On the Complexity of Real Solving of Bivariate Polynomial Systems

Ioannis Z. Emiris
Elias P. Tsigaridas

Vienna, Preprint ESI 1890 (2007)

February 9, 2007

Supported by the Austrian Federal Ministry of Education, Science and Culture
Available via http://www.esi.ac.at

On the complexity of real solving of bivariate polynomial systems

Ioannis Z. Emiris
National Kapodistrian University of Athens, Greece

Elias P. Tsigaridas
INRIA Sophia-Antipolis, France

December 2006

Abstract

In this paper we present algorithmic and complexity results for polynomial sign evaluation over two real algebraic numbers, and for real solving of bivariate polynomial systems. Our main tool is signed polynomial remainder sequences; we exploit recent advances in univariate root isolation as well as multipoint evaluation techniques.

Keywords: Real algebraic number, root isolation, bit complexity, Sturm sequence.

AMS Subject Classification: 12D10: Polynomials: Location of zeros. 14P05: Real algebraic sets.

1 Introduction

The problem of algebraic system solving is fundamental. However, most of the algorithms treat the general case or consider solutions over an algebraically closed field. We focus on real solving in the bivariate case in order to provide precise complexity bounds and compare different algorithms. The problem is closely related to computing the topology of a plane real algebraic curve and other important operations in non-linear computational geometry and Computer-Aided Geometric Design. Another field of application is quantifier elimination.

Our approach is based on polynomial sign evaluation over two real algebraic numbers. Our main tool is signed polynomial remainder sequences; we exploit recent advances in univariate root isolation and we adapt standard multipoint evaluation techniques. Currently, we are working on the implementation of these methods. Another direction for future work is to exploit the fact that several of our polynomials are expressed in terms of determinants.

In what follows $\mathcal{O}_B$ means bit complexity and the $\widetilde{\mathcal{O}}_B$-notation means that we ignore polylogarithmic factors. For a polynomial $f \in \mathbb{Z}[X]$, $\mathsf{deg}(f)$ denotes its total degree, while $\deg_X(f)$ denotes its degree with respect to X. By $\mathcal{L}(f)$ we denote an upper bound on the bit-size of the coefficients of f (including a bit for the sign). For $\mathsf{a} \in \mathbb{Q}$, $\mathcal{L}(\mathsf{a})$ is the maximum bit-size of the numerator and the denominator.

Let $\mathsf{M}(\tau)$ denote the bit complexity of multiplying two integers of bit-size at most τ and $\mathsf{M}(d, \tau)$ denote the bit complexity of multiplying two univariate polynomials of degrees bounded by d and coefficient bit-size at most τ. Using FFT [3, 25, 27], the complexities of these operations are $\mathsf{M}(\tau) = \mathcal{O}_B(\tau \lg^{c_1} \tau)$ and $\mathsf{M}(d, \tau) = \mathcal{O}_B(d\tau \lg^{c_2}(d\tau))$ for suitable constants c_1, c_2. So we get $\mathsf{M}(d, \tau) = \widetilde{\mathcal{O}}_B(d\tau)$. Similarly, we can compute the remainder $f_1 \bmod f_2$ in $\widetilde{\mathcal{O}}_B(d\tau)$ if d and τ bound the degree and bit-size of both f_1, f_2.

The paper is organized as follows. The next section introduces our main tools. Section 3 focuses on multipoint evaluation of polynomials. The following section extends the discussion to several variables. Section 5 proposes two different approaches for the real solving of bivariate systems. The appendix contains some background on fan-in/fan-out.

This work has been continued and extended in [8], where the implementation of our algorithms is discussed and experimental results shown.

2 Preliminaries

We recall the main ingredients related to signed polynomial remainder sequences. Let $f = \sum_{k=0}^{p} a_k x^k, g = \sum_{k=0}^{q} b_k x^k \in \mathbb{Z}[x]$ where $\mathsf{deg}(f) = p \geq q = \mathsf{deg}(g)$ and $\mathcal{L}(f), \mathcal{L}(g) \leq \tau$. We denote by $\mathsf{rem}(f, g)$ and $\mathsf{quo}(f, g)$ the remainder and the quotient, respectively, of the Euclidean division of f by g, in $\mathbb{Q}[x]$.

Definition 2.1 *[15] The* signed polynomial remainder sequence *of f and g, denoted by* $\mathbf{sPRS}(f, g)$*, is the polynomial sequence*

$$R_0 = f, R_1 = g, R_2 = -\mathsf{rem}(f, g), \ldots, R_k = -\mathsf{rem}(R_{k-2}, R_{k-1})$$

where $\mathsf{rem}(R_{k-1}, R_k) = 0$*. The* quotient sequence *of f and g is the polynomial sequence $\{Q_i\}_{0 \leq i \leq k}$, where $Q_i = \mathsf{quo}(R_i, R_{i+1})$ and the* quotient boot *is $(Q_0, Q_1, \ldots, Q_{k-1}, R_k)$.*

There is a huge bibliography on signed polynomial remainder sequences (c.f [2, 25, 27] and references therein). The work [26] presents a unified approach to subresultants, while [10] studied the subresultants in arbitrary commutative rings. For Sturm-Habicht (or Sylvester-Habicht) sequences the reader may refer to [13] (see also [2, 15, 16]).

In this paper we consider the Sturm-Habicht sequence of f and g, denoted $\mathbf{SR}(f, g)$, which contains polynomials that are proportional to the polynomials in $\mathbf{sPRS}(f, g)$. Sturm-Habicht sequences achieve better bounds on the bit-size of the coefficients and have good specialization properties, since they are defined through determinants.

Theorem 2.2 [19, 16] *We can compute* $\mathbf{SR}(f,g)$ *in* $\widetilde{\mathcal{O}}_B(p^2q\tau)$. *Moreover,* $\mathcal{L}\left(\mathbf{SR}_j(f,g)\right) = \mathcal{O}((p+q)\tau)$.

Theorem 2.3 [15, 19] *We can compute the quotient boot of* $\mathbf{SRQ}(f,g)$*, any polynomial in the sequence* $\mathbf{SR}(f,g)$*, the resultant, and the* gcd *of* f *and* g *in* $\widetilde{\mathcal{O}}_B(pq\tau)$. *Moreover,* $\mathcal{L}\left(\mathbf{SRQ}_j(f,g)\right) = \mathcal{O}(p\tau)$.

Theorem 2.4 [15, 19] *Consider polynomials* f, g *with degrees* p, q *respectively. We can evaluate* $\mathbf{SR}(f,g)$ *over a number* $\mathsf{a} \in \mathbb{Q} \cup \{\pm\infty\}$*, such that* $\mathcal{L}(\mathsf{a}) = \sigma$*, in* $\widetilde{\mathcal{O}}_B(qp\tau + q^2\sigma + p^2\sigma)$. *If the sign of* $f(a)$ *is known, then the bound becomes in* $\widetilde{\mathcal{O}}_B(qp\tau + q^2\sigma)$.

Proof. Here, $\mathbf{SR}(f,g;\,\mathsf{a})$ denotes the evaluated sequence. Notice that $\mathbf{SR}_{q+1} = f$ and $\mathbf{SR}_q = g$. Let us forget for the moment $\mathbf{SR}_{q+1}$. We may assume that $\mathbf{SR}_{q-1}$ is computed, since the cost of computing one element of $\mathbf{SR}$ is the same as that of computing $\mathbf{SRQ}$ (Th. 2.3).

We follow [15]. For two polynomials A, B of degree bounded by D and bit-size bounded by L, we can compute $\mathbf{SR}(A,B)(\mathsf{a})$, where $\mathcal{L}(\mathsf{a}) \leq L$, in $\widetilde{\mathcal{O}}(\mathsf{M}(D,L))$. In our case, $D = \mathcal{O}(q)$ and $L = \mathcal{O}(\max\{p\tau, q\sigma\})$, thus the total cost is $\widetilde{\mathcal{O}}_B(q\max\{p\tau, q\sigma\})$.

The last step is to evaluate $\mathbf{SR}_{q+1}(\mathsf{a}) = f(\mathsf{a})$, in $\widetilde{\mathcal{O}}_B(p\max\{\tau, p\sigma\})$. Thus, the whole procedure has complexity $\widetilde{\mathcal{O}}_B(q\max\{p\tau, q\sigma\} + p\max\{\tau, p\sigma\})$. $\square$

Corollary 2.5 *In the case where we do not know the ordering of* p *and* q *the bound of Th. 2.4 becomes*

$$\widetilde{\mathcal{O}}_B\left(\min\{p,q\}\max\{p,q\}\tau + \min\{p,q\}^2\sigma + \max\{p,q\}^2\sigma\right)$$

The last term is omitted if f *is already evaluated on the given point.*

The output of the algorithm is an ordered list of the real algebraic numbers that are roots of f, in isolating interval representation, where no two intervals share a common endpoint; the list also contains the corresponding multiplicities. Thus, if γ is a real root of f then its representation is $\gamma \cong (f_{red}, [\mathsf{a}, \mathsf{b}])$, where f_{red} is the square-free part of f and $[\mathsf{a}, \mathsf{b}]$ is an isolating interval. Notice that the endpoints of the isolating intervals have bit-size $\mathcal{O}(p\tau_f)$ and that $\mathcal{L}(f_{red}) = \mathcal{O}(p + \tau_f)$.

Theorem 2.6 [11, 9] *(*CONSTRUCT*) Let* $f \in \mathbb{Z}[x]$ *have degree* p *and bit-size* τ_f. *We can compute the isolating interval representation of the real roots of* f *and compute their multiplicities in* $\widetilde{\mathcal{O}}_B(p^6 + p^4\tau_f^2)$.

There are some observations leading to the next result on evaluating sequences. First, it is not needed to evaluate f over the end points of the isolating interval. We can assume that we know this from the root isolation process. Second, when $q > p$, the first elements of $\mathbf{SR}(f,g)$ are $f, g, -f, -(g \bmod (-f)) \ldots,$

thus in the evaluated sequence $f(\mathsf{a}), g(\mathsf{a}), -f(\mathsf{a}), \ldots$ there is one sign variation irrespective of the sign of $g(\mathsf{a})$. Therefore, there is no need to evaluate g over the endpoints.

Corollary 2.7 [11] *(*SIGN_AT_1*) Given a real algebraic number $\alpha \cong (f, [\mathsf{a}, \mathsf{b}])$, such that $\deg(f) = p$, $\mathcal{L}(f) = \tau_f$, $\mathcal{L}(\mathsf{a}) = \mathcal{L}(\mathsf{b}) = \mathcal{O}(p\tau_f)$, and a polynomial $g \in \mathbb{Z}[x]$ such that $\deg(g) = q$ and $\mathcal{L}(g) = \tau_g$ we can compute $\operatorname{sign} g(\alpha)$, which we denote by* SIGN_AT_1(g, α) *in $\widetilde{\mathcal{O}}_B(pq \max\{\tau_f, \tau_g\} + p \min\{p, q\}^2 \tau_f)$.*

Proof. As explained above, there is no need to evaluate the polynomial of the biggest degree, i.e the first (and the second if $p < q$) of $\mathbf{SR}(f, g)$ over a and b. Thus the complexity is that of Cor. 2.5 without the last summand. Hence we get

$$\widetilde{\mathcal{O}}_B(\min\{p, q\} \max\{p, q\} \max\{\tau_f, \tau_g\} + \min\{p, q\}^2 p\tau_f)$$

It is known that $\operatorname{sign}(g(\alpha)) = \mathtt{VAR}(\mathbf{SR}(f, g)(\mathsf{a}_1)) - \mathtt{VAR}(\mathbf{SR}(f, g)(\mathsf{a}_2))$. Thus the complexity of the operation is two times the complexity of the evaluation of the sequence over the endpoints of the isolating interval. □

Theorem 2.8 (Davenport-Mahler-Mignotte) [23] *Let $A \in \mathbb{Z}[X]$, with $\deg(A) = d$ and $\mathcal{L}(A) = \tau$, where $A(0) \neq 0$. Let Ω be any set of k couples of indices (i, j) such that $1 \leq i < j \leq d$ and let the non-zero (complex) roots of A be $0 < |\gamma_1| \leq |\gamma_2| \leq \cdots \leq |\gamma_d|$. Then*

$$2^k \mathcal{M}(A)^k \geq \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \geq 2^{k - \frac{d(d-1)}{2}} \mathcal{M}(A)^{1-d-k} \sqrt{\mathsf{disc}(A)}.$$

If we count the bit-size of all isolating intervals, we end up with $\widetilde{\mathcal{O}}_B(p^2\tau_f)$ or $\widetilde{\mathcal{O}}_B(rp\tau_f)$, since there are $r + 1 \leq p$ of them, where r is number of real roots. However, we can do better.

Corollary 2.9 (Aggregate separation) *The sum of the bit-size of **all** isolating points of the real roots of f is $\widetilde{\mathcal{O}}_B(p^2 + p\tau_f)$.*

Proof. Notice that in the worst case the isolating numbers is of magnitude $\Delta_i = \frac{1}{2}|\alpha_i - \alpha_{c_i}|$, where α_{c_i} is the complex root that is closest to α_i. Hence the separation points sum up to $\frac{1}{2}\sum_{i=1}^{r} \Delta_i$ and thus the bit-size of all of them is bounded by

$$\lg \prod_{i=1}^{r} \Delta_i$$

Using the Davenport-Mahler-Mignotte bound (Th. 2.8) one can prove [23] that

$$\prod_{i=1}^{r} \Delta_i \geq 2^{-\mathcal{O}(p^2 + p\tau_f)}$$

We conclude that the overall bit-size of the output is $\widetilde{\mathcal{O}}_B(p^2 + p\tau_f)$ in the worst case. □

3 Multipoint evaluation

In this section we obtain complexity bounds on evaluating a polynomial over a set of algebraic numbers, which have the same defining polynomial. We shall give more than one algorithm, where we improve the direct method to a more efficient one.

The central question is to evaluate $g(x)$ over all real roots of $f(x)$. Recall that the isolating points of f are of bit-size $p\tau_f$. In order to compute the sign g over the real root of f is suffices to compute the evaluation of $\mathbf{SR}(f,g)$ over all the endpoints. The obvious technique is to apply Cor. 2.7 as many as $r \le p$ times, where r is the number of real roots of f. But we may do better.

Lemma 3.1 *Let $f,g \in \mathbb{Z}[x]$, such that $\mathsf{deg}(f) = p$, $\mathsf{deg}(g) = q$, $\mathcal{L}(f) = \tau_f, \mathcal{L}(g) = \tau_g$ and $\tau = \max\{\tau_f, \tau_g\}$. Assume that we have isolated the r real roots of f and we know the values and the signs of f over the endpoints of the isolating intervals. Then, we can compute the sign of g evaluated over all r real roots of f in $\widetilde{\mathcal{O}}_B(p^2 q\tau + p^2\min\{p,q\}^2)$.*

Proof. Cor. 2.9 states that all isolating points together have bit-size $\mathcal{O}(p^2+p\tau_f)$. Let s_j be the bit-size of the j-th endpoint, where $0 \le j \le r \le p$. The evaluation of the remainder sequence over this endpoint, from cor 2.5, has complexity

$$\widetilde{\mathcal{O}}_B(pq\tau + \min\{p,q\}^2 s_j) \tag{1}$$

In order to compute the overall cost, we should sum over all the isolating points, i.e

$$\widetilde{\mathcal{O}}_B(pq\sum_j \tau + \min\{p,q\}^2 \sum_j s_{j_2}),$$

Since the number of roots is r the first summand is $\widetilde{\mathcal{O}}_B(rpq\tau)$. Using Cor. 2.9 the second summand becomes $\widetilde{\mathcal{O}}_B(p^2\min\{p,q\}^2 + p\min\{p,q\}^2\tau_f)$. Hence, the overall complexity is $\widetilde{\mathcal{O}}_B(rpq\tau + p^2\min\{p,q\}^2 + p\min\{p,q\}^2\tau_f)$. $\square$

The above proof corresponds to an algorithm that evaluates the quotient boot over each algebraic number, which reduces to evaluating the quotient boot over all isolating points.

An alternative would be to evaluate each remainder over the entire set of rational numbers and then combine these values in order to find the sign variations per algebraic number. This can be efficiently done by means of fan-in/fan-out, which allows for the fast evaluation of a single polynomial over a set of rational numbers. To simplify the discussion we focus on the only relevant case, namely $p \le q$.

Lemma 3.2 *Let $p \le q \in \mathbb{N}^*$. Take $g_i \in \mathbb{Z}[x]$, where i takes $\le p$ values, $\mathsf{deg}(g_i) = q_i \le p$, and $\mathcal{L}(g_i) \le \tau_g$. Consider $r \le p$ rational numbers of aggregate bit-size $S = O(p^2 + p\tau_f)$. Then, we can apply fan-in/fan-out to determine the sign of all g_i over all r numbers in $\widetilde{\mathcal{O}}_B(p^2\tau_g + p^3 S)$.*

Proof. The fan-in process is dominated by that of fan-out. The number of points to evaluate is $k = r$, the bit-size of the polynomial to evaluate is $h = \tau_g$ and its degree is $d \leq p$.

We apply Lem. 5.6 and 5.7. If $q_i \geq r$, then the complexity is $\widetilde{\mathcal{O}}_B(r\tau_g + rS(q_i - r) + r^2 S)$, otherwise, it is $\widetilde{\mathcal{O}}_B(r\tau_g + rSq_i)$. When we sum over the $O(p)$ values of i, we obtain $O(pr\tau_g)$ from the first term of each bound. The other terms are all bounded by $O(rSp)$, hence they sum up to $O(p^2 rS)$. Overall, the complexity is $\widetilde{\mathcal{O}}_B(pr\tau_g + p^2 rS)$. $\square$

4 Multivariate polynomials

We now prove some results concerning the computations with signed polynomial remainder sequence with multivariate polynomials. For this we use binary segmentation [19].

Let $f, g \in (\mathbb{Z}[y_1, \ldots, y_k])[x]$ such that $\mathsf{deg}_x(f) = p \geq q = \mathsf{deg}_x(g)$, $\mathsf{deg}_{y_i}(f) \leq d_i$ and $\mathsf{deg}_{y_i}(g) \leq d_i$. Moreover, let $d = \prod_{i=1}^{k} d_i$ and $\mathcal{L}(f), \mathcal{L}(g) \leq \tau$. In the rest of section, $\mathbf{SR}(f, g)$ implies the signed polynomial remainder sequence of f and g with respect to x.

The y_i-degree of the resultant of f and g, and thus of every polynomial in the sequence $\mathbf{SR}(f, g)$, is bounded by $\mathsf{deg}_{y_i}(\mathtt{res}(f, g)) \leq (p+q)d_i$, thus the ring homomorphism $\psi : \mathbb{Z}[y_1, \ldots, y_k] \to \mathbb{Z}[y]$ such that

$$\begin{aligned} y_1 &\mapsto y, \\ y_2 &\mapsto y^{(p+q)d_1}, \\ y_3 &\mapsto y^{(p+q)^2 d_1 d_2}, \\ &\vdots \\ y_k &\mapsto y^{(p+q)^{k-1} d_1 \cdots d_{k-1}} \end{aligned}$$

allows us to decode $\mathtt{res}(\psi(f), \psi(g)) = \psi(\mathtt{res}(f, g))$ in order to obtain $\mathtt{res}(f, g)$. The same holds for every polynomial in $\mathbf{SR}(f, g)$.

Polynomials $\psi(f), \psi(g) \in (\mathbb{Z}[y])[x]$ have $y-$degree lower than $d = (p + q)^{k-1} d_1 \cdots d_k$. This is so, since in worst case f or g may contain

$$y_1^{d_1}\, y_2^{d_2} \ldots y_k^{d_k},$$

thus $\psi(f)$ or $\psi(g)$ contain monomial

$$y^{d_1}\, y^{(p+q)d_1 d_2}\, y^{(p+q)^2 d_1 d_2 d_3} \ldots y^{(p+q)^{k-1} d_1 d_2 \ldots d_k}.$$

Thus, the degree in y is bounded by

$$\left(1 + (p+q) + (p+q)^2 + \cdots (p+q)^{k-1}\right) d_1 d_2 \ldots d_k.$$

Notice that

$$\sum_{i=0}^{k-1} D^i = \frac{D^k - 1}{D - 1} \leq \frac{D^{k-1} - \frac{1}{D}}{1 - \frac{1}{D}} \leq \frac{D^{k-1}}{1 - \frac{1}{D}} \leq \frac{4}{3} D^{k-1},$$

thus the $y-$degree is bounded by $(p+q)^{k-1} d_1 d_2 \cdots d_k = d(p+q)^{k-1}$ and that of their resultant is $\mathsf{deg}_y(\mathtt{res}(\psi(f), \psi(g))) < (p+q)d = (p+q)^k d_1 \cdots d_k$.

Theorem 4.1 *The computation of* $\mathbf{SR}(f,g)$ *has complexity* $\widetilde{\mathcal{O}}_B(q(p+q)^{k+2} d_1 d_2 \cdots d_k\, \tau)$.

Proof. Each polynomial in the $\mathbf{SR}(f,g)$ has coefficients of bit-size bounded $2^{c(p+q)\tau}$, for a suitable constant c and by assuming that $\tau > \lg(d)$. We consider the map $\chi : \mathbb{Z}[y] \mapsto \mathbb{Z}$, such that

$$y \mapsto 2^{\lceil c(p+q)\tau \rceil}.$$

Let $\phi = \psi \circ \chi : \mathbb{Z}[y_1, y_2 \ldots, y_k] \to \mathbb{Z}$. Then $\mathcal{L}(\phi(f)), \mathcal{L}(\phi(g)) \leq c\,(p+q)^k\, d$. Using Th. 2.2 we conclude that the complexity is $\widetilde{\mathcal{O}}_B(q(p+q)^{k+2} d\tau)$. □

Theorem 4.2 *[19] The computation of* $\mathbf{SRQ}(f,g)$ *any polynomial in* $\mathbf{SR}(f,g)$, $\mathtt{res}(f,g)$ *have complexity* $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1} d_1 d_2 \cdots d_k\, \tau)$.

Proof. We apply the same map as in the proof of the previous theorem and we use Th. 2.3. □

Theorem 4.3 *The evaluation of* $\mathbf{SR}(f,g)$ *over a rational number* a*, i.e* $\mathbf{SR}(f, g\,;\,\mathsf{a})$*, where* $\mathcal{L}(a) = \sigma$ *has complexity* $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1} d \max\{\tau, \sigma\})$.

Proof. First we compute the sequence $\mathbf{SRQ}(f,g)$ in $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1} d_1 d_2 \cdots d_k\, \tau)$ (Th. 4.2). Now we need to evaluate $\mathbf{SRQ}(f,g)$ on a. We shall use binary segmentation but, first, we bound the bit-size of the resulting polynomials.

Notice that the polynomials in $\mathbf{SR}_j(f,g)$ have total degree in $y_1, y_2, \ldots, y_k$ bound by $(p+q)\sum_{i=1}^{k} d_i$ and coefficient bit-size bounded by $(p+q)\tau$. Since with respect to x the polynomials in $\mathbf{SR}(f,g)$ have degree bounded by $\mathcal{O}(p)$ substitution $x = \mathsf{a}$ will yield numbers of bit-size $\mathcal{O}(p\sigma)$. Thus after the evaluation we obtain polynomials in $\mathbb{Z}[y_1, \ldots, y_k]$ with coefficient bit-size bounded by $\max\{(p+q)\tau, p\sigma\} \leq (p+q)\max\{\tau, \sigma\}$.

Consider map $\chi : \mathbb{Z}[y] \to \mathbb{Z}$, such that $y \mapsto 2^{\lceil c(p+q)\max\{\tau,\sigma\} \rceil}$, for a suitable constant c. We apply the map $\phi = \psi \circ \chi$ to f and g. Notice that $\mathcal{L}(\phi(f)), \mathcal{L}(\phi(g)) \leq cd(p+q)^k \max\{\tau, \sigma\}$. Using Thm. 2.4 we conclude that the evaluation costs $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1} d \max\{\tau, \sigma\})$. □

Using the previous theorems we can obtain the following corollaries for the bivariate case. Let $f, g \in (\mathbb{Z}[y])[x]$, such that $\mathsf{deg}_x(f) = p \geq q = \mathsf{deg}_x(g)$, $\mathsf{deg}_y(f), \mathsf{deg}_y(g) \leq d$ and $\mathcal{L}(f), \mathcal{L}(g) \leq \tau$.

Corollary 4.4 *We can compute* $\mathbf{SR}(f,g)$ *in* $\widetilde{\mathcal{O}}_B(\max\{p,q\}^3 qd\tau)$. *For any polynomial in the sequence, denoted* $\mathbf{SR}_j(f,g)$, *it holds that* $\mathsf{deg}_x(\mathbf{SR}_j(f,g)) = \mathcal{O}(p)$, $\mathsf{deg}_y(\mathbf{SR}_j(f,g)) \le \max\{p,q\}d$ *and* $\mathcal{L}\left(\mathbf{SR}_j(f,g)\right) = \mathcal{O}(p\tau)$.

Corollary 4.5 *We can compute* $\mathbf{SRQ}(f,g)$, *any polynomial in* $\mathbf{SR}(f,g)$, *and* $\mathtt{res}(f,g)$ *in* $\widetilde{\mathcal{O}}_B(q\max\{p,q\}^2 d\tau)$.

Corollary 4.6 *We can compute* $\mathbf{SR}(f,g\,;\,\mathsf{a})$, *where* $\mathcal{L}\left(\mathsf{a}\right) = \sigma$, *in* $\widetilde{\mathcal{O}}_B(q\max\{p,q\}^2 d\max\{\tau,\sigma\})$. *For the polynomials in the evaluated sequence, denoted by* $\mathbf{SR}_j(f,g\,;\,\mathsf{a}) \in \mathbb{Z}[y]$, *it holds that* $\mathsf{deg}_y(\mathbf{SR}_j(f,g\,;\,\mathsf{a})) \le \max\{p,q\}d$, *and* $\mathcal{L}\left(\mathbf{SR}_j(f,g\,;\,\mathsf{a})\right) = \mathcal{O}(\max\{p,q\}\max\{\tau,\sigma\})$.

4.1 Bivariate sign evaluation

This section focuses on computing the sign in $\{-1,0,1\}$ of a bivariate polynomial over one or more points in $\mathbb{Q}^2$.

Algorithm 1: SIGN_AT(F,α,β)

```
  Input: $F \in \mathbb{Z}[x,y], \alpha \cong (A, I_1 = [\mathsf{a}_1,\mathsf{a}_2]), \beta \cong (B, I_2 = [\mathsf{b}_1,\mathsf{b}_2])$
  Output: $\mathrm{sign}(F(\alpha,\beta))$
1 compute $\mathbf{SRQ}_x(A,F)$
2 $V_1 \leftarrow \emptyset$
3 $L_1 \leftarrow \mathbf{SR}_x(A,F\,;\,\mathsf{a}_1)$
4 foreach $f \in L_1$ do $V_1 \leftarrow$ ADD$(V_1,$ SIGN_AT$(f,\beta))$
5 $V_2 \leftarrow \emptyset$
6 $L_2 \leftarrow SR_x(A,F\,;\,\mathsf{a}_2)$
7 foreach $f \in L_2$ do $V_2 \leftarrow$ ADD$(V_2,$ SIGN_AT$(f,\beta))$
8 RETURN (VAR$(V_1)$ − VAR$(V_2)$) $\cdot\, \mathrm{sign}(A'(\alpha))$
```

We can compute the sign of a bivariate integer polynomial evaluated over two real algebraic numbers with an algorithm similar to the univariate case. Let $F \in \mathbb{Z}[x,y]$, $\alpha \cong (A(x), I_1)$ and $\beta \cong (B(X), I_2)$, where $I_1 = [\mathsf{a}_1,\mathsf{a}_2]$, $I_2 = [\mathsf{b}_1,\mathsf{b}_2]$. We wish to compute the sign of $F(\alpha,\beta)$. We denote the algorithm for this computation by SIGN_AT(F,α,β).

We consider polynomial $F \in (\mathbb{Z}[y])[x])$ and wish to compute its sign when evaluated over α. This means (see [11, 27]) that we have to obtain the sign variations of $\mathbf{SR}(A,F\,;\,\mathsf{a}_1)$, respectively $\mathbf{SR}(A,F\,;\,\mathsf{a}_2)$ and subtract them. After we evaluate the sequence on a_1, resp. a_2, we obtain polynomials in $\mathbb{Z}[y]$. In order to obtain the sign variations, we must compute the sign of these polynomials over β. The pseudo-code appears as Alg. 1.

Polynomial A is univariate and we assume that we already know its values on $\mathsf{a}_1, \mathsf{a}_2$, from the real root isolation process. Similarly, we know the values of B over $\mathsf{b}_1, \mathsf{b}_2$.

Theorem 4.7 (Bivariate sign_at) *Let $F \in \mathbb{Z}[x,y]$ such that $\mathsf{deg}_x(F) = \mathsf{deg}_y(F) = n_1$ and $\mathcal{L}(F) = \sigma$ and two real algebraic numbers $\alpha \cong (A, I_\alpha) = [\mathsf{a}_1, \mathsf{a}_2]$, $\beta \cong (B, I_\beta) = [\mathsf{b}_1, \mathsf{b}_2]$ where $A, B \in \mathbb{Z}[X]$, $\mathsf{deg}(A) = \mathsf{deg}(B) = n_2$, $\mathcal{L}(A) = \mathcal{L}(B) = \sigma$ and $I_\alpha, I_\beta \in \mathbb{Q}^2$. Then, one computes the sign of F evaluated over α and β, i.e* SIGN_AT(F, α, β)*, using Alg. 1 with complexity $\widetilde{\mathcal{O}}_B(n_1^3\, n_2^3\, \sigma)$, assuming that $n_1 \leq n_2$.*

Proof. First, we compute $\mathbf{SRQ}_x(A, F)$ in order to help us evaluate the sequence $\mathbf{SR}(A, F)$ on the endpoints of I_1. The complexity shall be dominated by that of evaluating the sequence.

We evaluate $\mathbf{SR}(A, F)$ on a_1 (Line 3 in Alg. 1). The first polynomial in the sequence is A, but there is no need to consider it, since we already know its value on a_1. This computation costs $\widetilde{\mathcal{O}}_B(n_1^2\, n_2^3\, \sigma)$ by applying Cor. 4.6 with $q = n_1$, $p = n_2$, $d = n_1$, $\tau = \sigma$, and $\sigma = n_2\sigma$, where the latter corresponds to the bit-size of the endpoints.

After the evaluation we obtain a list L_1, which contains $\mathcal{O}(n_1)$ polynomials in $\mathbb{Z}[y]$, of degrees bounded by $\mathcal{O}(n_1 n_2)$ and coefficient bit-size bounded by $\mathcal{O}(n_1 n_2 \sigma)$. To see why this is the coefficient bit-size, notice that the polynomials in the sequence are of degrees $\mathcal{O}(n_1)$ with respect to x and of coefficient bit-size $\mathcal{O}(n_2\sigma)$. Thus, after we evaluate on a_1 the coefficient bit-size becomes $\mathcal{O}(n_1 n_2 \sigma)$.

For each polynomial in L_1 we compute its sign over β (Line 4 in Alg. 1) and count the sign variations. Each sign determination costs $\widetilde{\mathcal{O}}_B(n_1^2 n_2^3 \sigma + n_2^3 \sigma) = \widetilde{\mathcal{O}}_B(n_1^2 n_2^3 \sigma)$. To obtain this, apply Cor. 2.7 with $p = n_2$, $q = n_1 n_2$, $\tau_f = \sigma$, $\sigma = n_1 n_2 \sigma$. Thus all the sign evaluations cost $\widetilde{\mathcal{O}}_B(n_1^3 n_2^3 \sigma)$.

We do the same for the other endpoint of I_1 and we subtract the sign variations. Notice that $\mathrm{sign}(A'(\alpha)) = \mathrm{sign}(A(\mathsf{b}_1) - A(\mathsf{a}_1))$, which would cost $\widetilde{\mathcal{O}}_B(n_2^3 \tau)$ to obtain; but it is known in advance from the real root isolation process that constructed α.

We conclude that the overall complexity of the algorithm is $\widetilde{\mathcal{O}}_B(n_1^3\, n_2^3\, \sigma)$. □

We extend the previous theorem to evaluating a bivariate polynomial over a sequence of algebraic numbers defined by the same polynomial. This constitutes our first approach to evaluating a bivariate polynomial over a sequence of pairs of algebraic numbers.

Corollary 4.8 *Consider the hypotheses of Thm. 4.7 and the problem of determining the signs of $F(\alpha, \beta_0), \ldots, F(\alpha, \beta_{r-1})$, where $\beta_i \cong (B, I_i) = [\mathsf{b}_{i1}, \mathsf{b}_{i2}]$ and the number of real roots of B is $r \leq n_2$. Then, the bit complexity for determining these r signs is $\widetilde{\mathcal{O}}_B(n_1^3 n_2^4 \sigma)$, by direct application of Thm. 4.7.*

A second approach to determining the signs of $F(\alpha, \beta_0), \ldots, F(\alpha, \beta_{r-1})$, where β_i are all real roots of $B \in \mathbb{Z}[x]$, can be based upon Lem. 3.1. But the most efficient approach is based on fan-in/fan-out:

Theorem 4.9 *Consider determining the signs of $F(\alpha, \beta_0), \ldots, F(\alpha, \beta_{r-1})$, where $\beta_i \cong (B, I_i) = [\mathsf{b}_{i1}, \mathsf{b}_{i2}]$ and $\sum_i \mathcal{L}(b_i) = S = \widetilde{\mathcal{O}}_B(n_2\sigma)$. In addition, the number of real roots of B is $r \leq n_2$. Then, the bit complexity is $\widetilde{\mathcal{O}}_B(n_1^2 n_2^3 \sigma + n_1 n_2^4 \sigma)$, by using fan-in/fan-out through Lem. 3.2.*

Proof. We follow Alg. 1 supporting Thm. 4.7. Computing the sequence $\mathbf{SRQ}(f, g)$ and the values $\mathbf{SR}(f, g, a_1)$ have complexity $\widetilde{\mathcal{O}}_B(n_1^2 n_2^3 \sigma)$, hence they are dominated.

The main difference from the proof of Thm. 4.7 is how to evaluate the list L_1 of $\mathcal{O}(n_1)$ polynomials in $\mathbb{Z}[y]$, of degrees bounded by $\mathcal{O}(n_1 n_2)$ and coefficient bit-size bounded by $\mathcal{O}(n_1 n_2 \sigma)$. For each polynomial in L_1 we compute its signs when evaluated over all β_i by using Lem. 3.2. The aggregate bit-size of the isolating points of the β_i is $S = \widetilde{\mathcal{O}}_B(n_2\sigma)$. In the lemma's statement, we also set $r \leq p = \min\{p, q\} = n_2, q \leq n_1 n_2$, $\tau_f = \sigma, \tau_g = \tau = n_1 n_2 \sigma$, and obtain $\widetilde{\mathcal{O}}_B(n_1 n_2^3 \sigma + n_2^4 \sigma) = \widetilde{\mathcal{O}}_B((n_1 + n_2) n_2^3 \sigma)$.

The rest of the steps are as in the analysis supporting Thm. 4.7. □

5 Algorithms for real solving

This section presents two algorithms for the real solving of bivariate systems. Consider the following system:

$$\begin{cases} F = \sum_{1 \leq i \leq d} \sum_{1 \leq j \leq d} a_{i,j} x^i y^j = 0 \\ G = \sum_{1 \leq i \leq d} \sum_{1 \leq j \leq d} d_{i,j} x^i y^j = 0 \end{cases} \tag{2}$$

We assume that $F, G \in \mathbb{Z}[x, y]$ are relatively prime and that $\mathsf{deg}(F) = \mathsf{deg}(G) = n$ and $\mathcal{L}(F) = \mathcal{L}(G) = \sigma$. The main idea behind our algorithms is to project the roots on the x and y axes, to compute the coordinates of the real solutions and somehow to match them. The difference between the algorithms is the way they match the solutions. To project we use resultants and signed polynomial remainder sequences. The output of the algorithms is a list with pairs of real algebraic numbers and, if possible, the multiplicities of the solutions.

5.1 The mrur_solve algorithm

The input of the algorithm are the two integer polynomials of the system (2). Notice that MRUR_SOLVE assumes that the polynomials are in generic position i.e no two distinct roots have the same x or y-coordinate. This assumption is without loss of generality since we can always put the system in such position by applying a shear transformation $(X, Y) \mapsto (X + aY, Y)$, where a is either a random number or computed deterministically [12, 22].

The algorithm is called MRUR_SOLVE, after modified RUR. We represent the ordinate of a solution in a rational univariate representation in terms of

Algorithm 2: MRUR_SOLVE (F, G)

```
Input: $F, G \in \mathbb{Z}[X, Y]$ such that they are in generic position
Output: The real solutions of the system $F = G = 0$
1  $\mathbf{SR} \leftarrow \mathbf{SR}_y(F, G)$
   /* Projection on the $X$ axis */
2  $R_x \leftarrow \texttt{res}_y(F, G)$
3  $P_x, M_x \leftarrow$ CONSTRUCT$(R_x)$
   /* Projection on the $Y$ axis */
4  $R_y \leftarrow \texttt{res}_x(F, G)$
5  $P_y, M_y \leftarrow$ CONSTRUCT$(R_y)$
6  $I \leftarrow$ INTERMEDIATE_POINTS$(P_y)$
   /* Consider the principal subresultant coefficients and
      compute the roots of $R_x$ that make them vanish */
7  $K \leftarrow$ COMPUTE_K$(\mathbf{SR}, P_x)$
8  $Q \leftarrow \emptyset$
   /* Matching the solutions */
9  foreach $\alpha \in P_x$ do
10     $\beta \leftarrow$ FIND$(\alpha, K, P_y, I)$
11     $Q \leftarrow$ ADD$(Q, \{\alpha, \beta\})$
12 RETURN $Q$
```

the abscissa. The RUR (Rational Univariate Representation) technique can be generalized to polynomial systems of many variables [4, 20, 5, 21, 2], and it is a generalization of the primitive element computation [24] and first presented by Kronecker.

The algorithm is similar to [14, 12], which refers to the closely related problem of the topology of a plane real algebraic curve. A crucial step in their algorithm is the real solving of a bivariate polynomial system. However, notice that their algorithm stops at the rational univariate representation of the ordinates. In many cases this is sufficient, but the representation of the ordinates is implicit. If we want to perform computations with these numbers, e.g compare them, then this representation is not sufficient.

We choose an alternative way, so that the output of the algorithm to be a list of pairs of real algebraic numbers, by modifying the algorithm of [14, 12] and that is why we name the algorithm modified RUR.

Another important difference between MRUR_SOLVE and the algorithm of [14, 12] is that they represent the real algebraic numbers using Thom's encoding [6], while we choose the isolating interval representation. The complexity of their algorithm is $\widetilde{\mathcal{O}}_B(N^{16})$, where $N = \max\{n, \sigma\}$.

5.1.1 The analysis of mrur_solve

The pseudo-code of MRUR_SOLVE is presented in Alg. 2. Roughly speaking the algorithm goes like this: We project on the X axis (Lines 2 and 3) and on the Y axis (Lines 4 and 5). Next for each real solution on the X axis we implicitly express its ordinate and then we match it to one of the computed projections on the Y-axis.

We assume that $\mathsf{deg}(F) = \mathsf{deg}(G) = n$ and $\mathcal{L}(F) = \mathcal{L}(G) = \sigma$. First we compute the sequence $\mathbf{SR}(F, G)$ wrt Y (Line 1 in Alg. 2) with complexity $\widetilde{\mathcal{O}}_B(n^5\,\sigma)$ (Cor. 4.4).

Projection on the X axis (Lines 2 and 3 in Alg. 2)
In order to compute the projection on the x axis of system (2) we should eliminate Y. For this we use the resultant, $R_x = \mathtt{res}(F, G)$ wrt Y (Line 2 in Alg. 2). The computation of R_x (Line 2 in Alg. 2) has complexity $\widetilde{\mathcal{O}}_B(n^4\,\sigma)$.

Notice that $R_x \in \mathbb{Z}[X]$ and that $\mathsf{deg}(R_x) = \mathcal{O}(n^2)$ and $\mathcal{L}(R_x) = \mathcal{O}(n\,\sigma)$. Next we compute the real algebraic numbers that are the real root of R_x in isolating interval representation (Line 3 in Alg. 2) using algorithm CONSTRUCT in $\widetilde{\mathcal{O}}_B(n^{10}\,\sigma^2)$. The representation of the real algebraic numbers contains the square-free part of R_x, which has bit-size $\mathcal{O}(n + n\,\sigma) = \mathcal{O}(n\,\sigma)$. The endpoints of the isolating intervals are rational numbers with bit-size $\mathcal{O}(n^3\,\sigma)$. Let these algebraic numbers be

$$\alpha_1 < \alpha_2 < \cdots < \alpha_{m-1} < \alpha_m \tag{3}$$

where $m \leq 2\,n^2$ is the number of real roots of R_x. The algorithm CONSTRUCT returns a list, P_x, that contains (in increasing order) the real algebraic numbers that are root of R_x and a list, M_x, with their multiplicities.

Finally, if α is a real root of R_x, then its multiplicity as a root of R_x is also the multiplicity of the pair (α, β) as a real solution of the system, for some $\beta \in \mathbb{R}_{alg}$, due to the generic position assumption.

Projection on the Y axis (Lines 4, 5 and 6 in Alg. 2)
We do exactly the same computations for the projection on the y axis. The real roots of R_y are now contained in list L_y and their multiplicities in M_y.

Algorithm INTERMEDIATE_POINTS computes rational numbers between the real roots of R_y in $\widetilde{\mathcal{O}}_B(n^5\sigma)$, and the numbers have bit-size $\mathcal{O}(n^3\,\sigma)$. Let the intermediate points be q_j, then the real roots of R_y and q_j are in the following order

$$q_0 < \beta_1 < q_1 < \beta_2 < \cdots < \beta_{\ell-1} < q_{\ell-1} < \beta_\ell < q_\ell \tag{4}$$

where $\ell \leq 2\,n^2$ is the number of the real roots of R_y. Since the system is in generic position, every real root of R_x has a correspondence with some real root of R_y. This is why it holds $\ell \leq m \leq 2\,n^2$. The intermediate points are contained in the list I.

The sub-algorithm compute_k (Line 7 in Alg. 2)
For every real root of R_x we must compute an index k such the assumptions of the theorem are fulfilled. This is what sub-algorithm COMPUTE_K does; it also stores k in list K. The generic position assumptions guarantee that there is always such a unique index. We define recursively the following family of polynomials, $\Gamma_j(x)$:

$$\Phi_0(X) = \frac{\mathbf{sr}_0(X)}{\gcd(\mathbf{sr}_0(X), \mathbf{sr}_0'(X))}$$

$$\Phi_1(X) = \gcd(\Phi_0(X), \mathbf{sr}_1(X)) \qquad \Gamma_1 = \frac{\Phi_0(X)}{\Phi_1(X)}$$

$$\Phi_2(X) = \gcd(\Phi_1(X), \mathbf{sr}_2(X)) \qquad \Gamma_2 = \frac{\Phi_1(X)}{\Phi_2(X)}$$

$$\vdots$$

$$\Phi_{n-1}(X) = \gcd(\Phi_{n-2}(X), \mathbf{sr}_{n-1}(X)) \qquad \Gamma_{n-1} = \frac{\Phi_{n-2}(X)}{\Phi_{n-1}(X)}$$

Recall that $\mathbf{sr}_i = \mathbf{psc}_i \in \mathbb{Z}[X]$ is the principal subresultant coefficient of $\mathbf{SR}_j \in (\mathbb{Z}[X])[Y]$. Polynomial $\Phi_0(x)$ is the square-free part of $R_x = \mathbf{sr}_0 = \mathbf{psc}_0 \in \mathbb{Z}[X]$, which we have already computed in algorithm CONSTRUCT(R_x) (Line 3 in Alg. 2).

Let α be an element of the list P_x, or in other words a real root of Φ_0. By construction it holds that $\Phi_0(X) = \prod_j \Gamma_j(X)$ and $\gcd(\Gamma_j, \Gamma_i) = 1$ if $j \neq i$. Now α is a real root of at most one Γ_j. For j such that $\Gamma_j(\alpha) = 0$, it holds that $\mathbf{sr}_0(\alpha) = \mathbf{sr}_1(\alpha) = 0, \ldots, \mathbf{sr}_j(\alpha) = 0$ and $\mathbf{sr}_{j+1}(\alpha) \neq 0$. Thus the index for α is $k = j + 1$. Now, α is in isolating interval representation, thus for Γ_j such that

$\Gamma_j(\alpha) = 0$, it should change signs at the endpoints of the isolating interval, by Rolle's theorem.

Let us summarize sub-algorithm COMPUTE_K: It computes the factorization $\Phi_0 = \prod_j \Gamma_j$ of the square-free part of R_x. Next, for every real root of R_x it computes Γ_j that changes sign on the endpoints of the isolating interval of α, the index that corresponds to α is $k = j + 1$.

Let's now study the complexity of the algorithm. Since the Φ_0 is the square-free part of R_x it holds that $\mathsf{deg}(\Phi_0) = \mathcal{O}(n^2)$ and $\mathcal{L}(\Phi_0) = \mathcal{O}(n+n\,\sigma) = \mathcal{O}(n\,\sigma)$. The polynomials Γ_j are divisors of the Φ_0, thus $\sum_j \mathsf{deg}(\Gamma_j) = \mathcal{O}(n^2)$ and, from Mignotte's bound [17, 18], $\mathcal{L}(\Gamma_j) = \mathcal{O}(n^3\sigma)$. In order to compute the factorization $\Phi_0(X) = \prod_j \Gamma_j(X)$ we need $\mathcal{O}(n)$ gcd computations of polynomials of degree $\mathcal{O}(n^2)$ and bit-size $\mathcal{O}(n^3\sigma)$. The GCD computation costs $\widetilde{\mathcal{O}}_B(n^7\,\sigma)$ and thus the overall cost is $\widetilde{\mathcal{O}}_B(n^8\,\sigma)$.

In order for each α to find κ we should compute the sign of Γ_j over the endpoints of the isolating interval of α. The number of α's is $\mathcal{O}(n^2)$, thus this is also the number of the endpoints of the isolating intervals. Moreover, their aggregate bit-size is $\mathcal{O}(n^3\,\sigma)$. We evaluate all Γ_j over all endpoints with complexity $\widetilde{\mathcal{O}}_B(n^6\,\sigma)$, using fan-in/fan-out.

Thus the overall complexity of COMPUTE_K is $\widetilde{\mathcal{O}}_B(n^8\,\sigma)$.

Matching the solutions, and algorithm Find (Lines 9–11 in Alg. 2) We have already computed the coordinates of the real solutions and what it remains is to match them, by sub-algorithm FIND. It takes as input a real root of R_x, and computes the ordinate of the real solution of the system, say β.

Now we describe in detail the steps of the algorithm and their complexity. For some real root α of R_x we have already computed index k. Thus we represent the ordinate using RUR, i.e

$$A(\alpha) = -\frac{1}{k}\frac{\mathbf{sr}_{k,k-1}(\alpha)}{\mathbf{sr}_k(\alpha)} = \frac{A_1(\alpha)}{A_2(\alpha)}$$

The generic position assumption guarantees that there is a unique β_j, in P_y, such that $\beta_j = A(\alpha)$, where $1 \le j \le \ell$. In order to compute j we use intermediate points in the list I, see (4). For this, it also holds that

$$q_j < A(\alpha) = \frac{A_1(\alpha)}{A_2(\alpha)} < q_{j+1}.$$

Thus j indicates the position of $A(\alpha)$ in the list of (ordered) numbers $q_0 < \cdots < q_\ell$. This can be computed by binary search in $\mathcal{O}(\lg \ell) = \mathcal{O}(\lg n)$ comparisons of $A(\alpha)$ with the rationals q_j. This is equivalent to computing the sign of $B_j(X) = A_1(X) - q_j\,A_2(X)$ over α. Thus it suffices to execute algorithm SIGN_AT(B_j, α) as many as $\mathcal{O}(\lg n)$ times.

For the complexity of SIGN_AT(B_j, α), recall that α is defined by a polynomial of degree $\mathcal{O}(n^2)$, bit-size $\mathcal{O}(n\,\sigma)$ and that the bit-size of the endpoints of the isolating intervals is $\mathcal{O}(n^3\,\sigma)$. Moreover, $\mathcal{L}(q_j) = \mathcal{O}(n^3\sigma)$ and $\mathsf{deg}(A_1) =$

Algorithm 3: GRID_SOLVE(F, G)

```
Input: $F, G \in \mathbb{Z}[x, y]$
Output: The real solutions of $F = G = 0$
1  $R_x \leftarrow \mathtt{res}_y(F, G)$
2  $L_x, M_x \leftarrow$ CONSTRUCT$(R_x)$
3  $R_y \leftarrow \mathtt{res}_x(F, G)$
4  $L_y, M_y \leftarrow$ CONSTRUCT$(R_y)$
5  $Q \leftarrow \emptyset$
6  foreach $\alpha \in L_x$ do
7      foreach $\beta \in L_y$ do
8          if SIGN_AT$(F, \alpha, \beta) = 0 \wedge$ SIGN_AT$(G, \alpha, \beta) = 0$ then
9              $Q \leftarrow$ ADD$(Q, \{\alpha, \beta\})$
10 RETURN $Q$
```

$\mathsf{deg}(\mathbf{sr}_{k,k-1}) = \mathcal{O}(n^2)$, $\mathsf{deg}(A_2) = \mathsf{deg}(\mathbf{sr}_k) = \mathcal{O}(n^2)$, $\mathcal{L}(A_1) = \mathcal{O}(n\,\sigma)$, $\mathcal{L}(A_2) = \mathcal{O}(n\,\sigma)$. Thus $\mathsf{deg}(B_j) = \mathcal{O}(n^2)$ and $\mathcal{L}(B_j) = \mathcal{O}(n^3\,\sigma)$. We conclude that the algorithms SIGN_AT(P_j, α) and FIND have total complexity $\widetilde{\mathcal{O}}_B(n^7\sigma)$.

As for the overall complexity of the loop (Lines 9-11) notice that the number of α's is $\mathcal{O}(n^2)$. Thus the overall complexity is $\widetilde{\mathcal{O}}_B(n^9\sigma)$.

Theorem 5.1 (mrur_solve) *Let $F, G \in \mathbb{Z}[X, Y]$ such that they are in generic position, are relative prime, their total degrees are bounded by n and their bitsize by σ. Real solving the system $F = G = 0$ using* MRUR_SOLVE *has complexity $\widetilde{\mathcal{O}}_B(n^{10}\sigma^2)$.*

Proof. The projection phases of the algorithm have complexity $\widetilde{\mathcal{O}}_B(n^{10}\,\sigma^2)$. The factorization of the square-free part of R_x and the computation of the indices k have complexity $\widetilde{\mathcal{O}}_B(n^8\,\sigma)$. The loop (Line 9) has complexity $\widetilde{\mathcal{O}}_B(n^9\,\sigma)$ since it is executed $\mathcal{O}(n^2)$ times and each step has complexity $\widetilde{\mathcal{O}}_B(n^7\,\sigma)$. $\square$

5.2 The grid_solve algorithm

The algorithm GRID_SOLVE, the pseudo-code of which appears in Alg. 3, is straightforward (Nicola Wolpert has used a similar method in her PhD thesis). We compute the real algebraic numbers that correspond to the x and y coordinates of the real solutions, as real roots of the resultants $\mathtt{res}_x(F, G)$ and $\mathtt{res}_y(F, G)$. Then, we match them using SIGN_AT (Alg. 1), by testing all rectangles in this grid.

The input of the algorithm is the polynomials $F, G \in \mathbb{Z}[x, y]$ and its output is a list of pairs of real algebraic numbers represented in isolating interval representation. This list is represented by Q in Alg. 3. The algorithm also outputs rational axis-aligned rectangles, guaranteed to contain a single root of the system.

The algorithm is correct since the coordinates of all the real roots of the system are real roots of $\texttt{res}_x(F, G)$ or $\texttt{res}_y(F, G)$ and from the fact that a pair $(\alpha, \beta) \in \mathbb{R}^2_{alg}$ is real solution of the system iff $F(\alpha, \beta) = G(\alpha, \beta) = 0$.

Even though the algorithm is obvious, to the best of our knowledge this is the first time that its complexity is studied. The disadvantage of the algorithm is that exact implementation of SIGN_AT (Alg. 1) is high. However, its simplicity makes it attractive and arithmetic filtering can speed up its implementation. The algorithm also requires no genericity assumption on the input.

Last but not least, the algorithm allows the use of heuristics. In particular, we may exploit easily computed bounds on the number of roots, such as the Mixed Volume, which also reflects the sparseness of the equations. When we have identified as many real roots, the algorithm stops. More importantly, we can test whether there are any real roots with a given abscissa α, in order to eliminate columns in the grid which are empty of real roots. The following theorem provides the complexity of the algorithm.

Theorem 5.2 (grid_solve) *Let $F, G \in \mathbb{Z}[x, y]$ relative prime with total degree bounded by n and coefficient bit-size bounded by σ. Isolating all real roots of the system $F = G = 0$ using* GRID_SOLVE *has complexity $\widetilde{\mathcal{O}}_B(n^{13} + n^{12}\sigma)$, provided $\sigma = O(n^2)$.*

Proof. First we project the system on the x axis, by computing the resultant of F and G wrt y, i.e R_x (Line 1 in Alg. 3). The complexity of this step is $\widetilde{\mathcal{O}}_B(n^4\sigma)$. To see this use Cor. 4.5 with $p = q = d = n$ and $\tau = \sigma$. Notice that $\mathsf{deg}(R_x) = \mathcal{O}(n^2)$ and $\mathcal{L}(R_x) = \mathcal{O}(n\,\sigma)$.

We compute the real algebraic numbers that real roots of R_x, in isolating interval representation (Line 2 in Alg. 3) with complexity $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma^2)$ (Th. 2.6 with $p = n^2$ and $\tau = n\sigma$) and we store them in L_x. This complexity shall be dominated provided that $\sigma = O(n^2)$. We do the same for the y axis (Lines 3 and 4 in Alg. 3) and store the algebraic numbers in L_y.

The representation of the real algebraic numbers that we have computed contains the square-free part of R_x, or R_y. In both cases the bit-size of the polynomial is $\mathcal{O}(n^2 + n\,\sigma)$ [2, 11]. The isolating intervals have as endpoints rational numbers of bit size $\mathcal{O}(n^4 + n^3\,\sigma)$.

Let r_x, resp. r_y be the size of L_x, resp. L_y, i.e the number of real roots of the corresponding resultants. Both are bounded by $\mathcal{O}(n^2)$.

Now it remains to check the candidate root pairs by evaluating F, G on them.

Here is a first method: form all possible pairs of real algebraic numbers from L_x and L_y and check for every such pair if both F and G vanish (Line 8 in Alg. 3). This is achieved by SIGN_AT (Alg. 1). Each evaluation costs $\widetilde{\mathcal{O}}_B(n^{11} + n^{10}\sigma)$ by Th. 4.7, with $n_1 = n$, $n_2 = n^2$ and $\sigma = n^2 + n\sigma$. In the worst case we perform $r_x\,r_y = O(n^4)$ sign evaluations and the overall complexity becomes $\widetilde{\mathcal{O}}_B(n^{15} + n^{14}\sigma)$. This complexity dominates that of isolating each resultant's roots.

Another method uses fan-in/fan-out on the remainder sequences. For each $\alpha \in L_x$, the algorithm evaluates $F(\alpha, y)$, then $G(\alpha, y)$, over all $\beta_j \in L_y$, by Thm. 4.9. There are $O(n^2)$ operations, each with bit complexity $\widetilde{\mathcal{O}}_B(n^9(n^2 + n\sigma))$, hence the total cost is $\widetilde{\mathcal{O}}_B(n^{13} + n^{12}\sigma)$. This complexity dominates that of isolating the resultant's roots provided that $\sigma = O(n^2)$. □

We now examine the multiplicity of a root (α, β) of the system. Refer to [7, sec.8.7] for its definition as the exponent of factor $(\beta x - \alpha y)$ in the resultant of the (homogenized) polynomials. This holds under the assumption that the two polynomials do not share any common factor, which is implicitly checked by the previous phase of real root isolation. Our algorithm reduces to bivariate sign determination and does not require bivariate factorization. We shall use the resultant, since it allows for multiplicities to "project". More formally, the sum of multiplicities of all roots (α, β_j) equals the multiplicity of root $x = \alpha$ in the respective resultant polynomial. It is possible to change the coordinate frame so as to ensure that different roots project to different points on the x-axis. Here, we take a more "symbolic" approach.

Assume, for now, that $(\alpha, \beta) = (0, 0)$ and consider the resultant

$$R_t(x) = \mathtt{res}_y \left(f(x + ty, y), g(x + ty, y) \right),$$

where t is a new parameter that will ensure that the (horizontal) shear of the coordinates is sufficiently generic. Let $m(t)$ be the multiplicity of 0 as a root of $R_t(x)$. The intersection multiplicity of $(0, 0)$ is $\min_t\{m(t)\}$.

Clearly, $R_t(x) \in (\mathbb{Z}[t])[x]$, i.e its coefficients lie in $\mathbb{Z}[t]$. Moreover, $m(t)$ is equal to the number of trailing coefficients that vanish. The minimum such number counts the trailing coefficients that are identically zero as elements of $\mathbb{Z}[t]$, and this is easy to find once we have computed $R_t(x)$.

Now take new parameters ρ, ξ and consider the resultant

$$R_t(x) = \mathtt{res}_y \left(f(x - \rho + ty - t\xi, y - \xi), g(x - \rho + ty - t\xi, y - \xi) \right),$$

which lies in $(\mathbb{Z}[t, \rho, \xi])[x]$. The intersection multiplicity of some root $(\alpha, \beta) \in (\mathbb{R}_{alg}, \mathbb{R}_{alg})$ will be obtained as the number of trailing coefficients in $(\mathbb{Z}[\rho, \xi])[t]$, which vanish when $\rho = \alpha, \xi = \beta$.

Example 5.3 *Take the circle $f = (x - 1)^2 + y^2 - a$, and line $g = y$ with roots $(0, 0), (2, 0)$. We focus on $(0, 0)$ and assume that $(\alpha, \beta) = (0, 0)$. In the text we have described the transformation*

$$f(x + ty, y) = y^2(1 + t^2) + 2ty(x - 1) + (x^2 - 2x), g(x + ty, y) = y.$$

Computing $R_t(x) = x^2 - 2x$ will give 1 as the multiplicity of $(0, 0)$. Notice that $R_t(x)$ happens to be independent of t. More interesting is the projection on the y-axis by transforming

$$f(x, y + tx) = x^2(1 + t^2) + 2x(ty - 1) + y^2, g(x, y + tx) = y + tx.$$

Then, $R(y) = y^2 - 2ty$ which gives again 1 as the multiplicity of $(0, 0)$.

Theorem 5.4 *Consider the setting of Thm. 5.2. Having isolated all real roots of the system $f = g = 0$, it is possible to determine their multiplicities with complexity $\widetilde{\mathcal{O}}_B(n^{15}\sigma)$.*

Proof. We use binary segmentation to compute the resultant of two polynomials in $(\mathbb{Z}[x, t, \rho, \xi])[y]$. We apply Thm. 4.2 by setting $p = q = d_i = n, k = 4, \tau = \sigma$, thus arriving at $\widetilde{\mathcal{O}}_B(n^{10}\sigma)$.

The resultant coefficients with respect to x (we call them x-coefficients) are themselves polynomials in $(\mathbb{Z}[\rho, \xi])[t]$ of degree $O(n^2)$ in t. To test whether an x-coefficient vanishes, we must check its own t-coefficients, which lie in $\mathbb{Z}[\rho, \xi]$, until one does not vanish. If they all vanish, the x-coefficient is zero and the multiplicity of (α, β) is incremented by one. There will be $O(n^2)$ x-coefficients that may vanish in the course of the entire algorithm, because this is the sum of all multiplicities.

Let us denote by $T_i(\rho, \xi)$ the t-coefficient corresponding to t^i. Testing $T_i(\rho, \xi)$ can be implemented by a bivariate sign-at operation. The polynomial has degree $O(n^2)$ in ρ and in ξ, and bit-size $n\sigma$. By Thm. 4.9, the complexity is $\widetilde{\mathcal{O}}_B(n^{11}\sigma)$; in fact, this returns all signs $T_i(\alpha, \beta_0), \ldots, T_i(\alpha, \beta_r)$, for all roots β_j of the defining polynomial. Hence the cost is $\widetilde{\mathcal{O}}_B(n^{15}\sigma)$. $\square$

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

[2] S. Basu, R. Pollack, and M-F.Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics.* Springer-Verlag, 2003.

[3] D. Bini and V.Y. Pan. *Polynomial and Matrix Computations*, volume 1: Fundamental Algorithms. Birkhäuser, Boston, 1994.

[4] J. Canny. *The Complexity of Robot Motion Planning.* ACM – MIT Press Doctoral Dissertation Award Series. MIT Press, Cambridge, MA, 1987.

[5] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proc. ACM Symp. Theory of Computing*, pages 460–467, 1988.

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

[7] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, 2nd edition, 1997.

[8] D. Diochnos, I. Z. Emiris, and E. P. Tsigaridas. On the Complexity of real solving bivariate systems. Manuscript, 2007.

[9] Z. Du, V. Sharma, and C. K. Yap. Amortized bound for root isolation via Sturm sequences. In D. Wang and L. Zhi, editors, *Intern. Workshop on Symbolic Numeric Computing*, School of Science, Beihang University, Beijing, China, 2005.

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

[11] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real Algebraic Numbers: Complexity Analysis and Experimentation. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, editors, *Reliable Implementations of Real Number Algorithms: Theory and Practice*, LNCS (to appear). Springer Verlag, 2006. also available in www.inria.fr/rrrt/rr-5897.html.

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

[13] L. González-Vega, H. Lombardi, T. Recio, and M-F. Roy. Sturm-Habicht Sequence. In *ISSAC*, pages 136–146, 1989.

[14] L. Gonzalez-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Computer Aided Geometric Design*, 19(9):719–743, December 2002.

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

[16] H. Lombardi, M-F. Roy, and M. Safey El Din. New Structure Theorem for Subresultants. *J. Symb. Comput.*, 29(4-5):663–689, 2000.

[17] M. Mignotte. *Mathematics for computer algebra.* Springer-Verlag, New York, 1991.

[18] M. Mignotte and D. Stefanescu. *Polynomials: An algorithmic approach.* Springer, 1999.

[19] D. Reischert. Asymptotically fast computation of subresultants. In *ISSAC*, pages 233–240, 1997.

[20] J. Renegar. On the worst-case arithmetic complexity of approximating zeros of systems of polynomials. *SIAM J. Computing*, 18:350–370, 1989.

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

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

[23] E. P. Tsigaridas and I. Z. Emiris. Univariate polynomial real root isolation: Continued fractions revisited. In Y. Azar and T. Erlebach, editors, *In Proc. 14th European Symposium of Algorithms (ESA)*, volume 4168 of *LNCS*, pages 817–828, Zurich, Switzerland, 2006. Springer Verlag.

[24] B. van der Waerden. *Modern Algebra.* Ungar, 1953. Volumes 1-2.

[25] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge Univ. Press, Cambridge, U.K., 2nd edition, 2003.

[26] J. von zur Gathen and T. Lücking. Subresultants revisited. *Theor. Comput. Sci.*, 1-3(297):199–239, 2003.

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

Appendix

Multipoint evaluation will be based on the fan-in and fan-out method, well-known from the multiple evaluation of univariate polynomials [3, 1]. The main idea is that evaluation at point b_i is equivalent to computing the remainder $\mathrm{mod}(x - b_i)$.

We are given k points $b_0, \ldots, b_{k-1}$, such that their bit-size is bounded by $\mathcal{L}(b_i) = \sigma_i \leq \sigma$. Let us denote the *sum* of all bit-sizes $\sum_{i=0}^{k-1} \sigma_i$ by $S \leq k\sigma$.

The fan-in algorithm constructs bottom-up a binary tree with $\lceil \lg k \rceil + 1$ levels. Wlog, let us assume k is a power of 2, i.e $k = 2^{\mu}$.

At level $\ell = 0$, the leaves of the tree, are the polynomials $m_{0,j}(x) = x - b_j$, where $0 \leq j < k - 1$. The root polynomial corresponds to level $\ell = \lg k = \mu$ and equals $m_{\mu,0}(x) = \prod_{j=0}^{k-1}(x - b_j)$. Hence, $\mathcal{L}(m_{\mu,0}) = S$.

At level ℓ, where $0 \leq \ell \leq \lg k = \mu$, there are $k/2^{\ell} = 2^{\mu-\ell}$ polynomials, which we denote by $m_{\ell,j}$, where $0 \leq j < 2^{\mu-\ell}$. Such a polynomial is the product of two polynomials at level $\ell - 1$, namely

$$m_{\ell,j}(x) = m_{\ell-1,2j}(x) \cdot m_{\ell-1,2j+1}(x),\ j = 0, \ldots, 2^{\mu-\ell} - 1.$$

Moreover, $\mathsf{deg}(m_{\ell,j}) = 2^{\ell}$ and

$$\mathcal{L}(m_{\ell,j}) = \sum_{i=j2^{\ell}}^{(j+1)2^{\ell}-1} \sigma_i.$$

Lemma 5.5 (Fan-in) *Given are points $b_0, \ldots, b_{k-1} \in \mathbb{Q}$ of bit-size σ_i respectively, where $\sum_{i=0}^{k-1} \sigma_i = S$. Then, the fan-in phase has complexity in $\widetilde{\mathcal{O}}_B(kS)$.*

Proof. The cost at each level is

$$\sum_{j=0}^{2^{\mu-\ell}-1} \mathsf{M}\left(2^\ell, \sum_{i=j2^\ell}^{(j+1)2^\ell-1} \sigma_i\right) = \widetilde{\mathcal{O}}_B(2^\ell S)$$

and the total cost is

$$\sum_{\ell=0}^{\mu} \widetilde{\mathcal{O}}_B(2^\ell S) = \widetilde{\mathcal{O}}_B(kS)$$

using the multi-linearity of the complexity of multiplication. □

Let g be a polynomial that we wish to evaluate on all points b_j; assume $\mathcal{L}(g) = h$. The fan-out phase uses the polynomials produced at fan-in, in top-down fashion, so as to compute, at the end, all remainders $g(x) \bmod b_j$. Recall that the fan-in tree has $\mu + 1 = \lg k + 1$ levels. The main idea is to use the following rule:

$$g(x) \bmod m_{\ell,j}(x) = [g(x) \bmod m_{\ell,j} \cdot M(x)] \bmod m_{\ell,j}(x), \qquad (5)$$

where $M(x)$ is the polynomial corresponding to that sibling of the node containing $m_{\ell,j}(x)$, which multiplies the latter in order to define their common parent. More specifically, $M(x)$ is either $m_{\ell,j+1}(x)$ or $m_{\ell,j-1}(x)$. Hence, the remainder inside the square brackets in (5) is computed at the parent node, because its divisor is $m_{\ell+1,\lfloor j/2 \rfloor}(x)$.

Lemma 5.6 (Fan-out, case 1) *Consider the setting of Lem. 5.5 and let $g(x)$ have degree $d \geq k$ and bit-size h. Each b_i, $i = 0, \ldots, k-1$, has size bounded by σ_i, respectively. If $\sum_{i=0}^{k-1} \sigma_i = S$, then the complexity is $\widetilde{\mathcal{O}}_B(kh + kS(d-k) + k^2 S)$.*

Proof. The first step of the fan-out phase is to compute $g(x) \bmod m_{0,\mu}(x)$ with complexity $\widetilde{\mathcal{O}}_B(d(h+S))$. The remainder has bit-size $O((d-k)S + h)$.

At the next level ($\ell = \mu - 1$) the degree drops at most $2^{\mu-1}$. Each of the two remainders has bit-size $2^{\mu-1} \cdot \sum_i \sigma_i + (d-k)S + h$, where the sum ranges over distinct sets of $k/2$ indices. For simplicity, let us focus on the first remainder, with bit-size $2^{\mu-1} \cdot \sum_{i=0}^{k/2-1} \sigma_i + (d-k)S + h$. The divisor's degree is $2^{\mu-1}$, hence the complexity of computing the 2 remainders of this level will be proportional (ignoring polylogarithmic factors) to that degree. Hence, the level's complexity is $\widetilde{\mathcal{O}}_B(k(d-k)S + kh + 4^\mu S)$.

At level $\ell = \mu - 2$, the first of the 4 remainders has bit-size

$$(d-k)S + h + 2^{\mu-2} \sum_{i=0}^{k/4-1} \sigma_i + 2^{\mu-1} \sum_{i=0}^{k/2-1} \sigma_i.$$

The divisor's degree is $2^{\mu-2}$, hence the complexity of computing the 4 remainders will be roughly proportional to it. Hence, the level's complexity is $\widetilde{\mathcal{O}}_B(k(d-k)S + kh + 4^{\mu-2}S + 4^{\mu-2}2S)$.

At level ℓ, the first remainder has bit-size

$$(d-k)S + h + 2^{\ell} \sum_{i=0}^{2^{\ell}-1} \sigma_i + \cdots + 2^{\mu-2} \sum_{i=0}^{k/4-1} \sigma_i + 2^{\mu-1} \sum_{i=0}^{k/2-1} \sigma_i.$$

The divisor's degree is $2^{\ell+1}$, hence the complexity of computing the $k/2^{\ell} = 2^{\mu-\ell}$ remainders of this level will be proportional to it. Hence, the level's complexity is

$$\frac{k}{2^{\ell}} 2^{\ell+1}[(d-k)S + h] + 4^{\ell}2S + 4^{\ell+1}2S + \cdots 4^{\mu-1}2S.$$

Let us explain the 2nd term above. Corresponding to the first remainder, the complexity is $2^{2\ell+1} \sum_{i=0}^{2^{\ell}-1} \sigma_i$. One must sum over all remainders, thus summing over all σ_i.

How about the 3rd term? For the first remainder the complexity is $2^{2\ell+2} \sum_{i=0}^{2^{\ell}} \sigma_i$. Now, summing over all remainders, we use all σ_i's twice, hence obtain $2S$. The other terms are deduced similarly.

Now, we sum for $\ell \geq 0$ up to $\ell < \mu = \lg k$. Imagine a "vertical" summation. The sum of first terms gives $\mu k[(d-k)S + h]$. The rest of terms yields

$$\sum_{\ell=0}^{\mu-1} (\ell+1)4^{\ell}2S = \frac{1}{2} \sum_{\ell=1}^{\mu} \ell 4^{\ell} S = O(\mu 4^{\mu} S) = O^*(k^2 S).$$

□

Lemma 5.7 (Fan-out, case 2) *Consider the setting of Lem. 5.6 and let $g(x)$ have degree $d < k$. Then, the fan-out phase has complexity $\widetilde{\mathcal{O}}_B(kh + kSd)$.*

Proof. If $d < k$ then the first levels give all remainders equal to $g(x)$. For maximum ℓ such that $2^{\ell} < d = \deg(g)$, the computation starts. Hence, at level $\ell \simeq \lg d$, all $k/2^{\ell}$ subtrees must be considered, each in a fan-out computation. Using the previous lemma, then each subtree costs $\widetilde{\mathcal{O}}_B(2^{\ell}h + 2^{2\ell}S)$. Multiplying this cost by $k/2^{\ell}$ yields the overall bound. □

In conclusion, the bit complexity of fan-in is always dominated by that of fan-out. Furthermore, this complexity cannot be asymptotically improved by use of modular arithmetic and the Chinese remainder theorem.