



HAL
open science

Improved algorithms for solving bivariate systems via Rational Univariate Representations

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

► **To cite this version:**

Yacine Bouzidi, Sylvain Lazard, Guillaume Moroz, Marc Pouget, Fabrice Rouillier, et al.. Improved algorithms for solving bivariate systems via Rational Univariate Representations. [Research Report] Inria. 2015. hal-01114767v2

HAL Id: hal-01114767

<https://inria.hal.science/hal-01114767v2>

Submitted on 17 Jun 2015

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

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

Improved algorithms for solving bivariate systems via Rational Univariate Representations

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

June 17, 2015

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 $\tilde{O}_B(d^6 + d^5\tau)$, where \tilde{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 $\tilde{O}_B(d^5 + d^4\tau)$. 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, unlike Monte-Carlo algorithms in which the output is only correct with some

^{*}INRIA, France. `Firstname.Name@inria.fr`

[†]LORIA laboratory, Nancy, France.

[‡]Institut de Mathématiques de Jussieu, Paris, France.

[§]Max-Planck-Institut für Informatik, Saarbrücken, Germany.

29 probability. We consider throughout the paper input polynomials of total degree at most d with
 30 integer coefficients of bitsize at most τ .

31 A classical approach for solving a system of polynomials with a finite number of solutions is to
 32 compute a rational parameterization of its solutions. A rational parameterization is a representation
 33 of the (complex) solutions by a set of univariate polynomials and associated rational one-to-one
 34 mappings that send the roots of the univariate polynomials to the solutions of the system. With
 35 such a representation, many queries on the system can be transformed into queries on univariate
 36 polynomials, which ease the computations. For instance, isolating the solutions of the system can
 37 be done by isolating the roots of the univariate polynomials of the rational parameterization and by
 38 computing the image of the resulting intervals through the associated mappings. Similarly, quering
 39 whether a polynomial P vanishes at the solutions of the system can be done by substiting in P the
 40 variables by their images in each of the one-to-one mappings and by testing whether this resulting
 41 univariate polynomial vanishes at the roots of the associated univariate polynomial in the rational
 42 parameterization.

43 The core of the algorithms that compute such rational parameterizations (see for example
 44 [ABRW96, BSS03, DET09, GLS01, GVEK96, Rou99] and references therein) is the computation of
 45 a so-called *linear separating form* for the solutions, that is, a linear combination of the coordinates
 46 that takes different values when evaluated at different solutions of the system. Then, a shear of
 47 the coordinate system using such a linear form ensures that the system is in generic position, in
 48 the sense that no two solutions are vertically aligned. Since a linear form chosen randomly in
 49 a sufficiently large finite set is separating with probability close to one, probabilist Monte-Carlo
 50 algorithms can avoid this computation by considering a random linear form. However, when it
 51 comes to deterministically computing a linear separating form, or even to check that an arbitrary
 52 (e.g. random) linear form is separating, this, surprisingly, was until very recently the bottleneck in
 53 the computation of rational parameterizations, even for bivariate systems, as discussed below.

54 For arbitrary multivariate systems, Rouillier [Rou99] gives an algorithm for deterministically
 55 computing a separating form, which computes the number of solutions of a system with the rank of
 56 the Hermite's quadratic form of a quotient algebra. The complexity of this computation dominates
 57 the one that follows for computing the rational representation. Considering the special case of
 58 systems of two bivariate polynomials of total degree bounded by d with integer coefficients of
 59 bitsize bounded by τ , another approach, based on a triangular decomposition, has been presented
 60 by Gonzalez-Vega and El Kahoui [GVEK96] for computing a separating linear form together with
 61 a rational parameterization of the solutions. The best-known bit complexity of this approach,
 62 analyzed by Diochnos et al. [DET09, Lemma 16 & Theorem 19]¹, shows a bit complexity in $\tilde{O}_B(d^{10} +$
 63 $d^9\tau)$ for computing a separating form and a bit complexity in $\tilde{O}_B(d^7 + d^6\tau)$ for computing the
 64 corresponding rational parameterization. The computation of a separating linear form was still
 65 the bottleneck in the computation of the rational parameterization. An algorithm using modular
 66 arithmetic was then introduced by Bouzidi et al. [BLPR15] reducing the complexity to $\tilde{O}_B(d^8 + d^7\tau)$.
 67 This algorithm was later simplified and improved by transforming the problem into the computation
 68 of a separating form for the critical points of a curve, which improved the bit complexity to $\tilde{O}_B(d^7 +$
 69 $d^6\tau)$ in the worst case and to $\tilde{O}_B(d^5 + d^4\tau)$ in a probabilistic Las Vegas setting [BLP+14]. Bouzidi
 70 et al. [BLPR15] also showed that, given such a separating linear form, an alternative rational
 71 parameterization called Rational Univariate Representation (RUR) [Rou99] can be computed using
 72 $\tilde{O}_B(d^7 + d^6\tau)$ bit operations. For the first time, the worst-case bit complexities of computing linear

¹The overall bit complexity stated in [DET09, Theorem 19] is $\tilde{O}_B(d^{12} + d^{10}\tau^2)$ 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 $\tilde{O}_B(d^{10} + d^9\tau)$ using Pan [Pan02] results on the complexity of isolating the real roots of a univariate polynomial.

73 separating forms and rational parameterizations (even RURs) of bivariate systems were both in
 74 the same class of complexity $\tilde{O}_B(d^7 + d^6\tau)$ and, also for the first time, the expected bit complexity
 75 for computing linear separating forms, in a Las-Vegas setting, was in a smaller class of complexity,
 76 $\tilde{O}_B(d^5 + d^4\tau)$.

77 Very recently, Kobel and Sagraloff [KS15b] presented an algorithm of worst-case bit complexity
 78 $\tilde{O}_B(d^6 + d^5\tau)$ for computing isolating boxes of the solutions of bivariate systems. Their approach
 79 is based on resultant computations, projecting the solutions on the x and y -axes, thus defining
 80 a grid of candidate solutions. Then, approximate evaluations combined with adaptive evaluation
 81 bounds enable to identify the solutions from the grid. This method does not need the knowledge of
 82 a separating form, but once the solutions are isolated with enough precision, such a separating form
 83 can be computed in $\tilde{O}_B(d^6 + d^5\tau)$ bit operations [KS15b]. This approach for computing separating
 84 linear forms has the best known worst-case complexity. However, it would be surprising that
 85 computing a separating form with such complexity would require to first isolate the solutions of the
 86 system, since separating forms are precisely instrumental for solving systems in parameterization-
 87 based approaches. The present work indeed demonstrates that separating linear forms and rational
 88 parameterizations (including RURs) can be directly computed with this $\tilde{O}_B(d^6 + d^5\tau)$ state-of-the-
 89 art worst-case bit complexity, and that the solutions of the system can be isolated from the RUR
 90 in the same worst-case complexity.

91 **Main results.** Let P and Q be two coprime polynomials in $\mathbb{Z}[x, y]$ of degree bounded by d and
 92 bitsize bounded by τ . We present three algorithms, one for each of the main steps of solving a
 93 bivariate system $\{P, Q\}$ via RURs, that is, computing (i) a linear separating form, (ii) a RUR
 94 decomposition of the system, and (iii) isolating boxes of the solutions. Each of these algorithms
 95 has worst-case bit complexity $\tilde{O}_B(d^6 + d^5\tau)$ and we also present Las-Vegas variants of expected bit
 96 complexity $\tilde{O}_B(d^5 + d^4\tau)$ of our two first algorithms (see Theorems 28, 29, 45, 46, 61). We do not
 97 present a Las-Vegas variant of our last algorithm for computing isolating boxes but it should be
 98 noticed that the complexity of that subdivision-based algorithm actually depends on the distances
 99 between the solutions and thus its worst-case complexity is not always reached; moreover, we do
 100 not know whether our $\tilde{O}_B(d^6 + d^5\tau)$ worst-case upper bound is tight for step (iii).

101 Our algorithm for computing a separating linear form is based on the one presented in [BLP⁺14],
 102 while improving its worst-case bit complexity by a factor d . Furthermore, its Las-Vegas variant
 103 is simpler than the one presented in [BLP⁺14] and it has the same expected bit complexity. As
 104 mentioned above the worst-case complexity of this new algorithm also matches the recent one
 105 by Kobel and Sagraloff [KS15b]. Our algorithm for computing a RUR decomposition of $\{P, Q\}$
 106 improves by a factor d the state-of-the-art worst-case bit complexity [BLPR15]. Furthermore, our
 107 Las-Vegas variant is, up to our knowledge, the first Las-Vegas algorithm whose expected complexity
 108 is asymptotically better than the worst-case complexity and, as a result, our Las-Vegas algorithm
 109 improves the state-of-the-art expected bit complexity by a factor d^2 . For the isolation problem
 110 from the RURs, we improve the state-of-the-art complexity by a factor d^2 [BLPR15, Proposition
 111 35], while matching the resultant-based complexity presented by Kobel and Sagraloff [KS15b].

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

117 We first present a detailed overview of our contributions in Section 2. Notation and prelim-
 118 inaries are then introduced in Section 3. We present in Section 4 our amortized analysis of the
 119 triangular decomposition and a related luckiness certificate which is a key feature of the following

120 multi-modular algorithms. Sections 5, 6 and 7 present respectively our algorithms for computing
121 separating forms, RUR decompositions and isolating boxes of the solutions.

122 2 Overview

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

125 2.1 Triangular decomposition and luckiness certificate

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

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

135 We then present in Section 4.3 another variation on the triangular decomposition algorithm,
136 which computes a luckiness certificate for this triangular decomposition. A luckiness certificate of
137 $\{P, Q\}$ is an integer such that if a prime μ does not divide it, then μ is lucky for the triangular
138 decomposition of $\{P, Q\}$ that is, the degree of the decomposition is preserved by the reduction
139 modulo μ and the decomposition commutes with the reduction modulo μ (see Definition 13). Our
140 deterministic algorithms for the separating form and the RUR computations will both use this
141 luckiness certificate.

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

149 2.2 Separating linear form

150 In Section 5, we present a new algorithm for computing separating linear forms for a bivariate
151 system $\{P, Q\}$. We actually present two algorithms, a deterministic one of worst-case bit complex-
152 ity $\tilde{O}_B(d^6 + d^5\tau)$ and a probabilistic Las-Vegas variant of expected bit complexity $\tilde{O}_B(d^5 + d^4\tau)$
153 (Theorems 28 and 29).

154 Our approach is based on the algorithms presented in [BLPR15] and [BLP+14] while improving
155 the worst-case bit complexity by one order of magnitude. We briefly recall the essence of these
156 algorithms. The first step of the algorithm presented in [BLPR15] is to compute the number
157 of distinct solutions and a so-called lucky prime for the system. Such a lucky prime is, roughly
158 speaking, a prime such that the system has the same number of distinct solutions as its image
159 modulo μ (see Definition 18). In a second step, all polynomials and computations are considered
160 modulo μ . The algorithm then considers iteratively a candidate separating form $x + ay$ with an
161 integer a incrementing from 0. The algorithm computes the number of distinct solutions after

162 projection along the direction of the line $x + ay = 0$ and stops when a value a is found such that the
 163 number of distinct projected solutions equals that of the system. The worst-case bit complexity of
 164 this algorithm is in $\tilde{O}_B(d^8 + d^7\tau)$.

165 The main additional ideas introduced in [BLP⁺14] are that it is sufficient to compute a separa-
 166 rating form for the system $\{H, \frac{\partial H}{\partial y}\}$ of critical points of a curve H associated to the input system
 167 $\{P, Q\}$ (see Section 5.2) and that the number of critical points can easily be computed as the differ-
 168 ence between the degrees of the triangular decompositions of the systems $\{H, (\frac{\partial H}{\partial y})^2\}$ and $\{H, \frac{\partial H}{\partial y}\}$
 169 (see Definition 11). This improves the worst-case bit complexity of the algorithm to $\tilde{O}_B(d^7 + d^6\tau)$.

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

174 In Section 5.3.2, we present a new algorithm for computing a lucky prime for the system
 175 $\{H, \frac{\partial H}{\partial y}\}$ using the luckiness certificates for triangular decompositions presented in Section 4.3.
 176 More precisely, we compute a lucky prime for $\{H, \frac{\partial H}{\partial y}\}$ by computing a prime μ that, essentially,
 177 does not divide the product of the luckiness certificates of the two systems $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$.
 178 By definition of the luckiness certificates, the degrees of the triangular decompositions of these two
 179 systems are the same over \mathbb{Z} and \mathbb{Z}_μ . The difference of these degrees, which is the number of
 180 solution of $\{H, \frac{\partial H}{\partial y}\}$, is thus also the same over \mathbb{Z} and \mathbb{Z}_μ , which essentially yields that μ is lucky
 181 for $\{H, \frac{\partial H}{\partial y}\}$.

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

186 In Section 5.4, we wrap up these results, which prove that we can compute a separating linear
 187 form for the input system $\{P, Q\}$ in $\tilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case (Theorem 28).

188 Finally, we show in Section 5.5 that our deterministic algorithm can be modified in a straight-
 189 forward manner into a probabilistic Las-Vegas algorithm of expected bit complexity $\tilde{O}_B(d^5 + d^4\tau)$
 190 (Theorem 29). This is done by choosing randomly a linear form $x + ay$ and a prime μ for the sys-
 191 tem $\{H, \frac{\partial H}{\partial y}\}$, until the number of distinct solutions of $\{H, \frac{\partial H}{\partial y}\}$ is equal to the number of distinct
 192 solutions of that system modulo μ and after projection along the direction of the line $x + ay = 0$.

193 This new algorithm is similar to one presented in [BLP⁺14] and it has the same expected
 194 bit complexity, while its worst-case counterpart is improved by a factor d . Furthermore, this new
 195 algorithm is simpler because, in particular, (i) we choose random values for a and μ together instead
 196 of first computing a lucky prime and only then a separating form and (ii) we avoid the explicit
 197 computation of the constant in the asymptotic upper bound on the number of unlucky prime.

198 2.3 Multimodular RUR decomposition

199 We present in Section 6 a new algorithm for computing a rational parameterization of the solutions
 200 of a bivariate system $\{P, Q\}$. As in Section 5, we actually present two algorithms, a deterministic
 201 one of worst-case bit complexity $\tilde{O}_B(d^6 + d^5\tau)$ and a probabilistic Las-Vegas variant of expected bit
 202 complexity $\tilde{O}_B(d^5 + d^4\tau)$ (Theorems 45 and 46). We consider here that a separating form $x + ay$
 203 has been computed for $\{P, Q\}$ as shown in Section 5.

204 Recall that the two algorithms with best known bit complexity for computing rational param-
 205 eterizations of the solutions are those by Gonzalez-Vega and El Kahoui [GVEK96] and by Bouzidi
 206 et al. [BLPR15], both of complexity $\tilde{O}_B(d^7 + d^6\tau)$. The former algorithm first shears the input

207 polynomials according the separating form (to ensure that no two solutions are vertically aligned)
 208 and then computes a parameterization of the solutions of every system of the triangular decompo-
 209 sition of the sheared system; the multiplicities of the solutions of $\{P, Q\}$ are thus not preserved.
 210 The latter algorithm computes a single RUR of $\{P, Q\}$, which preserves the multiplicities of the
 211 solutions (see Proposition 35). In this latter approach, the input bivariate polynomials are formally
 212 sheared using an additional variable that parameterizes a generic linear form, and the resultant of
 213 these (trivariate) polynomials is computed. The polynomials of the RUR are then expressed (and
 214 computed) as combinations of this resultant and its partial derivatives, specialized at the value a
 215 associated with the given linear separating form.

216 Here, we combine in essence these two approaches and compute a RUR for every system of
 217 the triangular decomposition of the sheared input system. However, in order to obtain the claimed
 218 complexities, we do not compute a RUR of every triangular system using the approach of [BLPR15].
 219 Instead, Algorithm 6 works as follows. First, we compute the triangular decomposition of the
 220 sheared input system as in [GVEK96]. We then show that the radical ideals of the triangular
 221 systems can easily be obtained (Lemma 39). Using the simple structure of these radical ideals, we
 222 derive formulas for their RURs (Lemma 40). For complexity issues, we do not use these formulas to
 223 directly compute RURs over \mathbb{Q} but we use instead a multi-modular approach. For that purpose, we
 224 use known bounds on the size of the RUR coefficients and the luckiness certificate of the triangular
 225 decomposition introduced in Section 4.3 to select the primes.

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

234 2.4 Computing isolating boxes from a RUR decomposition

235 Section 7 introduces Algorithm 9 computing isolating boxes for the complex solutions from a
 236 RUR. By definition, the RUR of an ideal I defines a mapping between the roots of a univariate
 237 polynomial and the solutions of I . A RUR is hence naturally designed to compute isolating boxes
 238 using univariate isolation and interval evaluation.

239 An algorithm with bit complexity $\tilde{O}_B(d^8 + d^7\tau)$ was presented in [BLPR15, §5.1 and Proposition
 240 35] for the isolation of the real solutions of a system $\{P, Q\}$ of two bivariate polynomials of degree
 241 bounded by d and bitsize bounded by τ . Section 7.2 presents a modified algorithm that isolates all
 242 complex solutions. Using several amortized bounds for the roots of polynomials (Section 7.1), we
 243 show that Algorithm 9 applied to a RUR decomposition of a system $\{P, Q\}$, isolates all complex
 244 solutions in $\tilde{O}_B(d^6 + d^5\tau)$ (Theorem 61).

245 3 Notation and preliminaries

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

249 The bitsize of an integer p is the number of bits needed to represent it, that is $\lfloor \log p \rfloor +$
 250 1 (log refers to the logarithm in base 2). The bitsize of a rational is the maximum bitsize of

251 its numerator and its denominator. The bitsize of a polynomial with integer coefficients is the
 252 *maximum* bitsize of its coefficients. As mentioned earlier, O_B refers to the bit complexity and \tilde{O}
 253 and \tilde{O}_B refer to complexities where polylogarithmic factors are omitted, i.e., $f(n) \in \tilde{O}(g(n))$ if
 254 $f(n) \in O(g(n) \log^k(n))$ for some $k \in \mathbb{N}$.

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

261 In the following, μ is a prime number and we denote by \mathbb{Z}_μ the quotient $\mathbb{Z}/\mu\mathbb{Z}$. We denote by
 262 $\phi_\mu: \mathbb{Z} \rightarrow \mathbb{Z}_\mu$ the reduction modulo μ , and extend this definition to the reduction of polynomials
 263 with integer coefficients. We denote by \mathbb{D} a unique factorization domain, typically $\mathbb{Z}[x, y]$, $\mathbb{Z}[x]$,
 264 $\mathbb{Z}_\mu[x]$, \mathbb{Z} or \mathbb{Z}_μ . We also denote by \mathbb{F} a field, typically \mathbb{Q} , \mathbb{C} , or \mathbb{Z}_μ and by $\mathbb{F}_{\mathbb{D}}$ the fraction field of \mathbb{D} .

265 For any polynomial P in $\mathbb{D}[x]$, let $\text{Lc}_x(P)$ denote its leading coefficient with respect to the
 266 variable x and $\text{d}_x(P)$ its degree with respect to x . The degree of a polynomial refers to its *total*
 267 degree, unless specified otherwise. For any curve defined by $H(x, y)$ in $\mathbb{D}[x, y]$, we call the critical
 268 points of H with respect to x or more shortly the critical point of H , the points that are solutions of
 269 the system $\{H, \frac{\partial H}{\partial y}\}$. In this paper, the solutions of a system of polynomials are always considered
 270 in the algebraic closure of $\mathbb{F}_{\mathbb{D}}$.

271 **Subresultant and gcd.** We first recall the concept of *polynomial determinant* of a matrix which
 272 is used in the definition of subresultants. Let M be an $m \times n$ matrix with $m \leq n$ and M_i be
 273 the square submatrix of M consisting of the first $m - 1$ columns and the i -th column of M , for
 274 $i = m, \dots, n$. The *polynomial determinant* of M is the polynomial defined as $\det(M_m)y^{n-m} +$
 275 $\det(M_{m+1})y^{n-(m+1)} + \dots + \det(M_n)$.

276 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
 277 generality, that $a_p b_q \neq 0$ and $p \geq q$.

278 The Sylvester matrix of P and Q , $\text{Syl}_y(P, Q)$ is the $(p + q)$ -square matrix whose rows are
 279 $y^{q-1}P, \dots, P, y^{p-1}Q, \dots, Q$ considered as vectors in the basis $y^{p+q-1}, \dots, y, 1$.

$$280 \quad \text{Syl}_y(P, Q) = \begin{array}{c} \overbrace{\hspace{10em}}^{\text{p+q columns}} \\ \left(\begin{array}{cccccc} a_p & a_{p-1} & \cdots & \cdots & a_0 & \\ & a_p & a_{p-1} & \cdots & \cdots & a_0 \\ & & \ddots & & & \\ & & & a_p & a_{p-1} & \cdots & \cdots & a_0 \\ b_q & b_{q-1} & \cdots & & b_0 & & & \\ & b_q & b_{q-1} & \cdots & b_0 & & & \\ & & \ddots & & & \ddots & & \\ & & & \ddots & & & \ddots & \\ & & & & b_q & b_{q-1} & \cdots & b_0 \end{array} \right) \begin{array}{l} \left. \vphantom{\begin{array}{c} a_p \\ a_p \\ \ddots \\ a_p \\ b_q \\ b_q \\ \ddots \\ b_q \end{array}} \right\} \text{q rows} \\ \left. \vphantom{\begin{array}{c} a_p \\ a_p \\ \ddots \\ a_p \\ b_q \\ b_q \\ \ddots \\ b_q \end{array}} \right\} \text{p rows} \end{array} \end{array}$$

281 For $i = 0, \dots, \min(q, p - 1)$, let $\text{Syl}_{y,i}(P, Q)$ be the $(p + q - 2i) \times (p + q - i)$ matrix obtained from
 282 $\text{Syl}_y(P, Q)$ by deleting the i last rows of the coefficients of P , the i last rows of the coefficients of
 283 Q , and the i last columns.

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

287 For practical consideration, when $q = p$, we define the q -th polynomial subresultant of P
 288 and Q as Q .² The polynomial $\text{Sres}_i(P, Q)$ has degree at most i in y and it can be written as
 289 $\text{sres}_{y,i}(P, Q)y^i + \text{sres}_{y,i,i-1}(P, Q)y^{i-1} + \cdots + \text{sres}_{y,i,0}(P, Q)$, where the coefficient of its monomial
 290 of degree i in y , $\text{sres}_i(P, Q)$, is called the i -th *principal subresultant coefficient*. Unless specified
 291 otherwise, the subresultants are always considered with respect to the variable y and then, for
 292 simplicity, we do not explicitly refer to the variable in the notation. Note that $\text{Sres}_0(P, Q) =$
 293 $\text{sres}_0(P, Q)$ is the *resultant* of P and Q with respect to y , which we also denote by $\text{Res}_y(P, Q)$.
 294 Again, when the resultant is considered with respect to y , we omit the reference to the variable
 295 and denote it by $\text{Res}(P, Q)$.

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

301 We state in Lemma 2 a fundamental property of subresultants which is instrumental in the
 302 triangular decomposition algorithm. For clarity, we state this property for bivariate polynomials
 303 $P = \sum_{i=0}^p a_i y^i$ and $Q = \sum_{i=0}^q b_i y^i$ in $\mathbb{D}[x, y]$, with $p \geq q$. This property is a direct consequence
 304 of the specialization property of subresultants and of the gap structure theorem; see for instance
 305 [EK03, Lemmas 2.3, 3.1 and Cor. 5.1].

306 Before stating Lemma 2, we recall that a greatest common divisor (gcd) of P and Q is a
 307 polynomial in $\mathbb{D}[x, y]$ that divides P and Q such that any common divisor of P and Q also divides
 308 the gcd in $\mathbb{D}[x, y]$. The greatest common divisor is unique only up to the multiplication by an
 309 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
 310 refer to any of them as *the* gcd for simplicity. On the other hand, when \mathbb{D} is a field, we refer to the
 311 monic gcd (with respect to a given ordering of the variables) as *to the* gcd. Furthermore, in the
 312 sequel, we sometimes compare gcds defined in $\mathbb{Z}_\mu[x, y]$ and the reduction modulo μ of gcds defined
 313 in $\mathbb{Z}[x, y]$; for simplicity, we often say they are equal if they are equal up to the multiplication by
 314 a non-zero constant in \mathbb{Z}_μ . Note finally that if P and Q are coprime in $\mathbb{Z}[x, y]$, then they define a
 315 zero-dimensional system.

316 **Lemma 2.** *For any α such that $a_p(\alpha)$ and $b_q(\alpha)$ do not both vanish,³ the first subresultant poly-*
 317 *nomial $\text{Sres}_k(P, Q)(\alpha, y)$ (for k increasing) that does not identically vanish is of degree k , and it is*
 318 *the gcd of $P(\alpha, y)$ and $Q(\alpha, y)$ (up to the multiplication by a non-zero constant in the fraction field*
 319 *of $\mathbb{D}(\alpha)$).*

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

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

- 324 • *The coefficients of $\text{Sres}_i(P, Q)$ have bitsize in $\tilde{O}(d_y \tau)$.*
- 325 • *The degree in x_j of $\text{Sres}_i(P, Q)$ is at most $2d(d_y - i)$.*

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.

³Note that this property is often stated with a stronger assumption, that is, that *none* of the leading coefficients $a_p(\alpha)$ and $b_q(\alpha)$ vanishes.

- For any $i \in \{0, \dots, \min(d_y(P), d_y(Q))\}$, the i -th Bézout's relation i.e. the polynomials $\text{Sres}_i(P, Q)$, U_i and V_i can be computed in $\tilde{O}(d^n d_y^{n+1})$ arithmetic operations and $\tilde{O}_B(d^n d_y^{n+2} \tau)$ bit operations. These complexities also hold for the computation of the sequence of principal subresultant coefficients $\text{sres}_i(P, Q)$.⁴

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 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 $q \leq p$ and of bitsizes τ_P and τ_Q , respectively.*

- *The coefficients of $\text{Sres}_i(P, Q)$ have bitsize in $\tilde{O}(p\tau_Q + q\tau_P)$.*
- *Any subresultant $\text{Sres}_i(P, Q)$ as well as the sequence of principal subresultant coefficients $\text{sres}_i(P, Q)$ can be computed in $\tilde{O}(p)$ arithmetic operations, and $\tilde{O}_B(p(p\tau_Q + q\tau_P))$ 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 $\tilde{O}(p)$ arithmetic operations, and $\tilde{O}_B(p(p\tau_Q + q\tau_P))$ 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 $\tilde{O}(p\tau_Q + q\tau_P)$ instead of simply $\tilde{O}(d\tau)$ 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 $\tilde{O}_B(p(p\tau_Q + q\tau_P))$ bit operations instead of simply $\tilde{O}(d^2\tau)$. 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 $\tilde{O}_B(p(p\tau_Q + q\tau_P))$ and have bitsize in $\tilde{O}(p\tau_Q + q\tau_P)$. 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 $\tilde{O}_B(p(p\tau_Q + q\tau_P))$ 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). \square

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].

⁴The 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.

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

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

375 We now consider P and G , the gcd of P and Q , as polynomials in y with coefficients in $\mathbb{Z}[x]$.
 376 The gcd-free part of P with respect to Q is the quotient of the Euclidean division between P and
 377 G . This division can be run in $\mathbb{Z}[x][y]$. Indeed, since the leading coefficient of G divides that of P ,
 378 it also divides the leading coefficient of each intermediate remainder $r_i = P - q_iG$ where q_i refers
 379 to the i -th truncation (with respect to y) of the quotient of P by G . Moreover, since G divides P
 380 and by Mignotte's lemma (see e.g. [BPR06, Corollary 10.12]), the polynomials q_i have coefficients
 381 of degree in $O(d)$ in x and bitsize in $O(d + \tau)$, and so as for the intermediate remainders r_i . The
 382 Euclidean division can thus be done using $O(d^2)$ additions and multiplications and $O(d)$ exact
 383 divisions between polynomials in $\mathbb{Z}[x]$ of degree in $O(d)$ and bitsize in $O(d + \tau)$, which yields a bit
 384 complexity in $\tilde{O}_B(d^4 + d^3\tau)$.

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

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

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

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

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

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

401 **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*
 402 *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*
 403 *the maximal ideal $\langle x - \alpha, y - \beta \rangle$. When this local ring is finite dimensional as \mathbb{F} -vector space, this*
 404 *dimension is called the **multiplicity of (α, β) as a zero of I** and is noted $\text{mult}((\alpha, \beta), I)$ [CLO05,*
 405 *§4.2].*

406 We call the **fiber** of a point $\mathbf{p} = (\alpha, \beta)$ the vertical line of equation $x = \alpha$. The **mul-**
 407 **tiplicity of \mathbf{p} in its fiber** with respect to a system of polynomials $\{P, Q\}$ in $\mathbb{F}[x, y]$, noted
 408 $\text{mult}_{\text{fiber}}((\alpha, \beta), \{P, Q\})$, is the multiplicity of β in the univariate polynomial $\text{gcd}(P(\alpha, y), Q(\alpha, y))$.⁵
 409 (This multiplicity is zero if P or Q does not vanish at \mathbf{p} .)

⁵The gcd is naturally considered over $\overline{\mathbb{F}(\alpha)[y]}$, the ring of polynomials in y with coefficients in the field extension of \mathbb{F} by α .

410 **Lemma 9.** *The multiplicity of any solution in its fiber with respect to the system $\{P, Q\}$ is smaller*
 411 *than or equal to its multiplicity in $\{P, Q\}$.*

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

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

413 Indeed, the first and last inclusions are trivial and the second one follows from the fact that
 414 $P(x, y) \in \langle P(\alpha, y), x - \alpha \rangle$ since $P(x, y)$ can be written as $P(\alpha, y) + \sum_{i \geq 1} \frac{\partial^i P(\alpha, y)}{\partial x^i} (x - \alpha)^i$. This ideal
 415 inclusion implies that the multiplicity of (α, β) in $\langle P, Q \rangle$ is larger than or equal to its multiplicity in
 416 $\langle x - \alpha, \gcd(P(\alpha, y), Q(\alpha, y)) \rangle$, which is equal to the multiplicity of β in $\gcd(P(\alpha, y), Q(\alpha, y))$ since
 417 $x - \alpha$ is squarefree. The latter is by definition the multiplicity in its fiber of the solution (α, β)
 418 with respect to the system $\{P, Q\}$. \square

419 4 Triangular decomposition and luckiness certificate

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

425 In Section 4.1, we recall Algorithm 1, the classical algorithm for computing the triangular decom-
 426 position of a zero-dimensional bivariate system $\{P, Q\}$. In Section 4.2, we present Algorithm 1',
 427 a variant that only computes the “degree” of the triangular decomposition. In Section 4.3, we
 428 present Algorithm 2, another variant that computes a luckiness certificate (see Definition 13) for
 429 the triangular decomposition. Finally, in Section 4.4, we present an amortized complexity analysis
 430 of these algorithms.

431 4.1 Triangular decomposition via subresultants

432 The idea is based on Lemma 2 which states that, after specialization at $x = \alpha$, the first (with
 433 respect to increasing i) non-zero subresultant $\text{Sres}_i(P, Q)(\alpha, y)$ is of degree i and is equal to the
 434 gcd of $P(\alpha, y)$ and $Q(\alpha, y)$. This induces a decomposition into triangular subsystems $\{A_i(x),$
 435 $\text{Sres}_i(P, Q)(x, y)\}$ where a solution α of $A_i(x) = 0$ is such that the system $\{P(\alpha, y), Q(\alpha, y)\}$ admits
 436 exactly i roots (counted with multiplicity), which are exactly those of $\text{Sres}_i(P, Q)(\alpha, y)$. Further-
 437 more, these triangular subsystems are regular chains, i.e., the leading coefficient of the bivariate
 438 polynomial (seen in y) is coprime with the univariate polynomial. We recall in Algorithm 1 how
 439 this decomposition is computed. Note that this algorithm performs $\tilde{O}(d^4)$ arithmetic operations.
 440 Indeed, the computation of the subresultant sequence has complexity $\tilde{O}(d^4)$ and there are at most

Algorithm 1 Triangular decomposition [GVEK96, LMMRS11]

Input: P, Q in $\mathbb{D}[x, y]$ coprime such that $\text{Lc}_y(P)$ and $\text{Lc}_y(Q)$ 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 $d_y(Q) \leq d_y(P)$
 - 2: Compute the subresultant sequence of P and Q with respect to y : $B_i = \text{Sres}_i(P, Q)$
 - 3: $G_0 = \text{squarefree part}(\text{Res}(P, Q))$ and $\mathcal{T} = \emptyset$
 - 4: **for** $i = 1$ **to** $d_y(Q)$ **do**
 - 5: $G_i = \text{gcd}(G_{i-1}, \text{sres}_i(P, Q))$
 - 6: $A_i = G_{i-1}/G_i$
 - 7: if $d_x(A_i) > 0$, add (A_i, B_i) to \mathcal{T}
 - 8: **return** $\mathcal{T} = \{(A_i(x), B_i(x, y))\}_{i \in \mathcal{I}}$
-

Algorithm 1' Degree of the triangular decomposition

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

Output: The degree of the triangular decomposition of $\{P, Q\}$

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

441 d gcd computations each of complexity $\tilde{O}(d^2)$ (see e.g. [BLPR15, Lemma 15] for details). The next
442 lemma summarizes the main properties of this triangular decomposition.

443 **Lemma 10** ([GVEK96, LMMRS11]). *Algorithm 1 computes a triangular decomposition $\{(A_i(x),$
444 $B_i(x, y))\}_{i \in \mathcal{I}}$ such that*

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

451 4.2 Degree of a triangular decomposition

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

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

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

Algorithm 2 Luckiness certificate

Input: P, Q in $\mathbb{Z}[x, y]$ coprime such that $\text{Lc}_y(P)$ and $\text{Lc}_y(Q)$ 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 $d_y(Q) \leq d_y(P)$
 - 2: Compute the sequence of principal subresultant coefficients of P and Q with respect to y : $\text{sres}_i(P, Q)$
 - 3: $G_0 = \text{squarefree part}(\text{Res}(P, Q))$
 - 4: $SG_0 = \text{sres}_{x,k}(\text{Res}(P, Q), \frac{\partial \text{Res}(P, Q)}{\partial x})$ the first non-null principal subresultant coefficient (for k increasing)
 - 5: **for** $i = 1$ **to** $d_y(Q)$ **do**
 - 6: $SG_i = \text{sres}_{x,k}(G_{i-1}, \text{sres}_i(P, Q))$ the first non-null principal subresultant coefficient (for k increasing)
 - 7: $G_i = \text{gcd}(G_{i-1}, \text{sres}_i(P, Q))$
 - 8: $\Pi = \text{Lc}_x(\text{Lc}_y(P)) \cdot \text{Lc}_x(\text{Lc}_y(Q)) \cdot \text{Res}_x(\text{Lc}_y(P), \text{Lc}_y(Q)) \cdot d_x(\text{Res}(P, Q)) \cdot \prod_{i=0}^{d_y(Q)} SG_i \cdot \text{Lc}_x(\text{sres}_i(P, Q))$
 - 9: **return** Π
-

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

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

469 *Proof.* Let $\{(A_i(x), B_i(x, y))\}_{i \in \mathcal{I}}$ denote the triangular decomposition of $\{P, Q\}$. By Lemma 10,
470 $B_i(x, y)$ 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
471 thus its degree is $d_x(G_{i-1}) - d_x(G_i)$. It follows that the degree of the triangular decomposition is
472 $\sum_{i \in \mathcal{I}} (d_x(G_{i-1}) - d_x(G_i)) i$. \square

473 4.3 Lucky primes for a triangular decomposition

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

477 **Definition 13.** *A prime μ is lucky for the triangular decomposition of Algorithm 1 applied*
478 *to P and Q if the decomposition commutes with the morphism ϕ_μ and its degree is invariant*
479 *through ϕ_μ .*⁶

⁶More precisely, if $\{(A_i, B_i)\}_{i \in \mathcal{I}} = \text{Algorithm 1}(P, Q)$ and $\{(A_i^\mu, B_i^\mu)\}_{i \in \mathcal{I}^\mu} = \text{Algorithm 1}(\phi_\mu(P), \phi_\mu(Q))$, then $\mathcal{I} = \mathcal{I}^\mu$, $\phi_\mu(A_i) = A_i^\mu$, $\phi_\mu(B_i) = B_i^\mu$ for every $i \in \mathcal{I}$ and the two triangular decompositions have the same degree. Note that (P, Q) and $(\phi_\mu(P), \phi_\mu(Q))$ are also required to satisfy the hypotheses of Algorithm 1.

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

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

487 We first prove that $\phi(P)$ and $\phi(Q)$ satisfy the conditions of Algorithm 1, that is that they are
488 coprime and that their leading coefficients are coprime. Observe first that $\phi(\text{Lc}_y(P)) = \text{Lc}_y(\phi(P))$
489 since μ does not divide $\text{Lc}_x(\text{Lc}_y(P))$, and similiraly for Q . Furthermore, since μ does not di-
490 vide the leading coefficients of $\text{Lc}_y(P)$ and $\text{Lc}_y(Q)$, their resultant and ϕ commute. Hence,
491 $\phi(\text{Res}_x(\text{Lc}_y(P), \text{Lc}_y(Q))) = \text{Res}_x(\text{Lc}_y(\phi(P)), \text{Lc}_y(\phi(Q)))$ and, since the left-hand-side term is non-
492 zero by assumption, so is the right-hand side, which means that that the leading coefficients of
493 $\phi(P)$ and $\phi(Q)$ are coprime. Furthermore, since μ does not divide $\text{sres}_0(P, Q) = \text{Res}(P, Q)$, we also
494 have that $\text{Res}(\phi(P), \phi(Q)) \neq 0$. We have proved that $\phi(P)$ and $\phi(Q)$ have a non-zero resultant and
495 that their leading coefficients are coprime, which implies that $\phi(P)$ and $\phi(Q)$ are coprime. Hence,
496 they satisfy the conditions of Algorithm 1.

497 We now prove that $\phi(A_i) = A_i^\mu$, $\phi(B_i) = B_i^\mu$ and $d_x(A_i) = d_x(A_i^\mu)$ for all $i > 0$. Since μ divides
498 neither the leadings of P nor Q , the specialization property of the subresultant polynomials writes
499 as $\phi(B_i) = \phi(\text{Sres}_i(P, Q)) = \text{Sres}_i(\phi(P), \phi(Q)) = B_i^\mu$. We now show by induction on $i \geq 0$ that
500 $\phi(G_i) = G_i^\mu$ and $d_x(G_i) = d_x(G_i^\mu)$, which implies that $\phi(A_i) = A_i^\mu$ and $d_x(A_i) = d_x(A_i^\mu)$ for $i > 0$
501 since $A_i = G_{i-1}/G_i$.

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

508 We now prove that $d_x(G_0) = d_x(G_0^\mu)$, which is now equivalent to proving that $d_x(G_0) =$
509 $d_x(\phi(G_0))$. Since the image through ϕ of any polynomial does not increase its degree, $d_x(\phi(G_0)) \leq$
510 $d_x(G_0)$. Furthermore, $d_x(\phi(G_0)) \geq d_x(G_0)$, because $G_0 = \frac{\text{Res}(P, Q)}{\text{gcd}(\text{Res}(P, Q), \frac{\partial \text{Res}(P, Q)}{\partial x})}$ and $\text{Res}(P, Q)$
511 and its image through ϕ have the same degree (since μ does not divide the leading coefficient of
512 $\text{Res}(P, Q)$).

513 **Case $i > 0$.** We assume that $\phi(G_{i-1}) = G_{i-1}^\mu$ and $d_x(G_{i-1}) = d_x(G_{i-1}^\mu)$. By Lemma 7, μ is lucky
514 for the gcd of G_{i-1} and $\text{sres}_i(P, Q)$. Indeed, μ divides none of the leading coefficients of G_{i-1} and
515 $\text{sres}_i(P, Q)$ (since G_{i-1} is a factor of $\text{sres}_{i-1}(P, Q)$), and μ does not divide SG_i either. This implies,
516 still by Lemma 7, that $\phi(G_i) = \phi(\text{gcd}(G_{i-1}, \text{sres}_i(P, Q))) = \text{gcd}(\phi(G_{i-1}), \phi(\text{sres}_i(P, Q)))$. This is
517 also equal to $\text{gcd}(G_{i-1}^\mu, \text{sres}_i(\phi(P), \phi(Q))) = G_i^\mu$ by the induction hypothesis and the property of
518 specialization of the subresultants. Hence, $\phi(G_i) = G_i^\mu$. Furthermore, since μ is lucky for the gcd
519 of G_{i-1} and $\text{sres}_i(P, Q)$, this gcd, which is G_i by definition, and $\text{gcd}(\phi(G_{i-1}), \phi(\text{sres}_i(P, Q))) = G_i^\mu$
520 have the same degree by Definition 6. This concludes the proof of the induction.

521 We have proved that $\phi(A_i) = A_i^\mu$, $\phi(B_i) = B_i^\mu$ and $d_x(A_i) = d_x(A_i^\mu)$ for all $i > 0$. The latter
522 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
523 equal to i by Lemma 10 (and Definition 1). This implies that the degrees of the decompositions
524 $\{(A_i, B_i)\}_{i \in \mathcal{I}}$ and $\{(A_i^\mu, B_i^\mu)\}_{i \in \mathcal{I}^\mu}$ are the same, which concludes the proof.

525 □

526 **4.4 Amortized complexity analysis**

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

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

- 531 • $d_i \leq \frac{d^2}{i+1}$ and $\tau_i = \tilde{O}(\frac{d^2+d\tau}{i+1})$,
- 532 • $\sum_{i=0}^{d_y(Q)-1} d_i \leq d^2$ and $\sum_{i=0}^{d_y(Q)-1} \tau_i = \tilde{O}(d^2 + d\tau)$.

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

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

535 up to the multiplication by a constant factor.

536 By Definition 8, the multiplicity of (α, β) in its fiber with respect to system $\{P, Q\}$, denoted by
537 $\text{mult}_{\text{fiber}}((\alpha, \beta), \{P, Q\})$, is the multiplicity of β in the univariate polynomial $\gcd(P(\alpha, y), Q(\alpha, y))$.
538 Thus, for any root α of A_i ,

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

539 According to Lemma 9, the multiplicity of a solution in its fiber is smaller than its multiplicity in
540 the system thus, for any root α of A_i ,

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

541 This latter sum is the multiplicity of α in the resultant $\text{Res}(P, Q)$ because the set of solutions of
542 $\{P, Q\}$ 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
543 the multiplicity in $\text{Res}(P, Q)$ of any root α of A_i is at least i and since A_i is squarefree (Lemma 10),
544 A_i^i divides $\text{Res}(P, Q)$. In addition, the A_i are pairwise coprime thus $\prod_{i=1}^{d_y(Q)} A_i^i$ divides $\text{Res}(P, Q)$.
545 On the other hand, $G_i = \prod_{j>i} A_j$ by construction, thus $\prod_{i=0}^{d_y(Q)-1} G_i = \prod_{i=1}^{d_y(Q)} A_i^i$ divides $\text{Res}(P, Q)$.

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

549 For proving the bounds on the bitsize of G_i , we introduce Mahler's measure. For
550 a univariate polynomial f with integer coefficients, its Mahler measure is $M(f) =$
551 $|\text{Lc}(f)| \prod_{z_i \text{ s.t. } f(z_i)=0} \max(1, |z_i|)$, where every complex root appears with its multiplicity. Mahler's
552 measure is multiplicative: $M(fg) = M(f)M(g)$ and, since it is at least 1 for any polynomial with in-
553 teger coefficients, f divides g implies that $M(g) \geq M(f)$. We also prove two inequalities connecting
554 the bitsize τ and degree d of f and its Mahler measure $M(f)$.

- 555 (i) $\tau \leq 1 + d + \log M(f)$. Indeed, [BPR06, Prop. 10.8] states that $\|f\|_1 \leq 2^d M(f)$, thus $\|f\|_\infty \leq$
556 $2^d M(f)$ and $\log \|f\|_\infty \leq d + \log M(f)$, which yields the result since $\tau = \lfloor \log \|f\|_\infty \rfloor + 1$.
- 557 (ii) $\log M(f) = O(\tau + \log d)$. Indeed, [BPR06, Prop. 10.9] states that $M(f) \leq \|f\|_2$, thus
558 $M(f) \leq \sqrt{d+1} \|f\|_\infty$ and $\log M(f) \leq \log \sqrt{d+1} + \log \|f\|_\infty$.

559 The fact that G_i^{i+1} divides $\text{Res}(P, Q)$ implies that $M(G_i)^{i+1} \leq M(\text{Res}(P, Q))$ and thus that
560 $\log M(G_i) \leq \frac{\log M(\text{Res}(P, Q))}{i+1}$. Inequality (i) together with $d_i \leq \frac{d^2}{i+1}$ then yields

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

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

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

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

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

573 In their loops, the three algorithms perform (in total) the computations of at most d gcd (or
574 sequences of principal subresultant coefficients) between polynomials G_{i-1} and sres_i . Polynomial
575 sres_i has bitsize $\tilde{O}(d\tau)$ and degree $O(d^2)$, and denoting by τ_i and d_i the bitsize and degree of
576 G_i , Lemma 4 yields a complexity in $\tilde{O}_B(d^2(d^2\tau_{i-1} + d_{i-1}d\tau))$ for the computation of G_i and SG_i .
577 According to Proposition 15, these complexities sum up over all i to $\tilde{O}_B(d^6 + d^5\tau)$. Finally, in
578 Line 6 of Algorithm 1, the division of G_{i-1} by G_i can be done in a bit complexity of the order
579 of the square of their maximum degree times their maximum bitsize [vzGG13, Theorem 9.6 and
580 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
581 10.21]). By Proposition 15, $d_i \leq d^2$ and $\tau_i = \tilde{O}(\frac{d^2+d\tau}{i+1})$, thus $\sum_i O_B(d_i^2\tau_i) = \tilde{O}_B(d^6 + d^5\tau)$. Hence
582 the worst-case bit complexity of all three algorithms is in $\tilde{O}_B(d^6 + d^5\tau)$.

583 We now show that the expected bit complexity of Algorithm 1' is in $\tilde{O}_B(d^5 + d^4\tau)$. As above
584 the worst-case bit complexity of Line 2 is in $\tilde{O}_B(d^4\tau)$. The rest of the algorithm performs $O(d)$ gcd
585 computations and one exact division (in Line 3) between polynomials of degree $O(d^2)$ and bitsize
586 $\tilde{O}(d\tau)$. Each of these operations can be done with an expected bit complexity of $\tilde{O}_B((d^2)^2 + d^2 \cdot d\tau)$
587 (the squared degree plus the degree times the bitsize) [vzGG13, Corollary 11.14 & Exercice 9.14].
588 The expected bit complexity of Algorithm 1' is thus in $\tilde{O}_B(d^5 + d^4\tau)$.

Concerning the last claim of the lemma, recall that

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

589 Since P and Q have degree at most d and bitsize at most τ , the first two terms have bitsize at most
590 τ and, by Lemma 4, $\text{Res}_x(\text{Lc}_y(P), \text{Lc}_y(Q))$ has bitsize $\tilde{O}(d\tau)$. Furthermore, as noted above, every
591 $\text{sres}_i(P, Q)$ (including the resultant of P and Q) has degree $O(d^2)$ and bitsize $\tilde{O}(d\tau)$. In
592 particular the bitsize of $\text{d}_x(\text{Res}(P, Q))$ is in $O(\log d)$ and that of $\text{Lc}_x(\text{sres}_i(P, Q))$ is in $\tilde{O}(d\tau)$. In
593 addition, still by Lemma 4, SG_0 has bitsize $\tilde{O}(d^2 \cdot d\tau)$. On the other hand, by Lemma 4, for $i \geq 1$,
594 SG_i has bitsize $\tilde{O}(d^2\tau_{i-1} + d_{i-1}d\tau)$ with d_i and τ_i the degree and bitsize of G_i . By Proposition 15,
595 these bitsizes sum up to $\tilde{O}(d^4 + d^3\tau)$. The bitsize of Π is bounded by the sum of all these bitsizes
596 and is thus in $\tilde{O}(d^4 + d^3\tau)$. \square

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

600 5 Separating linear form

601 This section presents a new algorithm of worst-case bit complexity $\tilde{O}_B(d^6 + d^5\tau)$ for computing a
 602 separating linear form for a bivariate system of two coprime polynomials P, Q in $\mathbb{Z}[x, y]$ of total
 603 degree at most d and maximum bitsize τ (Theorem 28). We also present a randomized version of
 604 this algorithm of expected bit complexity $\tilde{O}_B(d^5 + d^4\tau)$ (Theorem 29).

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

612 5.1 Notation and definitions

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

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

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

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

617 Note that these polynomials do not depend on t .

618 **Definition 18** ([BLPR15, Definition 8]). *A prime number μ is said to be **lucky for a zero-***
 619 ***dimensional system** $\{P, Q\}$ if $\{P, Q\}$ and $\{\phi_\mu(P), \phi_\mu(Q)\}$ have the same number of distinct*
 620 *solutions (in their respective algebraic closures), $\mu > 2d^4$ and $\phi_\mu(L_P(s)) \phi_\mu(L_Q(s)) \neq 0$.*

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

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

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

632 5.2 From a system to a curve

633 We prove here Proposition 22, which states that it is essentially equivalent from an asymptotic bit
 634 complexity point of view to compute a separating linear form for a system $\{P, Q\}$ and to compute
 635 a separating linear form for the critical points of a curve H . According to Definition 19, we refer to
 636 the latter as a separating linear form for H . The proof essentially follows that of [BLP⁺14, Lemma
 637 7] but we improve by a factor d the complexity of computing the curve H .

638 The critical points of a curve of equation H are the solutions of the system $\{H, \frac{\partial H}{\partial y}\}$, thus
 639 computing a separating linear form for a curve amounts, by definition, to computing a separating
 640 linear form for a system of two equations. Conversely, a separating linear form for the curve PQ
 641 is also separating for the system $\{P, Q\}$ since any solution of $\{P, Q\}$ is also solution of PQ and of
 642 $\frac{\partial PQ}{\partial y} = P\frac{\partial Q}{\partial y} + \frac{\partial P}{\partial y}Q$.

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

649 We analyze in the next lemma the complexity of computing a proper shearing of the coordinate
 650 system. Then, in the following lemma, we prove that the product of the squarefree parts of the two
 651 sheared polynomials does indeed define a curve for which separating linear forms exist and are all
 652 separating of the input system. Proposition 22 gather these two results.

653 **Lemma 20.** *Given P and Q in $\mathbb{Z}[x, y]$ of maximum degree d and maximum bitsize τ , we can
 654 compute a shearing of the coordinate system from (x, y) to $(t = x + \alpha y, y)$ with α an integer in
 655 $[0, 2d]$, such that the sheared polynomials \tilde{P} and \tilde{Q} have coefficients of bitsize $\tilde{O}(d + \tau)$ and have
 656 their leading coefficients $\text{Lc}_y(\tilde{P})$ and $\text{Lc}_y(\tilde{Q})$ in \mathbb{Z} . This computation can be done with $\tilde{O}_B(d^4 + d^3\tau)$
 657 bit operations in the worst case.*

658 *Proof.* We consider a generic shearing of the coordinate system from (x, y) to $(t = x + sy, y)$ in order
 659 to find a value $s = \alpha$ so that the sheared curves $\tilde{P}(t, y) = P(t - \alpha y, y)$ and $\tilde{Q}(t, y) = Q(t - \alpha y, y)$
 660 have no vertical asymptote, that is $\text{Lc}_y(\tilde{P})$ and $\text{Lc}_y(\tilde{Q})$ are in \mathbb{Z} .

661 The leading coefficient of $P(t - sy, y)$ (seen as a polynomial in y) is a polynomial of degree at
 662 most d in $\mathbb{Z}[s]$ (t does not appear in the leading term); furthermore an expanded form of $P(t - sy, y)$
 663 can be computed in complexity $\tilde{O}_B(d^4 + d^3\tau)$ and the coefficients have bitsize $\tilde{O}(d + \tau)$ (see e.g.
 664 [BLPR15, Lemma 7]). Finding an integer value $s = \alpha$ where the leading coefficient does not
 665 vanish can thus be done in d evaluations of complexity $\tilde{O}_B(d(d + \tau))$ each [BLPR15, Lemma 6]
 666 and such α can be found in $[0, d]$. Then, computing $P(t - \alpha y, y)$ can be done by evaluating each
 667 of the coefficients of $P(t - sy, y)$ at $s = \alpha$, which can again be done with $O(d^2)$ evaluations of
 668 complexity $\tilde{O}_B(d(d + \tau))$ each. Thus, we can shear the curve in complexity $\tilde{O}_B(d^4 + d^3\tau)$ so that
 669 the leading coefficient of the resulting polynomial $\tilde{P}(t, y) = P(t - \alpha y, y)$ (seen as a polynomial in
 670 y) is a constant. These computations can trivially be done for P and Q simultaneously and α
 671 in $[0, 2d]$. \square

672 **Lemma 21.** *Let P and Q be two coprime polynomials in $\mathbb{Z}[x, y]$ of maximum degree d , maximum
 673 bitsize τ , and with $\text{Lc}_y(P)$ and $\text{Lc}_y(Q)$ in \mathbb{Z} . The product, H , of the squarefree parts of P and Q
 674 is squarefree, it has degree at most $2d$, bitsize $\tilde{O}(d + \tau)$, with $\text{Lc}_y(H)$ in \mathbb{Z} , and it can be computed
 675 using $\tilde{O}_B(d^5 + d^4\tau)$ bit operations in the worst case. System $\{H, \frac{\partial H}{\partial y}\}$ is zero-dimensional and its
 676 separating forms are also separating for $\{P, Q\}$.*

Algorithm 3 Number of critical points of H

Input: H in $\mathbb{Z}[x, y]$ squarefree such that $\text{Lc}_y(H)$ 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, $\frac{\partial H}{\partial y}$)

677 *Proof.* The squarefree parts of P and Q can be computed in $\tilde{O}_B(d^5 + d^4\tau)$ bit complexity (by
678 Lemma 5) and they have bitsizes in $\tilde{O}(d + \tau)$ by Mignotte's lemma (see e.g. [BPR06, Corollary
679 10.12]). The sum of $O(d^2)$ terms of bitsize $\tilde{O}(d + \tau)$ increases the bitsize bound by $O(\log d^2)$, thus
680 H has coefficients of bitsizes $\tilde{O}(d + \tau)$. Thus, the $O(d^4)$ arithmetic operations for computing all
681 these coefficients can be done in $\tilde{O}_B(d^4(d + \tau))$ bit operations in the worst case.

682 Every solution of $\{P, Q\}$ is trivially solution of $\{H, \frac{\partial H}{\partial y}\}$, thus any separating linear form for
683 $\{H, \frac{\partial H}{\partial y}\}$ is separating for $\{P, Q\}$. Furthermore, since P and Q are coprime with constant leading
684 terms, H is squarefree with a constant leading term, which implies that H has finitely many critical
685 points, that is $\{H, \frac{\partial H}{\partial y}\}$ is zero-dimensional. \square

686 We can summarize the last two lemmas in the following proposition.

687 **Proposition 22.** *Let P and Q be two coprime polynomials in $\mathbb{Z}[x, y]$ of maximum degree d and*
688 *maximum bitsize τ . We can compute a shearing of the coordinate system from (x, y) to $(t =$
689 $x + \alpha y, y)$, with α an integer in $[0, 2d]$, and a squarefree polynomial H in $\mathbb{Z}[t, y]$ of degree at most*
690 *$2d$, bitsize $\tilde{O}(d + \tau)$, with $\text{Lc}_y(H)$ in \mathbb{Z} , so that any separating linear form for the zero-dimensional*
691 *system $\{H, \frac{\partial H}{\partial y}\}$ is also separating for $\{P, Q\}$ after being sheared back. The worst-case complexity*
692 *of this computation is in $\tilde{O}_B(d^5 + d^4\tau)$.*

693 5.3 Separating linear form of a curve

694 In this section, we consider an arbitrary curve defined by H in $\mathbb{Z}[x, y]$ of degree d and bitsize τ ,
695 squarefree and with a constant leading coefficient in y . In particular, the polynomial H defined in
696 Proposition 22 satisfies these two last conditions, which yield that the curve has a finite number of
697 critical points. We show in the following three subsections that computing (i) the number of the
698 critical points of H , (ii) a lucky prime for the system of critical points $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 18),
699 and finally (iii) a separating form for the curve H (Definition 19) can be done with a bit complexity
700 in $\tilde{O}_B(d^6 + d^5\tau)$.

701 5.3.1 Number of critical points

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

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

712 *Proof.* The complexity analysis of the algorithm directly follows from Proposition 16 noticing that
713 $\frac{\partial H}{\partial y}$ and $(\frac{\partial H}{\partial y})^2$ have degrees at most $2d$ and bitsizes $O(\tau + \log d)$ (since $\frac{\partial H}{\partial y}$ has bitsize $O(\tau + \log d)$
714 and that the sum of $O(d^2)$ terms of such bitsizes adds $O(\log d^2)$ to the bitsize bound, and that
715 $(\frac{\partial H}{\partial y})^2$ can be computed from $\frac{\partial H}{\partial y}$ in complexity $\tilde{O}_B(d^2\tau)$ [vzGG13, Cor. 8.28].

716 We now prove the correctness of the algorithm. Observe that systems $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$
717 satisfy the input requirements of Algorithm 1' since H is squarefree and $\text{Lc}_y(H)$ is in \mathbb{Z} .

718 We first prove that for any critical point (α, β) of H , the multiplicity of β in
719 $\text{gcd}(H(\alpha, y), (\frac{\partial H}{\partial y})^2(\alpha, y))$ is greater by one than the multiplicity of β in $\text{gcd}(H(\alpha, y), \frac{\partial H}{\partial y}(\alpha, y))$.
720 Since (α, β) is a critical point of H , it is solution of both the systems $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$.
721 This implies that β is a root of both $\text{gcd}(H(\alpha, y), \frac{\partial H}{\partial y}(\alpha, y))$ and $\text{gcd}(H(\alpha, y), (\frac{\partial H}{\partial y})^2(\alpha, y))$. If m
722 is the multiplicity of β in $H(\alpha, y)$ then β has multiplicity $m - 1$ in $\frac{\partial H}{\partial y}(\alpha, y)$ and thus, that it has
723 multiplicity $2m - 2$ in $(\frac{\partial H}{\partial y})^2$. It follows that β has multiplicity $m - 1$ in $\text{gcd}(H(\alpha, y), \frac{\partial H}{\partial y}(\alpha, y))$
724 and m in $\text{gcd}(H(\alpha, y), (\frac{\partial H}{\partial y})^2(\alpha, y))$ because $m \leq 2m - 2$, that is $m - 1 \geq 1$, since β is solution of
725 $\frac{\partial H}{\partial y}(\alpha, y)$.

726 We denote the multiplicity of β in $\text{gcd}(P(\alpha, y), Q(\alpha, y))$ as $\text{mult}(\beta, \text{gcd}(P(\alpha, y), Q(\alpha, y)))$. Sum-
727 ming over all the critical points of H and noticing that the set V_H of distinct solutions of $\{H, \frac{\partial H}{\partial y}\}$
728 is the same as that of $\{H, (\frac{\partial H}{\partial y})^2\}$, we obtain that the number of critical points is

$$\#V_H = \sum_{(\alpha, \beta) \in V_H} \text{mult}(\beta, \text{gcd}(H(\alpha, y), (\frac{\partial H}{\partial y})^2(\alpha, y))) - \sum_{(\alpha, \beta) \in V_H} \text{mult}(\beta, \text{gcd}(H(\alpha, y), \frac{\partial H}{\partial y}(\alpha, y))).$$

729 It remains to prove that this difference is equal to the difference of the degrees of the decom-
730 positions of $\{H, (\frac{\partial H}{\partial y})^2\}$ and $\{H, \frac{\partial H}{\partial y}\}$. More generally, we prove that the degree of the triangular
731 decomposition of $\{P, Q\}$ is equal to the sum, over all distinct solutions (α, β) of $\{P, Q\}$, of the mul-
732 ticipities of β in $\text{gcd}(P(\alpha, y), Q(\alpha, y))$. Indeed, by Lemma 10, the sets of solutions of the systems
733 of the triangular decomposition of Algorithm 1 are disjoint and polynomials A_i are squarefree. The
734 degree of the triangular decomposition of $\{P, Q\}$ is thus

$$\sum_{i \in \mathcal{I}} d_x(A_i(x)) d_y(B_i(x, y)) = \sum_{(\alpha, \beta) \in V} \text{mult}(\beta, B_i(\alpha, y)),$$

735 where V is the set of solutions of $\{P, Q\}$ and $\text{mult}(\beta, B_i(\alpha, y))$ denotes the multiplicity of β in
736 $B_i(\alpha, y)$. The claim follows since $B_i(\alpha, y) = \text{gcd}(P(\alpha, y), Q(\alpha, y))$ by Lemma 10 and this concludes
737 the proof of correctness of the algorithm. \square

738 5.3.2 Lucky prime for the system of critical points

739 Let $\Pi = \text{Algo 2}(H, \frac{\partial H}{\partial y}) \cdot \text{Algo 2}(H, (\frac{\partial H}{\partial y})^2)$ be the product of the luckiness certificates output by
740 Algo 2 for the triangular decompositions of $\{H, \frac{\partial H}{\partial y}\}$ and $\{H, (\frac{\partial H}{\partial y})^2\}$. Lemma 24 shows that we
741 can easily check using a divisibility test on Π whether a prime number is lucky for the system
742 $\{H, \frac{\partial H}{\partial y}\}$ (see Definition 18). Algorithm 4 finds such a lucky prime by an iterative application of
743 this divisibility test. To keep the complexity in the desired bound, the primes to be tested are
744 grouped and a remainder tree is used for the computation of the reduction of Π modulo all the
745 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.

746 **Lemma 24.** *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
747 divide Π then 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 $\text{Lc}_y(H)$ 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 $\Pi, L_H, L_{\frac{\partial H}{\partial y}}$
 - 7: **if** $\phi_\mu(\Pi) \phi_\mu(L_H(s)) \phi_\mu(L_{\frac{\partial H}{\partial y}}(s)) \neq 0$ **then**
 - 8: **return** μ
 - 9: $m =$ the largest prime in B
-

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

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

756 *Proof.* The correctness of Algorithm 4 follows directly from Lemma 24 since the condition in Line 7
757 (together with $\mu > 2d^4$) matches exactly the assumptions of Lemma 24.

758 We now analyze the complexity of this algorithm. It is straightforward that, in Line 1, $L_H(s)$
759 and $L_{\frac{\partial H}{\partial y}}(s)$ can be computed with $\tilde{O}_B(d^4 + d^3\tau)$ bit operations and that they have coefficients of
760 bitsizes $\tilde{O}(d + \tau)$ (see e.g. [BLPR15, Lemma 7]). Furthermore, since Π has bitsize $\tilde{O}(d^4 + d^3\tau)$ (by
761 Proposition 16), the number of prime divisors of $\Pi, L_H(s)$, and $L_{\frac{\partial H}{\partial y}}(s)$ is in $\tilde{O}(d^4 + d^3\tau)$. Hence,
762 the number of iterations of the loop in Line 3 is polylogarithmic in d and τ .

763 Each iteration of this loop consists in testing, for the $d^4 + d^3\tau$ primes in B , the non-vanishing of
764 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
765 all these coefficients has bitsize $\tilde{O}(d^4 + d^3\tau)$ and it can be computed in bit complexity $\tilde{O}_B(d^4 + d^3\tau)$.
766 The reduction of this product modulo all the primes in B can be computed via a remainder tree
767 in a bit complexity that is soft linear in the total bitsize of the input [MB74, Theorem 1], which is
768 in $\tilde{O}(d^4 + d^3\tau)$ because the sum of the bitsizes of the $d^4 + d^3\tau$ primes in B is also in $\tilde{O}(d^4 + d^3\tau)$
769 since each of these primes has bitsize $O(\text{polylog}(d\tau))$ (since there are $O(\text{polylog}(d\tau))$ iterations of
770 the loop in Line 3). Hence, the bit complexity of one iteration of the loop of Line 3 is $\tilde{O}_B(d^4 + d^3\tau)$
771 and since at most $O(\text{polylog}(d\tau))$ iterations are performed, the overall bit complexity of Algorithm
772 4 is in $\tilde{O}_B(d^4 + d^3\tau)$.

773 Finally, the bitsize of every considered prime is in $O(\log(d^4 + d^3\tau))$. Indeed, the number of
774 loop iterations is in $O(\text{polylog}(d\tau))$, thus the algorithm considers the first $O((d^4 + d^3\tau)\text{polylog}(d\tau))$

Algorithm 5 Separating form of a curve

Input: H in $\mathbb{Z}[x, y]$ of degree d and bitsize τ , squarefree and such that $\text{Lc}_y(H)$ is in \mathbb{Z}

Output: A separating linear form $x + ay$ of the curve H , with $a < 2d^4$

- 1: Compute $N = \text{Algorithm 3}(H)$, the number of distinct (complex) critical points of H
 - 2: Compute $\Pi = \text{Algo 2}(H, \frac{\partial H}{\partial y}) \cdot \text{Algo 2}(H, (\frac{\partial H}{\partial y})^2)$, 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 $\mu = \text{Algorithm 4}(H, \Pi)$, 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 $\{0, \dots, 2d^4\}$ 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** $\Upsilon_\mu(a) \neq 0^7$ and $N_a = N$
 - 13: **return** The linear form $x + ay$
-

775 first primes. The largest considered prime is thus in $\tilde{O}(d^4 + d^3\tau)$ [vzGG13, Theorem 18.10] and its
776 bitsize is thus in $O(\log d\tau)$. □

777 5.3.3 Separating linear form of a curve

778 In this section, we assume that we have already computed, using Algorithms 3 and 4, the number
779 of distinct (complex) critical points of a curve and a lucky prime μ for the system of critical points.
780 With this information, Algorithm 4 of [BLPR15] computes a separating form with a bit complexity
781 $\tilde{O}_B(d^8 + d^7\tau)$. In this section, we slightly modify this algorithm to improve its complexity to
782 $\tilde{O}_B(d^6 + d^5\tau)$.

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

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

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

795 *Proof.* We first prove the correctness of Algorithm 5 which essentially follows from [BLPR15, Al-
796 gorithm 4]. The only relevant difference for the correctness is the way to compute $R_\mu(t, s)$. In
797 [BLPR15], $R_\mu(t, s)$ is computed by computing the resultant $R(t, s)$ of $H(t - sy, y)$ and $\frac{\partial H}{\partial y}(t - sy, y)$,

⁷ $\Upsilon_\mu(s) \in \mathbb{Z}_\mu[s]$ and we consider $\Upsilon_\mu(a)$ in \mathbb{Z}_μ .

798 and reducing it modulo μ . Here, we first reduce the polynomials modulo μ before computing the
799 resultant. This yields the same result since μ is known to be lucky for the system $\{H, \frac{\partial H}{\partial y}\}$, thus it
800 does not divide the leading terms $L_H(s)$ and $L_{\frac{\partial H}{\partial y}}(s)$. This proves the correctness of Algorithm 5.
801 Note, furthermore, that the correctness of [BLPR15, Algorithm 4] also implies that the value a
802 output by our algorithm is less than $2d^4 - 2d$.⁸

803 We now prove the complexity of our algorithm. First, observe that, as argued in the proof of
804 Proposition 23, $\frac{\partial H}{\partial y}$ and $(\frac{\partial H}{\partial y})^2$ have degrees at most $2d$, bitsizes $O(\tau + \log d)$, and that they can
805 be computed in complexity $\tilde{O}_B(d^2\tau)$. Furthermore, in Lines 1–3, the input of Algorithms 2, 3, and
806 4 satisfy the requirements of these algorithms, since H is squarefree with $\text{Lc}_y(H)$ in \mathbb{Z} . The bit
807 complexity of Lines 1–3 is thus in $\tilde{O}_B(d^6 + d^5\tau)$ by Propositions 16, 23, 25.

808 It is straightforward that, in Line 4, the sheared polynomials $H(t - sy, y)$ and $\frac{\partial H}{\partial y}(t - sy, y)$
809 can be computed in bit complexity $\tilde{O}_B(d^4 + d^3\tau)$ and that their bitsizes are in $\tilde{O}(d + \tau)$ (see e.g.
810 [BLPR15, Lemma 7]). In Lines 5 and 6, the polynomials to be reduced modulo μ , in one or three
811 variables, have degree at most d and bitsize $\tilde{O}(d + \tau)$. The reduction of each of their $O(d^3)$
812 coefficients modulo μ can be done in a bit complexity that is softly linear in the maximum bitsizes
813 [vzGG13, Theorem 9.8], that is in a total bit complexity of $\tilde{O}_B(d^4 + d^3\tau)$. Then, computing in
814 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
815 operations in \mathbb{Z}_μ .

816 The resultant in Line 6 can be computed in $O(d^5)$ arithmetic operations in \mathbb{Z}_μ (see Lemma 3).
817 In Line 7, $R_\mu(t, s)$ is a polynomial of degree $O(d^2)$ in t with coefficients in s of degree $O(d^2)$.
818 The arithmetic complexity, in \mathbb{Z}_μ , of the evaluation of one such coefficient at $s = a$ is linear in
819 its degree (using for instance Horner’s scheme) but, using multipoint evaluation, the arithmetic
820 complexity of the evaluation of one such coefficient at $O(d^2)$ values is in $\tilde{O}(d^2)$ [vzGG13, Corollary
821 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
822 done with $\tilde{O}(d^4)$ arithmetic operations in \mathbb{Z}_μ . The overall arithmetic complexity of Line 7 is thus
823 $\tilde{O}(d^6)$. In Line 10, since $R_\mu(t, a)$ has degree $O(d^2)$, its squarefree part can be computed with $\tilde{O}(d^2)$
824 arithmetic operations in \mathbb{Z}_μ (see Lemma 4) and, in Line 12, each evaluation of $\Upsilon(a)$ can be done in
825 $O(d)$ arithmetic operations since Υ has degree $O(d)$. Furthermore, since the algorithm stops with
826 $a < 2d^4$, the arithmetic complexity of the whole loop is in $\tilde{O}(d^6)$.

827 We have shown that Lines 6 to 12 perform $\tilde{O}(d^6)$ arithmetic operations in \mathbb{Z}_μ . Since μ has
828 bitsize $O(\log d\tau)$, the bit complexity of these lines is in $O_B(d^6 \text{polylog}(d\tau))$, which concludes the
829 proof. \square

830 **Remark 27.** *From a worst-case complexity point of view, the knowledge of the number N of*
831 *(distinct) complex critical points of the input curve in Algorithm 5 is not mandatory since one*
832 *could instead compute the number of solutions N_a of $R_\mu(t, a)$ for all integers a smaller than $2d^4$*
833 *and output a value of a that maximizes N_a . However, knowing N , the algorithm can stop as soon*
834 *as a value of a is found such that $N_a = N$, which improves the expected complexity of the algorithm*
835 *in a Las-Vegas setting, as discussed in Section 5.5.*

836 5.4 Separating linear form of a system

837 Propositions 22 and 26 directly yield the following theorem where the separating form is obtained
838 by shearing back the separating form output by Algorithm 5.

⁸[BLPR15, Theorem 19] is stated with $a < 2d^4$ but its proof establishes $a \leq \binom{d^2}{2} + 2(d^2 + d)$ 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 28.

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 $\text{Lc}_y(H)$ is in \mathbb{Z}

Output: A separating linear form $x + ay$ of the curve H , with $a < 2d^4$

- 1: Compute $N = \text{Algorithm 3}(H)$, 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: $M = 2M$
 - 6: Choose uniformly at random an integer a in $[0, 2d^4 - 2d]$ and a prime μ in $(2d^4, M)$
 - 7: Compute $\Upsilon_\mu(a) = \phi_\mu(\Upsilon)(a)$
 - 8: 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: Compute the degree⁹ N_a of the squarefree part of $R_{\mu,a}(t)$
 - 10: **until** $\Upsilon_\mu(a) \neq 0$ ⁷ and $N_a = N$
 - 11: **return** The linear form $x + ay$
-

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

843 *Proof.* The first statement of the theorem follows directly from Propositions 22 and 26 where
844 the integer b is the sum of the integers α and a defined in these propositions. We prove below
845 the second statement. The integer a computed by Algorithm 5 is such that $\Upsilon(a) \neq 0$ and thus
846 $L_H(a) \neq 0$. Since $L_H(a) \in \mathbb{Z}$ is non-zero, it is the leading coefficient in y of the sheared polynomial
847 $H(t - ay, y)$. H is the product of the squarefree parts of the sheared polynomials \tilde{P} and \tilde{Q} where
848 $\tilde{P}(t, y) = P(t - \alpha y, y)$ and similarly for \tilde{Q} (see Lemmas 20 and 21). Hence, the leading coefficient
849 in y of the sheared polynomial $\tilde{P}(t - ay, y) = P(t - ay - \alpha y, y) = P(t - by, y)$ divides $L_H(a)$, which
850 is an integer. Similarly for \tilde{Q} . \square

851 5.5 Las-Vegas algorithm

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

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

858 Our Las-Vegas algorithm is obtained from our deterministic version by only modifying Algo-
859 rithm 5 into a randomized version, Algorithm 5'. The main difference between these two versions
860 is that, in Algorithm 5', we choose randomly a candidate separating linear form $x + ay$ and a
861 candidate lucky prime μ for $\{H, \frac{\partial H}{\partial y}\}$ (Definition 18) until the degree N_a of the squarefree part of

⁹We 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 $R_\mu(t, a) \neq 0$.

862 $R_\mu(t, a)$ is equal to the known number of solutions N . If a and μ are chosen randomly in sufficiently
863 large sets, the probability that $x + ay$ is separating and that μ is lucky is larger than a positive
864 constant, which implies that the expected number of such choices is a constant.

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

874 We now prove the correctness and complexity of Algorithm 5' in the two following lemmas and in
875 Proposition 33, which, together with Proposition 22, yield Theorem 29 similarly as for Theorem 28.

876 **Lemma 30** (Correctness of Algorithm 5'). *Algorithm 5' terminates if and only if the values of the*
877 *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*
878 *for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$, which implies that $x + ay$ is also separating for $\{H, \frac{\partial H}{\partial y}\}$.*

879 *Proof.* The proof relies on Lemma 10 and Propositions 9 and 12 in [BLPR15] which, together,
880 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.
881 We first prove that these hypotheses are satisfied when either side of the if-and-only-if claim holds
882 in the statement of the lemma.

883 First, $\Upsilon_\mu(a) \neq 0$ follows from Line 10 if Algorithm 5' terminates and it appears in the right
884 hand side of the if-and-only-if claim. Second, $a \leq 2d^4 < \mu$ by definition of a and μ (Line 6). Finally,
885 if Algorithm 5' terminates, $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is zero dimensional because, otherwise, $R_{\mu,a}(t) \equiv 0$,
886 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
887 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
888 number of solution as $\{H, \frac{\partial H}{\partial y}\}$.

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

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

890 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
891 $\#V(I)$ and $\#V(I_\mu)$ are the number of distinct solutions of the systems $\{H, \frac{\partial H}{\partial y}\}$ and $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$,
892 respectively.

893 Assume for now that $R_\mu(t, a) = R_{\mu,a}(t)$ (as defined in Line 8). This implies that
894 $d_t(\text{squarefree part}(R_\mu(t, a))) = N_a$ (Line 9). Since $N = \#V(I)$ by definition (Line 1), the al-
895 gorithm terminates if and only if a and μ are such that $\Upsilon_\mu(a) \neq 0$ and $N_a = N$ (Line 10), which
896 is equivalent to $\Upsilon_\mu(a) \neq 0$ and

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

897 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
898 10] and the second equality holds if and only if μ is lucky for the system $\{H, \frac{\partial H}{\partial y}\}$ (Definition 18).
899 This proves the if-and-only-if claim of the lemma (assuming that $R_\mu(t, a) = R_{\mu,a}(t)$). Furthermore,
900 when both equalities hold, $x + ay$ is also separating for the system $\{H, \frac{\partial H}{\partial y}\}$ by Proposition 9 in
901 [BLPR15].

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

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

904 This equality holds if the polynomials in the left-hand side resultant are such that their leading
 905 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
 906 that $\Upsilon_\mu(a) \neq 0$. Indeed, $\Upsilon_\mu(a) \neq 0$ implies $\phi_\mu(L_H(a)) \neq 0$. Then, $\phi_\mu(L_H(s)) \neq 0$ implies
 907 $\phi_\mu(L_H(s)) = L_{\phi_\mu(H)}(s)$ 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
 908 the proof. \square

909 **Lemma 31.** *The expected number of iterations of the loop in Algorithm 5' is in $O(\log d\tau)$. More
 910 precisely, after $O(\log d\tau)$ iterations, the probability that the algorithm terminates is at least $1/8$ at
 911 every iteration.*

912 *Proof.* The number of unlucky primes for $\{H, \frac{\partial H}{\partial y}\}$ is in $\tilde{O}(d^4 + d^3\tau)$ [BLPR15, Proposition 13].
 913 Let $K(d, \tau)$ in $\tilde{O}(d^4 + d^3\tau)$ be an upper bound on the number of unlucky primes, which we denote
 914 for simplicity by K . If the algorithm terminates with a value of M such that $\frac{M/2}{2 \ln M/2} \leq 2K$, the
 915 number of loop iterations is in $O(\log d\tau)$. Indeed, the number of iterations is less than $\log M$ which
 916 is in $O(\log K)$ since $\sqrt{M/2} < \frac{M/2}{\ln M/2} \leq 4K$. It is thus sufficient to prove that, for any iteration such
 917 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
 918 this implies that the expected number of such iterations is at most 8 and thus that the expected
 919 number of all iterations in the loop is in $O(\log d\tau)$.

920 We can thus assume that, in Line 6, μ is chosen uniformly at random in a set of primes of
 921 cardinality at least $2K$. Indeed, μ is chosen in $(2d^4, M) \supseteq (M/2, M)$ and the number of primes in
 922 $(M/2, M)$ is at least $\frac{M/2}{2 \ln M/2}$ [vzGG13, Theorem 18.7 (see also Exercise 18.18)].

923 By Lemma 30, the algorithm terminates if and only if a and μ are such that $\Upsilon_\mu(a) \neq 0$, μ is
 924 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
 925 probability that these three events simultaneously occur. We have

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

926 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
 927 uniformly at random in a set of primes of cardinality at least $2K$ and there are at most K unlucky
 928 primes.

929 The conditional probability that $x + ay$ is separating for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is also at least $1/2$.
 930 Indeed, we prove that the conditional probability that $x + ay$ is *not* separating for that system is
 931 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
 932 same number of distinct solutions as $\{H, \frac{\partial H}{\partial y}\}$, number which is at most d^2 solutions by Bézout's
 933 bound. Thus, for any choice of a lucky μ , there are at most $\binom{d^2}{2} < d^4 - d$ directions in which two
 934 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 $x + ay$

935 is not separating for that system. Since a is chosen uniformly at random in a set of cardinality
 936 $2d^4 - 2d + 1$, the conditional probability that $x + ay$ is not separating for $\{\phi_\mu(H), \phi_\mu(\frac{\partial H}{\partial y})\}$ is thus
 937 at most $d^4 - d$ times the number of choices of a lucky μ over the number of choices of couples of a
 938 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
 939 thus proves the claim.

940 Finally, we show that the conditional probability that $\Upsilon_\mu(a) \neq 0$ is also at least $1/2$. Given
 941 that μ is lucky, $\Upsilon_\mu(s) \neq 0$, by Definition 18. Thus, for any given lucky μ , $\Upsilon_\mu(s)$ has degree at most
 942 $2d$ and it vanishes for at most $2d$ values of a . The conditional probability that $\Upsilon_\mu(a) = 0$ is thus at
 943 most $2d$ times the number of choices of a lucky μ over the number of choices of couples of a lucky μ
 944 and a value a such that $x + ay$ is separating. This probability is thus equal to $2d$ over the number
 945 of choices of such values a . The number of such choices for a is at least d^4 since a is considered in
 946 $[0, 2d^4 - 2d]$ and there are at most $\binom{d^2}{2} < d^4 - 2d$ choices for which $x + ay$ is not separating. Hence
 947 the conditional probability that $\Upsilon_\mu(a) = 0$ is at most $2d/d^4$, which is less than $1/2$ for $d \geq 2$. This
 948 proves the claim that the conditional probability that $\Upsilon_\mu(a) \neq 0$ is at least $1/2$ and concludes the
 949 proof. \square

950 The next lemma factorizes a technical part of the expected complexity analysis of Proposi-
 951 tions 33 and 50.

952 **Lemma 32.** *If a while-loop is such that (i) the expected bit complexity of the i -th iteration is*
 953 *$\tilde{O}_B(Ai^k)$ where A is a polynomial in the input parameter sizes and (ii) the probability that the loop*
 954 *ends at the i -th iteration, given that it has not stopped before, is at least a constant $c > 0$, then the*
 955 *expected bit complexity of the entire loop is $\tilde{O}_B(A)$.*

956 *Proof.* Let $x_j > c$ be the probability that the loop stops at the j -th iteration given that it has not
 957 yet stopped before. The probability that the loop stops at the i -th iteration is $x_i \prod_{j=1}^{i-1} (1 - x_j) \leq$
 958 $(1 - c)^{i-1}$. On the other hand, the total expected bit complexity of all the iterations until the i -th
 959 is $\tilde{O}_B(Ai^{k+1})$. Hence the total expected bit complexity of the entire loop is

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

960 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
 961 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,
 962 the expected bit complexity of the entire loop is in $\tilde{O}_B(A)$. \square

963 **Proposition 33.** *Given H in $\mathbb{Z}[x, y]$ of degree d and bitsize τ , Algorithm 5' computes, with an*
 964 *expected bit complexity $\tilde{O}_B(d^5 + d^4\tau)$, an integer a in $[0, 2d^4 - 2d]$ such that the linear form $x + ay$*
 965 *is separating for the critical points of H .*

966 *Proof.* By Proposition 23, Line 1 has expected complexity $\tilde{O}_B(d^5 + d^4\tau)$. In Line 2, similarly as
 967 in Line 4 of Algorithm 5, $H(t - sy, y)$ and $\frac{\partial H}{\partial y}(t - sy, y)$ have coefficients of bitsize $\tilde{O}(d + \tau)$ and
 968 they can be computed with $\tilde{O}_B(d^4 + d^3\tau)$ bit operations (see the proof of Proposition 26). Still in
 969 Line 2, $\Upsilon(s)$ can be computed with $O(d^2)$ arithmetic operations on integers of bitsize $O(\tau + \log d)$,
 970 and thus with $\tilde{O}_B(d^2\tau)$ bit operations. The worst-case bit complexity of Lines 1 and 2 is thus in
 971 $\tilde{O}_B(d^5 + d^4\tau)$.

972 Consider now one iteration of the loop in Algorithm 5' and let I_M denote the interval $(2d^4, M)$.
 973 In Line 6, we can compute a prime μ by choosing uniformly at random an integer in I_M and testing
 974 whether it is prime until a prime is found. Finding a random integer smaller than M amounts

975 to computing a sequence of $\log M$ random bits, which we assume can be done in $O_B(\log M)$ bit
976 operations. A random integer smaller than M is larger than $2d^4$ with probability at least $1/2$,
977 thus a random integer in I_M can be computed in $O_B(\log M)$ bit operations. The number of primes
978 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
979 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
980 thus found after at most $4 \ln M/2$ trials on average. Testing whether an integer in I_M is prime
981 can be done with a polynomial bit complexity in the bitsize of M , $\tilde{O}_B(\log^{7.5} M)$ [AKS04]. The
982 expected bit complexity of computing a prime in Line 6 is thus in $\tilde{O}_B(\log^{8.5} M)$. Furthermore,
983 since a random integer a in $[0, 2d^4]$ can be computed in $\tilde{O}_B(\log d)$ bit operations, the expected bit
984 complexity of one iteration of Line 6 is in $\tilde{O}_B(\log^{8.5} M)$.

985 In Line 7, $O(d)$ coefficients of bitsize $O(\tau + \log d)$ are reduced modulo μ . Each reduction can
986 be done in a bit complexity that is softly linear in the maximum bitsizes [vzGG13, Theorem 9.8],
987 that is in a total bit complexity of $\tilde{O}_B(d(\tau + \log d + \log M))$. Evaluating $\Upsilon(s)$ at a can then be
988 done with $O(d)$ arithmetic operations in \mathbb{Z}_μ and thus with $O_B(d \log M)$ bit operations. The total
989 bit complexity of one iteration of Line 7 is thus in $\tilde{O}_B(d(\tau + \log M))$.

990 In Line 8, first notice that $\phi_\mu(H(t-ay, y)) = \phi_\mu(H(t-sy, y))|_{s=a}$. Similarly as above, the $O(d^3)$
991 coefficients of $H(t-sy, y)$ of bitsize $\tilde{O}(d+\tau)$ can be reduced modulo μ with $\tilde{O}_B(d^3(d+\tau+\log M))$
992 bit operations in total. The evaluation at $s = a$ in \mathbb{Z}_μ then amounts to evaluating $O(d^2)$ univariate
993 polynomials in s of degree $O(d)$. Similarly as above, this can be done with $O(d^3)$ arithmetic
994 operations \mathbb{Z}_μ and thus with $\tilde{O}_B(d^3 \log M)$ bit operations. Thus, $\phi_\mu(H(t-ay, y))$ and similarly
995 $\phi_\mu(\frac{\partial H}{\partial y}(t-ay, y))$ can be computed with $\tilde{O}_B(d^3(d+\tau+\log M))$ bit operations in the worst case. By
996 Lemma 4, their resultant $R_{\mu,a}(t)$ has degree $O(d^2)$, and it can be computed with $\tilde{O}(d^3)$ arithmetic
997 operations in \mathbb{Z}_μ and thus with $\tilde{O}_B(d^3 \log M)$ bit operations. The bit complexity of one iteration
998 of Line 8 is thus in $\tilde{O}_B(d^3(d+\tau+\log M))$ in the worst-case.

999 In Line 9, the squarefree part of $R_{\mu,a}(t)$, and thus its degree, can be computed with $\tilde{O}(d^2)$
1000 arithmetic operations in \mathbb{Z}_μ (by Lemma 4) and thus with $\tilde{O}_B(d^2 \log M)$ bit operations in the worst
1001 case.

1002 Hence, the expected bit complexity of one iteration of the loop is in $\tilde{O}_B(d^3(d+\tau+\log M) +$
1003 $\log^{8.5} M)$, which is also in $\tilde{O}_B(d^3(d+\tau+\log^9 M))$. More precisely, at the end of the j -th iteration
1004 of the loop, $M = 2^{j+1}d^4$, thus the expected bit complexity of the j -th iteration of the loop is
1005 in $\tilde{O}_B(d^4 + d^3\tau + d^3j^9)$. The expected bit complexity of the entire loop is thus $\tilde{O}_B(d^4 + d^3\tau)$, by
1006 Lemmas 31 and 32. Summing up with the complexity of Lines 1 and 2, we obtain that the expected
1007 bit complexity of the algorithm is in $\tilde{O}_B(d^5 + d^4\tau)$. \square

1008 6 RUR decomposition

1009 In this section, we consider that a separating form for the bivariate system $\{P, Q\}$ as been computed
1010 as shown in Section 5 and we focus on the computation of Rational Univariate Representations of
1011 the solutions. We present a new algorithm of worst-case bit complexity $\tilde{O}_B(d^6 + d^5\tau)$ for computing
1012 a RUR decomposition of $\{P, Q\}$, that is a sequence of RURs that encodes the solutions of $\{P, Q\}$ (see
1013 Definition 36 and Theorem 45). This algorithm is multi-modular and it relies on both the triangular
1014 decomposition and the luckiness certificate of Section 4. We also present a Las-Vegas version of
1015 this algorithm, of expected bit complexity $\tilde{O}_B(d^5 + d^4\tau)$ (Theorem 46), which only computes some
1016 coefficients of the above triangular decomposition and avoids computing the luckiness certificate.

1017 In Section 6.1, we first recall the definitions and main properties of RURs. We present our
1018 deterministic algorithm and its complexity analysis in Section 6.2 and its Las-Vegas version in

1020 **6.1 RUR definition and properties**

1021 **Definition 34** ([Rou99, Definition 3.3]). *Let $I \subset \mathbb{Q}[x, y]$ be a zero-dimensional ideal, $V(I) = \{\sigma \in$
1022 $\mathbb{C}^2, v(\sigma) = 0, \forall v \in I\}$ its associated variety, and let $(x, y) \mapsto x + ay$ be a linear form with a in \mathbb{Q} .
1023 The RUR-candidate of I associated to $x + ay$ (or simply, to a), denoted $\text{RUR}_{I,a}$, is the following
1024 set of four univariate polynomials in $\mathbb{C}[t]$*

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

1025 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
1026 $V(I)$, we say that the linear form $x + ay$ separates $V(I)$ (or is separating for I) and $\text{RUR}_{I,a}$ is
1027 called a RUR (the RUR of I associated to a).

1028 The following proposition states fundamental properties of RURs, which are all straightforward
1029 from the definition except for the fact that the RUR polynomials have rational coefficients [Rou99,
1030 Theorem 3.1].

Proposition 35 ([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 $\text{RUR}_{I,a}$ have rational coefficients. Furthermore, if $x + ay$
separates $V(I)$, the following mapping between $V(I)$ and $V(f_{I,a}) = \{\gamma \in \mathbb{C}, f_{I,a}(\gamma) = 0\}$*

$$\begin{aligned} V(I) &\rightarrow V(f_{I,a}) \\ (\alpha, \beta) &\mapsto \alpha + a\beta \\ \left(\frac{f_{I,a,x}}{f_{I,a,1}}(\gamma), \frac{f_{I,a,y}}{f_{I,a,1}}(\gamma) \right) &\leftarrow \gamma \end{aligned}$$

1031 is a bijection, which preserves the real roots and the multiplicities.

1032 Next, we define a RUR decomposition of an ideal.

1033 **Definition 36.** *Let $I \subset \mathbb{Q}[x, y]$ be a zero-dimensional ideal, $V(I) = \{\sigma \in \mathbb{C}^2, v(\sigma) = 0, \forall v \in I\}$
1034 its associated variety, and let $(x, y) \mapsto x + ay$ be a linear form with a in \mathbb{Q} . A RUR-candidate
1035 decomposition of I is a sequence of RUR-candidates, associated to $x + ay$, of ideals $I_i \supseteq I$, $i \in \mathcal{I}$
1036 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
1037 $i \in \mathcal{I}$, the RUR-candidate decomposition is a RUR decomposition of I .*

1038 The following proposition recalls an upper bound on the bitsize of a RUR of an ideal containing
1039 two coprime polynomials P and Q , that is a RUR parameterizing a subset of the solutions of
1040 the system $\{P, Q\}$. This bound applies to the RURs of our RUR decomposition and is used in
1041 Algorithm 6.

1042 **Proposition 37** ([BLPR15, Proposition 28]). *Let P and Q in $\mathbb{Z}[x, y]$ be two coprime polynomials
1043 of total degree at most d and maximum bitsize τ , let a be a rational of bitsize τ_a , and let J be any
1044 ideal of $\mathbb{Z}[x, y]$ containing P and Q . The polynomials of the RUR-candidate of J associated to a
1045 have degree at most d^2 and bitsize in $\tilde{O}(d^2\tau_a + d\tau)$.*

1046 Note that according to Theorem 28, a separating form $x + ay$ can be computed with an integer
1047 a of bitsize $O(\log d)$ and the bound in Proposition 37 becomes $\tilde{O}(d^2 + d\tau)$. In addition, even if
1048 Proposition 37 only states an asymptotic upper bound, an explicit upper bound $C(d^2 + d\tau) \log^k d\tau$
1049 with $C, k \in \mathbb{Z}$ can be obtained from straightforward, although unappealing, computations following
1050 the proof of that proposition. Indeed, this proof is based on Hadamard's inequality and Mignotte's
1051 lemma, which both state explicit bounds.

1052 Proposition 37 also yields the following bound on the total bisize of any RUR decomposition.

1053 **Corollary 38.** *Let P and Q in $\mathbb{Z}[x, y]$ be two coprime polynomials of total degree at most d and
1054 maximum bitsize τ , and let a be a rational of bitsize τ_a . The sum of the bitsizes of all coefficients
1055 of any RUR-candidate decomposition of $\langle P, Q \rangle$, associated to $x + ay$, is in $\tilde{O}(d^4\tau_a + d^3\tau)$.*

1056 *Proof.* By Definition 36, the ideals I_i defining a RUR-candidate decomposition of $\langle P, Q \rangle$ are such
1057 that (i) the solutions of I_i (counted with multiplicity) are included in those of $\langle P, Q \rangle$ (since $I_i \supseteq$
1058 $\langle P, Q \rangle$) and (ii) the sets $V(I_i)$ of (distinct) solutions of I_i are pairwise disjoint. Hence, the sum
1059 over all i of the number of solutions of I_i , counted with multiplicity, is at most d^2 , the Bézout
1060 bound of $\{P, Q\}$. By Definition 34, the sum over all i of the degrees of the first polynomial of the
1061 RUR-candidate of I_i is thus also at most d^2 . Moreover, still by Definition 34, the degree the first
1062 polynomial of a RUR-candidate bounds from above the degrees of the other polynomials of the
1063 RUR-candidate. Hence, the total number of coefficients of the RUR-candidate decomposition is
1064 $O(d^2)$. The result then follows from Proposition 37. \square

1065 6.2 Decomposition algorithm

1066 Algorithm 6 computes a RUR decomposition of a zero-dimensional system $\{P, Q\}$, by first
1067 computing a separating form $x + ay$ as shown in Section 5 (Line 1). We then use this separating
1068 form to shear the system in generic position (Line 2) and compute the radical of a triangular
1069 decomposition of this system (Line 3). Then, using a multimodular approach, we compute RURs
1070 of each of the resulting radical systems (Lines 4–10) and return these RURs after a shear back
1071 (Line 11).

1072 The section is organized as follows. We first prove some preliminary lemmas that are instrumen-
1073 tal for the proof of correctness of Algorithm 6. We show in Lemma 39 that the ideals we compute
1074 in Line 3 are the radicals of the ideals output by the triangular decomposition of Algorithm 1.
1075 We then determine in Lemma 40 formulas for the RURs of these radical ideals. These formulas
1076 are valid over the rationals but, for complexity issues, we use these formulas in a multimodular
1077 setting, in Lines 4 to 10. For this purpose, Lemma 41 states conditions on primes μ under which the
1078 reductions modulo μ of the RURs of these ideals are equal to the RURs of the reductions modulo
1079 μ of these ideals. We also show in Lemma 42 how to compute the image of the computed RURs
1080 through the reverse shearing of the one performed in Line 2. With these lemmas, we prove in
1081 Propositions 43 and 44 the correctness and complexity of Algorithm 6. Theorem 45 finally gathers
1082 these results.

1083 **Lemma 39.** *The radical \widehat{T}_i of ideal T_i is $\langle A_i, i \operatorname{sres}_i(\tilde{P}, \tilde{Q})y + \operatorname{sres}_{i-1}(\tilde{P}, \tilde{Q}) \rangle$ where A_i is squarefree
1084 and coprime with $\operatorname{sres}_i(\tilde{P}, \tilde{Q})$.*

1085 *Proof.* Since $x + ay$ separates the solutions of $\{P, Q\}$, the system $\{\tilde{P}, \tilde{Q}\}$ is in generic position in
1086 the sense that no two of its solutions are vertically aligned. The systems $T_i = \{A_i(t), B_i(t, y)\}$ of
1087 the triangular decomposition of $\{\tilde{P}, \tilde{Q}\}$ are thus also in generic position (since the set of solutions
1088 of $\{\tilde{P}, \tilde{Q}\}$ is the disjoint union of those of the T_i by Lemma 10). $B_i(t, y) = \operatorname{sres}_i(\tilde{P}, \tilde{Q})y^i +$
1089 $\operatorname{sres}_{i-1}(\tilde{P}, \tilde{Q})y^{i-1} + \dots$ is of degree i in y and its leading coefficient $\operatorname{sres}_i(\tilde{P}, \tilde{Q})$ is coprime with

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 $\tilde{O}(d^4 + d^3\tau)$

- 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 28)
 - 2: Compute $\tilde{P}(t, y) = P(t - ay, y)$ and $\tilde{Q}(t, y) = Q(t - ay, y)$, and let \tilde{d} and $\tilde{\tau}$ be their maximum degree and bitsize
 - 3: Compute $\{T_i\}_{i \in \mathcal{I}} = \text{Algorithm 1}(\tilde{P}, \tilde{Q})$
 Recall that $T_i = \{A_i(t), B_i(t, y)\}$ with $B_i(t, y) = \text{sres}_i(\tilde{P}, \tilde{Q})(t) y^i + \text{sres}_{i-1}(\tilde{P}, \tilde{Q})(t) y^{i-1} + \dots$
 Let $\hat{T}_i = \langle A_i, i \text{sres}_i(\tilde{P}, \tilde{Q}) y + \text{sres}_{i-1}(\tilde{P}, \tilde{Q}) \rangle$ be the radical ideal of T_i (see Lemma 39)
 - 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 \hat{T}_i (see Proposition 37 and subsequent discussion) and let $\Pi = \text{Algorithm 2}(\tilde{P}, \tilde{Q})$. 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 $\phi_{\mu}(\hat{T}_i)$ by reducing modulo μ the polynomials A_i , $\text{sres}_i(\tilde{P}, \tilde{Q})$ and $\text{sres}_{i-1}(\tilde{P}, \tilde{Q})$
 - 8: Compute RUR_{μ}^{μ} the RUR in \mathbb{Z}_{μ} of $\phi_{\mu}(\hat{T}_i)$ associated to the separating form $(t, y) \mapsto t$ (see Lemma 40)
 - 9: Lift $\{\text{RUR}_{\mu}^{\mu}\}_{\mu \in \mathcal{L}}$ to $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ in $\mathbb{Z}_{\Pi_{\mathcal{L}}}$ using the Chinese Remainder Algorithm
 - 10: Compute $\text{RUR}_i^{\mathbb{Q}}$, the RUR in \mathbb{Q} of \hat{T}_i associated to the separating form $(t, y) \mapsto t$, with a rational reconstruction from $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ (see the proof of Proposition 43)
 - 11: **return** the image of $\text{RUR}_i^{\mathbb{Q}}$, $i \in \mathcal{I}$, through the reverse shearing from (t, y) to (x, y) (see Lemma 42)
-

1090 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
 1091 multiplicity i . This multiple root is thus also root of the $(i - 1)$ -th derivative of $B_i(\alpha, y)$, which
 1092 is $i! \text{sres}_i(\tilde{P}, \tilde{Q})(\alpha) y + (i - 1)! \text{sres}_{i-1}(\tilde{P}, \tilde{Q})(\alpha)$. Hence, the distinct solutions of T_i are exactly
 1093 those of \hat{T}_i . Finally, \hat{T}_i is radical because $A_i(t)$ is univariate and squarefree (by Lemma 10) and
 1094 $i \text{sres}_i(\tilde{P}, \tilde{Q}) y + \text{sres}_{i-1}(\tilde{P}, \tilde{Q})$ has degree one in the other variable, y . \square

1095 The next lemma states formulas for the RURs of the radical ideals \hat{T}_i . We state this lemma in
 1096 general form because we also use it for computing the RURs of $\phi_{\mu}(\hat{T}_i)$.

1097 **Lemma 40.** *Let \mathbb{F} be a field, A, B_0, B_1 be three polynomials in $\mathbb{F}[t]$ and let $I = \langle A(t), B_1(t) y + B_0(t) \rangle$
 1098 be an ideal such that A is squarefree and coprime with B_1 . The linear form $(t, y) \mapsto t$ is separating
 1099 for that ideal and its associated RUR is given by¹⁰*

$$f_I = \frac{A}{\text{LC}(A)} \quad f_{I,1} = f_I' \quad f_{I,t} = t f_{I,1} \text{ rem } A \quad f_{I,y} = -B_0 U f_{I,1} \text{ rem } A$$

1100 where $U \in \mathbb{F}[t]$ is the inverse of B_1 modulo A , defined by Bézout's identity $U B_1 + V A = 1$ and
 1101 where $f \text{ rem } g$ denotes the remainder of the Euclidean division of f by g .

1102 *Proof.* By Definition 34, the first polynomial of the RUR of I associated to the form $(t, y) \mapsto t$
 1103 is the unique monic polynomial that encodes the t -coordinates of the solutions of I , counted with

¹⁰We omit in the subscript of the polynomials of the RUR the reference to the parameter, 0, of the separating form $(t, y) \mapsto t + 0y$.

1104 multiplicity in I . Since A is squarefree and coprime with B_1 , the solutions of I have multiplicity one
 1105 and their t -coordinates are exactly the roots of A . Hence, since A is squarefree, the first polynomial
 1106 of the RUR is $f_I = \frac{A}{\text{Lc}(A)}$. It also follows from the definition of the RUR that if f_I is squarefree
 1107 then $f_{I,1} = f_I'$.

1108 By Proposition 35, $\frac{f_{I,t}}{f_{I,1}}(\alpha) = \alpha$ for any root α of $f_I = \frac{A}{\text{Lc}(A)}$, since the separating form is
 1109 $(t, y) \mapsto t$. Hence, $f_{I,t}(t) = t f_{I,1}(t) \pmod{A}$. It follows that $f_{I,t} = t f_{I,1} \pmod{A}$ since $f_{I,t}$ has the
 1110 degree of A minus 1 by Definition 34.

1111 We also have by Proposition 35 that $f_{I,1}y - f_{I,y}$ is in I . Multiplying it by B_1 and subtracting
 1112 $f_{I,1}(B_1y + B_0)$, which is also in I , we obtain that $B_1f_{I,y} + f_{I,1}B_0$ is in I . This polynomial is
 1113 univariate in t , hence it is equal to zero modulo A . On the other hand, since A and B_1 are coprime,
 1114 by Bézout's identity, there exists a pair (U, V) of polynomials in $\mathbb{F}[t]$ such that $UB_1 + VA = 1$,
 1115 and we have that $UB_1 = 1 \pmod{A}$. It follows that $f_{I,y} + Uf_{I,1}B_0 = 0 \pmod{A}$ and thus that
 1116 $f_{I,y} = -Uf_{I,1}B_0 \pmod{A}$ since $f_{I,y}$ has the degree of A minus 1 by Definition 34. \square

1117 Even if the bitsize of the RUR of \widehat{T}_i is known to be in $\widetilde{O}(d^2 + d\tilde{\tau}) = \widetilde{O}(d^2 + d\tau)$ (Proposition 37
 1118 and [BLPR15, Lemma 7]), the naive computation of these RURs using the above formulas over
 1119 the rationals would suffer from large intermediate bitsizes.¹¹ To overcome this difficulty, we use in
 1120 Algorithm 6 a classical multimodular technique, which consists in first computing the polynomials
 1121 modulo a set of primes whose product is larger than the bitsize of the output coefficients, then lifting
 1122 the result using the Chinese Remainder Algorithm and finally performing a rational reconstruction.
 1123 However, to output a correct result, this technique requires that, for any selected prime μ , the
 1124 formulas of Lemma 40 commute with the reduction modulo μ . We show in Lemma 41 how to
 1125 satisfy this requirement using the luckiness certificate output by Algorithm 2. This lemma is
 1126 instrumental for the proof of correctness of Algorithm 6.

1127 **Lemma 41.** *Let $\mu > i$ be a prime that does not divide Π . The ideals \widehat{T}_i and $\phi_\mu(\widehat{T}_i)$ satisfy the
 1128 hypotheses of Lemma 40. In particular, the linear form $(t, y) \mapsto t$ is separating for both ideals. For
 1129 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 .*

1130 *Proof.* By Lemma 39, the ideal $\widehat{T}_i = \langle A_i, i \text{sres}_i(\widetilde{P}, \widetilde{Q})y + \text{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q}) \rangle$ is such that A_i is squarefree
 1131 and coprime with $\text{sres}_i(\widetilde{P}, \widetilde{Q})$. Lemma 40 thus applies and yields that the linear form $(t, y) \mapsto t$ is
 1132 separating for ideal \widehat{T}_i and that the associated RUR can be computed with the given formulas.

1133 In the following, we assume that $\mu > i$ is a prime that does not divide Π . We first show that
 1134 the ideal $\phi_\mu(\widehat{T}_i) = \langle \phi_\mu(A_i), i \phi_\mu(\text{sres}_i(\widetilde{P}, \widetilde{Q}))y + \phi_\mu(\text{sres}_{i,i-1}(\widetilde{P}, \widetilde{Q})) \rangle$ also satisfies the hypotheses
 1135 of Lemma 40. Recall the notation used in the proof of Lemma 14: let $(A_i^\mu(t), B_i^\mu(t, y))$ be the
 1136 triangular systems computed by Algorithm 1 applied to $\phi_\mu(\widetilde{P}(t, y))$ and $\phi_\mu(\widetilde{Q}(t, y))$. Lemma 14
 1137 implies that μ is lucky for the triangular decomposition of $\{\widetilde{P}, \widetilde{Q}\}$, hence $\phi_\mu(A_i) = A_i^\mu$ and $\phi_\mu(B_i) =$
 1138 B_i^μ , the latter being equivalent to $\phi_\mu(\text{Sres}_i(\widetilde{P}, \widetilde{Q})) = \text{Sres}_i(\phi_\mu(\widetilde{P}), \phi_\mu(\widetilde{Q}))$. We thus have that
 1139 $\text{gcd}(\phi_\mu(A_i), i \phi_\mu(\text{sres}_i(\widetilde{P}, \widetilde{Q}))) = \text{gcd}(A_i^\mu, i \text{sres}_i(\phi_\mu(\widetilde{P}), \phi_\mu(\widetilde{Q})))$, which is a non-zero constant in \mathbb{Z}_μ by
 1140 Lemma 10. In addition, Lemma 10 implies that $\phi_\mu(A_i) = A_i^\mu$ is squarefree. Lemma 40 thus applies
 1141 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
 1142 RUR can be computed with the formulas of Lemma 40.

1143 Second, we prove that μ does not divide any denominator of the rational coefficients of the
 1144 polynomials of the RUR of \widehat{T}_i and thus that the images of these polynomials by ϕ_μ are well defined.

¹¹More precisely, the computation of the RURs using the formulas of Lemma 40 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 $\text{sres}_i(\widetilde{P}, \widetilde{Q}) \pmod{A_i}$, which is a polynomial of degree $O(d^2)$ and bitsize in $\widetilde{O}(d^4 + d^3\tau)$.

1145 By definition, A_i divides the resultant of \tilde{P} and \tilde{Q} , which is equal to $\text{sres}_0(\tilde{P}, \tilde{Q})$. It follows that
1146 μ does not divide $\text{Lc}(A_i)$ because μ does not divide $\text{Lc}(\text{sres}_0(\tilde{P}, \tilde{Q}))$ by definition of Π . Thus μ
1147 does not divide any denominator of the coefficients of the RUR polynomials $f_{\hat{T}_i} = \frac{A_i}{\text{Lc}(A_i)}$ and
1148 $f_{\hat{T}_i,1} = f'_{\hat{T}_i}$. On the other hand, if F is a polynomial in $\mathbb{Q}[t]$ such that μ does not divide the
1149 denominators of its coefficients, then μ does not divide the denominators of the coefficients of
1150 $F \text{ rem } A_i$ (the denominator of a coefficient of the remainder is the product of $\text{Lc}(A_i)$ and some
1151 denominators of coefficients of F). It follows that μ does not divide the denominators of the
1152 coefficients of $f_{\hat{T}_i,t} = t f_{\hat{T}_i,1} \text{ rem } A_i$. Similarly, to prove that μ does not divide the denominators
1153 of the coefficients of $f_{\hat{T}_i,y} = -\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q}) U f_{\hat{T}_i,1} \text{ rem } A_i$, it is sufficient to prove the μ does
1154 not divide the denominators of the coefficients of U , the inverse of $i \text{sres}_i(\tilde{P}, \tilde{Q})$ modulo A_i (since
1155 $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})$ has integer coefficients). By definition of Π , μ does not divide $\text{Lc}(\text{sres}_i(\tilde{P}, \tilde{Q}))$.
1156 In addition, we have shown that μ does not divide $\text{Lc}(A_i)$, A_i and $i \text{sres}_i(\tilde{P}, \tilde{Q})$ are coprime by
1157 Lemma 39 and we have shown above that $\phi_\mu(A_i)$ and $\phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q}))$ are also coprime. It follows
1158 that μ is lucky for $\text{gcd}(A_i, i \text{sres}_i(\tilde{P}, \tilde{Q}))$ (Definition 6). Thus, by Lemma 7, μ does not divide
1159 $\text{res} = \text{Res}_t(A_i, i \text{sres}_i(\tilde{P}, \tilde{Q}))$. By [BPR06, Prop. 8.38.a] and since A_i and $i \text{sres}_i(\tilde{P}, \tilde{Q})$ are coprime,
1160 there exist u, v in $\mathbb{Z}[t]$ such that, $d_t(u) < d_t(A)$ and $u i \text{sres}_i(\tilde{P}, \tilde{Q}) + v A_i = \text{res}$, which is equivalent to
1161 $\frac{u}{\text{res}} i \text{sres}_i(\tilde{P}, \tilde{Q}) + \frac{v}{\text{res}} A_i = 1$. By unicity of Bézout's coefficients in $\mathbb{Q}[t]$, U the inverse of $i \text{sres}_i(\tilde{P}, \tilde{Q})$
1162 modulo A_i is equal to $\frac{u}{\text{res}}$ and μ does not divide any denominator of its coefficients.

1163 It is now clear that the image by ϕ_μ of the RUR polynomials $f_{\hat{T}_i}, f_{\hat{T}_i,1}, f_{\hat{T}_i,t}$ are those
1164 of the RUR of $\phi_\mu(\hat{T}_i)$. For $f_{\hat{T}_i,y} = -\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q}) U f_{\hat{T}_i,1} \text{ rem } A_i$, since we have shown
1165 that $\phi_\mu(\text{Sres}_i(\tilde{P}, \tilde{Q})) = \text{Sres}_i(\phi_\mu(\tilde{P}), \phi_\mu(\tilde{Q}))$, it is sufficient to show that $\phi_\mu(U)$ is the inverse
1166 of $\phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q}))$ modulo $\phi_\mu(A_i)$. As shown above, $\phi_\mu(U)$ is well defined and the relation
1167 $\phi_\mu(\frac{u}{\text{res}}) \phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q})) + \phi_\mu(\frac{v}{\text{res}}) \phi_\mu(A_i) = 1$ implies that it is the inverse of $\phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q}))$ mod-
1168 ulo $\phi_\mu(A_i)$. \square

1169 **Lemma 42.** Let $\{f_I, f_{I,1}, f_{I,t}, f_{I,y}\}$ be the RUR¹⁰ of an ideal I in $\mathbb{Q}[t, y]$ associated to the separating
1170 linear form $(t, y) \mapsto t$. Let J in $\mathbb{Q}[x, y]$ be the image of I through the mapping $(t, y) \mapsto (x = t - ay, y)$.
1171 The linear form $(x, y) \mapsto x + ay$ is separating for J and its associated RUR is given by

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

1172 *Proof.* By Definition 34, the RURs of I and J are defined by

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

1173 The change of coordinates $(t, y) \mapsto (x = t - ay, y)$ induces an affine transformation of the
1174 solutions that preserves their multiplicities, such that, for every solution σ of I , there exists a unique
1175 solution σ' of J with the same multiplicity and satisfying $x(\sigma') = t(\sigma) - ay(\sigma)$ and $y(\sigma') = y(\sigma)$.
1176 This directly implies all four equalities of the lemma. \square

1177 We finally prove the correctness and analyze the complexity of Algorithm 6 in the following two
1178 propositions.

1179 **Proposition 43** (Correctness of Algorithm 6). *Algorithm 6 computes a RUR decomposition of*
 1180 *$\{P, Q\}$ of total bitsize $\tilde{O}(d^4 + d^3\tau)$.*

1181 *Proof.* The correctness of Line 1 follows directly from Theorem 28. In particular, the sheared
 1182 polynomials \tilde{P} and \tilde{Q} are coprime and their leading coefficients with respect to y are also coprime.
 1183 In Line 3, Algorithm 1 can thus be applied to compute the ideals T_i . By Lemma 39, \hat{T}_i is the
 1184 radical ideal of T_i and, since both ideals have the same (distinct) solutions, the set of solutions of
 1185 $\{\tilde{P}, \tilde{Q}\}$ is the disjoint union of the sets of solutions of all \hat{T}_i , by Lemma 10.

1186 In Line 4, the upper bound K can be computed according Proposition 37 and the subsequent
 1187 discussion.

1188 In Line 8, the RUR of $\phi_\mu(\hat{T}_i)$ can be computed using the formulas of Lemma 40 (which applies
 1189 by Lemma 41). Moreover, by Lemma 41, the RUR of $\phi_\mu(\hat{T}_i)$ is the reduction modulo μ of the RUR
 1190 of \hat{T}_i . Thus, in Line 9, by the Chinese Remainder Theorem, $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ is the reduction modulo $\Pi_{\mathcal{L}}$
 1191 of the RUR of \hat{T}_i .

1192 Finally, in Line 10, we use a rational number reconstruction [vzGG13, Section 5.10] with pa-
 1193 rameter $2M$ with $M = 2^K$: for any coefficient c of $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ in $\mathbb{Z}_{\Pi_{\mathcal{L}}}$ a rational number $\frac{r}{t}$ with r, t in
 1194 \mathbb{Z} is computed such that $\gcd(r, t) = 1$, $\gcd(t, \Pi_{\mathcal{L}}) = 1$, $rt^{-1} = c \pmod{\Pi_{\mathcal{L}}}$, $|r| < 2M$, $0 < t \leq \frac{\Pi_{\mathcal{L}}}{2M}$.
 1195 According to [vzGG13, Theorem 5.26 (iv)], there exists at most one solution such that $|r| < M$. On
 1196 the other hand, $\text{RUR}_i^{\mathbb{Q}}$, the RUR of \hat{T}_i computed in \mathbb{Q} defines such a solution for each coefficient.
 1197 Indeed, let \tilde{r}/\tilde{t} be the coefficient in $\text{RUR}_i^{\mathbb{Q}}$ corresponding to c , with $\gcd(\tilde{r}, \tilde{t}) = 1$ and $\tilde{t} > 0$. By def-
 1198 inition, 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
 1199 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
 1200 $0 < \tilde{t} < M < \frac{\Pi_{\mathcal{L}}}{2M}$. On the other hand, we prove in Lemma 41 that, modulo a prime $\mu > i$ that does
 1201 not divide Π , the reduction of $\text{RUR}_i^{\mathbb{Q}}$ is well defined, thus $\gcd(\tilde{t}, \Pi_{\mathcal{L}}) = 1$. Finally, since, as shown
 1202 above, $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ is the reduction modulo $\Pi_{\mathcal{L}}$ of $\text{RUR}_i^{\mathbb{Q}}$, the RUR of \hat{T}_i , we have that $c = \phi_{\Pi_{\mathcal{L}}}(\tilde{r}/\tilde{t})$,
 1203 that is $\tilde{r}\tilde{t}^{-1} = c \pmod{\Pi_{\mathcal{L}}}$. The unique solution of the rational reconstruction of $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ is thus
 1204 well defined and equal to the RUR of \hat{T}_i in \mathbb{Q} .

1205 At the end of Line 10, we have thus computed the sequence of RURs of \hat{T}_i associated to the
 1206 separating form $(t, y) \mapsto t$, for all $i \in \mathcal{I}$. This is a RUR decomposition of $\langle \tilde{P}, \tilde{Q} \rangle$ since as shown
 1207 above, the set of solutions of $\{\tilde{P}, \tilde{Q}\}$ is the disjoint union of the sets of solutions of all \hat{T}_i and since
 1208 $\langle \tilde{P}, \tilde{Q} \rangle \subseteq T_i \subseteq \hat{T}_i$ by Lemma 10.

1209 By definition of \tilde{P} and \tilde{Q} , the images of these RURs through the mapping $(t, y) \mapsto (x =$
 1210 $t - ay, y)$ yield a RUR decomposition of $\langle P, Q \rangle$ associated to the form $(x, y) \mapsto x + ay$. This RUR
 1211 decomposition is computed in Line 11 using the formulas of Lemma 42.

1212 Finally, the total bitsize of $\tilde{O}(d^4 + d^3\tau)$ of all the coefficients of this RUR decomposition follows
 1213 from Corollary 38 since the bitsize of a is in $O(\log d)$ by Theorem 28. \square

1214 **Proposition 44.** *Algorithm 6 computes a RUR decomposition of $\{P, Q\}$ with $\tilde{O}_B(d^6 + d^5\tau)$ bit*
 1215 *operations in the worst case.*

1216 *Proof.* The bit complexity of Line 1 is $\tilde{O}_B(d^6 + d^5\tau)$ by Theorem 28. In Line 2, since a has bitsize
 1217 in $O(\log d)$, the sheared polynomials \tilde{P} and \tilde{Q} can be computed in bit complexity $\tilde{O}_B(d^4 + d^3\tau)$ and
 1218 their maximum degrees \tilde{d} and bitsizes $\tilde{\tau}$ are in $O(d)$ and $\tilde{O}(d + \tau)$, respectively (see e.g. [BLPR15,
 1219 Lemma 7]).

1220 In Lines 3 and 4, the bit complexities of Algorithms 1 and 2 applied on $\{\tilde{P}, \tilde{Q}\}$ are in $\tilde{O}_B(\tilde{d}^6 + \tilde{d}^5\tilde{\tau})$
 1221 by Proposition 16. In Line 4, computing K has bit complexity $\tilde{O}_B(\log \tilde{d}\tilde{\tau})$ (since the constants C
 1222 and k are known according to the discussion following Proposition 37). Still in Line 4, computing
 1223 \mathcal{L} can be done by (i) computing the first $2K + \lceil \log \Pi \rceil$ primes larger than \tilde{d} , then (ii) reducing Π

1224 modulo these primes using a remainder tree [MB74] and (iii) keeping the first $2K$ primes that do
 1225 not dividing Π (there exists at least $2K$ primes that do not dividing Π since the number of primes
 1226 that divide Π is smaller than $\lceil \log \Pi \rceil$). The bit complexity of computing the r first prime numbers
 1227 is in $\tilde{O}_B(r)$ and their maximum is in $\tilde{O}(r)$ [vzGG13, Theorem 18.10]. Hence, since Π has bitsize
 1228 $\tilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$ by Proposition 16, phase (i) has bit complexity $\tilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$, every prime has bitsize
 1229 $O(\log \tilde{d}\tilde{\tau})$ and their product has bitsize $\tilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$. Phase (ii) can be computed in a bit complexity
 1230 that is soft linear in the total bitsize of the input [MB74, Theorem 1], hence in $\tilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$ bit
 1231 operations. Therefore, the bit complexity of Lines 3 and 4 is $\tilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$.

1232 In Lines 5 and 6, the cardinality of \mathcal{I} is $O(\tilde{d})$ (see Algorithm 1) and the cardinality of \mathcal{L} is
 1233 $2K = \tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$.

1234 In Line 7, every subresultant of \tilde{P} and \tilde{Q} (including the resultant) has degree $O(\tilde{d}^2)$ in t and its
 1235 coefficients have bisize $\tilde{O}(\tilde{d}\tilde{\tau})$ by Lemma 3. Furthermore, A_i is factor of $\text{Res}(\tilde{P}, \tilde{Q})$ by construction,
 1236 hence the bitsize of its coefficients is in $\tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ by Mignotte's lemma (see e.g. [BPR06, Corollary
 1237 10.12]). Hence, in Line 7, A_i , $\text{sres}_i(\tilde{P}, \tilde{Q})$ and $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})$ have degree $O(\tilde{d}^2)$ and coefficients of
 1238 bisize $\tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$. For every i , the reductions of each of these coefficients modulo the $\tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$
 1239 primes μ (of bitsize $O(\log d\tau)$) in \mathcal{L} can be done, using again a remainder tree, in bit complexity
 1240 $\tilde{O}_B(\tilde{d}^2 + \tilde{d}\tilde{\tau})$. The reductions of all $O(\tilde{d}^2)$ coefficients for all $O(\tilde{d})$ $i \in \mathcal{I}$ and all $\mu \in \mathcal{L}$ can thus be
 1241 done in bit complexity $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

1242 In Line 8, for every i and μ , we compute RUR_i^μ using the formulas of Lemma 40 where the input
 1243 polynomials are $A = \phi_\mu(A_i)$, $B_1 = \phi_\mu(\text{sres}_i(\tilde{P}, \tilde{Q}))$ and $B_0 = \phi_\mu(\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q}))$ in $\mathbb{Z}_\mu[t]$. Following
 1244 these formulas, computing RUR_i^μ can be done with $O(1)$ additions, multiplication and inverse
 1245 computations in $\mathbb{Z}_\mu[t]/\langle A \rangle$ once B_0 and B_1 are reduced in $\mathbb{Z}_\mu[t]/\langle A \rangle$. These reductions amount
 1246 to computing the remainders of the divisions of B_0 and B_1 by A , whose arithmetic complexity
 1247 in \mathbb{Z}_μ is softly linear in their degrees $O(\tilde{d}^2)$ [vzGG13, Theorem 9.6]. Furthermore, the arithmetic
 1248 complexity in \mathbb{Z}_μ of every operation in $\mathbb{Z}_\mu[t]/\langle A \rangle$ is softly linear in the degree $O(\tilde{d}^2)$ of A [vzGG13,
 1249 Corollary 11.11]. Summing over all $i \in \mathcal{I}$ and all $\mu \in \mathcal{L}$, the $\tilde{O}(\tilde{d}^3 + \tilde{d}^2\tilde{\tau})$ RUR_i^μ can be computed
 1250 with $\tilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ arithmetic operations in $\mathbb{Z}_{\mu \in \mathcal{L}}$. Finally, since every $\mu \in \mathcal{L}$ has bitsize $O(\log \tilde{d}\tilde{\tau})$,
 1251 the total bit complexity of Line 8 is $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

1252 In Line 9, for any given i , the complexity of lifting $\{\text{RUR}_i^\mu\}_{\mu \in \mathcal{L}}$ to $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ in $\mathbb{Z}_{\Pi_{\mathcal{L}}}$ is the
 1253 complexity of lifting its $O(\tilde{d}^2)$ coefficients. Every coefficient reconstruction in $\mathbb{Z}_{\Pi_{\mathcal{L}}}$ can be done
 1254 using the Chinese Remainder Algorithm with $\tilde{O}_B(\log \Pi_{\mathcal{L}}) = \tilde{O}_B(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ bit operations [vzGG13,
 1255 Theorem 10.25]. Summing over all coefficients and all i , the total bit complexity of Line 9 is thus
 1256 in $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

1257 In Line 10, for any given i , the complexity of the rational reconstruction of $\text{RUR}_i^{\mathbb{Q}}$ from $\text{RUR}_i^{\Pi_{\mathcal{L}}}$
 1258 is the complexity of the rational reconstructions with parameter $2M = 2^{K+1}$ of the $O(\tilde{d}^2)$ rationals
 1259 coefficients of $\text{RUR}_i^{\mathbb{Q}}$ from those of $\text{RUR}_i^{\Pi_{\mathcal{L}}}$ (see the proof of Proposition 43 for details). The rational
 1260 reconstruction $r/t \in \mathbb{Q}$ of $c \in \mathbb{Z}_{\Pi_{\mathcal{L}}}$ with parameter $2M$ is the cost of computing the first line of the
 1261 Extended Euclidean Algorithm (EEA) for $\Pi_{\mathcal{L}}$ and c such that the remainder is smaller than $2M$
 1262 [vzGG13, Theorem 5.26]. Using binary search, we can compute at most a logarithmic number of
 1263 lines of the EEA. Since the total number of lines of the EEA and the bit complexity of computing
 1264 one line of the EEA are (at most) softly linear in the bitsize of the input [vzGG13, Corollary 11.9],
 1265 the rational reconstruction of one rational has bit complexity $\tilde{O}_B(\tilde{d}^2 + \tilde{d}\tilde{\tau})$. Summing over all
 1266 coefficients and all i , the total bit complexity of Line 10 is thus in $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

1267 In Line 11, for every i , the image of $\text{RUR}_i^{\mathbb{Q}}$ through the reverse shearing can be computed
 1268 with $O(\tilde{d}^2)$ arithmetic operations on integers of bitsize $\tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ by Lemma 42. Hence, the bit
 1269 complexity of Line 11 is trivially $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$. (Note that in Lines 9, 10 and 11 an amortized
 1270 analysis yields a complexity of $\tilde{O}_B(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$ by observing that the degrees $\tilde{d}_i \in O(\tilde{d}^2)$ of the first

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 $\tilde{O}(d^4 + d^3\tau)$

- 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 29)
 - 2: Compute $\tilde{P}(t, y) = P(t - ay, y)$ and $\tilde{Q}(t, y) = Q(t - ay, y)$, and let \tilde{d} and $\tilde{\tau}$ be their maximum degree and bitsize
 - 3: Compute the coefficients $\text{sres}_i(\tilde{P}, \tilde{Q})(t)$ of subresultant sequence of \tilde{P} and \tilde{Q} with respect to y and, for i such that $\text{sres}_i(\tilde{P}, \tilde{Q}) \neq 0$, compute $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})(t)$ (see Corollary 52)
 Compute the polynomials $A_i(t)$, $i \in \mathcal{I}$, of the triangular decomposition of \tilde{P} and \tilde{Q} following Algorithm 1.
 Let $\hat{T}_i = \langle A_i, i \text{ sres}_i(\tilde{P}, \tilde{Q}) y + \text{sres}_{i,i-1}(\tilde{P}, \tilde{Q}) \rangle$, $i \in \mathcal{I}$, be the radicals of the ideals output by Algorithm 1(\tilde{P}, \tilde{Q}) (see Lemma 39 and note that, in Algorithm 1, $\text{sres}_i(\tilde{P}, \tilde{Q}) \neq 0$ 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 \hat{T}_i (see Proposition 37 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 $i \text{ sres}_i(\tilde{P}, \tilde{Q})$ 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 $\phi_\mu(i \text{ sres}_i(\tilde{P}, \tilde{Q}))$
 - 9: **until** \mathcal{L} contains at least $2K$ distinct primes
 - 10: **return** the image of $\text{RUR}_i^\mathbb{Q}$, the RUR of \hat{T}_i , $i \in \mathcal{I}$, through the reverse shearing from (t, y) to (x, y) , as in Algorithm 6, Lines 5-11
-

1271 polynomials of RUR_i^μ sum up, over all i , to at most \tilde{d}^2 .)

1272 Finally, since \tilde{d} and $\tilde{\tau}$ are in $O(d)$ and $\tilde{O}(d + \tau)$, the total bit complexity of Algorithm 6 is
 1273 in $\tilde{O}_B(\tilde{d}^6 + \tilde{d}^5\tilde{\tau})$. □

1274 Propositions 43 and 44 directly yield the following theorem.

1275 **Theorem 45.** *Let P, Q in $\mathbb{Z}[x, y]$ be of total degree at most d and maximum bitsize τ . Algorithm 6*
 1276 *computes, with $\tilde{O}_B(d^6 + d^5\tau)$ bit operations in the worst case, a RUR decomposition of $\{P, Q\}$ of*
 1277 *total bitsize $\tilde{O}(d^4 + d^3\tau)$.*

1278 6.3 Las-Vegas algorithm

1279 We show here that the algorithm presented above for computing a RUR decomposition can
 1280 easily be transformed into an efficient Las-Vegas algorithm. We prove here the following.

1281 **Theorem 46.** *Let P, Q in $\mathbb{Z}[x, y]$ be of total degree at most d and maximum bitsize τ . Algorithm 6'*
 1282 *computes, with $\tilde{O}_B(d^5 + d^4\tau)$ bit operations on average, a RUR decomposition of $\{P, Q\}$ of total*
 1283 *bitsize $\tilde{O}(d^4 + d^3\tau)$.*

1284 Algorithm 6', our Las-Vegas version of Algorithm 6, is obtained from the latter with only three
 1285 modifications. First, in Line 2, we use the Las-Vegas version of our algorithm for computing a
 1286 separating linear form for $\{P, Q\}$, described in Section 5.5.

1287 Second, in Line 3, we modify the way we compute the radicals \hat{T}_i of the ideals T_i output by
 1288 Algorithm 1(\tilde{P}, \tilde{Q}). We still use the formula $\hat{T}_i = \langle A_i, i \text{ sres}_i(\tilde{P}, \tilde{Q}) y + \text{sres}_{i,i-1}(\tilde{P}, \tilde{Q}) \rangle$ of Lemma 39

1289 for computing these radical ideals, but instead of computing the T_i with Algorithm 1, we show in
 1290 Section 6.3.1 that the subresultant coefficients $\text{sres}_i(\tilde{P}, \tilde{Q})$ and $\text{sres}_{i-1}(\tilde{P}, \tilde{Q})$ can be computed
 1291 more efficiently.

1292 Third, we modify the way we compute in Algorithm 6, Line 4, a set \mathcal{L} of $2K$ prime numbers
 1293 $\mu > \tilde{d}$ that do not divide $\Pi = \text{Algorithm } 2(\tilde{P}, \tilde{Q})$. Here, in Line 9, we weaken the constraints on
 1294 these primes and we avoid, in particular, computing Π .

1295 We prove Theorem 46 by first proving its correctness in Proposition 47 and then its complexity
 1296 in Proposition 50.

1297 **Proposition 47** (Correctness of Algorithm 6'). *Algorithm 6' computes a RUR decomposition of*
 1298 *$\{P, Q\}$ of total bitsize $\tilde{O}(d^4 + d^3\tau)$.*

1299 *Proof.* As described above, Algorithm 6' is obtained with only three modifications from Algorithm 6,
 1300 whose correctness is proved in Proposition 43. The first two modifications do not jeopardize the
 1301 correctness of Algorithm 6' since we compute the same objects as in Algorithm 6 (in particular,
 1302 we use the same formula for \hat{T}_i , $i \in \mathcal{I}$). However, in the third modification, we weaken the
 1303 constraints on the primes of \mathcal{L} . In the proof of correctness of Algorithm 6, the constraints on the
 1304 primes of \mathcal{L} (that $\mu > \tilde{d}$ does not divide Π) are only used in Lemma 41. Furthermore, in proof of
 1305 Lemma 41, these constraints are only used for proving that $\phi_\mu(\hat{T}_i) = \langle \phi_\mu(A_i), \phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q})) y +$
 1306 $\phi_\mu(\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})) \rangle$ satisfies the hypotheses of Lemma 40, that is that $\phi_\mu(A_i)$ is squarefree and
 1307 coprime with $\phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q}))$, which are the constraints on μ we impose in Line 8 of Algorithm 6'.
 1308 The correctness of Algorithm 6' thus follows from that of Algorithm 6. \square

1309 We now analyse the complexity of Algorithm 6'. A key step of this algorithm is the computation,
 1310 in Line 3, of $\text{sres}_i(\tilde{P}, \tilde{Q})$ and $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})$, which we postpone to Section 6.3.1. Before proving
 1311 Proposition 50, which states the complexity of Algorithm 6', we prove two lemmas. The first one
 1312 bounds the number of primes that are rejected in Line 8 and the second one will be instrumental
 1313 for bounding the probability that the loop ends in Line 9.

1314 **Lemma 48.** *There are $\tilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ primes that are unlucky for $\gcd(A_i, i \text{sres}_i(\tilde{P}, \tilde{Q}))$ or $\gcd(A_i, A'_i)$,*
 1315 *for some i . Furthermore, if prime μ is lucky for these two gcds, for some i , then $\phi_\mu(A_i)$ is squarefree*
 1316 *and coprime with $\phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q}))$.*

1317 *Proof.* By Lemma 7, the unlucky primes for the gcd of two polynomials A and B in $\mathbb{Z}[t]$ are
 1318 exactly the divisors of their leading coefficients and the divisors of $\text{sres}_d(A, B)$ where d is the degree
 1319 of $\gcd(A, B)$. In order to bound the number of unlucky primes, we bound the bitsizes of the relevant
 1320 coefficients.

1321 By Lemma 10, A_i divides the resultant $\text{Res}(\tilde{P}, \tilde{Q})$. Thus, A_i has degree $O(\tilde{d}^2)$ and coefficients
 1322 of bitsize $\tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$, as shown in the proof of Proposition 44. It follows that the same bounds also
 1323 apply to A'_i . On the other hand, $i \text{sres}_i(\tilde{P}, \tilde{Q})$ has degree $O(\tilde{d}^2)$ and coefficients of bitsize $\tilde{O}(\tilde{d}\tilde{\tau})$,
 1324 by Lemma 3. Still by Lemma 3, the coefficients of the subresultant polynomials of any two of these
 1325 polynomials have bitsize $\tilde{O}(\tilde{d}^2(\tilde{d}^2 + \tilde{d}\tilde{\tau}))$. The number of primes divisor of any such coefficient is
 1326 thus also in $\tilde{O}(\tilde{d}^4 + \tilde{d}^3\tilde{\tau})$. Since i varies from 1 to at most \tilde{d} (see Algorithm 1), the number of
 1327 unlucky primes is in $\tilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$.

1328 Finally, for any i , both $\gcd(A_i, i \text{sres}_i(\tilde{P}, \tilde{Q}))$ and $\gcd(A_i, A'_i)$ are equal to constants, by Lemma 10.
 1329 Furthermore, if μ is lucky for these gcds, these gcds commute with ϕ_μ , by Lemma 7. Hence, $\phi_\mu(A_i)$
 1330 is squarefree and coprime with $\phi_\mu(i \text{sres}_i(\tilde{P}, \tilde{Q}))$. \square

1331 **Lemma 49.** *Let n_{2p} be the random variable that represents the number of distinct elements obtained*
 1332 *by choosing uniformly at random $2p$ elements among n with replacement. If $n \geq 2p \geq 4$, then the*
 1333 *probability that $n_{2p} > p$ is larger than $\frac{1}{2}$.*

1334 *Proof.* Consider one of the $\binom{n}{d}$ sets of d distinct elements among n . Denote it by S_d , denote the
1335 set of p random elements by S and its cardinal by $|S|$. The probability that $S \subseteq S_d$, which is the
1336 probability that the p random elements in S are all in S_d is $\left(\frac{d}{n}\right)^p$. On the other hand, $\Pr(|S| \leq d)$
1337 is less than the sum of all $\Pr(S \subseteq S_d)$ for the $\binom{n}{d}$ choices of sets S_d . Hence, $\Pr(|S| \leq d) < \binom{n}{d} \left(\frac{d}{n}\right)^p$
1338 and $\Pr(|S| > d) \geq 1 - \binom{n}{d} \left(\frac{d}{n}\right)^p$.

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

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

$$\leq \frac{e n^{n+\frac{1}{2}} e^{-n}}{2\pi d^{d+\frac{1}{2}} e^{-d} (n-d)^{n-d+\frac{1}{2}} e^{-(n-d)}} \frac{d^{2d}}{n^{2d}} = \frac{e}{2\pi} \frac{n^{n+\frac{1}{2}-2d} d^{d-\frac{1}{2}}}{(n-d)^{n-d+\frac{1}{2}}}. \quad (4)$$

1339 Replacing n by kd with $k \geq 2$, we get

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

1340 and the derivative with respect to d of the right-hand side of the inequality is

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

1341 It is straightforward to prove that the function $k \mapsto \frac{k^{k-2}}{(k-1)^{k-1}}$ is decreasing for $k \geq 2$, hence
1342 $\ln \frac{k^{k-2}}{(k-1)^{k-1}}$ is negative for $k > 2$ and (6) is negative for $k \geq 2$. It follows that, for $d > 2$, the right-
1343 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
1344 for $k \geq 2$ and it is thus less than $\frac{e}{2\pi} \frac{\sqrt{2}}{\sqrt{2}} = \frac{e}{2\pi} < \frac{1}{2}$. Therefore, for $n \geq 2d$ and $d \geq 2$, $\binom{n}{d} \left(\frac{d}{n}\right)^{2d} < \frac{1}{2}$
1345 and thus $\Pr(|S| > d) > \frac{1}{2}$. \square

1346 **Proposition 50.** *Algorithm 6' computes a RUR decomposition of $\{P, Q\}$ with $\tilde{O}_B(d^5 + d^4\tau)$ bit*
1347 *operations on average.*

1348 *Proof.* The expected bit complexity of Line 1 is $\tilde{O}_B(d^5 + d^4\tau)$ by Theorem 29 and, as in Algorithm 6,
1349 the (worst-case) bit complexity of Line 2 is $\tilde{O}_B(d^4 + d^3\tau)$ and \tilde{P} and \tilde{Q} have maximum degree
1350 $\tilde{d} \in O(d)$ and maximum bitsize $\tilde{\tau} \in \tilde{O}(d + \tau)$ (see the proof of Proposition 44).

1351 In Line 3, the sequence of coefficients $\text{sres}_i(\tilde{P}, \tilde{Q})$ and, for those that do not identically vanish,
1352 the coefficients $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})$ can be computed in $\tilde{O}_B(\tilde{d}^4\tilde{\tau})$ bit operations by Corollary 52. Hence,
1353 the sequence of polynomials A_i can be computed in $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ bit operations by Remark 17.
1354 We thus get, in Line 3, the sequence of ideals \tilde{T}_i in $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ bit operations.

1355 In Line 4, the complexity of computing K and U is $\tilde{O}_B(\log \tilde{d}\tilde{\tau})$, as in Algorithm 6.

1356 In Line 6, we choose uniformly at random, one at a time, $8K$ primes in $[1, U]$. Some primes
1357 might be chosen more than once and thus the resulting set of primes, \mathcal{P} , may be of cardinality
1358 smaller than $8K$. The analysis is similar to the one in Proposition 33. A random integer in $[1, U]$
1359 can be computed in $O_B(\log U)$ bit operations. There are at least $\frac{U}{\ln U}$ primes in $[1, U]$ [vzGG13,

1360 Theorem 18.7]. The probability that a randomly chosen integer in $[1, U]$ is prime is thus at least $\frac{1}{\ln U}$
1361 and a prime is thus found after at most $\ln U$ trials on average. Testing whether an integer in $[1, U]$
1362 is prime can be done with a polynomial bit complexity in the bitsize of U , $\tilde{O}_B(\log^{7.5} U)$ [AKS04].
1363 The expected bit complexity of computing a prime in Line 7 is thus $\tilde{O}_B(\log^{8.5} U)$ and the expected
1364 bit complexity of computing $8K \in \tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$ primes in Line 7 is thus in $\tilde{O}_B((\tilde{d}^2 + \tilde{d}\tilde{\tau}) \log^{8.5} U)$.

1365 In Line 7, each of the $O(\tilde{d})$ polynomials A_i and $i \text{ sres}_i(\tilde{P}, \tilde{Q})$ have $O(\tilde{d}^2)$ coefficients of bitsize
1366 $\tilde{O}(\tilde{d}^2 + \tilde{d}\tilde{\tau})$, as shown in the proof of Lemma 48. Using remainder trees [MB74], the reductions of
1367 one coefficient modulo all the primes in \mathcal{L} can be done in a bit complexity that is softly linear in
1368 the maximum bitsize of the coefficient and the product of the primes, that is in $\tilde{O}_B((\tilde{d}^2 + \tilde{d}\tilde{\tau}) +$
1369 $(\tilde{d}^2 + \tilde{d}\tilde{\tau}) \log U)$. Hence, the bit complexity of Line 7 is $\tilde{O}_B((\tilde{d}^5 + \tilde{d}^4\tilde{\tau}) \log U)$.

1370 In Line 8, for any i , $\gcd(\phi_\mu(A_i), \phi_\mu(i \text{ sres}_i(\tilde{P}, \tilde{Q})))$ and $\gcd(\phi_\mu(A_i), \phi_\mu(A'_i))$ can be computed
1371 in $\tilde{O}_B(d^4 \log U)$ bit operations, by Lemma 4, since the polynomials have degree $O(\tilde{d}^2)$ and μ has
1372 bitsize $O(\log U)$. Hence, the bit complexity of Line 8 is $\tilde{O}_B(\tilde{d}^5 \log U)$. (Note that considering the
1373 degrees d_i of the A_i , which sum up to $O(d^2)$ yields a finer bound of $\tilde{O}_B(\tilde{d}^4 \log U)$.)

1374 We have shown that the expected bit complexity of one iteration of the loop in Lines 5 to 9 is in
1375 $\tilde{O}_B((\tilde{d}^5 + \tilde{d}^4\tilde{\tau}) \log^9 U)$. At the end of the j -th iteration of the loop, $U = 2^j \cdot 8K$, thus the expected
1376 bit complexity of the j -th iteration of the loop is in $\tilde{O}_B((\tilde{d}^5 + \tilde{d}^4\tilde{\tau})j^9)$.

1377 We now bound the total expected bit complexity of all the iterations of the loop in Lines 5 to 9.
1378 By Lemma 48, the primes that are rejected in Line 8 are unlucky for some $\gcd(A_i, i \text{ sres}_i(\tilde{P}, \tilde{Q}))$ or
1379 $\gcd(A_i, A'_i)$ 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
1380 C' and k' . We refer in the rest of the proof to *these* unlucky primes simply as unlucky primes.
1381 It follows that the probability that the loop ends in Line 9 is larger than the probability that \mathcal{P}
1382 contains at least $2K$ distinct lucky primes. Furthermore,

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

1383 As seen above, $[1, U]$ contains at least $\frac{U}{\ln U}$ primes. Thus, when $\frac{U}{\ln U} \geq 8K$, \mathcal{P} contains at least
1384 $4K$ distinct primes with probability at least $\frac{1}{2}$, by Lemma 49. On the other hand, the primes in
1385 \mathcal{P} are chosen uniformly at random among at least $\frac{U}{\ln U}$ primes, thus if $\frac{U}{\ln U} \geq 2\Gamma$, the primes in \mathcal{P}
1386 are lucky with probability at least $\frac{1}{2}$. Thus, if $\frac{U}{\ln U} \geq 2\Gamma$, given that \mathcal{P} contains at least $4K$ primes,
1387 the probability that \mathcal{P} contains at least $2K$ lucky primes is at least $\frac{1}{2}$. We thus have proved that,
1388 if $\frac{U}{\ln U} \geq \max(8K, 2\Gamma)$, the loop ends in Line 9 with probability at least $\frac{1}{4}$.

1389 There are $O(\log \tilde{d}\tilde{\tau})$ loop iterations that are performed while $\frac{U}{\ln U}$ is smaller than $\max(8K, 2\Gamma)$.
1390 Indeed, $\log U \in O(\log \tilde{d}\tilde{\tau})$ while $\frac{U}{\ln U} < \max(8K, 2\Gamma) \in \tilde{O}(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$ since $\sqrt{U} < \frac{U}{\ln U}$. The overall bit
1391 complexity of these iterations is thus in $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$. It follows that the expected bit complexity
1392 of the entire loop is in $\tilde{O}_B(\tilde{d}^5 + \tilde{d}^4\tilde{\tau})$, by Lemma 32.

1393 Summing up the complexities of all lines and since $\tilde{d} \in \tilde{O}(d)$ and $\tilde{\tau} \in \tilde{O}(d + \tau)$, we obtain that
1394 the expected bit complexity of the algorithm is $\tilde{O}_B(d^5 + d^4\tau)$. \square

1395 6.3.1 Computation of subresultant coefficients

1396 A key step of Algorithm 6' is the computation of the coefficients $\text{sres}_i(\tilde{P}, \tilde{Q})$ and the computation
1397 of $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})$ when $\text{sres}_i(\tilde{P}, \tilde{Q}) \neq 0$. We show that all these coefficients can be computed in
1398 $\tilde{O}_B(d^4\tau)$ bit complexity in Theorem 51 and Corollary 52. This result generalizes [vzGG13, Corollary
1399 11.18] to the case where one wants to compute the k terms of greater degrees in the sequence of
1400 remainders in the Euclidean algorithm.

1401 Given two polynomials $P, Q \in \mathbb{F}[y]$ such that $\deg(P) \geq \deg(Q)$, we denote by r_j and q_j the
 1402 polynomials appearing in the Euclidean algorithm such that $r_0 = P, r_1 = Q$ and $r_{i-1} = q_i r_i + r_{i+1}$.
 1403 For any polynomial $P \in \mathbb{F}[y]$ and any integer n , we denote by P_n the coefficient of its term of
 1404 degree $\deg(P) - n$, if any, and 0 otherwise. It follows that $r_{i|j}$ denotes the coefficient of the term
 1405 of r_i of degree $\deg(r_i) - j$.

1406 **Theorem 51.** *Let k be an integer and $P, Q \in \mathbb{F}[y]$ be two polynomials with $d = \deg(P) \geq \deg(Q)$.
 1407 We can compute, for all $0 \leq j \leq k$ and for all the remainders r_i appearing in the Euclidean
 1408 algorithm, the coefficients $r_{i|j}$ in $O(k^2 d + M(d) \log d)$ arithmetic operations, where $M(d)$ is the
 1409 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 di-
 rectly the coefficients $r_{0|0}$ and $r_{1|0}$, and from the formula

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

1410 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
 1411 divisions.

1412 Assume now that we have computed the coefficients $r_{i|j}$ for all i and $0 \leq j \leq k-1$. We show
 1413 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,

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

1414 which yields $r_{i|k}$ from the values of $r_{i-1|k}, r_{i|j}, r_{i+1|l}$, with $j, l \leq k-1$, in $2k+2$ arithmetic operations.
 1415 Thus, given the coefficients $r_{i|j}$ for all i and $0 \leq j \leq k-1$, we can compute the $r_{i|k}$, for all i , in
 1416 $O(kd)$ arithmetic operations, which trivially concludes the proof. \square

1417 We can now state the corollary that we use in the analysis of Algorithm 6'.

1418 **Corollary 52.** *Let $P, Q \in \mathbb{Z}[x, y]$ be of degree at most d with coefficients of bitsize at most τ . We
 1419 can compute in $\tilde{O}_B(d^4 \tau)$ bit operations in the worst case the sequence of all subresultant coefficients
 1420 $\text{sres}_i(P, Q)$ and, for i such that $\text{sres}_i(P, Q) \neq 0$, the coefficients $\text{sres}_{i,i-1}(P, Q)$.*

1421 *Proof.* We compute the subresultant coefficients using multimodular and interpolation techniques.
 1422 First, we select pairs (μ, k) with μ prime and k a value in \mathbb{Z}_μ satisfying the specialization property of
 1423 the subresultants. Second, we compute the subresultant coefficients $\text{sres}_i(P, Q)$ and $\text{sres}_{i,i-1}(P, Q)$
 1424 evaluated at $x = k$ in \mathbb{Z}_μ . Third, we interpolate the results in $\mathbb{Z}_\mu[x]$ and apply the Chinese remainder
 1425 algorithm to recover the final results in $\mathbb{Z}[x]$.

1426 To use the specialization property of subresultants, the leading coefficients of P and Q seen
 1427 as polynomials in y , $\text{Lc}_y(P)$ and $\text{Lc}_y(Q)$, must not vanish when evaluated at $x = k$ in \mathbb{Z}_μ . The
 1428 coefficients of P and Q being of bitsize at most τ , there are at most 2τ primes μ such that $\text{Lc}_y(P)$
 1429 or $\text{Lc}_y(Q)$ identically vanish modulo μ . When both do not identically vanish modulo μ , they are
 1430 polynomials of degree at most d , hence there are at most $2d$ values in \mathbb{Z}_μ at which one of them
 1431 vanishes. In \mathbb{Z}_μ , we will compute the subresultant coefficients via evaluation and interpolation.

1432 The number of evaluation values must be larger than the degrees of the subresultant coefficients
1433 $\text{sres}_i(P, Q)$ and $\text{sres}_{i,i-1}(P, Q)$, which are at most $2d^2$. It is sufficient to consider primes μ larger
1434 than $2d^2 + 2d$ because, then, there are at least $2d^2$ values in \mathbb{Z}_μ such that none of $\text{Lc}_y(P)$ and
1435 $\text{Lc}_y(Q)$ vanishes modulo μ . For lifting the subresultants using the Chinese remainder algorithm,
1436 the sum of the bitsizes of the primes must be larger than the bitsizes of the subresultants coefficients
1437 $\text{sres}_i(P, Q)$ and $\text{sres}_{i,i-1}(P, Q)$, which are at most $N = 2d(\tau + 2 \log d)$ [BPR06, Proposition 8.46].

1438 According to [vzGG13, Theorem 18.10], we can compute the M first primes $\mu_j \in \mathbb{Z}$ of bitsizes τ_j
1439 in $\tilde{O}_B(M)$ bit operations and their maximum bitsize is in $O(\log M)$. Among this set, the constraint
1440 for the specialization property of subresultants discards at most 2τ primes, and the constraint for the
1441 interpolation discards at most the first $2d^2 + 2d$ primes. Choosing $M = N + 2d^2 + 2d + 2\tau = O(d^2 + d\tau)$
1442 is thus sufficient to select a set of N primes satisfying these constraints. In addition, the sum of
1443 the bitsizes of these N primes is larger than N and in $O(N \log M) = \tilde{O}(d\tau)$.

1444 We now analyze the complexity of selecting N primes μ_j satisfying the above constraints and
1445 specializing P and Q at $2d^2$ values $x = k$ in $\mathbb{Z}_{\mu_j}[y]$. The reduction of one coefficient of P and Q
1446 modulo all the $N + 2\tau$ primes larger than $2d^2 + 2d$ can be computed via a remainder tree in a
1447 bit complexity that is soft linear in the total bitsize of the input [MB74, Theorem 1], which is in
1448 $\tilde{O}(d\tau)$. The reductions of all the $O(d^2)$ coefficients of P and Q can hence be done in $\tilde{O}_B(d^3\tau)$ bit
1449 operations. We select N primes μ_j such that $\text{Lc}_y(P)$ and $\text{Lc}_y(Q)$ do not identically vanish modulo
1450 μ_j . For a given prime μ_j , the evaluation of the reduction of $P(x, y)$ in $\mathbb{Z}_{\mu_j}[x, y]$ at $2d^2 + 2d$ values
1451 $x = k_\ell \in \mathbb{Z}_{\mu_j}$ involves $O(d^2)$ evaluations of $O(d)$ polynomials of degree $O(d)$ in $\mathbb{Z}_{\mu_j}[x]$. For a given
1452 prime μ_j , this can be done using multi-evaluation in $\tilde{O}(d^3)$ arithmetic operations in \mathbb{Z}_{μ_j} [vzGG13,
1453 Corollary 10.8] and thus with $\tilde{O}_B(d^3\tau_j) = \tilde{O}_B(d^3 \log M) = \tilde{O}_B(d^3 \log d\tau)$ bit operations. For all
1454 N primes, the total bit complexity of these evaluations is thus in $\tilde{O}_B(Nd^3 \log d\tau) = \tilde{O}_B(d^4\tau)$. For
1455 each prime μ_j , we select $2d^2$ values k_ℓ , among the $2d^2 + 2d$ values considered in \mathbb{Z}_{μ_j} , at which
1456 neither $\text{Lc}_y(P)$ nor $\text{Lc}_y(Q)$ vanishes when evaluated at $x = k_\ell$ in \mathbb{Z}_{μ_j} .

1457 In this paragraph, all polynomials are considered evaluated at $x = k$ and in $\mathbb{Z}_{\mu_j}[y]$ and, to
1458 clarify the presentation, any polynomial \tilde{K} refers to $K(k, y) \bmod \mu_j$. Then computing, for all
1459 i , $\text{sres}_i(\tilde{P}, \tilde{Q})$ can be done in a total of $\tilde{O}_B(d\tau_j)$ bit operations [vzGG13, Corollary 11.18]. If
1460 $\text{sres}_i(\tilde{P}, \tilde{Q}) \neq 0$, let r be the remainder of degree i appearing in the Euclidean algorithm of \tilde{P}
1461 and \tilde{Q} . We know that r and $\text{Sres}_i(\tilde{P}, \tilde{Q})$ are equal up to a constant [BPR06, Corollary 8.34],
1462 thus $\text{Sres}_i(\tilde{P}, \tilde{Q}) = \frac{\text{Lc}_y(\text{Sres}_i(\tilde{P}, \tilde{Q}))}{\text{Lc}_y(r)} r = \frac{\text{sres}_i(\tilde{P}, \tilde{Q})}{r_{|0}} r$, which directly implies that $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q}) =$
1463 $\frac{\text{sres}_i(\tilde{P}, \tilde{Q})}{r_{|0}} r_{|1}$. Using Theorem 51, we can compute $r_{|0}$ and $r_{|1}$ in $\mathbb{Z}_{\mu_j}[y]$ in $\tilde{O}_B(d\tau_j)$ bit operations,
1464 which yields $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})$.

1465 Thus, for a given μ_j , computing the two first subresultant coefficients $\text{sres}_i(\tilde{P}, \tilde{Q})$ and
1466 $\text{sres}_{i,i-1}(\tilde{P}, \tilde{Q})$ for $2d^2$ values of k in \mathbb{Z}_{μ_j} costs $\tilde{O}_B(d^3\tau_j)$ bit operations. Then using fast interpola-
1467 tion [vzGG13, Corollary 10.12], we can recover $\text{sres}_i(P, Q) \bmod \mu_j$ and $\text{sres}_{i,i-1}(P, Q) \bmod \mu_j$ in
1468 $\tilde{O}_B(d^3\tau_j) = \tilde{O}_B(d^3 \log d\tau)$ bit operations, which sums up to $\tilde{O}_B(d^4\tau)$ for all $N = \tilde{O}(d\tau)$ values of
1469 μ_j . Finally, recovering all the $O(d^3)$ coefficients of $\text{sres}_i(P, Q)$ and $\text{sres}_{i,i-1}(P, Q)$ (whose bitsizes
1470 are smaller than N) can be done with $\tilde{O}_B(d^3N \log M) = \tilde{O}_B(d^4\tau)$ bit operations with the Chinese
1471 remainder algorithm [vzGG13, Theorem 10.25]. \square

1472 7 Computing isolating boxes from a RUR decomposition

1473 By definition, the RUR of an ideal I defines a mapping between the roots of a univariate poly-
1474 nomial and the solutions of I . Based on this mapping, Algorithm 9 computes isolating boxes

1475 using univariate isolation and approximate polynomial evaluation. Section 7.1 recalls or proves
 1476 several complexity results on isolation and evaluation of univariate polynomials. In Section 7.2,
 1477 the isolation algorithm using fast approximate multipoint evaluation is presented and analyzed in
 1478 Theorem 61.

1479 7.1 Preliminaries

1480 We start with some basic definitions. In addition, we recall some bounds on univariate polynomial
 1481 roots and their separation (for a single root and also amortized over all the roots), the complexity
 1482 of isolating the roots of a univariate polynomial, and elementary results on approximate polynomial
 1483 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

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

1484 of width 2^{-L+1} in \mathbb{C} that contains x .

1485 For a complex root γ of a polynomial $f \in \mathbb{Z}[x]$ and an arbitrary positive integer L , we say that a
 1486 connected region D in \mathbb{C} (typically, we consider a disk or a box) is isolating for γ (or that D isolates
 1487 γ) if it contains γ but no other root of f . We define the *separation of γ* (with respect to f), denoted
 1488 $\text{sep}(\gamma, f)$, to be the minimal distance between γ and any root γ' of f , with $\gamma' \neq \gamma$. The *separation*
 1489 *of f* is defined as $\text{sep}(f) = \min_{\gamma: f(\gamma)=0} \text{sep}(\gamma, f)$. The same notions for a zero-dimensional ideal of
 1490 $\mathbb{Z}[x, y]$ are also naturally defined.

1491 We now recall some well-known facts about the separations and the magnitudes of the complex
 1492 roots of a univariate polynomial f of degree d with integer coefficients of bitsize at most τ .

1493 **Lemma 53** ([Yap00, §6.2 Lemma 6.5]). *For any root $\gamma \in \mathbb{C}$ of f , $M(\gamma) = 2^{O(\tau)}$.*

1494 **Lemma 54** ([SY11]). *If f is squarefree, $\prod_{\{\gamma \text{ root of } f\}} \min(1, \text{sep}(\gamma, f)) = 2^{-\tilde{O}(d\tau)}$.*

1495 **Lemma 55** ([Yap00, Lemma 6.34]). *Let f and g be coprime polynomials of degree at most d with
 1496 integer coefficients of bitsize at most τ . Then, for any root $\gamma \in \mathbb{C}$ of f , $|g(\gamma)| = 2^{-O(d(\tau + \log d))}$.*

1497 **Lemma 56** ([MSW15, Theorem 5]). *We can compute isolating disks D_i with radius $r_i < \frac{\text{sep}(\gamma_i, f)}{64d}$
 1498 for all complex roots γ_i of f using $\tilde{O}_B(d^3 + d^2\tau)$ bit operations. For an arbitrary positive integer
 1499 L , we can compute corresponding L -bit approximations $\tilde{\gamma}_i$ for all roots using $\tilde{O}_B(d^3 + d^2\tau + dL)$ bit
 1500 operations.*

1501 *Proof.* The first part follows directly from [MSW15, Theorem 5]. In addition, [MSW15, Theorem
 1502 5] also states that we can further refine the disks D_i such that each of them has radius less than
 1503 2^{-L-2} using $\tilde{O}_B(d^3 + d^2\tau + dL)$ bit operations. In addition, the centers of the disks are computed
 1504 in dyadic form. We can thus round the center of each disk D_i to an absolute precision of size
 1505 2^{-L-2} to obtain an L -bit approximation $\tilde{\gamma}_i$ of each root γ_i of f . The bit complexity of rounding
 1506 all the disks' centers is linear in the total bitsize of the dyadic coordinates, which is bounded by
 1507 $\tilde{O}_B(d^3 + d^2\tau + dL)$, the complexity of the algorithm that computes them. \square

1508 We further remark that there also exist dedicated real root isolation and refinement meth-
 1509 ods [SM15, KS15a] that compute isolating intervals of size 2^{-L} for all real roots of f with a number
 1510 of bit operations that is comparable to the bound stated in Lemma 56. When computing the solu-
 1511 tions of a bivariate system (see Section 7.2), the choice of an efficient univariate solver is critical,
 1512 and thus we propose to use a dedicated method for real root finding if only the real solutions of
 1513 the bivariate system are asked for.

1514 Now, suppose that we want to approximately evaluate a polynomial $g \in \mathbb{Z}[x]$ of degree d_g with
 1515 integer coefficients of bitsize τ_g at all roots of f . More precisely, for a given positive integer L , we
 1516 are aiming for L -bit approximations \tilde{y}_i of the values $y_i = g(\gamma_i)$, where $\gamma_1, \dots, \gamma_d$ denote the roots
 1517 of f . For this, we use fast approximate multipoint evaluation.

1518 **Lemma 57** ([KS15b, Theorem 22]). *Let $x_1, \dots, x_{d_g} \in \mathbb{C}$ such that, for each of them, an L' -*
 1519 *bit approximation can be accessed in $O_B(L')$ bit operations. For any positive integer L , we can*
 1520 *compute L -bit approximations of all values $y_i = g(x_i)$ using $\tilde{O}_B(d_g(L + \tau_g + d_g\Gamma))$ bit operations,*
 1521 *where $\Gamma \geq 1$ is an upper bound on the maximum of all values $\log M(x_i)$. For the computation, we*
 1522 *need L' -bit approximations of all points x_i , where $L' = L + \tilde{O}(\tau_g + d_g\Gamma)$.*

1523 We remark that the multipoint evaluation algorithm from [KS15b] uses certified interval arith-
 1524 metic based on fixed-point computations. It adaptively increases the (absolute) working precision
 1525 L' during the computation. That is, in each iteration, it asks for L' -bit approximations \tilde{x}_i of the
 1526 points x_i , and if it does not succeed to compute L -bit approximations \tilde{y}_i of the values y_i , it doubles
 1527 the precision and restarts. Hence, the algorithm might also succeed with a smaller precision than
 1528 the precision predicted in the worst-case.

1529 **Lemma 58.** *Let $f \in \mathbb{Z}[x]$ be a polynomial of degree d with integer coefficients of bitsize at most*
 1530 *τ and let $\gamma_1, \dots, \gamma_d$ denote the roots of f . Let $g \in \mathbb{Z}[x]$ be a polynomial of degree $d_g = O(d)$ with*
 1531 *integer coefficients of bitsize at most τ_g . Then, for any given positive integer L , we can compute*
 1532 *L -bit approximations of all values $g(\gamma_i)$ using a number of bit operations bounded by $\tilde{O}_B(d^3 + d^2\tau +$
 1533 *$d(L + \tau_g)$).**

1534 *Proof.* Applying Lemma 57 $\lceil d/d_g \rceil$ times, L -bit approximations of d values $g(x_i)$ can be computed
 1535 with $\tilde{O}_B(\lceil d/d_g \rceil d_g(L + \tau_g + d_g\Gamma))$ bit operations assuming that we can access each L' -bit approxi-
 1536 mation of x_i in $O_B(L')$ bit operations. Moreover, as mentioned above, the L -bit approximations
 1537 of the $g(x_i)$ are computed iteratively by doubling L' at every iteration and the algorithm stops with
 1538 $L' = L + \tilde{O}(\tau_g + d_g\Gamma)$. Thus, the number of iterations is in $O(\log(L + \tau_g + d_g\Gamma))$.

1539 By Lemma 56, L' -bit approximations of the d roots of f can be computed in $\tilde{O}_B(d^3 + d^2\tau + dL')$
 1540 bit operations. Thus, these approximations can be computed for all iterations in $\tilde{O}_B(d^3 + d^2\tau +$
 1541 $d(L + \tau_g + d_g\Gamma))$ bit operations.

1542 The total complexity is thus in $\tilde{O}_B(\lceil d/d_g \rceil d_g(L + \tau_g + d_g\Gamma)) + \tilde{O}_B(d^3 + d^2\tau + d(L + \tau_g + d_g\Gamma))$.
 1543 The result follows since $d_g = O(d)$ and since $\Gamma = O(\tau)$ by Lemma 53. \square

1544 We can further extend the above result to the evaluation of a fraction $G = \frac{g_1}{g_2}$ at the roots γ_i
 1545 of f , where g_1 and g_2 are both polynomials of degree bounded by $O(d)$ with integer coefficients of
 1546 bitsize less than τ_G , and g_2 is coprime with f .

1547 **Lemma 59.** *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 co-*
 1548 *efficients of bitsize at most τ_G . Suppose that g_2 does not vanish at any of the roots $\gamma_1, \dots, \gamma_d$*
 1549 *of f . Then, for any given positive integer L , we can compute L -bit approximations of all values*
 1550 *$y_i = G(\gamma_i)$ using a number of bit operations bounded by $\tilde{O}_B(d^3 + d^2(\tau + \tau_G) + dL)$.*

Proof. According to Lemma 55, it holds that $|g_2(\gamma_i)| = 2^{-\tilde{O}(d(\tau+\tau_G))}$ for all i . Now, in a first step, we compute L' -approximations $\tilde{y}_{2,i}$ of all $y_{2,i} = g_2(\gamma_i)$ for $L' = 1, 2, 4, \dots$ until $|\tilde{y}_{2,i}| > 2^{-L'+1}$, and thus $2|\tilde{y}_{2,i}| > |y_{2,i}| > |\tilde{y}_{2,i}|/2$. Notice that we succeed in doing so for an $L' = L'_0$ in $\tilde{O}(d(\tau + \tau_G))$. Then, for an $L' \geq L'_0$, we can compute L' -approximations $\tilde{y}_{1,i} = 2^{-L'-2} \cdot (m_{1,i} + i \cdot n_{1,i})$ and $\tilde{y}_{2,i} = 2^{-L'-2} \cdot (m_{2,i} + i \cdot n_{2,i})$ of the values $y_{1,i} = g_1(\gamma_i)$ and $y_{2,i}$, respectively, with $m_{1,i}, m_{2,i}, n_{1,i}, n_{2,i} \in \mathbb{Z}$. Notice that each of the latter integers has bitsize $\tilde{O}(d(\tau + \tau_G))$ as $|g_1(\gamma_i)|, |g_2(\gamma_i)| \leq (d_G + 1) \cdot 2^{\tau_G} \cdot M(\gamma_i)^{d_G} \leq 2^{O(\log d + \tau_G + d\tau)}$ for all i . Hence, we conclude that

$$\begin{aligned} \left| \frac{\tilde{y}_{1,i}}{\tilde{y}_{2,i}} - G(x_i) \right| &= \frac{|\tilde{y}_{1,i} \cdot y_{2,i} - y_{1,i} \cdot \tilde{y}_{2,i}|}{|y_{2,i} \cdot \tilde{y}_{2,i}|} \leq \frac{|\tilde{y}_{1,i} - y_{1,i}| \cdot |y_{2,i}| + |y_{1,i}| \cdot |y_{2,i} - \tilde{y}_{2,i}|}{|y_{2,i} \cdot \tilde{y}_{2,i}|} \\ &\leq 2^{-L'} \cdot \frac{4}{|\tilde{y}_{2,i}|^2} \cdot (d_G + 1) \cdot 2^{\tau_G} \cdot M(\gamma_i)^{d_G} = 2^{-L'} \cdot 2^{\tilde{O}(d(\tau+\tau_G))}. \end{aligned}$$

1551 Notice that the above bound on the approximation error is explicit (i.e. computable). Thus, we
 1552 can directly estimate the error from the given values L' , $|\tilde{y}_{2,i}|$, d_G , τ_G , and $M(\gamma_i)$. Hence, we
 1553 may consider $L' = L'_0, L'_0 + 2, L'_0 + 4, \dots$ until we can guarantee that $\left| \frac{\tilde{y}_{1,i}}{\tilde{y}_{2,i}} - G(x_i) \right| < 2^{-L-2}$. For
 1554 this, we need to increase L' at most $O(\log(d(\tau + \tau_G)))$ many times, and we succeed for an L' in
 1555 $\tilde{O}(L + d(\tau + \tau_G))$. Then, we approximate each fraction $\frac{\tilde{y}_{1,i}}{\tilde{y}_{2,i}} = \frac{m_{1,i} + i \cdot n_{1,i}}{m_{2,i} + i \cdot n_{2,i}}$ by a corresponding $(L+1)$ -
 1556 bit approximation to obtain an L -bit approximation of $G(x_i)$. Due to Lemma 58, the total bit
 1557 complexity for computing the fractions $\frac{\tilde{y}_{1,i}}{\tilde{y}_{2,i}}$ is in $\tilde{O}_B(d^3 + d^2\tau + d(L + d(\tau + \tau_G) + \tau_G)) = \tilde{O}_B(d^3 +$
 1558 $d^2\tau + d^2\tau_G + dL)$, whereas the total bit complexity for computing the $(L+1)$ -bit approximations
 1559 of the fractions $\frac{\tilde{y}_{1,i}}{\tilde{y}_{2,i}}$ is in $\tilde{O}_B(d^2(\tau + \tau_G) + dL)$. Indeed, using fast integer division, computing an
 1560 L -bit approximation from a rational has a bit complexity that is softly linear in L and the bitsize
 1561 of the rational. \square

1562 7.2 Isolating boxes

1563 We now give a method for computing disjoint isolating boxes for the solutions $\sigma \in \mathbb{C}^2$ of a zero-
 1564 dimensional system $P = Q = 0$, where $P, Q \in \mathbb{Z}[x, y]$ are coprime polynomials of total degree
 1565 at most d with integer coefficients of bitsize at most τ . More specifically, for a given L , we first
 1566 compute L -bit approximations¹² $\tilde{\sigma}_{i,j}$ of the solutions $\sigma_{i,j} = (x_{i,j}, y_{i,j})$, $1 \leq j \leq d_i = \deg f_i$, of each
 1567 factor $\text{RUR}_i = (f_i, f_{i,1}, f_{i,x}, f_{i,y})$ in the RUR decomposition $(\text{RUR}_i)_{i \leq d}$ of $\{P, Q\}$ as computed by
 1568 Algorithm 6 or 6'. This is achieved by first computing sufficiently small isolating disks for the roots
 1569 $\gamma_{i,j}$ 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 $\frac{f_{i,y}}{f_{i,1}}$
 1570 at the roots $\gamma_{i,j}$ to an absolute error less than 2^{-L} . From the corresponding L -bit approximations
 1571 $\tilde{x}_{i,j}$ 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}
 1572 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
 1573 j , the boxes $B_{i,j}$ do not overlap, then they are already isolating for the solutions of $P = Q = 0$.
 1574 Otherwise, we have to increase L until the boxes do not overlap. We give details in Algorithm 9.

1575 In order to bound the complexity of the above approach, we first need to derive bounds on the
 1576 separations of the solutions $\sigma_{i,j} = (x_{i,j}, y_{i,j})$ of the factor RUR_i . In addition, we derive amortized
 1577 bounds on the separations of all solutions of the system $P = Q = 0$.

1578 **Lemma 60.** *Let $P, Q \in \mathbb{Z}[x, y]$ and $\text{RUR}_i = (f_i, f_{i,1}, f_{i,x}, f_{i,y})$, $i \in \mathcal{I}$, be as defined in the input of*
 1579 *Algorithm 9 and let d_i and τ_i be the maximum degree and bitsize of the polynomials in RUR_i . In*

¹²We 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.

1580 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
 1581 of solutions. Then,

1582 (a) $\sum_{\sigma \in V(I_i)} \log M(\text{sep}(\sigma, I_i)^{-1}) = \tilde{O}(d_i \tau_i) = \tilde{O}(d_i(d^2 + d\tau)) = \tilde{O}(d^4 + d^3\tau),$

1583 (b) $\log M(|\sigma|) = \tilde{O}(d\tau)$ for all $\sigma = (\sigma_x, \sigma_y) \in V(I_i)$, where $|\sigma| = \max(|\sigma_x|, |\sigma_y|),$

1584 (c) $\sum_{\sigma \in V(I)} \log M(\text{sep}(\sigma, I)^{-1}) = \tilde{O}(d^4 + d^3\tau).$

1585 *Proof.* Let $(x, y) \mapsto x + ay$ be the separating form for $\{P, Q\}$ with $a \in \mathbb{Z}$ of bitsize $O(\log d),$
 1586 as computed by Algorithm 6 or 6' as part of the input of Algorithm 9. This separating
 1587 form defines a one-to-one mapping from the set of solutions $\sigma \in V(I_i)$ to the set of roots
 1588 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
 1589 $\text{sep}(\sigma, I_i) = |\sigma - \sigma'|,$ and let γ and γ' be the corresponding roots of f_i . Then, we have
 1590 $\text{sep}(\gamma, f_i) \leq |\sigma_x - \sigma'_x| + |a| \cdot |\sigma_y - \sigma'_y| \leq (|a| + 1) \text{sep}(\sigma, I_i),$ or equivalently $\text{sep}(\sigma, I_i)^{-1} \leq$
 1591 $(|a| + 1) \text{sep}(\gamma, f_i)^{-1}.$ We thus have $\log M(\text{sep}(\sigma, I_i)^{-1}) \leq \log(|a| + 1) + \log M(\text{sep}(\gamma, f_i)^{-1}).$ On the
 1592 other hand, f_i is squarefree since it is the first polynomial of the RUR of a radical ideal (see Algo-
 1593 rithm 6 or 6'). Thus, Lemma 54 yields that $\prod_{\{\gamma \text{ root of } f_i\}} \min(1, \text{sep}(\gamma, f_i))^{-1} = 2^{\tilde{O}(d_i \tau_i)}$ and thus
 1594 $\sum_{\{\gamma \text{ root of } f_i\}} \log M(\text{sep}(\gamma, f_i)^{-1}) = \tilde{O}(d_i \tau_i).$ Part (a) follows directly since a has bitsize $O(\log d)$
 1595 and, by Theorem 37, $d_i \leq d^2$ and $\tau_i = \tilde{O}(d^2 + d\tau).$

1596 Part (b) follows directly from the fact that each coordinate of a solution σ is a root of either the
 1597 resultant polynomial $\text{Sres}_{x,0}(P, Q) \in \mathbb{Z}[y]$ or $\text{Sres}_{y,0}(P, Q) \in \mathbb{Z}[x],$ and both of these polynomials
 1598 have integer coefficients of bitsize $\tilde{O}(d\tau)$ by Lemma 3. For part (c), notice that, by definition of
 1599 RUR decompositions (Definition 36), the roots of $f = \prod_i f_i$ are exactly the images of the solutions
 1600 of $\{P, Q\}$ through the mapping $(x, y) \mapsto x + ay.$ The degree of f is thus at most $d^2.$ Furthermore,
 1601 the f_i are monic (by Definition 34), thus f is monic and the bitsize of its coefficients is at most that
 1602 of the resultant of the sheared polynomials $P(t - ay, y)$ and $Q(t - ay, y)$ with respect to $y,$ which
 1603 bitsize is in $\tilde{O}(d^2 + d\tau)$ (see e.g. [BLPR15, Lemma 7]). Hence, the same argument as for the proof
 1604 of part (a) yields that $\text{sep}(\gamma, f) \leq (|a| + 1) \text{sep}(\sigma, I)$ and then the result. \square

1605 The following theorem analyzes the complexity of the isolation of a system $\{P, Q\}$ from a RUR
 1606 decomposition as computed in Section 6.

1607 **Theorem 61.** *Let $P, Q \in \mathbb{Z}[x, y]$ be coprime polynomials of degree at most d with integer coefficients*
 1608 *of bitsize at most $\tau.$ Algorithm 9 computes isolating boxes for all complex solutions of $P = Q = 0$*
 1609 *using $\tilde{O}_B(d^6 + d^5\tau)$ bit operations.*

1610 *Proof.* As argued at the end of the proof of Lemma 60, f is a polynomial of degree at most d^2
 1611 with coefficients of bitsize $\tilde{O}(d^2 + d\tau),$ and thus the bit complexity of Step 2 in Algorithm 9 is
 1612 $\tilde{O}_B(d^6 + d^5\tau)$ (Lemma 56). In addition, the degree of all polynomials $f_{i,y}$ and $f_{i,1}$ is at most the
 1613 degree d_i of f_i (Definition 34), and the bitsize of their coefficients is in $\tilde{O}(d^2 + d\tau)$ (Theorem 37).
 1614 According to Lemma 60(c), the distance between any two solutions of $P = Q = 0$ is lower bounded
 1615 by $2^{-\tilde{O}(d^4 + d^3\tau)},$ which implies that Algorithm 9 terminates with L in $\tilde{O}(d^4 + d^3\tau).$ We also have
 1616 that the loop in Line 5 is executed $\log L = O(\log d\tau)$ times and thus, ignoring the polylogarithmic
 1617 factors, it is sufficient to study the complexity of the last call to this loop. From Lemma 59, the
 1618 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
 1619 in Step 8 is in $\tilde{O}_B(d_i^3 + d_i^2(d^2 + d\tau) + d_i(d^4 + d^3\tau)) = \tilde{O}_B(d_i^3 + d_i(d^4 + d^3\tau)).$ Summing over all
 1620 i yields the bound $\tilde{O}_B(d^6 + d^5\tau)$ for the total bit complexity of this step, since the sum of all d_i
 1621 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

Algorithm 9 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 $(\text{RUR}_i)_{i \in \mathcal{I}} = (f_i, f_{i,1}, f_{i,x}, f_{i,y})_{i \in \mathcal{I}}$ the RUR decomposition of $\{P, Q\}$ as computed by Algorithm 6 or 6'

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

- 1: $f = \prod_{i \in \mathcal{I}} f_i$
 - 2: Compute isolating disks $D_\gamma \subset \mathbb{C}$ for all complex roots γ of f
 - 3: $S = \{(\gamma, \gamma') \mid \gamma \text{ and } \gamma' \text{ distinct roots of } f\}$
 - 4: $L = 1$
 - 5: **repeat**
 - 6: $L = 2L$
 - 7: **for** $i \in \mathcal{I}$ **do**
 - 8: 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 59)
 - 9: **until** for all pairs $(\gamma, \gamma') \in S$, $|\tilde{\sigma}_{\gamma,x} - \tilde{\sigma}_{\gamma',x}| > 2^{-L+2}$ or $|\tilde{\sigma}_{\gamma,y} - \tilde{\sigma}_{\gamma',y}| > 2^{-L+2}$
 - 10: **return** $\{B(\tilde{\sigma}_{\gamma,x}) \times B(\tilde{\sigma}_{\gamma,y}) \mid \gamma \text{ root of } f\}$
-

1622 the corresponding solutions in $I = \langle P, Q \rangle$. From Definition (7) of the box associated to an L -bit
 1623 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
 1624 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
 1625 algorithm. Testing these inequalities can be done in $O_B(\log(M(|\sigma|) + M(|\sigma'|)) + \log M(|\sigma - \sigma'|^{-1}))$
 1626 bit operations (where $|\sigma| = \max(|\sigma_x|, |\sigma_y|)$) because, for each comparison, the first term bounds
 1627 the number of bits before the binary point, and in the case where these bits coincide, the second
 1628 term bounds the number of bits after the binary point that need to be considered. Notice that
 1629 the sum of $\log M(|\sigma - \sigma'|^{-1})$ over the $O(d^4)$ pairs (σ, σ') is at most $d^2 \sum_{\sigma \in V(I)} \log M(\text{sep}(\sigma, I)^{-1})$.
 1630 Thus, summing over the $O(d^4)$ pairs and using Lemma 60 yields the bound $\tilde{O}_B(d^6 + d^5\tau)$ for the
 1631 total bit complexity of Step 9, which concludes the proof. \square

1632 **Remark 62.** Algorithm 9 computes isolating boxes for only the solutions of one specific RUR_i if
 1633 we set $f = f_i$ in Step 1. Following the proof of Theorem 61, it is straightforward to prove that the
 1634 bit complexity of the algorithm then decreases to $\tilde{O}_B(d_i^2(d^2 + d\tau))$ where the degree d_i of RUR_i can
 1635 be much smaller than d^2 .

1636 **Remark 63.** In order to isolate only the real solutions of $P = Q = 0$, it suffices to iterate in
 1637 Algorithm 9 over the real roots of f since the separating form $(x, y) \mapsto x + ay$ is a one-to-one
 1638 mapping between the real solutions of $P = Q = 0$ and the real roots of f (see Proposition 35). Note
 1639 that, in this case, it is preferable to consider a dedicated real root isolation method in Step 2 for
 1640 computing the real roots of f .

1641 **Remark 64.** In order to achieve the complexity bound $\tilde{O}_B(d^6 + d^5\tau)$ in Step 8, we used an asymp-
 1642 totically fast method for the evaluation of a polynomial at many points (in Lemma 57). In practice,
 1643 such methods have not proven to be very efficient, and thus sequential evaluation is typically used
 1644 instead. Without detailing the proof, we claim that a more careful analysis would show that even
 1645 sequential evaluation yields the same complexity bound. The main reason is that, for each solutions
 1646 σ of $P = Q = 0$, it suffices to compute a box of a size that is not much smaller than the separation
 1647 $\text{sep}(\sigma, I)$ of σ . Hence, in the algorithm, it is sufficient to stop the refinement of the approximation

1648 of σ as soon as the corresponding box is already isolated from all other boxes. Then, using amortized
1649 bounds instead of those given in Lemmas 53 and 55 yields the claimed bound.

1650 8 Conclusion

1651 We have studied the problem of solving a bivariate system of two polynomials of degree bounded
1652 by d and bitsize bounded by τ via a combination of triangular decomposition and RUR. We have
1653 designed algorithms of worst-case complexity $\tilde{O}_B(d^6 + d^5\tau)$ for all the steps: finding a separating
1654 linear form, computing a RUR decomposition and computing isolating boxes of the solutions. This
1655 worst-case upper bound is not likely to be easily improved since it is also the best one known for the
1656 isolation of the roots of the resultant of the input polynomials [MSW15, Theorem 5]. However, one
1657 hope to improve this complexity is to consider an adaptive complexity that depends on geometric
1658 parameters such as the minimal distance between two roots, as in [KS15a] for example.

1659 In the Las Vegas setting, we also proposed an algorithm of expected complexity $\tilde{O}_B(d^5 + d^4\tau)$
1660 for finding a separating form and computing a RUR decomposition. In the radical and generic
1661 case, the Monte-Carlo algorithm of [LMS13] computes a modified equiprojectable decomposition
1662 that coincides in this case to a RUR. Even if it is restricted to the radical case, this algorithm is
1663 remarkable since its complexity is $\tilde{O}_B(d^{4+\varepsilon} + d^{3+\varepsilon}\tau)$, for $\varepsilon > 0$ arbitrarily small, which almost
1664 matches the upper bound $\tilde{O}(d^4 + d^3\tau)$ on the size of the output (see Corollary 38) which is most
1665 likely tight in the worst case. One natural question is whether it is possible to achieve such a
1666 complexity in the Las Vegas setting.

1667 Finally, we note that, for computing a separating linear form of an *arbitrary* system $\{P, Q\}$,
1668 the algorithm presented here is likely to be purely theoretical because (i) considering the system
1669 $\{PQ, \frac{\partial PQ}{\partial y}\}$ instead $\{P, Q\}$ essentially doubles the degree of the input polynomials, and (ii) the
1670 shearing of the coordinate systems, done to avoid vertical asymptotes, spoils the sparsity of the
1671 coefficients and increases their bitsize, which is not efficient in practice. However, for the problem
1672 of computing the critical points of a curve, there is some hope that our algorithm can be efficient
1673 not only in theory but also in practice.

1674 References

- 1675 [ABRW96] M.-E. Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Multiplicities and idempotents
1676 for zerodimensional systems. In *Algorithms in Algebraic Geometry and Applications*,
1677 volume 143 of *Progress in Mathematics*, pages 1–20. Birkhäuser, 1996.
- 1678 [AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Ann. of Math*, 160(2):781–793,
1679 2004.
- 1680 [BLP⁺14] Y. Bouzidi, S. Lazard, M. Pouget, G. Moroz, and F. Rouillier. Improved algorithm
1681 for computing separating linear forms for bivariate systems. In *Proceedings of the*
1682 *39th International Symposium on International Symposium on Symbolic and Algebraic*
1683 *Computation, ISSAC '14, Kobe, Japan, July 2014*. ACM.
- 1684 [BLPR15] Y. Bouzidi, S. Lazard, M. Pouget, and F. Rouillier. Separating linear forms and
1685 rational univariate representations of bivariate systems. *J. Symb. Comput.*, pages
1686 84–119, 2015.
- 1687 [BPR06] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10
1688 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2nd edition, 2006.

- 1689 [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.
- 1690
- 1691
- 1692 [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.
- 1693
- 1694 [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.
- 1695
- 1696
- 1697 [EK03] M. El Kahoui. An elementary approach to subresultants theory. *J. Symb. Comput.*, 35(3):281–292, 2003.
- 1698
- 1699 [GLS01] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner free alternative for solving polynomial systems. *J. of Complexity*, 17(1):154–211, 2001.
- 1700
- 1701 [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.
- 1702
- 1703
- 1704 [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.
- 1705
- 1706
- 1707 [KS15b] A. Kobel and M. Sagraloff. On the complexity of computing with planar algebraic curves. *Journal of Complexity*, 31(2):206–236, 2015.
- 1708
- 1709 [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.
- 1710
- 1711 [LMS13] R. Lebreton, E. Mehrabi, and É. Schost. On the complexity of solving bivariate systems: The case of non-singular solutions. In *Proceedings of the 38th International Symposium on International Symposium on Symbolic and Algebraic Computation*, ISSAC ’13, pages 251–258, New York, NY, USA, 2013. ACM.
- 1712
- 1713
- 1714
- 1715 [LR01] T. Lickteig and M.-F. Roy. Sylvester-Habicht Sequences and Fast Cauchy Index Computation. *J. Symb. Comput.*, 31(3):315–341, 2001.
- 1716
- 1717 [MB74] R. Moenck and A. Borodin. Fast modular transforms. *Journal of Computer and System Sciences*, 8:366–386, 1974.
- 1718
- 1719 [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.
- 1720
- 1721
- 1722 [Pan02] V. Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *J. Symb. Comput.*, 33(5):701 – 733, 2002.
- 1723
- 1724 [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.
- 1725
- 1726

- 1727 [Rou99] F. Rouillier. Solving zero-dimensional systems through the rational univariate repre-
1728 sentation. *J. of Applicable Algebra in Engineering, Communication and Computing*,
1729 9(5):433–461, 1999.
- 1730 [SM15] M. Sagraloff and K. Mehlhorn. Computing real roots of real polynomials. *Journal of*
1731 *Symbolic Computation*, (0):-, 2015.
- 1732 [SY11] M. Sagraloff and C. K. Yap. A simple but exact and efficient algorithm for complex
1733 root isolation. In *Proceedings of the 36th International Symposium on Symbolic and*
1734 *Algebraic Computation*, ISSAC '11, pages 353–360, New York, NY, USA, 2011. ACM.
- 1735 [vzGG13] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge Univ.
1736 Press, Cambridge, U.K., 3rd edition, 2013.
- 1737 [Yap00] C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press,
1738 Oxford-New York, 2000.