On the complexity of real solving bivariate systems

Dimitrios Diochnos, Ioannis Z. Emiris, Elias Tsigaridas

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On the complexity of real solving bivariate systems

Dimitrios I. Diochnos — Ioannis Z. Emiris — Elias P. Tsigaridas

N° 6116

January 2007

Thème SYM

ISSN 0249-6399

On the complexity of real solving bivariate systems

Dimitrios I. Diochnos* , Ioannis Z. Emiris† , Elias P. Tsigaridas‡ §

Thème SYM — Systèmes symboliques
Projet VEGAS

Rapport de recherche n° 6116 — January 2007 — 31 pages

Abstract: This paper is concerned with exact real solving of well-constrained, bivariate algebraic systems. The main problem is to isolate all common real roots in rational rectangles, and to determine their intersection multiplicities. We present three algorithms and analyze their asymptotic bit complexity, obtaining a bound of $\widetilde{\mathcal{O}}_B(N^{14})$ for the purely projection-based method, and $\widetilde{\mathcal{O}}_B(N^{12})$ for two subresultant-based methods: we ignore polylogarithmic factors, and N bounds the degree and the bitsize of the polynomials. The previous record bound was $\widetilde{\mathcal{O}}_B(N^{16})$.

Our main tool is signed subresultant sequences, extended to several variables by the technique of binary segmentation. We exploit recent advances on the complexity of univariate root isolation, and extend them to multipoint evaluation, to sign evaluation of bivariate polynomials over two algebraic numbers, and real root counting for polynomials over an extension field. Our algorithms apply to the problem of simultaneous inequalities; they also compute the topology of real plane algebraic curves in $\widetilde{\mathcal{O}}_B(N^{12})$, whereas the previous bound was $\widetilde{\mathcal{O}}_B(N^{16})$.

All algorithms have been implemented in MAPLE, in conjunction with numeric filtering. We compare them against GBRS and system solvers from SYNAPS; we also consider MAPLE libraries INSULATE and TOP, which compute curve topology. Our software is among the most robust, and its runtimes are comparable, or within a small constant factor, with respect to the C/C++ libraries.

Key-words: MAPLE code, real solving, polynomial systems, real algebraic numbers, complexity

This work is partially supported by the european project ACS (Algorithms for Complex Shapes, IST FET Open 006413) and ARC ARCADIA (http://www.loria.fr/∼petitjea/Arcadia/)

* Department of Informatics and Telecommunications National Kapodistrian University of Athens, HELLAS
† Department of Informatics and Telecommunications National Kapodistrian University of Athens, HELLAS
‡ LORIA-INRIA Lorraine, Nancy, FRANCE
§ The work on this project started while at INRIA Sophia-Antipolis.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

On the complexity of real solving bivariate systems

Résumé : On s'intéresse à la résolution exacte, dans les réels, de systèmes algébriques bien-contraints en deux variables. Le problème principal est d'isoler les racines communes en des rectangles rationaux, et de calculer leur multiplicités. Nous présentons trois algorithmes et nous analysons leur complexité asymptotique binaire, arrivant à une borne de $\widetilde{\mathcal{O}}_B(N^{14})$ pour la méthode de projection, et de $\widetilde{\mathcal{O}}_B(N^{12})$ pour les deux méthodes basées sur le résultant: ces bornes ignorent les facteurs poly-logarithmiques, où N borne le degré et la taille binaire des polynômes. La borne précedente était de $\widetilde{\mathcal{O}}_B(N^{16})$.

Notre outil principal sont les séquences de sous-résultants, généralisées en plusieurs variables. Nous nous basons sur les avancés récentes sur la complexité d'isolation univariée, et nous généralisons ces résultats pour borner la complexité de l' évaluation en plusieurs points, pour déterminer le signe de polynômes en deux variables, et pour compter le nombre de racines d'un polynôme en coefficients algébriques. Nos algorithmes s'appliquent au problème d'inégalités simultanées; ils calculent aussi la topologie d'une courbe algébrique dans le plan en $\widetilde{\mathcal{O}}_B(N^{12})$, tandis que la borne existante était de $\widetilde{\mathcal{O}}_B(N^{16})$.

Nos algorithmes ont tous été implementés en Maple, avec filtrage numérique. Nous les comparons avec GBRS, des solveurs de SYNAPS, ainsi que les programmes INSULATE et TOP en MAPLE, qui calculent la topologie d'une courbe. Notre logiciel est parmi les plus robustes et ses temps de calculs sont comparables, ou plus lents d'un petit facteur constant, par rapport aux logiciels C/C++.

Mots-clés : code Maple, résolution dans les réels, systèmes algébriques, nombres algébriques, complexité

1 Introduction

The problem of well-constrained 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 study different algorithms in practice. We expect to obtain faster algorithms than in the general case. This is important in several applications ranging from nonlinear computational geometry and computer-aided geometric design to real quantifier elimination and robotics. A question of independent interest is to compute the topology of a plane real algebraic curve, which is studied in this paper.

Our algorithms isolate all common real roots inside non-overlapping rational rectangles, and determine the intersection multiplicity per root. We present three projection-based algorithms and analyze their asymptotic bit complexity. This leads to a bound of $\widetilde{\mathcal{O}}_B(N^{12})$, when ignoring polylogarithmic factors, whereas the previous record bound was $\widetilde{\mathcal{O}}_B(N^{16})$ [11], derived from the closely related problem of computing the topology of real plane algebraic curves, where N bounds the degree and the bitsize of the input polynomials. The approach in [11] depends on Thom's encoding. We choose the isolating interval representation, since it is used in applications, and demonstrate that it supports very efficient algorithms. In [11] it is stated that "isolating intervals provide worst bounds" and that"isolating intervals together with seminumerical techniques provide much better computing times than symbolic methods". Moreover, it is widely believed that isolating intervals do not produce good theoretical results. Our work suggests that isolating intervals should be re-evaluated.

Our main tool is signed subresultant sequences, extended to several variables by the technique of binary segmentation. We exploit the recent breakthroughs on univariate root isolation, which reduced complexity by at least one order of magnitude to $\widetilde{\mathcal{O}}_B(N^6)$ [8, 10, 9]. This brought complexity closer to $\widetilde{\mathcal{O}}_B(N^4)$, which is achieved by numerical methods, e.g. [23]. Using the aggregate separation bound we derive new bounds for the sign evaluation of uni-, bi- and multi-variate polynomials over all real roots of a polynomial system.

Previous work includes the following. In [16] 2×2 systems are solved and the multiplicities computed under the assumption that a generic shear has been obtained. The latter is based on [27]. Wolpert [31] studied 2×2 systems of bounded degree, obtained as projections of the arrangement of 3D quadrics. Her algorithm is a precursor of ours, except that in her case the phase of matching and of multiplicity computation was simpler; for this, she introduces Jacobi curves. In [21] a subdivision algorithm is proposed, exploiting the properties of the Bernstein basis, with unknown bit complexity, and arithmetic complexity based on the characteristics of the graphs of the polynomials.

Determining the topology of a real algebraic plane curve is a closely related problem. The best asymptotic complexity bound is $\widetilde{\mathcal{O}}_B(N^{16})$, given in [11], using Thom's encoding and Sturm-Habicht sequences. In [32] three projections are used; this method is implemented in INSULATE, with which we make several comparisons. The recent work in [14] seems to be the state-of-the-art implementation of resultant-based methods. For an alternative using

Gröbner basis see [6]. To the best of our knowledge the only complexity result for the topology determination using isolating intervals, is [1], where a $\widetilde{\mathcal{O}}_B(N^{30})$ bound is proved.

We establish a bound of $\widetilde{\mathcal{O}}_B(N^{12})$. It seems that the complexity in [11] could be improved to $\widetilde{\mathcal{O}}_B(N^{10})$ using fast multiplication algorithms, fast algorithms for computations of signed subresultant sequences and improved bounds for the bitsize of the integers appearing in computations.

We have implemented in MAPLE a package for computations with real algebraic numbers and for implementing our algorithms for real solving. The implementation is easy to use and integrates seminumerical filtering to speed up computation when the roots are well-separated. It guarantees exactness and completeness of results; moreover, the runtimes seem very encouraging. We illustrate it by experiments against well-established C/C++ libraries GBRS and certain solvers in SYNAPS. We also examine MAPLE libraries INSULATE and TOP, which compute curve topology. Our software is found to be the most robust; its runtime is always comparable, or within a small constant factor, wrt the fastest C/C++ library.

The main contributions of this paper are the following: We improve the complexity bound for computing the sign of a polynomial evaluated over all real roots of another (lem. 2.7). We establish a complexity bound for bivariate sign evaluation (th. 3.7), that helps us derive bounds for the problem of simultaneous inequalities (cor. 5.1), and for root counting in an extension field (th. 3.8). We study the complexity of bivariate polynomial real solving, using three projection-based algorithms: a straightforward grid method (th. 4.1), a specialized RUR approach (th. 4.5), and an improvement of the latter using fast GCD (th. 4.6). Our best bound is $\widetilde{\mathcal{O}}_B(N^{12})$ for arbitrary systems; in this time, we also compute the root multiplicities. Using the previous results, we prove that computing the topology of a real plane algebraic curve is in $\widetilde{\mathcal{O}}_B(N^{12})$ (th. 5.2). Last but not least, we describe our publicly available MAPLE software.

To put our bounds into perspective, consider that the input is in $\widetilde{\mathcal{O}}_B(N^3)$, and the bitsize of all output isolation points for univariate solving is $\widetilde{\mathcal{O}}_B(N^2)$, and this is tight.

The next section presents basic results concerning real solving and operations on univariate polynomials, notably evaluation. Sec. 3 extends the discussion to several variables, and focuses on bivariate polynomials and real root counting over extension fields. The algorithms for bivariate solving and their analyses appear in sec. 4, which is followed by applications to simultaneous inequalities and the topology of real curves. Our implementation and experiments appear in sec. 6. Ancillary results and certain proofs appear in the Appendix.

2 Univariate polynomials

In this paper, $\mathcal{O}_B$ means bit complexity and $\widetilde{\mathcal{O}}_B$ means that we are ignoring polylogarithmic factors. For $f \in \mathbb{Z}[y_1, \ldots, y_k, x]$, $\mathsf{dg}(f)$ denotes its total degree, while $\deg_x(f)$ denotes its degree wrt x. $\mathcal{L}(f)$ bounds the bitsize of the coefficients of f (including a bit for the sign).

We assume $\lg(\mathsf{dg}(f)) = \mathcal{O}(\mathcal{L}(f))$. For $\mathsf{a} \in \mathbb{Q}$, $\mathcal{L}(\mathsf{a})$ is the maximum bitsize of numerator and denominator. Let $\mathsf{M}(\tau)$ denote the bit complexity of multiplying two integers of size τ, and $\mathsf{M}(d,\tau)$ the complexity of multiplying two univariate polynomials of degrees $\leq d$ and coefficient bit size $\leq \tau$. Using FFT, $\mathsf{M}(\tau) = \widetilde{\mathcal{O}}_B(\tau)$, $\mathsf{M}(d,\tau) = \widetilde{\mathcal{O}}_B(d\tau)$. We compute $f_1 \bmod f_2$ in $\widetilde{\mathcal{O}}_B(d(\delta\tau_2+\tau_1))$, where $d \geq \mathsf{dg}(f_i), \delta = \mathsf{dg}(f_1) - \mathsf{dg}(f_2), \mathcal{L}(f_i) = \tau_i$, see lem. A.1.

Let $f, g \in \mathbb{Z}[x]$, $\mathsf{dg}(f) = p \geq q = \mathsf{dg}(g)$ and $\mathcal{L}(f), \mathcal{L}(g) \leq \tau$. We use $\mathsf{rem}(f,g)$ and $\mathsf{quo}(f,g)$ for the Euclidean remainder and quotient, respectively. The *signed polynomial remainder sequence* of f, g is $R_0 = f$, $R_1 = g$, $R_2 = -\mathsf{rem}(f,g)$, $\ldots$, $R_k = -\mathsf{rem}(R_{k-2}, R_{k-1})$, where $\mathsf{rem}(R_{k-1}, R_k) = 0$. The *quotient sequence* contains the $Q_i = \mathsf{quo}(R_i, R_{i+1})$, and the *quotient boot* is $(Q_0, \ldots, Q_{k-1}, R_k)$.

Here, we consider signed subresultant sequences, which contain polynomials similar to the polynomials in the signed polynomial remainder sequence; see [30] for a unified approach to subresultants, but they achieve better bounds on the coefficient bitsize and have good specialization properties. In our implementation we use Sturm-Habicht sequences, e.g [12]. By $\mathbf{SR}(f,g)$ we denote the signed subresultant sequence, by $\mathbf{sr}(f,g)$ the sequence of the principal subresultant coefficients, and by $\mathbf{SR}(f,g;\mathsf{a})$ the evaluated sequence over $\mathsf{a} \in \mathbb{Q}$.

Proposition 2.1 *[17, 24, 18] Assuming $p \geq q$, $\mathbf{SR}(f,g)$ is computed in $\widetilde{\mathcal{O}}_B(p^2q\tau)$ and $\mathcal{L}(\mathbf{SR}_j(f,g)) = \mathcal{O}(p\tau)$. For any f, g, their quotient boot, any polynomial in $\mathbf{SR}(f,g)$, their resultant, and their* gcd *are computed in $\widetilde{\mathcal{O}}_B(pq\tau)$.*

Proposition 2.2 *[17, 24] Let $p \geq q$. We can compute $\mathbf{SR}(f,g;\mathsf{a})$, where $\mathsf{a} \in \mathbb{Q} \cup \{\pm\infty\}$ and $\mathcal{L}(\mathsf{a}) = \sigma$, in $\widetilde{\mathcal{O}}_B(pq\tau + q^2\sigma + p^2\sigma)$. If $f(\mathsf{a})$ is known, then the last term can be omitted.*

When $q > p$, $\mathbf{SR}(f,g)$ starts with $f, g, -f, -(g \bmod (-f)) \ldots$, thus $\mathbf{SR}(f,g;\mathsf{a})$ starts with a sign variation irrespective of $\mathrm{sign}(g(\mathsf{a}))$. Therefore, if only the sign variations are needed, there is no need to evaluate g, and pr. 2.2 yields $\widetilde{\mathcal{O}}_B(pq\tau + p^2\sigma)$.

Corollary 2.3 *For any f, g, $\mathbf{SR}(f,g;\mathsf{a})$ is computed in $\widetilde{\mathcal{O}}_B(pq\tau + \min\{p,q\}^2\sigma)$, provided* $\mathrm{sign}(f(\mathsf{a}))$ *is known.*

We choose to represent a real algebraic number $\alpha \in \mathbb{R}_{alg}$ by the *isolating interval* representation because it is more intuitive, it facilitates geometric applications, and turns out to be as efficient as other representations. It includes a square-free polynomial which vanishes on α and a (rational) interval containing α and no other root.

Proposition 2.4 *[10, 8] Let $f \in \mathbb{Z}[x]$ have degree p and bitsize τ_f. We compute the isolating interval representation of its real roots and their multiplicities in $\widetilde{\mathcal{O}}_B(p^6 + p^4\tau_f^2)$. The endpoints of the isolating intervals have bitsize $\mathcal{O}(p\,\tau_f)$ and $\mathcal{L}(f_{red}) = \mathcal{O}(p + \tau_f)$.*

The sign of the square-free part f_{red} over the interval's endpoints is known; moreover, $f_{red}(\mathsf{a}) f_{red}(\mathsf{b}) < 0$.

Corollary 2.5 [10] *Given a real algebraic number* $\alpha \cong (f, [\mathsf{a}, \mathsf{b}])$*, where* $\mathcal{L}(\mathsf{a}) = \mathcal{L}(\mathsf{b}) = \mathcal{O}(p\tau_f)$*, and* $g \in \mathbb{Z}[x]$*, such that* $\mathsf{dg}(g) = q, \mathcal{L}(g) = \tau_g$*, we compute* $\operatorname{sign}(g(\alpha))$ *in bit complexity* $\widetilde{\mathcal{O}}_B(pq \max\{\tau_f, \tau_g\} + p \min\{p, q\}^2 \tau_f)$.

Pr. 2.4 expresses the state-of-the-art in univariate root isolation. It relies on fast computation of polynomial sequences and the Davenport-Mahler bound, c.f e.g [33]. The following lemma is crucial; it is derived from the Davenport-Mahler bound.

Lemma 2.6 (Aggregate separation) *Given* $f \in \mathbb{Z}[x]$*, the sum of the bitsize of* all *isolating points of the real roots of* f *is* $\mathcal{O}(p^2 + p\,\tau_f)$.

Proof. Let there be $r \leq p$ real roots. The isolating point between two consecutive real roots α_j, α_{j+1} is of magnitude at most $\frac{1}{2}|\alpha_j - \alpha_{j+1}| := \frac{1}{2}\Delta_j$. Thus their product is $\frac{1}{2^r}\prod_j \Delta_j$. Using the Davenport-Mahler bound, $\prod_j \Delta_j \geq 2^{-\mathcal{O}(p^2 + p\tau_f)}$ and we take logarithms. □

We present a new complexity bound on evaluating the sign of a polynomial over a set of algebraic numbers, which have the same defining polynomial, namely of $g(x)$ over all real roots of $f(x)$. It suffices to evaluate $\mathbf{SR}(f, g)$ over all the isolating endpoints of f. The obvious technique, e.g. [10], is to apply cor. 2.5 r times, where r is the number of real roots of f. But we can do better by exploiting aggregate separation:

Lemma 2.7 *Let* $\tau = \max\{\tau_f, \tau_g\}$*. Assume that we have isolated the* r *real roots of* f *and we know the signs of* f *over the isolating endpoints. Then, we can compute the sign of* g *over all* r *roots of* f *in* $\widetilde{\mathcal{O}}_B(p^2 q\tau)$.

Proof. Let s_j be the bitsize of the j-th endpoint, where $0 \leq j \leq r$. The evaluation of $\mathbf{SR}(f, g)$ over this endpoint, by cor. 2.3, has complexity $\widetilde{\mathcal{O}}_B(pq\tau + \min\{p, q\}^2 s_j)$. To compute the overall cost, we should sum over all the isolating points. The first summand is $\widetilde{\mathcal{O}}_B(p^2 q\tau)$. By pr. 2.6, the second summand becomes $\widetilde{\mathcal{O}}_B(\min\{p, q\}^2 p\tau_f)$ and is dominated. □

3 Multivariate polynomials

We discuss multivariate polynomials, using binary segmentation [24]. An alternative approach could be [15]. Let $f, g \in (\mathbb{Z}[y_1, \ldots, y_k])[x]$ with $\mathsf{dg}_x(f) = p \geq q = \mathsf{dg}_x(g)$, $\mathsf{dg}_{y_i}(f) \leq d_i$ and $\mathsf{dg}_{y_i}(g) \leq d_i$. Let $d = \prod_{i=1}^k d_i$ and $\mathcal{L}(f), \mathcal{L}(g) \leq \tau$. $\mathbf{SR}(f, g)$ stands for the signed subresultant sequence of f, g wrt x, except if stated otherwise. The y_i-degree of every polynomial in $\mathbf{SR}(f, g)$, is bounded by $\mathsf{dg}_{y_i}(\mathtt{res}(f, g)) \leq (p + q)d_i$. Thus homomorphism $\psi : \mathbb{Z}[y_1, \ldots, y_k] \to \mathbb{Z}[y]$:

$$y_1 \mapsto\ y,\ y_2 \mapsto\ y^{(p+q)d_1},\ \ldots\ y_k \mapsto\ y^{(p+q)^{k-1} d_1 \cdots d_{k-1}},$$

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

Now $\psi(f),\psi(g) \in (\mathbb{Z}[y])[x]$ have $y-$degree $\leq (p+q)^{k-1}d$ since, in the worst case, f or g hold a monomial such as $y_1^{d_1}\, y_2^{d_2} \ldots y_k^{d_k}$. Thus, $\mathtt{dg}_y(\mathtt{res}(\psi(f),\psi(g))) < (p+q)^k d$.

Proposition 3.1 *[24] We can compute* $\mathbf{SRQ}(f,g)$*, any polynomial in* $\mathbf{SR}(f,g)$*, and* $\mathtt{res}(f,g)$ *in* $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1}d\tau)$.

Lemma 3.2 $\mathbf{SR}(f,g)$ *is computed in* $\widetilde{\mathcal{O}}_B(q(p+q)^{k+2}d\tau)$.

The proof is in the Appendix.

Theorem 3.3 *We can compute* $\mathbf{SR}(f,g\,;\,\mathtt{a})$*, where* $\mathtt{a} \in \mathbb{Q}\cup\{\infty\}$ *and* $\mathcal{L}\,(a) = \sigma$ *in* $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1}d\max\{\tau,\sigma\})$.

Proof. First we compute the sequence $\mathbf{SRQ}(f,g)$ in $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1}d_1d_2\cdots d_k\,\tau)$ (pr. 3.1). Then we evaluate $\mathbf{SRQ}(f,g)$ over $\mathtt{a}$, using binary segmentation but, first, we need to bound the bitsize of the resulting polynomials.

The polynomials in $\mathbf{SR}(f,g)$ have total degree in $y_1,\ldots,y_k$ bounded by $(p+q)\sum_{i=1}^k d_i$ and coefficient bitsize bounded by $(p+q)\tau$. With respect to x, the polynomials in $\mathbf{SR}(f,g)$ have degree bounded by $\mathcal{O}(p)$, so substitution $x = \mathtt{a}$ yields values of size $\widetilde{\mathcal{O}}(p\sigma)$. After the evaluation we obtain polynomials in $\mathbb{Z}[y_1,\ldots,y_k]$ with coefficient bitsize bounded by $\max\{(p+q)\tau, p\sigma\} \leq (p+q)\max\{\tau,\sigma\}$.

Consider $\chi : \mathbb{Z}[y] \to \mathbb{Z}$, such that $y \mapsto 2^{\lceil \mathtt{c}(p+q)\max\{\tau,\sigma\}\rceil}$, for a suitable constant $\mathtt{c}$. Apply the map $\phi = \psi\circ\chi$ to f,g. Now, $\mathcal{L}\,(\phi(f))\,,\mathcal{L}\,(\phi(g)) \leq \mathtt{c}d(p+q)^k\max\{\tau,\sigma\}$. By pr. 2.2, the evaluation costs $\widetilde{\mathcal{O}}_B(q(p+q)^{k+1}d\max\{\tau,\sigma\})$. □

We obtain the following for $f,g \in (\mathbb{Z}[y])[x]$, such that $\mathtt{dg}_x(f) = p$, $\mathtt{dg}_x(g) = q$ and $\mathtt{dg}_y(f), \mathtt{dg}_y(g) \leq d$.

Corollary 3.4 *We compute* $\mathbf{SR}(f,g)$ *in* $\widetilde{\mathcal{O}}_B(pq\max\{p,q\}^2d\tau)$*. For any polynomial* $\mathbf{SR}_j(f,g)$ *in the sequence,* $\mathtt{dg}_x(\mathbf{SR}_j(f,g)) = \mathcal{O}(\max\{p,q\})$*,* $\mathtt{dg}_y(\mathbf{SR}_j(f,g)) = \mathcal{O}(\max\{p,q\}d)$*, and* $\mathcal{L}\,(\mathbf{SR}_j(f,g)) = \mathcal{O}(\max\{p,q\}\tau)$.

Corollary 3.5 *We compute* $\mathbf{SRQ}(f,g)$*, any polynomial in* $\mathbf{SR}(f,g)$*, and* $\mathtt{res}(f,g)$ *with complexity* $\widetilde{\mathcal{O}}_B(pq\max\{p,q\}d\tau)$.

Corollary 3.6 *We compute* $\mathbf{SR}(f,g\,;\,\mathtt{a})$*, where* $\mathtt{a} \in \mathbb{Q}\cup\{\infty\}$ *and* $\mathcal{L}\,(\mathtt{a}) = \sigma$*, with complexity* $\widetilde{\mathcal{O}}_B(pq\max\{p,q\}d\max\{\tau,\sigma\})$*. For the polynomials* $\mathbf{SR}_j(f,g\,;\,\mathtt{a}) \in \mathbb{Z}[y]$*, except for* f,g*, we have* $\mathtt{dg}_y(\mathbf{SR}_j(f,g\,;\,\mathtt{a})) = \mathcal{O}((p+q)d)$ *and* $\mathcal{L}\,(\mathbf{SR}_j(f,g\,;\,\mathtt{a})) = \mathcal{O}(\max\{p,q\}\tau + \min\{p,q\}\sigma)$.

We now reduce the computation of the sign of $F \in \mathbb{Z}[x,y]$ over $(\alpha,\beta) \in \mathbb{R}^2_{alg}$ to that over several points in $\mathbb{Q}^2$. Let $\mathtt{dg}_x(F) = \mathtt{dg}_y(F) = n_1$, $\mathcal{L}\,(F) = \sigma$ and $\alpha \cong (A,[\mathtt{a}_1,\mathtt{a}_2])$, $\beta \cong (B,[\mathtt{b}_1,\mathtt{b}_2])$, where $A,B \in \mathbb{Z}[X]$, $\mathtt{dg}(A) = \mathtt{dg}(B) = n_2$, $\mathcal{L}\,(A) = \mathcal{L}\,(B) = \sigma$. We assume $n_1 \leq n_2$, which is relevant below. The algorithm is Alg. 1, and generalizes the univariate case, cf [10, 33]. For A, resp. B, we assume that we know their values on $\mathtt{a}_1,\mathtt{a}_2$, resp. $\mathtt{b}_1,\mathtt{b}_2$.

Algorithm 1: SIGN_AT(F, α, β)

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

Theorem 3.7 (sign_at) *We compute the sign of $F(x,y)$ over α, β in $\widetilde{\mathcal{O}}_B(n_1^2\, n_2^3\, \sigma)$.*

Proof. First, we compute $\mathbf{SRQ}_x(A, F)$ so as to evaluate $\mathbf{SR}(A, F)$ on the endpoints of α, in $\widetilde{\mathcal{O}}_B(n_1^2 n_2^2 \sigma)$ (cor. 3.5).

We compute the evaluation $\mathbf{SR}(A, F; \mathsf{a}_1)$. The first polynomial in the sequence is A, but we already know its value on a_1. This computation costs $\widetilde{\mathcal{O}}_B(n_1^2\, n_2^3\, \sigma)$ by cor. 3.6 with $q = n_1$, $p = n_2$, $d = n_1$, $\tau = \sigma$, and $\sigma = n_2\sigma$, where the latter corresponds to the size of the endpoints. After the evaluation we obtain a list L_1, which contains $\mathcal{O}(n_1)$ polynomials, say $f \in \mathbb{Z}[y]$, such that $\mathsf{dg}(f) = \mathcal{O}(n_1 n_2)$. To bound the bitsize, notice that the polynomials in the sequence are of degrees $\mathcal{O}(n_1)$ wrt x and of bitsize $\mathcal{O}(n_2\sigma)$. After we evaluate on a_1, $\mathcal{L}(f) = \mathcal{O}(n_1 n_2 \sigma)$.

For each $f \in L_1$ we compute its sign over β and count the sign variations. We could apply directly cor. 2.5, but we can do better. If $\mathsf{dg}(f) \geq n_2$ then $\mathbf{SR}(B, f) = (B, f, -B, g = -\,\mathsf{prem}(-B, f), \dots)$. We should start the evaluations from g, which can be computed in $\widetilde{\mathcal{O}}_B(n_1^2 n_2^3 \sigma)$, using pr. 2.1 and that $\mathsf{dg}(g) = \mathcal{O}(n_2)$ and $\mathcal{L}(g) = \mathcal{O}(n_1 n_2 \sigma)$ (lem. A.1). Thus evaluate $\mathbf{SR}(-B, g; \mathsf{a}_1)$ in $\widetilde{\mathcal{O}}_B(n_1 n_2^3 \sigma)$, by cor. 2.5, with $p = q = n_2$, $\tau_f = \sigma$, $\tau = n_1 n_2 \sigma$. If $\mathsf{dg}(f) < n_2$ the complexity is dominated. Since we must perform $\mathcal{O}(n_1)$ such evaluations, all them cost $\widetilde{\mathcal{O}}_B(n_1^2 n_2^3 \sigma)$.

We do the same for the other endpoint of α and we subtract the sign variations. Notice that $\text{sign}(A'(\alpha)) = \text{sign}(A(\mathsf{b}_1) - A(\mathsf{a}_1))$, which is known from the real root isolation process that constructed α. □

We wish to count the number of real roots of $\bar{F} = F(\alpha, y) \in (\mathbb{Z}(\alpha))[y]$ in $(-\infty, +\infty)$, in $(\mathsf{c}, +\infty)$ and in $(\beta, +\infty)$. This is essential in finding the topology of real plane algebraic curves, but also for analyzing the complexity of real root isolation of univariate polynomials over an extension field.

Let $F \in \mathbb{Z}[x,y]$, such that $\mathsf{dg}_x(F) = \mathsf{dg}_y(F) = n_1$ and $\mathcal{L}(F) = \sigma$. Let $\alpha = (A, [\mathsf{a}_1, \mathsf{a}_2])$ and $\beta = (B, [\mathsf{b}_1, \mathsf{b}_2])$, where $\mathsf{dg}(A), \mathsf{dg}(B) = n_2, \mathcal{L}(A), \mathcal{L}(B) \leq \tau$ and $\mathsf{c} \in \mathbb{Q}$, such that $\mathcal{L}(\mathsf{c}) = \lambda$. Moreover, assume that $n_1^2 = \mathcal{O}(n_2)$, as is the case in applications.

Theorem 3.8 *We count the real roots of $\bar{F}$ in $(-\infty, +\infty)$, $(\mathsf{c}, +\infty)$ and $(\beta, +\infty)$ with complexities $\widetilde{\mathcal{O}}_B(n_1^4 n_2\sigma + n_1^5 n_2\tau)$, $\widetilde{\mathcal{O}}_B(n_1^5 n_2^3 \max\{n_1\sigma, \tau\})$ and $\widetilde{\mathcal{O}}_B(n_1^4 n_2 \max\{n_1\tau, \sigma, \lambda\})$, respectively.*

The proof uses Sturm's theorem and the good specialization properties of subresultants in order to switch the order of substitution $x = \alpha$ and sequence computation; see the Appendix for the full proof.

4 Bivariate real solving

Let $F, G \in \mathbb{Z}[x, y]$ be relatively prime and $\mathtt{dg}(F) = \mathtt{dg}(G) = n$ and $\mathcal{L}(F) = \mathcal{L}(G) = \sigma$. We study three algorithms and their complexity for real solving the system $F = G = 0$. The main idea behind the 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.

4.1 The GRID algorithm

Algorithm GRID, cf. Alg. 3, is straightforward and its first phase was also used in [31]. 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 the algorithm 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. The algorithm also outputs rational axis-aligned rectangles, guaranteed to contain a single root of the system.

To the best of our knowledge, this is the first time that the algorithm's complexity is studied. The disadvantage of the algorithm is that exact implementation of its sub-algorithm SIGN_AT (Alg. 1) is not efficient. However, its simplicity makes it attractive and arithmetic filtering can speed up its implementation. The algorithm requires no genericity assumption on the input; we study a generic shear that brings the system to generic position in order to compute the multiplicities within the same complexity bound.

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. More importantly, we can count the real roots with a given abscissa α by th. 3.8.

Theorem 4.1 *Isolating all real roots of the system $F = G = 0$ using* GRID *has complexity $\widetilde{\mathcal{O}}_B(n^{14} + n^{13}\sigma)$, provided $\sigma = O(n^3)$.*

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 $\widetilde{\mathcal{O}}_B(n^4\sigma)$, using cor. 3.5 with $p = q = d = n$ and $\tau = \sigma$. Notice that $\mathtt{dg}(R_x) = \mathcal{O}(n^2)$ and $\mathcal{L}(R_x) = \mathcal{O}(n\,\sigma)$.

We represent the real roots of R_x in isolating interval representation with complexity $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma^2)$ (pr. 2.4) and store them in L_x. This complexity shall be dominated provided that $\sigma = O(n^3)$. We do the same for the y axis and store the roots 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 bitsize of the polynomial is $\mathcal{O}(n^2 + n\,\sigma)$ [2, 10]. The isolating intervals have endpoints of size $\mathcal{O}(n^4 + n^3\,\sigma)$.

Let r_x, resp. r_y be the number of real roots of the corresponding resultants. Both are bounded by $\mathcal{O}(n^2)$. We 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. This is achieved by SIGN_AT (Alg. 1). Each evaluation costs $\widetilde{\mathcal{O}}_B(n^{10} + n^9\sigma)$ using th. 3.7, with $n_1 = n$, $n_2 = n^2$ and $\sigma = n^2 + n\sigma$. We perform $r_x\, r_y = O(n^4)$ sign evaluations. □

We now examine the multiplicity of a root (α, β) of the system. Refer to [3, sec.II.6] for its definition as the exponent of factor $(\beta x - \alpha y)$ in the resultant of the (homogenized) polynomials, under certain assumptions.

The 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 apply a shear transform to the coordinate frame so as to ensure that different roots project to different points on the x-axis. We determine an adequate (horizontal) shear by specializing $t \mapsto t_0 \in \mathbb{Z}$ such that

$$R_t(x) = \mathtt{res}_y\left(F(x + ty, y), G(x + ty, y)\right), \tag{1}$$

when specialized to t_0, has distinct roots corresponding to the projections of the common roots of the system $F(x, y) = G(x, y) = 0$ and the degree of the polynomials remains the same. Previous work includes [11, 27, 32]. Let $R_{red} \in (\mathbb{Z}[t])[x]$ denote the square-free part of the resultant as an element of UFD $(\mathbb{Z}[t])[x]$, and let its discriminant, with respect to x, be $\Delta \in \mathbb{Z}[t]$. Then the specialization t_0 must be such that $\Delta(t_0) \neq 0$.

Lemma 4.2 *Computing a $t \in \mathbb{Z}$, such that the corresponding shear is sufficiently generic, has complexity $\widetilde{\mathcal{O}}_B(n^9\sigma)$.*

The idea here is to use explicit candidate values of t right from the start; see the Appendix. In practice, the above complexity becomes $\widetilde{\mathcal{O}}_B(n^5\sigma)$, because a constant number of tries will typically suffice. Now we use the value of t to compute the multiplicities of the roots of the sheared system. Then, we need to match the latter with the roots of the original system. The proof of the following is in the Appendix.

Theorem 4.3 *Consider the setting of th. 4.1. Having isolated all real roots of the system $F = G = 0$, it is possible to determine their multiplicities with complexity $\widetilde{\mathcal{O}}_B(n^{12} + n^{11}\sigma + n^{10}\sigma^2)$.*

Algorithm 2: M_RUR (F, G)

Input: $F, G \in \mathbb{Z}[X, Y]$ in generic position
Output: The real solutions of the system $F = G = 0$

```
1  SR ← SR_y(F, G)
   /* Projections                                                    */
2  R_x ← res_y(F, G)
3  P_x, M_x ← SOLVE(R_x)
4  R_y ← res_x(F, G)
5  P_y, M_y ← SOLVE(R_y)
6  I ← INTERMEDIATE_POINTS(P_y)
   /* Factorization of R_x according to sr                           */
7  K ← COMPUTE_K(SR, P_x)
8  Q ← ∅
   /* Matching the solutions                                         */
9  foreach α ∈ P_x do
10 |  β ← FIND(α, K, P_y, I)
11 |  Q ← ADD(Q, {α, β})
12 RETURN Q
```

4.2 The M_RUR algorithm

M_RUR assumes that the polynomials are in generic position. Then:

Proposition 4.4 *[11, 2] Let F, G square-free and co-prime polynomials, in generic position. If $\mathbf{SR}_j(x, y) = \mathbf{sr}_j(x)y^j + \mathbf{sr}_{j,j-1}(x)y^{j-1} + \cdots + \mathbf{sr}_{j,0}(x)$, then if (α, β) is a real solution of the system $F = G = 0$, then there exists k, such that $\mathbf{sr}_0(\alpha) = \cdots = \mathbf{sr}_{k-1}(\alpha) = 0$, $\mathbf{sr}_k(\alpha) \neq 0$ and $\beta = -\frac{1}{k}\frac{\mathbf{sr}_{k,k-1}(\alpha)}{\mathbf{sr}_k(\alpha)}$.*

Pr. 4.4 represents the ordinate of a solution in a Rational Univariate Representation (RUR) of the abscissa. The RUR applies to multivariate algebraic systems [4, 25, 5, 26, 7, 2]; it generalizes the primitive element method [28], initiated by Kronecker. Here we adapt it to small-dimensional systems.

The algorithm that we present is similar to the algorithm of [13, 11]. However, notice their algorithm computes only a RUR of the roots using pr. 4.4, so the representation of the ordinates remains implicit. If we wish to compute with these numbers, this representation is not sufficient (we can always compute the minimal polynomial of the roots [19] in this algorithm, but this is highly inefficient). We modify the algorithm, so that the output includes isolating interval rectangles, hence the name modified-RUR (M_RUR).

Our most important difference with [11] is that they represent algebraic numbers by Thom's encoding while we use isolating intervals, which were thought of having high theoretical complexity. We will prove that this is not the case.

Analysis of M_RUR.

The pseudo-code is in Alg. 2. We project on the x and the y-axis; for each real solution on the x-axis we compute its ordinate using pr. 4.4. First we compute the sequence $\mathbf{SR}(F,G)$ wrt y in $\widetilde{\mathcal{O}}_B(n^5\,\sigma)$ (cor. 3.4).

Projection. The projection on the two axis is similar to GRID algorithm. The complexity is dominated by real solving the resultants, i.e $\widetilde{\mathcal{O}}_B(n^{12}+n^{10}\,\sigma^2)$. Let α_i, resp. β_j, be the real roots in the x, resp. y axis. Moreover, we compute rational numbers q_j between the β_j's in $\widetilde{\mathcal{O}}_B(n^5\sigma)$; the q_j have aggregate bitsize $\mathcal{O}(n^3\,\sigma)$:

$$q_0 < \beta_1 < q_1 < \beta_2 < \cdots < \beta_{\ell-1} < q_{\ell-1} < \beta_\ell < q_\ell, \tag{2}$$

where $\ell \leq 2\,n^2$. Every β_j corresponds to a unique α_i. Notice that the multiplicity of α_i as a root of R_x is the multiplicity of real solution of the system, that has it as abscissa.

The sub-algorithm COMPUTE_K. In order to apply pr. 4.4, for every α_i we must compute $k \in \mathbb{N}^*$ such the assumptions of the theorem are fulfilled; this is possible by genericity. We follow [22, 11] and define recursively polynomials $\Gamma_j(x)$: Let $\Phi_0(x) = \frac{\mathbf{sr}_0(x)}{\mathsf{gcd}(\mathbf{sr}_0(x),\mathbf{sr}_0'(x))}$ and $\Phi_j(x) = \mathsf{gcd}(\Phi_{j-1}(x),\mathbf{sr}_j(x))$ and $\Gamma_j = \frac{\Phi_{j-1}(x)}{\Phi_j(x)}$, for $j>0$. Now $\mathbf{sr}_i(x) \in \mathbb{Z}[x]$ is the principal subresultant coefficient of $\mathbf{SR}_i \in (\mathbb{Z}[x])[y]$, and $\Phi_0(x)$ is the square-free part of $R_x = \mathbf{sr}_0(x)$. By construction, $\Phi_0(x) = \prod_j \Gamma_j(x)$ and $\mathsf{gcd}(\Gamma_j,\Gamma_i)=1$, if $j\neq i$. Hence every α_i is a root of a unique Γ_j and the latter switches sign at the interval's endpoints. Then, $\mathbf{sr}_0(\alpha) = \mathbf{sr}_1(\alpha) = 0,\ldots,\mathbf{sr}_j(\alpha)=0$, $\mathbf{sr}_{j+1}(\alpha)\neq 0$; thus $k=j+1$.

It holds that $\mathsf{dg}(\Phi_0) = \mathcal{O}(n^2)$ and $\mathcal{L}(\Phi_0) = \mathcal{O}(n^2+n\,\sigma)$. Moreover, $\sum_j \mathsf{dg}(\Gamma_j) = \sum_j \delta_j = \mathcal{O}(n^2)$ and, by Mignotte's bound [20], $\mathcal{L}(\Gamma_j) = \mathcal{O}(n^3\sigma)$. In order to compute the factorization $\Phi_0(x) = \prod_j \Gamma_j(x)$ wrt to $\mathbf{sr}_j(x)$, we perform $\mathcal{O}(n)$ gcd computations of polynomials of degree $\mathcal{O}(n^2)$ and bitsize $\mathcal{O}(n^3\sigma)$. Each gcd computation costs $\widetilde{\mathcal{O}}_B(n^7\,\sigma)$ (pr. 2.1) and thus the overall cost is $\widetilde{\mathcal{O}}_B(n^8\,\sigma)$.

We compute the sign of the Γ_j over all the $O(n^2)$ isolating endpoints of of the α_i, that have aggregate bitsize $\mathcal{O}(n^3\,\sigma)$ (lem. 2.6) in $\widetilde{\mathcal{O}}_B(\delta_j n^5\sigma + \delta_j^2 n^3\sigma)$, using Horner's rule. Summing over all δ_j, we conclude that the complexity is $\widetilde{\mathcal{O}}_B(n^7\sigma)$. Thus the overall complexity of COMPUTE_K is $\widetilde{\mathcal{O}}_B(n^8\,\sigma)$.

Matching the solutions and the algorithm FIND. The process takes a real root of R_x and computes the ordinate β of the corresponding root of the system. For some real root α of R_x we represent the ordinate $A(\alpha) = -\frac{1}{k}\frac{\mathbf{sr}_{k,k-1}(\alpha)}{\mathbf{sr}_k(\alpha)} = \frac{A_1(\alpha)}{A_2(\alpha)}$. The generic position assumption guarantees that there is a unique β_j, in P_y, such that üôé $\beta_j = A(\alpha)$, where $1\leq j\leq \ell$. In order to compute j we use (2): $q_j < A(\alpha) = \frac{A_1(\alpha)}{A_2(\alpha)} = \beta_j < q_{j+1}$. Thus j can be computed by binary search in $\mathcal{O}(\lg\ell) = \mathcal{O}(\lg n)$ comparisons of $A(\alpha)$ with the q_j. This is equivalent to computing the sign of $B_j(X) = A_1(X) - q_j\,A_2(X)$ over α by executing $\mathcal{O}(\lg n)$ times, SIGN_AT(B_j,α).

Now, $\mathcal{L}(q_j) = \mathcal{O}(n^3\sigma)$ and $\mathsf{dg}(A_1) = \mathsf{dg}(\mathbf{sr}_{k,k-1}) = \mathcal{O}(n^2)$, $\mathsf{dg}(A_2) = \mathsf{dg}(\mathbf{sr}_k) = \mathcal{O}(n^2)$, $\mathcal{L}(A_1) = \mathcal{O}(n\,\sigma)$, $\mathcal{L}(A_2) = \mathcal{O}(n\,\sigma)$. Thus $\mathsf{dg}(B_j) = \mathcal{O}(n^2)$ and $\mathcal{L}(B_j) = \mathcal{O}(n^3\,\sigma)$. We conclude that SIGN_AT(B_j,α) and FIND have complexity $\widetilde{\mathcal{O}}_B(n^7\sigma)$ (cor. 2.5). As for the

overall complexity of the loop (Lines 9-11) the complexity is $\widetilde{\mathcal{O}}_B(n^9\sigma)$, since it can be executed $\mathcal{O}(n^2)$ times.

Theorem 4.5 *Let $F, G \in \mathbb{Z}[x, y]$ such that they are in generic position, their total degrees are bounded by n, and their bitsize by σ. If the polynomials are not relatively prime, the algorithm reports this and stops. Otherwise, it isolates all real roots of the system $F = G = 0$ with complexity $\widetilde{\mathcal{O}}_B(n^{10}\sigma^2)$.*

The generic position assumption is without loss of generality since we can always put the system in such position by applying a shear transformation $(X, Y) \mapsto (X + tY, Y)$, where t is either a random number or computed deterministically, c.f [11, 27] and lem. 4.2. The bitsize of the (sheared) system becomes $\widetilde{\mathcal{O}}(n + \sigma)$ and does not change the bound of th. 4.5. However, now is raised the problem of expressing the real roots in the original coordinate system. We will report on this in a future work (see also the proof of th. 4.3).

4.3 The G_RUR algorithm

We present an algorithm that uses some ideas from RUR but relies on GCD computations of polynomials with coefficients in an extension field to achieve efficiency (hence the name G_RUR). For the GCD computations we use the algorithm (and the implementation) of [29].

The first steps are similar to the previous algorithms: We project on the axes, we perform real solving and compute the intermediate points on the y-axis. The complexity is $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma^2)$.

For each x solution, say α, we compute the square-free part of $F(\alpha, y)$ and $G(\alpha, y)$, say $\bar{F}$ and $\bar{G}$. The complexity is that of computing the gcd with the derivative. In [29] it is stated that the cost is $\widetilde{\mathcal{O}}_B(mMND+mN^2D^2+m^2nD)$, where M is the bitsize of the largest integer coefficient appeared, N is the degree of the largest polynomial, D is the degree of the extension, n is the degree of the gcd, and m is the number of primes needed. The complexity does not assume fast multiplication algorithms, thus, under this assumption, the complexity becomes $\widetilde{\mathcal{O}}_B(mMND + mND + mnD)$.

In our case $M = \mathcal{O}(\sigma)$, $N = \mathcal{O}(n)$, $D = \mathcal{O}(n^2)$, $n = \mathcal{O}(n)$, and $m = \mathcal{O}(n\sigma)$. The cost is $\widetilde{\mathcal{O}}_B(n^4\sigma^2)$ and since we have to do it $\mathcal{O}(n^2)$ times, the overall cost is $\widetilde{\mathcal{O}}_B(n^6\sigma^2)$. Notice the bitsize of the result is $\widetilde{\mathcal{O}}_B(n + \sigma)$ [2].

Now for each α, we compute the polynomial $H = \gcd(\bar{F}, \bar{G})$. In this case we have $M = \mathcal{O}(n + \sigma)$, $N = \mathcal{O}(n)$, $D = \mathcal{O}(n^2)$, $n = \mathcal{O}(n)$, and $m = \mathcal{O}(n^2 + n\sigma)$ and so the cost of each operation is $\widetilde{\mathcal{O}}_B(n^6 + n^4\sigma^2)$ and the overall $\widetilde{\mathcal{O}}_B(n^8 + n^6\sigma^2)$. The size of m comes from Mignotte's bound [20]. Notice that H is a square-free polynomial in $(\mathbb{Z}(\alpha))[y]$, of degree $\mathcal{O}(n)$ and bitsize $\mathcal{O}(n^2 + n\sigma)$, the real roots of which correspond to the real solutions of the system with abscissa α. It should change sign only over the intervals that contains its real roots. To check these signs, we have to substitute y in H by the intermediate points, thus obtaining a polynomial in $\mathbb{Z}(\alpha)$, of degree $\mathcal{O}(n)$ and bitsize $\mathcal{O}(n^2+n\sigma+ns_j)$, where s_j is the bitsize of the j intermediate point. Now, we consider this polynomial in $\mathbb{Z}[X]$ and we have to

evaluate it over α. Using cor. 2.5 with $p = n^2$, $\tau_f = n\sigma$, $q = n$, and $\tau_g = n^2 + n\sigma + ns_j$, this costs $\widetilde{\mathcal{O}}_B(n^6 + n^5\sigma + n^4 s_j)$. Summing over all the points, there are $\mathcal{O}(n^2)$, and using pr. 2.6, we obtain a complexity of $\widetilde{\mathcal{O}}_B(n^8 + n^7\sigma)$. Thus the overall complexity is $\widetilde{\mathcal{O}}_B(n^{10} + n^9\sigma)$.

Theorem 4.6 *We can isolate the real roots of the system $F = G = 0$, using* G_RUR *in $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma)$.*

5 Applications

Simultaneous inequalities in two variables. Let $P, Q, A_1, \ldots, A_{\ell_1}, B_1, \ldots, B_{\ell_2}, C_1, \ldots, C_{\ell_3} \in \mathbb{Z}[X, Y]$, such that their total degrees are bounded by n and their bitsize by σ. We wish to compute real number pairs (α, β) such that they are the real solutions of system $P(X, Y) = Q(X, Y) = 0$ and also $A_i(\alpha, \beta) > 0$, $B_j(\alpha, \beta) < 0$ and $C_k(\alpha, \beta) = 0$, where $1 \leq i \leq \ell_1, 1 \leq j \leq \ell_2, 1 \leq k \leq \ell_3$. Let $\ell = \ell_1 + \ell_2 + \ell_3$.

Corollary 5.1 *There is an algorithm that solves the problem of ℓ simultaneous inequalities of degree $\leq n$ and bitsize $\leq \sigma$, in $\widetilde{\mathcal{O}}_B(\ell n^{12} + \ell n^{11}\sigma + n^{10}\sigma^2)$.*

The complexity of topology. We apply our techniques to improve the complexity of computing the topology of a real plane algebraic curve. We consider the curve defined by $F \in \mathbb{Z}[x, y]$, such that $\mathsf{dg}(F) = n$ and $\mathcal{L}(F) = \sigma$.

The first steps consist in computing the critical points of the curve, i.e to solve the system $F = F_y = 0$. Using any of the previous two algorithms, this can be done in $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma^2)$.

Next, we compute the intermediate points on the x axis, in $\widetilde{\mathcal{O}}_B(n^3\sigma)$ (lem. 2.6). For each intermediate point, say q_j we need to compute the number of branches of the curve that cross the vertical line $x = q_j$. This is equivalent to computing the number of real solutions of the polynomial $F(q_j, y)$. Notice that $\mathcal{L}(q_j) = \mathcal{O}(n^3\sigma)$, thus $\mathcal{L}(F(q_j, y)) = \mathcal{O}(n^4\sigma)$. We can use th. 3.8 with complexity $\widetilde{\mathcal{O}}_B(n^4\tau + n^5\mathcal{L}(q_j))$. Since the q_j's are $\mathcal{O}(n^2)$, the total cost of this step is $\widetilde{\mathcal{O}}_B(n^6\sigma)$ by applying pr. 2.6.

In the sequel, for each critical point, say (α, β) we need to compute the number of branches of the curve that cross the vertical line $x = \alpha$, and the number of them that are above $y = \beta$. The first task corresponds to computing the number of real roots of $F(\alpha, y)$, by application of th. 3.8, in $\widetilde{\mathcal{O}}_B(n^9 + n^8\sigma)$, where $n_1 = n$, $n_2 = n^2$, $\sigma = \sigma$ and $\tau = n^2 + n\sigma$. Since there are $\mathcal{O}(n^2)$ critical values, the overall cost of the step is $\widetilde{\mathcal{O}}_B(n^{11} + n^{10}\sigma)$.

Finally, we should compute the number of branches that cross the line $x = \alpha$ and are above $y = \beta$. We can do it by application of th. 3.8 in $\widetilde{\mathcal{O}}_B(n^{13} + n^{12}\sigma)$. Since there are $\mathcal{O}(n^2)$ critical points, the complexity of the step is $\widetilde{\mathcal{O}}_B(n^{15} + n^{14}\sigma)$. It remains to connect the critical points according to the information that we have for the branches. The complexity of this step is negligible.

It now follows that the complexity of the algorithm is $\widetilde{\mathcal{O}}_B(n^{15} + n^{14}\sigma + n^{10}\sigma^2)$, or $\widetilde{\mathcal{O}}_B(N^{15})$, which improves the previously known bound [11] by a factor.

However, we can do better, since we can improve the complexity of the last step. Recall, that M_RUR also computes the RUR representation of the ordinates of the solutions. Thus, instead of performing bivariate sign evaluations in order to compute the number of branches above $y = \beta$, we can substitute the RUR representation of β and perform univariate sign evaluations.

This corresponds [11], to compute the sign of $\mathcal{O}(n^2)$ polynomials of degree $\mathcal{O}(n^2)$ and bitsize $\mathcal{O}(n^3\sigma)$, over all the α's. Using lem. 2.7 for each polynomial the cost is $\widetilde{\mathcal{O}}_B(n^9\sigma)$, and since there are $\widetilde{\mathcal{O}}_B(n^2)$, the total cost becomes $\widetilde{\mathcal{O}}_B(n^{11}\sigma)$.

Thus the overall complexity of the algorithm is as follows which improves the previously known bound by a factor of N^4.

Theorem 5.2 *We can compute the topology of a real plane algebraic curve, defined by a polynomial of degree n and bitsize σ, in $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma)$.*

We have assumed generic position, since we can apply an adequate transformation, or apply the online algorithm of [13], cf [2]. This does not affect the complexity of the algorithm.

6 Implementation and experiments

This section describes our MAPLE implementation and illustrates its capabilities through comparative experiments. Our library is open source software[1]. The design of the library is based on object oriented programming and the generic programming paradigm, common to `C++`, so as to be easy to transfer our implementation in `C++`, in the future.

The core object in our library is the class of real algebraic numbers, that are represented in isolating interval representations. We provide functionalities for computing signed polynomial remainder sequences with various algorithms; real solving univariate polynomials using Sturm's algorithm; computations with one and two real algebraic numbers, such as construction, sign evaluation, comparison; and, of course, our algorithms for real solving of bivariate polynomial systems. In order to speed up the computations we have implemented a filter layer, i.e, computations are performed first using intervals with floating point arithmetic and, if they fail, then an exact algorithm using rational arithmetic is called. For GCD computations in an extension field we use the MAPLE package of [29]. We have not implemented, yet, the optimal algorithms for computing and evaluating polynomial remainder sequences. For a demonstration of our software, see below and refer to the Appendix.

We have implemented the algorithms for bivariate polynomial real solving, namely, GRID, M_RUR and G_RUR. The following is an example of how to use our library with GRID: notice the 2nd root is integer. To use another algorithm, the user changes `solveGRID` to `solveMRUR` or `solveGRUR`.

```
f := 1 + 2*x + x^2*y - 5*x*y + x^2:
g := 2*x + y - 3:
bivsols := SLV:-solveGRID ( f, g ):
SLV:-display_2 ( bivsols );
```

[1] www.di.uoa.gr/~stud1098/SLV

```
< 2*x^2-12*x+1, [ 3, 7], 5.915475965 > ,
  < x^2+6*x-25, [ -2263/256, -35/4], -8.830718995 >
< x-1, [ 1, 1], 1 > , < x-1, [ 1, 1], 1 >
< 2*x^2-12*x+1, [3/64, 3/32], .8452400565e-1 > ,
  < x^2+6*x-25, [23179/8192, 2899/1024], 2.830943108 >
```

We performed various experiments with our solvers using bivariate polynomial systems and curves, of small and medium total degree. The experiments were performed on an AMD64@3000+ under 64bit Linux (kernel 2.6.15-27-amd64-generic) with 1GB RAM on a MAPLE 9.5 console. Overall performance results are shown on tab. 1, averaged over 10 iterations. Refer to the Appendix for the polynomial systems used.

Let us start with reporting on seminumerical filtering. Among our algorithms, GRID and M_RUR benefit the most from the filtering techniques. They gain a factor of 10, while G_RUR gains only a factor of 2. It seems that G_RUR is our solver of choice. However, this is true only when the extension field is not of high degree. In tab. 2 you can see the time that each algorithm spends on the various steps, e.g projection, real solving, matching the solutions and in tab. 4 the statistics.

Apart from the filtering described earlier, M_RUR uses one more filtering technique. After computing the intermediate points on the x axis, we perform refining (up to 20 times on systems with high degree) on the candidate solutions on the x-axis. M_RUR searches for solutions on a stratum basis; hence it is reasonable that the candidate solutions that stem from the resultant are refined so as to help our filtering techniques in sign determination and avoid costly operations. Of course the refinement must not be excessive since this will increase the bitsize of the coefficients and as a result lead to costlier operations in case where filtering techniques fail. This technique has proven very efficient in practice.

If a polynomial system did not comply with the Generic Position criterion required by M_RUR, we deterministically computed a value for the required shear; in all cases our first candidate ($t = 3$) worked. The time required for this computation can be viewed on tab. 3. If someone wishes to consider a deterministic shear, he has to add the timings presented in the two tables. This seems inexpensive on systems where polynomials are of degree up to 5. In worst case scenarios, e.g. system D_1, we have indications that random shear is the proper choice.

We also tested other software packages: INSULATE, which is a MAPLE package that implements [32] for computing the topology of real algebraic curves. The same holds for TOP, which implements [13]. We tried to modify the previous packages so as to stop them as soon as they compute the real solutions of the corresponding bivariate system. Both packages were kindly provided to us by their authors.

We also tested GBRS [26], which performs exact real solving using Gröbner basis and RUR, through its MAPLE interface. Lastly, we examined 3 SYNAPS solvers: STURM is a very naive implementation of the GRID algorithm, based on a previous work of ours; SUBDIV implements the algorithm in [21], based on the Bernstein basis and `double` arithmetic; and NEWMAC, which is a general purpose solver based on normal-form algorithms and computations of eigenvectors using LAPACK, which computes all complex solutions.

For the first data sets there are no timings for INSULATE and TOP since it was not easy to us to modify their code so as to deal with polynomial systems. The second part of the data sets consider algebraic curves, i.e polynomial systems of the form $f = f_y = 0$, that all packages can deal with.

We underline that we do not consider experiments as competition, but a crucial step for improving existing software packages. Moreover, it is very difficult to compare software, since in most cases they are made for different needs. In addition, accurate timing in MAPLE is very hard, since it is a general purpose package and a lot of overhead is added to its function calls. For example this is the case for GBRS.

Our solvers GRID and M_RUR demonstrate a high fluctuation in runtimes, compared, e.g, to the stability of GBRS. However, GBRS is based on a fundamentally different approach to Real Solving making it exhibit these results. On the other hand, the rest of the solvers demonstrate a similar fluctuation, especially those that are based on MAPLE, which indicates that all of them are using filtering techniques while trying to exploit MAPLE's modular built-in procedures.

Regarding the performance of G_RUR, although on average it performs slower than GBRS by a factor of around 3, it yields solutions in less than a second, apart from systems C_5 and W_5 where its performance is similar to that of GBRS. This discrepancy is expected since GBRS is not based entirely on high-level MAPLE coding, which is the case of our solvers.

Our implementation and INSULATE have demonstrated the most robust behaviour, not only by replying within our specified time limits, but also no errors were generated during their execution. In the case of GBRS, some errors regarding the communication of the application with the MAPLE kernel were generated, especially on the difficult systems C_5 and W_5. On the other hand TOP reported an error in subscript selection in TOPTYP in the case of systems M_4 and D_1. Finally, the STURM solver of SYNAPS, presented a floating point exception on the sheared system M_4 and failed to reply within our time limits in two more cases.

As for NEWMAC, some error is introduced since it is based on LAPACK for computing eigenvalues. This is also the case for the SUBDIV solver; we have to mention that this solver was originally made for computing self-intersections in the unit cube. As for the STURM solver, its inefficiency is basically due to the fact that it evaluates determinants in order to compute square-free parts.

To summarize, we believe that the implementation of the projection-based algorithms presented give very encouraging results, at least for polynomial systems of moderate degree.

system	deg		sheared	solutions	Average Time (msecs)									
					BIVARIATE SOLVING							TOPOLOGY		
					SLV			GBRS	SYNAPS			INS	TOP	
	f	g			GRID	MRUR	GRUR		sturm	subdiv	nm		Top_{60}	top_{500}
R_1	3	4	☑	2	10	11	6	27	6	1,372*	12	–	–	–
R_2	3	1		1	25	21	40	24	1	2	2	–	–	–
R_3	3	1		1	1	2	1	24	1	17	2*	–	–	–
M_1	3	3	☑	4	119	61	46	26	7	27*	2	–	–	–
M_2	4	2		3	6	6	5	25	1	317*	2	–	–	–
M_3	6	3	☑	5	2,627	850	441	32	950	12,660*	9	–	–	–
M_4	9	10		2	348	323	280	171	f.p.e.	5	384	–	–	–
D_1	4	5		1	7	11	6	30	4	fail	7	–	–	–
D_2	2	2		4	383	114	147	27	28	5	2	–	–	–
C_1	7	6	☑	6	2,259	918	257	91	1,198	63*	150*	1,258	330	1040
C_2	4	3	☑	6	365	174	106	30	101	2,043*	6*	127	38	120
C_3	8	7	☑	13	309	1,723	148	61	154	111*	173*	1,740	436	1,410
C_4	8	7	☑	17	4,127	5,029	492	148	8,575	121*	343*	8,417	1,109	3,590
C_5	16	15	☑	17	353,512	48,327	6,130	6,106	>7 hrs	18,726*	6,392*	252,123	40,674	100,955
W_1	7	6	☑	9	2,813	1,945	404	96	2,368	106*	38	1,290	321	1,059
W_2	4	3	☑	5	843	263	218	30	160	1,235*	12*	128	38	122
W_3	8	7	☑	13	1,958	1,822	243	72	3,332	153*	272*	1,722	432	1,424
W_4	8	7	☑	17	8,912	4,988	747	162	26,779	169*	341*	8,303	1,051	3,442
W_5	16	15	☑	17	411,563	50,949	6,442	5,716	>7 hrs	19,238*	3,580*	249,919	40,950	97,894

Table 1: Running Times are averages over 10 runs.

Acknowledgements. The authors acknowledge fruitful discussions with B. Mourrain and also thank J-P. Pavone for his help with SYNAPS. The third author also thanks F. Rouillier for various comments.

References

[1] D. Arnon and S. McCallum. A polynomial time algorithm for the topological type of a real algebraic curve. *JSC*, 5:213–236, 1988.

[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] E. Brieskorn and H. Knörrer. *Plane Algebraic Curves*. Birkhäuser, Basel, 1986.

[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. STOC*, pages 460–467, 1988.

[6] F. Cazals, J.-C. Faugère, M. Pouget, and F. Rouillier. The implicit structure of ridges of a smooth parametric surface. Technical Report 5608, INRIA, 2005.

[7] A. Dickenstein and I. Emiris, editors. *Solving Polynomial Equations: Foundations, Algorithms and Applications*, volume 14 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, May 2005.

[8] Z. Du, V. Sharma, and C. K. Yap. Amortized bound for root isolation via Sturm sequences. In D. Wang and L. Zhi, editors, *Int. Workshop on Symbolic Numeric Computing*, pages 81–93, Beijing, China, 2005.

[9] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the descartes method. In *ISSAC*, pages 71–78, New York, NY, USA, 2006. ACM Press.

[10] 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, 2007. also available in www.inria.fr/rrrt/rr-5897.html.

[11] 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.

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

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

[14] M. Kerber. Analysis of real algebraic plane curves. Diploma thesis, MPI Saarbrücken, 2006.

[15] J. Klose. Binary segmentation for multivariate polynomials. *J. Complexity*, 11(3):330–343, 1995.

[16] K. Ko, T. Sakkalis, and N. Patrikalakis. Resolution of multiple roots of nonlinear polynomial systems. *Int. J. of Shape Modeling*, 11(1):121–147, 2005.

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

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

[19] R. Loos. Computing in algebraic extensions. In B. Buchberger, G. E. Collins, R. Loos, and R. Albrecht, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 173–187. Springer-Verlag, 1983.

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

[21] B. Mourrain and J.-P. Pavone. Subdivision methods for solving polynomial equations. Technical Report RR-5658, INRIA Sophia-Antipolis, 2005.

[22] B. Mourrain, S. Pion, S. Schmitt, J.-P. Técourt, E. Tsigaridas, and N. Wolpert. Algebraic issues in computational geometry. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 117–155. Springer-Verlag, 2006.

[23] V. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and rootfinding. *JSC*, 33(5):701–733, 2002.

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

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

[26] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *J. of AAECC*, 9(5):433–461, 1999.

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

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

[29] M. van Hoeij and M. Monagan. A modular GCD algorithm over number fields presented with multiple extensions. In *ISSAC*, pages 109–116, July 2002.

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

[31] N. Wolpert. *An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics*. PhD thesis, MPI fuer Informatik, Oct. 2002.

[32] N. Wolpert and R. Seidel. On the Exact Computation of the Topology of Real Algebraic Curves. In *SoCG*, pages 107–115. ACM, 2005.

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

System	SLV														
	GRID			MRUR							GRUR				
	Projections	Univariate	Bivariate	Projection on x-axis	Univariate	StHa Sequence	Interm. Points	Filtering on x-axis	Compute K	FIND (Biv. Sol.)	Projections	Univariate	Interm. Points	Rational Bivariate	$\mathbb{R}_{alg}$ Bivariate
R_1	0.47	25.83	73.40	0.00	25.79	14.66	0.56	0.56	14.29	44.16	0.30	45.97	1.09	52.39	0.00
R_2	0.08	38.31	61.59	0.00	14.03	0.57	0.19	74.50	3.79	6.92	0.05	6.54	0.11	0.21	93.10
R_3	1.47	60.29	37.75	0.13	30.17	17.21	0.86	2.92	22.99	25.71	0.70	40.68	2.93	55.56	0.05
M_1	0.00	12.29	87.63	0.06	23.99	1.73	0.17	31.09	3.40	39.56	0.03	25.15	0.40	5.76	68.58
M_2	0.57	44.01	53.90	0.05	25.08	6.85	1.05	1.87	24.53	40.57	3.69	37.28	3.75	52.00	0.07
M_3	0.02	5.17	94.79	0.05	8.10	0.14	0.05	69.20	1.16	21.30	0.01	13.26	0.19	0.32	86.21
M_4	0.03	99.78	0.18	0.39	91.00	0.47	0.00	0.70	4.85	2.58	0.27	99.16	0.03	0.54	0.00
D_1	0.06	99.11	0.89	0.07	37.77	10.76	0.20	23.71	22.61	4.88	1.22	81.30	0.54	16.95	0.00
D_2	0.01	14.63	85.35	0.00	20.68	0.41	0.16	47.36	2.50	28.90	0.02	18.00	0.20	0.10	81.64
C_1	0.06	3.97	95.97	0.19	5.93	2.50	0.00	40.69	2.35	48.35	0.05	18.40	0.15	2.56	78.80
C_2	0.00	9.35	90.61	0.11	12.48	0.38	0.10	17.67	2.62	66.63	0.03	22.41	0.31	2.33	74.83
C_3	0.04	12.79	86.98	0.05	2.23	1.25	0.00	34.44	1.72	60.32	0.04	21.05	0.12	10.73	67.87
C_4	0.27	5.04	94.67	0.22	2.59	1.01	0.01	20.98	1.46	73.74	0.27	26.14	0.10	2.39	70.99
C_5	0.71	0.66	98.63	1.67	2.87	46.88	0.00	9.16	1.88	37.54	4.01	13.52	0.01	0.27	82.18
W_1	0.06	5.18	94.74	0.08	4.00	1.18	0.02	37.28	1.69	55.75	0.03	16.84	0.10	1.66	81.17
W_2	0.00	9.55	90.44	0.01	12.15	0.26	0.20	26.79	1.96	58.63	0.03	17.94	0.39	0.80	80.58
W_3	0.07	2.36	97.55	0.04	2.48	1.04	0.00	32.67	1.62	62.14	0.05	13.49	0.15	6.56	79.63
W_4	0.01	2.82	97.15	0.02	3.13	1.62	0.02	21.56	1.50	72.14	0.01	19.41	0.11	1.65	78.66
W_5	0.15	0.58	99.27	0.43	1.87	48.16	0.00	8.84	2.26	38.44	1.14	12.17	0.01	0.22	86.45

Table 2: Analyzing the percent of time required for various procedures in each algorithm. The table above presents the values in the *sheared case* (whenever it was necessary).

A Algorithms and complexity

Lemma A.1 *Let $A, B \in \mathbb{Z}[x]$ such that $\mathsf{dg}(A) = m \geq n = \mathsf{dg}(B)$ and $\mathcal{L}(A) = \sigma$, $\mathcal{L}(B) = \tau$. Then $\mathcal{L}(\mathsf{prem}(A,B)) = \mathcal{L}(\mathsf{pquo}(A,B)) = \widetilde{\mathcal{O}}(\delta\tau + \sigma)$, where $\delta = m - n$.*

Proof. Let $A = \sum_{i=0}^{m} a_i x^i$ and $B = \sum_{i=0}^{n} b_i x^i$ and consider

$$M = \begin{bmatrix} b_n & b_{n-1} & \dots & b_0 & & & & \\ & b_n & \dots & b_1 & b_0 & & & \\ & & \ddots & & & \ddots & & \\ & & & b_n & b_{n-1} & \dots & b_1 & b_0 \\ a_m & a_{m-1} & \dots & a_{m-n} & a_{m-n-1} & \dots & a_1 & a_0 \end{bmatrix}$$

The dimension is $(m-n+2) \times (m+1)$. The coefficients of the remainder can be computed as determinants of certain sub-matrices of M [2, 33]. By Bareiss' algorithm, this also holds for pseudo-remainders.

The coefficient of x^{n-1-j} in the remainder, where $n-1 \geq j \geq 0$, is the determinant of matrix M_j, formed by taking the first $m-n+1$ columns of M and column $m-n+2+j$.

Using Hadamard's inequality on the rows

$$|c_j| = |\det M_j| \leq \left((\delta+2)\, 2^{\tau}\right)^{\delta+1} \cdot 2^{\sigma}(\delta+2):$$

The coefficients of $\mathsf{pquo}(A,B)$ can be computed as principal minors of

$$\begin{bmatrix} a_m & a_{m-1} & \dots & a_{m-n} & a_{m-n-1} & \dots & a_1 & a_0 \\ b_n & b_{n-1} & \dots & b_0 & & & & \\ & b_n & \dots & b_1 & b_0 & & & \\ & & \ddots & & & \ddots & & \\ & & & b_n & b_{n-1} & \dots & b_1 & b_0 \end{bmatrix}$$

□

Proof.[of lem. 3.2] Each polynomial in the $\mathbf{SR}(f,g)$ has coefficients of magnitude bounded $2^{c(p+q)\tau}$, for a suitable constant c, assuming $\tau > \lg(d)$. Consider map $\chi : \mathbb{Z}[y] \mapsto \mathbb{Z}$, such that $y \mapsto 2^{\lceil c(p+q)\tau \rceil}$, and let $\phi = \psi \circ \chi : \mathbb{Z}[y_1, y_2 \dots, y_k] \to \mathbb{Z}$. Then $\mathcal{L}(\phi(f)), \mathcal{L}(\phi(g)) \leq c\,(p+q)^k\, d$. Now apply pr. 2.1. □

Now we consider Real root counting. Let $F \in \mathbb{Z}[x,y]$, such that $\mathsf{dg}_x(F) = \mathsf{dg}_y(F) = n_1$ and $\mathcal{L}(F) = \sigma$. Let $\alpha, \beta \in \mathbb{R}_{alg}$, such that $\alpha = (A, [\mathsf{a}_1, \mathsf{a}_2])$ and $\beta = (B, [\mathsf{b}_1, \mathsf{b}_2])$, where $\mathsf{dg}(A), \mathsf{dg}(B) = n_2, \mathcal{L}(A), \mathcal{L}(B) \leq \tau$ and $\mathsf{c} \in \mathbb{Q}$, such that $\mathcal{L}(\mathsf{c}) = \lambda$. Moreover, assume that $n_1^2 = \mathcal{O}(n_2)$, in order to simplify notation. We want to count the number of real roots of $\bar{F} = F(\alpha, y) \in (\mathbb{Z}(\alpha))[y]$ in $(-\infty, +\infty)$, in $(\mathsf{c}, +\infty)$ and in $(\beta, +\infty)$.

We may assume that the leading coefficient of $\bar{F}$ is nonzero. This is wlog since we can easily check it, and/or we can use the good specialization properties of the subresultants [17, 12, 11].

Using Sturm's theorem, e.g [2, 33], the number of real roots of $\bar{F}$ is $\mathtt{VAR}(\mathbf{SR}(\bar{F}, \bar{F}_y; -\infty)) - \mathtt{VAR}(\mathbf{SR}(\bar{F}, \bar{F}_y; +\infty))$. Hence, we have to compute the sequence $\mathbf{SR}(\bar{F}, \bar{F}_y)$ wrt y, and evaluate it on $\pm\infty$, or equivalently to compute the signs of the principal subresultant coefficients, which lie in $\mathbb{Z}(\alpha)$.

The above procedure is equivalent, due to the good specialization properties of subresultants [2, 12], to that of computing the principal subresultant coefficients of $\mathbf{SR}(F, F_y)$, which are polynomials in $\mathbb{Z}[x]$, and to evaluate them over α. I.e the specialization properties assure that we can compute a nominal sequence by considering the bivariate polynomials, and then perform the substitution $x = \alpha$.

The sequence, $\mathbf{sr}$, of the principal subresultant coefficients can be computed in $\widetilde{\mathcal{O}}_B(n_1^4\sigma)$, using cor. 3.5 with $p = q = d = n_1$, and $\tau = \sigma$. The sequence $\mathbf{sr}$, contains $\mathcal{O}(n_1)$ polynomials in $\mathbb{Z}[x]$, each of degree $\mathcal{O}(n_1^2)$ and bitsize $\mathcal{O}(n_1\sigma)$. We compute each one's sign evaluated over α in $\widetilde{\mathcal{O}}_B(n_1^2 n_2 \max\{\tau, n_1\sigma\} + n_2 \min\{n_1^2, n_2\}^2\tau)$ using cor. 2.5 with $p = n_2$, $q = n_1^2$, $\tau_f = \tau$, and $\tau_g = n_1\sigma$. This proves the following:

Lemma A.2 *We count the number of real roots of $\bar{F}$ in $\widetilde{\mathcal{O}}_B(n_1^4 n_2\sigma + n_1^5 n_2\tau)$.*

In order to compute the number of real roots of $\bar{F}$ in $(\beta, +\infty)$, we use again Sturm's theorem. The complexity is dominated by the cost of computing $\mathtt{VAR}(\mathbf{SR}(\bar{F}, \bar{F}_y; \beta))$, which is equivalent to computing $\mathbf{SR}(F, F_y)$ wrt to y, which contains bivariate polynomials, and to compute their signs over (α, β). The cost of computing $\mathbf{SR}(F, F_y)$ is $\widetilde{\mathcal{O}}_B(n_1^5\sigma)$ using cor. 3.4 with $p = q = d = n_1$, and $\tau = \sigma$. The sequence contains $\mathcal{O}(n_1)$ polynomials in $\mathbb{Z}[x, y]$ of degrees $\mathcal{O}(n_1)$ and $\mathcal{O}(n_1^2)$, wrt x and y respectively, and bitsize $\mathcal{O}(n_1\sigma)$. We can compute the sign of each of them evaluated it over (α, β) in $\widetilde{\mathcal{O}}_B(n_1^4 n_2^3 \max\{n_1\sigma, \tau\})$ (th. 3.7). This proves the following:

Lemma A.3 *We can count the number of real roots of $\bar{F}$ in $(\beta, +\infty)$ in $\widetilde{\mathcal{O}}_B(n_1^5 n_2^3 \max\{n_1\sigma, \tau\})$.*

By a more involved analysis, taking into account the difference in the degrees of the bivariate polynomials, we can gain a factor. We omit it for reasons of simplicity and space.

Finally, in order to count the real roots of $\bar{F}$ in $(\mathsf{c}, +\infty)$, it suffices to evaluate the sequence $\mathbf{SR}(F, F_y)$ wrt y on c, thus obtaining polynomials in $\mathbb{Z}[x]$ and compute the signs of these polynomials evaluated over α.

The cost of the evaluation $\mathbf{SR}(F, F_y; \mathsf{c})$ is $\widetilde{\mathcal{O}}_B(n_1^2 \max\{\sigma, \lambda\})$, using cor. 3.6 with $p = q = d = n_1$, $\tau = \sigma$ and $\sigma = \lambda$. The evaluated sequence contains $\mathcal{O}(n_1)$ polynomials in $\mathbb{Z}[x]$, of degree $\mathcal{O}(n_1^2)$ and bitsize $\mathcal{O}(n_1 \max\{\sigma, \lambda\})$. The sign of each one evaluated over α can be compute in $\widetilde{\mathcal{O}}_B(n_1^2 n_2 \max\{\tau, n_1\sigma, n_1\lambda\} + n_1^4 n_2\tau)$, using cor. 2.5 with $p = n_2$, $q = n_1^2$, $\tau_f = \tau$ and $\tau_g = n_1 \max\{\sigma, \lambda\}$. This leads to the following:

Lemma A.4 *We can count the number of real roots of $\bar{F}$ in $(\mathsf{c}, +\infty)$ in $\widetilde{\mathcal{O}}_B(n_1^4 n_2 \max\{n_1\tau, \sigma, \lambda\})$.*

Proof.[of lem. 4.2] Suppose t is such that the degree does not change. It suffices to find, among n^4 integer numbers, one that does not make Δ vanish; note that all candidate values are of size $O(\log n)$.

Algorithm 3: GRID(F, G)

Input: $F, G \in \mathbb{Z}[x, y]$
Output: The real solutions of $F = G = 0$

```
1 R_x ← res_y(F, G)
2 L_x, M_x ← SOLVE(R_x)
3 R_y ← res_x(F, G)
4 L_y, M_y ← SOLVE(R_y)
5 Q ← ∅
6 foreach α ∈ L_x do
7     foreach β ∈ L_y do
8         if SIGN_AT(F, α, β) = 0 ∧ SIGN_AT(G, α, β) = 0 then Q ← ADD(Q, {α, β})
9 RETURN Q
```

We specialize $t \mapsto t_0$ in the system and compute $R = R_{t_0}(x) \in \mathbb{Z}[x])$. The complexity is bounded by $\widetilde{\mathcal{O}}_B(n^2\sigma)$, where $\deg_x R \leq n^2$, and $\mathcal{L}(R) \leq n\sigma$. Computing $R_{red} \in \mathbb{Z}[x])$ reduces to computing $\gcd(R, R')$ in $\widetilde{\mathcal{O}}_B(n^5\sigma)$, and $\deg_x R_{red} = r \leq n^2$, where r is the number of distinct roots of the original system. By pr. 2.4, $\mathcal{L}(R)_{red} = O(n^2 + n\sigma) = \widetilde{\mathcal{O}}_B(n\sigma)$, since we assume $n = \widetilde{\mathcal{O}}_B(\sigma)$. $\Delta \in \mathbb{Z}$ is essentially $\mathtt{res}(R_{red}, R'_{red})$, so we compute it in $\widetilde{\mathcal{O}}_B(n^5\sigma)$, where $\mathcal{L}(\Delta) = \widetilde{\mathcal{O}}_B(n^3\sigma)$. □

Proof.[of th. 4.3] By the previous lemma, $t \in \mathbb{Z}$ is determined with $\mathcal{L}(t) = O(\log n)$ in $\widetilde{\mathcal{O}}_B(n^9\sigma)$. Using this value, we isolate all roots of $R_t(x)$, defined in (1), and determine their multiplicities in $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma^2)$. Let $\rho_j \simeq (R_t(x), [r_j, r'_j])$ be the real roots, for $j = 0, \ldots, r-1$.

By assumption, we have already isolated the roots of the system, denoted by $(\alpha_i, \beta_i) \in [a_i, a'_i] \times [b_i, b'_i]$, where $a_i, a'_i, b_i, b'_i \in \mathbb{Q}$ for $i = 0, \ldots, r-1$. It remains to match each pair (α_i, β_i) to a unique ρ_j by determining function $\phi : \{0, \ldots, r-1\} \to \{0, \ldots, r-1\}$, such that $\phi(i) = j$ iff $(\rho_j, \beta_i) \in \mathbb{R}^2_{alg}$ is a root of the sheared system and $\alpha_i = \rho_j + t\beta_i$.

Let $[c_i, c'_i] = [a_i, a'_i] - t[b_i, b'_i] \in \mathbb{Q}^2$. These intervals may be overlapping. Since the endpoints have bitsize $\mathcal{O}(n^4 + n^3\sigma)$, the intervals $[c_i, c'_i]$ are sorted in $\widetilde{\mathcal{O}}_B(n^7 + n^5\sigma)$. The same complexity bounds the operation of merging this interval list with the list of intervals $[r_j, r'_j]$. If there exist more than one $[c_i, c'_i]$ overlapping with some $[r_j, r'_j]$, some subdivision steps are required so that the intervals reach the bitsize of s_j, where 2^{s_j} bounds the separation distance associated to the j-th root. By pr. 2.6, $\sum_i s_i = \mathcal{O}(n^4 + n^3\sigma)$.

Our analysis resembles that of [10] for proving pr. 2.4. The total number of steps is $\mathcal{O}(\sum_i s_i) = \mathcal{O}(n^4 + n^3\sigma)$, each requiring an evaluation of $R(x)$ over a endpoint of size $\leq s_i$. This evaluation costs $\widetilde{\mathcal{O}}_B(n^4 s_i)$, leading to an overall cost of $\widetilde{\mathcal{O}}_B(n^8 + n^7\sigma)$ per level of the tree of subdivisions. Hence the overall complexity is bounded by $\widetilde{\mathcal{O}}_B(n^{12} + n^{11}\sigma + n^{10}\sigma^2)$.

□

Proof.[of cor. 5.1] Initially we compute the isolating interval representation of the real roots of $P = Q = 0$ in $\widetilde{\mathcal{O}}_B(n^{12} + n^{10}\sigma^2)$, using GRUR_SOLVE. There are $\mathcal{O}(n^2)$ real solutions, which are represented in isolating interval representation, with polynomials of degrees $\mathcal{O}(n^2)$ and bitsize $\mathcal{O}(n^2 + n\sigma)$.

For each real solution, say (α, β), for each polynomial A_i, B_j, C_k we compute the signs of $\text{sign}(A_i(\alpha, \beta))$, $\text{sign}\,(B_i(\alpha, \beta))$ and $\text{sign}\,(C_i(\alpha, \beta))$. Each sign evaluation costs $\widetilde{\mathcal{O}}_B(n^{10}+n^9\sigma)$, using th. 3.7 with $n_1 = n$, $n_2 = n^2$ and $\sigma = n^2 + n\sigma$. In the worst case we need n^2 of them, hence, the cost for all sign evaluations is $\widetilde{\mathcal{O}}_B(\ell n^{12} + \ell\, n^{11}\, \sigma)$. □

B Experiments

System R_1:

$$\begin{aligned} f &= 1 + 2x - 2x^2y - 5xy + x^2 + 3x^2y \\ g &= 2 + 6x - 6x^2y - 11xy + 4x^2 + 5x^3y \end{aligned}$$

System R_2:

$$\begin{aligned} f &= x^3 + 3x^2 + 3x - y^2 + 2y - 2 \\ g &= 2x + y - 3 \end{aligned}$$

System R_3:

$$\begin{aligned} f &= x^3 - 3x^2 - 3xy + 6x + y^3 - 3y^2 + 6y - 5 \\ g &= x + y - 2 \end{aligned}$$

System M_1:

$$\begin{aligned} f &= y^2 - x^2 + x^3 \\ g &= y^2 - x^3 + 2x^2 - x \end{aligned}$$

System M_2:

$$\begin{aligned} f &= x^4 - 2x^2y + y^2 + y^4 - y^3 \\ g &= y - 2x^2 \end{aligned}$$

System M_3:

$$\begin{aligned} f &= x^6 + 3x^4y^2 + 3x^2y^4 + y^6 - 4x^2y^2 \\ g &= y^2 - x^2 + x^3 \end{aligned}$$

System M_4:

$$\begin{aligned} f &= x^9 - y^9 - 1 \\ g &= x^{10} + y^{10} - 1 \end{aligned}$$

System D_1:

$$\begin{aligned} f &= x^4 - y^4 - 1 \\ g &= x^5 + y^5 - 1 \end{aligned}$$

System D_2:

$$f = -312960 - 2640x^2 - 4800xy - 2880y^2 + 58080x + 58560y$$
$$g = -584640 - 20880x^2 + 1740xy + 1740y + 274920x - 59160y$$

System C_1:

$$f = (x^3 + x - 1 - xy + 3y - 3y^2 + y^3)$$
$$(x^4 + 2x^2y^2 - 4x^2 - y^2 + y^4)$$
$$g = \mathrm{diff}(f, y)$$

System C_2:

$$f = y^4 - 6y^2x + x^2 - 4x^2y^2 + 24x^3$$
$$g = \mathrm{diff}(f, y)$$

System C_3:

$$f = ((x-1)^2 + y^2 - 2)((x+1)^2 + y^2 - 2)$$
$$((x-1)^2 + (y+2)^2 - 2)((x+1)^2 + (y+2)^2 - 2)$$
$$g = \mathrm{diff}(f, y)$$

System C_4:

$$f = (x^2 - 2x - 1 + y^2)(x^2 + 2x - 1 + y^2)$$
$$(x^2 - 2x + 3 + y^2 + 4y)$$
$$(100000x^2 + 200000x + 299999 + 100000y^2 + 400000y)$$
$$g = \mathrm{diff}(f, y)$$

System C_5:

$$f = (x^4 + 4x^3 + 6x^2 + 4x + y^4 + 4y^3 + 6y^2 + 4y)$$
$$(x^4 + 4x^3 + 6x^2 + 4x + y^4 - 4y^3 + 6y^2 - 4y)$$
$$(x^4 - 4x^3 + 6x^2 - 4x + y^4 + 4y^3 + 6y^2 + 4y)$$
$$(100000x^4 - 400000x^3 + 600000x^2 - 400000x$$
$$-1 + 100000y^4 - 400000y^3 + 600000y^2 - 400000y)$$
$$g = \mathrm{diff}(f, y)$$

System W_1:

$$f = (x^3 + x - 1 - yx + 3y - 3y^2 + y^3)$$
$$(x^4 + 2y^2x^2 - 4x^2 - y^2 + y^4)$$
$$g = \mathrm{diff}(f, y)$$

System W_2:

$$f = y^4 - 6y^2x + x^2 - 4x^2y^2 + 24x^3$$
$$g = \mathrm{diff}(f, y)$$

System W_3:

$$f = (x^2 - 2x - 1 + y^2)(x^2 + 2x - 1 + y^2)$$
$$(x^2 - 2x + 3 + y^2 + 4y)(x^2 + 2x + 3 + y^2 + 4y)$$
$$g = \mathrm{diff}(f, y)$$

System W_4:

$$\begin{aligned} f = (x^2 - 2x - 1 + y^2)(x^2 + 2x - 1 + y^2) \\ (x^2 - 2x + 3 + y^2 + 4y) \\ (100000x^2 + 200000x + 299999 + 100000y^2 + 400000y) \\ g = \mathrm{diff}(f, y) \end{aligned}$$

System W_5:

$$\begin{aligned} f = (x^4 + 4x^3 + 6x^2 + 4x + y^4 + 4y^3 + 6y^2 + 4y) \\ (x^4 + 4x^3 + 6x^2 + 4x + y^4 - 4y^3 + 6y^2 - 4y) \\ (x^4 - 4x^3 + 6x^2 - 4x + y^4 + 4y^3 + 6y^2 + 4y) \\ (100000x^4 - 400000x^3 + 600000x^2 - 400000x \\ -1 + 100000y^4 - 400000y^3 + 600000y^2 - 400000y) \\ g = \mathrm{diff}(f, y) \end{aligned}$$

system	**total degree**		$\deg_{\mathbf{t}}(\Delta)$	**time (msecs)**
	f	g		
R_1	3	4	18	32
R_2	3	1	6	4
R_3	3	1	0	4
M_1	3	3	16	8
M_2	4	2	4	8
M_3	6	3	110	44
M_4	9	10	5,402	1,402
D_1	4	5	180	176
D_2	2	2	12	8
C_1	7	6	364	1,252
C_2	4	3	28	16
C_3	8	7	174	572
C_4	8	7	308	5,576
C_5	16	15	7,620	2,095,007
W_1	7	6	332	840
W_2	4	3	42	20
W_3	8	7	190	296
W_4	8	7	328	812
W_5	16	15	7,724	1,018,864

Table 3: Time to determine deterministic shear t per test.

	phase of the algorithm	range		median	mean	std dev
		min	max			
GRID	projections	00.00	01.47	00.06	00.21	00.36
	univ. solving	00.58	99.78	12.29	23.78	30.40
	biv. solving	00.18	99.27	90.61	75.86	30.56
	sorting	00.00	01.52	00.02	00.15	00.35
MRUR	projection	00.00	01.67	00.07	00.19	00.37
	univ. solving	01.87	91.00	12.48	17.18	20.48
	StHa seq.	00.14	48.16	01.62	08.27	14.35
	inter. points	00.00	01.05	00.10	00.19	00.30
	filter x-cand	00.56	74.50	26.79	26.42	20.85
	compute K	01.16	24.53	02.50	06.27	07.92
	biv. solving	02.58	73.74	44.16	41.49	21.57
GRUR	projections	00.01	04.01	00.05	00.63	01.16
	univ. solving	06.54	99.16	21.05	28.88	23.39
	inter. points	00.01	03.75	00.19	00.56	00.99
	rational biv.	00.10	55.56	02.39	11.21	18.71
	$\mathbb{R}_{alg}$ biv.	00.00	93.10	78.80	58.46	35.41
	sorting	00.00	03.21	00.09	00.26	00.70

Table 4: Statistics on the performance of the various phases for the our three projection-based algorithms; drawn from tab. 2

C Sample Usage

Our library requires a definition for variable `LIBPATH` which should point on the appropriate path where the source code is stored in your system. On the following, we assume that our library is located under `/opt/AlgebraicLibs/SLV/`. The following is an example for univariate solving:

```
> LIBPATH := "/opt/AlgebraicLibs/SLV/":
> read cat ( LIBPATH, "system.mpl" ):
> f := 3*x^3 - x^2 - 6*x + 2:
> sols := SLV:-solveUnivariate( f ):
> SLV:-display_1 ( sols );
  < x^2-2, [ -93/64, -45/32], -1.414213568 >
  < 3*x-1, [ 1/3, 1/3], 1/3 >
  < x^2-2, [ 45/32, 93/64], 1.414213568 >
```

Note, that the multiplicities of the roots do not appear, although they have been computed. Instead, the third argument of each component in the *printed* list is an approximation of the root. However, whenever possible we provide rational representation of the root.

The following is an example for bivariate solving, where the second root lies in $\mathbb{Z}^2$:

```
> LIBPATH := "/opt/AlgebraicLibs/SLV/":
> read cat ( LIBPATH, "system.mpl" ):
> f := 1+2*x+x^2*y-5*x*y+x^2:
> g := 2*x+y-3:
> bivsols := SLV:-solveGRID ( f, g ):
> SLV:-display_2 ( bivsols );
  < 2*x^2-12*x+1, [ 3, 7], 5.915475965 > ,
  < x^2+6*x-25, [ -2263/256, -35/4], -8.830718995 >

  < x-1, [ 1, 1], 1 > , < x-1, [ 1, 1], 1 >

  < 2*x^2-12*x+1, [ 3/64, 3/32], .8452400565e-1 > ,
  < x^2+6*x-25, [ 23179/8192, 2899/1024], 2.830943108 >
```

Again, just like in the case of univariate solving, the third argument that is printed on the component that describes each algebraic number is an approximation of the number and not the multiplicity of the root. Similarly, one could have used one of the other solvers on the above example by referring to their names, i.e. call the solvers with one of the following commands:

```
> bivsols := SLV:-solveMRUR ( f, g ):
> bivsols := SLV:-solveGRUR ( f, g ):
```

For those interested in the numerical values or rough approximations of the solutions one can get the appropriate output via `display_float_1` and `display_float_2` procedures. Hence, for the above examples we have:

```
> SLV:-display_float_1 ( sols );
  < -1.4142136 >
  <  0.3333333 >
  <  1.4142136 >
> SLV:-display_float_2 ( bivsols );
```

```
  [   5.9154759, -8.8309519, ]
  [   1.0000000,  1.0000000, ]
  [   0.0845241,  2.8309519, ]
```

Consider the list `sols` of $\mathbb{R}_{alg}$ numbers that was returned in the univariate case above; the following are examples on the usage of the `signAt` function provided by our Filtered Kernel[2]:

```
> FK:-signAt( 2*x + 3, sols[1] );
                                1
> FK:-signAt( x^2*y + 2, sols[3], sols[1] );
                               -1
```

Our class on Polynomial Remainder Sequences[3] exports functions allowing the computation of Subresultant and Sturm-Habicht sequences. Let $f, g \in \mathbb{Z}[x, y]$, then you can use *any* of the following commands in order to compute the desired PRS:

```
L := PRS:-StHa ( f, g, y ):
L := PRS:-StHaByDet ( f, g, y ):
L := PRS:-subresPRS ( f, g, y ):
L := PRS:-SubResByDet ( f, g, y ):
```

`PrintPRS` is used for viewing the PRS. For example, let f, g be those from the example on Bivariate Solving above:

```
> L := PRS:-subresPRS ( f, g, y ):
> PRS:-PrintPRS( L );
                  / 2      \                2
                  \x  - 5 x/ y + 1 + 2 x + x
                         y + 2 x - 3
                       3       2
                    2 x  - 14 x  + 13 x - 1
```

Finally, the variance of the above sequence evaluated at $(1, 0)$ can be computed by:

```
> G := PRS:-Eval ( L, 1, 0 );
                          G := [4, -1, 0]
> PRS:-var( G );
                               1
```

[2]Located in file: FK.mpl
[3]Located in file: `PRS.mpl`

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399