Real solving of bivariate polynomial systems

Ioannis Z. Emiris and Elias P. Tsigaridas

Department of Informatics and Telecommunications, National University of Athens, HELLAS
`{emiris,et}@di.uoa.gr`,
`http://www.di.uoa.gr/~{emiris,et}`

Abstract. We propose exact, complete and efficient methods for 2 problems: First, the real solving of systems of two bivariate rational polynomials of arbitrary degree. This means isolating all common real solutions in rational rectangles and calculating the respective multiplicities. Second, the computation of the sign of bivariate polynomials evaluated at two algebraic numbers of arbitrary degree. Our main motivation comes from nonlinear computational geometry and computer-aided design, where bivariate polynomials lie at the inner loop of many algorithms. The methods employed are based on Sturm-Habicht sequences, univariate resultants and rational univariate representation. We have implemented them very carefully, using advanced object-oriented programming techniques, so as to achieve high practical performance. The algorithms are integrated in the public-domain C++ software library SYNAPS, and their efficiency is illustrated by 9 experiments against existing implementations. Our code is faster in most cases; sometimes it is even faster than numerical approaches.

1 Introduction

Our motivation comes from computer-aided geometric design and computational geometry on curved objects, where predicates rely on the real solving of small algebraic systems and on computing the sign of polynomials evaluated at solutions of such systems. These are crucial in software libraries such as ESOLID [15], EXACUS (eg. [12]), and CGAL (eg. [8]). Predicates must be decided exactly in all cases, including degeneracies. We focus on bivariate polynomial systems of arbitrary degree. Efficiency is critical because such systems appear in the inner loop of most algorithms, including those for computing the arrangement of algebraic curves or surfaces, the Voronoi diagrams of curved objects, e.g. [6, 12] and kinetic data-structures ([11]).

Solving polynomial systems in a real field is an active area of research. There are several algorithms that tackle this problem, cf. e.g. [1, 25] and the references therein. Every method designed for the real-solving of algebraic systems could be compared against ours. We focus on those that, in the best of our knowledge, have efficient implementations, enumerated below, and perform experiments against them. Note that we aim at fully accurate computation, in the sense that we avoid any numerical computation and thus we do not compare against software based on homotopies, interval arithmetic or Newton-like methods. Besides our software, only GbRs achieves this goal completely, among the examined implementations. Of course many generalizations of Gröbner bases can lead to solution of this problem, but we chose GbRs and normal forms (solver newmac below) as representatives of them. We also do not test against computer algebra systems, like maple or reduce, or against algorithms for the related but different question on parametric polynomials ([23]).

Our methods are exact, in the sense that they provide rational isolating rectangles for all common roots and calculate their multiplicity. We concentrate on solvers that output this representation. Our methods are also complete, since they can handle all cases, including degeneracies. And they are efficient as testified by their implementation and experiments.

In an earlier paper ([7]), in order to solve *quadratic* bivariate systems, without assuming generic position, we precomputed resultants and static Sturm-Habicht sequences in two variables ([9]) and we combined the rational isolating points with a simple version of rational univariate representation. Here we generalize that approach and use Sturm-Habicht sequences, in a dynamic setting since the polynomial degree is not bounded. Our approach is based on projecting the roots along the two coordinates axes using univariate resultants. We again combine the rational isolating points, computed around the resultants' roots, using a specialized version of rational univariate representation, in order to lift them to two dimensions.

Additionally we compute the ordinates of the solutions as algebraic numbers in isolating interval representation, avoiding computations of minimal polynomials. This is important since further computations with the solutions of a system is often required. For example in [13], where computations of Eggers singularities and Milnor numbers are required, or [24], where projections of the roots on three lines are computed, as well as interval refinement in order to compute the critical points of a curve. Our approach allows us to compute the sign of a bivariate polynomial function evaluated over algebraic numbers, all of arbitrary degree. Our approach is similar to [22] and can be easily extended to polynomials with an arbitrary number of variables. However, our approach is more efficient since we use Sturm-Habicht sequences.

The main contribution of this paper is a package for solving bivariate systems and computing the sign of bivariate polynomials evaluated at algebraic numbers, as part of the synaps software library [5], which is developed in C++. For this we use modern object-oriented programming techniques, such as partial specialization and traits classes, so as to achieve high performance in practice.

We performed experiments against existing methods and implementations. mapc [14] had used Sturm sequences, but did not handle degeneracies. Since mapc is no longer maintained, we do not compare against it. In our tests, we compared against other solvers in synaps [1], namely newmac, which is based on normal forms [19], sth, which also uses Sturm-Habicht sequences but uses a double approximation in order to compute the second coordinate of the solutions

[1] www-sop.inria.fr/galaad/logiciels

and is based on the work of [10], and RES, which is based on computing the generalized eigenvalues of a Bézoutian matrix [2]. Additionally, we test against GBRS [2], through its MAPLE interface, which uses Gröbner bases and rational univariate representation [21].

A more recent work is the one of [16], where sparse resultant, rational univariate representation and certified numerical approximations are used so as to solve polynomial systems, with arbitrary number of variables and equations. Their code is not yet freely available. From the running times that they provide it seems that our approach for bivariate systems is faster.

We show that our code compares favourably with other software on most instances. Sometimes it is even faster than methods using some numerical computation; such methods may compromise the accuracy of the output. This shows that for specific instances of real solving, namely when the problem dimension is small, a careful implementation of the Sturm-Habicht approach can be very competitive and even the method of choice.

This paper is organized as follows. The next two sections survey some necessary notions on root multiplicity, and on the theory of Sturm-Habicht sequences. Section 4 presents computations with real algebraic numbers. Section 5 presents our two variants for solving bivariate polynomial system. The following section describes our implementation and experiments. We conclude with open questions.

2 Root multiplicity

The results of this section can be found for example in [3, 1, 25]. We follow the approach and the terminology of [13] and [2].

In what follows $\mathbf{F}$ is a commutative field of characteristic zero and $\overline{\mathbf{F}}$ its algebraic closure. Typically $\mathbf{F} = \mathbb{Q}$ and $\overline{\mathbf{F}} = \overline{\mathbb{Q}}$. Moreover, f, g are bivariate polynomials in $\mathbf{F}[X, Y]$ and $\mathcal{C}_f$ and $\mathcal{C}_g$ are the corresponding affine algebraic plane curves. By $\deg(f)$ we denote the total degree of f, while by $\deg_X(f)$ (respectively $\deg_Y(f)$) denotes the degree of f considered as a univariate polynomial in X (resp. Y) with coefficients in $\mathbf{F}[Y]$ (resp. $\mathbf{F}[X]$).

Let $f, g \in \mathbf{F}[X, Y]$ be two coprime polynomials and $\mathcal{C}_f$, $\mathcal{C}_g$ be their corresponding affine algebraic plane curves, over the field $\overline{\mathbf{F}}$, defined by the equations $(\Sigma) : f(X, Y) = g(X, Y) = 0$. Let $\mathcal{I} = < f, g >$ the ideal that they generate in $\mathbf{F}[X, Y]$ and so the associated quotient ring is $\mathcal{A} = \overline{\mathbf{F}}[X, Y]/\mathcal{I}$. Let the distinct intersection points, which are the distinct roots of (Σ), be $\mathcal{C}_f \cap \mathcal{C}_g = \{\zeta_i = (\alpha_i, \beta_i)\}_{1 \leq i \leq r}$, where $\zeta_i \in \overline{\mathbf{F}}^2$.

The multiplicity of a point ζ_i is

$$\mathrm{mult}(\zeta_i : \mathcal{C}_f \cap \mathcal{C}_g) = \dim_{\overline{\mathbf{F}}} \mathcal{A}_{\zeta_i} < \infty$$

where $\mathcal{A}_{\zeta_i}$ is the local ring obtained by localizing $\mathcal{A}$ at the maximal ideal $\mathcal{I} =< X - \alpha_i, Y - \beta_i >$ ([3]).

[2] http://fgbrs.lip6.fr

If $\mathcal{A}_{\zeta_i}$ is a finite dimensional vector space over $\overline{\mathbf{F}}$, then $\zeta_i = (\alpha_i, \beta_i)$ is an isolated zero of $\mathcal{I}$ and its multiplicity is called the intersection number of the two curves. The finite $\overline{\mathbf{F}}$-algebra $\mathcal{A}$ can be decomposed as a direct sum $\mathcal{A} = \mathcal{A}_{\zeta_1} \oplus \mathcal{A}_{\zeta_2} \oplus \cdots \oplus \mathcal{A}_{\zeta_r}$ and thus $\dim_{\overline{\mathbf{F}}} \mathcal{A} = \sum_{i=1}^{r} \mathrm{mult}(\zeta_i : \mathcal{C}_f \cap \mathcal{C}_g)$.

A polynomial $f \in \mathbf{F}[X,Y]$ or a curve $\mathcal{C}_f$, is called $y-$regular if $\deg(f) = \deg_Y(f)$.

Proposition 1. *Let $f, g \in \mathbf{F}[X,Y]$ be two coprime curves, and let* $\mathrm{p} \in \overline{\mathbf{F}}^2$ *be a point. Then*

$$\mathrm{mult}\,(\mathrm{p} : fg) \geq \mathrm{mult}\,(\mathrm{p} : f)\,\mathrm{mult}\,(\mathrm{p} : g)$$

where equality holds if and only if $\mathcal{C}_f$ and $\mathcal{C}_g$ have no common tangents at p.

Real-solving of (Σ) is equivalent to finding the intersections of $\mathcal{C}_f$ and $\mathcal{C}_g$ in the real plane. We assume that $\mathcal{C}_f$ and $\mathcal{C}_g$ are $y-$regular and that (Σ) is in generic position, meaning that every solution has a distinct $x-$coordinate. This is without loss of generality, since we can achieve this by a linear change of coordinates. A generic linear transformation (shear) of coordinates puts the points of $\mathcal{C}_g \cap \mathcal{C}_g$ in one to one correspondence with roots of the resultant of f and g with respect to Y.

3 Sturm–Habicht sequences

In this section we present some results for Sturm-Habicht sequences. For more information the reader may refer to [1, 10, 25].

Let $P, Q \in \mathbb{Z}[X]$ such that $\deg(P) = p$ and $\deg Q = q$ and $P = \sum_{k=0}^{p} a_k X^k, Q = \sum_{k=0}^{q} b_k X^k$. If $i \in \{0, \ldots, \inf(p,q)\}$ we define the polynomial subresultant associated to P and Q of index i, as follows:

$$\mathbf{Sres}_i(P,Q) = \sum_{j=0}^{i} M_i^j x^j$$

where every M_i^j is the determinant of the matrix built with columns $1, 2, \ldots, p+q-2i-1$ and $p+q-i-j$ in the matrix:

$$m_i = \begin{pmatrix} a_p & \ldots & a_0 & & \\ & \ddots & & \ddots & \\ & & a_p & \ldots & a_0 \\ b_p & \ldots & b_0 & & \\ & \ddots & & \ddots & \\ & & b_p & \ldots & b_0 \end{pmatrix}$$

where the coefficients of P and Q are repeated $q-i$ and $p-i$ times respectively. The determinant $M_j^i(P,Q)$ is called the $i-$th principal subresultant coefficient and denoted by $\mathbf{sres}_i$.

Definition 1. *Let P be polynomials in $\mathbb{Z}[X]$ with $p = \deg(P)$. If we write*

$$\delta_k = (-1)^{\frac{k(k+1)}{2}}$$

for every integer k, the Sturm-Habicht sequence associated to P is defined as in the list of polynomials $\{\mathbf{StHa}_j(P)\}_{j=0,\dots,p}$, where $\mathbf{StHa}_p(P) = P, \mathbf{StHa}_{p-1}(P) = P'$, and for every $j \in \{0, \dots, p-2\}$:

$$\mathbf{StHa}_j(P) = \delta_{p-j-1}\mathbf{Sres}_j(P, P')$$

For every $j \in \{0, \dots, p\}$ the principal $j-th$ Sturm-Habicht coefficient is defined as:

$$\mathbf{stha}_j(P) = \underset{j}{\operatorname{coeff}}(\mathbf{StHa}_j(P)),$$

i.e. the coefficient of x^j in the polynomial $\mathbf{StHa}_j(P)$.

It is important to mention that the polynomial $\mathbf{stha}_0$, modulo its sign is the discriminant of P.

Moreover the greatest common divisor of P and P' is obtained as a by-product of the Sturm-Habicht sequence, together with the following equivalence:

$$\mathbf{StHa}_i(P) = \gcd(P, P') \Leftrightarrow \begin{cases} \mathbf{stha}_0(P) = \cdots = \mathbf{stha}_{i-1}(P) = 0 \\ \mathbf{stha}_i(P) \neq 0 \end{cases}$$

The Sturm-Habicht sequence has very nice specialization properties. Let P and Q be two polynomials with parametric coefficients, such that their degree does not change after a specialization in the parameters. If we compute their Sturm-Habicht sequence before we specialize the coefficients, the obtained sequence is guaranteed to be valid under every specialization. We use this property so as to compute such a sequence for polynomials in $\mathbb{Z}[X, Y]$, regarding them either as polynomials in $(\mathbb{Z}[X])[Y]$ or in $(\mathbb{Z}[Y])[X]$. The last polynomial in the sequence is the resultant with respect to X or Y, respectively.

Theorem 1. *[1, 10] Let f, g square-free and coprime polynomials, such that $\mathcal{C}_f$ and $\mathcal{C}_g$ are in generic position. If*

$$H_j(X, Y) = \mathbf{StHa}_j(f, g) = h_j(X)Y^j + h_{j,j-1}(X)Y^{j-1} + \cdots + h_{j,0}(X)$$

then if $\zeta = (\alpha, \beta) \in \mathcal{C}_f \cap \mathcal{C}_g$ then there exists k, such that

$$h_0(\alpha) = \cdots = h_{k-1}(\alpha) = 0, \quad h_k(\alpha) \neq 0, \quad \beta = -\frac{1}{k}\frac{h_{k,k-1}(\alpha)}{h_k(\alpha)}$$

4 Real algebraic numbers and sign evaluation

The real algebraic numbers, i.e. those real numbers that satisfy a polynomial equation with integer coefficients, form a real closed field denoted by $\mathbb{R}_{alg} = \overline{\mathbb{Q}}$. From all integer polynomials that have an algebraic number α as root, the one with the minimum degree is called *minimal polynomial.* The minimal polynomial is unique, primitive and irreducible ([25]). In our approach, since we use Sturm-Habicht sequences, it suffices to deal with algebraic numbers, as roots of any square-free polynomial and not as roots of their minimal ones.

In order to represent a real algebraic number we chose the *isolating interval representation.*

Definition 2. *The isolating-interval representation of real algebraic number $\alpha \in \mathbf{F}$ is $\alpha \cong (P(X), I)$, where $P(X) \in \mathbf{D}[X]$ is square-free and $P(\alpha) = 0$, $I = [a, b]$, $a, b, \in \mathbf{Q}$ and P has no other root in I.*

Let $\overline{P}$ denote the Sturm-Habicht sequence of P and P'. For a Sturm-Habicht sequence $\overline{P}$, $V_{\overline{P}}(p)$ denotes the number of sign variations of the evaluation of the sequence at p. By $V_{P,Q}(a)$ we denote the sign variations of the Sturm-Habicht sequence of P and Q, evaluated over a.

Theorem 2. *[25] Let $P, Q \in \mathbf{D}[x]$ be relatively prime polynomials and P square-free. If $a < b$ are both non-roots of P and γ ranges over the roots of P in $[a, b]$, then*

$$V_{P,Q}[a, b] := V_{P,Q}(a) - V_{P,Q}(b) = \sum_{\gamma} \operatorname{sign}(P^{'}(\gamma)Q(\gamma)).$$

where $P^{'}$ is the derivative of P.

We can use Sturm-Habicht sequences in order to find the sign of a univariate polynomial, evaluated over a real algebraic number (cf. [9] for degree ≤ 4).

Corollary 1. *Let $Q(X) \in \mathbf{D}[X]$ and a real algebraic number where $\alpha \cong (P, [a, b])$. By th. 2,* $\operatorname{sign}(Q(\alpha)) = \operatorname{sign}(V_{P,Q}[a, b] \cdot Q^{'}(\alpha))$.

Corollary 2. *Th. 2 holds if in place of Q we use $R = \operatorname{prem}(Q, P)$, where $\operatorname{prem}(Q, P)$, stands for the pseudo-remainder of Q divided by P.*

Corollary 3. *Using th. 2 we can compare two real algebraic numbers in isolating interval representation.*

Proof. Let two algebraic numbers $\gamma_1 \cong (P_1(x), I_1)$ and $\gamma_2 \cong (P_2(x), I_2)$ where $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$. Let $J = I_1 \cap I_2$. When $J = \emptyset$, or only one of γ_1 and γ_2 belong to J, we can easily order the 2 algebraic numbers. If $\gamma_1, \gamma_2 \in J$, then $\gamma_1 \geq \gamma_2 \Leftrightarrow P_2(\gamma_1) \cdot P_2^{'}(\gamma_2) \geq 0$. We can easily obtain the sign of $P_2^{'}(\gamma_2)$, and from th. 2, we obtain the sign of $P_2(\gamma_1)$. □

The previous tools suffice to compute the sign of a bivariate polynomial function evaluated over two algebraic numbers. Consider $F \in \mathbf{D}[X, Y]$ and $\alpha \cong (A(x), I_1)$ and $\beta \cong (B(X), I_2)$ where $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$. We wish to compute the sign of $F(\alpha, \beta)$.

– Consider F as a univariate polynomial with respect to X and compute the Sturm-Habicht sequence of A and F. Note that the polynomials in the sequence are bivariate.
– Taking advantage of the good specialization properties of Sturm-Habicht sequences, specialize X in the sequence by a_1 and b_1, thus producing two sequences, that contain univariate polynomials.
– For each sequence, for every polynomial in each, compute its sign evaluated at β.
– Finally, count the sign variations for each sequence and the required sign is the difference of these sign variations.

We can extend this approach to polynomials with arbitrary numbers of variables, similar to [22]. However the usage of Sturm-Habicht sequences, instead of generalized Sturm sequences, improves both the theoretical (cf. [1]) and the practical complexity (cf. [4, 25]).

5 Two variants of bivariate real solving

We can make use of th.1, following [10], so as to compute the solution of bivariate polynomial systems. We consider polynomials $f, g \in \mathbb{Q}[X, Y]$, such that $\mathcal{C}_f, \mathcal{C}_g$ are in generic position and we compute the resultant of f, g with respect to Y, which is a polynomial in X. The real solutions of the polynomial correspond to the $x-$ coordinates of the solution of the system. Then, using th.1, we lift these solutions in order to determine the $y-$coordinates, as a rational univariate function evaluated over an algebraic number.

Even though the previous approach is straightforward, it has one main disadvantage. The y-coordinates are computed implicitly. If this is all that we want then this is not a problem. However in most cases we want to further manipulate the solutions of the system, i.e. to compare two $y-$coordinates or to count the number of branches of each curve above or below this ordinate. Of course we can always find the minimal polynomial of these algebraic numbers, but this is quite expensive. Thus we chose an alternatively way.

We compute the resultant, using the Sturm-Habicht Sequence, both with respect to Y and X, R_x and R_y respectively. We solve the univariate polynomials R_x and R_y using Sturm sequences (a faster solver like the one in [20], may also be used). Let $\alpha_1 < \cdots < \alpha_k$ and $\beta_1 < \cdots < \beta_l$ be the real roots of R_x and R_y, respectively. For the real roots of R_y we compute rational intermediate points, $q_0 < \beta_1 < q_1 < \cdots < q_{l-1} < \beta_l < q_l$ where $q_j \in \mathbb{Q}, 0 \leq j \leq l$. We can easily compute the intermediate points, since the algebraic numbers are in isolating interval representation.

For every root $\alpha_i, 1 \leq i \leq l$, using th.1, we compute a rational univariate representation of the corresponding y-coordinate, which is without loss of generality, of the form $\gamma_i = \frac{A(\alpha_i)}{B(\alpha_i)}$. Since have already computed the real solutions of R_y, it suffices to determine to which β_j, γ_i equals to, that is to find an index

j such that

$$q_j < \frac{A(\alpha_i)}{B(\alpha_i)} < q_{j+1}$$

or, if we assume that $A(\alpha_i) > 0$, this can be checked using cor. 1, then

$$q_j A(\alpha_i) < B(\alpha_i) < q_{j+1} A(\alpha_i)$$

Actually what we really want is to determine the sign of univariate polynomials of the form $U(X) = q_j A(X) - B(X)$ evaluated over the real algebraic numbers that are solutions of $R_x = 0$. This can be done with cor. 1.

However the previous approach works only with the assumption of generic position. This is without loss of generality, since we can apply a transformation of the form $(X, Y) \mapsto (X + aY, Y)$, where a is a random number before the execution of the algorithm, or we can detect non-generic position during the execution ([10, 1]), then apply a transformation of the form $(X, Y) \mapsto (X + Y, Y)$ and start the algorithm recursively. However if such a transformation is performed then it is a very hard computational task to apply the inverse transformation so as to represent the solutions to the original coordinate system. Moreover such transformations destroy the sparsity of the system.

In order to overcome such barriers we suggest one more variant for bivariate polynomial system solving. As before we compute the two resultants R_x and R_y and their real solutions α_j and β_j, $1 \leq i \leq k$, $1 \leq j \leq k$, respectively. Then for every pair (a_i, b_j) we test if both f and g vanish. If so, then this pair is the solution. Actually, we do not need to test every pair, since we can take into account the multiplicities of α_i and β_j. The sign of a bivariate polynomial evaluated over two algebraic numbers can be computed using the results of the previous section. This variant is more generic than the previous one, but as one can easily imagine, in most of the cases it is clearly slower. However, it is useful, since a combination of both variants can be used, for example when a non-generic position is detected.

6 Implementation and experimentation

6.1 Implementation

We have implemented a software package in C++, as part of library SYNAPS [5, 18] inside the `namespace ALGEBRAIC`, for dealing with algebraic numbers and bivariate polynomial system solving, which is optimized for small degree. Our implementation is generic in the sense that it can be used with any number type and any polynomial class that supports elementary operations and evaluations and can handle all degenerate cases. We used various advanced C++ programming techniques, such as template specialization, traits classes for number types, etc. Additionally we have precomputed various quantities, and factor several common expressions so as to minimize the computational effort.

In what follows `root_of<RT>` is a class that represents real algebraic numbers, computed as roots of polynomial in isolating interval representation. `UPoly<RT>`

is a class for univariate polynomial while `BPoly<RT>` is a class for multivariate polynomials (for our approach we need only the bivariate ones). All classes are parametrized by a ring number type (`RT`).

We present a portion of the functionality that we provide. Actually the presentation is very abstract since various parameters can be determined. For example the univariate solver that can be used and operations between algebraic numbers are not presented here (see [18]). The full description of the functionality, as well as various design and optimization techniques, will be part of a future presentation. The interesting reader may refer to the documentation of SYNAPS for additional details.

– `Seq<root_of<RT> > solve(UPoly<RT>` f`)`
  Solves a univariate polynomial and returns a sorted sequence of real algebraic numbers. For degree up to 4 all the quantities are precomputed using the algorithms of [9]. For higher degrees we use an algorithm for real root isolation based on Sturm-Habicht sequences [4, 1, 25].
– `int compare(root_of<RT>` α`, root_of<RT>` β`)`
  Compares two algebraic numbers and returns $-1, 0$ or $+1$, depending on the order. For degree up to 4 we use precomputed sequences ([9]). For higher degree we use dynamic (that is computed on the fly) Sturm-Habicht sequences. We use cor. 3 in order to compare them.
– `int sign_at(UPoly<RT>` f`, root_of<RT>` α`)`
  Computes the sign of a univariate polynomial evaluated over an algebraic number returns $-1, 0$ or $+1$. Both the polynomial and the real algebraic number can be of arbitrary degree. We implemented this by using cor. 3.
– `int sign_at(BPoly<RT>` f`, root_of<RT>` γ_x`, root_of<RT> <RT>` γ_y`)`
  Computes the sign of a bivariate polynomial evaluated over two real algebraic numbers and returns $-1, 0$ or $+1$. The total degree of the bivariate polynomial, as well as the degree of the algebraic number may be arbitrary. We use cascaded Sturm-Habicht sequences. For total degree of the bivariate polynomial up to 2 and if at least one of the algebraic numbers is of degree less than 5 we use precomputed sequences ([9]).
– `Seq < pair<root_of<RT> > > solve(BPoly<RT>` f_3`, BPoly<RT>` f_2`)`
  Computes the real solutions of a bivariate polynomial system and returns a sequence of pairs of real algebraic numbers sorted lexicographically. If the total degree of the polynomials is less than 3, we use precomputed sequences ([9]). In all the other cases we use the algorithm described in sec. 5.

6.2 Experiments

In order to test our implementation we solved the following systems:

$$(R_1)\begin{cases} 1 + 2X - 2X^2Y - 5XY + X^2 + 3X^2Y \quad = 0 \\ 2 + 6X - 6X^2Y - 11XY + 4X^2 + 5X^3Y = 0 \end{cases}$$
$$(R_2)\begin{cases} X^3 + 3X^2 + 3X - Y^2 + 2Y - 2 = 0 \\ 2X + Y - 3 = 0 \end{cases}$$
$$(R_3)\begin{cases} X^3 - 3X^2 - 3XY + 6X + Y^3 - 3Y^2 + 6Y - 5 = 0 \\ X + Y - 2 = 0 \end{cases}$$

$$(M_1)\begin{cases} Y^2 - X^2 + X^3 = 0 \\ Y^2 - X^3 + 2X^2 - X = 0 \end{cases}$$
$$(M_2)\begin{cases} X^4 - 2X^2Y + Y^2 + Y^4 - Y^3 = 0 \\ Y - 2X^2 = 0 \end{cases}$$
$$(M_3)\begin{cases} X^6 + 3X^4Y^2 + 3X^2Y^4 + Y^6 - 4X^2Y^2 = 0 \\ Y^2 - X^2 + X^3 = 0 \end{cases}$$
$$(M_4)\begin{cases} X^9 - Y^9 - 1 = 0 \\ X^{10} + Y^{10} - 1 = 0 \end{cases}$$
$$(D_1)\begin{cases} X^4 - Y^4 - 1 = 0 \\ X^5 + Y^5 - 1 = 0 \end{cases}$$
$$(D_1)\begin{cases} -312960 - 2640X^2 - 4800XY - 2880Y^2 + 58080X + 58560Y = 0 \\ -584640 - 20880X^2 + 1740XY + 1740Y + 274920X - 59160Y = 0 \end{cases}$$

where systems $R_{\{1,2,3\}}$ are from [16] and systems $M_{\{1,2,3,4\}}$ are from [2]. The results are on table 6.2, where times presented are in `msec` and are the average of 100 runs. We performed all tests on a 2.6GHz Pentium with 512MB memory, running Linux, with kernel version 2.6.10. We compiled the programs with `g++`, v. 3.3.5, with option `-O3`.

We test against NEWMAC [19]. It is a general purpose polynomial system solver. STH, in SYNAPS, is based on Sturm-Habicht sequences and subresultants, following [10]. RES is a bivariate polynomial solver based on the Bézoutian matrix and LAPACK [2]. For GBRS [21] we use its MAPLE interface with 10 digits accuracy, since the source code is not available. S^2 refers to our solver using only the `sign_at` functions, S^2-RUR is our algorithm based the on rational univariate representation.

We have to emphasize that our approach is exact, i.e. it outputs isolating boxes with rational endpoints containing a unique root whose multiplicity is also calculated. This is not the case for STH and RES. STH, uses a double approximation in order to compute the ordinate of the solution. RES works only with doubles, since it has to compute generalized eigenvalues and eigenvectors. These approximations is the reason why they both failed on some tests. NEWMAC also relies on the computation of eigenvalues but in addition computes all the complex solutions of the system.

S^2 is competitive on all data sets, while S^2-RUR is almost always faster than any other solver, even from those that they use `double` arithmetic. However the system of interest is system M_4. Note that this system is very sparse. Sturm-Habicht sequences do not take advantage of the sparsity of a problem. This particular system, is not in generic position, so a linear transformation is applied. Then, at one hand, the sparsity is destroyed and, on the other, the Sturm-Habicht sequences become quite long. We noticed that most of the time is spent for the real solution of the two resultants. This is a strong indication, that a more sophisticated solver for univariate polynomials, like the one in [20], must be adopted.

As for the approach of [16], they quote that, on a faster machine with 3GHz CPU, the timings for solving system R_1, R_2 and R_3 are 2590, 86.5 and 103 msec respectively. So it seems that our approach for bivariate systems is faster. Of course all these are only indications and a more subtle study is required.

`msec`	R_1	R_2	R_3	M_1	M_2	M_3	M_4	D_1	D_2
S^2	10	1	1	2	3	433	5010	50	1
S^2-RUR	1	1	0.1	1	1	44	1010	11	1
NEWMAC	6	2	3	3	3	20	1020	20	20
RES	0.3	0.3	0.6	0.6	01.2	8.4	150	-	0.5
STH	1	0.2	0.2	0.5	0.4	1.3	-	280	0.4
GbRs	24	22	21	18	23	28	25	25	27

Table 1. Experiments on bivariate system solving

7 Conclusions and future work

We are currently working on better bounds on the bit complexity of our algorithms. We expect this investigation to identify possible bottlenecks and lead to better performance in practice. We plan to apply our tools in computing the topology of algebraic curves in 2D and 3D, as well as the topology of surfaces in 3D. Other possible approaches, to be implemented and compared at a practical level, include the adoption of fast Cauchy-index computations ([17]) and Thom's encoding ([1]). Last but not least, we intend to use arithmetic filtering to handle cases that are far from degenerate, so as to improve the speed of our software for generic inputs.

Acknowledgements: The second author thanks Bernard Mourrain, for his continuing help during the implementation of the package. Both authors acknowledge partial support by Kapodistrias, project 70/4/6452 of the Research Council of National University of Athens, by PYTHAGORAS, project 70/3/7392 under the EPEAEK program funded by the Greek Ministry of Educational Affairs and EU, and IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes).

References

1. S. Basu, R. Pollack, and M-F.Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics.* Springer-Verlag, 2003.
2. L. Busé, B. Mourrain, and H. Khalil. Resultant-based methods for curves intersection problems. Manuscript, 2005.
3. D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry.* Number 185 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1998.
4. J.H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra.* Academic Press, London, 1988.

5. G. Dos Reis, B. Mourrain, R. Rouillier, and P. Trébuchet. An environment for symbolic and numeric computation. In *Proc. Int. Conf. Math. Software*, World Scientific, pages 239–249, 2002.
6. L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics. In *Proc. ACM SoCG*, pages 246–255., June 2003.
7. I. Z. Emiris and E. P. Tsigaridas. Real algebraic numbers and polynomial systems of small degree. manuscript, 2005. (www.di.uoa.gr/˜et)
8. I.Z. Emiris, A.V. Kakargias, M. Teillaud, E.P. Tsigaridas, and S. Pion. Towards an open curved kernel. In *Proc. ACM SoCG*pages 438–446, New York, 2004.
9. I.Z. Emiris and E.P. Tsigaridas. Computing with real algebraic numbers of small degree. In *Proc. ESA*, LNCS, pages 652–663. Springer Verlag, 2004.
10. L. Gonzalez-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Comp. Aided Geom. Design*, 19(9):719–743, 2002.
11. L.J. Guibas, M.I. Karavelas, and D. Russel. A computational framework for handling motion. In *Proc. 6th Work. Algor. Engin. & Experim. (ALENEX)*, 2004.
12. M. Hemmer, E. Schömer, and N. Wolpert. Computing a 3-dimensional cell in an arrangement of quadrics: Exactly and actually! In *Proc. Annual ACM Symp. Comput. Geometry*, pages 264–273, 2001.
13. M. El Kahoui. Computing with algebraic curves in generic position. submitted, 2005.(www.mpi-sb.mpg.de/˜elkahoui)
14. J. Keyser, T. Culver, D. Manocha, and S .Krishnan. MAPC: A library for efficient and exact manipulation of algebraic points and curves. In *Proc. Annual ACM Symp. Comput. Geometry*, pages 360–369, New York, N.Y., June 1999. ACM Press.
15. J. Keyser, T. Culver, D. Manocha, and S. Krishnan. ESOLID: A system for exact boundary evaluation. *Comp. Aided Design*, 36(2):175–193, 2004.
16. J. Keyser, K. Ouchi, and M. Rojas. The Exact Rational Univariate Representation for Detecting Degeneracies. In *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. AMS Press, 2004. to appear.
17. T. Lickteig and M.-F. Roy. Semi-algebraic complexity of quotients and sign determination of remainders. *J. Complexity*, 12(4):545–571, December 1996.
18. B. Mourrain, J. P. Pavone, P. Trébuchet, and E. Tsigaridas. SYNAPS, a library for symbolic-numeric computation. In *8th Int. Symp. on Effective Methods in Algebraic Geometry, MEGA*, Italy, May 2005. Software presentation, to appear.
19. B. Mourrain and Ph. Trébuchet. Algebraic methods for numerical solving. In *Proc. of the 3rd International Workshop on Symbolic and Numeric Algorithms for Scientific Computing'01 (Timisoara, Romania)*, pages 42–57, 2002.
20. B. Mourrain, M. Vrahatis, and J.C. Yakoubsohn. On the complexity of isolating real roots and computing with certainty the topological degree. *J. Complexity*, 18(2), 2002.
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. Signs of algebraic numbers. *Computers and Mathematics*, pages 131–134, 1989.
23. V. Weispfenning. Solving parametric polynomial equations and inequalities by symbolic algorithms. In Proc. *Computer Algebra in Science and Engineering* World Scientific, 1995
24. N. Wolpert and Seidel. On the Exact Computation of the Topology of Real Algebraic Curves. In *Symposium of Computational Geometry*. ACM, 2005. to appear.
25. C.K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.