Journal of Symbolic Computation 68 (2015) 204–224

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

A generic position based method for real root isolation of zero-dimensional polynomial systems

Jin-San Cheng, Kai Jin

KLMM, Institute of Systems Science, AMSS, Chinese Academy of Sciences, China

ARTICLE INFO

Article history:
Received 16 September 2013
Accepted 1 July 2014
Available online 22 September 2014

Keywords:
Polynomial systems
Real root isolation
Linear univariate representation
Generic position

ABSTRACT

We improve the local generic position method for isolating the real roots of a zero-dimensional bivariate polynomial system with two polynomials and extend the method to general zero-dimensional polynomial systems. The method mainly involves resultant computation and real root isolation of univariate polynomial equations. The roots of the system have a linear univariate representation. The complexity of the method is $\tilde{O}_B(N^{10})$ for the bivariate case, where $N = \max(d, \tau)$, d resp., τ is an upper bound on the degree, resp., the maximal coefficient bitsize of the input polynomials. The algorithm is certified with probability 1 in the multivariate case. The implementation shows that the method is efficient, especially for bivariate polynomial systems.

1. Introduction

Real root isolation of zero-dimensional polynomial systems is a fundamental problem in symbolic computation and it has many applications. The problem has been studied for a long time and there are a lot of results. One can compute the real roots of a zero-dimensional polynomial system by symbolic methods, numeric methods and symbolic–numeric methods. In context of symbolic methods, we can mention the characteristic set methods, Gröbner basis methods, the resultant methods and so on. In this paper, we focus on the resultant methods. We consider the zero-dimensional system as $\{f_1, \ldots, f_m\} \subset \mathbb{Z}[x_1, \ldots, x_n]$, where $\mathbb{Z}$ is the ring of integers.

E-mail addresses: jcheng@amss.ac.cn (J.-S. Cheng), jinkaijl@163.com (K. Jin).

http://dx.doi.org/10.1016/j.jsc.2014.09.017
0747-7171/© 2014 Elsevier Ltd. All rights reserved.

The idea of this paper comes from a geometric property of the roots of a polynomial system: generic position. Generic position was used in the polynomial system solving for a long time (Alonso et al., 1996; Becker and Wörmann, 1996; Canny, 1988; Cheng et al., 2009; Diochnos et al., 2009; Giusti et al., 2001; Gao and Chou, 1999; Giusti and Heintz, 1991; Kobayashi et al., 1988; Rouillier, 1999; Tan and Zhang, 2009; Yokoyama et al., 1989). Let's explain it for the bivariate case. Simply speaking, a zero-dimensional bivariate system is said to be in a **generic position** if we can find a complex plane, say the x-axis, such that different complex zeros of the system are projected to different complex points on the complex x-axis. In the rest of this paper, when we say root(s), we mean real root(s) if there is no special illustration.

Solving bivariate polynomial systems is widely studied in recent years (Busé et al., 2005; Cheng et al., 2009; Corless et al., 1997; Emiris et al., 2008; Emiris and Tsigaridas, 2005; Diochnos et al., 2009; Emeliyanenko et al., 2011; Hong et al., 2008; Qin et al., 2013). Most of these methods projected the systems to two directions (x-axis, y-axis) and identified whether a root pair (one x-coordinate and one y-coordinate) was a true root or not (Diochnos et al., 2009; Emeliyanenko et al., 2011; Hong et al., 2008; Qin et al., 2013). In Busé et al. (2005), Corless et al. (1997), they projected the roots of the bivariate system to x-axis, using a matrix formulation, and lifted them up to recover the roots of the original system. The multiplicity of the roots are also considered.

A **local generic position** method was proposed to isolate the real roots of a zero-dimensional bivariate polynomial system in Cheng et al. (2009). In the local generic position method, the roots of a zero-dimensional bivariate polynomial system $\Sigma = \{f(x, y), g(x, y)\}$ are represented as linear combinations of the roots of two univariate polynomial equations $R_1(x) = \mathrm{Res}_y(f, g) = 0$ and $R_2(x) = \mathrm{Res}_y(f(x + sy, y), g(x + sy, y)) = 0$:

$$\left\{x = \alpha,\ y = \frac{\beta - \alpha}{-s} \,\middle|\, \alpha \in \mathbb{V}\big(R_1(x)\big),\ \beta \in \mathbb{V}\big(R_2(x)\big),\ |\beta - \alpha| < S\right\},$$

where s, S are constants satisfying certain given conditions. Each root (α, β) of $\Sigma = 0$ is projected in $R_2(x) = 0$ such that the corresponding root is in a neighborhood of $\alpha : E = \{v \mid |v - \alpha| < S\}$. All the roots of $R_2(x) = 0$ in E correspond to the roots of $\Sigma = 0$ on the fiber $x = \alpha$. Thus we can recover the y-coordinates of the roots of $\Sigma = 0$ from the roots of $R_2(x) = 0$. The multiplicities of the roots of $\Sigma = 0$ are also preserved in the corresponding roots of $R_2(x) = 0$. The implementation of the method shows that it is efficient and stable when compared to the best methods at that time, especially when the system has multiple roots. But the local generic position method has a bottleneck. When some of the roots of $R_1(x)$ are very close, s will be very small. Thus computing $R_2(x)$ and isolating its roots is time-consuming. Sometimes, it is more than 90% of the total computing time! The rate increases when the degrees of the polynomials in the systems increase.

The contribution of the paper is that we present a method to overcome the bottleneck of the local generic position method and extend the method to general zero-dimensional multivariate polynomial system mainly involving resultant computation and univariate polynomial root isolation, which is easy to implement. We also analyze the complexity of the algorithm for the bivariate case. We compare our implementation with several other efficient related softwares, such as local generic position method (Cheng et al., 2009), Hybird method (Hong et al., 2008), Discovery (Xia and Yang, 2002) and Isolate (Rouillier, 1999). The results show that our algorithm is efficient, especially in bivariate case.

In order to overcome the drawback of the local generic position method, we present a method to search for a better s with a small bitsize and present another way to recover the roots of the system. This is the main contribution of the paper. Finding the correspondence between the roots of $\Sigma = 0$ and $R_2(x) = 0$, we can recover the roots of $\Sigma = 0$. It works as follows. First, we compute $R_1(x)$ and its roots. From the isolating intervals of the roots of $R_1(x) = 0$, we get the root isolating interval candidates of $f = g = 0$ by computing the roots of interval polynomials. We compute a rational number s such that any two isolating interval candidates are not overlapping under a linear transformation $\varphi : (x, y) \to (x - sy, y)$ and $\{\varphi(f), \varphi(g)\}$ is in a generic position. Then for each isolating interval candidate $K = [a, b] \times [c, d]$, we can isolate the roots of $R_2(x) = 0$ in the interval $\pi_y(\varphi(K))$ ($\pi_y : (x, y) \to (x)$) to recover the isolating intervals of $f = g = 0$. The multiplicity(ies) of the root(s) of the system in K is (are) the multiplicity(ies) of the corresponding root(s) in $\pi_y(\varphi(K))$. The bivariate polynomial system with several polynomials can be solved using the method with a little modification (see Section 4). In some special case, we can certify the root candidates by the monotonicity of the interval polynomial without computing the second resultant, which is a speedup of our new algorithm.

We extend the method to zero-dimensional polynomial systems in the multivariate case. Let's consider the trivariate case for example. For a zero-dimensional polynomial system $\{f_1, f_2, f_3\} \subset \mathbb{Z}[x, y, z]$, we can get a bivariate polynomial system $\{g_1, g_2\} \subset \mathbb{Z}[x, y]$, where $g_1 = \mathrm{Res}_z(f_1, f_2)$, $g_2 = \mathrm{Res}_z(f_1, f_3)$. Isolating the roots of $\{g_1, g_2\}$, using the isolating intervals to construct interval polynomials for f_1, f_2, f_3, isolating the roots of these interval polynomials, we can get the root isolating interval candidates of the system $\{f_1, f_2, f_3\}$. For all the root isolating interval candidates, we separate them into different groups such that the first coordinates of the isolating boxes in each group are the same. We compute an s such that for each group, the last two coordinates of the corresponding roots of $\{f_1(x, y + sz, z), f_2(x, y + sz, z), f_3(x, y + sz, z)\}$ in the group are in a generic position. Solving $\mathcal{P} = \{\mathrm{Res}_z(f_1(x, y + sz, z), f_2(x, y + sz, z)), \mathrm{Res}_z(f_1(x, y + sz, z), f_3(x, y + sz, z))\}$, we can check whether the root candidates of $\{f_1, f_2, f_3\} = 0$ containing its real roots or not from the roots of $\mathcal{P} = 0$. Sometimes we need to take a linear combination of f_i's to construct a new system to ensure that the two projection polynomials form a zero-dimensional system. In a similar way, we can solve a general zero-dimensional polynomial system. This method usually works well for the systems with 2 or 3 variables.

The complexity of the bivariate system solving is studied before. One is $\tilde{O}_B(N^{12})$ (Diochnos et al., 2009), the other is $\tilde{O}_B(N^8)$ (Emeliyanenko and Sagraloff, 2012). Ours is $\tilde{O}_B(N^{10})$, where N is the maximum between the degree bound and the bitsize bound of the coefficients of the polynomials in the system.

The rest of this paper is organized as follows. In Sections 2 and 3, the basic tools related to interval polynomials and generic position are introduced. In Section 4, we present the improved bivariate systems solving method. In Section 5, the improved method is extended to general zero-dimensional systems. In Section 6, we give the complexity analysis of this algorithm. Experimental results are presented in Section 7.

2. The interval polynomial and its real roots

In this section, we will show how to construct an interval polynomial related to a polynomial and how to compute the real roots of an interval polynomial. Interval methods were also used to solve polynomial systems before (Mantzaflaris et al., 2011; Stahl, 1995; Mourrain and Pavone, 2009).

Let $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ be the fields of rational numbers, real numbers and complex numbers respectively.

Denote $\mathbb{V}(f)$ as the zeros in $\mathbb{C}^n$ of $f \in \mathbb{Q}[x_1, \ldots, x_n]$ and $\mathbb{V}_{\mathbb{R}}(f) = \mathbb{V}(f) \cap \mathbb{R}^n$. Here f can be placed by a polynomial system.

Given $f = a_0 + a_1x + \ldots + a_nx^n \in \mathbb{Z}[x]$, we can rewrite it in Horner form.

$$f_h = a_0 + \big(a_1 + \big(a_2 + \ldots + (a_{n-1} + a_nx) \cdots x\big)x\big)x.$$

If $a_i \in \mathbb{Z}[x_1]$ and we rewrite it in Horner form, then $f_h \in \mathbb{Z}[x_1, x]$ is a bivariate polynomial in Horner form with order $x_1 \prec x$. Recursively, we can rewrite a multivariate polynomial $f \in \mathbb{Z}[x_1, \ldots, x_n]$ in Horner form in a fixed variable order $x_1 \prec x_2 \prec \cdots \prec x_n$.

Let $f \in \mathbb{Z}[x_1, \ldots, x_n, x]$ and rewrite it as below

$$f = h_0 + h_1x + \ldots + h_mx^m,$$

where $h_i \in \mathbb{Z}[x_1, \ldots, x_n]$ $(i = 0, \ldots, m)$ are in Horner form in a fixed variable order $x_1 \prec x_2 \prec \cdots \prec x_n$.

Let $\mathbb{IQ}$ denote the set of intervals whose endpoints are rational numbers and $\mathbb{IQ}^n$ denote a set of intervals as $I_1 \times \cdots \times I_n$, where $I_i \in \mathbb{IQ}$. Let $\mathbb{I} = I_1 \times \cdots \times I_n \in \mathbb{IQ}^n$. Evaluating $\mathbb{I}$ for $x_1, \ldots, x_n$ in h_i $(i = 0, \ldots, m)$, we can derive an interval, say $A_i = h_i(\mathbb{I}) = [a_i, b_i]$. One can find more details on the properties and techniques of interval arithmetics in Moore et al. (2009), Stahl (1995). It is clear that

$h_i(x_0) \in h_i(\mathbb{I}) = A_i$. $h_i(x_0)$ is strictly inside (a_i, b_i) if not all $a_i = b_i$ for $i = 0, \ldots, m$. We can derive an interval polynomial for f related to $\mathbb{I}$.

$$f_{\mathbb{I}}(x) = f(\mathbb{I}, x) = \sum_{i=0}^{m} A_i x^i = \sum_{i=0}^{m} [a_i, b_i] x^i.$$

Consider $\mathbb{V}_{\mathbb{R}}(f(x_1, \ldots, x_n, x))$ in the region $\mathbb{I} \times [0, +\infty]$. Note that we can get the related information of $\mathbb{V}_{\mathbb{R}}(f(x_1, \ldots, x_n, x))$ in the region $\mathbb{I} \times [-\infty, 0]$ by considering $f(x_1, \ldots, x_n, -x) = 0$ in the region $\mathbb{I} \times [0, +\infty]$. Denote

$$f_{\mathbb{I}}^u(x) = b_0 + b_1 x + \ldots + b_m x^m, \qquad f_{\mathbb{I}}^d(x) = a_0 + a_1 x + \ldots + a_m x^m.$$

We can find that $f_{\mathbb{I}}^u(x)$, $f_{\mathbb{I}}^d(x)$ are the bounding polynomials of the interval polynomial $f(\mathbb{I}, x)$, that is, the region defined by $f(\mathbb{I}, x) = 0$ is bounded by $f_{\mathbb{I}}^u(x) = 0$, $f_{\mathbb{I}}^d(x) = 0$.

The following inequality holds (see Cheng et al., 2009).

$$\frac{\partial^k f_{\mathbb{I}}^d(x)}{\partial x^k} \le \frac{\partial^k f(x_0, x)}{\partial x^k} \le \frac{\partial^k f_{\mathbb{I}}^u(x)}{\partial x^k}, \quad \forall x \ge 0,\ \forall x_0 \in \mathbb{I},\ k = 0, 1. \tag{1}$$

The inequalities are strict if x_0 is strictly inside $\mathbb{I}$ with $\mathbb{I}$ does not correspond to a point.

Definition 1. We call an open interval (s, t) **a real root** of $f_{\mathbb{I}}(x) = 0$ if

(1) s, t $(s < t)$ are real root(s) of $f_{\mathbb{I}}^u(x) f_{\mathbb{I}}^d(x) = 0$ or $0, +\infty$;
(2) $\mathrm{sign}(f_{\mathbb{I}}^u(x))\mathrm{sign}(f_{\mathbb{I}}^d(x)) < 0$, $\forall x \in (s, t)$.

Lemma 2. *Use the same notations as above. Any real root of $f(x_0, x) = 0$ is inside some real root of $f_{\mathbb{I}}(x) = 0$ for $x_0 \in \mathbb{I}$.*

Proof. Let $\bar{x} \ge 0$ be a real root of $f(x_0, x) = 0$. By (1), $f_{\mathbb{I}}^d(\bar{x}) < f(x_0, \bar{x}) = 0 < f_{\mathbb{I}}^u(\bar{x})$. Thus $\bar{x}$ is in some real root of $f_{\mathbb{I}}(x) = 0$. □

The lemma shows that all the real roots of $f(x_0, x) = 0$ are contained in the real roots of $f_{\mathbb{I}}(x) = 0$.

Definition 3. (See Cheng et al., 2009.) We call $f_{\mathbb{I}}(x)$ **monotonous** in its real root (s, t) if

$$\begin{cases} s \in \mathbb{V}_{\mathbb{R}}(f_{\mathbb{I}}^u), \quad t \in \mathbb{V}_{\mathbb{R}}(f_{\mathbb{I}}^d), \quad \text{and} \quad \mathbb{V}_{\mathbb{R}}\left(\dfrac{\partial f_{\mathbb{I}}^d}{\partial x}\right) \cap (s, t) = \emptyset, \quad (*) \\ \text{or} \\ s \in \mathbb{V}_{\mathbb{R}}(f_{\mathbb{I}}^d), \quad t \in \mathbb{V}_{\mathbb{R}}(f_{\mathbb{I}}^u), \quad \text{and} \quad \mathbb{V}_{\mathbb{R}}\left(\dfrac{\partial f_{\mathbb{I}}^u}{\partial x}\right) \cap (s, t) = \emptyset. \quad (**) \end{cases}$$

Note that $(*)$ means the bounding polynomial $f_{\mathbb{I}}^d(x)$ is strictly increasing in (s, t) and $(**)$ means the bounding polynomial $f_{\mathbb{I}}^u(x)$ is strictly decreasing in (s, t).

Lemma 4. *Use the same notations as above. If $f(\mathbb{I}, x)$ is* **monotonous** *in (s, t), then $f(x_0, x) = 0$ has exactly one real root in (s, t) for any $x_0 \in \mathbb{I}$.*

Proof. At first, we prove that there exists one real root. Assume that $(*)$ holds, the proof for $(**)$ is similar. For any $x_0 \in \mathbb{I}$, since (1) holds, $f(x_0, s) < f_{\mathbb{I}}^u(s) = 0$ and $f(x_0, t) > f_{\mathbb{I}}^d(t) = 0$. Thus $f(x_0, x) = 0$ has real roots in (s, t). We will prove that there is only one real root. Since $V(\frac{\partial f_{\mathbb{I}}^d}{\partial x}) \cap (s, t) = \emptyset$, $f_{\mathbb{I}}^d(x)$ is monotonous in (s, t). From (1), we know $f(x_0, x)$ is also monotonous in (s, t) (see the detailed proof in Cheng et al., 2009). Thus it has only one real root in (s, t). □

Now we construct an effective version for the real roots of $f_{\mathbb{I}}(x)=0$. We will use rational numbers a, b to replace algebraic numbers s, t such that $(s, t) \subset [a, b]$. We will show how to construct the effective roots in $[0, \infty)$ with the following algorithm.

Algorithm 1. Compute the effective real roots of $f_{\mathbb{I}}(x)=0$.
Input: $f_{\mathbb{I}}(x)$.
Output: the effective real roots of $f_{\mathbb{I}}(x)=0$.

(1) Isolate the real zeros of $f^d_{\mathbb{I}}(x)$ and $f^u_{\mathbb{I}}(x)$, denoted by $\mathcal{I}^d = \{I^d_i = [a^d_i, b^d_i] \mid i = 1, \dots, m_1\}$ and $\mathcal{I}^u = \{I^u_i = [a^u_i, b^u_i] \mid i = 1, \dots, m_2\}$ respectively. Assume $\mathcal{I}^d \cup \mathcal{I}^u = \{[\bar{a}_i, \bar{b}_i] \mid i = 1, \dots, m\}$, where $0 \le \bar{a}_1 \le \bar{b}_1 < \cdots < \bar{a}_i \le \bar{b}_i < \cdots < \bar{a}_m \le \bar{b}_m$.

(2) If $f^d_{\mathbb{I}}(0) f^u_{\mathbb{I}}(0) \le 0$, add $[0, 0]$ as the first element of $\mathcal{I}^d \cup \mathcal{I}^u$ if it is not contained in and $f^d_I(\frac{\bar{a}_1}{2}) f^u_I(\frac{\bar{a}_1}{2}) \ge 0$; set $\bar{a}_1 := 0$ if $f^d_I(\frac{\bar{a}_1}{2}) f^u_I(\frac{\bar{a}_1}{2}) < 0$.

(3) Denote $J := [\bar{a}_1, \infty)$. For i from 1 to $m-1$, do
Denote $c_i := \frac{\bar{b}_i + \bar{a}_{i+1}}{2}$. If $f^d_{\mathbb{I}}(c_i) f^u_{\mathbb{I}}(c_i) > 0$, then delete the open interval $(\bar{b}_i, \bar{a}_{i+1})$ from J, that is $J := J \setminus (\bar{b}_i, \bar{a}_{i+1})$.
Denote $c_m = b_m + 1$. If $f^d_{\mathbb{I}}(c_m) f^u_{\mathbb{I}}(c_m) > 0$, $J := J \setminus (\bar{b}_i, \infty)$. Else, compute a bound on x, say b, $J := J \setminus (b, \infty)$.

(4) After this process, the obtained interval set $J \triangleq \{[\tilde{a}_i, \tilde{b}_i] \mid i = 1, \dots, m_0\}$ is the effective roots of $f_{\mathbb{I}}(x)=0$. Output J.

The correctness and termination of the algorithm are clear. We would like to mention that when $f^d_{\mathbb{I}}(c_m) f^u_{\mathbb{I}}(c_m) < 0$ in Step 3, the signs of the leading coefficients of $f^d_{\mathbb{I}}(x)$, $f^u_{\mathbb{I}}(x)$ are different. We can check that whether x_0 vanishes at the leading coefficient of f w.r.t. x easily for the case f is a bivariate polynomial. Then we can remove the leading term of f w.r.t. x when we construct the interval polynomial for $\mathbb{I}$. In doing so, we can ensure that x_0 does not vanish at the new polynomial related to f. Thus we can ensure that the leading coefficients of $f^d_{\mathbb{I}}(x)$, $f^u_{\mathbb{I}}(x)$ have the same sign. Note that sometimes a refinement of $\mathbb{I}$ may be necessary. In fact, a similar checking can be done for the multivariate case though it is much complicated than the bivariate case. But for all the case, we can compute a univariate polynomial in x by resultant computation to get its largest positive root as the bound.

Let $\Sigma = \{f_1, \dots, f_m\}$ be a zero-dimensional polynomial system. $\mathbb{I} = I_1 \times \cdots \times I_{n-1}$ is an isolating interval for a real root $\alpha = (\alpha_1, \dots, \alpha_{n-1})$ of an $(n-1)$ projection system of Σ (see Section 5), where the leading coefficients of f_i's in x_n are not all vanishing on α. Otherwise, a linear coordinate transformation on Σ can avoid it. Let $J_1, \dots, J_k$ be the intersection of the effective real roots of $f_i(\mathbb{I}, x_n) = 0$, $i = 1, \dots, m$. Thus J_i are bounded. We call $I_1 \times \cdots \times I_{n-1} \times J_j$, $j = 1, \dots, k$ the **real root candidates** of $\Sigma = 0$ (w.r.t. α).

3. Generic position

In this section, we will show how to compute an s such that a shear mapping

$$\varphi_{s,n} : (x_1, \dots, x_{n-2}, x_{n-1}, x_n) \to (x_1, \dots, x_{n-2}, x_{n-1} - s x_n, x_n)$$

on a zero-dimensional polynomial system is in a generic position w.r.t. x_{n-1}, x_n (see Definition 6).

At first, we will consider a bivariate polynomial system. Let $f, g \in \mathbb{Z}[x, y]$ such that $\gcd(f, g) = 1$. We say the system $\{f, g\}$ is in a **generic position** w.r.t. y if

1) The leading coefficients of f and g w.r.t. y have no common factors.

2) Let h be the resultant of f and g w.r.t. y. For any $\alpha \in \mathbb{C}$ such that $h(\alpha) = 0$, $f(\alpha, y), g(\alpha, y)$ have only one common zero in $\mathbb{C}$.

Since we isolate the real roots of the system, the condition $\alpha \in \mathbb{C}$ can be revised as $\alpha \in \mathbb{R}$ in this paper.

Before we show how to compute a generic position of a system w.r.t. two variables, we will introduce some notations. Let π_i $(1 \le i < n)$ be the projection map:

$$\pi_i : (z_1, \ldots, z_n) \longrightarrow (z_1, \ldots, z_i). \tag{2}$$

Then for a polynomial system $\Sigma \subset \mathbb{Z}[x_1, \ldots, x_n]$, we denote

$$\pi_i(\Sigma) = \Sigma \cap \mathbb{Z}[x_1, \ldots, x_i],$$

that is, the polynomial set in the ideal generated by Σ with only the variables $x_1, \ldots, x_i$.

Let $J_i = [a_i, b_i] \times [c_i, d_i] \in \mathbb{IQ}^2$, $i = 1, 2$. We simply denote $\varphi_{s,2}(f(x, y)) = f(x + sy, y)$. Taking the map on J_i, we have

$$\varphi_{s,2}(J_i) = \begin{cases} [a_i - sc_i, b_i - sd_i] \times [c_i, d_i], & s \le 0, \\ [a_i - sd_i, b_i - sc_i] \times [c_i, d_i], & s > 0, \end{cases}$$

and denote

$$\pi_1\big(\varphi_{s,2}(J_i)\big) = \begin{cases} [a_i - sc_i, b_i - sd_i], & s \le 0, \\ [a_i - sd_i, b_i - sc_i], & s > 0. \end{cases} \tag{3}$$

We say an s is **generic** w.r.t. J_1, J_2 if $\pi_1(\varphi_{s,2}(J_1)) \cap \pi_1(\varphi_{s,2}(J_2)) = \emptyset$. We say an interval (set) $S \subset \mathbb{R}$ is **generic** w.r.t. J_1, J_2 if $\forall s \in S$, $\pi_1(\varphi_{s,2}(J_1)) \cap \pi_1(\varphi_{s,2}(J_2)) = \emptyset$.

It is obvious that for any point $P_i \in J_i$, $i = 1, 2$, $\varphi_{s,2}(P_1)$ and $\varphi_{s,2}(P_2)$ will not overlap if s is generic w.r.t. J_1, J_2.

Let $\mathcal{J}$ be a list of finite boxes as J_i. We say an interval set $S \subset \mathbb{R}$ is **non-generic** w.r.t. $\mathcal{J}$ if $\forall s \in S$, $\exists$ two boxes $J_1, J_2 \in \mathcal{J}$, $\pi_1(\varphi_{s,2}(J_1)) \cap \pi_1(\varphi_{s,2}(J_2)) \ne \emptyset$. We call also S a non-generic interval set w.r.t. $\mathcal{J}$.

In order to compute S, we need to compute a non-generic interval set related to J_1, J_2, which can be achieved by solving the inequalities related to $\pi_1(\varphi_{s,2}(J_1)) \cap \pi_1(\varphi_{s,2}(J_2)) = \emptyset$. We will show an example to illustrate it.

Example 5. We will show how to compute a non-generic interval set for two boxes $J_i \in \mathbb{IQ}^2$, $i = 1, 2$, where $J_1 = [1, 2] \times [3, 4]$, $J_2 = [5, 6] \times [10, 11]$. When $s \le 0$, $T_1 = \pi_1(\varphi_{s,2}(J_1)) = [1 - 3s, 2 - 4s]$, $T_2 = \pi_1(\varphi_{s,2}(J_2)) = [5 - 10s, 6 - 11s]$ and $2 - 4s < 5 - 10s$. Thus $T_1 \cap T_2 = \emptyset$. When $s > 0$, $T'_1 = \pi_1(\varphi_{s,2}(J_1)) = [1 - 4s, 2 - 3s]$, $T'_2 = \pi_1(\varphi_{s,2}(J_2)) = [5 - 11s, 6 - 10s]$. The conditions that $T'_1 \cap T'_2 = \emptyset$ are $2 - 3s < 5 - 11s$ or $6 - 10s < 1 - 4s$. Solving them, we have $0 < s < 3/8$ or $s > 5/6$. Thus the condition that $T'_1 \cap T'_2 \ne \emptyset$ is $3/8 < s < 5/6$. So the generic interval set for J_1, J_2 is $[\![3/8, 5/6]\!]$. And the non-generic interval set is $[(-\infty, 3/8), (5/6, +\infty)]$.

Definition 6. We say a zero-dimensional polynomial system $\Delta \subset \mathbb{Z}[z_1, \ldots, z_n, x, y]$ is in **a generic position w.r.t.** x, y **in order** $x \prec y$ (generic position to x, y for short) if for any (complex) zero P of $\pi_n(\Delta)$, all the (complex) zeros of the system Δ on P have distinct x-coordinates.

For the definition above, since we consider only real roots of the system in this paper, we can revise the condition as $\forall P \in \mathbb{V}_{\mathbb{R}}(\pi_n(\Delta))$, (P, α_1, α_2) is a root of Δ and $\alpha_1 \in \mathbb{R}$, there is only one common complex root of Δ on the fiber $(x_1, \ldots, x_n, x) = (P, \alpha_1)$.

Let $\beta \in \mathbb{V}_{\mathbb{R}}(\pi_{n-2}(\Sigma))$ and $\mathbb{I}$ the isolating interval for β. γ_i, $i = 1, \ldots, k$ are all the real roots of $\pi_{n-1}(\Sigma)$ at β and $\mathbb{I} \times J_i$ are the corresponding isolating intervals of (β, γ_i). Let $\mathbb{I} \times J_i \times K_{i,j}$ be all the real root candidates of Σ w.r.t. β, where $i = 1, \ldots, k$, $j = 1, \ldots, t_i$, t_i $(1 \le i \le k)$ are positive integers. We can compute a non-generic interval set w.r.t. $\{J_i \times K_{i,j}\}$, denoted as S_β. We take the union of this kind of intervals for all possible $\beta \in \mathbb{V}_{\mathbb{R}}(\pi_{n-2}(\Sigma))$. We can get a non-generic interval set

$$S = \bigcup_{\beta \in \mathbb{V}_{\mathbb{R}}(\pi_{n-2}(\Sigma))} S_\beta. \tag{4}$$

Since the root candidates are finite and bounded, $\mathbb{R} \setminus S \ne \emptyset$ if the isolating boxes are not very big. We can refine the isolating boxes if needed. Our aim is to choose an $s \in \mathbb{R} \setminus S$ such that the bitsize of s is as small as possible. The reason is that when taking a shear mapping on f_i $(i=1,\ldots,n)$, the bitsizes of the coefficients of $\varphi_{s,n}(f_i)$ are expected to be as small as possible. Thus the time (or you can say, the bit complexity) of computing resultants and the roots of the univariate polynomial equations is short (low). A possible way is that choose a rational number s in $\mathbb{R}\setminus S$ such that its bitsize is as small as possible. That is,

$$0\neq s\in\mathbb{Q}\setminus S,\quad \text{and}\quad \mathcal{L}(s)\leq\mathcal{L}(t),\quad \forall t\in\mathbb{Q}\setminus S, \tag{5}$$

where $\mathcal{L}(a)$ is the maximal bitsize of the numerator and the denominator of $a\in\mathbb{Q}$. Of course, choose the best s as (5) is not easy. We can choose one that looks good. Usually, we can choose s as below:

$$0\neq s\in\mathbb{Z}\setminus S,\quad \text{and}\quad |s|\leq|t|,\quad \forall t\in\mathbb{Z}\setminus S.$$

We would like to mention that since $\{J_i\times K_{i,j}\}$ contain all the real roots of Σ at β, the real roots of $\varphi_{s,n}(\Sigma)$ at β do not overlap when projected to x_{n-1}-axis. So the method presented here computes a generic position w.r.t. all the real roots. But it is not a guaranteed generic position for all the complex roots since we compute only the real roots. Of course, we can compute a guaranteed generic position by computing the isolating interval of all the complex roots with the method in Cheng et al. (2012). But the aim of this paper is to find all the real roots of the given system efficiently. With the method above, the roots of the system is probability 1 in a generic position w.r.t. x_{n-1}, x_n in order $x_{n-1}\prec x_n$. The reason is that there may exist a fiber $(x_1,\ldots,x_{n-1})=(\beta,\gamma_i)$ such that $f_j(\beta,\gamma_i,x_n)=0$ for $j=1,\ldots,n$ have common conjugate complex roots. Thus when we do certification of the real root candidates, some empty candidates may be regarded as containing real roots. But most of this case can avoid when we compute the root candidates by interval arithmetic.

The following lemma is obvious.

Lemma 7. *Let $\Sigma\subset\mathbb{Z}[x_1,\ldots,x_n]$ $(n\geq 2)$. If we compute an integer s as above from its real root candidates, then $\varphi_{s,n}(\Sigma)$ is in a generic position w.r.t. x_{n-1}, x_n in order $x_{n-1}\prec x_n$ with probability 1, where*

$$\varphi_{s,n}:=(x_1,\ldots,x_{n-1},x_n)\rightarrow(x_1,\ldots,x_{n-1}-sx_n,x_n).$$

For a bivariate polynomial system, we can compute an s satisfying (5) to derive a new system $\varphi_{s,2}(f,g)$ in a generic position.

Except for computing all the complex roots of the system to get a guaranteed generic position, there is another method to check whether a sheared bivariate system $\Sigma_s=\{f(x+sy,y), g(x+sy,y)\}$ is in a generic position or not. Let

$$\bar{R}_s(x)=\mathrm{Res}_y\big(f(x+sy,y), g(x+sy,y)\big). \tag{6}$$

Denote its square free part as $R_s(x)$. The discriminant of $R_s(x)$ with respect to x is denoted as $W(s)$. If $0\neq s_0\notin V_{\mathbb{R}}(W)$, then Σ_{s_0} is in a generic position.

We can modify the method as below.

Lemma 8. *Use the notations as before. Σ_{s_0} is in a generic position if $R_{s_0}(x)$ (the content is assumed to be 1) is squarefree.*

Proof. It is clear that $\gcd(R_{s_0}(x), \frac{\partial R_{s_0}(x)}{\partial x})=1$ if $R_{s_0}(x)$ is squarefree, which means s_0 is not a zero of the discriminant of $R_s(x)$ w.r.t. x. So the lemma is proved. □

The following corollary is obvious from Lemma 8.

Corollary 9. *A zero-dimensional polynomial system $\{f,g\}\subset\mathbb{Z}[x,y]$ is in a generic position if $\mathrm{Res}_y(f,g)$ is squarefree and the leading coefficients of f and g w.r.t. y have no common factors.*

It is a special case for a bivariate system. The roots of the system are simple and do not overlap when projected to x-axis.

4. Bivariate systems solving

In this section, we will consider a zero-dimensional bivariate polynomial system $\{f, g\} \subset \mathbb{Z}[x, y]$. If it is not zero-dimensional, $\gcd(f, g)$ is not a constant and $\mathrm{Res}_y(f, g) = 0$.

The following lemma is deduced from Section 1.6 of Fulton (1984).

Lemma 10. *Let $f, g \in \mathbb{Z}[x, y]$ be in a generic position w.r.t. y, $\gcd(f, g) = 1$ and $R_1 = \mathrm{Res}_y(f, g)$, then π_1 is a one-to-one and multiplicity-preserving map from $\{f, g\}$ to R_1.*

One can find the definition of multiplicity in §2, Chapter 4 of Cox et al. (1998). The lemma tells us that a zero (x_0, y_0) of $\{f, g\}$ has the same multiplicity as x_0 in $R_1 = 0$. We can directly derive the corollary below from Lemma 10.

Corollary 11. *Let $\Sigma = \{f, g\} \subset \mathbb{Z}[x, y]$ be zero-dimensional. If we compute s as (5), $\Sigma' = \varphi_{s,2}(\Sigma) = \{f(x+sy, y), g(x+sy, y)\}$ are probability 1 in a generic position w.r.t y. If Σ' is in a generic position, the real root(s) of $\pi_1(\Sigma')$ in $J - sK$ exactly corresponds to the real root(s) of $f = g = 0$ in any real root candidate $J \times K$ of $f = g = 0$ including the multiplicities.*

Proof. A random shearing will put the system in a generic position w.r.t. y with probability 1. So the first part of the corollary is correct. The second part of the corollary is guaranteed by Lemma 10. □

Even when we compute s as (5), the sheared system may not be in a generic position as we mentioned in the last section. Denote $R_2 = \mathrm{Res}_y(\varphi_{s,2}(f), \varphi_{s,2}(g))$. When two conjugate complex roots are common roots of $\varphi_{s,2}(f) = \varphi_{s,2}(g) = 0$ on a fiber $x = \alpha$ ($R_2(\alpha) = 0$ and $\alpha \in \mathbb{R}$), and $\alpha \in \pi_1(\varphi_{s,2}(J \times K))$ for some real root candidate $J \times K$ of $f = g = 0$, $J \times K$ will be regarded to be containing a real root even if it does not. Then there may be an error since we consider only the real roots of the system and ensure only that all the real roots (not all complex roots) of $\varphi_{s,2}(\Sigma)$ are in "a generic position" (not overlap when projected to x-axis). But we can use Lemma 8 to ensure that the systems $\varphi_{s,2}(\Sigma)$ are in a generic position (for all the roots with real x-coordinates). It is similar for the multivariate case.

Let

$$R_2(x) = \prod_{i=1}^{m} r_i(x)^i,$$

where, $r_i(x)$ is the factor of $R_2(x)$ with power i and m is the highest power of the factors in $R_2(x)$. By Corollary 11, the corresponding real roots of $\mathbb{V}_{\mathbb{R}}(\Sigma)$ to the real roots $0 = r_i(x)$ have multiplicity i if the system is in a generic position.

Now we will show how to identify the roots in $J - sK$. The case when there is no root or one root of $R_2(x) = 0$ in $J - sK$ is simple. We will show how to deal with the case that two or more real roots are inside $J - sK$. That means there exist two or more real roots of $f = g = 0$ in $J \times K$. We need to construct the corresponding isolating boxes for them. Assume that there are two real roots of $R_2(x) = 0$ in $J - sK$ and $I_i = [a_i, b_i]$ ($b_1 < a_2, i = 1, 2$) are their isolating intervals. The case for more than two real roots is similar. Assume that the corresponding isolating boxes of the roots of $f = g = 0$ are $J \times K_i$, $i = 1, 2$. Since we know $J - sK_i = I_i$, $K_i = -(I_i - J)/s$. Let $J = [c, d]$. We need to ensure that K_i ($i = 1, 2$) are disjoint. It is not difficult to find that the condition is satisfied if $a_2 - b_1 > d - c$ (one can find the proof from Cheng et al., 2009). We can refine J if needed to get the isolating boxes for the roots inside $J \times K$.

We can also have an algebraic representation for the roots of the system: linear univariate representation. The representation has a little difference with the original representation as in Cheng et al. (2012). Let $\mathcal{I} = \{I_i \times J_i, i = 1, \ldots, m\}$ be the set of all the isolating boxes of the roots of the system. If we have computed an s (there exists the case that s is not necessary), the linear univariate representation of a bivariate system is

$$\left\{\mathcal{I}, \left(\alpha, \frac{\beta-\alpha}{-s}\right), R_1, R_2 \in \mathbb{Z}[x] \;\middle|\; R_1(\alpha) = R_2(\beta) = 0\right\}.$$

Based on the analysis above, we have the following algorithm for isolating real roots of a zero-dimensional bivariate system.

Algorithm 2. Isolate the real roots of a zero-dimensional bivariate polynomial system.
Input: $f, g \in \mathbb{Z}[x, y]$ such that $\gcd(f, g) = 1$.
Output: the isolating intervals of the real roots of $f = g = 0$ as well as the multiplicities of the corresponding roots.

(1) Compute $R_1 := \mathrm{Res}_y(f, g)$.
(2) Isolate the real roots of $R_1 = 0$.
(3) For each real root isolating interval I of $R_1 = 0$, compute real root candidates of $\{f, g\}$ with Algorithm 1.
(4) For all real root candidates of $\{f, g\}$, compute s as (5).
(5) Compute $R_2 := \mathrm{Res}_y(f(x + sy, y), g(x + sy, y)) = \prod_{i=1}^{m} r_i(x)^i$.
(6) For each real root candidate $J \times K$ of $\{f, g\}$, the real root(s) of $R_2 = 0$ in the interval $J - sK$ correspond(s) to the real root(s) of $f = g = 0$ in $J \times K$. Separate $J \times K$ into several isolating boxes if it contains several roots.
(7) Return all the isolating boxes with the multiplicities of the corresponding roots of the system.

Remarks for the algorithm:

(1) The termination of the algorithm is clear. The correctness of the algorithm is guaranteed by Corollary 11 with probability 1.
(2) We can choose an s_0 such that Lemma 8 holds after Step 4 and set $R_2(x) = \bar{R}_{s_0}(x)$ to replace Step 5. Then the revised algorithm is certified.
(3) Let $T(x) = \gcd(R_1, R_2)$. Then on the fiber $x = \alpha \in \mathbb{V}_{\mathbb{R}}(T(x))$, the system has real root $(\alpha, 0)$. We can easily find this from the linear coordinate transformation since $\alpha + s0 = \alpha$.
(4) For some system $\Sigma = \{f, g\}$, if it is in a generic position and satisfies certain conditions, we can identify its real roots without shearing the system. Thus we can stop at Step 3. The following lemma shows the result.

Lemma 12. *Let f, g be in a generic position w.r.t. y, $\alpha \in V_{\mathbb{R}}(\mathrm{Res}_y(f, g))$ and J the isolating interval of α. If there is only one root candidate $J \times K$ and, $f(J, y)$ or $g(J, y)$ is monotonous in K, then $\{f, g\}$ has at most one real root on the fiber $x = \alpha$.*

Proof. It is clear that the possible common real roots of $f = g = 0$ on the fiber $x = \alpha$ appear in $J \times K$. From Lemma 4, $f(\alpha, y) = 0$ or $g(\alpha, y) = 0$ has and only has one real root in K. Thus $f = g = 0$ has at most one real root in $J \times K$. □

This lemma gives a speedup of our algorithm without computing the second resultant. If on each fiber $x = \alpha$ (where $\alpha \in V_{\mathbb{R}}(\mathrm{Res}_y(f, g))$), the condition in Lemma 12 holds, $\{f, g\}$ can be regarded as in a generic position. And $J \times K$ is regarded as an isolating box of a real root of $f = g = 0$. Note that there may exist the case that two conjugate complex roots have the same real x coordinate α and the real root candidate of $f = g = 0$ on the fiber $x = \alpha$ has no real root(s). Then there is an error. But this case seldom happens. One special case is guaranteed by Corollary 9.

Example 13. Isolate the real roots of the system $\Sigma = \{f, g\}$, where $f = x^2 + y^2 - 2$, $g = (x - 2y^2)^2 - 2$. Following Algorithm 2, we have

(1) Compute the resultant of f, g, we have $R_1(x) = (4x^2 + 4x - 7)^2(x^2 - 2)^2$.

(2) Isolate the real roots of $R_1 = 0$ with precision 2^{-10}, we have $II = [[-\frac{1961}{1024}, -\frac{245}{128}], [-\frac{1449}{1024}, -\frac{181}{128}], [\frac{117}{128}, \frac{937}{1024}], [\frac{181}{128}, \frac{1449}{1024}]]$.

(3) For each $I \in II$, compute the real root candidates of $f = g = 0$, we have the candidates below. $[[[-\frac{1449}{1024}, -\frac{181}{128}], [-\frac{72\,905}{8\,388\,608}, \frac{72\,905}{8\,388\,608}]], [[\frac{117}{128}, \frac{937}{1024}], [-\frac{70\,721}{65\,536}, -\frac{141\,401}{131\,072}]], [[\frac{117}{128}, \frac{937}{1024}], [\frac{141\,401}{131\,072}, \frac{70\,721}{65\,536}]], [[\frac{181}{128}, \frac{1449}{1024}], [-\frac{42\,621}{2\,097\,152}, \frac{42\,621}{2\,097\,152}]]]$.

(4) Compute S as (4), we have $S = [[-\infty, -\frac{2\,3724\,032}{243\,389}], [-\frac{19\,546\,112}{8\,976\,759}, -\frac{19\,529\,728}{9\,125\,193}], [-\frac{1\,050\,624}{2\,219\,795}, -\frac{1\,046\,528}{2\,305\,693}], [-\frac{64}{141\,401}, \frac{64}{141\,401}], [\frac{1\,046\,528}{2\,305\,693}, \frac{1\,050\,624}{2\,219\,795}], [\frac{19\,529\,728}{9\,125\,193}, \frac{19\,546\,112}{8\,976\,759}], [\frac{23\,724\,032}{243\,389}, \infty]]$.

(5) There are two choices for this step. One is a probability 1 algorithm and the other is a certified one.

- From S in the last step, we can choose $s = 1$. Then we compute $R_2 = 4(4x^4 + 8x^3 - 8x^2 - 44x - 7)(x^2 - 2)^2$. And we can denote the square-free part of R_2 as $\bar{R}_2$.
- Or we compute $\bar{R}_s(x) = \mathrm{Res}_y(f(x + sy, y), g(x + sy, y)) = -(x^2 - 2)^2(-16x^4 - 32x^3 + 40x^2 - 8x^2s^2 + 120xs^2 + 56x - 49 + 46s^2 + 31s^4)$, and its square free part is $\tilde{R}_s(x) = (x^2 - 2)(-16x^4 - 32x^3 + 40x^2 - 8x^2s^2 + 120xs^2 + 56x + 31s^4 + 46s^2 - 49)$. Let $s = 1$, we have $\tilde{R}_1(x) = (x^2 - 2)(-4x^4 - 8x^3 + 8x^2 + 44x + 7)$ (removing the content 4). It is squarefree, so we know the system $\Sigma_1 = \{f(x + y, y), g(x + y, y)\}$ is in a generic position from Lemma 8. This guarantees that the final result is certified.

(6) Since $L = [-\frac{1449}{1024}, -\frac{181}{128}] - [-\frac{72\,905}{8\,388\,608}, \frac{72\,905}{8\,388\,608}] = [-\frac{11\,943\,113}{8\,388\,608}, -\frac{11\,789\,111}{8\,388\,608}]$. We can find that $\bar{R_2}$ has different signs at the endpoints of the interval and $\frac{\partial \bar{R_2}}{\partial x} = 0$ has no roots in L. So $[[-\frac{1449}{1024}, -\frac{181}{128}], [-\frac{72\,905}{8\,388\,608}, \frac{72\,905}{8\,388\,608}]]$ is an isolating interval of $f = g = 0$. We can also find that the root in L corresponding to $x^2 - 2$. So its multiplicity is 2. The multiplicity of another related root is also 2. The multiplicities of the left roots are 1. Since $(x^2 - 2) | \gcd(R_1(x), R_2(x))$, $(\pm\sqrt{2}, 0)$ are real roots of the original system from remark (4) of Algorithm 2. Thus $[[-\frac{1449}{1024}, -\frac{181}{128}], [0, 0]]$ is an isolating interval of $f = g = 0$. The other isolating intervals can be identified similarly. Denote all the isolating intervals as K.

$$K = \left[\left[\left[-\frac{1449}{1024}, -\frac{181}{128}\right], [0, 0]\right], \left[\left[\frac{117}{128}, \frac{937}{1024}\right], \left[-\frac{70\,721}{65\,536}, -\frac{141\,401}{131\,072}\right]\right], \left[\left[\frac{117}{128}, \frac{937}{1024}\right], \left[\frac{141\,401}{131\,072}, \frac{70\,721}{65\,536}\right]\right], \left[\left[\frac{181}{128}, \frac{1449}{1024}\right], [0, 0]\right]\right].$$

Then we can get the LUR of the system:

$$\left\{K, \left(\alpha, \frac{\beta - \alpha}{-s}\right), R_1(x), R_2(x) \,\middle|\, R_1(\alpha) = 0, R_2(\beta) = 0\right\}.$$

Now we consider a bivariate zero-dimensional systems with $m(> 2)$ polynomials, we just take $m = 3$ for an illustration, it is similar for the case of $m > 3$. Let $\Sigma = \{f, g, h\}$, where $f, g, h \in \mathbb{Z}[x, y]$. Let $p = \gcd(f, g)$, $f^* = \frac{f}{p}$, $g^* = \frac{g}{p}$. We have $\mathbb{V}_{\mathbb{R}}(f, g, h) = \mathbb{V}_{\mathbb{R}}(f^*, g^*, h) \cup \mathbb{V}_{\mathbb{R}}(p, h)$. Furthermore, let $q = \gcd(g^*, h)$ and $g^{**} = \frac{g^*}{q}$, $h^* = \frac{h}{q}$, then we obtain $\mathbb{V}_{\mathbb{R}}(f^*, g^*, h) = \mathbb{V}_{\mathbb{R}}(f^*, g^{**}, h^*) \cup \mathbb{V}_{\mathbb{R}}(f^*, q)$. Hence, we have

$$\mathbb{V}_{\mathbb{R}}(f, g, h) = \mathbb{V}_{\mathbb{R}}\big(f^*, g^{**}, h^*\big) \cup \mathbb{V}_{\mathbb{R}}\big(f^*, q\big) \cup \mathbb{V}_{\mathbb{R}}(p, h). \tag{7}$$

On the right side of (7), both $\{f^*, q\}$ and $\{p, h\}$ are zero-dimensional, thus we can solve these two systems using Algorithm 2. Now we will show how to solve the system $\{f^*, g^{**}, h^*\}$. Actually, $\mathbb{V}_{\mathbb{R}}(f^*, g^{**}, h^*) = \mathbb{V}_{\mathbb{R}}(f^*, g^{**}) \cap \mathbb{V}_{\mathbb{R}}(g^{**}, h^*)$ and $\{f^*, g^{**}\}$, $\{g^{**}, h^*\}$ are zero-dimensional polynomial systems. Assume that the LUR of the systems $\{f^*, g^{**}\}$, $\{g^{**}, h^*\}$ are

$$\left\{K_1, \left(\alpha, \frac{\beta-\alpha}{-s_1}\right) \,\middle|\, R_{1,1}(\alpha)=0,\ R_{1,2}(\beta)=0\right\} \tag{8}$$

and

$$\left\{K_2, \left(\alpha, \frac{\beta-\alpha}{-s_2}\right) \,\middle|\, R_{2,1}(\alpha)=0,\ R_{2,2}(\beta)=0\right\} \tag{9}$$

respectively. What's more, if we chose the same value for s in Eqs. (8) and (9) (this can be easily achieved), that is $s_1 = s_2$ then we can get the LUR of the system $\{f^*, g^{**}, h^*\}$:

$$\left\{K_1 \cap K_2, \left(\alpha, \frac{\beta-\alpha}{-s}\right) \,\middle|\, R_1(\alpha)=0,\ R_2(\beta)=0\right\}, \tag{10}$$

where $s = s_1 = s_2$, $R_1 = \gcd(R_{1,1}, R_{2,1})$, $R_2 = \gcd(R_{2,1}, R_{2,2})$ and in each isolating box of $K_1 \cap K_2$ there exists only one real root of the system. We can also ensure that we take the same s for $\mathbb{V}_{\mathbb{R}}(f^*, q)$ and $\mathbb{V}_{\mathbb{R}}(p, h)$ when solving them. Thus we can check their real roots are exactly the same or not as the roots of $\{f^*, g^{**}, h^*\}$. In the end, we get all the solutions of the original system $\{f, g, h\}$.

5. Multivariate systems solving

In this section, we will show how to isolate the real roots of a general zero-dimensional polynomial system.

Let $\Sigma_n = \{f_1, \ldots, f_m\} \subset \mathbb{Z}[x_1, \ldots, x_n]$, $m \geq n$ be a zero-dimensional polynomial system. Let $f_i' = \sum_{j=1}^{m} t_{i,j} f_j$ $(1 \leq i \leq n)$, where $t_{i,j} \in \mathbb{Z}$ and the rank of the matrix $(t_{i,j})$ is of full rank n, denoted as $\operatorname{rank}(t_{i,j}) = n$. Let $g_i = \operatorname{Res}_{x_n}(f_i', f_n')$, $i = 1, \ldots, n-1$. Then $\Sigma_{n-1} = \{g_1, \ldots, g_{n-1}\}$ is probability 1 to be a zero-dimensional polynomial system. We call Σ_{n-1} is an $(n-1)$**-projection system** of Σ_n if it is zero-dimensional. We denote $\Sigma_{n-1} = \prod_{n-1}(\Sigma_n)$ and $\prod_i(\Sigma_n) = \prod_i(\prod_{i+1}(\cdots \prod_{n-1}(\Sigma_n)\cdots))$. We want to mention that $\pi_i(\Sigma_n) \subset \prod_i(\Sigma_n)$. Recursively, we can eliminate variables to get a univariate polynomial. Assume that we know how to derive the roots of $\Sigma_i = 0$ since we know how to get the real roots of a univariate polynomial equation or a zero-dimensional bivariate polynomial system. Using the real root isolating intervals of Σ_i, we can compute the real root candidates of $\Sigma_{i+1} = \{p_1(x_1, \ldots, x_{i+1}), \ldots, p_{i+1}(x_1, \ldots, x_{i+1})\} = 0$. Computing s as (5), we can get a new system $\Sigma_{i+1}' = \{p_1(x_1, \ldots, x_{i-1}, x_i + s x_{i+1}, x_{i+1}), \ldots, p_{i+1}(x_1, \ldots, x_{i-1}, x_i + s x_{i+1}, x_{i+1})\}$. Projecting it to obtain a zero-dimensional system $\overline{\Sigma_i}$ in i-space as above, we can isolate its real roots and check whether there exist real roots in the real root candidates of $\Sigma_{i+1} = 0$. Then we get the real root isolating intervals of $\Sigma_{i+1} = 0$. In a recursive way, we can obtain the real root isolating intervals of $\Sigma_n = 0$.

Lemma 14. *Use the notations as above.*

$$\mathbb{V}\left(\prod_{i-1}(\Sigma_{i+1})\right) = \mathbb{V}\left(\prod_{i-1}(\Sigma_{i+1}')\right) \subset \mathbb{V}\left(\prod_{i-1}(\overline{\Sigma_i})\right).$$

Proof. The equality is true since the roots of $\Sigma_{i+1} = 0$ and $\Sigma_{i+1}' = 0$ have a one-to-one map and their corresponding roots differ only on the i-th coordinate. And the inclusion relationship is clear. □

Theorem 15. *Use the notations as above and compute s for the real root candidates of $\Sigma_{i+1} = 0$ as (5). Then Σ_{i+1}' is in a generic position w.r.t. x_i, x_{i+1} in order $x_i \prec x_{i+1}$ with probability 1.*

Proof. We can find that there are many extraneous roots in $\Sigma_i = 0$ corresponding to $\prod_i(\Sigma_{i+1})$. But the number is finite. So are the roots in $\mathbb{C}^{i+1}$. It is similar for $\overline{\Sigma_i} = 0$ and $\prod_i(\Sigma_{i+1}')$. Thus there are only finite complex zeros of Σ_{i+1} in $\mathbb{C}^{i+1}$ such that Σ_{i+1} is not in a generic position w.r.t. x_i, x_{i+1} in order $x_i \prec x_{i+1}$. So we prove the theorem. □

We give the following algorithm to isolate the real roots of a general zero-dimensional polynomial system.

Algorithm 3. Isolate the real roots of a zero-dimensional polynomial system.
Input: $\Sigma_n = \{f_1, \dots, f_m\} \subset \mathbb{Z}[x_1, \dots, x_n]$.
Output: Isolating intervals of the real roots of $\Sigma_n = 0$.

```
(1) Let $f'_i = \sum_{j=1}^{m} t_{i,j} f_j$ $(1 \le i \le n)$, where $t_{i,j} \in \mathbb{Z}$ and $\mathrm{rank}(t_{i,j}) = n$, and $\Sigma_{n-1} = \{g_1, \dots, g_{n-1}\}$, where $g_i = \mathrm{Res}_{x_n}(f'_i, f'_n)$, $i = 1, \dots, n-1$. In a similar way, we can get $\Sigma_{n-2}, \dots, \Sigma_1$.
(2) For $i = 1, \dots, n-1$, do the following computation.
    (a) Isolate the real roots of $\Sigma_i = 0$.
    (b) For each root isolating interval $I = I_1 \times \cdots \times I_i$ of $\Sigma_i = 0$, compute root candidates of $\Sigma_{i+1}$ with Algorithm 1.
    (c) Compute $S_i$ as (4) and choose $s_i$ as (5).
    (d) Assume that $\Sigma_{i+1} = \{p_1, \dots, p_{i+1}\}$. Let $p'_k = \sum_{j=1}^{i+1} t'_{k,j} p_j(x_1, \dots, x_{i-1}, x_i + s_i x_{i+1}, x_{i+1})$, where $t'_{k,j} \in \mathbb{Z}$, $k = 1, \dots, i+1$ and $\mathrm{rank}(t'_{k,j}) = i+1$. $\overline{\Sigma_i} = \{q_1, \dots, q_i\}$, where $q_k = \mathrm{Res}_{x_{i+1}}(p'_k, p'_{i+1})$, $k = 1, \dots, i$.
    (e) Isolate the real roots of $\overline{\Sigma_i} = 0$.
    (f) For each root candidate $I_1 \times \cdots \times I_{i-1} \times I_i \times K$ of $\Sigma_{i+1}$, it is a real root isolating interval if $I_1 \times \cdots \times I_{i-1} \times (I_i - s_i K)$ has non-empty intersection with some real root isolating interval of $\overline{\Sigma_i} = 0$. It should be subdivided into two or more isolating intervals if $I_1 \times \cdots \times I_{i-1} \times (I_i - s_i K)$ has intersection with two or more real root isolating interval of $\overline{\Sigma_i} = 0$ similarly as Step 6 in Algorithm 2.
(3) Output the isolating boxes of $\Sigma_n$.
```

Remarks.

(1) The termination of the algorithm is clear. The algorithm is probability 1 correct. It is guaranteed by Theorem 15.

(2) Now we consider the LUR of Σ_i $(i > 2)$. Assume that we have got the LUR for Σ_i. The univariate polynomials are $T_1(y_1), \dots, T_i(y_i)$. The s_j's are $s_1, \dots, s_{i-1}$. The real root isolating intervals are $\mathbb{I}_k = I_{k,1} \times I_{k,2} \times \cdots \times I_{k,i}$, $k = 1, \dots, p$. We know that $T_2(y_2) = 0$ has a real root in $I_{k,1} - s_1 I_{k,2}$, $T_3(y_3) = 0$ has a real root in $I_{k,1} - s_1(I_{k,2} - s_2 I_{k,3}), \dots,$ and $T_i(y_i) = 0$ has a real root in $\bar{\mathbb{I}}_k = I_{k,1} + \Sigma_{j=1}^{i}(-1)^{j-1} s_1 \cdots s_{j-1} I_{k,j}$. Of course, we require that $\bar{\mathbb{I}}_k$ are disjoint for any k. The zeros of Σ_i can be represented as

$$\left\{\mathbb{I}_k, \left(\alpha_1, \frac{\alpha_2 - \alpha_1}{-s_1}, \dots, \frac{\alpha_i - \alpha_{i-1}}{(-1)^{i-1} s_1 \cdots s_{i-1}}\right), T_t(x) \;\middle|\; T_t(\alpha_t) = 0 \text{ for } t = 1, \dots, i\right\}.$$

Example 16. Let's consider isolating the real roots of $\Sigma = \{f, g, h\} = \{3x - y - 5z - 4, 8x^2 + 8y^2 + z^2 - 8, x^2 + 2y^2 + 4z^2 - 4\}$.

At first, we compute $p = \mathrm{Res}_y(f, g) = 209x^2 + 201y^2 - 184 - 6xy - 24x + 8y$, $q = \mathrm{Res}_y(f, h) = 61x^2 + 54y^2 - 36 - 24xy - 96x + 32y$.

Isolate the real roots of the bivariate polynomial system $\{p, q\}$ with Algorithm 2. Denote its LUR as

$$\left\{K, \left(\alpha, \frac{\beta - \alpha}{-s_1}\right), T_1(x), T_2(x) \;\middle|\; T_1(\alpha) = 0, T_2(\beta) = 0\right\},$$

where $K = [[[-\frac{433}{2048}, -\frac{865}{4096}], [-\frac{15\,433}{16\,384}, -\frac{123\,453}{131\,072}]], [[\frac{95}{256}, \frac{761}{2048}], [\frac{116\,549}{131\,072}, \frac{58\,287}{65\,536}]]]$, $T_1 = 11\,667x^4 + 185\,368x^2 - 24\,960x - 48\,480x^3 - 14\,032$, $T_2 = 35\,001x^4 - 95\,104x + 1\,203\,952x^2 - 393\,936x^3 - 429\,504$. $s_1 = 1$.

For each isolating box of K, compute the real root candidates of $f=g=h=0$. They are

$$U=\Bigg[\bigg[\Big[-\frac{433}{2048},-\frac{865}{4096}\Big],\Big[-\frac{15\,433}{16\,384},-\frac{123\,453}{131\,072}\Big],\Big[-\frac{96\,789}{131\,072},-\frac{96\,779}{131\,072}\Big]\bigg],$$
$$\bigg[\Big[\frac{95}{256},\frac{761}{2048}\Big],\Big[\frac{116\,549}{131\,072},\frac{58\,287}{65\,536}\Big],\Big[-\frac{49\,489}{65\,536},-\frac{98\,955}{131\,072}\Big]\bigg]\Bigg].$$

We can compute a number: $s_2=1$.

The new system is $\{f',g',h'\}=\{3x-y-6z-4, 8x^2+8y^2+16zy+9z^2-8, x^2+2y^2+4zy+6z^2-4\}$. The resultants of f' and g',h' w.r.t. z are $\{p',q'\}=\{90x^2+54y^2-48+36xy-48y-144x, 369x^2+201y^2-144+234xy-312y-216x\}$. Its isolating intervals are $K'=[[[-\frac{433}{2048},-\frac{865}{4096}],[-\frac{106\,759}{524\,288},-\frac{106\,637}{524\,288}]],[[\frac{95}{256},\frac{761}{2048}],[\frac{53\,871}{32\,768},\frac{215\,573}{131\,072}]]]$.

We can check that $(U[1][2]-U[1][3])\cap K'[2]\neq\emptyset$. Thus, $U[1]$ is an isolating box of the original system. Similarly, we can find that U are the isolating boxes of the original system. The LUR is as follows.

$$\big\{U,(\alpha,\alpha-\beta,\gamma-\beta),T_1(x),T_2(x),T_3(x)\;\big|\;T_1(\alpha)=0,T_2(\beta)=0,T_3(\gamma)=0\big\},$$

where $T_3=11\,667x^4+790\,528x+420\,544x^2-139\,632x^3+6144$. The multiplicities of the roots are the same. All are 1.

In fact, for this example, we can use the monotonicity of the interval polynomial to certify the root candidates without computing the second resultant in practice.

6. The algorithm complexity

In this section, we will analyze the complexity of Algorithms 2.

At first, we will introduce some notations. In what follows $\mathcal{O}_B$ means bit complexity and the $\tilde{\mathcal{O}}_B$-notation means that we ignore logarithmic factors. For a polynomial $f\in\mathbb{Z}[X]$, $\deg(f)$ denotes its degree. By $\mathcal{L}(f)$ we denote an upper bound on the bitsize of the coefficients of f (including a bit for the sign), sometimes we also take the conventions in Kerber and Sagraloff (2012) that an integer polynomial is called of magnitude (n,τ) if its total degree is bounded by n, and each integer coefficient is bounded by 2^τ in its absolute value. $\tilde{\mathcal{O}}$ indicates that we omit logarithmic factors. For $a\in\mathbb{Q}$, $\mathcal{L}(a)$ is the maximal bitsize of the numerator and the denominator.

Lemma 17. *Let $f\in\mathbb{Z}[x]$ such that $\deg(f)\le d$, $\mathcal{L}(f)\le\tau$. We can isolate the real roots of f using no more than $\tilde{\mathcal{O}}_B(d^3\tau)$ bit operations (Pan, 2000; Sagraloff, 2012; Schönhage, 1982) or $\tilde{\mathcal{O}}_B(d^2\tau)$ bit operations (Pan, 2002). We can refine all the isolating intervals to a width 2^{-L} or less using $\tilde{\mathcal{O}}_B(d^3\tau+d^2L)$ (Sagraloff, 2012) or a single islating interval to a width 2^{-L} or less using $\tilde{\mathcal{O}}_B(d^2\tau+dL)$ (Pan and Tsigaridas, 2013) bit operations.*

In this paper, we use $\tilde{\mathcal{O}}_B(d^3\tau)$ for real root isolation and $\tilde{\mathcal{O}}_B(d^3\tau+d^2L)$ for refinement of isolating intervals.

Lemma 18. *(See Kerber and Sagraloff, 2012.) Let $f(x)\in\mathbb{Z}[x]$ be a polynomial of $\deg_x(f)\le d$, $\mathcal{L}(f)\le\tau$, and a rational value $\frac{c}{d}$ such that c and d have a bitsize of at most σ, then evaluating $f(\frac{c}{d})$ has a complexity of $\tilde{\mathcal{O}}(d(\tau+d\sigma))$.*

According to the lemma above, we have the following lemma directly:

Lemma 19 *(Rational evaluation for bivariate polynomials). Let $f(x,y)\in\mathbb{Z}[x,y]$ such that $\deg(f)\le d$, $\mathcal{L}(f)\le\tau$, and $\frac{c}{d}$ a rational value such that $\mathcal{L}(c),\mathcal{L}(d)\le\sigma$. Then evaluating $f(\frac{c}{d},y)$ has a complexity of $\tilde{\mathcal{O}}(d^2(\tau+d\sigma))$. Moreover, $\deg(f(\frac{c}{d},y))\le d$, $\mathcal{L}(f(\frac{c}{d},y))\le\mathcal{O}(d\sigma+\tau)$.*

Lemma 20 *(Square-free part). (See Kerber and Sagraloff, 2012.) Let* $g \in \mathbb{Z}[x]$ *such that* $\deg(g) \leq d$ *and* $\mathcal{L}(g) \leq \lambda$*. Its square-free part* g^* *can be computed in* $\tilde{\mathcal{O}}(d^2\lambda)$*. Furthermore,* $\deg(g^*) \leq d$*,* $\mathcal{L}(g^*) \leq \tilde{\mathcal{O}}(d+\lambda)$*.*

One can find the following result in some references, such as Kerber and Sagraloff (2012); Reischert (1997).

Lemma 21. *Let* $f, g \in \mathbb{Z}[x]$ *such that* $\deg(h) \leq d$*,* $\mathcal{L}(h) \leq \tau$ *for* $h = f, g$*. Computing their gcd, denoted as* p*, has a complexity of* $\tilde{\mathcal{O}}(d^2\tau)$ *and* $\deg(p) \leq d$*,* $\mathcal{L}(p) \leq \mathcal{O}(d+\tau)$*.*

Lemma 22. *(See Basu et al., 2003; Mignotte, 1992; Yap, 2000.) Let* $f(x) \in \mathbb{Z}[x]$ *such that* $\deg(f) \leq d$*,* $\mathcal{L}(f) \leq \tau$*. Then the separation bound of* f *is*

$$\operatorname{sep}(f) \geq d^{-\frac{d+2}{2}}(d+1)^{\frac{1-d}{2}}2^{\tau(1-d)},$$

thus $\log(\operatorname{sep}(f)) = \tilde{\mathcal{O}}(d\tau)$*. The latter provides a bound on the bit size of the endpoints of the isolating intervals.*

Lemma 23. *(See Diochnos et al., 2009.) Let* $f, g \in (Z[y_1, \ldots, y_k])[x]$ *with* $\deg_x(f) = p \geq q = \deg_x(g)$*,* $\deg_{y_i}(f) \leq p$ *and* $\deg_{y_i}(g) \leq q$*,* $\mathcal{L}(f) = \tau \geq \sigma = \mathcal{L}(g)$*. We can compute* $\operatorname{Res}_x(f, g)$ *in* $\tilde{\mathcal{O}}_B(q(p+q)^{k+1}p^k\tau)$*. And* $\deg_{y_i}(\operatorname{Res}_x(f, g)) \leq 2pq$*, and the bit size of resultant is* $\tilde{\mathcal{O}}(p\sigma + q\tau)$*.*

The following lemma shows how to compute the non-generic interval set of two isolating boxes. It will be used to bound the bitsize of s.

Lemma 24. *Let* $L_i = J_i \times K_i = [a_i, b_i] \times [c_i, d_i] \in \mathbb{IQ}^2, i = 1, 2$ *be two real root candidates of* $\Sigma = 0$*. The widthes of* J_i, K_i *are bounded such that* $|J_i|, |K_i| \leq 2^{-D^3\tau - D^3}$*, where* $D(>1)$*,* τ *are the degree bound and the bitsize bound of the coefficients of the polynomials. Assume that* $a_1 \leq a_2$*,* $c_1 \leq c_2$*,* $a_2 - b_1 > 2^{-D^3\tau-1}$ *if* $a_1 \neq a_2$*,* $b_1 = b_2$ *if* $a_1 = a_2$*, and* $c_2 - d_1 > 2^{-D^3\tau-1}$ *if* $c_1 \neq c_2$*,* $d_1 = d_2$ *if* $c_1 = c_2$*. Denote the non-generic interval set of* $J_1 \times K_1$ *and* $J_2 \times K_2$ *as* L*. Then either* L *contains at most one integer or the integers inside* L *is larger than* D^6 *(less than* $-D^6$*).*

Proof. There are three cases for the position relationship of L_1, L_2. We will discuss them one by one.

The first case is $a_1 < a_2$ and $c_1 < c_2$. When we choose $s > 0$, $\varphi_{s,2}(L_1) \cap \varphi_{s,2}(L_2) = \emptyset$ is always true. So the non-generic interval set contains only negative s. From formula (3), we have the following inequalities.

$$b_1 + sc_1 \geq a_2 + sd_2, \qquad a_1 + sd_1 \leq b_2 + sc_2.$$

Solving them we have

$$\frac{a_1 - b_2}{c_2 - d_1} \leq s \leq \frac{b_1 - a_2}{d_2 - c_1}.$$

Since $\frac{b_1-a_2}{d_2-c_1} < \frac{b_1-a_2}{c_2-d_1} \leq \frac{a_1-b_2+2^{-D^3\tau-D^3+1}}{c_2-d_1} < \frac{a_1-b_2}{c_2-d_1} + \frac{2^{-D^3\tau-D^3+1}}{2^{-D^3\tau-1}} = \frac{a_1-b_2}{c_2-d_1} + 2^{-D^3+2}$, the non-generic interval set of L_1, L_2 contains at most one integer.

The second case is $a_1 < a_2$ and $c_1 = c_2$. There are two non-generic intervals for L_1, L_2. We consider only $s > 0$ (it is similar for $s < 0$). From formula (3), we have

$$b_1 + sd_1 \geq a_2 + sc_2.$$

Thus $s > \frac{a_2-b_1}{d_1-c_2} > \frac{2^{-D^3\tau-1}}{2^{-D^3\tau-D^3}} = 2^{D^3-1} > D^6$. So $s > D^6$ (similarly, $s < -D^6$ if we choose $s < 0$).

The last case is $a_1 = a_2$ and $c_1 < c_2$. We have the following from formula (3) (considering only $s \geq 0$).

$$b_1 + sd_1 \geq a_2 + sc_2.$$

Thus we have $s \le \frac{b_1-a_2}{c_2-d_1} < \frac{2^{-D^3\tau-D^3}}{2^{-D^3\tau-1}} = 2^{-D^3+1}$. Considering both $s \ge 0$ and $s < 0$, we have $-2^{-D^3+1} < s < 2^{-D^3+1}$. Thus the non-generic interval set of L_1 and L_2 contains only one integer. The lemma is proved. □

Theorem 25. *Let $\Sigma = \{f, g\} \subset \mathbb{Z}[x, y]$ be a zero-dimensional polynomial system such that* $\deg(h) \le D$, $\mathcal{L}(h) \le \tau$ *for $h = f, g$. Then we can isolate the real roots of $\Sigma = 0$ with the bit complexity $\tilde{O}_B(N^{10})$ based on Algorithm 2, where $N = \max\{D, \tau\}$.*

Proof. Following Algorithm 2, we analyze the bit complexity of each step. For the first step, the bit complexity is $\tilde{O}_B(D^4\tau)$ by Lemma 23.

For Step 2, the bit complexity is $\tilde{O}_B(D^7\tau)$ by Lemma 17. The bitsize of the endpoints of the isolating intervals of Step 2 is $\tilde{O}_B(D^3\tau)$ by Lemma 22.

In Step 3, when we construct the interval polynomials from the isolating intervals derived in Step 2, the bitsizes of the coefficients are bounded by $\tilde{O}(D^4\tau)$. Thus the bit complexity of obtaining the real root candidates of $\Sigma = 0$ is bounded by $\tilde{O}_B(D^3 * D^4\tau) * D^2 = \tilde{O}_B(D^9\tau)$ by Lemma 17. The bitsizes of the y-coordinate of the candidates are bounded by $D * \tilde{O}(D^4\tau) = \tilde{O}(D^5\tau)$. But it can be relaxed to $\tilde{O}(D^3\tau)$ by computing the isolating intervals of the real roots of $\mathrm{Res}_x(f, g) = 0$ directly and identifying them. The width of the isolating intervals can be regarded as $2^{-D\tau}$ by Lemma 17.

In Step 4, the number of the isolating boxes in the real root candidate set $\mathcal{J}$ is bounded by D^3. The bit complexity to compute a non-generic interval set w.r.t. two candidates is $\tilde{O}_B(D^3\tau)$ since the bitsizes of the endpoints of the candidates are $\tilde{O}(D^3\tau)$ by Step 3. The number of the different intervals of the non-generic interval set w.r.t. two candidates is at most two. In fact, for any two candidates, there is only one interval in their non-generic interval set if there y-coordinates are disjoint whatever their x coordinates are the same or not. Otherwise, there are two connected intervals for $s > 0$ and $s < 0$. So the bit complexity to get a non-generic interval set w.r.t. $\mathcal{J}$, that is, any two real root candidates computing non-generic interval set and joining all non-generic interval sets together, is $D^3 * (D^3 - 1)/2 * \tilde{O}_B(D^3\tau) = \tilde{O}_B(D^9\tau)$. The number of the intervals in the non-generic interval set w.r.t. $\mathcal{J}$ is bounded by $\tilde{O}(D^6)$. Note that the bitsizes of the endpoints of the non-generic intervals are also $\tilde{O}(D^3\tau)$. Thus the bit complexity to find a generic s w.r.t. $\mathcal{J}$ is bounded by $\tilde{O}_B(D^9\tau)$. Now let us consider how to bound the bitsize of s. Since Σ has at most D^3 root candidates, the number of the non-generic intervals w.r.t. $\mathcal{J}$ we choose is at most $D^3 * (D^3 - 1)/2 * 2 = D^6 - D^3$. We can refine the real root candidates of $\Sigma = 0$ such that the conditions in Lemma 24 hold. Thus there is at least one generic s w.r.t. $\mathcal{J}$ in D^6 integers. Note that the bit complexity of the refinement does not increase the total complexity by Lemma 17. By the result of Lemma 24, the bitsize of s is bounded by $\log(D^6)$ (at most $O(D^6)$ non-generic s w.r.t. $\mathcal{J}$). So in Step 5, the complexity is $\tilde{O}_B(D^4\tau)$.

In Step 6, isolating the real roots of $R_2(x) = 0$ is bounded by $\tilde{O}_B(D^7\tau)$. We deal with the case that one candidate contains more than one root. Let the separation bound of $R_2 = 0$ be L. We can refine the isolating intervals of $R_2 = 0$ to $L/4$ and the isolating intervals of $R_1 = 0$ to $L/2$, the condition to ensure that the isolating boxes of the system are disjoint. The separation bound of the roots of $R_2 = 0$ is $\tilde{O}(D^3\tau)$. From the results in Sagraloff (2012), the refinements of the isolating intervals of both $R_1 = 0$ and $R_2 = 0$ are bounded $\tilde{O}_B(D^5\tau)$.

So the total complexity of the algorithm is bounded by $\tilde{O}_B(D^9\tau)$. □

For the certified version of Algorithm 2 in its remark, the bit complexity of computing $\bar{R}_s(x)$ and $R_s(x)$ is bounded by $\tilde{O}_B(D^6\tau)$ according to Lemma 20 and Lemma 23. We have $\deg(R_s(x)) \le D^2$, $\mathcal{L}(R_s(x)) \le D^2 + 2D\tau$. Its discriminant $W(s)$ has a degree at most D^4. Thus, if we chose a rational value s, it takes at most D^4 times such that $\gcd(R_s(x), \frac{\partial R_s(x)}{\partial x}) = 1$. So the bitsize of s_0 in Lemma 8 is $4\log D$, and the bit complexity of evaluating $R_s(x)$ at $s = s_0$ is bounded by $\tilde{\mathcal{O}}((D^2)^2(D^2 + 2D\tau + D^2 * 4\log D)) = \tilde{\mathcal{O}}(D^5(D + \tau))$ according to Lemma 19. We also have $\deg(R_{s_0}(x)) \le D^2$, $\mathcal{L}(R_{s_0}(x)) \le D^2\log D + D^2 + 2D\tau$ which leads to the bit complexity of computing $\gcd(R_{s_0}(x), \frac{\partial R_{s_0}(x)}{\partial x})$ is $\tilde{\mathcal{O}}((D^2)^2(D^2\log D + D^2 + 2D\tau)) = \tilde{\mathcal{O}}(D^5(D + \tau))$. The complexity of the left part is bounded by $\tilde{O}_B(D^{10} + D^9\tau)$. So the complexity is also $\tilde{O}_B(N^{10})$.

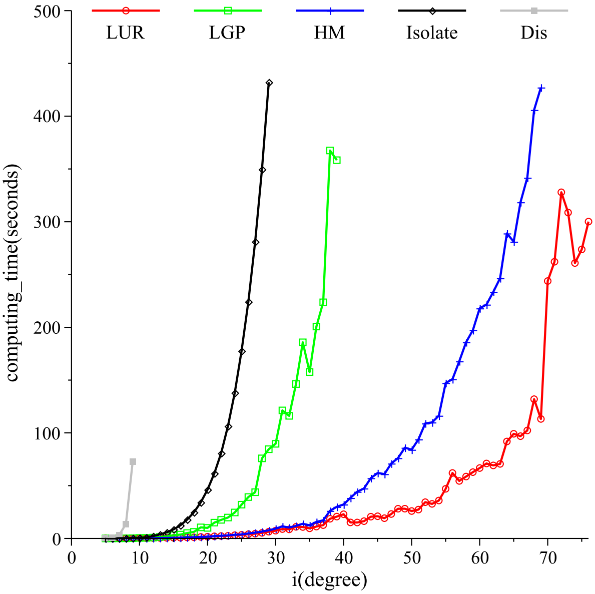

Fig. 1. Timings for the system $\{f, g\}$ with simple roots, where $f, g = randpoly([x, y],\ degree = i,\ coeffs = rand(-100..100),\ dense)$.

For a general zero-dimensional polynomial system $\Sigma = \{f_1, \ldots, f_m\} \subset \mathbb{Z}[x_1, \ldots, x_n]$, it is not difficult to find that our method is double exponential.

7. Experiments

In this section, we compare our algorithm with some existing methods, especially with the efficient ones. We implement our algorithm in Maple. For the univariate solver, we can use Emiris et al. (2008), Rouillier and Zimmermann (2003). We use Rouillier and Zimmermann (2003) in Maple. We mainly compare with some bivariate system solvers. We compare our algorithm (in Maple), named LUR, with local generic position (LGP, in Maple) (Cheng et al., 2009), Hybird method (HM, in Maple) (Hong et al., 2008), Discovery (Dis, in Maple) (Xia and Yang, 2002) and Isolate (the core is in C) in Maple (Rouillier, 1999). The readers who are interested in our code can ask for the code from the corresponding author.

We have four groups of examples. Each example is a set $\{f, g\}$ of two random dense polynomials. We get the timings from a PC with 2Quad CPU 2.66G Hz, 3.37G memory and Windows XP operating system. We stop the computation for each solver and each system when the computing time is larger than 500 seconds. For each case we consider 10 examples for all solvers and get their average computing time.

For the four groups, we mainly test the influences of the degree, the multiple roots, the sparsity and the bitsizes of the coefficients of the input polynomials to the different solvers. The results are shown in Figs. 1–4 respectively. The way to form examples are shown below the figures.

Fig. 1 shows that LUR is the most efficient one among the five solvers. Then it is HM, LGP, Isolate and Dis in decreasing order. LUR stops for a system with degree 76 because the univariate polynomial equation solver outputs error since the equation is out of the ability of the univariate solver.

Fig. 2 shows a comparison among different solvers for systems with multiple zeros. LUR is also the most efficient one among the five solvers. It works for the systems with multiple roots of degree [49, 48]. Note that the bitsizes of the coefficients of the polynomials are larger than 100. That is why LUR seems slower comparing to itself in Fig. 1. The solver HM becomes very slow for systems with multiple roots.

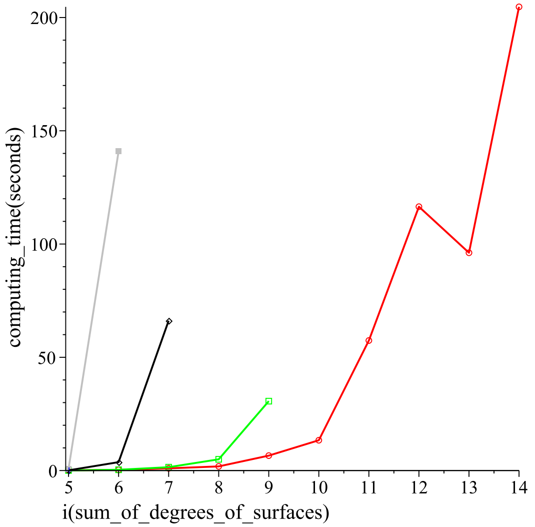

Fig. 2. Timings for the system $\{f, g\}$ with multiple roots, where $p = \mathit{randpoly}([x, y, z],\ \mathit{degree} = \mathit{ceil}(i/2),\ \mathit{coeffs} = \mathit{rand}(-10..10),\ \mathit{dense})$, $q = \mathit{randpoly}([x, y, z],\ \mathit{degree} = i - \mathit{ceil}(i/2),\ \mathit{coeffs} = \mathit{rand}(-10..10),\ \mathit{dense})$ and f is the square free part of $\mathrm{Res}_z(p, q)$, $g := \frac{\partial f}{\partial y}$, where $\mathit{ceil}(t)$ is the minimal integer larger than a given real number t. The symbols for different solvers are the same as in Fig. 1.

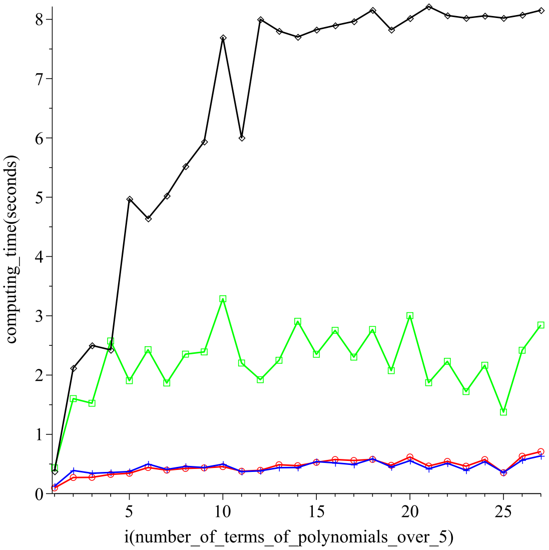

Fig. 3. Timings for the system $\{f, g\}$ with simple roots, where $f, g = \mathit{randpoly}([x, y],\ \mathit{degree} = 15,\ \mathit{terms} = 5i,\ \mathit{coeffs} = \mathit{rand}(-100..100),\ \mathit{dense})$. The symbols for different solvers are the same as in Fig. 1.

From Figs. 3 and 4, we can find that LUR, LGP, HM are stable for sparse systems. The reason is that all of them mainly involve resultant computation. Isolate is faster for sparse systems than for dense systems. The bitsizes of the polynomials influence all the solvers, especially for Isolate.

There is another efficient bivariate systems solver: Bisolve (Emeliyanenko et al., 2011). It is implemented in C and use GPU parallel technique to deal with some symbolic computations such as resultant and gcd computations. In another paper related to Bisolve, the computing times running on the same machine and the same examples were improved a lot (Emeliyanenko et al., 2013) compared

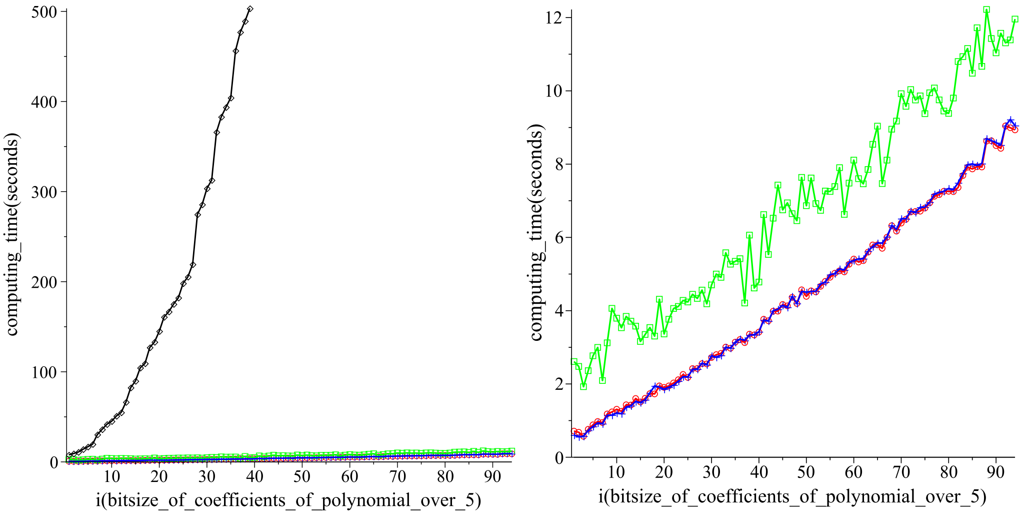

Fig. 4. Timings for the system $\{f, g\}$ with simple roots, where $f, g = randpoly([x, y],\ degree = 15,\ coeffs = rand(-2^{5i}..2^{5i}),$ $dense)$. The right one is the figure with large size without the timing for Isolate. The symbols for different solvers are the same as in Fig. 1.

to Emeliyanenko et al. (2011). It is around a half computing time compared to the old one. When comparing LUR and Bisolve, we use their new data in this paper. We do not compare with their implementation directly. But we compute the same examples taken from Emeliyanenko et al. (2013) on our machine. We compare the two methods in Table 1. Here, one part of data is taken from Emeliyanenko et al. (2013) directly. The other part of data is derived by running on our machine. Please see Table 1 for details. We denote our machine as M2, theirs as M1 for convenience. We can find that LGP runs the same examples on M2 taking around twice computing times (but a little less than) as on M1 in the average level. In Emeliyanenko et al. (2013), they used some filtering techniques to validate a majority of the candidates early. BS means without filters, BS + all means with all filters enabled. For BS + all, there are two groups of data. One uses GPU, denoted as GPU-BS + all, the other does not, denoted as CPU-BS + all. BS in the table means BS using GPU, denoted as BS + GPU. For LUR, we list the times of computing the first resultant and isolating its real roots, denoted as T_1 in Table 1. The total computing time is denoted as T.

Through we do not compare Bisolve with LUR directly, we compare them in an indirect way. The data in their paper shows that the filtering techniques improved the computing times a lot (usually more than one half) for Bisolve, especially for the systems with large bitsizes in coefficients. The parallel technique improved the Bisolve a lot (usually more than one half), but the improvement was not remarkable for systems with large bitsizes in coefficients. LGP is tested on both M1 and M2. The computing times of LGP on M1 are always around one half faster than on M2 for the same examples. We can find that LUR is usually faster than LGP, except for one or two examples. For some examples, LUR on M2 is faster than BS, CPU-BS + all on M1. The bitsizes of the coefficients of the systems influence BS and LUR deeper than GPU-BS + all and CPU-BS + all. We can find that for many examples, the total computing times of GPU-BS + all are less than the computing times for computing only the first resultant and its real root isolation. We use the computing times of GPU-BS + all and LGP to get a rate on M1, denoted as R1. Similarly, we can get R2 for LUR and LGP on M2. We can find that R1 is usually less than R2 except for some examples. The average level is around $R1 : R2 \approx 1 : 3$. Note that the part of computing resultants, real root isolation and computing s in LUR can be parallelized. Considering the influence of machines, parallel techniques and coding languages, our algorithm can be improved a lot.

From the comparisons before, we can conclude that LUR is efficient and stable for zero-dimensional bivariate polynomial systems.

Table 1
Timings for LUR and Bisolve in bivariate case for the fixed examples.

Comparing the computing times of Bisolve and LUR on special curves									
Machine	Linux platform on a 2.8 GHz 8-Core Inter Xeon W3530 with 8MB of L2 cache					Win XP on Inter(R) Core(TM) 2 quad CPU Q9400 @2.66 GHz with 2 × 3 MB of L2 cache			
Code language	C++				Maple				
GPU speedup	YES			NO					
Curves	BS	BS + all	$\frac{\text{BS+all}}{\text{LGP}}$	BS + all	LGP	$\frac{\text{LUR}}{\text{LGP}}$	LGP	LUR	
								T_1	T
13_sings_9	2.13	0.97	0.35	1.65	2.81	0.83	4.78	1.78	3.95
FTT_5_4_4	48.03	20.51	0.10	52.21	195.65	0.18	279.48	2.20	50.34
L4_circles	0.92	0.74	0.10	1.72	7.58	0.16	13.86	0.49	2.22
L6_circles	3.91	2.60	0.05	16.16	51.60	0.18	47.45	2.33	8.77
SA_2_4_eps	0.97	0.44	0.09	4.45	4.69	0.89	8.92	2.20	7.92
SA_4_4_eps	4.77	2.01	0.04	91.90	54.51	1.15	88.63	12.23	102.17
challenge_12	21.54	7.35	0.20	18.90	37.07	0.85	57.20	4.45	48.63
challenge_12_1	84.63	19.17	0.07	72.57	277.68	0.32	385.28	7.99	123.86
compact_surf	12.42	4.06	0.34	12.18	12.00	2.81	15.39	2.20	43.19
cov_sol_20	28.18	5.77	0.03	16.57	171.62	0.03	393.84	5.11	12.97
curve24	85.91	8.22	0.22	25.36	37.94	0.21	65.11	6.56	13.75
curve_issac	2.39	0.88	0.02	1.82	3.29	0.39	6.39	0.63	2.47
cusps_and_flexes	1.17	0.63	0.26	1.27	2.43	0.83	5.47	1.78	4.56
degree_7_surf	29.92	7.74	0.06	90.50	131.25	0.14	203.30	10.58	28.80
dfold_10_6	3.30	1.55	0.41	17.85	3.76	0.50	6.19	0.13	3.08
grid_deg_10	2.49	1.20	0.45	2.49	2.64	0.71	6.06	2.19	4.30
huge_cusp	9.64	6.44	0.06	13.67	116.67	0.41	224.98	76.00	91.28
mignotte_xy	timeout	243.16	–	310.13	timeout	–	timeout	322.00	325.08
spider	167.30	46.47	–	216.86	timeout	–	timeout	101.19	202.02
swinnerton_dyer	28.39	5.28	0.19	24.38	27.92	1.10	46.36	1.03	51.00
ten_circles	4.62	1.33	0.27	3.74	4.96	0.54	9.09	0.55	4.95
15, 10, dense	56.40	1.55	0.27	2.66	5.65	0.29	13.49	1.84	3.89
15, 128, dense	95.35	2.01	0.19	2.30	10.46	0.38	21.50	5.94	8.20
15, 512, dense	195.01	3.95	0.12	4.22	33.87	0.46	28.27	12.30	13.06
15, 2048, dense	timeout	19.89	0.10	20.45	190.86	0.45	233.13	100.58	105.58
15, 10, sparse	3.66	1.00	0.44	1.39	2.25	0.30	4.49	0.69	1.33
15, 128, sparse	12.14	1.25	0.29	1.35	4.27	0.38	8.83	2.73	3.36
15, 512, sparse	43.36	2.54	0.16	2.54	15.48	0.45	28.72	12.22	12.95
15, 2048, sparse	408.90	10.97	0.12	10.98	89.35	0.61	245.14	148.19	150.49

Table 2
Timings for random dense systems with the given degrees in multivariate case.

Degree type	LUR	Isolate	Dis
$[3, 3, 3]$	0.7644	0.0874	340.7092
$[5, 5, 5]$	27.1829	3.8826	–
$[2, 9, 9]$	10.1908	11.7686	–
$[7, 7, 7]$	614.1030	106.7302	–
$[3, 15, 15]$	498.4531	1720.2013	–

We also compare LUR with efficient solvers for multivariate polynomial systems. We compare mainly with Dis and Isolate, see Table 2. LUR is always faster than Dis. When there is a polynomial with lower degree in the system, LUR is faster than Isolate and it is slower than Isolate for the systems with equal degrees. The reason is that the former case can be projected to a bivariate system of lower degree. For the system with more variables, it is similar. Note that the core of Isolate is in C, ours is in Maple. For the same algorithm, the implementation in C is usually several times faster than that in Maple.

Acknowledgements

The authors would like to thank Prof. Xiao-Shan Gao for his good advices on the paper. All the authors would like to thank the anonymous referees, their suggestions improve the paper. The work is partially supported by NKBRPC (2011CB302400), NSFC Grants (11471327, 11001258, 60821002, 91118001), SRF for ROCS, SEM, and China-France cooperation project EXACTA (60911130369).

References

Alonso, M.E., Becker, E., Roy, M.F., Wörmann, T., 1996. Zeros, multiplicities, and idempotents for zerodimensional systems. In: Algorithms in Algebraic Geometry and Applications. Birkhäuser, pp. 1–15.

Basu, S., Pollack, R., Roy, M.F., 2003. Algorithms in Real Algebraic Geometry. Springer, Berlin.

Becker, E., Wörmann, T., 1996. Radical computations of zero-dimensional ideals and real root counting. Math. Comput. Simul. 42 (4–6), 561–569.

Busé, L., Khalil, H., Mourrain, B., 2005. Resultant-based methods for plane curves intersection problems. In: CASC 2005, pp. 75–92.

Canny, J.F., 1988. Some algebraic and geometric computation in PSPACE. In: ACM Symp. on Theory of Computing. SIGACT, pp. 460–469.

Cheng, J.S., Gao, X.S., Guo, L., 2012. Root isolation of zero-dimensional polynomial systems with linear univariate representation. J. Symb. Comput. 47 (7), 843–858.

Cheng, J.S., Gao, X.S., Li, J., 2009. Root isolation for bivariate polynomial systems with local generic position method. In: ISSAC 2009, pp. 103–110.

Cheng, J.S., Gao, X.S., Yap, C.K., 2009. Complete numerical isolation of real roots in zero-dimensional triangular systems. J. Symb. Comput. 44 (7), 768–785.

Corless, R., Gianni, P., Trager, B., 1997. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In: ISSAC 1997, pp. 133–140.

Cox, D.A., Little, J.B., O'Shea, D., 1998. Using Algebraic Geometry. Grad. Texts Math., vol. 185. Springer.

Diochnos, D.I., Emiris, I.Z., Tsigaridas, E.P., 2009. On the asymptotic and practical complexity of solving bivariate systems over the reals. J. Symb. Comput. 44 (7), 818–835. Special issue for ISSAC 2007.

Emeliyanenko, P., Berberich, E., Sagraloff, M., 2011. An elimination method for solving bivariate polynomial systems: eliminating the usual drawbacks. In: Algorithm Engineering and Experiments (ALENEX).

Emeliyanenko, P., Kobel, A., Berberich, E., Sagraloff, M., 2013. Exact symbolic-numeric computation of planar algebraic curves. Theor. Comput. Sci. 491, 1–32.

Emeliyanenko, P., Sagraloff, M., 2012. On the complexity of solving a bivariate polynomial system. In: ISSAC 2012, pp. 154–161.

Emiris, I.Z., Mourrain, B., Tsigaridas, E.P., 2008. Real algebraic numbers: complexity analysis and experimentation. In: Reliable Implementation of Real Number Algorithms, pp. 57–82.

Emiris, I.Z., Tsigaridas, E.P., 2005. Real solving of bivariate polynomial systems. In: CASC 2005, pp. 150–161.

Fulton, W., 1984. Introduction to Intersection Theory in Algebraic Geometry. Amer. Math. Soc., Providence, RI, Washington, DC.

Gao, X.S., Chou, S.C., 1999. On the theory of resolvents and its applications. Syst. Sci. Math. Sci. 12, 17–30.

Giusti, M., Heintz, J., 1991. Algorithmes – disons rapides – pour la dècomposition d'une variètè algébrique en composantes irréducibles et équidimensionnelles. In: Proc. MEGA' 90. Birkhäuser, pp. 169–193.

Giusti, M., Lecerf, G., Salvy, B., 2001. A Gröbner free alternative for polynomial system solving. J. Complex. 17, 154–211.

Hong, H., Shan, M., Zeng, Z., 2008. Hybrid method for solving bivariate polynomial system. In: SRATC 2008.

Kerber, M., Sagraloff, M., 2012. A worst-case bound for the topology computation of algebraic curves. J. Symb. Comput. 47, 239–258.

Kobayashi, H., Moritsugu, S., Hogan, R.W., 1988. Solving systems of algebraic equations. In: ISSAC 1988, pp. 139–149.

Mantzaflaris, A., Mourrain, B., Tsigaridas, E.P., 2011. On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. Theor. Comput. Sci. 412 (22), 2312–2330.

Mignotte, M., 1992. Mathematics for Computer Algebra. Springer-Verlag.

Moore, Ramon E., Baker Kearfott, R., Cloud, Michael J., 2009. Introduction to Interval Analysis. Society for Industrial and Applied Mathematics, Philadelphia.

Mourrain, B., Pavone, J.-P., 2009. Subdivision methods for solving polynomial equations. J. Symb. Comput. 44 (3), 292–306.

Pan, V.Y., 2000. Approximating complex polynomial zeros: modified Weyl's quadtree construction and improved Newton's iteration. J. Complex. 16 (1), 213–264.

Pan, V.Y., 2002. Univariate polynomials: nearly optimal algorithms for numerical factorization and root-finding. J. Symb. Comput. 33 (5), 701–733.

Pan, V.Y., Tsigaridas, E.P., 2013. On the Boolean complexity of real root refinement. In: ISSAC 2013, pp. 299–306.

Qin, X., Feng, Y., Chen, J., Zhang, J., 2013. Parallel computation of real solving bivariate polynomial systems by zero-matching method. Appl. Math. Comput. 219 (14), 7533–7541.

Reischert, D., 1997. Asymptotically fast computation of resultants. In: Proceedings of ISSAC '97. ACM Press, Hawaii, pp. 233–240.

Rouillier, F., 1999. Solving zero-dimensional systems through the rational univariate representation. Appl. Algebra Eng. Commun. Comput. 9 (5), 433–461.

Rouillier, F., Zimmermann, P., 2003. Efficient isolation of polynomial real roots. J. Comput. Appl. Math. 162 (1), 33–50.

Sagraloff, M., 2012. When Newton meets Descartes: a simple and fast algorithm to isolate the real roots of a polynomial. In: ISSAC 2012, pp. 297–304.

Schönhage, A., 1982. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany. http://www.iai.uni-bonn.de/~schoe/fdthmrep.ps.gz.

Stahl, V., 1995. Interval methods for bounding the range of polynomials and solving systems of nonlinear equations. PhD thesis. Johannes Kepler University, Austria.

Tan, C., Zhang, S.C., 2009. Separating element computation for the rational univariate representation with short coefficients in zero-dimensional algebraic varieties. J. Jilin University (Science Edition) 47, 174–178.

Xia, B., Yang, L., 2002. An algorithm for isolating the real solutions of semi-algebraic systems. J. Symb. Comput. 34 (5), 461–477.

Yap, C., 2000. Fundamental Problems of Algorithmic Algebra. Oxford University Press, New York.

Yokoyama, K., Noro, M., Takeshima, T., 1989. Computing primitive elements of extension fields. J. Symb. Comput. 8 (6), 553–580.