Bivariate Triangular Decompositions in the Presence of Asymptotes

Sylvain Lazard, Marc Pouget, Fabrice Rouillier

Bivariate Triangular Decompositions in the Presence of Asymptotes

Sylvain Lazard*† Marc Pouget*† Fabrice Rouillier*‡

September 17, 2015

Abstract

Given two coprime polynomials P and Q in $\mathbb{Z}[x, y]$ of degree at most d and coefficients of bitsize at most τ, we address the problem of computing a triangular decomposition $\{(U_i(x), V_i(x, y))\}_{i\in\mathcal{I}}$ of the system $\{P, Q\}$.

The state-of-the-art worst-case bit complexity for computing such triangular decompositions when the curves defined by the input polynomials do not have common vertical asymptotes is in $\widetilde{O}_B(d^6+d^5\tau)$ [BLM+15, Proposition 16], where $\widetilde{O}$ refers to the complexity where polylogarithmic factors are omitted and O_B refers to the bit complexity.

We show that the same worst-case bit complexity can be achieved even when the curves defined by the input polynomials may have common vertical asymptotes. We actually present a refined bit complexity in $\widetilde{O}_B(d_x^3 d_y^3 + (d_x^2 d_y^3 + d_x d_y^4)\tau)$ where d_x and d_y bound the degrees of P and Q in x and y, respectively. We also prove that the total bitsize of the decomposition is in $\widetilde{O}((d_x^2 d_y^3 + d_x d_y^4)\tau)$.

1 Introduction

Computing triangular decompositions of algebraic systems is a well-known problem. In the special case of bivariate systems, a classical algorithm using subresultant sequences was first introduced by González-Vega and El Kahoui in the context of computing the topology of curves [GVEK96]. This algorithm is based on a direct consequence of the specialization property of subresultants and of the gap structure theorem, which implies the following (see Theorem 3): given two polynomials $P = \sum_{i=0}^{p} a_i(x)y^i$ and $Q = \sum_{i=0}^{q} b_i(x)y^i$ in $\mathbb{Z}[x, y]$ and $\alpha \in \mathbb{R}$ such that the leading coefficients $a_p(\alpha)$ and $b_q(\alpha)$ do not both vanish, then the first (with respect to increasing i) nonzero subresultant $\mathrm{Sres}_{y,i}(P, Q)(\alpha, y)$ is of degree i and is equal to the gcd of $P(\alpha, y)$ and $Q(\alpha, y)$. Note that values α such that $a_p(\alpha)$ and $b_q(\alpha)$ both vanish are exactly the x-coordinates of the common vertical asymptotes of the curves defined by P and Q, which we refer to as the common vertical asymptotes of the polynomials, for simplicity. Hence, when P and Q do not have common vertical asymptotes, the gap structure theorem induces a decomposition of the system $\{P, Q\}$ into triangular subsystems $\{U_i(x), \mathrm{Sres}_{y,i}(P, Q)(x, y)\}$ where the product of the U_i is the (squarefree part of the) resultant of P and Q with respect to y.

If the input polynomials have degree at most d and coefficients of bitsize at most τ, the worst-case bit complexity of this algorithm was initially analyzed in $\widetilde{O}_B(d^{16}+d^{14}\tau^2)$ [GVEK96]. The complexity analysis was later improved to $\widetilde{O}_B(d^7 + d^6\tau)$ [DET09, §4.2] and more recently to $\widetilde{O}_B(d^6 + d^5\tau)$ by considering amortized bounds on the degrees and bitsizes of factors of the resultant [BLM+15,

*Inria, France. `Firstname.Name@inria.fr`

†LORIA laboratory (Inria, CNRS, Université de Lorraine), Nancy, France.

‡Institut de Mathématiques de Jussieu, Paris, France.

Proposition 16]. No better complexity is known for computing triangular decompositions, even in the expected Las-Vegas or Monte-Carlo settings and even in the absence of common vertical asymptotes.

In the general case when P and Q (may) admit common vertical asymptotes, the natural solution for computing a (full) triangular decomposition is to first use González-Vega and El Kahoui algorithm to compute the triangular decomposition of the solutions of $\{P, Q\}$ that do not lie on common vertical asymptotes (this can be done by removing from the resultant of P and Q the solutions corresponding to these asymptotes, i.e., $\gcd(a_p, b_q)$). Then, the triangular decomposition algorithm is called recursively on P and Q reduced modulo $\gcd(a_p, b_q)$. This natural approach was presented by Li et al. [LMMRS11].[1] The drawback of this approach is that the number of recursive calls may be linear in the minimum of the degrees in x and y of the input polynomials (it may happen that only one vertical asymptote is "handled" at each recursive call) and that the bitsize of the coefficients of the reduction of P and Q increases at each recursive call. However, Li et al. did not provide a complexity analysis of their algorithm.

We present here a simple variation on this natural algorithm where, instead of considering P and Q modulo $\gcd(a_p, b_q)$ at the first recursive call (and similarly for the other calls), we simply remove the leading terms $a_p y^p$ and $b_q y^q$ of P and Q. The number of recursive calls may still be linear in d but we show that, with this simple modification, the bit complexity of the overall recursive algorithm is the same as the bit complexity of the non-recursive algorithm (with no vertical asymptotes), that is $\widetilde{O}_B(d^6 + d^5\tau)$. More precisely, we prove a worst-case bit complexity in $\widetilde{O}_B(d_x^3 d_y^3 + (d_x^2 d_y^3 + d_x d_y^4)\tau)$ where d_x and d_y bound the degrees of P and Q in x and y, respectively. We also prove that the total bitsize of the decomposition is in $\widetilde{O}((d_x^2 d_y^3 + d_x d_y^4)\tau)$. This implies in particular that, unless improving this upper bound, there is not much room for improving the bit complexity of the computation of the triangular decomposition. This also shows that, when there is a disparity between degrees d_x and d_y, the ordering of variables in the triangular decomposition impacts the complexity of the algorithm and of the output.

It is worthwhile to mention that in the general context of solving systems, one standard approach is to shear the coordinate system and to compute a triangular decomposition of the sheared system. This approach does not solve the given problem of computing a triangular decomposition of the input system since it computes a triangular decomposition of another system. Nevertheless, this approach, which naturally gets rid of vertical asymptotes, is theoretically straightforward, easy to implement, and its overall bit complexity is still in $\widetilde{O}_B(d^6 + d^5\tau)$ (see e.g., [BLPR15, Lemma 7]). However, it has the practical drawback that a shear $(x, y) \mapsto (x + ay, y)$ on polynomial $P = \sum_{i=0}^{d_y} a_i(x)y^i$ does not preserve its sparsity, increases the bitsize of its coefficients from τ up to $\tau + \widetilde{O}(d_x + d_y)$ and increases its degree in y from d_y up to $d_x + d_y$. Since the bit complexity of the triangular decomposition algorithm is quartic in d_y (even in the absence of asymptotes; see Lemma 9), one should expect that shearing dramatically impacts the practical efficiency, which is observed in experiments. In addition, on a theoretical basis, if d_y is small compared to d_x then the overall bit complexity may drastically increase; for instance, in the extreme case where d_y is initially in $O(1)$, the overall worst-case complexity goes from $\widetilde{O}_B(d_x^3 + d_x^2\tau)$ to $\widetilde{O}_B(d_x^6 + d_x^5\tau)$. Still, it should be noted that, when complexities are expressed in terms of the total degree d, shearing leads to theoretically more efficient probabilistic algorithms for solving the system, both in the Las-Vegas setting [BLM+15] and in the Monte-Carlo setting [MS15].

[1] Note that, in [LMMRS11], the reduction of P and Q modulo $\gcd(a_p, b_q)$ is understood from context because it is not clearly specified: P and Q are replaced by their "reductums" with respect to y but no definition of reductums is given. Note also that their algorithm misses the fact that, during the recursion, the reduced versions of P and Q may not define a zero-dimensional system and also that they may be both univariate.

In the next section, we recall standard definitions and results about multiplicities, subresultant sequences, and gcds. We then present and analyze our triangular decomposition algorithm in Section 3.

2 Notation and preliminaries

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 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. In this paper, most complexities are expressed in terms of d_x and d_y, the maximum degrees in x and in y of $P, Q \in \mathbb{Z}[x, y]$, and in τ, their maximum bitsize. We also denote by d the maximum of d_x and d_y.

For any polynomial $P \in \mathbb{D}[x]$ where $\mathbb{D}$ denotes a unique factorization domain, let $\mathrm{Lc}_x(P)$ denote its leading coefficient with respect to the variable x, $\deg_x(P)$ its degree with respect to x (or simply $\deg(P)$ when P is univariate), and $\overline{P}$ its squarefree part. A polynomial P that vanishes identically is denoted by $P \equiv 0$. In the following, unless specified otherwise, the solutions of the considered polynomials are always considered in the algebraic closure of the coefficient ring. Consequently, a polynomial system is called zero-dimensional if its set of solutions over that algebraic closure is finite.

In the rest of this section, we recall standard definitions and results about multiplicities, subresultant sequences, and gcds.

Multiplicities. Geometrically, the notion of multiplicity of intersection of two regular curves is intuitive. If the intersection is transverse, the multiplicity is one; otherwise, it is greater than one and it measures the level of degeneracy of the tangential contact between the curves. Defining the multiplicity of the intersection of two curves at a point that is singular for one of them (or possibly both) is more involved and an abstract and general concept of multiplicity in an ideal is needed. We recall this classical, though non-trivial, notion. We also introduce a simple notion of *multiplicity in fibers* that will be output by our solver and that are relevant for the topology of a plane curve (see e.g. [SW05]). Let $\mathbb{F}$ be a field and $\overline{\mathbb{F}}$ be its algebraic closure.

Definition 1. *Let I be an ideal of $\mathbb{F}[x, y]$. To each zero (α, β) of I corresponds a local ring $(\overline{\mathbb{F}}[x, y]/I)_{(\alpha,\beta)}$ obtained by localizing the ring $\overline{\mathbb{F}}[x, y]/I$ at the maximal ideal $\langle x - \alpha, y - \beta\rangle$. When this local ring is finite dimensional as $\overline{\mathbb{F}}$-vector space, we say that (α, β) is an isolated zero of I and this dimension is called the* **multiplicity** *of (α, β) as a zero of I [CLO05, §4.2].*

We call the fiber of a point $p = (\alpha, \beta)$ the vertical line of equation $x = \alpha$. The **multiplicity of p in its fiber** *with respect to a system of polynomials $\{P, Q\}$ in $\mathbb{F}[x, y]$ is the multiplicity of β in the univariate polynomial* $\gcd(P(\alpha, y),\ Q(\alpha, y))$.[2] *(This multiplicity is zero if P or Q does not vanish at p.)*

Subresultant sequences. 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 $\det(M_m)y^{n-m} + \det(M_{m+1})y^{n-(m+1)} + \ldots + \det(M_n)$.

[2]The gcd is naturally considered over $\mathbb{F}(\alpha)[y]$, the ring of polynomials in y with coefficients in the field extension of $\mathbb{F}$ by α.

Let $P=\sum_{i=0}^{p} a_i y^i$ and $Q=\sum_{i=0}^{q} b_i y^i$ be two polynomials in $\mathbb{D}[y]$ (where $\mathbb{D}$ is a unique factorization domain such as $\mathbb{Q}[x]$) and assume without loss of generality that $p \geqslant q$. The Sylvester matrix of P and Q, $\mathrm{Sylv}(P,Q)$ 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 $y^{p+q-1},\ldots,y,1$.

Definition 2. *([EK03, §3]). For $i=0,\ldots,\min(q,p-1)$, let $\mathrm{Sylv}_i(P,Q)$ be the $(p+q-2i)\times(p+q-i)$ matrix obtained from $\mathrm{Sylv}(P,Q)$ 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.*

For $i=0,\ldots,\min(q,p-1)$, the i-th polynomial subresultant of P and Q, denoted by $\mathrm{Sres}_{y,i}(P,Q)$ is the polynomial determinant of $\mathrm{Sylv}_i(P,Q)$.

For practical consideration, when $q=p$, we define the q-th polynomial subresultant of P and Q as Q.[3] $\mathrm{Sres}_{y,i}(P,Q)$ has degree at most i in y, and the coefficient of its monomial of degree i in y, denoted by $\mathrm{sres}_{y,i}(P,Q)$ or sres_i, is called the i-th *principal subresultant coefficient*. Note that $\mathrm{Sres}_{y,0}(P,Q)=\mathrm{sres}_{y,0}(P,Q)$ is the *resultant* of P and Q with respect to y, which we also denote by $\mathrm{Res}_y(P,Q)$. Note also that the subresultants of P and Q are equal to either 0 or to polynomials in the remainder sequence of P and Q in Euclid's algorithm (up to multiplicative factors in $\mathbb{D}$) [BPR06, §8.3.3 & Cor. 8.32].

Consider now two bivariate polynomials with coefficients in $\mathbb{D}=\mathbb{Z}$: $P=\sum_{i=0}^{p} a_i(x)y^i$ and $Q=\sum_{i=0}^{q} b_i(x)y^i$ with $p \geqslant q$. The following fundamental property of subresultant sequences is instrumental in the triangular decomposition algorithms. Note that this result is often stated with the stronger assumption that *none* of the leading terms $a_p(\alpha)$ and $b_q(\alpha)$ vanish. This property is a direct consequence of the specialization property of subresultants and of the gap structure theorem; see [EK03, Lemmas 2.3, 3.1 and Corollary 5.1] for a proof.

Theorem 3. *For any α such that $a_p(\alpha)$ and $b_q(\alpha)$ do not both vanish, the first $\mathrm{Sres}_{y,k}(P,Q)(\alpha,y)$ (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 a nonzero constant in the fraction field of $\mathbb{D}(\alpha)$).*

Lemma 4 ([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_x.*

- *The coefficients of $\mathrm{Sres}_{y,i}(P,Q)$ have bitsize in $\widetilde{O}(d_y\tau)$.*
- *The degree in x_j of $\mathrm{Sres}_{y,i}(P,Q)$ is at most $2d_x(d_y-i)$.*
- *For any $i\in\{0,\ldots,\min(\deg_y(P),\deg_y(Q))\}$, $\mathrm{Sres}_{y,i}(P,Q)$ can be computed in $\widetilde{O}(d_x^n d_y^{n+1})$ arithmetic operations and $\widetilde{O}_B(d_x^n d_y^{n+2}\tau)$ bit operations. These complexities also hold for the computation of the sequence of principal subresultant coefficients $\mathrm{sres}_i(P,Q)$.*[4]

Gcds. We often consider the gcd of two univariate polynomials P and Q in $\mathbb{Z}[x]$ and the gcd-free part of P with respect to Q, that is, $P/\gcd(P,Q)$. Note that, when $Q=P'$, the latter is the squarefree part $\overline{P}$. When P and Q have degree at most d and bitsize at most τ, their gcd and gcd-free parts can be computed with a bit complexity in $\widetilde{O}_B(d^2\tau)$ [BPR06, Remark 10.19]. However, we will need a finer complexity in the case of two polynomials with different degrees and bitsizes.

[3] 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 Theorem 3 holds when $P(\alpha,y)$ and $Q(\alpha,y)$ have same degree and are multiple of one another.

[4] 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.

Lemma 5 ([LR01][5]). *Let P and Q be two polynomials in $\mathbb{Z}[x]$ of degrees p and q and of bitsizes τ_P and τ_Q, respectively. A gcd of P and Q of bitsize $O(\min(p+\tau_P, q+\tau_Q))$ in $\mathbb{Z}[x]$, can be computed in $\widetilde{O}_B(\max(p,q)(p\tau_Q+q\tau_P))$ bit operations. A gcd-free part of P with respect to Q, of bitsize $O(p+\tau_P)$ in $\mathbb{Z}[x]$, can be computed in the same bit complexity.*

3 Triangular decomposition

We present here our algorithm that decomposes a zero-dimensional system $\{P, Q\}$ of polynomials in $\mathbb{Z}[x, y]$ into a set of regular triangular systems of the form $\{U(x), V(x, y)\}$. Recall that such a system is said regular if U and $\mathrm{Lc}_y(V)$ are coprime. Algorithm 2 is the main algorithm, which recursively calls Algorithm 1, the latter being in essence that of Gonzalez-Vega and El Kahoui [GVEK96]. For clarity and completeness, we briefly describe this latter algorithm with an emphasis on the main differences with that of Gonzalez-Vega and El Kahoui, and give a succinct proof of correctness. We then describe Algorithm 2, prove its correctness in Lemmas 6 and 7 and analyze its complexity in Proposition 10.

Algorithm 1: Triangular decomposition of $\{P, Q\}$ away from their common vertical asymptotes and such that A vanishes. Algorithm 1 takes as input $P, Q \in \mathbb{Z}[x, y]$ and a univariate polynomial $A \in \mathbb{Z}[x]$ such that system $\{P, Q, A\}$ is zero dimensional and it computes a set of triangular systems whose solutions are the solutions of $\{P, Q, A\}$ that do not lie on a common vertical asymptote of the curves defined by P and Q. Considering the calls to Algorithm 1 made by Algorithm 2, Algorithm 1 will first run with $A \equiv 0$ and compute a triangular decomposition of the solutions away from the common vertical asymptotes of P and Q; then Algorithm 1 will be called with A encoding a subset of these common vertical asymptotes and two polynomials that coincide with P and Q on these asymptotes. Algorithm 1 is essentially that of Gonzalez-Vega and El Kahoui [GVEK96] in the case where $\{P, Q\}$ is zero dimensional, P and Q do not have any common vertical asymptote, and $A \equiv 0$.

The projection onto the x-axis of the solutions of system $\{P, Q\}$ that do not lie on a common vertical asymptote of the curves defined by P and Q are exactly the roots of the resultant of P and Q with respect to y divided by the gcd of the leading coefficients of P and Q with respect to y. We actually consider the squarefree parts of these polynomials, $\overline{\mathrm{Res}_y(P, Q)}$ and $\overline{\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$, which is critical for our property on the multiplicity of the solutions in their fibers (Lemma 7). In order to restrict the set of solutions of $\{P, Q\}$ that do not lie on a common vertical asymptote to those where A vanishes, we consider the gcd of $\frac{\overline{\mathrm{Res}_y(P,Q)}}{\overline{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))}}$ with A. However, this does not work when $\mathrm{Res}_y(P, Q) \equiv 0$, that is when $\{P, Q\}$ is not zero dimensional (and in generic position). We thus consider instead $F = \frac{\overline{\mathrm{Res}_y(P,Q)}}{\overline{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))}}$ when $A \equiv 0$ and, otherwise, $F = \frac{\overline{\gcd(\mathrm{Res}_y(P,Q),A)}}{\overline{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q),A)}}$, which is equal to $\frac{\gcd(\mathrm{Res}_y(P,Q),A)}{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q),A)}$ since A is squarefree. Then, the roots of F are the projections (on x) of the solutions of $\{P, Q, A\}$ that are not on the common vertical asymptotes of P and Q.

[5] The algorithm in [LR01] uses the well-known half-gcd approach to compute any polynomial in the Sylvester-Habicht and cofactors sequence in a softly-linear number of arithmetic operations, and it exploits Hadamard's bound on determinants to bound the size of intermediate coefficients. When the two input polynomials have different degrees and bitsizes, Hadamard's bound reads as $\widetilde{O}(p\tau_Q + q\tau_P)$ instead of simply $\widetilde{O}(d\tau)$ and, similarly as in Lemma 5, the algorithm in [LR01] yields a gcd and gcd-free parts of P and Q in $\widetilde{O}_B(\max(p, q)(p\tau_Q + q\tau_P))$ bit operations. Furthermore, the gcd and gcd-free parts computed this way are in $\mathbb{Z}[x]$ with coefficients of bitsize $\widetilde{O}(p\tau_Q + q\tau_P)$, thus, dividing them by the gcd of their coefficients can be done with $\widetilde{O}_B(\max(p, q)(p\tau_Q + q\tau_P))$ bit operations and yields a gcd and gcd-free parts in $\mathbb{Z}[x]$ with minimal bitsize, which is as claimed by Mignotte's bound; see e.g. [BPR06, Corollary 10.12].

Algorithm 1 decomposes F into factors according to Theorem 3. Recall that $\mathrm{sres}_{y,i}(P,Q)$ denotes the coefficient of the monomial of degree i in y of $\mathrm{Sres}_{y,i}(P,Q)$, the i-th polynomial subresultant of P and Q with respect to y. Polynomial F is decomposed into factors F_i, $i=1,2,\ldots$, such that for any root α of F_i, $\mathrm{sres}_{y,i}(P,Q)(\alpha)$ is the first (for i increasing) non-vanishing coefficient. The algorithm then returns the set of non-trivial triangular systems $\{F_i, \mathrm{Sres}_{y,i}(P,Q)\}$ whose solutions are, by Theorem 3, those of $\{P,Q,A\}$ that are not on the common vertical asymptotes of P and Q. The triangular systems are regular by construction.

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

Lemma 6. *Given P,Q in $\mathbb{Z}[x,y]$ defining a zero-dimensional system, Algorithm 2 outputs a set of regular triangular systems, each of the form $\{U(x),V(x,y)\}$ with coefficients in $\mathbb{Z}$, whose sets of solutions are disjoint and are exactly those of $\{P,Q\}$.*

Proof. Let P_h, Q_h, A_h and B_h be the polynomials P,Q,A and B defined in Algorithm 2 when Algorithm 1 is called for the h-th time (which might be different from the h-th iteration of the loop). We have $P_1=P$, $Q_1=Q$, $A_1\equiv 0$ and $B_1=\overline{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))}$, thus the first call to Algorithm 1 returns triangular systems encoding the solutions of $\{P,Q\}$ that are not over the common vertical asymptotes of P and Q. For $h\geqslant 1$, B_h encodes the common vertical asymptotes of $P_1,Q_1,\ldots,P_h,Q_h$ and, for $h\geqslant 2$, A_h encodes the common vertical asymptotes of $P_1,Q_1,\ldots,P_{h-1},Q_{h-1}$ that are not common vertical asymptotes of P_h and Q_h.

Thus P_h coincides with P on the vertical asymptotes encoded by A_h, and similarly for Q_h. This first implies that $\{P_h,Q_h,A_h\}$ is zero-dimensional, since $\{P,Q\}$ is. Furthermore, A_h is squarefree because it is either identically equal to 0 (when $h=1$) or it divides B_{h-1}, which divides $B_1=\overline{\gcd(\mathrm{Lc}_y(P),\mathrm{Lc}_y(Q))}$. Hence $\{P_h,Q_h,A_h\}$ satisfies the requirements of Algorithm 1.

Algorithm 1 when called on P_h,Q_h,A_h returns a set of regular triangular systems whose solutions are those of $\{P_h,Q_h,A_h\}$ away from the common asymptotes of P_h and Q_h. But, for $h\geqslant 2$, A_h does not vanish on these asymptotes so the solutions are those of $\{P_h,Q_h,A_h\}$. Furthermore, P_h and Q_h coincide with P and Q when A_h vanishes, thus these solutions are also those of $\{P,Q,A_h\}$. Finally, the above property on A_h also implies that the A_h, for $h\geqslant 2$, are coprime and that their product encodes the common asymptotes of P and Q. Thus, the set of systems returned by all the calls to Algorithm 1 except the first one have sets of solutions that are disjoint and are the solutions of $\{P,Q\}$ that lie on their common asymptotes. This concludes the proof since the systems output by the first call to Algorithm 1 are those of $\{P,Q\}$ away from these asymptotes. □

We now prove that Algorithm 2 preserves the multiplicities of the solutions, in the following sense (see Definition 1).

Lemma 7. *The multiplicity of any solution in the triangular systems output by Algorithm 2 is its multiplicity in its fiber with respect to the system $\{P,Q\}$.*

Proof. Consider a solution (α,β) of a triangular system $\{U(x),V(x,y)\}$ output by Algorithm 2. This triangular system is output by Algorithm 1 called on some polynomials P_h,Q_h,A_h at the h-th call of Algorithm 1. By construction, $U(x)$ is squarefree because, in Algorithm 1, F_i divides

Algorithm 1 Triangular decomposition away from asymptotes

Input: P, Q in $\mathbb{Z}[x,y]$ and A squarefree in $\mathbb{Z}[x]$ such that system $\{P,Q,A\}$ is zero-dimensional.

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

1. **if** $A \equiv 0$ **then**
2. $R(x) = \overline{\mathrm{Res}_y(P,Q)}$, $B(x) = \overline{\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$, $F = R/B$
3. **else**
4. $R(x) = \mathrm{Res}_y(P,Q)$, $B(x) = \gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$, $F = \frac{\gcd(R,A)}{\gcd(B,A)}$
5. **if** neither P nor Q is in $\mathbb{Z}[x]$ **then**
6.   If needed, exchange P and Q so that $\deg_y(Q) \leqslant \deg_y(P)$
7.   Compute $\{\mathrm{sres}_{y,i}(P,Q)\}_{i=0,\ldots,\deg_y(Q)}$, the principal subresultant sequence of P and Q w.r.t. y
8. $G_0 = F$, $\mathcal{TD} = \emptyset$
9.   **for** $i = 1$ **to** $\deg_y(Q)$ **do**
10. $G_i = \gcd(G_{i-1}, \mathrm{sres}_{y,i}(P,Q))$
11. $F_i = G_{i-1}/G_i$
12.   **if** $\deg_x(F_i) > 0$ **then**
13.   Compute $\mathrm{Sres}_{y,i}(P,Q)$
14. $\mathcal{TD} = \mathcal{TD} \cup \{F_i, \mathrm{Sres}_{y,i}(P,Q)\}$
15.   **return** $\mathcal{TD}$
16. **else if** P and Q are in $\mathbb{Z}[x]$ **then**
17.   **return** $\emptyset$
18. **else** {Assume wlog that P is in $\mathbb{Z}[x]$ (and Q is not)}
19.   **return** $\{F, Q\}$

Algorithm 2 Complete triangular decomposition

Input: P, Q in $\mathbb{Z}[x,y]$ defining a zero-dimensional system.

Output: A set of regular triangular systems, each of the form $\{U(x), V(x,y)\}$ with coefficients in $\mathbb{Z}$, whose sets of solutions are disjoint and are exactly those of $\{P,Q\}$. The multiplicity of any solution in these triangular systems is the multiplicity of the solution in its fiber with respect to the system $\{P,Q\}$ (see Definition 1).

1. $A = 0$, $B = \overline{\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$, $\mathcal{TD} = \emptyset$
2. **repeat**
3.   **if**[6] $\deg_x(A) \neq 0$ **then**
4. $\mathcal{TD} = \mathcal{TD} \cup$ Algorithm 1(P,Q,A)
5. $P = P - \mathrm{Lc}_y(P) y^{\deg_y(P)}$, $Q = Q - \mathrm{Lc}_y(Q) y^{\deg_y(Q)}$
6. $B_{new} = \gcd(B, \mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$
7. $A = \frac{B}{B_{new}}$
8. **until** $\deg(B) = 0$
9. **return** $\mathcal{TD}$

F, which is squarefree; indeed the first time Algorithm 1 is called F is squarefree by definition (Line 2) and, in the other calls, F divides A, which divides B, which divides $\overline{\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$

[6] Using the convention that the degree of the null polynomial is $-\infty$.

(see Algorithm 2). Thus, the multiplicity of (α, β) in $\{U(x), V(x,y)\}$ is the multiplicity of β in the univariate polynomial $V(\alpha, y)$. The bivariate polynomial V is defined either as P_h or Q_h (Line 19) or as $\mathrm{Sres}_{y,i}(P_h, Q_h)$ (Line 14).

In the latter case, $V(\alpha, y) = \mathrm{Sres}_{y,i}(P_h, Q_h)(\alpha, y)$ is equal to the gcd of $P_h(\alpha, y)$ and $Q_h(\alpha, y)$ by Theorem 3. By construction, if $\{U(x), V(x,y)\}$ is output by the h-th call of Algorithm 1, then the $h-1$ first (non-zero) coefficients of P and Q (seen as polynomials in y) vanish at α. In other words, $P_h(\alpha, y) = P(\alpha, y)$ and similarly for Q_h. Thus, the multiplicity of β in $V(\alpha, y)$ is the multiplicity of β in $\gcd(P(\alpha, y), Q(\alpha, y))$, which is by definition the multiplicity of (α, β) in its fiber with respect to $\{P, Q\}$.

In the former case, if say $V = Q_h$ then $P_h \in \mathbb{Z}[x]$ and $P_h(\alpha) = 0$. The gcd of $P_h(\alpha)$ and $Q_h(\alpha, y)$ is thus $Q_h(\alpha, y)$. The multiplicity of β in $V(\alpha, y) = Q_h(\alpha, y)$ is thus its multiplicity in $\gcd(P_h(\alpha), Q_h(\alpha, y))$ which is equal to $\gcd(P(\alpha, y), Q(\alpha, y))$, as above. Hence, as above, the multiplicity of β in $V(\alpha, y)$ is the multiplicity of (α, β) in its fiber with respect to $\{P, Q\}$, which concludes the proof. ☐

We now analyze the complexity of Algorithms 2 and start by two preliminary lemmas, which are direct generalizations of Propositions 15 and 16 in [BLM+15] but expressed in terms of d_x and d_y instead of the total degree.

Lemma 8. *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 with $A \equiv 0$. We have:*

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

Proof. Bouzidi et al. [BLM+15, Prop. 15] proved the above bounds with d^2 in place of $d_x d_y$ and $d\tau$ in place of $d_y \tau$. There, d^2 and $d\tau$ refer to the bounds on the degree and the bitsize of $\mathrm{Res}_y(P, Q)$. The degree and bitsize of this resultant can also be expressed as $O(d_x d_y)$ and $\widetilde{O}(d_y \tau)$ (by Lemma 4) and literally replacing in [BLM+15, Prop. 15] the bound $O(d^2)$ on the degree of the resultant by $O(d_x d_y)$ and the bound $\widetilde{O}(d\tau)$ on its bitsize by $\widetilde{O}(d_y \tau)$ directly yields the result. ☐

The following lemma is a direct and straightforward generalization of [BLM+15, Prop. 16], which proves a bit complexity of $\widetilde{O}_B(d^6 + d^5\tau)$ for Algorithm 1 with $A \equiv 0$.[7]

Lemma 9. *If P, Q in $\mathbb{Z}[x,y]$ have degree at most d_x in x, d_y in y, and bitsize at most τ, Algorithm 1 with $A \equiv 0$ performs $\widetilde{O}_B(d_x^3 d_y^3 + (d_x^2 d_y^3 + d_x d_y^4)\tau)$ bit operations in the worst case.*

Proof. By Lemma 4, each of the principal subresultant coefficients $\mathrm{sres}_{y,i}$ (including the resultant) has degree $O(d_x d_y)$ and bitsize $\widetilde{O}(d_y \tau)$. Thus, in Line 2, by Lemma 5, the squarefree part of the resultant can be computed in $\widetilde{O}_B((d_x d_y)^2 (d_y \tau)) = \widetilde{O}_B(d_x^2 d_y^3 \tau)$ bit operations and its bitsize is in $O(d_x d_y + d_y \tau) = O(d_y(d_x + \tau))$. In the same line, still by Lemma 5, $\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$ has bitsize $O(d_x + \tau)$ and it can be computed in $\widetilde{O}_B(d_x^2 \tau)$ bit operations; its squarefree part can thus be computed in $\widetilde{O}_B(d_x^2(d_x + \tau))$ bit operations and its bitsize is still in $O(d_x + \tau)$. Still in Line 2, the exact division R/B, which is a gcd-free computation, can be done with $\widetilde{O}_B((d_x d_y)^2 (d_y(d_x + \tau))) = \widetilde{O}_B(d_x^2 d_y^3 (d_x + \tau))$ bit operations.

[7] Note that there is nonetheless a minor difference between Algorithm 1 (with $A \equiv 0$) and the one analyzed in [BLM+15, Prop. 16], which is that in the former we consider $F = \overline{\mathrm{Res}_y(P,Q)/\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))}$ instead of $\overline{\mathrm{Res}_y(P,Q)}$ with the assumption that $\mathrm{Lc}_y(P)$ and $\mathrm{Lc}_y(Q)$ are coprime in the latter. However, this has no impact on the complexity because, by Mignotte's lemma, F has degree $O(d^2)$ and bitsize $\widetilde{O}(d^2 + d\tau)$ as $\overline{\mathrm{Res}_y(P,Q)}$.

By Lemma 4, the sequence of the subresultants $\mathrm{Sres}_{y,i}(P,Q)$ can be computed in $\widetilde{O}_B(d_x d_y^4 \tau)$ bit operations and the sequence of their principal coefficients $\mathrm{sres}_i(P,Q)$ (including the resultant) can be computed in $\widetilde{O}_B(d_x d^3 \tau)$ bit operations. Thus, the overall bit complexity of Lines 7 and 13 is $\widetilde{O}_B(d_x d_y^4 \tau)$.

Line 10 performs, in total, d_y gcd computations between polynomials G_{i-1} and $\mathrm{sres}_{y,i}$. Polynomial $\mathrm{sres}_{y,i}$ has bitsize $\widetilde{O}(d_y \tau)$ and degree $O(d_x d_y)$, and denoting by τ_i and d_i the bitsize and degree of G_i, Lemma 5 yields a complexity in $\widetilde{O}_B((d_x d_y)((d_x d_y)\tau_{i-1} + d_{i-1} d_y \tau))$ for the computation of G_i. According to Lemma 8, these complexities sum up over all i to $\widetilde{O}_B((d_x d_y)^2 (d_x d_y + d_y \tau))$.

Finally, in Line 11, by Lemma 5, the exact division of G_{i-1} by G_i can be done with a bit complexity $O_B(d_i^2 \tau_i)$. Since $d_i \leqslant d_x d_y$ by Lemma 8, $\sum_i O_B(d_i^2 \tau_i) = \widetilde{O}_B((d_x d_y)^2 (d_x d_y + d_y \tau))$.

Hence, the overall bit complexity of the algorithm is in $\widetilde{O}_B(d_x^3 d_y^3 + (d_x^2 d_y^3 + d_x d_y^4)\tau)$. $\square$

Proposition 10. *Let P, Q in $\mathbb{Z}[x,y]$ be two polynomials of degrees at most d_x and d_y in x and y, with coefficients of bitsize at most τ, and defining a zero-dimensional system. With $d = \max(d_x, d_y)$, Algorithm 2 computes a triangular decomposition of $\{P, Q\}$ using $\widetilde{O}_B(d^6 + d^5 \tau)$ bit operations in the worst case. In terms of d_x and d_y, this complexity is $\widetilde{O}_B(d_x^3 d_y^3 + (d_x^2 d_y^3 + d_x d_y^4)\tau)$. Moreover, the total bitsize of the decomposition is in $\widetilde{O}((d_x^2 d_y^3 + d_x d_y^4)\tau)$.*

Proof. The number of iterations of the loop in Algorithm 2 is at most $d_y + 1$. Beside the calls to Algorithm 1, Algorithm 2 thus performs $O(d_y)$ gcd operations and exact divisions of univariate polynomials. The degree of these polynomials is trivially at most d_x and their bitsizes are in $O(d_x + \tau)$ by Mignotte's lemma [BPR06, Corollary 10.12] because the gcds always divide some coefficients of P (and Q) seen in $\mathbb{Z}[x][y]$. Thus, by Lemma 5, the bit complexity of each of the gcd and exact division (i.e., a gcd-free) computations is in $\widetilde{O}_B(d_x^2(d_x + \tau))$, which yields a total bit complexity in $\widetilde{O}_B(d_y d_x^2 (d_x + \tau))$.

We now analyze the complexity of the calls to Algorithm 1. Denote by $P_h, Q_h, A_h, F_h, F_{h,i}, G_{h,i}$ the instances of P, Q, A, F, F_i, G_i in the h-th call to Algorithm 1. Since Algorithm 1 is called only if $\deg_x(A) \neq 0$, we have that $\deg_x(A_{h>1}) > 0$. It follows that h varies from 1 to at most d_x because $\prod_{h>1} A_h$ encodes the common vertical asymptotes of P and Q (as noted in the proof of Lemma 6) and there are at most d_x such asymptotes.

By Lemma 9, the first call to Algorithm 1 with $A_1 \equiv 0$ has bit complexity $\widetilde{O}_B(d_x^3 d_y^3 + (d_x^2 d_y^3 + d_x d_y^4)\tau)$.

In the rest of the proof, we consider the calls to Algorithm 1 except for the first one. In all these calls, the polynomials $F_{h,i}$ are pairwise coprime by construction and their product encodes a subset of the common vertical asymptotes of the initial input polynomials P and Q (i.e. $\prod_{h,i} F_{h,i}$ divides $\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$). Hence, in Line 13, at most d_x subresultant polynomials $\mathrm{Sres}_{y,i}(P_h, Q_h)$ are computed over all calls (but the first one). Since P_h and Q_h are truncated versions of P and Q, their degrees and bitsize are still bounded by d_x, d_y and τ, hence all subresultant polynomials can be computed in a total bit complexity of $\widetilde{O}_B(d_x^2 d_y^3 \tau)$, by Lemma 4.

In Line 4, by Lemma 4, the resultant R_h of P_h and Q_h can be computed with $\widetilde{O}_B(d_x d_y^3 \tau)$ bit operations and its degree and bitsize are in $O(d_x d_y)$ and $\widetilde{O}(d_y \tau)$, respectively. By Lemma 5, the gcd B_h of $\mathrm{Lc}_y(P_h)$ and $\mathrm{Lc}_y(Q_h)$ can be computed in $\widetilde{O}_B(d_x^2 \tau)$ bit operations and its degree and bitsize are in $O(d_x)$ and $O(d_x + \tau)$, respectively. On the other hand, A_h divides $\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$ thus its degree and bitsize are in $O(d_x)$ and $O(d_x + \tau)$, respectively, by Mignotte's lemma. Thus, by Lemma 5, $\gcd(R_h, A_h)$ and $\gcd(B_h, A_h)$ can be computed in $\widetilde{O}_B(\max(d_x d_y, d_x)((d_x d_y)(d_x + \tau) + d_x(d_y \tau))) = \widetilde{O}_B((d_x d_y)^2 (d_x + \tau))$ bit operations and their degree and bitsize are in $O(d_x)$ and $O(d_x + \tau)$, respectively. Furthermore, the same lemma yields that the exact division $\gcd(R_h, A_h)/\gcd(B_h, A_h)$, that is the gcd-free part of $\gcd(R_h, A_h)$ with respect to $\gcd(B_h, A_h)$, can be computed in $\widetilde{O}_B(d_x^2(d_x+\tau))$ bit operations. One iteration of Line 4 thus has a bit complexity in $\widetilde{O}_B((d_xd_y)^2(d_x+\tau))$, which yields a bit complexity in $\widetilde{O}_B(d_y(d_xd_y)^2(d_x+\tau))$ for all calls (but the first one).

Similarly, in Line 11, one division $G_{h,i-1}/G_{h,i}$ has complexity $\widetilde{O}_B(d_x^2(d_x+\tau))$. Indeed $G_{h,i}$ divides $G_{h,0} = F_h$, which divides $\mathrm{Lc}_y(P)$, which has degree at most d_x and bitsize at most τ. Thus, $G_{h,i}$ has degree at most d_x and bitsize $O(d_x+\tau)$ by Mignotte's lemma, which yields the complexity bound $\widetilde{O}_B(d_x^2(d_x+\tau))$. There are $O(d_y^2)$ calls to Line 11 and thus the total bit complexity of that line is in $\widetilde{O}_B((d_xd_y)^2(d_x+\tau))$.[8]

In Line 7, for every call, the bit complexity of the computation of the principal subresultant sequence is in $\widetilde{O}_B(d_xd_y^3\tau)$ by Lemma 4, yielding a complexity in $\widetilde{O}_B(d_x^2d_y^3\tau)$ for all the $O(d_x)$ calls to Algorithm 1.

We finally analyze the complexity of Line 10 where $G_{h,i} = \gcd(G_{h,i-1}, \mathrm{sres}_{y,i}(P_h, Q_h))$ is computed. For that purpose, we need to amortize the sum of the degrees $d_{h,i}$ and the sum of the bitsizes $\tau_{h,i}$ of $G_{h,i}$ over h. For the degree, it is straightforward that $\sum_h d_{h,i} \leqslant d_x$, for any i, because, as noted above, $G_{h,i}$ divides $G_{h,0} = F_h$, thus $\prod_h G_{h,i}$ divides $\prod_h F_h$, which divides $\gcd(\mathrm{Lc}_y(P), \mathrm{Lc}_y(Q))$.

We now prove that $\sum_h \tau_{h,i} = O(d_x+\tau)$, for any i, using Mahler's measure, as in [BLM+15, Prop. 15]. 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 h (i.e., $h = fg$) implies that $M(h) \geqslant M(f)$. We also have the following 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 $\log M(f) \leqslant \log \sqrt{d+1} + \log ||f||_\infty$.

By Inequality (i),

$$\sum_h \tau_{h,i} \leqslant d_x + \sum_h d_{h,i} + \log M(\prod_h G_{h,i}).$$

As noted above, $\sum_h d_{h,i} \leqslant d_x$ and $\prod_h G_{h,i}$ divides $\mathrm{Lc}_y(P)$, thus $M(\prod_h G_{h,i}) \leqslant M(\mathrm{Lc}_y(P))$ and by Inequality (ii), $\log M(\prod_h G_{h,i}) \leqslant \log M(\mathrm{Lc}_y(P)) = O(\tau + \log d_x)$. Hence, $\sum_h \tau_{h,i} = O(d_x + \tau)$.

Now, since $\mathrm{sres}_{y,i}(P_h, Q_h)$ has degree $O(d_xd_y)$ and bitsize $\widetilde{O}(d_y\tau)$ by Lemma 4, computing $G_{h,i} = \gcd(G_{h,i-1}, \mathrm{sres}_{y,i}(P_h, Q_h))$ has bit complexity $\widetilde{O}_B(d_xd_y(d_xd_y\tau_{h,i-1} + d_{h,i-1}d_y\tau))$ by Lemma 5. Summing over h gives $\widetilde{O}_B((d_xd_y)^2(d_x+\tau))$ and summing over i multiplies the complexity by d_y, which yields $\widetilde{O}_B(d_x^3d_y^3 + d_x^2d_y^3\tau)$. This concludes the proof that the overall bit complexity of Algorithm 2 is in $\widetilde{O}_B(d_x^3d_y^3 + (d_x^2d_y^3 + d_xd_y^4)\tau)$.

We now analyze the size of the output triangular decomposition. The first call to Algorithm 1 outputs at most d_y triangular systems and the other calls at most d_x triangular systems as already noticed above. The product of the univariate polynomials of all these systems divides the resultant of P and Q which has degree in $O(d_xd_y)$ and bitsize in $\widetilde{O}(d_y\tau)$ (Lemma 4). The univariate polynomials of the decomposition all together thus have $O(d_xd_y)$ coefficients, and by Mignotte's

[8] Note that this complexity is actually overestimated because (i) we need to perform the division $F_{h,i} = G_{h,i-1}/G_{h,i}$ only if $F_{h,i}$ has positive degree, which occurs at most d_x times in total, as noted above, and (ii) the exact division can be performed with a bit complexity that is softly linear in the squared degree plus the degree times the bitsize [vzGG13, Exercice 9.14].

lemma, their bitsizes are in $\widetilde{O}(d_y\tau + d_xd_y)$. Their total bitsize is thus in $\widetilde{O}(d_x^2d_y^2 + d_xd_y^2\tau)$. The bivariate polynomials of the decomposition are subresultant polynomials of the input polynomials P and Q or truncated versions of them. According to Lemma 4, each subresultant polynomial has degree $O(d_xd_y)$ in x, degree at most d_y in y, and bitsize in $\widetilde{O}(d_y\tau)$. The bivariate polynomials thus have, in total, $O((d_x + d_y)(d_xd_y)d_y)$ coefficients of bitsize $\widetilde{O}(d_y\tau)$. Their total bitsize is thus in $\widetilde{O}((d_x^2d_y^3 + d_xd_y^4)\tau)$ and the total bitsize of the decomposition has the same complexity. □

References

[BLM+15] Y. Bouzidi, S. Lazard, G. Moroz, M. Pouget, F. Rouillier, and M. Sagraloff. Improved algorithms for solving bivariate systems via Rational Univariate Representations. Research report, Inria, June 2015.

[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.

[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.

[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.

[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.

[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.

[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.

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

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