide the leading coefficient $\mathrm{Lc}_x(\mathrm{sres}_0(P,Q))$ of $\mathrm{sres}_0(P,Q)$. It follows that μ does not divide the leading coefficient of $\frac{\partial \mathrm{sres}_0(P,Q)}{\partial x}$ since μ does not divide $\deg_x(\mathrm{Res}(P,Q)) = \deg_x(\mathrm{sres}_0(P,Q))$. Finally, μ does not divide SG_0. It follows, still by Lemma 7, that ϕ and gcd commute on $\mathrm{sres}_0(P,Q)$ and $\frac{\partial \mathrm{sres}_0(P,Q)}{\partial x}$. Hence, $\phi(G_0) = G_0^\mu$ by the specialization property of the subresultants since the leading coefficients of P, Q do not vanish modulo μ.

We now prove that $\deg_x(G_0) = \deg_x(G_0^\mu)$, which is now equivalent to proving that $\deg_x(G_0) = \deg_x(\phi(G_0))$. Since the image through ϕ of any polynomial does not increase its degree, $\deg_x(\phi(G_0)) \leqslant \deg_x(G_0)$. Furthermore, $\deg_x(\phi(G_0)) \geqslant \deg_x(G_0)$, because $G_0 = \frac{\mathrm{Res}(P,Q)}{\gcd(\mathrm{Res}(P,Q), \frac{\partial \mathrm{Res}(P,Q)}{\partial x})}$ and $\mathrm{Res}(P,Q)$ and its image through ϕ have the same degree (since μ does not divide the leading coefficient of $\mathrm{Res}(P,Q)$).

Case $i > 0$. We assume that $\phi(G_{i-1}) = G_{i-1}^\mu$ and $\deg_x(G_{i-1}) = \deg_x(G_{i-1}^\mu)$. By Lemma 7, μ is lucky for the gcd of G_{i-1} and $\mathrm{sres}_i(P,Q)$. Indeed, μ divides none of the leading coefficients of G_{i-1} and $\mathrm{sres}_i(P,Q)$ (since G_{i-1} is a factor of $\mathrm{sres}_{i-1}(P,Q)$), and μ does not divide SG_i either. This implies, still by Lemma 7, that $\phi(G_i) = \phi(\gcd(G_{i-1}, \mathrm{sres}_i(P,Q))) = \gcd(\phi(G_{i-1}), \phi(\mathrm{sres}_i(P,Q)))$. This is also equal to $\gcd(G_{i-1}^\mu, \mathrm{sres}_i(\phi(P), \phi(Q))) = G_i^\mu$ by the induction hypothesis and the property of specialization of the subresultants. Hence, $\phi(G_i) = G_i^\mu$. Furthermore, since μ is lucky for the gcd of G_{i-1} and $\mathrm{sres}_i(P,Q)$, this gcd, which is G_i by definition, and $\gcd(\phi(G_{i-1}), \phi(\mathrm{sres}_i(P,Q))) = G_i^\mu$ have the same degree by Definition 6. This concludes the proof of the induction.

We have proved that $\phi(A_i) = A_i^\mu$, $\phi(B_i) = B_i^\mu$ and $\deg_x(A_i) = \deg_x(A_i^\mu)$ for all $i > 0$. The latter property directly implies that $\mathcal{I} = \mathcal{I}^\mu$. Now, for $i \in \mathcal{I} = \mathcal{I}^\mu$, the degrees in y of B_i and B_i^μ are equal to i by Lemma 10 (and Definition 1). This implies that the degrees of the decompositions $\{(A_i, B_i)\}_{i\in\mathcal{I}}$ and $\{(A_i^\mu, B_i^\mu)\}_{i\in\mathcal{I}^\mu}$ are the same, which concludes the proof.

□

4.4 Amortized complexity analysis

For the analysis of Algorithms 1, 1' and 2, we first prove amortized bounds on the degree and bitsize of the factors G_i of the resultant in the triangular decomposition.

Proposition 15. *For $i = 0, \ldots, \deg_y(Q)-1$, let d_i and τ_i be the degree and bitsize of the polynomial G_i in the triangular decomposition of P and Q computed in Algorithm 1. We have:*

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

Proof. Let $\{(A_i(x), B_i(x,y))\}_{i\in\mathcal{I}}$ be the sequence of triangular systems output by Algorithm 1 on P and Q. By the properties of the triangular decomposition (Lemma 10), for any root α of A_i,

$$\deg_y(B_i(\alpha, y)) = i \text{ and } B_i(\alpha, y) = \gcd(P(\alpha, y), Q(\alpha, y))$$

up to the multiplication by a constant factor. Thus, for any root α of A_i,

$$\sum_{\beta \text{ s.t. } B_i(\alpha,\beta)=0} \mathrm{mult}(\beta, \gcd(P(\alpha, y), Q(\alpha, y))) = i$$

and, by Lemma 9,

$$\sum_{\beta \text{ s.t. } B_i(\alpha,\beta)=0} \mathrm{mult}((\alpha, \beta), \{P, Q\}) \geqslant i.$$

This latter sum is the multiplicity of α in the resultant $\mathrm{Res}(P,Q)$ because the set of solutions of $\{P,Q\}$ is the disjoint union of the sets of solutions of the $\{(A_i(x),B_i(x,y))\}_{i\in\mathcal{I}}$ (Lemma 10). Hence the multiplicity in $\mathrm{Res}(P,Q)$ of any root α of A_i is at least i and since A_i is squarefree (Lemma 10), A_i^i divides $\mathrm{Res}(P,Q)$. In addition, the A_i are pairwise coprime thus $\prod_{i=1}^{\deg_y(Q)} A_i^i$ divides $\mathrm{Res}(P,Q)$. On the other hand, $G_i = \prod_{j>i} A_j$ by construction, thus $\prod_{i=0}^{\deg_y(Q)-1} G_i = \prod_{i=1}^{\deg_y(Q)} A_i^i$ divides $\mathrm{Res}(P,Q)$.

The bound on the degrees, $\sum_{i=0}^{\deg_y(Q)-1} d_i \leqslant \deg_x(\mathrm{Res}(P,Q)) \leqslant d^2$, is then a consequence of Bézout's bound on the system $\{P,Q\}$. In addition, A_i^i divides $\mathrm{Res}(P,Q)$ implies that $G_i^{i+1} = \prod_{j>i} A_j^{i+1}$ also divides $\mathrm{Res}(P,Q)$, which yields $d_i \leqslant \frac{d^2}{i+1}$.

For proving the bounds on the bitsize of G_i, we introduce Mahler's measure. For a univariate polynomial f with integer coefficients, its Mahler measure is $M(f) = |\mathrm{Lc}(f)| \prod_{z_i \text{ s.t. } f(z_i)=0} \max(1,|z_i|)$, where every complex root appears with its multiplicity. Mahler's measure is multiplicative: $M(fg) = M(f)M(g)$ and, since it is at least 1 for any polynomial with integer coefficients, f divides g implies that $M(g) \geqslant M(f)$. We also prove two inequalities connecting the bitsize τ and degree d of f and its Mahler measure $M(f)$.

(i) $\tau \leqslant 1+d+\log M(f)$. Indeed, [BPR06, Prop. 10.8] states that $||f||_1 \leqslant 2^d M(f)$, thus $||f||_\infty \leqslant 2^d M(f)$ and $\log ||f||_\infty \leqslant d + \log M(f)$, which yields the result since $\tau = \lfloor \log ||f||_\infty \rfloor + 1$.

(ii) $\log M(f) = O(\tau + \log d)$. Indeed, [BPR06, Prop. 10.9] states that $M(f) \leqslant ||f||_2$, thus $M(f) \leqslant \sqrt{d+1}||f||_\infty$ and $\log M(f) \leqslant \log\sqrt{d+1} + \log ||f||_\infty$.

The fact that G_i^{i+1} divides $\mathrm{Res}(P,Q)$ implies that $M(G_i)^{i+1} \leqslant M(\mathrm{Res}(P,Q))$ and thus that $\log M(G_i) \leqslant \frac{\log M(\mathrm{Res}(P,Q))}{i+1}$. Inequality (i) together with $d_i \leqslant \frac{d^2}{i+1}$ then yields

$$\tau_i \leqslant 1 + \frac{d^2}{i+1} + \frac{\log M(\mathrm{Res}(P,Q))}{i+1}.$$

Inequality (ii) then yields $\tau_i = \widetilde{O}(\frac{d^2+d\tau}{i+1})$ since the bitsize of $\mathrm{Res}(P,Q)$ is in $\widetilde{O}(d\tau)$. The bound on the sum of the bitsizes is then straightforward using the fact that $\sum_{i=0}^{\deg_y(Q)-1} \frac{1}{i+1} = O(\log d)$. □

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

Proof. By Lemma 3, the sequence of the subresultants $\mathrm{Sres}_i(P,Q)$ can be computed in $\widetilde{O}_B(d^5\tau)$ bit operations and the sequence of their principal coefficients $\mathrm{sres}_i(P,Q)$ (including the resultant) can be computed in $\widetilde{O}_B(d^4\tau)$ bit operations. Thus, Line 2 has complexity $\widetilde{O}_B(d^5\tau)$ in Algorithm 1 and $\widetilde{O}_B(d^4\tau)$ in Algorithms 1' and 2.

By Lemma 3, each of the principal subresultant coefficients sres_i (including the resultant) has degree $O(d^2)$ and bitsize $\widetilde{O}(d\tau)$. Thus, by Lemma 4, in Line 3 of all three algorithms and in Line 4 of Algorithm 2, G_0 and SG_0 can be computed in $\widetilde{O}_B((d^2)^2(d\tau)) = \widetilde{O}_B(d^5\tau)$ bit operations.

In their loops, the three algorithms perform (in total) the computations of at most d gcd (or sequences of principal subresultant coefficients) between polynomials G_{i-1} and sres_i. Polynomial sres_i has bitsize $\widetilde{O}(d\tau)$ and degree $O(d^2)$, and denoting by τ_i and d_i the bitsize and degree of G_i, Lemma 4 yields a complexity in $\widetilde{O}_B(d^2(d^2\tau_{i-1} + d_{i-1}d\tau))$ for the computation of G_i and SG_i. According to Proposition 15, these complexities sum up over all i to $\widetilde{O}_B(d^6+d^5\tau)$. Finally, in Line 6 of Algorithm 1, the division of G_{i-1} by G_i can be done in a bit complexity of the order of the square of their maximum degree times their maximum bitsize [vzGG13, Theorem 9.6 and

subsequent discussion], that is in $O_B(d_i^2\tau_i)$ (or actually $O_B(d_i^2+d_i\tau_i)$ according to [vzGG13, Exercise 10.21]). By Proposition 15, $d_i \leqslant d^2$ and $\tau_i = \widetilde{O}(\frac{d^2+d\tau}{i+1})$, thus $\sum_i O_B(d_i^2\tau_i) = \widetilde{O}_B(d^6 + d^5\tau)$. Hence the worst-case bit complexity of all three algorithms is in $\widetilde{O}_B(d^6 + d^5\tau)$.

We now show that the expected bit complexity of Algorithm 1' is in $\widetilde{O}_B(d^5 + d^4\tau)$. As above the worst-case bit complexity of Line 2 is in $\widetilde{O}_B(d^4\tau)$. The rest of the algorithm performs $O(d)$ gcd computations and one exact division (in Line 3) between polynomials of degree $O(d^2)$ and bitsize $\widetilde{O}(d\tau)$. Each of these operations can be done with an expected bit complexity of $\widetilde{O}_B((d^2)^2 + d^2 \cdot d\tau)$ (the squared degree plus the degree times the bitsize) [vzGG13, Corollary 11.14 & Exercice 9.14]. The expected bit complexity of Algorithm 1' is thus in $\widetilde{O}_B(d^5 + d^4\tau)$.

Concerning the last claim of the proposition, recall that

$$\Pi = \mathrm{Lc}_x(\mathrm{Lc}_y(P))\cdot\mathrm{Lc}_x(\mathrm{Lc}_y(Q))\cdot\mathrm{Res}_x(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))\cdot\deg_x(\mathrm{Res}(P,Q))\cdot\prod_{i=0}^{\deg_y(Q)} SG_i\cdot\mathrm{Lc}_x(\mathrm{sres}_i(P,Q)).$$

Since P and Q have degree at most d and bitsize at most τ, the first two terms have bitsize at most τ and, by Lemma 4, $\mathrm{Res}_x(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))$ has bitsize $\widetilde{O}(d\tau)$. Furthermore, as noted above, every $\mathrm{sres}_i(P,Q)$ (including the resultant of P and Q) has degree has degree $O(d^2)$ and bitsize $\widetilde{O}(d\tau)$. In particular the bitsize of $\deg_x(\mathrm{Res}(P,Q))$ is in $O(\log d)$ and that of $\mathrm{Lc}_x(\mathrm{sres}_i(P,Q))$ is in $\widetilde{O}(d\tau)$. In addition, still by Lemma 4, SG_0 has bitsize $\widetilde{O}(d^2 \cdot d\tau)$. On the other hand, by Lemma 4, for $i \geqslant 1$, SG_i has bitsize $\widetilde{O}(d^2\tau_{i-1} + d_{i-1}d\tau)$ with d_i and τ_i the degree and bitsize of G_i. By Proposition 15, these bitsizes sum up to $\widetilde{O}(d^4 + d^3\tau)$. The bitsize of Π is bounded by the sum of all these bitsizes and is thus in $\widetilde{O}(d^4 + d^3\tau)$. □

Remark 17. *Following the proof for the expected complexity of Algorithm 1', we directly get that Algorithm 1, except for Line 2, performs $\widetilde{O}_B(d^5+d^4\tau)$ bit operations on average. This will be useful for the proof of complexity of Algorithm 6'.*

5 Separating linear form

This section presents a new algorithm of worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ for computing a separating linear form for a bivariate system of two coprime polynomials P, Q in $\mathbb{Z}[x, y]$ of total degree at most d and maximum bitsize τ (Theorem 26). We also present a randomized version of this algorithm of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ (Theorem 27).

As mentioned in Section 2, this algorithm is based on those presented in [BLPR15] and [BLP+14]. In Section 5.2, we improve a result from [BLP+14] showing that computing a separating linear form for a system $\{P, Q\}$ is essentially equivalent (in terms of asymptotic bit complexity) to computing a separating linear form for the critical points of a curve. Section 5.3 then presents our algorithm for computing a separating linear form for the critical points of such a curve. In Section 5.4, we gather our results for deterministically computing separating linear forms of bivariate systems. Finally, in Section 5.5, we present the randomized version of our algorithm.

5.1 Notation and definitions

We first introduce some notation and formally define lucky primes for a system. Given the two input polynomials P and Q, we consider the "generic" change of variables $x = t - sy$, and define the "sheared" polynomials $P(t - sy, y)$, $Q(t - sy, y)$, and their resultant with respect to y,

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

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

$$L_P(s) = \mathrm{Lc}_y(P(t - sy, y)), \quad L_Q(s) = \mathrm{Lc}_y(Q(t - sy, y)). \tag{1}$$

Note that these polynomials do not depend on t.

Definition 18 ([BLPR15, Definition 8])**.** *A prime number μ is said to be* **lucky for a zero-dimensional system** $\{P, Q\}$ *if $\{P, Q\}$ and $\{\phi_\mu(P), \phi_\mu(Q)\}$ have the same number of distinct solutions (in their respective algebraic closures), $\mu > 2d^4$ and $\phi_\mu(L_P(s))\ \phi_\mu(L_Q(s)) \not\equiv 0$.*

Note that we consider μ in $\Omega(d^4)$ in Definition 18 because, in Algorithm 5, we want to ensure that there exists, for the system $\{\phi_\mu(P), \phi_\mu(Q)\}$ (resp. $\{P, Q\}$), a separating form $x + ay$ with a in $\mathbb{F}_\mu$ (resp. $0 \leqslant a < \mu$ in $\mathbb{Z}$). The constant 2 in the bound $2d^4$ is an overestimate, which simplifies some proofs in [BLPR15].

Definition 19. *Let H be a polynomial in $\mathbb{Z}[x, y]$. A* **separating form for the curve** *defined by H is a separating form for the system $\{H, \frac{\partial H}{\partial y}\}$ of critical points of the curve.*

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

5.2 From a system to a curve

We prove here Proposition 20, which states that it is essentially equivalent from an asymptotic bit complexity point of view to compute a separating linear form for a system $\{P, Q\}$ and to compute a separating linear form for the critical points of a curve H. According to Definition 19, we refer to the latter as a separating linear form for H. The proof essentially follows that of [BLP+14, Lemma 7] but we improve by a factor d the complexity of computing the curve H.

The critical points of a curve of equation H are the solutions of the system $\{H, \frac{\partial H}{\partial y}\}$, thus computing a separating linear form for a curve amounts, by definition, to computing a separating linear form for a system of two equations. Conversely, a separating linear form for the curve PQ is also separating for the system $\{P, Q\}$ since any solution of $\{P, Q\}$ is also solution of PQ and of $\frac{\partial PQ}{\partial y} = P\frac{\partial Q}{\partial y} + \frac{\partial P}{\partial y}Q$.

However, it may happen that the curve PQ admits no separating linear form even if $\{P, Q\}$ admits one. Indeed, $\{P, Q\}$ can be zero-dimensional while PQ is not squarefree (and such that the infinitely many critical points cannot be separated by a linear form). Nevertheless, if P and Q are coprime and squarefree, then PQ is squarefree and thus it has finitely many singular points. Still the curve $H = PQ$ may contain vertical lines, and thus infinitely many critical points, but this issue can easily be handled by shearing the coordinate system.

Proposition 20. *Let P and Q be two coprime polynomials in $\mathbb{Z}[x, y]$ of maximum degree d and maximum bitsize τ. We can compute a shearing of the coordinate system from (x, y) to $(t = x + \alpha y, y)$, with α an integer in $[0, 2d]$, and a squarefree polynomial H in $\mathbb{Z}[t, y]$ of degree at most $2d$, bitsize $\widetilde{O}(d + \tau)$, with $\mathrm{Lc}_y(H)$ in $\mathbb{Z}$, so that any separating linear form for the zero-dimensional system $\{H, \frac{\partial H}{\partial y}\}$ is also separating for $\{P, Q\}$ after being sheared back. The worst-case complexity of this computation is in $\widetilde{O}_B(d^5 + d^4\tau)$.*

Proof. The proof is identical to the one of [BLP+14, Lemma 7] except for the fact that the complexity of the gcd-free computation stated in Lemma 5 improves the overall complexity by a factor d. □

Algorithm 3 Number of critical points of H

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

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

5.3 Separating linear form of a curve

In this section, we consider an arbitrary curve defined by H in $\mathbb{Z}[x, y]$ of degree d and bitsize τ, squarefree and with a constant leading coefficient in y. In particular, the polynomial H defined in Proposition 20 satisfies these two last conditions, which yield that the curve has a finite number of critical points. We show in the following three subsections that computing (i) the number of the critical points of H, (ii) a lucky prime for the system of critical points $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 18), and finally (iii) a separating form for the curve H (Definition 19) can be done with a bit complexity in $\widetilde{O}_B(d^6 + d^5\tau)$.

5.3.1 Number of critical points

Algorithm 3 computes the number of critical points of a curve H as the difference between the degrees of the triangular decompositions of the systems $\{H, (\frac{\partial H}{\partial y})^2\}$ and $\{H, \frac{\partial H}{\partial y}\}$. This algorithm is identical to the one we presented in [BLP+14, Algorithm 4], however, our improvement on the complexity analysis of the triangular decomposition (Proposition 16) immediately improves the complexity of this counting algorithm. The correctness and complexity of Algorithm 3 follows directly from that of [BLP+14, Algorithm 4] and from Proposition 16.

Proposition 21. *If H in $\mathbb{Z}[x, y]$ has degree d and bitsize τ, Algorithm 3 computes the number of distinct critical points of H in $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case and $\widetilde{O}_B(d^5 + d^4\tau)$ bit operations on average.*

5.3.2 Lucky prime for the system of critical points

Let Π = Algo 2$(H, \frac{\partial H}{\partial y}) \cdot$ Algo 2$(H, (\frac{\partial H}{\partial y})^2)$ be the product of the luckiness certificates output by Algo 2 for the triangular decompositions of $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$. Lemma 22 shows that we can easily check using a divisibility test on Π whether a prime number is lucky for the system $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 18). Algorithm 4 finds such a lucky prime by an iterative application of this divisibility test. To keep the complexity in the desired bound, the primes to be tested are grouped and a remainder tree is used for the computation of the reduction of Π modulo all the primes in a group. In the following, $L_H(s)$ and $L_{\frac{\partial H}{\partial y}}(s)$ are defined similarly as in Section 5.1.

Lemma 22. *Let μ be a prime such that $\mu > 2d^4$ and $\phi_\mu(L_H(s))\ \phi_\mu(L_{\frac{\partial H}{\partial y}}(s)) \not\equiv 0$. If μ does not divide Π then μ is lucky for the system $\{H, \frac{\partial H}{\partial y}\}$.*

Proof. If μ does not divide Π then it is a lucky prime for the triangular decompositions of $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$ (by Lemma 14). By definition of a lucky prime for a triangular decomposition (Definition 13), the degrees of the decompositions are the same over $\mathbb{Z}$ or $\mathbb{F}_\mu$. Algorithm 3 computes the number of solutions of the system $\{H, \frac{\partial H}{\partial y}\}$ only from these degrees and thus the results are the same over $\mathbb{Z}$ or $\mathbb{F}_\mu$. Together with the assumptions that $\mu > 2d^4$ and $\phi_\mu(L_H(s))\ \phi_\mu(L_{\frac{\partial H}{\partial y}}(s)) \not\equiv 0$, this yields that μ is lucky for the system $\{H, \frac{\partial H}{\partial y}\}$. □

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

Input: H in $\mathbb{Z}[x, y]$ of degree d and bitsize τ, squarefree such that $\mathrm{Lc}_y(H)$ is in $\mathbb{Z}$ and $\Pi = \text{Algo } 2(H, \frac{\partial H}{\partial y}) \cdot \text{Algo } 2(H, (\frac{\partial H}{\partial y})^2)$.

Output: A lucky prime μ for the system $\{H, \frac{\partial H}{\partial y}\}$.

1: Compute $L_H(s)$ and $L_{\frac{\partial H}{\partial y}}(s)$ (defined as in Section 5.1).
2: $m = 2d^4$
3: **while** true **do**
4: Compute the set B of the first $d^4 + d^3\tau$ primes $> m$.
5: **for all** μ in B **do**
6: Compute the reduction mod. μ of Π, L_H, $L_{\frac{\partial H}{\partial y}}$.
7: **if** $\phi_\mu(\Pi)\ \phi_\mu(L_H(s))\ \phi_\mu(L_{\frac{\partial H}{\partial y}}(s)) \not\equiv 0$ **then**
8: **return** μ
9: $m =$ the largest prime in B

Proposition 23. *Given H in $\mathbb{Z}[x, y]$ of degree d and bitsize τ, Algorithm 4 computes a lucky prime of bitsize $O(\log(d\tau))$ for the system $\{H, \frac{\partial H}{\partial y}\}$ using $\widetilde{O}_B(d^4 + d^3\tau)$ bit operations.*

Proof. The correctness of Algorithm 4 follows directly from Lemma 22 since the condition in Line 7 (together with $\mu > 2d^4$) matches exactly the assumptions of Lemma 22.

We now analyze the complexity of this algorithm. It is straightforward that, in Line 1, $L_H(s)$ and $L_{\frac{\partial H}{\partial y}}(s)$ can be computed with $\widetilde{O}_B(d^4 + d^3\tau)$ bit operations and that they have coefficients of bitsizes $\widetilde{O}(d + \tau)$ (see e.g. [BLPR15, Lemma 7]). Furthermore, since Π has bitsize $\widetilde{O}(d^4 + d^3\tau)$ (by Proposition 16), the number of prime divisors of Π, $L_H(s)$, and $L_{\frac{\partial H}{\partial y}}(s)$ is in $\widetilde{O}(d^4 + d^3\tau)$. In other words, there exists a function F that is polylogarithmic in d and τ such that the number of these prime divisors is less than $(d^4 + d^3\tau)\, F(d, \tau)$ (without $O(\cdot)$). Hence, the number of iterations of the loop in Line 3 is polylogarithmic in d and τ since it is bounded by F. This also implies that the primes computed in Line 4 are among the M first primes with $M = 2d^4 + (d^4 + d^3\tau)\, F(d, \tau)$. The bit complexity of computing the M first prime is in $\widetilde{O}_B(M) = \widetilde{O}_B(d^4 + d^3\tau)$ and their maximum is in $\widetilde{O}(M) = \widetilde{O}(d^4 + d^3\tau)$ [vzGG13, Theorem 18.10]. Thus, the total bit complexity of Line 4 is in $\widetilde{O}_B(d^4 + d^3\tau)$, the bitsize of every considered prime is in $O(\log(d\tau))$ and the sum of their bitsizes is in $\widetilde{O}(d^4 + d^3\tau)$.

The loop of line 5 consists in testing, for the $d^4 + d^3\tau$ primes of B, the non-vanishing of the reduction of the integer Π and of the two polynomials $L_H(s)$, $L_{\frac{\partial H}{\partial y}}(s)$. The product of Π and of all these coefficients has bitsize $\widetilde{O}(d^4 + d^3\tau)$ and it can be computed in bit complexity $\widetilde{O}_B(d^4 + d^3\tau)$. The reduction of this product in Line 6 modulo all the primes in B can be computed via a remainder tree in a bit complexity that is soft linear in the total bitsize of the input [MB74, Theorem 1], which is in $\widetilde{O}(d^4 + d^3\tau)$.

Hence, the bit complexity of one iteration of the loop of Line 3 is $\widetilde{O}_B(d^4 + d^3\tau)$ and since at most polylog$(d\tau)$ iterations are performed, the overall bit complexity of Algorithm 4 is $\widetilde{O}_B(d^4 + d^3\tau)$. □

5.3.3 Separating linear form of a curve

In this section, we assume that we have already computed, using Algorithms 3 and 4, the number of distinct (complex) critical points of a curve and a lucky prime μ for the system of critical points. With this information, Algorithm 4 of [BLPR15] computes a separating form with a bit complexity

Algorithm 5 Separating form of a curve

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

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

$\widetilde{O}_B(d^8 + d^7\tau)$. In this section, we slightly modify this algorithm to improve its complexity to $\widetilde{O}_B(d^6 + d^5\tau)$.

More precisely, Algorithm 4 of [BLPR15] computes a separating linear form for a system $\{P,Q\}$ by considering iteratively linear forms $x+ay$, where a is an integer incrementing from 0 and by computing the degree of the squarefree part of the reduction modulo μ of $R(t,a)$ until this degree is equal to the (known) number of distinct solutions of the system and such that $\phi_\mu(L_P(a))\ \phi_\mu(L_Q(a)) \neq 0$. Doing so, the algorithm computes a separating form for the system modulo μ, which, under the hypothesis of the luckiness of μ, is proven to be also separating for the system $\{P,Q\}$ [BLPR15, Proposition 9].

Specialized to the system of critical points, Algorithm 5 follows the same approach except for the way the reductions modulo μ of the $R(t,a)$ are computed.

Proposition 24. *Given H in $\mathbb{Z}[x,y]$ of degree d and bitsize τ, Algorithm 5 computes, with a worst-case bit complexity $\widetilde{O}_B(d^6+d^5\tau)$, an integer a in $[0, 2d^4-2d]$ such that the linear form $x+ay$ is separating for the system $\{H, \frac{\partial H}{\partial y}\}$ of critical points of the curve $H=0$.*

Proof. We first prove the correctness of Algorithm 5 which essentially follows from [BLPR15, Algorithm 4]. The only relevant difference for the correctness is the way to compute $R_\mu(t,s)$. In [BLPR15], $R_\mu(t,s)$ is computed by computing the resultant $R(t,s)$ of $H(t-sy,y)$ and $\frac{\partial H}{\partial y}(t-sy,y)$, and reducing it modulo μ. Here, we first reduce the polynomials modulo μ before computing the resultant. This yields the same result since μ is known to be lucky for the system $\{H, \frac{\partial H}{\partial y}\}$, thus it does not divide the leading terms $L_H(s)$ and $L_{\frac{\partial H}{\partial y}}(s)$. This proves the correctness of Algorithm 5. Furthermore, the correctness of [BLPR15, Algorithm 4] implies that the value a output by our algorithm is less than $2d^4 - 2d$.[4]

[4] [BLPR15, Theorem 19] is stated with $a < 2d^4$ but its proof establishes $a \leqslant \binom{d^2}{2} + 2(d^2+d)$ which is less than $2d^4 - 2d$ for $d > 1$. This refined bound will be convenient for yielding the simple bound of $2d^4$ when shearing back the separating form in Theorem 26.

We now prove the complexity of our algorithm. First, observe that, as argued in the proof of Proposition 21, $\frac{\partial H}{\partial y}$ and $(\frac{\partial H}{\partial y})^2$ have degrees at most $2d$, bitsizes $O(\tau + \log d)$, and that they can be computed in complexity $\widetilde{O}_B(d^2\tau)$. Furthermore, in Lines 1–3, the input of Algorithms 2, 3, and 4 satisfy the requirements of these algorithms, since H is squarefree with $\mathrm{Lc}_y(H)$ in $\mathbb{Z}$. The bit complexity of Lines 1–3 is thus in $\widetilde{O}_B(d^6 + d^5\tau)$ by Propositions 16, 21, 23.

It is straightforward that, in Line 4, the sheared polynomials $H(t - sy, y)$ and $\frac{\partial H}{\partial y}(t - sy, y)$ can be computed in bit complexity $\widetilde{O}_B(d^4 + d^3\tau)$ and that their bitsizes are in $\widetilde{O}(d + \tau)$ (see e.g. [BLPR15, Lemma 7]). In Lines 5 and 6, the polynomials to be reduced modulo μ, in one or three variables, have degree at most d and bitsize $\widetilde{O}(d + \tau)$. The reduction of each of their $O(d^3)$ coefficients modulo μ can be done in a bit complexity that is softly linear in the maximum bitsizes [vzGG13, Theorem 9.8], that is in a total bit complexity of $\widetilde{O}_B(d^4 + d^3\tau)$. Then, computing in Line 5 the product of $\phi_\mu(L_H(s))$ and $\phi_\mu(L_{\frac{\partial H}{\partial y}(t-sy,y)}(s))$ amounts to computing $O(d^2)$ arithmetic operations in $\mathbb{F}_\mu$.

The resultant in Line 6 can be computed in $O(d^5)$ arithmetic operations in $\mathbb{F}_\mu$ (see Lemma 3). In Line 7, $R_\mu(t, s)$ is a polynomial of degree $O(d^2)$ in t with coefficients in s of degree $O(d^2)$. The arithmetic complexity, in $\mathbb{F}_\mu$, of the evaluation of one such coefficient at $s = a$ is linear in its degree (using for instance Horner's scheme) but, using multipoint evaluation, the arithmetic complexity of the evaluation of one such coefficient at $O(d^2)$ values is in $\widetilde{O}(d^2)$ [vzGG13, Corollary 10.8]. It follows that the evaluation of all the $O(d^2)$ coefficients of $R_\mu(t, s)$ at d^2 values of a can be done with $\widetilde{O}(d^4)$ arithmetic operations in $\mathbb{F}_\mu$. The overall arithmetic complexity of Line 7 is thus $\widetilde{O}(d^6)$. In Line 10, since $R_\mu(t, a)$ has degree $O(d^2)$, its squarefree part can be computed with $\widetilde{O}(d^2)$ arithmetic operations in $\mathbb{F}_\mu$ (see Lemma 4) and, in Line 12, each evaluation of $\Upsilon(a)$ can be done in $O(d)$ arithmetic operations since Υ has degree $O(d)$. Furthermore, since the algorithm stops with $a < 2d^4$, the arithmetic complexity of the whole loop is in $\widetilde{O}(d^6)$.

We have shown that Lines 6 to 12 perform $\widetilde{O}(d^6)$ arithmetic operations in $\mathbb{F}_\mu$. Since μ has bitsize $O(\log(d\tau))$, the bit complexity of these lines is in $O_B(d^6\mathrm{polylog}(d\tau))$, which concludes the proof. □

Remark 25. *From a worst-case complexity point of view, the knowledge of the number N of (distinct) complex critical points of the input curve in Algorithm 5 is not mandatory since one could instead compute the number of solutions N_a of $R_\mu(t, a)$ for all integers a smaller than $2d^4$ and output a value of a that maximizes N_a. However, knowing N, the algorithm can stop as soon as a value of a is found such that $N_a = N$, which improves the expected complexity of the algorithm in a Las Vegas setting, as discussed in Section 5.5.*

5.4 Separating linear form of a system

Propositions 20 and 24 directly yield the following theorem where the separating form is obtained by shearing back the separating form output by Algorithm 5.

Theorem 26. *Let P, Q in $\mathbb{Z}[x, y]$ be of total degree at most d and maximum bitsize τ. A separating linear form $x + by$ for $\{P, Q\}$ with an integer b in $[0, 2d^4]$ can be computed using $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case. Furthermore, b is such that the leading coefficients of $P(t - by, y)$ and $Q(t - by, y)$ in y are in $\mathbb{Z}$.*

Proof. The first statement of the theorem follows directly from Propositions 20 and 24 where the integer b is the sum of the integers α and a defined in these propositions. We prove below the second statement. The integer a computed by Algorithm 5 is such that $\Upsilon(a) \neq 0$ and thus

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

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

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

$L_H(a) \neq 0$. Since $L_H(a) \in \mathbb{Z}$ is non-zero, it is the leading coefficient in y of the sheared polynomial $H(t-ay,y)$. H is the product of the squarefree parts of the sheared polynomials $\widetilde{P}$ and $\widetilde{Q}$ where $\widetilde{P}(t,y) = P(t-\alpha y, y)$ and similarly for $\widetilde{Q}$. Hence, the leading coefficient in y of the sheared polynomial $\widetilde{P}(t-ay,y) = P(t-ay-\alpha y, y) = P(t-by,y)$ divides $L_H(a)$, which is an integer. Similarly for $\widetilde{Q}$. □

5.5 Las Vegas algorithm

We show here that the algorithm presented above for computing a separating linear form can easily be transformed into an efficient Las Vegas algorithm.

Theorem 27. *Let P, Q in $\mathbb{Z}[x,y]$ be of total degree at most d and maximum bitsize τ. A separating linear form $x+by$ for $\{P,Q\}$ with an integer b in $[0,2d^4]$ can be computed with $\widetilde{O}_B(d^5+d^4\tau)$ bit operations on average. Furthermore, b is such that the leading coefficients of $P(t-by,y)$ and $Q(t-by,y)$ in y are in $\mathbb{Z}$.*

Our Las Vegas algorithm is obtained from our deterministic version by only modifying Algorithm 5 into a randomized version, Algorithm 5'. The main difference between these two versions is that, in Algorithm 5', we choose randomly a candidate separating linear form $x+ay$ and a candidate lucky prime μ for $\{H, \frac{\partial H}{\partial y}\}$ (Definition 18) until the degree N_a of the squarefree part of $R_\mu(t,a)$ is equal to the known number of solutions N. If a and μ are chosen randomly in sufficiently large sets, the probability that $x+ay$ is separating and that μ is lucky is larger than a positive constant, which implies that the expected number of such choices is a constant.

This modification yields a major simplification: since we do not compute anymore a lucky prime in a deterministic way, we do not need Algorithm 4 (Lucky prime), which again implies that Algorithm 2 (Luckiness certificate) is not needed. Furthermore, note that, in Algorithm 5', we do not need anymore to use multipoint evaluation for evaluating $R_\mu(t,s)$ at a since the expected number of choices of a is a constant. Note finally that we choose the candidate lucky prime μ

[5]We use the convention that the degree of the zero polynomial is $+\infty$ because we want N_a to be the number of distinct roots of $R_{\mu,a}(t)$. Note that in Algorithm 5, this issue was not relevant because μ was known to be lucky for the zero-dimensional system $\{H, \frac{\partial H}{\partial y}\}$, implying by Definition 18 that the system $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ was zero dimensional and thus that $R_\mu(t,a) \not\equiv 0$.

in increasingly larger sets. The reason is that, if we wanted to compute a unique set for which a random prime would be lucky with probability at least some constant, we would need an explicit upper bound (without $\widetilde{O}$ notation) on the number of unlucky primes and such a computation is highly unappealing.

We now prove the correctness and complexity of Algorithm 5' in the two following lemmas and in Proposition 31, which, together with Proposition 20, yield Theorem 27 similarly as for Theorem 26.

Lemma 28 (Correctness of Algorithm 5')**.** *Algorithm 5' terminates if and only if the values of the random variables a and μ are such that $\Upsilon_\mu(a) \neq 0$, μ is lucky for $\{H, \frac{\partial H}{\partial y}\}$ and $x + ay$ is separating for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$, which implies that $x + ay$ is also separating for $\{H, \frac{\partial H}{\partial y}\}$.*

Proof. The proof relies on Lemma 10 and Propositions 9 and 12 in [BLPR15] which, together, require the hypotheses that $\Upsilon_\mu(a) \neq 0$, $a < \mu$, $2d^4 < \mu$, and $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is zero dimensional. We first prove that these hypotheses are satisfied when either side of the if-and-only-if claim holds in the statement of the lemma.

First, $\Upsilon_\mu(a) \neq 0$ follows from Line 10 if Algorithm 5' terminates and it appears in the right hand side of the if-and-only-if claim. Second, $a \leqslant 2d^4 < \mu$ by definition of a and μ (Line 6). Finally, if Algorithm 5' terminates, $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is zero dimensional because, otherwise, $R_{\mu,a}(t) \equiv 0$, thus $N_a = +\infty$ cannot be equal to N in Line 10 (since $\{H, \frac{\partial H}{\partial y}\}$ is zero dimensional). On the other hand, if μ is lucky for $\{H, \frac{\partial H}{\partial y}\}$ then $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is zero dimensional since it has the same number of solution as $\{H, \frac{\partial H}{\partial y}\}$.

We can now apply Lemma 10 and Proposition 12 in [BLPR15], which state

$$\deg_t(\text{squarefree part}(R_\mu(t, a))) \leqslant \#V(I_\mu) \leqslant \#V(I)$$

where $R_\mu(t, s)$ refers to the resultant with respect to y of $\phi_\mu(H)(t - sy, y)$ and $\phi_\mu(\frac{\partial H}{\partial y})(t - sy, y)$, and $\#V(I)$ and $\#V(I_\mu)$ are the number of distinct solutions of the systems $\{H, \frac{\partial H}{\partial y}\}$ and $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$, respectively.

Assume for now that $R_\mu(t, a) = R_{\mu,a}(t)$ (as defined in Line 8). This implies that $\deg_t(\text{squarefree part}(R_\mu(t, a))) = N_a$ (Line 9). Since $N = \#V(I)$ by definition (Line 1), the algorithm terminates if and only if a and μ are such that $\Upsilon_\mu(a) \neq 0$ and $N_a = N$ (Line 10), which is equivalent to $\Upsilon_\mu(a) \neq 0$ and

$$\deg_t(\text{squarefree part}(R_\mu(t, a))) = \#V(I_\mu) = \#V(I).$$

The first equality holds if and only if $x + ay$ is separating for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ [BLPR15, Lemma 10] and the second equality holds if and only if μ is lucky for the system $\{H, \frac{\partial H}{\partial y}\}$ (Definition 18). This proves the if-and-only-if claim of the lemma (assuming that $R_\mu(t, a) = R_{\mu,a}(t)$). Furthermore, when both equalities hold, $x + ay$ is also separating for the system $\{H, \frac{\partial H}{\partial y}\}$ by Proposition 9 in [BLPR15].

It remains to show that $R_\mu(t, a) = R_{\mu,a}(t)$, that is that the resultant commutes with the evaluation at $s = a$ in the following way:

$$\text{Res}(\phi_\mu(H)(t - sy, y), \phi_\mu(\frac{\partial H}{\partial y})(t - sy, y))\big|_{s=a} = \text{Res}(\phi_\mu(H(t - ay, y)), \phi_\mu(\frac{\partial H}{\partial y}(t - ay, y))).$$

This equality holds if the polynomials in the left-hand side resultant are such that their leading coefficients (in y) $L_{\phi_\mu(H)}(s)$ and $L_{\phi_\mu(\frac{\partial H}{\partial y})}(s)$ do not vanish at $s = a$. This follows from the hypothesis that $\Upsilon_\mu(a) \neq 0$. Indeed, $\Upsilon_\mu(a) \neq 0$ implies $\phi_\mu(L_H(a)) \neq 0$. Then, $\phi_\mu(L_H(s)) \not\equiv 0$ implies $\phi_\mu(L_H(s)) = L_{\phi_\mu(H)}(s)$ and thus $L_{\phi_\mu(H)}(a) \neq 0$. Similarly for $\frac{\partial H}{\partial y}$, $L_{\phi_\mu(\frac{\partial H}{\partial y})}(a) \neq 0$, which concludes the proof. □

Lemma 29. *The expected number of iterations of the loop in Algorithm 5' is in $O(\log(d\tau))$. More precisely, after $O(\log(d\tau))$ iterations, the probability that the algorithm terminates is at least $1/8$ at every iteration.*

Proof. The number of unlucky primes for $\{H, \frac{\partial H}{\partial y}\}$ is in $\widetilde{O}(d^4 + d^3\tau)$ [BLPR15, Proposition 13]. Let $K(d, \tau)$ in $\widetilde{O}(d^4 + d^3\tau)$ be an upper bound on the number of unlucky primes, which we denote for simplicity by K. If the algorithm terminates with a value of M such that $\frac{M/2}{2\ln M/2} \leqslant 2K$, the number of loop iterations is in $O(\log(d\tau))$. Indeed, the number of iterations is less than $\log M$ which is in $O(\log K)$ since $\sqrt{M/2} < \frac{M/2}{\ln M/2} \leqslant 4K$. It is thus sufficient to prove that, for any iteration such that $\frac{M/2}{2\ln M/2} > 2K$, the probability that $\Upsilon_\mu(a) \neq 0$ and $N_a = N$ (Line 10) is at least $1/8$. Note that this implies that the expected number of such iterations is at most 8 and thus that the expected number of all iterations in the loop is in $O(\log(d\tau))$.

We can thus assume that, in Line 6, μ is chosen uniformly at random in a set of primes of cardinality at least $2K$. Indeed, μ is chosen in $(2d^4, M) \supseteq (M/2, M)$ and the number of primes in $(M/2, M)$ is at least $\frac{M/2}{2\ln M/2}$ [vzGG13, Theorem 18.7 (see also Exercise 18.18)].

By Lemma 28, the algorithm terminates if and only if a and μ are such that $\Upsilon_\mu(a) \neq 0$, μ is lucky for $\{H, \frac{\partial H}{\partial y}\}$ (Definition 18) and $x + ay$ is separating for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$. Let P denote the probability that these three events simultaneously occur. We have

$$\begin{aligned}
P &= \Pr(\mu \text{ is lucky for } \{H, \tfrac{\partial H}{\partial y}\} \text{ and } x + ay \text{ is separating for } \{\phi_\mu(H), \phi_\mu(\tfrac{\partial H}{\partial y})\}) \\
&\quad \cdot \Pr(\Upsilon_\mu(a) \neq 0 \mid \mu \text{ is lucky for } \{H, \tfrac{\partial H}{\partial y}\} \text{ and } x + ay \text{ is separating for } \{\phi_\mu(H), \phi_\mu(\tfrac{\partial H}{\partial y})\}) \\
&= \Pr(\mu \text{ is lucky for } \{H, \tfrac{\partial H}{\partial y}\}) \\
&\quad \cdot \Pr(x + ay \text{ is separating for } \{\phi_\mu(H), \phi_\mu(\tfrac{\partial H}{\partial y})\} \mid \mu \text{ is lucky for } \{H, \tfrac{\partial H}{\partial y}\}) \\
&\quad \cdot \Pr(\Upsilon_\mu(a) \neq 0 \mid \mu \text{ is lucky for } \{H, \tfrac{\partial H}{\partial y}\} \text{ and } x + ay \text{ is separating for } \{\phi_\mu(H), \phi_\mu(\tfrac{\partial H}{\partial y})\}).
\end{aligned}$$

The probability that μ is lucky for $\{H, \frac{\partial H}{\partial y}\}$ is at least $1/2$ since, as argued above, μ is chosen uniformly at random in a set of primes of cardinality at least $2K$ and there are at most K unlucky primes.

The conditional probability that $x + ay$ is separating for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is also at least $1/2$. Indeed, we prove that the conditional probability that $x + ay$ is *not* separating for that system is at most $1/2$. For any choice of a lucky μ, $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is zero dimensional since it has the same number of distinct solutions as $\{H, \frac{\partial H}{\partial y}\}$, which is at most d^2 by Bézout's bound. Thus, for any choice of a lucky μ, there are at most $\binom{d^2}{2} < d^4 - d$ directions in which two distinct solutions of $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ are aligned, that is, at most $d^4 - d$ values of a for which $x + ay$ is not separating for that system. Since a is chosen uniformly at random in a set of cardinality $2d^4 - 2d + 1$, the conditional probability that $x + ay$ is not separating for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is thus at most $d^4 - d$ times the number of choices of a lucky μ over the number of choices of couples of a and a lucky μ. In other words, it is at most $d^4 - d$ over $2d^4 - 2d + 1$, which is less than $1/2$, and thus proves the claim.

Finally, we show that the conditional probability that $\Upsilon_\mu(a) \neq 0$ is also at least $1/2$. Given that μ is lucky, $\Upsilon_\mu(s) \not\equiv 0$, by Definition 18. Thus, for any given lucky μ, $\Upsilon_\mu(s)$ has degree at most $2d$ and it vanishes for at most $2d$ values of a. The conditional probability that $\Upsilon_\mu(a) = 0$ is thus at most $2d$ times the number of choices of a lucky μ over the number of choices of couples of a lucky μ and a value a such that $x + ay$ is separating. This probability is thus equal to $2d$ over the number of choices of such values a. The number of such choices for a is at least d^4 since a is considered in

$[0, 2d^4 - 2d]$ and there are at most $\binom{d^2}{2} < d^4 - 2d$ choices for which $x + ay$ is not separating. Hence the conditional probability that $\Upsilon_\mu(a) = 0$ is at most $2d/d^4$, which is less that $1/2$ for $d \geqslant 2$. This proves the claim that the conditional probability that $\Upsilon_\mu(a) \neq 0$ is at least $1/2$ and concludes the proof. □

The next technical lemma is instrumental for the complexity analyses of Propositions 31 and 48.

Lemma 30. *If a while-loop is such that (i) the expected bit complexity of the i-th iteration is $\widetilde{O}_B(A\, i^k)$ where A is a polynomial in the input parameter sizes and (ii) the probability that the loop ends at the i-th iteration, given that it has not stopped before, is at least a constant $c > 0$, then the expected bit complexity of the entire loop is $\widetilde{O}_B(A)$.*

Proof. Let $x_j > c$ be the probability that the loop stops at the j-th iteration given that it has not yet stopped before. The probability that the loop stops at the i-th iteration is $x_i \prod_{j=1}^{i-1}(1 - x_j) \leqslant (1 - c)^{i-1}$. On the other hand, the total expected bit complexity of all the iterations until the i-th is $\widetilde{O}_B(A\, i^{k+1})$. Hence the total expected bit complexity of the entire loop is

$$\sum_{i=1}^{\infty} \widetilde{O}_B(A\, i^{k+1}(1 - c)^{i-1}) = \widetilde{O}_B(A \sum_{i=1}^{\infty} i^{k+1}(1 - c)^{i-1}).$$

The series $\sum_{i=0}^{\infty} W(i)\lambda^i$ is convergent for any polynomial W and $0 < \lambda < 1$. Indeed, $|W(i)| < \delta^i$ for any $1 < \delta < \frac{1}{\lambda}$ and i sufficiently large, which implies $|W(i)\lambda^i| < (\delta\lambda)^i$ with $0 < \delta\lambda < 1$. Therefore, the expected bit complexity of the entire loop is in $\widetilde{O}_B(A)$. □

Proposition 31. *Given H in $\mathbb{Z}[x, y]$ of degree d and bitsize τ, Algorithm 5' computes, with an expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$, an integer a in $[0, 2d^4 - 2d]$ such that the linear form $x + ay$ is separating for the critical points of H.*

Proof. By Proposition 21, Line 1 has expected complexity $\widetilde{O}_B(d^5 + d^4\tau)$. In Line 2, similarly as in Line 4 of Algorithm 5, $H(t - sy, y)$ and $\frac{\partial H}{\partial y}(t - sy, y)$ have coefficients of bitsize $\widetilde{O}(d + \tau)$ and they can be computed with $\widetilde{O}_B(d^4 + d^3\tau)$ bit operations (see the proof of Proposition 24). Still in Line 2, $\Upsilon(s)$ can be computed with $O(d^2)$ arithmetic operations on integers of bitsize $O(\tau + \log d)$, and thus with $\widetilde{O}_B(d^2\tau)$ bit operations. The worst-case bit complexity of Lines 1 and 2 is thus in $\widetilde{O}_B(d^5 + d^4\tau)$.

Consider now one iteration of the loop in Algorithm 5' and let I_M denote the interval $(2d^4, M)$. In Line 6, we can compute a prime μ by choosing uniformly at random an integer in I_M and testing whether it is prime until a prime is found. Finding a random integer smaller than M amounts to computing a sequence of $\log M$ random bits, which we assume can be done in $O_B(\log M)$ bit operations. A random integer smaller than M is larger than $2d^4$ with probability at least $1/2$, thus a random integer in I_M can be computed in $O_B(\log M)$ bit operations. The number of primes in $(M/2, M) \subseteq I_M$ is at least $\frac{M/2}{2 \ln M/2}$ [vzGG13, Theorem 18.7 (see also Exercise 18.18)]. The probability that a randomly chosen integer in I_M is prime is thus at least $\frac{1}{4 \ln M/2}$ and a prime is thus found after at most $4 \ln M/2$ trials on average. Testing whether an integer in I_M is prime can be done with a polynomial bit complexity in the bitsize of M, $\widetilde{O}_B(\log^{7.5} M)$ [AKS04]. The expected bit complexity of computing a prime in Line 6 is thus in $\widetilde{O}_B(\log^{8.5} M)$. Furthermore, since a random integer a in $[0, 2d^4]$ can be computed in $\widetilde{O}_B(\log d)$ bit operations, the expected bit complexity of one iteration of Line 6 is in $\widetilde{O}_B(\log^{8.5} M)$.

In Line 7, $O(d)$ coefficients of bitsize $O(\tau + \log d)$ are reduced modulo μ. Each reduction can be done in a bit complexity that is softly linear in the maximum bitsizes [vzGG13, Theorem 9.8],

that is in a total bit complexity of $\widetilde{O}_B(d(\tau + \log d + \log M))$. Evaluating $\Upsilon(s)$ at a can then be done with $O(d)$ arithmetic operations in $\mathbb{F}_\mu$ and thus with $\widetilde{O}_B(d \log M)$ bit operations. The total bit complexity of one iteration of Line 7 is thus in $\widetilde{O}_B(d(\tau + \log M))$.

In Line 8, first notice that $\phi_\mu(H(t-ay, y)) = \phi_\mu(H(t-sy, y))\big|_{s=a}$. Similarly as above, the $O(d^3)$ coefficients of $H(t-sy, y)$ of bitsize $\widetilde{O}(d+\tau)$ can be reduced modulo μ with $\widetilde{O}_B(d^3(d+\tau+\log M))$ bit operations in total. The evaluation at $s = a$ in $\mathbb{F}_\mu$ then amounts to evaluating $O(d^2)$ univariate polynomials in s of degree $O(d)$. Similarly as above, this can be done with $O(d^3)$ arithmetic operations $\mathbb{F}_\mu$ and thus with $\widetilde{O}_B(d^3 \log M)$ bit operations. Thus, $\phi_\mu(H(t-ay, y))$ and similarly $\phi_\mu(\frac{\partial H}{\partial y}(t-ay, y))$ can be computed with $\widetilde{O}_B(d^3(d+\tau+\log M))$ bit operations in the worst case. By Lemma 4, their resultant $R_{\mu,a}(t)$ has degree $O(d^2)$, and it can be computed with $\widetilde{O}(d^3)$ arithmetic operations in $\mathbb{F}_\mu$ and thus with $\widetilde{O}_B(d^3 \log M)$ bit operations. The bit complexity of one iteration of Line 8 is thus in $\widetilde{O}_B(d^3(d + \tau + \log M))$ in the worst case.

In Line 9, the squarefree part of $R_{\mu,a}(t)$, and thus its degree, can be computed with $\widetilde{O}(d^2)$ arithmetic operations in $\mathbb{F}_\mu$ (by Lemma 4) and thus with $\widetilde{O}_B(d^2 \log M)$ bit operations in the worst case.

Hence, the expected bit complexity of one iteration of the loop is in $\widetilde{O}_B(d^3(d + \tau + \log M) + \log^{8.5} M)$, which is also in $\widetilde{O}_B(d^3(d + \tau + \log^9 M))$. More precisely, at the end of the j-th iteration of the loop, $M = 2^{j+1}d^4$, thus the expected bit complexity of the j-th iteration of the loop is in $\widetilde{O}_B(d^4 + d^3\tau + d^3 j^9)$. The expected bit complexity of the entire loop is thus $\widetilde{O}_B(d^4 + d^3\tau)$, by Lemmas 29 and 30. Summing up with the complexity of Lines 1 and 2, we obtain that the expected bit complexity of the algorithm is in $\widetilde{O}_B(d^5 + d^4\tau)$. □

6 RUR decomposition

In this section, we consider that a separating form for the bivariate system $\{P, Q\}$ has been computed as shown in Section 5 and we focus on the computation of Rational Univariate Representations of the solutions. We present a new algorithm of worst-case bit complexity $\widetilde{O}_B(d^6 + d^5\tau)$ for computing a RUR decomposition of $\{P, Q\}$, that is a sequence of RURs that encodes the solutions of $\{P, Q\}$ (see Definition 34 and Theorem 43). This algorithm is multi-modular and it relies on both the triangular decomposition and the luckiness certificate of Section 4. We also present a Las Vegas version of this algorithm, of expected bit complexity $\widetilde{O}_B(d^5 + d^4\tau)$ (Theorem 44), which only computes some coefficients of the above triangular decomposition and avoids computing the luckiness certificate.

In Section 6.1, we first recall the definitions and main properties of RURs. We present our deterministic algorithm and its complexity analysis in Section 6.2 and its Las Vegas version in Section 6.3.

6.1 RUR definition and properties

Definition 32 ([Rou99, Definition 3.3]). *Let $I \subset \mathbb{Q}[x, y]$ be a zero-dimensional ideal, $V(I) = \{\sigma \in \mathbb{C}^2, v(\sigma) = 0, \forall v \in I\}$ its associated variety, and let $(x, y) \mapsto x + ay$ be a linear form with a in $\mathbb{Q}$. The* RUR*-candidate of I associated to $x + ay$ (or simply, to a), denoted* $\mathrm{RUR}_{I,a}$*, is the following*

set of four univariate polynomials in $\mathbb{C}[t]$

$$\begin{aligned} f_{I,a}(t) &= \prod_{\sigma \in V(I)} (t - x(\sigma) - ay(\sigma))^{\mu_I(\sigma)} \\ f_{I,a,v}(t) &= \sum_{\sigma \in V(I)} \mu_I(\sigma) v(\sigma) \prod_{\varsigma \in V(I), \varsigma \neq \sigma} (t - x(\varsigma) - ay(\varsigma)), \quad \text{for } v \in \{1, x, y\} \end{aligned} \tag{2}$$

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

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

Proposition 33 ([Rou99, Theorem 3.1])**.** *If* $I \subset \mathbb{Q}[x, y]$ *is a zero-dimensional ideal and* a *in* $\mathbb{Q}$, *the four polynomials of the* RUR*-candidate* $\mathrm{RUR}_{I,a}$ *have rational coefficients. Furthermore, if* $x + ay$ *separates* $V(I)$, *the following mapping between* $V(I)$ *and* $V(f_{I,a}) = \{\gamma \in \mathbb{C}, f_{I,a}(\gamma) = 0\}$

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

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

Next, we define a RUR decomposition of an ideal.

Definition 34. *Let* $I \subset \mathbb{Q}[x, y]$ *be a zero-dimensional ideal,* $V(I) = \{\sigma \in \mathbb{C}^2, v(\sigma) = 0, \forall v \in I\}$ *its associated variety, and let* $(x, y) \mapsto x + ay$ *be a linear form with* a *in* $\mathbb{Q}$. *A RUR-candidate decomposition of* I *is a sequence of RUR-candidates, associated to* $x + ay$, *of ideals* $I_i \supseteq I$, $i \in \mathcal{I}$ *such that* $V(I)$ *is the disjoint union of the varieties* $V(I_i)$, $i \in \mathcal{I}$. *If* $x + ay$ *separates* $V(I_i)$ *for all* $i \in \mathcal{I}$, *the RUR-candidate decomposition is a RUR decomposition of* I.

The following proposition recalls an upper bound on the bitsize of a RUR of an ideal containing two coprime polynomials P and Q, that is a RUR parameterizing a subset of the solutions of the system $\{P, Q\}$. This bound applies to the RURs of our RUR decomposition and is used in Algorithm 6.

Proposition 35 ([BLPR15, Proposition 28])**.** *Let* P *and* Q *in* $\mathbb{Z}[x, y]$ *be two coprime polynomials of total degree at most* d *and maximum bitsize* τ, *let* a *be a rational of bitsize* τ_a, *and let* J *be any ideal of* $\mathbb{Z}[x, y]$ *containing* P *and* Q. *The polynomials of the* RUR*-candidate of* J *associated to* a *have degree at most* d^2 *and bitsize in* $\widetilde{O}(d^2\tau_a + d\tau)$.

Note that according to Theorem 26, a separating form $x + ay$ can be computed with an integer a of bitsize $O(\log d)$ and the bound in Proposition 35 becomes $\widetilde{O}(d^2 + d\tau)$. In addition, even if Proposition 35 only states an asymptotic upper bound, an explicit upper bound $C(d^2 + d\tau)\log^k(d\tau)$ with $C, k \in \mathbb{Z}$ can be obtained from straightforward, although unappealing, computations following the proof of that proposition. Indeed, this proof is based on Hadamard's inequality and Mignotte's lemma, which both state explicit bounds.

Proposition 35 also yields the following bound on the total bitsize of any RUR decomposition.

Algorithm 6 RUR decomposition

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

1: Compute a separating form $x+ay$ for $\{P, Q\}$ with $a \in \mathbb{Z}$ of bitsize $O(\log d)$ such that the leading coefficients of $P(t - ay, y)$ and $Q(t - ay, y)$ with respect to y are coprime (see Theorem 26).
2: Compute $\widetilde{P}(t, y) = P(t - ay, y)$ and $\widetilde{Q}(t, y) = Q(t - ay, y)$, and let $\tilde{d}$ and $\tilde{\tau}$ be their maximum degree and bitsize.
3: Compute $\{T_i\}_{i\in\mathcal{I}}$ = Algorithm 1$(\widetilde{P}, \widetilde{Q})$.
Recall that $T_i = \{A_i(t), B_i(t, y)\}$ with $B_i(t, y) = \operatorname{sres}_i(\widetilde{P}, \widetilde{Q})(t)\, y^i + \operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})(t)\, y^{i-1} + \cdots$
Let $\widehat{T}_i = \langle A_i, i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q})\, y + \operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})\rangle$ be the radical ideal of T_i (see Lemma 37).
4: Let $K = \lceil C(\tilde{d}^2 + \tilde{d}\tilde{\tau}) \log^k(\tilde{d}\tilde{\tau})\rceil$ be an integer that bounds from above the bitsize of the coefficients of the RURs of the systems $\widehat{T}_i$ (see Proposition 35 and subsequent discussion) and let Π = Algorithm 2$(\widetilde{P}, \widetilde{Q})$. Compute the set $\mathcal{L}$ of the $2K$ first prime numbers that are larger than $\tilde{d}$ and that do not divide Π. Let $\Pi_{\mathcal{L}}$ be the product of all primes in $\mathcal{L}$.
5: **for all** i in $\mathcal{I}$ **do**
6: **for all** μ in $\mathcal{L}$ **do**
7: Compute $\phi_\mu(\widehat{T}_i)$ by reducing modulo μ the polynomials A_i, $\operatorname{sres}_i(\widetilde{P}, \widetilde{Q})$ and $\operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$.
8: Compute RUR_i^μ the RUR in $\mathbb{F}_\mu$ of $\phi_\mu(\widehat{T}_i)$ associated to the separating form $(t, y) \mapsto t$ (see Lemma 38).
9: Lift $\{\mathrm{RUR}_i^\mu\}_{\mu\in\mathcal{L}}$ to $\mathrm{RUR}_i^{\Pi_{\mathcal{L}}}$ in $\mathbb{Z}/\Pi_{\mathcal{L}}\mathbb{Z}$ using the Chinese Remainder Algorithm.
10: Compute $\mathrm{RUR}_i^{\mathbb{Q}}$, the RUR in $\mathbb{Q}$ of $\widehat{T}_i$ associated to the separating form $(t, y) \mapsto t$, with a rational reconstruction from $\mathrm{RUR}_i^{\Pi_{\mathcal{L}}}$ (see the proof of Proposition 41).
11: **return** the image of $\mathrm{RUR}_i^{\mathbb{Q}}$, $i \in \mathcal{I}$, through the reverse shearing from (t, y) to (x, y) (see Lemma 40).

Corollary 36. *Let P and Q in $\mathbb{Z}[x, y]$ be two coprime polynomials of total degree at most d and maximum bitsize τ, and let a be a rational of bitsize τ_a. The sum of the bitsizes of all coefficients of any RUR-candidate decomposition of $\langle P, Q\rangle$, associated to $x + ay$, is in $\widetilde{O}(d^4\tau_a + d^3\tau)$.*

Proof. By Definition 34, the ideals I_i defining a RUR-candidate decomposition of $\langle P, Q\rangle$ are such that (i) the solutions of I_i (counted with multiplicity) are included in those of $\langle P, Q\rangle$ (since $I_i \supseteq \langle P, Q\rangle$) and (ii) the sets $V(I_i)$ of (distinct) solutions of I_i are pairwise disjoint. Hence, the sum over all i of the number of solutions of I_i, counted with multiplicity, is at most d^2, the Bézout bound of $\{P, Q\}$. By Definition 32, the sum over all i of the degrees of the first polynomial of the RUR-candidate of I_i is thus also at most d^2. Moreover, still by Definition 32, the degree the first polynomial of a RUR-candidate bounds from above the degrees of the other polynomials of the RUR-candidate. Hence, the total number of coefficients of the RUR-candidate decomposition is $O(d^2)$. The result then follows from Proposition 35. □

6.2 Decomposition algorithm

Algorithm 6 computes a RUR decomposition of a zero-dimensional system $\{P, Q\}$, by first computing a separating form $x + ay$ as shown in Section 5 (Line 1). We then use this separating form to shear the system in generic position (Line 2) and compute the radical of a triangular decomposition of this system (Line 3). Then, using a multimodular approach, we compute RURs of each of the resulting radical systems (Lines 4–10) and return these RURs after a shear back

(Line 11).

The section is organized as follows. We first prove some preliminary lemmas that are instrumental for the proof of correctness of Algorithm 6. We show in Lemma 37 that the ideals we compute in Line 3 are the radicals of the ideals output by the triangular decomposition of Algorithm 1. We then determine in Lemma 38 formulas for the RURs of these radical ideals. These formulas are valid over the rationals but, for complexity issues, we use these formulas in a multimodular setting, in Lines 4 to 10. For this purpose, Lemma 39 states conditions on primes μ under which the reductions modulo μ of the RURs of these ideals are equal to the RURs of the reductions modulo μ of these ideals. We also show in Lemma 40 how to compute the image of the computed RURs through the reverse shearing of the one performed in Line 2. With these lemmas, we prove in Propositions 41 and 42 the correctness and complexity of Algorithm 6. Theorem 43 finally gathers these results.

Lemma 37. *The radical $\widehat{T}_i$ of ideal T_i is $\langle A_i, i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})\,y+\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})\rangle$ where A_i is squarefree and coprime with $\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$.*

Proof. Since $x+ay$ separates the solutions of $\{P,Q\}$, the system $\{\widetilde{P},\widetilde{Q}\}$ is in generic position in the sense that no two of its solutions are vertically aligned. The systems $T_i=\{A_i(t),B_i(t,y)\}$ of the triangular decomposition of $\{\widetilde{P},\widetilde{Q}\}$ are thus also in generic position (since the set of solutions of $\{\widetilde{P},\widetilde{Q}\}$ is the disjoint union of those of the T_i by Lemma 10). $B_i(t,y)=\mathrm{sres}_i(\widetilde{P},\widetilde{Q})\,y^i+\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})\,y^{i-1}+\cdots$ is of degree i in y and its leading coefficient $\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$ is coprime with A_i (by Lemma 10). Hence, for any α solution of $A_i(t)$, $B_i(\alpha,y)$ has a unique root, which is of multiplicity i. This multiple root is thus also root of the $(i-1)$-th derivative of $B_i(\alpha,y)$, which is $i!\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})(\alpha)\,y+(i-1)!\,\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})(\alpha)$. Hence, the distinct solutions of T_i are exactly those of $\widehat{T}_i$. Finally, $\widehat{T}_i$ is radical because $A_i(t)$ is univariate and squarefree (by Lemma 10) and $i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})\,y+\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})$ has degree one in the other variable, y. □

The next lemma states formulas for the RURs of the radical ideals $\widehat{T}_i$. We state this lemma in general form because we also use it for computing the RURs of $\phi_\mu(\widehat{T}_i)$.

Lemma 38. *Let $\mathbb{F}$ be a field, A,B_0,B_1 be three polynomials in $\mathbb{F}[t]$ and let $I=\langle A(t),B_1(t)\,y+B_0(t)\rangle$ be an ideal such that A is squarefree and coprime with B_1. The linear form $(t,y)\mapsto t$ is separating for that ideal and its associated RUR is given by*[6]

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

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

Proof. By Definition 32, the first polynomial of the RUR of I associated to the form $(t,y)\mapsto t$ is the unique monic polynomial that encodes the t-coordinates of the solutions of I, counted with multiplicity in I. Since A is squarefree and coprime with B_1, the solutions of I have multiplicity one and their t-coordinates are exactly the roots of A. Hence, since A is squarefree, the first polynomial of the RUR is $f_I=\frac{A}{\mathrm{Lc}(A)}$. It also follows from the definition of the RUR that if f_I is squarefree then $f_{I,1}=f_I'$.

By Proposition 33, $\frac{f_{I,t}}{f_{I,1}}(\alpha)=\alpha$ for any root α of $f_I=\frac{A}{\mathrm{Lc}(A)}$, since the separating form is $(t,y)\mapsto t$. Hence, $f_{I,t}(t)=t\,f_{I,1}(t)\mod A$. It follows that $f_{I,t}=t\,f_{I,1}\ \mathrm{rem}\ A$ since $f_{I,t}$ has the degree of A minus 1 by Definition 32.

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

We also have by Proposition 33 that $f_{I,1}\,y - f_{I,y}$ is in I. Multiplying it by B_1 and subtracting $f_{I,1}\,(B_1\,y + B_0)$, which is also in I, we obtain that $B_1\,f_{I,y} + f_{I,1}\,B_0$ is in I. This polynomial is univariate in t, hence it is equal to zero modulo A. On the other hand, since A and B_1 are coprime, by Bézout's identity, there exists a pair (U, V) of polynomials in $\mathbb{F}[t]$ such that $UB_1 + VA = 1$, and we have that $UB_1 = 1 \mod A$. It follows that $f_{I,y} + U\,f_{I,1}\,B_0 = 0 \mod A$ and thus that $f_{I,y} = -U\,f_{I,1}\,B_0 \text{ rem } A$ since $f_{I,y}$ has the degree of A minus 1 by Definition 32. □

Even if the bitsize of the RUR of $\widehat{T}_i$ is known to be in $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau}) = \widetilde{O}(d^2 + d\tau)$ (Proposition 35 and [BLPR15, Lemma 7]), the naive computation of these RURs using the above formulas over the rationals would suffer from large intermediate bitsizes.[7] To overcome this difficulty, we use in Algorithm 6 a classical multimodular technique, which consists in first computing the polynomials modulo a set of primes whose product is larger than the bitsize of the output coefficients, then lifting the result using the Chinese Remainder Algorithm and finally performing a rational reconstruction. However, to output a correct result, this technique requires that, for any selected prime μ, the formulas of Lemma 38 commute with the reduction modulo μ. We show in Lemma 39 how to satisfy this requirement using the luckiness certificate output by Algorithm 2. This lemma is instrumental for the proof of correctness of Algorithm 6.

Lemma 39. *Let $\mu > i$ be a prime that does not divide Π. The ideals $\widehat{T}_i$ and $\phi_\mu(\widehat{T}_i)$ satisfy the hypotheses of Lemma 38. In particular, the linear form $(t, y) \mapsto t$ is separating for both ideals. For this linear form, the RUR of $\phi_\mu(\widehat{T}_i)$ is equal to the reduction modulo μ of the RUR of $\widehat{T}_i$.*

Proof. By Lemma 37, the ideal $\widehat{T}_i = \langle A_i, i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})\,y + \mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})\rangle$ is such that A_i is squarefree and coprime with $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$. Lemma 38 thus applies and yields that the linear form $(t, y) \mapsto t$ is separating for ideal $\widehat{T}_i$ and that the associated RUR can be computed with the given formulas.

In the following, we assume that $\mu > i$ is a prime that does not divide Π. We first show that the ideal $\phi_\mu(\widehat{T}_i) = \langle \phi_\mu(A_i), i\,\phi_\mu(\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}))\,y + \phi_\mu(\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q}))\rangle$ also satisfies the hypotheses of Lemma 38. Recall the notation used in the proof of Lemma 14: let $(A_i^\mu(t), B_i^\mu(t, y))$ be the triangular systems computed by Algorithm 1 applied to $\phi_\mu(\widetilde{P}(t, y))$ and $\phi_\mu(\widetilde{Q}(t, y))$. Lemma 14 implies that μ is lucky for the triangular decomposition of $\{\widetilde{P}, \widetilde{Q}\}$, hence $\phi_\mu(A_i) = A_i^\mu$ and $\phi_\mu(B_i) = B_i^\mu$, the latter being equivalent to $\phi_\mu(\mathrm{Sres}_i(\widetilde{P}, \widetilde{Q})) = \mathrm{Sres}_i(\phi_\mu(\widetilde{P}), \phi_\mu(\widetilde{Q}))$. We thus have that $\gcd(\phi_\mu(A_i), i\,\phi_\mu(\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}))) = \gcd(A_i^\mu, i\,\mathrm{sres}_i(\phi_\mu(\widetilde{P}), \phi_\mu(\widetilde{Q})))$, which a non-zero constant in $\mathbb{F}_\mu$ by Lemma 10. In addition, Lemma 10 implies that $\phi_\mu(A_i) = A_i^\mu$ is squarefree. Lemma 38 thus applies to the ideal $\phi_\mu(\widehat{T}_i)$. Hence, the linear form $(t, y) \mapsto t$ is separating for $\phi_\mu(\widehat{T}_i)$ and the associated RUR can be computed with the formulas of Lemma 38.

Second, we prove that μ does not divide any denominator of the rational coefficients of the polynomials of the RUR of $\widehat{T}_i$ and thus that the images of these polynomials by ϕ_μ are well defined. By definition, A_i divides the resultant of $\widetilde{P}$ and $\widetilde{Q}$, which is equal to $\mathrm{sres}_0(\widetilde{P}, \widetilde{Q})$. It follows that μ does not divide $\mathrm{Lc}(A_i)$ because μ does not divide $\mathrm{Lc}(\mathrm{sres}_0(\widetilde{P}, \widetilde{Q}))$ by definition of Π. Thus μ does not divide any denominator of the coefficients of the RUR polynomials $f_{\widehat{T}_i} = \frac{A_i}{\mathrm{Lc}(A_i)}$ and $f_{\widehat{T}_i,1} = f'_{\widehat{T}_i}$. On the other hand, if F is a polynomial in $\mathbb{Q}[t]$ such that μ does not divide the denominators of its coefficients, then μ does not divide the denominators of the coefficients of $F \text{ rem } A_i$ (the denominator of a coefficient of the remainder is the product of $\mathrm{Lc}(A_i)$ and some denominators of coefficients of F). It follows that μ does not divide the denominators of the

[7] More precisely, the computation of the RURs using the formulas of Lemma 38 over the rationals would require $\widetilde{O}_B(d^8 + d^7\tau)$ bit operations for each triangular system and $\widetilde{O}_B(d^9 + d^8\tau)$ for all of them. This bit complexity corresponds roughly to the cost of multiplications and divisions involving the inverse of $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}) \text{ rem } A_i$, which is a polynomial of degree $O(d^2)$ and bitsize in $\widetilde{O}(d^4 + d^3\tau)$.

coefficients of $f_{\widehat{T}_i,t} = t f_{\widehat{T}_i,1} \mod A_i$. Similarly, to prove that μ does not divide the denominators of the coefficients of $f_{\widehat{T}_i,y} = -\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})\, U\, f_{\widehat{T}_i,1} \mod A_i$, it is sufficient to prove the μ does not divide the denominators of the coefficients of U, the inverse of $i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$ modulo A_i (since $\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})$ has integer coefficients). By definition of Π, μ does not divide $\mathrm{Lc}(\mathrm{sres}_i(\widetilde{P},\widetilde{Q}))$. In addition, we have shown that μ does not divide $\mathrm{Lc}(A_i)$, A_i and $i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$ are coprime by Lemma 37 and we have shown above that $\phi_\mu(A_i)$ and $\phi_\mu(i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q}))$ are also coprime. It follows that μ is lucky for $\gcd(A_i, i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q}))$ (Definition 6). Thus, by Lemma 7, μ does not divide $res = \mathrm{Res}_t(A_i, i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q}))$. By [BPR06, Prop. 8.38.a] and since A_i and $i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$ are coprime, there exist u, v in $\mathbb{Z}[t]$ such that, $\deg_t(u) < \deg_t(A)$ and $u\,i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q}) + vA_i = res$, which is equivalent to $\frac{u}{res} i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q}) + \frac{v}{res} A_i = 1$. By uniqueness of Bézout's coefficients in $\mathbb{Q}[t]$, U the inverse of $i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$ modulo A_i is equal to $\frac{u}{res}$ and μ does not divide any denominator of its coefficients.

It is now clear that the image by ϕ_μ of the RUR polynomials $f_{\widehat{T}_i}, f_{\widehat{T}_i,1}, f_{\widehat{T}_i,t}$ are those of the RUR of $\phi_\mu(\widehat{T}_i)$. For $f_{\widehat{T}_i,y} = -\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})\, U\, f_{\widehat{T}_i,1} \mod A_i$, since we have shown that $\phi_\mu(\mathrm{Sres}_i(\widetilde{P},\widetilde{Q})) = \mathrm{Sres}_i(\phi_\mu(\widetilde{P}),\phi_\mu(\widetilde{Q}))$, it is sufficient to show that $\phi_\mu(U)$ is the inverse of $\phi_\mu(i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q}))$ modulo $\phi_\mu(A_i)$. As shown above, $\phi_\mu(U)$ is well defined and the relation $\phi_\mu(\frac{u}{res})\phi_\mu(i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q})) + \phi_\mu(\frac{v}{res})\phi_\mu(A_i) = 1$ implies that it is the inverse of $\phi_\mu(i\,\mathrm{sres}_i(\widetilde{P},\widetilde{Q}))$ modulo $\phi_\mu(A_i)$. □

Lemma 40. *Let $\{f_I, f_{I,1}, f_{I,t}, f_{I,y}\}$ be the RUR[6] of an ideal I in $\mathbb{Q}[t,y]$ associated to the separating linear form $(t,y) \mapsto t$. Let J in $\mathbb{Q}[x,y]$ be the image of I through the mapping $(t,y) \mapsto (x = t - ay, y)$. The linear form $(x,y) \mapsto x + ay$ is separating for J and its associated RUR is given by*

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

Proof. By Definition 32, the RURs of I and J are defined by

$$f_I(t) = \prod_{\sigma \in V(I)} (t - t(\sigma))^{\mu_J(\sigma)} \qquad f_{I,v}(t) = \sum_{\sigma \in V(I)} \mu_I(\sigma) v(\sigma) \prod_{\varsigma \in V(I), \varsigma \neq \sigma} (t - t(\varsigma))$$
$$\text{for } v \in \{1, t, y\},$$
$$f_{J,a}(t) = \prod_{\sigma' \in V(J)} (t - x(\sigma') - a y(\sigma'))^{\mu_J(\sigma')} \qquad f_{J,a,v}(t) = \sum_{\sigma' \in V(J)} \mu_J(\sigma') v(\sigma') \prod_{\varsigma' \in V(J), \varsigma' \neq \sigma'} (t - x(\varsigma') - a y(\varsigma'))$$
$$\text{for } v \in \{1, x, y\}.$$

The change of coordinates $(t,y) \mapsto (x = t - ay, y)$ induces an affine transformation of the solutions that preserves their multiplicities, such that, for every solution σ of I, there exists a unique solution σ' of J with the same multiplicity and satisfying $x(\sigma') = t(\sigma) - a y(\sigma)$ and $y(\sigma') = y(\sigma)$. This directly implies all four equalities of the lemma. □

We finally prove the correctness and analyze the complexity of Algorithm 6 in the following two propositions.

Proposition 41 (Correctness of Algorithm 6)**.** *Algorithm 6 computes a RUR decomposition of $\{P, Q\}$ of total bitsize $\widetilde{O}(d^4 + d^3\tau)$.*

Proof. The correctness of Line 1 follows directly from Theorem 26. In particular, the sheared polynomials $\widetilde{P}$ and $\widetilde{Q}$ are coprime and their leading coefficients with respect to y are also coprime. In Line 3, Algorithm 1 can thus be applied to compute the ideals T_i. By Lemma 37, $\widehat{T}_i$ is the

radical ideal of T_i and, since both ideals have the same (distinct) solutions, the set of solutions of $\{\widetilde{P}, \widetilde{Q}\}$ is the disjoint union of the sets of solutions of all $\widehat{T}_i$, by Lemma 10.

In Line 4, the upper bound K can be computed according Proposition 35 and the subsequent discussion.

In Line 8, the RUR of $\phi_\mu(\widehat{T}_i)$ can be computed using the formulas of Lemma 38 (which applies by Lemma 39). Moreover, by Lemma 39, the RUR of $\phi_\mu(\widehat{T}_i)$ is the reduction modulo μ of the RUR of $\widehat{T}_i$. Thus, in Line 9, by the Chinese Remainder Theorem, $\mathrm{RUR}_i^{\Pi_\mathcal{L}}$ is the reduction modulo $\Pi_\mathcal{L}$ of the RUR of $\widehat{T}_i$.

Finally, in Line 10, we use a rational number reconstruction [vzGG13, Section 5.10] with parameter $2M$ with $M = 2^K$: for any coefficient c of $\mathrm{RUR}_i^{\Pi_\mathcal{L}}$ in $\mathbb{Z}/\Pi_\mathcal{L}\mathbb{Z}$ a rational number $\frac{r}{t}$ with r, t in $\mathbb{Z}$ is computed such that $\gcd(r,t) = 1$, $\gcd(t, \Pi_\mathcal{L}) = 1$, $rt^{-1} = c \mod \Pi_\mathcal{L}$, $|r| < 2M$, $0 < t \leqslant \frac{\Pi_\mathcal{L}}{2M}$. According to [vzGG13, Theorem 5.26 (iv)], there exists at most one solution such that $|r| < M$. On the other hand, $\mathrm{RUR}_i^{\mathbb{Q}}$, the RUR of $\widehat{T}_i$ computed in $\mathbb{Q}$ defines such a solution for each coefficient. Indeed, let $\tilde{r}/\tilde{t}$ be the coefficient in $\mathrm{RUR}_i^{\mathbb{Q}}$ corresponding to c, with $\gcd(\tilde{r}, \tilde{t}) = 1$ and $\tilde{t} > 0$. By definition, M is larger than $|\tilde{r}|$ and $\tilde{t}$. In Line 4, $\Pi_\mathcal{L}$ is defined such that $\Pi_\mathcal{L} > 2M^2$ (indeed, $\Pi_\mathcal{L}$ is the product of $2K$ primes and with $K > 1$ at least one is larger than 4 thus $\Pi_\mathcal{L} > 2^{2K+1} = 2M^2$), thus $0 < \tilde{t} < M < \frac{\Pi_\mathcal{L}}{2M}$. On the other hand, we prove in Lemma 39 that, modulo a prime $\mu > i$ that does not divide Π, the reduction of $\mathrm{RUR}_i^{\mathbb{Q}}$ is well defined, thus $\gcd(\tilde{t}, \Pi_\mathcal{L}) = 1$. Finally, since, as shown above, $\mathrm{RUR}_i^{\Pi_\mathcal{L}}$ is the reduction modulo $\Pi_\mathcal{L}$ of $\mathrm{RUR}_i^{\mathbb{Q}}$, the RUR of $\widehat{T}_i$, we have that $c = \phi_{\Pi_\mathcal{L}}(\tilde{r}/\tilde{t})$, that is $\tilde{r}\tilde{t}^{-1} = c \mod \Pi_\mathcal{L}$. The unique solution of the rational reconstruction of $\mathrm{RUR}_i^{\Pi_\mathcal{L}}$ is thus well defined and equal to the RUR of $\widehat{T}_i$ in $\mathbb{Q}$.

At the end of Line 10, we have thus computed the sequence of RURs of $\widehat{T}_i$ associated to the separating form $(t, y) \mapsto t$, for all $i \in \mathcal{I}$. This is a RUR decomposition of $\langle \widetilde{P}, \widetilde{Q} \rangle$ since as shown above, the set of solutions of $\{\widetilde{P}, \widetilde{Q}\}$ is the disjoint union of the sets of solutions of all $\widehat{T}_i$ and since $\langle \widetilde{P}, \widetilde{Q} \rangle \subseteq T_i \subseteq \widehat{T}_i$ by Lemma 10.

By definition of $\widetilde{P}$ and $\widetilde{Q}$, the images of these RURs through the mapping $(t, y) \mapsto (x = t - ay, y)$ yield a RUR decomposition of $\langle P, Q \rangle$ associated to the form $(x, y) \mapsto x + ay$. This RUR decomposition is computed in Line 11 using the formulas of Lemma 40.

Finally, the total bitsize of $\widetilde{O}(d^4 + d^3\tau)$ of all the coefficients of this RUR decomposition follows from Corollary 36 since the bitsize of a is in $O(\log d)$ by Theorem 26. □

Proposition 42. *Algorithm 6 computes a RUR decomposition of $\{P, Q\}$ with $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case.*

Proof. The bit complexity of Line 1 is $\widetilde{O}_B(d^6 + d^5\tau)$ by Theorem 26. In Line 2, since a has bitsize in $O(\log d)$, the sheared polynomials $\widetilde{P}$ and $\widetilde{Q}$ can be computed in bit complexity $\widetilde{O}_B(d^4 + d^3\tau)$ and their maximum degrees $\tilde{d}$ and bitsizes $\tilde{\tau}$ are in $O(d)$ and $\widetilde{O}(d + \tau)$, respectively (see e.g. [BLPR15, Lemma 7]).

In Lines 3 and 4, the bit complexities of Algorithms 1 and 2 applied on $\{\widetilde{P}, \widetilde{Q}\}$ are in $\widetilde{O}_B(\tilde{d}^6 + \tilde{d}^5\tilde{\tau})$ by Proposition 16. In Line 4, computing K has bit complexity $\widetilde{O}_B(\log(\tilde{d}\tilde{\tau}))$ (since the constants C and k are known according to the discussion following Proposition 35). Still in Line 4, computing $\mathcal{L}$ can be done by (i) computing the first $2K + \lceil \log \Pi \rceil$ primes larger than $\tilde{d}$, then (ii) reducing Π modulo these primes using a remainder tree [MB74] and (iii) keeping the first $2K$ primes that do not divide Π (there exists at least $2K$ primes that do not divide Π since the number of primes that divide Π is smaller than $\lceil \log \Pi \rceil$). The bit complexity of computing the r first prime numbers is in $\widetilde{O}_B(r)$ and their maximum is in $\widetilde{O}(r)$ [vzGG13, Theorem 18.10]. Hence, since Π has bitsize $\widetilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$ by Proposition 16, phase (i) has bit complexity $\widetilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$, every prime has bitsize $O(\log(\tilde{d}\tilde{\tau}))$ and their product has bitsize $\widetilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$. Phase (ii) can be computed in a bit complexity

that is soft linear in the total bitsize of the input [MB74, Theorem 1], hence in $\widetilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$ bit operations. Therefore, the bit complexity of Lines 3 and 4 is $\widetilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$.

In Lines 5 and 6, the cardinality of $\mathcal{I}$ is $O(\tilde{d})$ (see Algorithm 1) and the cardinality of $\mathcal{L}$ is $2K = \widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$.

In Line 7, every subresultant of $\widetilde{P}$ and $\widetilde{Q}$ (including the resultant) has degree $O(\tilde{d}^2)$ in t and its coefficients have bitsize $\widetilde{O}(\tilde{d}\tilde{\tau})$ by Lemma 3. Furthermore, A_i is factor of $\mathrm{Res}(\widetilde{P}, \widetilde{Q})$ by construction, hence the bitsize of its coefficients is in $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ by Mignotte's lemma (see e.g. [BPR06, Corollary 10.12]). Hence, in Line 7, A_i, $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ and $\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$ have degree $O(\tilde{d}^2)$ and coefficients of bitsize $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$. For every i, the reductions of each of these coefficients modulo the $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ primes μ (of bitsize $O(\log(d\tau))$) in $\mathcal{L}$ can be done, using again a remainder tree, in bit complexity $\widetilde{O}_B(\tilde{d}^2 + \tilde{d}\tilde{\tau})$. The reductions of all $O(\tilde{d}^2)$ coefficients for all $i \in \mathcal{I}$ and all $\mu \in \mathcal{L}$ can thus be done in bit complexity $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

In Line 8, for every i and μ, we compute RUR_i^μ using the formulas of Lemma 38 where the input polynomials are $A = \phi_\mu(A_i)$, $B_1 = \phi_\mu(\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}))$ and $B_0 = \phi_\mu(\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q}))$ in $\mathbb{F}_\mu[t]$. Following these formulas, computing RUR_i^μ can be done with $O(1)$ additions, multiplication and inverse computations in $\mathbb{F}_\mu[t]/\langle A\rangle$ once B_0 and B_1 are reduced in $\mathbb{F}_\mu[t]/\langle A\rangle$. These reductions amount to computing the remainders of the divisions of B_0 and B_1 by A, whose arithmetic complexity in $\mathbb{F}_\mu$ is softly linear in their degrees $O(\tilde{d}^2)$ [vzGG13, Theorem 9.6]. Furthermore, the arithmetic complexity in $\mathbb{F}_\mu$ of every operation in $\mathbb{F}_\mu[t]/\langle A\rangle$ is softly linear in the degree $O(\tilde{d}^2)$ of A [vzGG13, Corollary 11.11]. Summing over all $i \in \mathcal{I}$ and all $\mu \in \mathcal{L}$, the $\widetilde{O}(\tilde{d}^3 + \tilde{d}^2\tilde{\tau})$ RUR_i^μ can be computed with $\widetilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ arithmetic operations in $\mathbb{F}_{\mu\in\mathcal{L}}$. Finally, since every $\mu \in \mathcal{L}$ has bitsize $O(\log(\tilde{d}\tilde{\tau}))$, the total bit complexity of Line 8 is $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

In Line 9, for any given i, the complexity of lifting $\{\mathrm{RUR}_i^\mu\}_{\mu\in\mathcal{L}}$ to $\mathrm{RUR}_i^{\Pi_\mathcal{L}}$ in $\mathbb{Z}/\Pi_\mathcal{L}\mathbb{Z}$ is the complexity of lifting its $O(d^2)$ coefficients. Every coefficient reconstruction in $\mathbb{Z}/\Pi_\mathcal{L}\mathbb{Z}$ can be done using the Chinese Remainder Algorithm with $\widetilde{O}_B(\log \Pi_\mathcal{L}) = \widetilde{O}_B(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ bit operations [vzGG13, Theorem 10.25]. Summing over all coefficients and all i, the total bit complexity of Line 9 is thus in $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

In Line 10, for any given i, the complexity of the rational reconstruction of $\mathrm{RUR}_i^\mathbb{Q}$ from $\mathrm{RUR}_i^{\Pi_\mathcal{L}}$ is the complexity of the rational reconstructions with parameter $2M = 2^{K+1}$ of the $O(d^2)$ rationals coefficients of $\mathrm{RUR}_i^\mathbb{Q}$ from those of $\mathrm{RUR}_i^{\Pi_\mathcal{L}}$ (see the proof of Proposition 41 for details). The rational reconstruction $r/t \in \mathbb{Q}$ of $c \in \mathbb{Z}/\Pi_\mathcal{L}\mathbb{Z}$ with parameter $2M$ is the cost of computing the first line of the Extended Euclidean Algorithm (EEA) for $\Pi_\mathcal{L}$ and c such that the remainder is smaller than $2M$ [vzGG13, Theorem 5.26]. Using binary search, we can compute at most a logarithmic number of lines of the EEA. Since the total number of lines of the EEA and the bit complexity of computing one line of the EEA are (at most) softly linear in the bitsize of the input [vzGG13, Corollary 11.9], the rational reconstruction of one rational has bit complexity $\widetilde{O}_B(\tilde{d}^2 + \tilde{d}\tilde{\tau})$. Summing over all coefficients and all i, the total bit complexity of Line 10 is thus in $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

In Line 11, for every i, the image of $\mathrm{RUR}_i^\mathbb{Q}$ through the reverse shearing can be computed with $O(\tilde{d}^2)$ arithmetic operations on integers of bitsize $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ by Lemma 40. Hence, the bit complexity of Line 11 is trivially $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$. (Note that in Lines 9, 10 and 11 an amortized analysis yields a complexity of $\widetilde{O}_B(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$ by observing that the degrees $\tilde{d}_i \in O(d^2)$ of the first polynomials of RUR_i^μ sum up, over all i, to at most $\tilde{d}^2$.)

Finally, since $\tilde{d}$ and $\tilde{\tau}$ are in $O(d)$ and $\widetilde{O}(d + \tau)$, the total bit complexity of Algorithm 6 is in $\widetilde{O}_B(\tilde{d}^6 + \tilde{d}^5\tilde{\tau})$. □

Propositions 41 and 42 directly yield the following theorem.

Algorithm 6' RUR decomposition – Las Vegas version

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

1: Compute a separating form $x+ay$ for $\{P, Q\}$ with $a \in \mathbb{Z}$ of bitsize $O(\log d)$ such that the leading coefficients of $P(t - ay, y)$ and $Q(t - ay, y)$ with respect to y are coprime (see Theorem 27).
2: Compute $\widetilde{P}(t, y) = P(t - ay, y)$ and $\widetilde{Q}(t, y) = Q(t - ay, y)$, and let $\tilde{d}$ and $\tilde{\tau}$ be their maximum degree and bitsize.
3: Compute the coefficients $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})(t)$ of subresultant sequence of $\widetilde{P}$ and $\widetilde{Q}$ with respect to y and, for i such that $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}) \not\equiv 0$, compute $\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})(t)$ (see Corollary 50).
Compute the polynomials $A_i(t)$, $i \in \mathcal{I}$, of the triangular decomposition of $\widetilde{P}$ and $\widetilde{Q}$ following Algorithm 1.
Let $\widehat{T}_i = \langle A_i, i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})\, y + \mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})\rangle$, $i \in \mathcal{I}$, be the radicals of the ideals output by Algorithm 1$(\widetilde{P}, \widetilde{Q})$ (see Lemma 37 and note that, in Algorithm 1, $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}) \not\equiv 0$ for $i \in \mathcal{I}$).
4: Let $K = \lceil C(\tilde{d}^2 + \tilde{d}\tilde{\tau})\log^k(\tilde{d}\tilde{\tau})\rceil$ be an integer that bounds from above the bitsize of the coefficients of the RURs of the systems $\widehat{T}_i$ (see Proposition 35 and subsequent discussion). Let $U = 8K$ and $\mathcal{L} = \emptyset$.
5: **repeat**
6: Double U, choose uniformly at random $8K$ primes in $[1, U]$, and let $\mathcal{P}$ be the resulting set.
7: For all $i \in \mathcal{I}$, $\mu \in \mathcal{P}$, reduce A_i and $i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ modulo μ (using remainder trees).
8: Add in $\mathcal{L}$ the $\mu \in \mathcal{P}$ such that, $\forall i$, $\phi_\mu(A_i)$ is squarefree and coprime with $\phi_\mu(i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}))$.
9: **until** $\mathcal{L}$ contains at least $2K$ distinct primes.
10: **return** The image of $\mathrm{RUR}_i^{\mathbb{Q}}$, the RUR of $\widehat{T}_i$, $i \in \mathcal{I}$, through the reverse shearing from (t, y) to (x, y), as in Algorithm 6, Lines 5-11.

Theorem 43. *Let P, Q in $\mathbb{Z}[x, y]$ be of total degree at most d and maximum bitsize τ. Algorithm 6 computes, with $\widetilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case, a RUR decomposition of $\{P, Q\}$ of total bitsize $\widetilde{O}(d^4 + d^3\tau)$.*

6.3 Las Vegas algorithm

We show here that the algorithm presented above for computing a RUR decomposition can easily be transformed into an efficient Las Vegas algorithm. We prove here the following.

Theorem 44. *Let P, Q in $\mathbb{Z}[x, y]$ be of total degree at most d and maximum bitsize τ. Algorithm 6' computes, with $\widetilde{O}_B(d^5 + d^4\tau)$ bit operations on average, a RUR decomposition of $\{P, Q\}$ of total bitsize $\widetilde{O}(d^4 + d^3\tau)$.*

Algorithm 6', our Las Vegas version of Algorithm 6, is obtained from the latter with only three modifications. First, in Line 2, we use the Las Vegas version of our algorithm for computing a separating linear form for $\{P, Q\}$, described in Section 5.5.

Second, in Line 3, we modify the way we compute the radicals $\widehat{T}_i$ of the ideals T_i output by Algorithm 1$(\widetilde{P}, \widetilde{Q})$. We still use the formula $\widehat{T}_i = \langle A_i, i\,\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})\, y + \mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})\rangle$ of Lemma 37 for computing these radical ideals, but instead of computing the T_i with Algorithm 1, we show in Section 6.3.1 that the subresultant coefficients $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ and $\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$ can be computed more efficiently.

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

We prove Theorem 44 by first proving its correctness in Proposition 45 and then its complexity in Proposition 48.

Proposition 45 (Correctness of Algorithm 6'). *Algorithm 6' computes a RUR decomposition of* $\{P, Q\}$ *of total bitsize* $\widetilde{O}(d^4 + d^3\tau)$.

Proof. As described above, Algorithm 6' is obtained with only three modifications from Algorithm 6, whose correctness is proved in Proposition 41. The first two modifications do not jeopardize the correctness of Algorithm 6' since we compute the same objects as in Algorithm 6 (in particular, we use the same formula for $\widehat{T}_i$, $i \in \mathcal{I}$). However, in the third modification, we weaken the constraints on the primes of $\mathcal{L}$. In the proof of correctness of Algorithm 6, the constraints on the primes of $\mathcal{L}$ (that $\mu > \tilde{d}$ does not divide Π) are only used in Lemma 39. Furthermore, in proof of Lemma 39, these constraints are only used for proving that $\phi_\mu(\widehat{T}_i) = \langle \phi_\mu(A_i), \phi_\mu(i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q}))\, y + \phi_\mu(\operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q}))\rangle$ satisfies the hypotheses of Lemma 38, that is that $\phi_\mu(A_i)$ is squarefree and coprime with $\phi_\mu(i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q}))$, which are the constraints on μ we impose in Line 8 of Algorithm 6'. The correctness of Algorithm 6' thus follows from that of Algorithm 6. □

We now analyse the complexity of Algorithm 6'. A key step of this algorithm is the computation, in Line 3, of $\operatorname{sres}_i(\widetilde{P}, \widetilde{Q})$ and $\operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$, which we postpone to Section 6.3.1. Before proving Proposition 48, which states the complexity of Algorithm 6', we prove two lemmas. The first one bounds the number of primes that are rejected in Line 8 and the second one will be instrumental for bounding the probability that the loop ends in Line 9.

Lemma 46. *There are* $\widetilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ *primes that are unlucky for* $\gcd(A_i, i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q}))$ *or* $\gcd(A_i, A_i')$, *for some* i. *Furthermore, if prime* μ *is lucky for these two gcds, for some* i, *then* $\phi_\mu(A_i)$ *is squarefree and coprime with* $\phi_\mu(i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q}))$.

Proof. By Lemma 7, the unlucky primes for the gcd of two polynomials A and B in $\mathbb{Z}[t]$ are exactly the divisors of their leading coefficients and the divisors of $\operatorname{sres}_d(A, B)$ where d is the degree of $\gcd(A, B)$. In order to bound the number of unlucky primes, we bound the bitsizes of the relevant coefficients.

By Lemma 10, A_i divides the resultant $\operatorname{Res}(\widetilde{P}, \widetilde{Q})$. Thus, A_i has degree $O(\tilde{d}^2)$ and coefficients of bitsize $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$, as shown in the proof of Proposition 42. It follows that the same bounds also apply to A_i'. On the other hand, $i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q})$ has degree $O(\tilde{d}^2)$ and coefficients of bitsize $\widetilde{O}(\tilde{d}\tilde{\tau})$, by Lemma 3. Still by Lemma 3, the coefficients of the subresultant polynomials of any two of these polynomials have bitsize $\widetilde{O}(\tilde{d}^2(\tilde{d}^2 + \tilde{d}\tilde{\tau}))$. The number of prime divisors of any such coefficient is thus also in $\widetilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$. Since i varies from 1 to at most $\tilde{d}$ (see Algorithm 1), the number of unlucky primes is in $\widetilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

Finally, for any i, both $\gcd(A_i, i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q}))$ and $\gcd(A_i, A_i')$ are equal to constants, by Lemma 10. Furthermore, if μ is lucky for these gcds, these gcds commute with ϕ_μ, by Lemma 7. Hence, $\phi_\mu(A_i)$ is squarefree and coprime with $\phi_\mu(i \operatorname{sres}_i(\widetilde{P}, \widetilde{Q}))$. □

Lemma 47. *Let* n_{2p} *be the random variable that represents the number of distinct elements obtained by choosing uniformly at random* $2p$ *elements among* n *with replacement. If* $n \geqslant 2p \geqslant 4$, *then the probability that* $n_{2p} > p$ *is larger than* $\frac{1}{2}$.

Proof. Consider one of the $\binom{n}{d}$ sets of d distinct elements among n. Denote it by S_d, denote the set of p random elements by S and its cardinal by $|S|$. The probability that $S \subseteq S_d$, which is the probability that the p random elements in S are all in S_d is $\left(\frac{d}{n}\right)^p$. On the other hand, $\Pr(|S| \leqslant d)$

is less than the sum of all $\Pr(S \subseteq S_d)$ for the $\binom{n}{d}$ choices of sets S_d. Hence, $\Pr(|S| \leqslant d) < \binom{n}{d}\left(\frac{d}{n}\right)^p$ and $\Pr(|S| > d) \geqslant 1 - \binom{n}{d}\left(\frac{d}{n}\right)^p$.

Setting $p = 2\,d$ and using Stirling's approximation $\sqrt{2\pi}\,n^{n+1/2}e^{-n} \leqslant n! \leqslant e\,n^{n+1/2}e^{-n}$, we obtain that

$$\binom{n}{d}\left(\frac{d}{n}\right)^{2d} = \frac{n!}{d!(n-d)!}\frac{d^{2d}}{n^{2d}} \tag{3}$$

$$\leqslant \frac{e\,n^{n+\frac{1}{2}}e^{-n}}{2\pi d^{d+\frac{1}{2}}e^{-d}(n-d)^{n-d+\frac{1}{2}}e^{-(n-d)}}\frac{d^{2d}}{n^{2d}} = \frac{e}{2\pi}\frac{n^{n+\frac{1}{2}-2d}d^{d-\frac{1}{2}}}{(n-d)^{n-d+\frac{1}{2}}}. \tag{4}$$

Replacing n by kd with $k \geqslant 2$, we get

$$\binom{n}{d}\left(\frac{d}{n}\right)^{2d} \leqslant \frac{e}{2\pi}\frac{\left(\frac{k}{k-1}\right)^{(k-2)d+\frac{1}{2}}}{d^{\frac{1}{2}}(k-1)^d} \tag{5}$$

and the derivative with respect to d of the right-hand side of the inequality is

$$\frac{e}{2\pi}\frac{\left(\frac{k}{k-1}\right)^{(k-2)d+\frac{1}{2}}}{d^{\frac{3}{2}}(k-1)^d}\left(-1 + 2d\ln\frac{k^{k-2}}{(k-1)^{k-1}}\right). \tag{6}$$

It is straightforward to prove that the function $k \mapsto \frac{k^{k-2}}{(k-1)^{k-1}}$ is decreasing for $k \geqslant 2$, hence $\ln\frac{k^{k-2}}{(k-1)^{k-1}}$ is negative for $k > 2$ and (6) is negative for $k \geqslant 2$. It follows that, for $d > 2$, the right-hand side of (5) is smaller than $\frac{e}{2\pi}\frac{\left(\frac{k}{k-1}\right)^{2k-\frac{7}{2}}}{\sqrt{2}(k-1)^2}$. It is straightforward to show that this is decreasing for $k \geqslant 2$ and it is thus less than $\frac{e}{2\pi}\frac{\sqrt{2}}{\sqrt{2}} = \frac{e}{2\pi} < \frac{1}{2}$. Therefore, for $n \geqslant 2d$ and $d \geqslant 2$, $\binom{n}{d}\left(\frac{d}{n}\right)^{2d} < \frac{1}{2}$ and thus $\Pr(|S| > d) > \frac{1}{2}$. □

Proposition 48. *Algorithm 6' computes a RUR decomposition of $\{P, Q\}$ with $\widetilde{O}_B(d^5 + d^4\tau)$ bit operations on average.*

Proof. The expected bit complexity of Line 1 is $\widetilde{O}_B(d^5 + d^4\tau)$ by Theorem 27 and, as in Algorithm 6, the (worst-case) bit complexity of Line 2 is $\widetilde{O}_B(d^4 + d^3\tau)$ and $\widetilde{P}$ and $\widetilde{Q}$ have maximum degree $\tilde{d} \in O(d)$ and maximum bitsize $\tilde{\tau} \in \widetilde{O}(d + \tau)$ (see the proof of Proposition 42).

In Line 3, the sequence of coefficients $\operatorname{sres}_i(\widetilde{P}, \widetilde{Q})$ and, for those that do not identically vanish, the coefficients $\operatorname{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$ can be computed in $\widetilde{O}_B(\tilde{d}^4\tilde{\tau})$ bit operations by Corollary 50. Hence, the sequence of polynomials A_i can be computed in $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ bit operations by Remark 17. We thus get, in Line 3, the sequence of ideals $\widehat{T}_i$ in $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ bit operations.

In Line 4, the complexity of computing K and U is $\widetilde{O}_B(\log(\tilde{d}\tilde{\tau}))$, as in Algorithm 6.

In Line 6, we choose uniformly at random, one at a time, $8K$ primes in $[1, U]$. Some primes might be chosen more than once and thus the resulting set of primes, $\mathcal{P}$, may be of cardinality smaller than $8K$. The analysis is similar to the one in Proposition 31. A random integer in $[1, U]$ can be computed in $O_B(\log U)$ bit operations. There are at least $\frac{U}{\ln U}$ primes in $[1, U]$ [vzGG13, Theorem 18.7]. The probability that a randomly chosen integer in $[1, U]$ is prime is thus at least $\frac{1}{\ln U}$ and a prime is thus found after at most $\ln U$ trials on average. Testing whether an integer in $[1, U]$ is prime can be done with a polynomial bit complexity in the bitsize of U, $\widetilde{O}_B(\log^{7.5} U)$ [AKS04].

The expected bit complexity of computing a prime in Line 6 is thus $\widetilde{O}_B(\log^{8.5} U)$ and the expected bit complexity of computing $8K \in \widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ primes in Line 6 is thus in $\widetilde{O}_B((\tilde{d}^2 + \tilde{d}\tilde{\tau}) \log^{8.5} U)$.

In Line 7, each of the $O(\tilde{d})$ polynomials A_i and i $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ have $O(\tilde{d}^2)$ coefficients of bitsize $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$, as shown in the proof of Lemma 46. Using remainder trees [MB74], the reductions of one coefficient modulo all the primes in $\mathcal{L}$ can be done in a bit complexity that is softly linear in the maximum bitsize of the coefficient and the product of the primes, that is in $\widetilde{O}_B((\tilde{d}^2 + \tilde{d}\tilde{\tau}) + (\tilde{d}^2 + \tilde{d}\tilde{\tau}) \log U)$. Hence, the bit complexity of Line 7 is $\widetilde{O}_B((\tilde{d}^5 + \tilde{d}^4\tilde{\tau}) \log U)$.

In Line 8, for any i, $\gcd(\phi_\mu(A_i), \phi_\mu(i\ \mathrm{sres}_i(\widetilde{P}, \widetilde{Q})))$ and $\gcd(\phi_\mu(A_i), \phi_\mu(A_i'))$ can be computed in $\mathbb{F}_\mu[t]$ in $\widetilde{O}_B(\tilde{d}^2 \log U)$ bit operations, by Lemma 4, since the polynomials have degree $O(\tilde{d}^2)$ and μ has bitsize $O(\log U)$. Hence, the bit complexity of Line 8 is $\widetilde{O}_B((\tilde{d}^5 + \tilde{d}^4\tilde{\tau}) \log U)$ since $i \leqslant \tilde{d}$ and the number of primes in $\mathcal{P}$ is in $\widetilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$.

We have shown that the expected bit complexity of one iteration of the loop in Lines 5 to 9 is in $\widetilde{O}_B((\tilde{d}^5 + \tilde{d}^4\tilde{\tau}) \log^9 U)$. At the end of the j-th iteration of the loop, $U = 2^j \cdot 8K$, thus the expected bit complexity of the j-th iteration of the loop is in $\widetilde{O}_B((\tilde{d}^5 + \tilde{d}^4\tilde{\tau})j^9)$.

We now bound the total expected bit complexity of all the iterations of the loop in Lines 5 to 9. By Lemma 46, the primes that are rejected in Line 8 are unlucky for some $\gcd(A_i, i\ \mathrm{sres}_i(\widetilde{P}, \widetilde{Q}))$ or $\gcd(A_i, A_i')$ and there are less than $\Gamma = C'(\tilde{d}^5 + \tilde{d}^4\tilde{\tau}) \log^{k'}(\tilde{d}\tilde{\tau})$ such unlucky primes for some constants C' and k'. We refer in the rest of the proof to *these* unlucky primes simply as unlucky primes. It follows that the probability that the loop ends in Line 9 is larger than the probability that $\mathcal{P}$ contains at least $2K$ distinct lucky primes. Furthermore,

$$\begin{aligned}\Pr(\mathcal{P} \text{ contains } 2K \text{ lucky primes}) &\geqslant \Pr(\mathcal{P} \text{ contains } 2K \text{ lucky primes and } 4K \text{ primes}) \\ &\geqslant \Pr(\mathcal{P} \text{ contains } 4K \text{ primes}) \\ &\quad \cdot \Pr(\mathcal{P} \text{ contains } 2K \text{ lucky primes } \mid \mathcal{P} \text{ contains } 4K \text{ primes}).\end{aligned}$$

As seen above, $[1, U]$ contains at least $\frac{U}{\ln U}$ primes. Thus, when $\frac{U}{\ln U} \geqslant 8K$, $\mathcal{P}$ contains at least $4K$ distinct primes with probability at least $\frac{1}{2}$, by Lemma 47. On the other hand, the primes in $\mathcal{P}$ are chosen uniformly at random among at least $\frac{U}{\ln U}$ primes, thus if $\frac{U}{\ln U} \geqslant 2\Gamma$, the primes in $\mathcal{P}$ are lucky with probability at least $\frac{1}{2}$. Thus, if $\frac{U}{\ln U} \geqslant 2\Gamma$, given that $\mathcal{P}$ contains at least $4K$ primes, the probability that $\mathcal{P}$ contains at least $2K$ lucky primes is at least $\frac{1}{2}$. We thus have proved that, if $\frac{U}{\ln U} \geqslant \max(8K, 2\Gamma)$, the loop ends in Line 9 with probability at least $\frac{1}{4}$.

There are $O(\log(\tilde{d}\tilde{\tau}))$ loop iterations that are performed while $\frac{U}{\ln U}$ is smaller than $\max(8K, 2\Gamma)$. Indeed, $\log U \in O(\log(\tilde{d}\tilde{\tau}))$ while $\frac{U}{\ln U} < \max(8K, 2\Gamma) \in \widetilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ since $\sqrt{U} < \frac{U}{\ln U}$. The overall bit complexity of these iterations is thus in $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$. It follows that the expected bit complexity of the entire loop is in $\widetilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$, by Lemma 30.

Summing up the complexities of all lines and since $\tilde{d} \in \widetilde{O}(d)$ and $\tilde{\tau} \in \widetilde{O}(d + \tau)$, we obtain that the expected bit complexity of the algorithm is $\widetilde{O}_B(d^5 + d^4\tau)$. □

6.3.1 Computation of subresultant coefficients

A key step of Algorithm 6' is the computation of the coefficients $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q})$ and the computation of $\mathrm{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})$ when $\mathrm{sres}_i(\widetilde{P}, \widetilde{Q}) \not\equiv 0$. We show that all these coefficients can be computed in $\widetilde{O}_B(d^4\tau)$ bit complexity in Theorem 49 and Corollary 50. This result generalizes [vzGG13, Corollary 11.18] to the case where one wants to compute the k terms of greater degrees in the sequence of remainders in the Euclidean algorithm.

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

degree $\deg(P) - n$, if any, and 0 otherwise. It follows that $r_{i|j}$ denotes the coefficient of the term of r_i of degree $\deg(r_i) - j$.

Theorem 49. *Let k be an integer and $P, Q \in \mathbb{F}[y]$ be two polynomials with $d = \deg(P) \geqslant \deg(Q)$. We can compute, for all $0 \leqslant j \leqslant k$ and for all the remainders r_i appearing in the Euclidean algorithm, the coefficients $r_{i|j}$ in $O(k^2 d + M(d)\log d)$ arithmetic operations, where $M(d)$ is the complexity of the multiplication of degree d polynomials.*

Proof. First, all the quotients q_i appearing in the remainder sequence can be computed in $O(M(d)\log d)$ arithmetic operations ([vzGG13, Corollary 11.9]). Then, for $k = 0$, we have directly the coefficients $r_{0|0}$ and $r_{1|0}$, and from the formula

$$r_{i-1} = q_i r_i + r_{i+1} \text{ such that } \deg(r_{i+1}) < \deg(r_i)$$

we deduce that $r_{i|0} = \frac{r_{i-1|0}}{q_{i|0}}$. Thus we can compute by recurrence all the $r_{i|0}$ with less than d divisions.

Assume now that we have computed the coefficients $r_{i|j}$ for all i and $0 \leqslant j \leqslant k-1$. We show that in this case, we can compute the coefficients $r_{i|k}$, for all i, in $O(kd)$ arithmetic operations.

From the recurrence formula in the Euclidean algorithm, we can derive the following equality:

$$r_{i-1|k} = r_{i|k} q_{i|0} + \cdots + r_{i|0} q_{i|k} + r_{i+1|l}$$

where $l = k + \deg(r_{i+1}) - \deg(r_{i-1}) < k$. Thus,

$$r_{i|k} = \frac{r_{i-1|k} - r_{i|k-1} q_{i|1} - \cdots - r_{i|0} q_{i|k} - r_{i+1|l}}{q_{i|0}},$$

which yields $r_{i|k}$ from the values of $r_{i-1|k}$, $r_{i|j}, r_{i+1|l}$, with $j, l \leqslant k-1$, in $2k+2$ arithmetic operations. Thus, given the coefficients $r_{i|j}$ for all i and $0 \leqslant j \leqslant k-1$, we can compute the $r_{i|k}$, for all i, in $O(kd)$ arithmetic operations, which trivially concludes the proof. □

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

Corollary 50. *Let $P, Q \in \mathbb{Z}[x, y]$ be of degree at most d with coefficients of bitsize at most τ. We can compute in $\widetilde{O}_B(d^4\tau)$ bit operations in the worst case the sequence of all subresultant coefficients $\mathrm{sres}_i(P, Q)$ and, for i such that $\mathrm{sres}_i(P, Q) \not\equiv 0$, the coefficients $\mathrm{sres}_{i,i-1}(P, Q)$.*

Proof. We compute the subresultant coefficients using multimodular and interpolation techniques. First, we select pairs (μ, k) with μ prime and k a value in $\mathbb{F}_\mu$ satisfying the specialization property of the subresultants. Second, we compute the subresultant coefficients $\mathrm{sres}_i(P, Q)$ and $\mathrm{sres}_{i,i-1}(P, Q)$ evaluated at $x = k$ in $\mathbb{F}_\mu$. Third, we interpolate the results in $\mathbb{F}_\mu[x]$ and apply the Chinese remainder algorithm to recover the final results in $\mathbb{Z}[x]$.

To use the specialization property of subresultants, the leading coefficients of P and Q seen as polynomials in y, $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$, must not vanish when evaluated at $x = k$ in $\mathbb{F}_\mu$. The coefficients of P and Q being of bitsize at most τ, there are at most 2τ primes μ such that $\mathrm{Lc}_y(P)$ or $\mathrm{Lc}_y(Q)$ identically vanish modulo μ. When both do not identically vanish modulo μ, they are polynomials of degree at most d, hence there are at most $2d$ values in $\mathbb{F}_\mu$ at which one of them vanishes. In $\mathbb{F}_\mu$, we will compute the subresultant coefficients via evaluation and interpolation. The number of evaluation values must be larger than the degrees of the subresultant coefficients $\mathrm{sres}_i(P, Q)$ and $\mathrm{sres}_{i,i-1}(P, Q)$, which are at most $2d^2$. It is sufficient to consider primes μ larger than $2d^2 + 2d$ because, then, there are at least $2d^2$ values in $\mathbb{F}_\mu$ such that none of $\mathrm{Lc}_y(P)$ and

$\mathrm{Lc}_y(Q)$ vanishes modulo μ. For lifting the subresultants using the Chinese remainder algorithm, the sum of the bitsizes of the primes must be larger than the bitsizes of the subresultants coefficients $\mathrm{sres}_i(P,Q)$ and $\mathrm{sres}_{i,i-1}(P,Q)$, which are at most $N = 2d(\tau + 2\log d)$ [BPR06, Proposition 8.46].

According to [vzGG13, Theorem 18.10], we can compute the M first primes $\mu_j \in \mathbb{Z}$ of bitsizes τ_j in $\widetilde{O}_B(M)$ bit operations and their maximum bitsize is in $O(\log M)$. Among this set, the constraint for the specialization property of subresultants discards at most 2τ primes, and the constraint for the interpolation discards at most the first $2d^2+2d$ primes. Choosing $M = N+2d^2+2d+2\tau = O(d^2+d\tau)$ is thus sufficient to select a set of N primes satisfying these constraints. In addition, the sum of the bitsizes of these N primes is larger than N and in $O(N \log M) = \widetilde{O}(d\tau)$.

We now analyze the complexity of selecting N primes μ_j satisfying the above constraints and specializing P and Q at $2d^2$ values $x = k$ in $\mathbb{F}_{\mu_j}[y]$. The reduction of one coefficient of P and Q modulo all the $N + 2\tau$ primes larger than $2d^2 + 2d$ can be computed via a remainder tree in a bit complexity that is soft linear in the total bitsize of the input [MB74, Theorem 1], which is in $\widetilde{O}(d\tau)$. The reductions of all the $O(d^2)$ coefficients of P and Q can hence be done in $\widetilde{O}_B(d^3\tau)$ bit operations. We select N primes μ_j such that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ do not identically vanish modulo μ_j. For a given prime μ_j, the evaluation of the reduction of $P(x,y)$ in $\mathbb{F}_{\mu_j}[x,y]$ at $2d^2+2d$ values $x = k_\ell \in \mathbb{F}_{\mu_j}$ involves $O(d^2)$ evaluations of $O(d)$ polynomials of degree $O(d)$ in $\mathbb{F}_{\mu_j}[x]$. For a given prime μ_j, this can be done using multi-evaluation in $\widetilde{O}(d^3)$ arithmetic operations in $\mathbb{F}_{\mu_j}$ [vzGG13, Corollary 10.8] and thus with $\widetilde{O}_B(d^3\tau_j) = \widetilde{O}_B(d^3 \log M) = \widetilde{O}_B(d^3 \log(d\tau))$ bit operations. For all N primes, the total bit complexity of these evaluations is thus in $\widetilde{O}_B(Nd^3 \log(d\tau)) = \widetilde{O}_B(d^4\tau)$. For each prime μ_j, we select $2d^2$ values k_ℓ, among the $2d^2+2d$ values considered in $\mathbb{F}_{\mu_j}$, at which neither $\mathrm{Lc}_y(P)$ nor $\mathrm{Lc}_y(Q)$ vanishes when evaluated at $x = k_\ell$ in $\mathbb{F}_{\mu_j}$.

In this paragraph, all polynomials are considered evaluated at $x = k$ and in $\mathbb{F}_{\mu_j}[y]$ and, to clarify the presentation, any polynomial $\widetilde{K}$ refers to $K(k,y) \mod \mu_j$. Then computing, for all i, $\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$ can be done in a total of $\widetilde{O}_B(d\tau_j)$ bit operations [vzGG13, Corollary 11.18]. If $\mathrm{sres}_i(\widetilde{P},\widetilde{Q}) \neq 0$, let r be the remainder of degree i appearing in the Euclidean algorithm of $\widetilde{P}$ and $\widetilde{Q}$. We know that r and $\mathrm{Sres}_i(\widetilde{P},\widetilde{Q})$ are equal up to a constant [BPR06, Corollary 8.34], thus $\mathrm{Sres}_i(\widetilde{P},\widetilde{Q}) = \frac{\mathrm{Lc}_y(\mathrm{Sres}_i(\widetilde{P},\widetilde{Q}))}{\mathrm{Lc}_y(r)} r = \frac{\mathrm{sres}_i(\widetilde{P},\widetilde{Q})}{r_{|0}} r$, which directly implies that $\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q}) = \frac{\mathrm{sres}_i(\widetilde{P},\widetilde{Q})}{r_{|0}} r_{|1}$. Using Theorem 49, we can compute $r_{|0}$ and $r_{|1}$ in $\mathbb{F}_{\mu_j}[y]$ in $\widetilde{O}_B(d\tau_j)$ bit operations, which yields $\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})$.

Thus, for a given μ_j, computing the two first subresultant coefficients $\mathrm{sres}_i(\widetilde{P},\widetilde{Q})$ and $\mathrm{sres}_{i,i-1}(\widetilde{P},\widetilde{Q})$ for $2d^2$ values of k in $\mathbb{F}_{\mu_j}$ costs $\widetilde{O}_B(d^3\tau_j)$ bit operations. Then using fast interpolation [vzGG13, Corollary 10.12], we can recover $\mathrm{sres}_i(P,Q) \mod \mu_j$ and $\mathrm{sres}_{i,i-1}(P,Q) \mod \mu_j$ in $\widetilde{O}_B(d^3\tau_j) = \widetilde{O}_B(d^3 \log(d\tau))$ bit operations, which sums up to $\widetilde{O}_B(d^4\tau)$ for all $N = \widetilde{O}(d\tau)$ values of μ_j. Finally, recovering all the $O(d^3)$ coefficients of $\mathrm{sres}_i(P,Q)$ and $\mathrm{sres}_{i,i-1}(P,Q)$ (whose bitsizes are smaller than N) can be done with $\widetilde{O}_B(d^3 N \log M) = \widetilde{O}_B(d^4\tau)$ bit operations with the Chinese remainder algorithm [vzGG13, Theorem 10.25]. □

7 Computing isolating boxes from a RUR decomposition

By definition, the RUR of an ideal I defines a mapping between the roots of a univariate polynomial and the solutions of I. Based on this mapping, Algorithm 7 computes isolating boxes using univariate isolation and approximate polynomial evaluation. Section 7.1 recalls or proves several complexity results on isolation and evaluation of univariate polynomials. In Section 7.2, the isolation algorithm using fast approximate multipoint evaluation is presented and analyzed in

Theorem 59.

7.1 Preliminaries

We start with some basic definitions. In addition, we recall some bounds on univariate polynomial roots and their separation (for a single root and also amortized over all the roots), the complexity of isolating the roots of a univariate polynomial, and elementary results on approximate polynomial evaluation.

For an arbitrary complex value x, we define $M(x) = \max(1, |x|)$. In addition, let L be an arbitrary positive integer. Then, we define $\tilde{x} \in \mathbb{Q}+i\,\mathbb{Q}$ to be an *absolute dyadic L-bit approximation of x* (or just L-bit approximation for short) if $\tilde{x}$ is of the form $\tilde{x} = (m_{\Re} + i\, m_{\Im}) \cdot 2^{-L-2}$, with $m_{\Re}, m_{\Im} \in \mathbb{Z}$, and $|x - \tilde{x}| < 2^{-L}$. Notice that an L-bit approximation $\tilde{x} = (m_{\Re} + i\, m_{\Im}) \cdot 2^{-L-2}$ of some point $x \in \mathbb{C}$ naturally defines a box

$$B(\tilde{x}) = \frac{[m_{\Re} - 4, m_{\Re} + 4]}{2^{L+2}} + i \cdot \frac{[m_{\Im} - 4, m_{\Im} + 4]}{2^{L+2}} \subset \mathbb{C} \tag{7}$$

of width 2^{-L+1} in $\mathbb{C}$ that contains x.

For a complex root γ of a polynomial $f \in \mathbb{Z}[x]$ and an arbitrary positive integer L, we say that a connected region D in $\mathbb{C}$ (typically, we consider a disk or a box) is isolating for γ (or that D isolates γ) if it contains γ but no other root of f. We define the *separation of* γ (with respect to f), denoted $\mathrm{sep}(\gamma, f)$, to be the minimal distance between γ and any root γ' of f, with $\gamma' \neq \gamma$. The *separation of* f is defined as $\mathrm{sep}(f) = \min_{\gamma : f(\gamma)=0} \mathrm{sep}(\gamma, f)$. The same notions for a zero-dimensional ideal of $\mathbb{Z}[x, y]$ are also naturally defined.

We now recall some well-known facts about the separations and the magnitudes of the complex roots of a univariate polynomial f of degree d with integer coefficients of bitsize at most τ.

Lemma 51 ([Yap00, §6.2 Lemma 6.5])**.** *For any root $\gamma \in \mathbb{C}$ of f, $M(\gamma) = 2^{O(\tau)}$.*

Lemma 52 ([SY11])**.** *If f is squarefree, $\prod_{\{\gamma \text{ root of } f\}} \min(1, \mathrm{sep}(\gamma, f)) = 2^{-\widetilde{O}(d\tau)}$.*

Lemma 53 ([Yap00, Lemma 6.34])**.** *Let f and g be coprime polynomials of degree at most d with integer coefficients of bitsize at most τ. Then, for any root $\gamma \in \mathbb{C}$ of f, $|g(\gamma)| = 2^{-O(d(\tau + \log d))}$.*

Lemma 54 ([MSW15, Theorem 5])**.** *We can compute isolating disks D_i with radius $r_i < \frac{\mathrm{sep}(\gamma_i, f)}{64d}$ for all complex roots γ_i of f using $\widetilde{O}_B(d^3 + d^2\tau)$ bit operations. For an arbitrary positive integer L, we can compute corresponding L-bit approximations $\tilde{\gamma}_i$ for all roots using $\widetilde{O}_B(d^3 + d^2\tau + dL)$ bit operations.*

Proof. The first part follows directly from [MSW15, Theorem 5]. In addition, [MSW15, Theorem 5] also states that we can further refine the disks D_i such that each of them has radius less than 2^{-L-2} using $\widetilde{O}_B(d^3 + d^2\tau + dL)$ bit operations. In addition, the centers of the disks are computed in dyadic form. We can thus round the center of each disk D_i to an absolute precision of size 2^{-L-2} to obtain an L-bit approximation $\tilde{\gamma}_i$ of each root γ_i of f. The bit complexity of rounding all the disks' centers is linear in the total bitsize of the dyadic coordinates, which is bounded by $\widetilde{O}_B(d^3 + d^2\tau + dL)$, the complexity of the algorithm that computes them. □

We further remark that there also exist dedicated real root isolation and refinement methods [SM16, KS15a] that compute isolating intervals of size 2^{-L} for all real roots of f with a number of bit operations that is comparable to the bound stated in Lemma 54. When computing the solutions of a bivariate system (see Section 7.2), the choice of an efficient univariate solver is critical,

and thus we propose to use a dedicated method for real root finding if only the real solutions of the bivariate system are asked for.

Now, suppose that we want to approximately evaluate a polynomial $g \in \mathbb{Z}[x]$ of degree d_g with integer coefficients of bitsize τ_g at all roots of f. More precisely, for a given positive integer L, we are aiming for L-bit approximations $\tilde{y}_i$ of the values $y_i = g(\gamma_i)$, where $\gamma_1, \ldots, \gamma_d$ denote the roots of f. For this, we use fast approximate multipoint evaluation.

Lemma 55 ([KS15b, Theorem 22])**.** *Let $x_1, \ldots, x_{d_g} \in \mathbb{C}$ be such that, for each of them, an L'-bit approximation can be accessed in $O_B(L')$ bit operations. For any positive integer L, we can compute L-bit approximations of all values $y_i = g(x_i)$ using $\widetilde{O}_B(d_g(L+\tau_g+d_g\Gamma))$ bit operations, where $\Gamma \geqslant 1$ is an upper bound on the maximum of all values $\log M(x_i)$. For the computation, we need L'-bit approximations of all points x_i, where $L' = L + \widetilde{O}(\tau_g + d_g\Gamma)$.*

We remark that the multipoint evaluation algorithm from [KS15b] uses certified interval arithmetic based on fixed-point computations. It adaptively increases the (absolute) working precision L' during the computation. That is, in each iteration, it asks for L'-bit approximations $\tilde{x}_i$ of the points x_i, and if it does not succeed to compute L-bit approximations $\tilde{y}_i$ of the values y_i, it doubles the precision and restarts. Hence, the algorithm might also succeed with a smaller precision than the precision predicted in the worst case.

Lemma 56. *Let $f \in \mathbb{Z}[x]$ be a polynomial of degree d with integer coefficients of bitsize at most τ and let $\gamma_1, \ldots, \gamma_d$ denote the roots of f. Let $g \in \mathbb{Z}[x]$ be a polynomial of degree $d_g = O(d)$ with integer coefficients of bitsize at most τ_g. Then, for any given positive integer L, we can compute L-bit approximations of all values $g(\gamma_i)$ using a number of bit operations bounded by $\widetilde{O}_B(d^3 + d^2\tau + d\,(L + \tau_g))$.*

Proof. Applying Lemma 55 $\lceil d/d_g \rceil$ times, L-bit approximations of d values $g(x_i)$ can be computed with $\widetilde{O}_B(\lceil d/d_g \rceil d_g(L + \tau_g + d_g\Gamma))$ bit operations assuming that we can access each L'-bit approximation of x_i in $O_B(L')$ bit operations. Moreover, as mentioned above, the L-bit approximations of the $g(x_i)$ are computed iteratively by doubling L' at every iteration and the algorithm stops with $L' = L + \widetilde{O}(\tau_g + d_g\Gamma)$. Thus, the number of iterations is in $O(\log(L + \tau_g + d_g\Gamma))$.

By Lemma 54, L'-bit approximations of the d roots of f can be computed in $\widetilde{O}_B(d^3 + d^2\tau + dL')$ bit operations. Thus, these approximations can be computed for all iterations in $\widetilde{O}_B(d^3 + d^2\tau + d(L + \tau_g + d_g\Gamma))$ bit operations.

The total complexity is thus in $\widetilde{O}_B(\lceil d/d_g \rceil d_g(L + \tau_g + d_g\Gamma)) + \widetilde{O}_B(d^3 + d^2\tau + d(L + \tau_g + d_g\Gamma))$. The result follows since $d_g = O(d)$ and since $\Gamma = O(\tau)$ by Lemma 51. □

We can further extend the above result to the evaluation of a fraction $G = \frac{g_1}{g_2}$ at the roots γ_i of f, where g_1 and g_2 are both polynomials of degree bounded by $O(d)$ with integer coefficients of bitsize less than τ_G, and g_2 is coprime with f.

Lemma 57. *Let $G = \frac{g_1}{g_2}$, with $g_1, g_2 \in \mathbb{Z}[x]$ polynomials of degree at most $d_G = O(d)$ with coefficients of bitsize at most τ_G. Suppose that g_2 does not vanish at any of the roots $\gamma_1, \ldots, \gamma_d$ of f. Then, for any given positive integer L, we can compute L-bit approximations of all values $y_i = G(\gamma_i)$ using a number of bit operations bounded by $\widetilde{O}_B(d^3 + d^2(\tau + \tau_G) + dL)$.*

Proof. According to Lemma 53, it holds that $|g_2(\gamma_j)| = 2^{-\widetilde{O}(d(\tau+\tau_G))}$ for all j. Now, in a first step, we compute L'-approximations $\tilde{y}_{2,j}$ of all $y_{2,j} = g_2(\gamma_j)$ for $L' = 1, 2, 4, \ldots$ until $|\tilde{y}_{2,j}| > 2^{-L'+1}$, and thus $2|\tilde{y}_{2,j}| > |y_{2,j}| > |\tilde{y}_{2,j}|/2$. Notice that we succeed in doing so for an $L' = L'_0$ in $\widetilde{O}(d(\tau + \tau_G))$. Then, for an $L' \geqslant L'_0$, we can compute L'-approximations $\tilde{y}_{1,j} = 2^{-L'-2} \cdot (m_{1,j} + i \cdot n_{1,j})$ and $\tilde{y}_{2,j} =$

$2^{-L'-2}\cdot(m_{2,j}+i\cdot n_{2,j})$ of the values $y_{1,j}=g_1(\gamma_j)$ and $y_{2,j}$, respectively, with $m_{1,j},m_{2,j},n_{1,j},n_{2,j}\in\mathbb{Z}$. Notice that each of the latter integers has bitsize $\widetilde{O}(d(\tau+\tau_G))$ as $|g_1(\gamma_j)|,|g_2(\gamma_j)|\leqslant(d_G+1)\cdot 2^{\tau_G}\cdot M(\gamma_j)^{d_G}\leqslant 2^{O(\log d+\tau_G+d\tau)}$ for all j. Hence, we conclude that

$$\left|\frac{\tilde{y}_{1,j}}{\tilde{y}_{2,j}}-G(x_j)\right|=\frac{|\tilde{y}_{1,j}\cdot y_{2,j}-y_{1,j}\cdot\tilde{y}_{2,j}|}{|y_{2,j}\cdot\tilde{y}_{2,j}|}\leqslant\frac{|\tilde{y}_{1,j}-y_{1,j}|\cdot|y_{2,j}|+|y_{1,j}|\cdot|y_{2,j}-\tilde{y}_{2,j}|}{|y_{2,j}\cdot\tilde{y}_{2,j}|}$$
$$\leqslant 2^{-L'}\cdot\frac{4}{|\tilde{y}_{2,j}|^2}\cdot(d_G+1)\cdot 2^{\tau_G}\cdot M(\gamma_j)^{d_G}=2^{-L'}\cdot 2^{\widetilde{O}(d(\tau+\tau_G))}.$$

Notice that the above bound on the approximation error is explicit (i.e. computable). Thus, we can directly estimate the error from the given values L', $|\tilde{y}_{2,j}|$, d_G, τ_G, and $M(\gamma_j)$. Hence, we may consider $L'=L'_0,L'_0+2,L'_0+4,\ldots$ until we can guarantee that $\left|\frac{\tilde{y}_{1,j}}{\tilde{y}_{2,j}}-G(x_j)\right|<2^{-L-2}$. For this, we need to increase L' at most $O(\log(d(\tau+\tau_G)))$ many times, and we succeed for an L' in $\widetilde{O}(L+d(\tau+\tau_G))$. Then, we approximate each fraction $\frac{\tilde{y}_{1,j}}{\tilde{y}_{2,j}}=\frac{m_{1,j}+i\cdot n_{1,j}}{m_{2,j}+i\cdot n_{2,j}}$ by a corresponding $(L+1)$-bit approximation to obtain an L-bit approximation of $G(x_j)$. Due to Lemma 56, the total bit complexity for computing the fractions $\frac{\tilde{y}_{1,j}}{\tilde{y}_{2,j}}$ is in $\widetilde{O}_B(d^3+d^2\tau+d\,(L+d(\tau+\tau_G)+\tau_G))=\widetilde{O}_B(d^3+d^2\tau+d^2\tau_G+dL)$, whereas the total bit complexity for computing the $(L+1)$-bit approximations of the fractions $\frac{\tilde{y}_{1,j}}{\tilde{y}_{2,j}}$ is in $\widetilde{O}_B(d^2(\tau+\tau_G)+dL)$. Indeed, using fast integer division, computing an L-bit approximation from a rational has a bit complexity that is softly linear in L and the bitsize of the rational. □

7.2 Isolating boxes

We now give a method for computing disjoint isolating boxes for the solutions $\sigma\in\mathbb{C}^2$ of a zero-dimensional system $P=Q=0$, where $P,Q\in\mathbb{Z}[x,y]$ are coprime polynomials of total degree at most d with integer coefficients of bitsize at most τ. More specifically, for a given L, we first compute L-bit approximations[8] $\tilde{\sigma}_{i,j}$ of the solutions $\sigma_{i,j}=(x_{i,j},y_{i,j})$, $1\leqslant j\leqslant d_i=\deg f_i$, of each factor $\mathrm{RUR}_i=(f_i,f_{i,1},f_{i,x},f_{i,y})$ in the RUR decomposition $(\mathrm{RUR}_i)_{i\leqslant d}$ of $\{P,Q\}$ as computed by Algorithm 6 or 6'. This is achieved by first computing sufficiently small isolating disks for the roots $\gamma_{i,j}$ of the univariate polynomial $f_i\in\mathbb{Z}[x]$ in RUR_i, and then evaluating the fractions $\frac{f_{i,x}}{f_{i,1}}$ and $\frac{f_{i,y}}{f_{i,1}}$ at the roots $\gamma_{i,j}$ to an absolute error less than 2^{-L}. From the corresponding L-bit approximations $\tilde{x}_{i,j}$ and $\tilde{y}_{i,j}$, we can then derive boxes $B_{i,j}=B(\tilde{\sigma}_{i,j})=B(\tilde{x}_{i,j})\times B(\tilde{y}_{i,j})\subset\mathbb{C}^2$ of width 2^{-L+1} containing all solutions of RUR_i; see (7) for the definition of $B(\tilde{x}_{i,j})$ and $B(\tilde{y}_{i,j})$. If, for all i and j, the boxes $B_{i,j}$ do not overlap, then they are already isolating for the solutions of $P=Q=0$. Otherwise, we have to increase L until the boxes do not overlap. We give details in Algorithm 7.

In order to bound the complexity of the above approach, we first need to derive bounds on the separations of the solutions $\sigma_{i,j}=(x_{i,j},y_{i,j})$ of the factor RUR_i. In addition, we derive amortized bounds on the separations of all solutions of the system $P=Q=0$.

Lemma 58. *Let $P,Q\in\mathbb{Z}[x,y]$ and $\mathrm{RUR}_i=(f_i,f_{i,1},f_{i,x},f_{i,y})$, $i\in\mathcal{I}$, be as defined in the input of Algorithm 7 and let d_i and τ_i be the maximum degree and bitsize of the polynomials in RUR_i. In addition, let $I_i\supseteq I=\langle P,Q\rangle$ be the ideal corresponding to RUR_i and $V(I_i)$ be the corresponding set of solutions. Then,*

(a) $\sum_{\sigma\in V(I_i)}\log M(\mathrm{sep}(\sigma,I_i)^{-1})=\widetilde{O}(d_i\tau_i)=\widetilde{O}(d_i(d^2+d\tau))=\widetilde{O}(d^4+d^3\tau)$,

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

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

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

Output: Isolating boxes for all solutions of $P = Q = 0$.

$f = \prod_{i\in\mathcal{I}} f_i$
Compute isolating disks $D_\gamma \subset \mathbb{C}$ for all complex roots γ of f.
$S = \{(\gamma, \gamma') \mid \gamma \text{ and } \gamma' \text{ distinct roots of } f\}$
$L = 1$
repeat
  $L = 2L$
  for $i \in \mathcal{I}$ **do**
    For all roots γ of f_i, compute L-bit approximations $\tilde{\sigma}_{\gamma,x}$ and $\tilde{\sigma}_{\gamma,y}$ of $\sigma_{\gamma,x} = \frac{f_{i,x}(\gamma)}{f_{i,1}(\gamma)}$ and $\sigma_{\gamma,y} = \frac{f_{i,y}(\gamma)}{f_{i,1}(\gamma)}$, respectively (Lemma 57).
until for all pairs $(\gamma, \gamma') \in S$, $|\tilde{\sigma}_{\gamma,x} - \tilde{\sigma}_{\gamma',x}| > 2^{-L+2}$ or $|\tilde{\sigma}_{\gamma,y} - \tilde{\sigma}_{\gamma',y}| > 2^{-L+2}$
return $\{B(\tilde{\sigma}_{\gamma,x}) \times B(\tilde{\sigma}_{\gamma,y}) \mid \gamma \text{ root of } f\}$

(b) $\log M(|\sigma|) = \widetilde{O}(d\tau)$ *for all* $\sigma = (\sigma_x, \sigma_y) \in V(I_i)$, *where* $|\sigma| = \max(|\sigma_x|, |\sigma_y|)$,

(c) $\sum_{\sigma\in V(I)} \log M(\mathrm{sep}(\sigma, I)^{-1}) = \widetilde{O}(d^4 + d^3\tau)$.

Proof. Let $(x, y) \mapsto x + ay$ be the separating form for $\{P, Q\}$ with $a \in \mathbb{Z}$ of bitsize $O(\log d)$, as computed by Algorithm 6 or 6' as part of the input of Algorithm 7. This separating form defines a one-to-one mapping from the set of solutions $\sigma \in V(I_i)$ to the set of roots γ of f_i. Now let $\sigma = (\sigma_x, \sigma_y) \in V(I_i)$ and $\sigma' = (\sigma'_x, \sigma'_y) \in V(I_i)$ be two solutions with $\mathrm{sep}(\sigma, I_i) = |\sigma - \sigma'|$, and let γ and γ' be the corresponding roots of f_i. Then, we have $\mathrm{sep}(\gamma, f_i) \leqslant |\sigma_x - \sigma'_x| + |a| \cdot |\sigma_y - \sigma'_y| \leqslant (|a| + 1)\,\mathrm{sep}(\sigma, I_i)$, or equivalently $\mathrm{sep}(\sigma, I_i)^{-1} \leqslant (|a|+1)\,\mathrm{sep}(\gamma, f_i)^{-1}$. We thus have $\log M(\mathrm{sep}(\sigma, I_i)^{-1}) \leqslant \log(|a|+1) + \log M(\mathrm{sep}(\gamma, f_i)^{-1})$. On the other hand, f_i is squarefree since it is the first polynomial of the RUR of a radical ideal (see Algorithm 6 or 6'). Thus, Lemma 52 yields that $\prod_{\{\gamma \text{ root of } f_i\}} \min(1, \mathrm{sep}(\gamma, f_i))^{-1} = 2^{\widetilde{O}(d_i\tau_i)}$ and thus $\sum_{\{\gamma \text{ root of } f_i\}} \log M(\mathrm{sep}(\gamma, f_i)^{-1}) = \widetilde{O}(d_i\tau_i)$. Part (a) follows directly since a has bitsize $O(\log d)$ and, by Theorem 35, $d_i \leqslant d^2$ and $\tau_i = \widetilde{O}(d^2 + d\tau)$.

Part (b) follows directly from Lemma 51 since each coordinate of a solution σ is a root of either the resultant polynomial $\mathrm{Sres}_{x,0}(P, Q) \in \mathbb{Z}[y]$ or $\mathrm{Sres}_{y,0}(P, Q) \in \mathbb{Z}[x]$, and both of these polynomials have integer coefficients of bitsize $\widetilde{O}(d\tau)$ by Lemma 3. For part (c), notice that, by definition of RUR decompositions (Definition 34), the roots of $f = \prod_i f_i$ are exactly the images of the solutions of $\{P, Q\}$ through the mapping $(x, y) \mapsto x + ay$. The degree of f is thus at most d^2. Furthermore, the f_i are monic (by Definition 32), thus f is monic and the bitsize of its coefficients is at most that of the resultant of the sheared polynomials $P(t - ay, y)$ and $Q(t - ay, y)$ with respect to y, whose bitsize is in $\widetilde{O}(d^2 + d\tau)$ (see e.g. [BLPR15, Lemma 7]). Hence, the same argument as for the proof of part (a) yields that $\mathrm{sep}(\gamma, f) \leqslant (|a| + 1)\,\mathrm{sep}(\sigma, I)$ and then the result. □

The following theorem analyzes the complexity of the isolation of a system $\{P, Q\}$ from a RUR decomposition as computed in Section 6.

Theorem 59. *Let* $P, Q \in \mathbb{Z}[x, y]$ *be coprime polynomials of degree at most* d *with integer coefficients of bitsize at most* τ. *Algorithm 7 computes isolating boxes for all complex solutions of* $P = Q = 0$

using $\widetilde{O}_B(d^6 + d^5\tau)$ *bit operations.*

Proof. As argued at the end of the proof of Lemma 58, f is a polynomial of degree at most d^2 with coefficients of bitsize $\widetilde{O}(d^2 + d\tau)$, and thus the bit complexity of Step 2 in Algorithm 7 is $\widetilde{O}_B(d^6 + d^5\tau)$ (Lemma 54). In addition, the degree of all polynomials $f_{i,y}$ and $f_{i,1}$ is at most the degree d_i of f_i (Definition 32), and the bitsize of their coefficients is in $\widetilde{O}(d^2 + d\tau)$ (Theorem 35). According to Lemma 58(c), the distance between any two solutions of $P = Q = 0$ is lower bounded by $2^{-\widetilde{O}(d^4+d^3\tau)}$, which implies that Algorithm 7 terminates with L in $\widetilde{O}(d^4 + d^3\tau)$. We also have that the loop in Line 5 is executed $\log L = O(\log(d\tau))$ times and thus, ignoring the polylogarithmic factors, it is sufficient to study the complexity of the last call to this loop. From Lemma 57, the bit complexity of computing the L-bit approximations $\tilde{\sigma}_{\gamma,x}$ and $\tilde{\sigma}_{\gamma,y}$ for all roots of the factor f_i in Step 8 is in $\widetilde{O}_B(d_i^3 + d_i^2(d^2 + d\tau) + d_i(d^4 + d^3\tau)) = \widetilde{O}_B(d_i^3 + d_i(d^4 + d^3\tau))$. Summing over all i yields the bound $\widetilde{O}_B(d^6 + d^5\tau)$ for the total bit complexity of this step, since the sum of all d_i is at most d^2. In Step 9, consider a fixed pair (γ, γ') of distinct roots of f and let σ and σ' be the corresponding solutions in $I = \langle P, Q\rangle$. From Definition (7) of the box associated to an L-bit approximation, the inequalities $|\tilde{\sigma}_{\gamma,x} - \tilde{\sigma}_{\gamma',x}| > 2^{-L+2}$ or $|\tilde{\sigma}_{\gamma,y} - \tilde{\sigma}_{\gamma',y}| > 2^{-L+2}$ imply that the boxes $B(\tilde{\sigma}_{\gamma,x}) \times B(\tilde{\sigma}_{\gamma,y})$ and $B(\tilde{\sigma}_{\gamma',x}) \times B(\tilde{\sigma}_{\gamma',y})$ do not overlap, which implies the correctness of the algorithm. Testing these inequalities can be done in $O_B(\log(M(|\sigma|) + M(|\sigma'|)) + \log M(|\sigma - \sigma'|^{-1}))$ bit operations (where $|\sigma| = \max(|\sigma_x|, |\sigma_y|)$) because, for each comparison, the first term bounds the number of bits before the binary point, and in the case where these bits coincide, the second term bounds the number of bits after the binary point that need to be considered. Notice that the sum of $\log M(|\sigma - \sigma'|^{-1})$ over the $O(d^4)$ pairs (σ, σ') is at most $d^2 \sum_{\sigma \in V(I)} \log M(\mathrm{sep}(\sigma, I)^{-1})$. Thus, summing over the $O(d^4)$ pairs and using Lemma 58 yields the bound $\widetilde{O}_B(d^6 + d^5\tau)$ for the total bit complexity of Step 9, which concludes the proof. □

Remark 60. *If we set $f = f_i$ in Step 1 of Algorithm 7, the algorithm only computes isolating boxes for the solutions of the specific* RUR_i*. Following the proof of Theorem 59, it is straightforward to prove that the bit complexity of the algorithm then decreases to $\widetilde{O}_B(d_i^2(d^2 + d\tau))$ where the degree d_i of* RUR_i *can be much smaller than d^2.*

Remark 61. *In order to isolate only the real solutions of $P = Q = 0$, it suffices to iterate in Algorithm 7 over the real roots of f since the separating form $(x, y) \mapsto x + ay$ is a one-to-one mapping between the real solutions of $P = Q = 0$ and the real roots of f (see Proposition 33). Note that, in this case, it is preferable to consider a dedicated real root isolation method in Step 2 for computing the real roots of f.*

8 Conclusion

We have studied the problem of solving a bivariate system of two polynomials of degree bounded by d and bitsize bounded by τ via a combination of triangular decomposition and RUR. We have designed algorithms of worst-case complexity $\widetilde{O}_B(d^6 + d^5\tau)$ for all the steps: (i) finding a separating linear form $x + ay$ with $a \in \{0, \ldots, 2d^4\}$, (ii) computing a RUR decomposition and (iii) computing isolating boxes of the solutions. We have also proposed Las Vegas algorithms of expected complexity $\widetilde{O}_B(d^5 + d^4\tau)$ for Steps (i) and (ii).

Since the size of the RURs output is upper bounded by $\widetilde{O}(d^4 + d^3\tau)$ (Corollary 36), one natural question is whether it is possible to improve our algorithms. Consider Step (i) or actually the simpler problem of checking whether a linear form is separating in the regular case, that is when the system has no multiple solutions. In this simpler case, the problem amounts to shearing the

coordinate system, then computing a resultant and checking whether it is squarefree. The best known complexities for that problem are $\widetilde{O}_B(d^5 + d^4\tau)$ in the Las Vegas setting and $\widetilde{O}_B(d^6 + d^5\tau)$ in the worst case. Indeed, after shearing, the input polynomials have degree d and bitsize $\widetilde{O}(d+\tau)$ (see e.g. [BLPR15, Lemma 7]). The best known complexity for computing their resultant is $\widetilde{O}_B(d^4(d+\tau))$ in the worst case (Lemma 3) and, up to our knowledge, there is no better complexity even in the Las Vegas setting. Finally, the squarefreeness test amounts to a gcd computation and the best-known complexities are $\widetilde{O}_B(d^4 + d^3\tau)$ in the Las Vegas setting and $\widetilde{O}_B(d^6 + d^5\tau)$ in the worst case [vzGG13, corollary 11.14]. Since the resulting bounds match those of our algorithms, it is likely that improving our Las Vegas and deterministic algorithms for computing a separating linear form would require improving the complexities of computing resultants of gcds. On the other hand, considering Step (iii), we observe that our worst-case bound $\widetilde{O}_B(d^6 + d^5\tau)$ also matches the best known complexity for the isolation of the roots of a given polynomial of degree and bitsize comparable to those of the resultant of the input polynomials [MSW15, Theorem 5]. All these observations hint that it is likely difficult to improve the Las Vegas and worst-case complexities of our sequence of algorithms for solving bivariate systems.

Finally, we note that, for computing a separating linear form of an *arbitrary* system $\{P, Q\}$, the algorithm presented here is likely to be purely theoretical because (i) considering the system $\{PQ, \frac{\partial PQ}{\partial y}\}$ instead $\{P, Q\}$ essentially doubles the degree of the input polynomials, and (ii) the shearing of the coordinate systems, done to avoid vertical asymptotes, spoils the sparsity of the coefficients and increases their bitsize, which is not efficient in practice. However, for the problem of computing the critical points of a curve, there is some hope that our algorithm can be efficient not only in theory but also in practice.

References

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

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Ann. of Math*, 160(2):781–793, 2004.

[BLP+14] Y. Bouzidi, S. Lazard, M. Pouget, G. Moroz, and F. Rouillier. Improved algorithm for computing separating linear forms for bivariate systems. In *Proceedings of the 39th International Symposium on International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, Kobe, Japan, July 2014. ACM.

[BLPR15] Y. Bouzidi, S. Lazard, M. Pouget, and F. Rouillier. Separating linear forms and rational univariate representations of bivariate systems. *J. Symb. Comput.*, pages 84–119, 2015.

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

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

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

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

[EK03] M. El Kahoui. An elementary approach to subresultants theory. *J. Symb. Comput.*, 35(3):281–292, 2003.

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

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

[KS15a] M. Kerber and M. Sagraloff. Root refinement for real polynomials using quadratic interval refinement. *Journal of Computational and Applied Mathematics*, 280(0):377–395, 2015.

[KS15b] A. Kobel and M. Sagraloff. On the complexity of computing with planar algebraic curves. *Journal of Complexity*, 31(2):206–236, 2015.

[LMMRS11] X. Li, M. Moreno Maza, R. Rasheed, and É. Schost. The modpn library: Bringing fast polynomial arithmetic into Maple. *J. Symb. Comput.*, 46(7):841–858, 2011.

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

[MB74] R. Moenck and A. Borodin. Fast modular transforms. *Journal of Computer and System Sciences*, 8:366–386, 1974.

[MS15] E. Mehrabi and É. Schost. A quasi-optimal monte carlo algorithm for the symbolic solution of polynomial systems in $\mathbb{Z}[x, y]$. Manuscript, 2015.

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

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

[Rei97] D. Reischert. Asymptotically fast computation of subresultants. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, ISSAC'97, pages 233–240, New York, NY, USA, 1997. ACM.

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

[SM16] M. Sagraloff and K. Mehlhorn. Computing real roots of real polynomials. *Journal of Symbolic Computation*, 73:46–86, 2016.

[SY11] M. Sagraloff and C. K. Yap. A simple but exact and efficient algorithm for complex root isolation. In *Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, ISSAC '11, pages 353–360, New York, NY, USA, 2011. ACM.

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

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