Applied Mathematics and Computation 219 (2013) 7533–7541

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Parallel computation of real solving bivariate polynomial systems by zero-matching method

Xiaolin Qin [a,b,c,*], Yong Feng [a], Jingwei Chen [a], Jingzhong Zhang [a]

[a] Laboratory for Automated Reasoning and Programming, Chengdu Institute of Computer Applications, CAS, Chengdu 610041, PR China
[b] Department of Mathematics, Sichuan University, Chengdu 610064, PR China
[c] Graduate School of the Chinese Academy of Sciences, Beijing 100049, PR China

ARTICLE INFO

Keywords:
Bivariate polynomial system
Zero-matching method
Real roots
Symbolic-numerical computation
Parallel computation

ABSTRACT

We present a new algorithm for solving the real roots of a bivariate polynomial system $\Sigma = \{f(x,y), g(x,y)\}$ with a finite number of solutions by using a zero-matching method. The method is based on a lower bound for the bivariate polynomial system when the system is non-zero. Moreover, the multiplicities of the roots of $\Sigma = 0$ can be obtained by the associated quotient ring technique and a given neighborhood. From this approach, the parallelization of the method arises naturally. By using a multidimensional matching method this principle can be generalized to the multivariate equation systems.

1. Introduction

Considering the following system:

$$\Sigma = \{f(x,y), g(x,y)\}, \tag{1}$$

we assume that $f(x,y), g(x,y) \in \mathbb{Q}[x,y]$, where $\mathbb{Q}$ is the field of rational numbers. We call the **zero-dimension** if the bivariate polynomial system (1) has a finite number of solutions.

Real solving bivariate polynomial system in a real field is an active area of research. It is equivalent to finding the intersections of $f(x,y)$ and $g(x,y)$ in the real plane. The problem is closely related to computing the topology of a plane real algebraic curve and other important operations in non-linear computational geometry and Computer-Aided Geometric Design [1,10,12,14,16,20]. Another field of applications is the quantifier elimination [7,18]. There are several algorithms that tackle this problem such as the Gröber basis method [21,24], the resultant method [13,27], the characteristic set method [4], and the subdivision method [3,23]. The first three methods are symbolic manipulation techniques, which are principally exact and stable. However, they have the disadvantage of intermediate expression swell. They can only be used to solve small- and middle-scale problems in practice. The last one is a hybrid symbolic and numerical method. Nevertheless, its procedure is very complicated. In this paper, we propose an efficient parallel numerical verification approach to remedy these drawbacks.

In [9], Diochnos et al. presented three algorithms to real solving bivariate polynomial systems and analyzed their asymptotic bit complexities. Among the three algorithms, the difference is the way they match solutions. The method of specialized Rational Univariate Representation (RUR) based on the fast greatest common divisor (hence the name GCD) computations of polynomials with coefficients in an extension field to achieve efficiency (hence the name GRUR) has the lowest complexity

* Corresponding author at: Laboratory for Automated Reasoning and Programming, Chengdu Institute of Computer Applications, CAS, Chengdu 610041, PR China.
E-mail address: qinxl@casit.ac.cn (X. Qin).

0096-3003/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2013.01.039

and performs best in numerous experiments. The GRUR method projects the roots to the x-axis and y-axis, for each x-coordinate α computes the GCD $h(\alpha,y)$ of the square-free parts of $f(\alpha,y)$ and $g(\alpha,y)$, and isolates the roots of $h(\alpha,y)=0$ based on computations of algebraic numbers and the RUR techniques. Our algorithm only uses resultant computation and real solving for univariate polynomial equations with rational coefficients.

The hybrid method proposed by Hong et al. [17] that projects the roots of Σ to the x-axis and y-axis respectively and uses the improved slope-based Hansen-Sengupta to determine whether the boxes formed by the projection intervals contain a root of Σ. The numerical method only works for simple roots of Σ. When the system has multiple roots, the RUR technique is used to isolate the roots. Compared with this method, our approach also computes two resultants of the same total degrees. However, our method is a complete one. Their numerical iteration method needs to use the RUR technique to finding multiple roots. Meanwhile, we also consider discarding the extraneous factors arising from resultant computations to obtain the reliable real roots.

In [2], Bekker et al. presented a Combinatorial Optimization Root Selection method (hence the name CORS) to match the roots of a system of polynomial equations. However, the method is only suitable for solving a small system of polynomial equations, and does not work for the multiple roots. Recently, Cheng et al. [5] proposed a local generic position method to solve the bivariate polynomial equation system. The method can be used to represent the roots of a bivariate equation system as the linear combination of the roots of two univariate equations. Moreover, the multiplicities of the roots of the bivariate polynomial system are also derived. However, the method is very complicated to extend to solve the multivariate equation systems. Our method can solve the larger systems and easily generalize to the multivariate equation systems.

In this paper, we propose a zero-matching method to solve the real roots of an equation system like (1). The basic idea of zero-matching method is as follows: First projecting the roots of Σ to the x-axis, gives the real roots $\{\alpha_1,\alpha_2,\cdots,\alpha_u\}$, and the y-axis, gives the real roots $\{\beta_1,\beta_2,\cdots,\beta_v\}$, where u and v are the number of real roots on x and y for the rest of this paper respectively. Subsequently, $f(x_i,y_j)$ is computed for every x_i and y_j. To that end, for some root x_i there is the corresponding one or more roots y_j to be determined satisfying Σ.

Our original contributions in this paper are the following: Finding a lower bound for a bivariate polynomial system is succeeded when the system is non-zero. The numerical method to determine the real roots of $\Sigma=0$ and the multiplicities of the roots are given. Moreover, our approach that has given solutions to this situation can be the design of parallelized algorithms. It can solve the larger systems and easily generalize to the multivariate equation systems. Our algorithm can be considered as the parallel numerical verification method.

Meanwhile, in the literature [19] we find out the multiplicities of roots for the points of intersection of two curves are not completely correct based on our discarding extraneous factors and multiplicities computing techniques.

The rest of this paper is organized as follows. Section 2 gives some notations, a lower bound for a bivariate polynomial equation if it is non-zero, and how to determine the root multiplicity. Section 3 proposes the algorithm to real solving the bivariate polynomial system and gives a detailed example. Section 4 presents some comparisons of our algorithm. The final section concludes this paper.

2. Notations and main results

2.1. Notations

In what follows $\mathbf{D}$ is a ring, $\mathbf{F}$ is a commutative field of characteristic zero and $\overline{\mathbf{F}}$ its algebraic closure. Typically $\mathbf{D}=\mathbb{Z}$, $\mathbf{F}=\mathbb{Q}$ and $\overline{\mathbf{F}}=\overline{\mathbb{Q}}$.

In this paper, we consider the zero-dimensional bivariate polynomial system as follows:

$$\begin{cases} f(x,y)=\sum\limits_{0\leqslant i\leqslant n}\sum\limits_{0\leqslant j\leqslant m}a_{i,j}x^iy^j=0,\\ g(x,y)=\sum\limits_{0\leqslant i\leqslant p}\sum\limits_{0\leqslant j\leqslant q}b_{i,j}x^iy^j=0. \end{cases} \tag{2}$$

Throughout this paper, note that $deg_x=\max(n,p)$, $deg_y=\max(m,q)$, $N=\max(||f||_1,||g||_1)$, where the $||f||_1$ and $||g||_1$ are the one norm of the vector $(a_{00},a_{01},\cdots,a_{0m},\cdots,a_{n0},\cdots,a_{nm})$ and $(b_{00},b_{01},\cdots,b_{0q},\cdots,b_{p0},\cdots,b_{pq})$, so $||f||_1=\Sigma_i\Sigma_j|a_{ij}|$, and $||g||_1=\Sigma_i\Sigma_j|b_{ij}|$, respectively. Let $M=\max(||t||_1,||T||_1)$, where the $t(x)$ and $T(y)$ are the no extraneous factors in the resultant polynomial of Σ. $|\Sigma|$ denotes that the bivariate polynomial system Σ has been assigned values to two variables.

Let π be the projection map from the Σ to the x-axis:

$$\pi:\mathbb{R}^2\to\mathbb{R},\quad \text{such that } \pi(x,y)=x. \tag{3}$$

For a zero-dimensional system Σ defined in (2), let $t(x)\in\mathbb{Q}[x]$ be the resultant of $f(x,y)$ and $g(x,y)$ with respect to y:

$$t(x)=\mathrm{Res}_y(f(x,y),g(x,y)). \tag{4}$$

Since Σ is zero-dimensional, we have $t(x)\not\equiv 0$. Then $\pi(\mathbf{V}(\Sigma))\subseteq\mathbf{V}(t(x))$, where $\mathbf{V}(f_1,f_2,\cdots,f_m)$ is the set of common real zeros of $f_i=0$. If $t(x)$ is irreducible, then denote the highest degree by deg_t. Let the real roots of $t(x)=0$ be

$$\alpha_1 < \alpha_2 < \cdots < \alpha_u. \tag{5}$$

By using the same method, let $T(y) \in \mathbb{Q}[y]$ be the resultant of $f(x,y)$ and $g(x,y)$ with respect to x:

$$T(y) = \mathrm{Res}_x(f(x,y), g(x,y)). \tag{6}$$

If $T(y)$ is irreducible, then denote the highest degree by deg_T. Let the real roots of $T(y)$ be as follows:

$$\beta_1 < \beta_2 < \cdots < \beta_v. \tag{7}$$

We observe that the above projection map may generate extraneous roots. Fortunately, we can easily discard these extraneous factors by computing the determinant of the sub-matrix of the coefficient matrix. Moreover, if the resultant is irreducible, then it is no extraneous factors. However, when the resultant is reducible, it may suffer from the extraneous factors. The method of removing extraneous factors mentioned can be adapted to the resultant for the bivariate polynomial system [28]. The following theorem is to remove the extraneous roots.

Theorem 2.1. *Σ is defined as in (2). If the resultants of Σ for one variable is reducible, denoted by tem, then the resultant of bivariate polynomial system is the only some irreducible factors in which the other variable appears.*

Proof. The proof can be given similarly to that in Proposition 4.6 of Chapter 3 of [8]. □

2.2. A lower bound for $|\Sigma|$, if $\Sigma \neq 0$

The purpose of this subsection is to prove the following theorem.

Theorem 2.2. *Σ is defined as in (2). Let α, β be two approximate real algebraic numbers. Denote by the integer $s = deg_t \cdot deg_T$, and N as above. If $|\Sigma| \neq 0$, then*

$$|\Sigma| \geqslant N^{1-s} M^{-c \cdot s}, \tag{8}$$

where c is the constant satisfying certain conditions, $|\Sigma|$ is defined as:

(a) If $f(\alpha,\beta) = 0$ or $g(\alpha,\beta) = 0$, then $|\Sigma| = \max\{|f(\alpha,\beta)|, |g(\alpha,\beta)|\}$;
(b) If $f(\alpha,\beta) \neq 0$ and $g(\alpha,\beta) \neq 0$, then $|\Sigma| = \min\{|f(\alpha,\beta)|, |g(\alpha,\beta)|\}$.

Before giving the proof of theorem 2.2, we recall two lemmas:

Lemma 2.1. ([22, Lemma 3]). *Let $\alpha_1, \alpha_2, \ldots, \alpha_q$ be algebraic numbers of exact degree of $d_1, d_2, \ldots, d_q$ respectively. Define $D = [\mathbb{Q}(\alpha_1, \alpha_2, \ldots, \alpha_q) : \mathbb{Q}]$. Let $P \in \mathbb{Z}[x_1, x_2, \ldots, x_q]$ have degree at most N_h in $x_h (1 \leqslant h \leqslant q)$. If $P(\alpha_1, \alpha_2, \ldots, \alpha_q) \neq 0$, then*

$$|P(\alpha_1, \alpha_2, \ldots, \alpha_q)| \geqslant \|P\|_1^{1-D} \prod_{h=1}^{q} M(\alpha_h)^{-DN_h/d_h},$$

where the $M(\alpha_h)$ is the Mahler measure of α_h.

Proof. See the Lemma 4 of [22]. □

Lemma 2.2. *Let α be an algebraic number. Denote by the $M(\alpha)$ of the Mahler measure of α. If P is a polynomial over $\mathbb{Z}$, then*

$$M(\alpha) \leqslant ||P||_1.$$

Proof. For any polynomial $P = \sum_{i=0}^{d} p_i \in \mathbb{Z}[x]$ of degree d with the all roots $\sigma^{(1)}, \sigma^{(2)}, \cdots, \sigma^{(d)}$, we define the *measure* $M(P)$ by

$$M(P) = |p_d| \Pi_{i=1}^{d} \max\{1, |\sigma^{(i)}|\}.$$

The Mahler measure of an algebraic number is defined to be the Mahler measure of its minimal polynomial over $\mathbb{Q}$. We know from Landau ([15], p. 154, Thm. 6. 31) that for each algebraic number α

$$M(\alpha) \leqslant ||P||_2,$$

where $||P||_2 = (\sum_{i=0}^{d} |p_i|^2)^{1/2}$. It is very easy to get that $||P||_2 \leqslant ||P||_1$. This completes the proof of the lemma. □

Now we turn to give the proof of Theorem 2.2.

Proof. From the assumption of the theorem, since Σ is defined as in (2). Let the pair (α,β) be corresponding value to the variable x and y for Σ respectively. We have the following equations:

$$f(\alpha,\beta)=\sum_{0\leqslant i\leqslant n}\sum_{0\leqslant j\leqslant m}a_{i,j}\alpha^i\beta^j, \tag{9a}$$

$$g(\alpha,\beta)=\sum_{0\leqslant i\leqslant p}\sum_{0\leqslant j\leqslant q}b_{i,j}\alpha^i\beta^j. \tag{9b}$$

At first, we consider the lower bound for the equation (9a). Define $k=[\mathbb{Q}(\alpha,\beta):\mathbb{Q}]$. Denote by $|f|=|f(\alpha,\beta)|$, and r,t by the exact degree of algebraic numbers α,β respectively. From Lemma (2.1), if $|f|\neq 0$, then

$$|f|\geqslant ||f||_1^{1-k}M(\alpha)^{-kn/r}M(\beta)^{-km/t}.$$

We observe that $M(\alpha)$ and $M(\beta)$ derive from $t(x)$ and $T(y)$ respectively. From Lemma (2.2), we can get the following inequality:

$$M(\alpha)\leqslant ||t||_1,\quad M(\beta)\leqslant ||T||_1.$$

So we can obtain that

$$|f|\geqslant ||f||^{1-k}||t||_1^{-kn/r}||T||_1^{-km/t}. \tag{10}$$

By using the same technique as above, we can obtain the lower bound for the equation (9b). Denote by $|g|=|g(\alpha,\beta)|$. If $|g|\neq 0$, then

$$|g|\geqslant ||g||^{1-k}||t||_1^{-kn/r}||T||_1^{-km/t}. \tag{11}$$

Since we have the following two cases:

(a) If $f(\alpha,\beta)=0$ or $g(\alpha,\beta)=0$, then $|\Sigma|=\max\{|f(\alpha,\beta)|,|g(\alpha,\beta)|\}$;
(b) If $f(\alpha,\beta)\neq 0$ and $g(\alpha,\beta)\neq 0$, then $|\Sigma|=\min\{|f(\alpha,\beta)|,|g(\alpha,\beta)|\}$.

Hence we are able to obtain the lower bound for the bivariate polynomial system. From the above assumption, we can get the following parameters:

$$k=[\mathbb{Q}(\alpha,\beta):\mathbb{Q}]\leqslant deg\{t(x)\}\cdot deg\{T(y)\}=deg_t\cdot deg_T, \tag{12a}$$

$$N=\max\{||f||_1,||g||_1\},M=\max\{||t||_1,||T||_1\},r=deg_t,t=deg_T. \tag{12b}$$

Combined with the Eq. (12a) and (12b), it is obvious that $s=k$ and the constant $c=\frac{deg_t}{r}+\frac{deg_T}{t}+1$. Finally, note that the constant c satisfies both cases. This proves the theorem. □

As the corollary of Theorem 2.2, we have

Corollay 2.1. *Under the same condition of Theorem 2.2, if* $|\Sigma|<N^{1-s}M^{-c\cdot s}$, *then* $|\Sigma|=0$. *We say that* α *is associated with* β *for the real root of* Σ. *Denote by the* $\varepsilon=N^{1-s}M^{-c\cdot s}$ *for the rest of this paper.*

Proof. The proof is very easy by contradiction. □

2.3. Root multiplicity

The results of this subsection can be provided for the root multiplicity of Σ. We follow the approach and terminology of [8,11].

Let $\mathcal{C}_f,\mathcal{C}_g$ be f,g corresponding affine algebraic plane curves, defined by the equations Σ. Let $I=<f,g>$ be the ideal that they generate in $\mathbf{F}[x,y]$, and so the associated quotient ring is $\mathcal{A}=\overline{\mathbf{F}}[x,y]/I$. Let the distinct intersection points, which are the distinct roots of Σ, be $\mathcal{C}_f\cap\mathcal{C}_g\subset\{S_{ij}=(\alpha_i,\beta_j)\}_{1\leqslant i\leqslant u,1\leqslant j\leqslant v}$.

The multiplicity of a point S_{ij} is

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

where $\mathcal{A}_{S_{ij}}$ is the local ring obtained by localizing $\mathcal{A}$ at the maximal ideal $I=<x-\alpha_i,y-\beta_j>$.

If $\mathcal{A}_{S_{ij}}$ is a finite dimensional vector space, then $S_{ij}=(\alpha_i,\beta_j)$ is an isolated zero of I and its multiplicity is called the intersection number of the two curves. The finite $\mathcal{A}$ can be decomposed as a direct sum $\mathcal{A}=\mathcal{A}_{S_{11}}\bigoplus\mathcal{A}_{S_{12}}\bigoplus\cdots\bigoplus\mathcal{A}_{S_{uv}}$ and thus $dim_{\overline{\mathbf{F}}}\mathcal{A}=\sum_{i=1}^{uv}mult(S_{ij}:\mathcal{C}_f\cap\mathcal{C}_g)$.

Proposition 2.1 [11, Proposition 1]. *Let* $f,g\in\mathbf{F}[x,y]$ *be two coprime curves, and let* $p\in\overline{\mathbf{F}}^2$ *be a point. Then*

$$mult(p:fg)\geqslant mult(p:f)mult(p:g),$$

where equality holds if and only if C_f and C_g have no common tangents at p.

Proposition 2.2. *Let us obtain the real roots of $\Sigma = 0$ in (5) and (7). If the two matching pairs (α_i, β_j) and $(\alpha_{i+1}, \beta_{j+1})$(for $1 \leqslant i \leqslant u, 1 \leqslant j \leqslant v$) are satisfying $\Sigma = 0$, $|\alpha_i - \alpha_{i+1}| < \varepsilon$ and $|\beta_j - \beta_{j+1}| < \varepsilon$, then the (α_i, β_j) is multiple root of $\Sigma = 0$.*

Proof. From Theorem 2.2 and Corollary 2.1, it is obvious that $\Sigma = 0$ if and only if

$$|\Sigma| < \varepsilon.$$

Therefore, the error controlling is less than ε in numerical computation. Under the assumption of the proposition, we get $|\alpha_i - \alpha_{i+1}| < \varepsilon$ and $|\beta_j - \beta_{j+1}| < \varepsilon$. So we are able to obtain that $|\alpha_i - \alpha_{i+1}| = 0$ and $|\beta_j - \beta_{j+1}| = 0$ in the truncated error. This proves the proposition. □

From Corollary 2.1, the two-tuple (α, β) is the real root of $\Sigma = 0$. This method is called a **zero-matching method**. The technique is a posteriori method to match the solutions for the bivariate polynomial systems. It can be easily generalized to real solving the multivariate polynomial systems.

3. Derivation of the algorithm

The aim of this section is to describe an algorithm for real solving bivariate polynomial equations by using zero-matching method. We first find the parameters N, c and s, then obtain the no extraneous factors $t(x)$ and $T(y)$ with the resultant elimination methods, and real solving two univariate polynomials, and finally match the real roots for the systems.

3.1. Description of algorithm

Algorithm 1 is to discard the extraneous factors from the resultant method, and Algorithm 2 is to obtain the solutions of bivariate polynomial systems.

Algorithm 1. **def**(Σ, var)

Input:$\{f(x,y), g(x,y)\}$, *var* is one variable.

Output: No extraneous factors resultant of Σ.

1: $tem \leftarrow Res_{var}\{f(x,y), g(x,y)\}$;
2: **if** *tem* is irreducible **then**
3: **return** *tem*;
4: **else**
5: $tem \leftarrow Res \cdot extraneousfacotrs$;
6: **return** *Res*.
7: **end if**

Now we can give the Algorithm 2 to compute the real roots for $\Sigma = 0$.

Algorithm 2. **zmm**(Σ)

Input: $\Sigma = \{f(x,y), g(x,y)\}$ is a zero-dimensional bivariate polynomial system.

Output: A set for the real roots of $\Sigma = 0$.

1: Project on the x-axis such that $t(x) = Res_y(f(x,y), g(x,y))$;
2: Project on the y-axis such that $T(y) = Res_x(f(x,y), g(x,y))$;
3: Discard the extraneous factors from $t(x)$ and $T(y)$ by using Algorithm 1;
4: Find the parameters N and s, and Compute c according to the Theorem 2.2;
5: Obtain the lower bound ε by Corollary 2.1;
6: Solve the real roots of the resultant $t(x)$ for the set $\mathbf{S_x} = \{\alpha_1, \alpha_2, \cdots, \alpha_u\}$;
7: Solve the real roots of the resultant $T(y)$ for the set $\mathbf{S_y} = \{\beta_1, \beta_2, \cdots, \beta_v\}$;
8: Match the real root pair to get the solving set $\mathbf{S} = \{(\alpha_i, \beta_j), 1 \leqslant i \leqslant u, 1 \leqslant j \leqslant v\}$ by Corollary 2.1;
9: Check the root multiplicity of the set **S** by Proposition 2.2.

The parallelization of the algorithm that we have just described can be easily done because it performs the same computations on different steps of data without the necessity of communication between the processors. Observe that the Step 1 and Step 2, Step 6 and Step 7 of the Algorithm 2 can be easily paralleled, respectively.

Now we give a theorem about the computational complexity of the whole algorithm.

Theorem 3.1. *Algorithm 2 works correctly as specified and its complexity includes as follows:*

(a) $O(d\tau + dlgd)$ *for computation of real solving univariate polynomial, where* d *is the degree of corresponding polynomial,* $\tau = 1 + \max_{i\leqslant d} lg|a_i|$ *and* a_i *is the coefficients.*
(b) $O(\frac{uv}{r})$ *for matching the solutions of bivariate polynomial systems, where* r *is the number of computing processors.*

Proof. Correctness of the algorithm follows from Theorem 2.2.

(a) The number of arithmetic operations required to isolate all real roots is the number of real root isolation of univariate polynomial by using subdivision-based Descartes' rule of sign. Using exactly the same arguments we know that they perform the same number of steps, that is $O(d\tau + dlgd)$.
(b) As indicated before, the problem of matching the real roots of polynomial system mainly relies on the scale of solutions of every variable and the number of computing processors, respectively. □

3.2. A small example in detail

Example 3.1. We propose a simple example $f(x,y) = x^2 - y^2 - 3$ and $g(x,y) = 3x^2 - 2y^3 - 1$ to illustrate our algorithms.

Step 1: $t(x) = 4x^6 - 45x^4 + 114x^2 - 109$;
Step 2: $T(y) = (-2y^3 + 8 + 3y^2)^2$;
Step 3: Discard the extraneous factors $T(y) = -3y^2 - 8 + 2y^3$;
Step 4: Obtain the parameters $N = 5, c = 2, s = 4$;
Step 5: Obtain the lower bound $\varepsilon = .1280 \times 10^{-4}$;
Step 6: Solve the real roots of the resultant $t(x)$ for the set $\mathbf{S_x} = \{-2.858288520, 2.858288520\}$;
Step 7: Solve the real roots of the resultant $T(y)$ for the set $\mathbf{S_y} = \{2.273722337\}$;
Step 8: Combine the pairs from $\mathbf{S_x}$ and $\mathbf{S_y}$ respectively, Substitute the pairs into Σ for variables x and y, determine whether less than the lower bound ε, finally we find that the pairs $\mathbf{S} = \{\{x = -2.858288520, y = 2.273722337\}, \{x = 2.858288520, y = 2.273722337\}\}$ are the solutions for Σ;
Step 9: The multiplicity of the root of the system is one.

3.3. Generalization and applications

As for the generalization of the algorithm to real solving the multivariate equation systems case, we have to say that the situation is completely analogous to the bivariate case. However, its key technique is to transform the multivariate polynomial equations to the corresponding univariate polynomial equations. We can consider the Dixon Resultant Method to break this problem [6]. However, we observe that how to improve the projection algorithm in resultant methods is a significant challenge.

Moreover, our algorithm is applicable for rapidly computing the minimum distance between two objects collision detection [26]. This also enables us to improve the complexity of computing the topology of a real plane algebraic curve [9].

By using our discarding extraneous factors and multiplicities computing techniques, we find out the literature [19], the multiplicities of roots for the points of intersection of two curves are not completely correct in Fig. 1. As a matter of fact, the multiplicity of the points of intersection $(x = 1., y = 1.)$ should be a triple root, but they get quintuplicate root. We correct their illustrated example as follows:

$$f(x,y) = x^3 - 3x^2 + 5x - 4 + y^3 - 3y^3 + 5y - 2xy, \quad g(x,y) = 2x^3 - 2x^2 + x - 4 - 4x^2y + 2xy + 9y + 3xy^2 - 8y^2 + y^3.$$

Based on Algorithm 1, we obtain the following polynomial systems:

$$t(x) = 56x^7 - 592x^6 + 2640x^5 - 6432x^4 + 9240x^3 - 7824x^2 + 3616x - 704,$$

$$T(y) = -56y^7 + 496y^6 - 1776y^5 + 3264y^4 - 3192y^3 + 1488y^2 - 160y - 64.$$

So,we execute Algorithm 2 to get the solutions in Table 1.

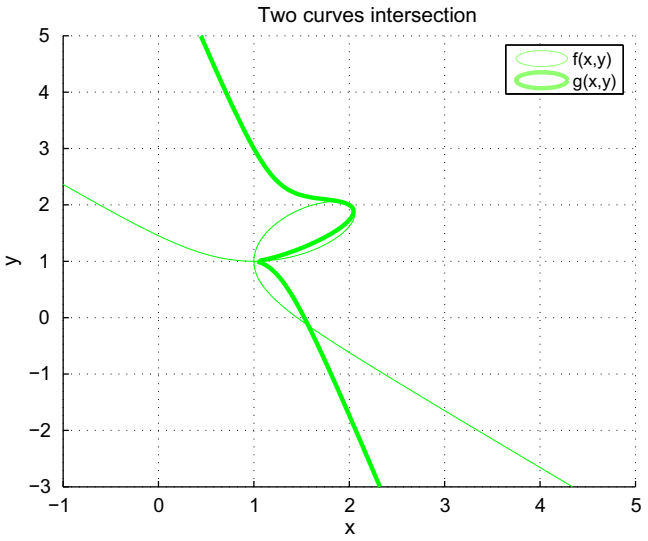

Fig. 1. Intersection of two curves.

Table 1
Solutions for computing intersection of two curves.

Root (x,y)	d
[x = 1., y = 1.]	3
[x = 1.571428571, y = -.1428571429]	1
[x = 2., y = 2.]	3

4. Some comparisons

We have implemented the above algorithms as a software package ZMM in *Maple* 12. For problems of small size like the example of Section 3, any method can obtain the solutions in little time. But when the size of the problems is not small the differences appear clearly. Extensive experiments with this package show that this approach is efficient and stable, especially for larger and more complex bivariate polynomial systems.

We compare our method with LGP [5], Isolate [24], DISCOVERER [25], and GRUR [9]. LGP is a software package for root isolation of bivariate polynomial systems with local generic position method. Isolate is a tool to solve general equation systems based on the Realsolving C library by Rouillier. DISCOVERER is a tool for solving semi-algebraic systems. GRUR is a tool to solve bivariate equation systems. The following examples run in the same platform of*Maple* 12 under Windows and AMD Athlon (tm) 2.70 GHZ, 2.00 GB of main memory. We did three sets of experiments. The precision in these experiments is set to be high. In the three tables, where '?' represents that the computation is not finished.

Table 2
Time for computing dense bivariate polynomials with no multiple roots.

system	deg		solutions	Average Time (sec)				
	f	*g*		ZMM	LGP	Isolate	DISCOVERER	GRUR
S1	4	7	2	0.031	0.031	0.047	0.313	2.734
S2	6	8	6	0.415	1.328	0.500	1.828	247.203
S3	7	8	6	1.204	2.734	1.500	7.047	382.640
S4	8	9	6	4.211	8.906	4.672	20.437	2714.438
S5	9	10	2	4.070	8.485	4.687	89.235	1645.312
S6	10	7	6	1.805	3.860	2.109	22.250	978.421
S7	10	11	4	21.078	43.734	22.828	?	?
S8	12	11	2	26.945	54.969	29.094	?	?
S9	12	13	4	118.266	241.734	123.469	?	?
S10	13	11	1	15.446	31.485	17.796	?	?
S11	14	10	8	63.914	200.828	68.594	?	?

In Table 2 the results are given both f and g are randomly generated dense polynomials with the same degree and with integer coefficients between -20 and 20. The command of *Maple* is as follows:

$randpoly([x,y], coeffs = rand(-20..20), dense, degree = 10)$.

In Table 3 the results are given both f and g are randomly generated sparse polynomials in the same degree, with sparsity *default*, and with integer coefficients between -20 and 20. The command of *Maple* is as follows:

$randpoly([x,y], coeffs = rand(-20..20), sparse, degree = 10)$.

In Table 4 the results are given is done with polynomial systems with multiple roots. We randomly generate a polynomial $h(x,y,z)$ and take $f(x,y) = Res_z(h,h_z), g(x,y) = f_y(x,y)$. Since $f(x,y)$ is the projection of a space curve to the xy-plane, it most probably has singular points and $f = g = 0$ is an equation system with multiple roots. The command of *Maple* is as follows:

$h := randpoly([x,y,z], coeffs = rand(-5..5), degree = 5); f := resultant(h, diff(h,z), z); g := diff(f,y)$.

From the Table 2–4, we have the following observations.

In the first two cases, the equations are randomly generated and hence may have no multiple roots. For systems without multiple roots, ZMM is the fastest method, which is significantly faster than LGP and Isolate. Both ZMM and LGP compute two resultants and isolate their real roots. LGP is slow, because the polynomials obtained by the shear map are usually dense and with large coefficients [5]. DISCOVERER and GRUR generally work for equation systems with degrees not higher than ten within reasonable time.

For systems with multiple roots, in the sparse and low degree cases, all methods are fast. Note that our method is quite stable for equation systems with or without multiple roots. LGP and Isolate are also quite stable, but slower than ZMM for bivariate equation systems.

We also observe that all methods spend more time with sparse and dense polynomials than polynomials with multiple roots in the same high degree. This phenomenon needs further exploration.

Remark 4.1. Of course, we should mention that DISCOVERER and Isolate can be used to solve general polynomial equations and even inequalities. Here our comparison is limited to the bivariate case. In further work, we would like to consider solving multivariate polynomial equations.

Remark 4.2. As is well known, the parallel algorithm is well suited for the implementation on parallel computers that allows the increase of the calculation speed. If our algorithms have been fully parallelized by using a large enough number of processors for each case, the real solutions of all the examples will have been computed in a couple of seconds.

Table 3
Time for computing sparse bivariate polynomials with no multiple roots.

system	deg		solutions	Average Time (sec)				
	f	g		ZMM	LGP	Isolate	DISCOVERER	GRUR
S1	5	6	1	0.015	0.032	0.015	0.141	1.032
S2	6	7	3	0.040	0.062	0.047	0.188	5.375
S3	7	5	3	0.024	0.047	0.047	0.265	2.688
S4	8	6	5	0.031	0.031	0.047	0.094	1.031
S5	9	8	2	0.047	0.172	0.078	1.828	51.000
S6	10	11	3	0.063	0.297	0.125	0.656	11.110
S7	11	9	2	0.164	0.609	0.375	3.938	877.875
S8	12	13	2	1.141	2.593	1.453	6.703	1607.719
S9	13	11	4	2.508	5.344	2.969	?	?
S10	15	17	1	0.532	1.234	1.266	?	?
S11	20	17	4	18.180	39.688	20.235	?	?

Table 4
Time for computing bivariate polynomials with multiple roots.

system	deg		solutions	Average Time (sec)				
	f	g		ZMM	LGP	Isolate	DISCOVERER	GRUR
S1	3	2	2	0.016	0.016	0.016	0.016	0.062
S2	4	3	2	0.	0.032	0.031	0.016	0.094
S3	4	6	7	0.024	0.016	0.047	0.109	1.109
S4	5	4	3	0.	0.016	0.	0.016	0.109
S5	6	5	2	0.015	0.	0.	0.016	0.063
S6	9	8	2	0.016	0.046	0.032	0.015	0.063
S7	12	11	3	0.109	0.234	0.187	0.063	0.094
S8	13	12	7	2.875	137.641	3.141	1.328	207.094
S9	14	13	4	0.860	2.891	0.953	0.141	0.3110
S10	19	18	1	0.672	1.547	0.797	22.156	1520.812
S11	16	15	5	7.945	27.047	9.000	?	?

5. Conclusion

In this paper, we propose a zero-matching method to real solving bivariate polynomial equation systems. The basic idea of this method is to find the lower bound for a bivariate polynomial system when the system is non-zero. Moreover, we provide an algorithm for discarding extraneous factors with resultant computation and show how to construct the parallel numerical verification algorithm for real solving the bivariate polynomial system. An efficient method for multiplicities of the roots is also derived. The complexity of our method has increased steadily with the growth of a bivariate polynomial system. Extensive experiments show that our approach is efficient and stable. The result of this paper can be extended to real solving of bivariate polynomial equations with more than two polynomials by using the resultant method. Furthermore, our method can be easily generalized to the multivariate polynomial systems.

Acknowledgement

This research was partly supported by China 973 Project NKBRPC-2011CB302402, the National Natural Science Foundation of China (Grant Nos. 91118001, 11171053), the West Light Foundation of the Chinese Academy of Sciences, and China Postdoctoral Science Foundation funded project (Grant No. 2012M521692).

We would like to thank Prof. Xiaoshan Gao and Dr. Jinsan Cheng for providing the Maple code of their method, available at: http://www.mmrc.iss.ac.cn/~xgao/software.html.

The first author is also grateful to Dr. Shizhong Zhao for his valuable discussions about discarding the extraneous factors in resultant computations.

The authors are also grateful to the anonymous referees for their helpful comments and suggestions.

References

[1] D.S. Arnon, G. Collins, S. McCallum, Cylindrical algebraic decomposition. Part II: An adjacency algorithm for plane, SIAM Journal on Computing 13 (4) (1984) 878–889.
[2] H. Bekker, E.P. Braad, B. Goldengorin, Using bipartite and multidimensional matching to select the roots of a system of polynomial equations, in: Computational Science and Its Applications, ICCSA 2005, Springer, Berlin, 2005, pp. 397–406.
[3] J.P. Boyd, Computing real roots of a polynomial in Chebyshev series form through subdivision with linear testing and cubic solves, Applied Mathematics and Computation 174 (2006) 1642–1648.
[4] J.S. Cheng, X.S. Gao, C.K. Yap, Complete numerical isolation of real zeros in zero-dimensional triangular systems, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Ontario, 2007, ACM Press, New York, 2007, pp. 92–99.
[5] J.S. Cheng, X.S. Gao, J. Li, Root isolation for bivariate polynomial systems with local generic position method, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Seoul, 2009, ACM Press, New York, 2009, pp. 103–110.
[6] E.W. Chionh, M. Zhang, R.N. Goldman, Fast computation of the Bezout and Dixon resultant matrices, Journal of Symbolic Computation 33 (1) (2002) 13–29.
[7] G. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decompostion, in: Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, Kaiserslautern, 1975, Springer, Berlin, 1975, pp. 134–183.
[8] D.A. Cox, J. Little, D. OShea, Using Algebraic Geometry, second ed., Springer-Verlag, Berlin, Heidelberg, 2005.
[9] D.I. Diochnos, I.Z. Emiris, E.P. Tsigaridas, On the complexity of real solving bivariate systems, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Ontario, 2007, ACM Press, New York, 2007, pp. 127–134.
[10] A. Eigenwillig, M. Kerber, N. Wolpert, Fast and exact geometric analysis of real algebraic plane curves, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Ontario, 2007, ACM Press, New York, 2007, pp. 151–158.
[11] I.Z. Emiris, E.P. Tsigaridas, Real solving of bivariate polynomial systems, in: Proceedings of the International Workshop on Computer Algebra in Scientific Computing, Kalamata, 2005, Springer, Heidelberg, 2005, pp. 150–161.
[12] I.Z. Emiris, E.P. Tsigaridas, Real algebraic numbers and polynomial systems of small degree, Theoretical Computer Science 409 (2) (2008) 186–199.
[13] H.G. Fu, Y. Wang, S.H. Zhao, et al, A recursive algorithm for constructing complicated Dixon matrices, Applied Mathematics and Computation 217 (2010) 2595–2601.
[14] X.S. Gao, M. Li, Rational quadratic approximation to real algebraic curves, Computer Aided Geometric Design 21 (2004) 805–828.
[15] J. von zur Gathen, J. Gerhard, Modern Computer Algebra, Cambridge University Press, Cambridge, 1999.
[16] L. Gonzalez-Vega, I. Necula, Efficient topology determination of implicitly defined algebraic plane curves, Computer Aided Geometric Design 19 (2002) 719–743.
[17] H. Hong, M. Shan, Z. Zeng, Hybrid method for solving bivariate polynomial system, SRATC 2008, Shanghai, 2008. <http://www.is.pku.edu.cn/~xbc/SRATC2008/meijing.pdf>
[18] M. Jirstrand, Nonlinear control system design by quantifier elimination, Journal of Symbolic Computation 24 (2) (1997) 137–152.
[19] K. Ko, T. Sakkalis, N. Patrikalakis, Resolution of multiple roots of nonlinear polynomial systems, International Journal of Shape Modeling 11 (1) (2005) 121–147.
[20] K.H. Ko, T. Sakkalis, N.M. Patrikalakis, A reliable algorithm for computing the topological degree of a mapping in R^2, Applied Mathematics and Computation 196 (2) (2008) 666–678.
[21] D. Lazard, Thirty years of polynomial system solving, and now?, Journal of Symbolic Computation 44 (3) (2009) 222–231
[22] M. Mignotte, M. Waldschmidt, Linear forms in two logarithms and schneider's method, Mathematische Annalen 231 (1978) 241–267.
[23] B. Mourrain, J.-P. Pavone, Subdivision methods for solving polynomial equations, Technical Report RR-5658, INRIA Sophia-Antipolis. <http://www.inria.fr/rrrt/rr-5658.html>
[24] F. Rouillier, Solving zero-dimensional systems through the rational univariate representation, Journal of Applicable Algebra in Engineering, Communication and Computing 9 (5) (1999) 433–461.
[25] B. Xia, L. Yang, An algorithm for isolating the real solutions of semi-algebraic systems, Journal of Symbolic Computation 34 (2002) 461–477.
[26] L. Yang, Y. Feng, X.L. Qin, An efficient approach for computing distance between two quadratic surfaces, Proceedings 2009 of the International Conference on Computer Science and Information Technology, Beijing, 2009, 2, IEEE press, Piscataway, 2009, pp. 244–248.
[27] L. Yang, J.Z. Zhang, X.R. Hou, Nonlinear Algbraic Equation System and Automated Theorem Proving, Shanghai Scientific and Technological Education Publishing House, Shanghai, 1996. pp. 95–100 (in Chinese).
[28] S.Z. Zhao, H.G. Fu, Three kinds of extraneous factors in Dixon resultants, Science in China Series A: Mathematics 52 (1) (2009) 160–172.