Real Root Isolation of Polynomial Equations Based on Hybrid Computation

Fei Shen, Wenyuan Wu and Bican Xia

Abstract A new algorithm for real root isolation of zero-dimensional nonsingular square polynomial systems based on hybrid computation is presented in this paper. First, approximate the (complex) roots of the given polynomial equations via homotopy continuation method. Then, for each approximate root, an initial box relying on the Kantorovich theorem is constructed, which contains the corresponding accurate root. Finally, the Krawczyk interval iteration with interval arithmetic is applied to the initial boxes so as to check whether or not the corresponding approximate roots are real and to obtain the real root isolation boxes. Moreover, an empirical construction of initial box is provided for speeding-up the computation in practice. Our experiments on many benchmarks show that the new hybrid method is very efficient. The method can find all real roots of any given zero-dimensional nonsingular square polynomial systems provided that the homotopy continuation method can find all complex roots of the equations.

1 Introduction

The *Real Root Isolation* of polynomial equations is a procedure that produces disjoint regions to isolate all the distinct real roots of polynomial equations, with only one root in each region. Formally speaking, let $\mathbf{F} = (f_1, f_2, \ldots, f_n)^{\mathrm{T}}$ be polynomial equations defined on $\mathbf{R}^n$, i.e. $f_i \in \mathbf{R}[x_1, x_2, \ldots, x_n]$. Suppose $\mathbf{F}(\mathbf{x}) = 0$ has only finite many real roots, say $\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(m)}$. The target of real root isolation is

F. Shen (✉) · B. Xia (✉)
LMAM & School of Mathematical Sciences,
Peking University, Beijing, China
e-mail: shenfei@pku.edu.cn

B. Xia
e-mail: xbc@math.pku.edu.cn

W. Wu (✉)
Chongqing Institute of Green and Intelligent Technology,
Chinese Academy of Sciences, Chongqing, China
e-mail: wuwenyuan@cigit.ac.cn

R. Feng et al. (eds.), *Computer Mathematics*, DOI 10.1007/978-3-662-43799-5_26

to compute a family of regions $S_1, S_2, \ldots, S_m$, $S_j \subset \mathbf{R}^n (1 \leq j \leq m)$, such that $\xi^{(j)} \in S_j$ and $S_i \cap S_j = \emptyset$ $(1 \leq i, j \leq m)$. Usually, we use rectangular boxes to denote the regions above. So we often call these isolated boxes *intervals* in this paper. Theoretically, the width of intervals for some special problems can be very small. Hence, we assume that the accuracy of numerical computation in this paper can be arbitrarily high. However, it is also important to point out that such case rarely happens in nearly 50 examples we computed and double-precision is usually enough to obtain very small intervals.

Real root isolation is an important problem in symbolic computation. It is an algorithm for solving equations accurately since no root formula is available in general situation. It is also a critical part of some other important algorithms, such as CAD and real root classification for semi-algebraic systems, etc. Improvement on real root isolation will benefit all of these algorithms.

We impose some hypothesis on the problem discussed here. The first is that the system is square, i.e., the number of equations is the same as that of variables. Then we only handle the systems with finitely many complex roots. Positive dimensional complex solution is beyond the scope of this paper. Moreover, we suppose that the Jacobian matrix of $\mathbf{F}$ is nonsingular at each root of $\mathbf{F}(\mathbf{x}) = 0$. So we only deal with the simple root cases. For the singular situation, the deflation method [12, 16, 17, 32] can be applied, which is one of our ongoing work.

Most of the previous real root isolation algorithms are based on symbolic computations. For instance, the Collins–Akritas algorithm [10, 11] based on Descartes' rule of signs is for polynomials in one variable. In multi-variable scenario, there are many work using different theories and techniques with focuses on complexity and/or solving benchmarks, see for example [6–9, 13, 15, 21, 24, 29, 34, 35].

An advantage of those symbolic methods is that exact results can be obtained since they use symbolic computation and some of them can be extended to semi-algebraic systems. However, there are also some disadvantages. Some of these methods could only handle the isolation of complex roots. And some of them need to triangularize the system first, which is very time-consuming in computation when the scale (such as the number of variables or the degrees of polynomials) of the system is big. While some methods that do not use triangularization have to give a huge initial interval to include all the real roots [37, 38], which is extremely inefficient.

In order to avoid these problems and design a new algorithm that could efficiently solve more complicated systems and provide accurate interval results, we employ hybrid computation to take the advantages of both symbolic and numerical methods.

The basic idea of this paper is to use a numerical method to obtain all the approximate roots of polynomial systems, including possible nonreal ones. With these approximations, small initial intervals which contains the corresponding real roots are constructed. Then, we apply a symbolic method to these initial intervals to verify whether there is a real root in it or not. The main method we use during numerical computation is homotopy continuation, and for symbolic process we use the Krawczyk iteration.

Most of the work in this paper comes from [27]. In Sect. 2, we will introduce some preliminaries, including homotopy continuation and interval arithmetic. A new

real root isolation algorithm is discussed in Sect. 3. To test our new method, our experimental results on benchmarks together with comparison and analysis will be presented in Sect. 4. Finally, there is a summary in Sect. 5 and some future work will also be discussed.

2 Preliminary

We introduce in this section the basic theory and tools that would be used in our algorithm.

2.1 Homotopy Continuation Method

Homotopy continuation method is an important numerical computation method, which is used in various fields. We only treat it as an "algorithm black box" here, where the input is a polynomial system, and the output is its approximate roots. Please find the details about the theory in [18, 28].

Our method to find all real roots of any given zero-dimensional nonsingular square polynomial equations relies on the homotopy continuation methods which theoretically obtain all complex roots of the equations. For the real homotopies, we refer to the work by Li and Wang [20]. For the certified numerical homotopy tracking and its complexity, we first refer to the work by M. Shub and S. Smale on complexity of Bezout's theorem [5] and the readers can also find the recent results on this topic by Beltran and Leykin in [1–3].

For our purpose, it is convenient to utilize some existing software, such as Hom4ps-2.0 [19], PHCpack [30] and HomLab [31].

In our implementation, we use Hom4ps-2.0, which could return all the approximate complex roots of a given polynomial system efficiently, along with residues and condition numbers.

2.2 Interval Arithmetic

Interval arithmetic plays an important role in real root isolation algorithms [34, 37, 38]. The two main differences between our new algorithm and the traditional ones in [37, 38] are: (1) Verification only carry out on the localized "small" intervals; (2) symbolic computation is replaced with floating point numerical computation.

Most of the interval operations in this paper's algorithms are based on Rump's floating point verification work [26] and accomplished by using the Matlab package Intlab [25], including interval arithmetic operations and Jacobian matrix, Hessian matrix calculations.[1]

[1] See reference [26], Sect. 11, *Automatic differentiation*.

2.2.1 Basic Concepts

We introduce some basic interval arithmetic theory in this section. See reference [22] for more details.

For given numbers $\underline{x}, \overline{x} \in \mathbf{R}$, if $\underline{x} \leq \overline{x}$, we call

$$X = [\underline{x}, \overline{x}] = \{x \in \mathbf{R} | \underline{x} \leq x \leq \overline{x}\}$$

a *bounded closed interval*, or *interval* for short. Denote by $I(\mathbf{R})$ the set of all the bounded close intervals on $\mathbf{R}$, and $I(A) = \{X \in I(\mathbf{R}) | X \subseteq A\}$ all the intervals on $A \subseteq \mathbf{R}$. Especially, if $\underline{x} = \overline{x}$, we call X a *point interval*.

For intervals, there are some common quantities:

midpoint $\quad \mathrm{mid}(X) = (\underline{x} + \overline{x})/2$
width $\quad \mathrm{W}(X) = \overline{x} - \underline{x}$
radius $\quad \mathrm{rad}(X) = \frac{1}{2}\mathrm{W}(X)$
low end point $\quad \inf(X) = \underline{x}$
high end point $\quad \sup(X) = \overline{x}$

Obviously we have $X = [\mathrm{mid}(X) - \mathrm{rad}(X), \mathrm{mid}(X) + \mathrm{rad}(X)]$. An interval is usually expressed by its midpoint and radius. For example, if $m = \mathrm{mid}(X)$, $r = \mathrm{rad}(X)$, then we can write the formula above as $X = \mathrm{midrad}(m, r)$.

We can also define the arithmetic operations over intervals. Let $X = [\underline{x}, \overline{x}]$, $Y = [\underline{y}, \overline{y}] \in I(\mathbf{R})$,

- $X + Y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$
- $X - Y = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$
- $X \cdot Y = [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})]$
- $X/Y = [\underline{x}, \overline{x}] \cdot [1/\overline{y}, 1/\underline{y}],\ 0 \notin Y$

A vector is called an *interval vector* if all its components are intervals. *Interval matrix* can be similarly defined. For interval vectors and interval matrices, the concepts such as midpoint, width, radius, etc., and the arithmetic operations are defined in components.

Let $f : \mathbf{R}^n \to \mathbf{R}$ be a function, if there exists an interval map

$$F : I(\mathbf{R}^n) \to I(\mathbf{R})$$

such that for all $x_i \in X_i (i = 1, 2, \ldots, n)$,

$$F([x_1, x_1], [x_2, x_2], \ldots, [x_n, x_n]) = f(x_1, x_2, \ldots, x_n)$$

holds, then we call F an *interval expansion* of f.

We call $F : I(\mathbf{R}^n) \to I(\mathbf{R})$ an interval map with *inclusive monotonicity* if $\mathbf{X} \subseteq \mathbf{Y}$ implies $F(\mathbf{X}) \subseteq F(\mathbf{Y})$ for any given intervals $\mathbf{X}$ and $\mathbf{Y}$. The definitions above can all be extended to the situations in $I(\mathbf{R}^n) \to I(\mathbf{R}^n)$. And it is easy to prove that all the polynomial operations satisfy the inclusive monotonicity.

2.2.2 Krawczyk Operator

The Krawczyk operator plays a key role in the real root verification of interval arithmetic. The main accomplishment comes from the work of Krawczyk and Moore. We only list some important results here. Complete proofs can be found in [22].

Suppose $\mathbf{f} : D \subseteq \mathbf{R}^n \to \mathbf{R}^n$ is continuous differentiable on D. Consider the equation

$$\mathbf{f}(\mathbf{x}) = 0. \tag{1}$$

Let $\mathbf{f}'$ be the Jacobi matrix of $\mathbf{f}$, $\mathbf{F}$ and $\mathbf{F}'$ be the interval expansion of $\mathbf{f}$ and $\mathbf{f}'$ with inclusive monotonicity, respectively. For $\mathbf{X} \in I(D)$ and any $\mathbf{y} \in \mathbf{X}$, define the *Krawczyk operator* as:

$$K(\mathbf{y}, \mathbf{X}) = \mathbf{y} - Y\mathbf{f}(\mathbf{y}) + (\mathbf{I} - Y\mathbf{F}'(\mathbf{X}))(\mathbf{X} - \mathbf{y}) \tag{2}$$

where $\mathbf{I}$ is the $n \times n$ unit matrix and Y is any $n \times n$ nonsingular matrix.

Especially, we assign $\mathbf{y} = \text{mid}(\mathbf{X})$, so Formula (2) becomes

$$K(\mathbf{X}) = \text{mid}(\mathbf{X}) - Y\mathbf{f}(\text{mid}(\mathbf{X})) + (\mathbf{I} - Y\mathbf{F}'(\mathbf{X}))\text{rad}(\mathbf{X})[-1, 1]. \tag{3}$$

Formula (3) is often used in practice.

The reason why the Krawczyk operator is so important is that it has some nice properties.

Proposition 1 *Suppose $K(\mathbf{y}, \mathbf{X})$ is defined as Formula (2), then*

1. *If $\mathbf{x}^* \in \mathbf{X}$ is a root of Eq. (1), then for any $\mathbf{y} \in \mathbf{X}$, we have $\mathbf{x}^* \in K(\mathbf{y}, \mathbf{X})$;*
2. *For any $\mathbf{y} \in \mathbf{X}$, if $\mathbf{X} \cap K(\mathbf{y}, \mathbf{X}) = \emptyset$ holds, then there is no roots in $\mathbf{X}$;*
3. *For any $\mathbf{y} \in \mathbf{X}$ and any nonsingular matrix Y, if $K(\mathbf{y}, \mathbf{X}) \subseteq \mathbf{X}$ holds, then Eq. (1) has a solution in $\mathbf{X}$;*
4. *Moreover, for any $\mathbf{y} \in \mathbf{X}$ and any nonsingular matrix Y, if $K(\mathbf{y}, \mathbf{X})$ is strict inclusive in $\mathbf{X}$, then Eq. (1) has only one root in $\mathbf{X}$.*

With the properties above, we can easily develop a real root verification method which will be explained later in this paper.

Meanwhile, with the hypothesis we set in introduction, all the systems considered here are nonsingular ones with only simple roots. So the Jacobian matrix at the roots are all invertible. Thus, we often set $Y = (\text{mid}\,\mathbf{F}'(\mathbf{X}))^{-1}$ and the Krawczyk operator becomes

$$\begin{aligned} K(\mathbf{X}) = {} & \text{mid}(\mathbf{X}) - (\text{mid}\,\mathbf{F}'(\mathbf{X}))^{-1}\mathbf{f}(\text{mid}(\mathbf{X})) \\ & + (\mathbf{I} - (\text{mid}\,\mathbf{F}'(\mathbf{X}))^{-1}\mathbf{F}'(\mathbf{X}))\text{rad}(\mathbf{X})[-1, 1]. \end{aligned} \tag{4}$$

This is also called the Moore form of the Krawczyk operator.

3 Real Root Isolation Algorithm

In this section, we will present our new algorithm for real root isolation based on hybrid computation. As mentioned before, we first construct initial intervals for the approximate roots obtained by homotopy continuation, and then find out those intervals containing real roots via the Krawczyk interval iteration.

3.1 Construction of Initial Intervals

To apply the Krawczyk interval iteration, obviously the construction of initial intervals is a key procedure. We should guarantee both the *correctness* and *efficiency*, that is, make sure the initial box contains the corresponding accurate real root, and meanwhile keep the interval radius as small as possible so as to shorten the iteration time.

Thus a valid error estimate for the initial approximate roots should be established. And we discuss this issue in both theoretical and practical aspects here.

3.1.1 Error Estimate Theory

The core problem of the construction of initial box is the choice of interval radius, which is essentially an error estimate for the approximate root. There are many work on error analysis, from classical results to modern ones. For example, in [5], Smale et al. gave a systemic method which is now often called alpha theory.

Here we employ the Kantorovich Theorem to give our error estimate.

Theorem 2 (Kantorovich) *Let X and Y be Banach spaces and $F : D \subseteq X \to Y$ be an operator, which is Fréchet differentiable on an open convex set $D_0 \subseteq D$. For equation $F(x) = 0$, if the given approximate root $x_0 \in D_0$ meets the following three conditions:*

1. *$F'(x_0)^{-1}$ exists, and there are real numbers B and η such that*

$$\|F'(x_0)^{-1}\| \leq B, \quad \|F'(x_0)^{-1}F(x_0)\| \leq \eta,$$

2. *F' satisfies the Lipschitz condition on D_0:*

$$\|F'(x) - F'(y)\| \leq K\|x - y\|, \ \forall x, y \in D_0,$$

3. *$h = BK\eta \leq \frac{1}{2}$, $O(x_0, \frac{1-\sqrt{1-2h}}{h}\eta) \subset D_0$,*

then we claim that:

1. *$F(x) = 0$ has a root x^* in $\overline{O(x_0, \frac{1-\sqrt{1-2h}}{h}\eta)} \subset \overline{D_0}$, and the sequence $\{x_k : x_{k+1} = x_k - F'(x_k)^{-1}F(x_k)\}$ of Newton method converges to x^*;*
2. *For the convergence of x^*, we have:*

$$\|x^* - x_{k+1}\| \leq \frac{\theta^{2^{k+1}}(1-\theta^2)}{\theta(1-\theta^{2^{k+1}})}\eta \tag{5}$$

where $\theta = \frac{1-\sqrt{1-2h}}{1+\sqrt{1-2h}}$;

3. *The root* x^* *is unique in* $\overline{D_0} \cap \overline{O(x_0, \frac{1+\sqrt{1-2h}}{h}\eta)}$.

In the theorem, $O(x, r)$ denotes the ball neighborhood whose center is x and radius is r, and $\overline{O(x, r)}$ refers to the closure of the ball neighborhood. The proof can be found in [14].

Since the approximation x_0 is already the result of a homotopy process, what we care about is the initial interval w.r.t. x_0, i.e. the proper upper bound for $\|x^* - x_0\|$. So we have the following proposition, which is a direct corollary of the Kantorovich Theorem.

Proposition 3 *Let* $\mathbf{F} = (f_1, \ldots, f_n)^{\mathrm{T}}$ *be a polynomial system, where* $f_i \in \mathbf{R}[x_1, \ldots, x_n]$. *Denote by* $\mathbf{J}$ *the Jacobian matrix of* $\mathbf{F}$. *For an approximation* $\mathbf{x}_0 \in \mathbf{C}^n$, *if the following conditions hold:*

1. $\mathbf{J}^{-1}(\mathbf{x}_0)$ *exists, and there are real numbers* B *and* η *such that*

$$\|\mathbf{J}^{-1}(\mathbf{x}_0)\| \leq B, \quad \|\mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{F}(\mathbf{x}_0)\| \leq \eta,$$

2. *There exists a ball neighborhood* $O(\mathbf{x}_0, \omega)$ *such that* $\mathbf{J}(\mathbf{x})$ *satisfies the Lipschitz condition on it:*

$$\|\mathbf{J}(\mathbf{x}) - \mathbf{J}(\mathbf{y})\| \leq K\|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in O(\mathbf{x}_0, \omega)$$

3. *Let* $h = BK\eta$,

$$h \leq \frac{1}{2}, \quad \textit{and} \quad \omega \geq \frac{1-\sqrt{1-2h}}{h}\eta,$$

then $\mathbf{F}(\mathbf{x}) = 0$ *has only one root* $\mathbf{x}^*$ *in* $\overline{O(\mathbf{x}_0, \omega)} \cap \overline{O(\mathbf{x}_0, \frac{1+\sqrt{1-2h}}{h}\eta)}$.

Proof We consider F as an operator on $\mathbf{C}^n \to \mathbf{C}^n$, obviously it is Fréchet differentiable, and from

$$\mathbf{F}(\mathbf{x} + \mathbf{h}) = \mathbf{F}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h} + o(\mathbf{h})$$

we can get

$$\lim_{\mathbf{h}\to 0} \frac{\|\mathbf{F}(\mathbf{x} + \mathbf{h}) - \mathbf{F}(\mathbf{x}) - \mathbf{J}(\mathbf{x})\mathbf{h}\|}{\|\mathbf{h}\|} = 0.$$

Thus the first order Fréchet derivative of $\mathbf{F}$ is just the Jacobian matrix $\mathbf{J}$, i.e. $\mathbf{F}'(\mathbf{x}) = \mathbf{J}(\mathbf{x})$. So by Theorem 2, the proof is completed immediately after checking the situation of $\|\mathbf{x}^* - \mathbf{x_0}\|$.

It is easy to know that $\frac{1-\sqrt{1-2h}}{h} \leq 2$. So we can just assign $\omega = 2\eta$. Then we need to check whether $BK\eta \leq \frac{1}{2}$ in the neighborhood $O(\mathbf{x}_0, 2\eta)$. Even though the initial $\mathbf{x}_0$ does not satisfy the conditions, we can still find a proper $\mathbf{x}_k$ after several Newton iterations, since B and K are bounded and η will approach zero. And we only need to find an upper bound for the Lipschitz constant K.

3.1.2 Constructive Algorithm

Now we will give a constructive procedure for the Lipschitz constant K.

Let $J_{ij} = \partial f_i / \partial x_j$, apply mean value inequality [23] to each element of $\mathbf{J}$ on $O(\mathbf{x}_0, \omega)$ to get

$$\|J_{ij}(\mathbf{y}) - J_{ij}(\mathbf{x})\| \leq \sup_{\kappa_{ij} \in line(\mathbf{x},\mathbf{y})} \|\nabla J_{ij}(\kappa_{ij})\| \cdot \|\mathbf{y} - \mathbf{x}\|, \ \forall \mathbf{x}, \mathbf{y} \in O(\mathbf{x}_0, \omega) \quad (6)$$

where $\nabla = (\partial/\partial x_1, \partial/\partial x_2, \ldots, \partial/\partial x_n)$ is the gradient operator and $line(\mathbf{x}, \mathbf{y})$ refers to the line connecting $\mathbf{x}$ with $\mathbf{y}$. Since ∇J is continous, we can find a $\zeta_{ij} \in line(\mathbf{x}, \mathbf{y})$ such that $\|\nabla J_{ij}(\zeta_{ij})\| = \sup_{\kappa_{ij} \in line(\mathbf{x},\mathbf{y})} \|\nabla J_{ij}(\kappa_{ij})\|$. So we get

$$\|J_{ij}(\mathbf{y}) - J_{ij}(\mathbf{x})\| \leq \|\nabla J_{ij}(\zeta_{ij})\| \cdot \|\mathbf{y} - \mathbf{x}\|, \quad \forall \mathbf{x}, \mathbf{y} \in O(\mathbf{x}_0, \omega) \quad (7)$$

Setting $\left(\|\nabla J_{ij}(\zeta_{ij})\| \cdot \|\mathbf{y} - \mathbf{x}\|\right)_{n\times n} = \triangle\mathbf{J}$, then $\|J(y) - J(x)\| \leq \|\triangle\mathbf{J}\|$. And for $\triangle\mathbf{J}$ we have

$$\begin{aligned} \|\triangle\mathbf{J}\|_\infty &= \|\left(\|\nabla J_{ij}(\zeta_{ij})\|_\infty \|\mathbf{y} - \mathbf{x}\|_\infty\right)_{n\times n}\|_\infty \\ &\leq \|\left(\|\nabla J_{ij}(\zeta_{ij})\|_\infty\right)_{n\times n}\|_\infty \cdot \|\mathbf{y} - \mathbf{x}\|_\infty \\ &= \max_{1\leq i\leq n} \sum_{j=1}^{n} \|\nabla J_{ij}(\zeta_{ij})\|_\infty \cdot \|\mathbf{y} - \mathbf{x}\|_\infty \end{aligned} \quad (8)$$

Note that $\nabla J_{ij}(\zeta_{ij})$ is a vector, so if we use $|\cdot|_{\max}$ to denote the maximum module component of a vector, then we have

$$\|\triangle\mathbf{J}\|_\infty \leq \max_{1\leq i\leq n} \sum_{j=1}^{n} |\nabla J_{ij}(\zeta_{ij})|_{\max} \cdot \|\mathbf{y} - \mathbf{x}\|_\infty. \quad (9)$$

Let $H_i = (\frac{\partial^2 f_i}{\partial x_j \partial x_k})_{n\times n}$ be the Hessian matrix of f_i, and let $H_i = (h_1^{(i)}, \ldots, h_n^{(i)})$, where $h_j^{(i)}$ are the column vectors. Then we have

$$\|\triangle\mathbf{J}\|_\infty \leq \max_{1\leq i\leq n} \sum_{j=1}^{n} |h_j^{(i)}(\zeta_{ij})|_{\max} \cdot \|\mathbf{y} - \mathbf{x}\|_\infty. \quad (10)$$

For convenience, we construct $\mathbf{X}_0 = \text{midrad}(\mathbf{x}_0, \omega)$ with $\mathbf{x}_0$ as centre and $\omega = 2\eta$ as radius.

Now we have $|h_j^{(i)}(\zeta_{ij})|_{\max} \le |h_j^{(i)}(\mathbf{X}_0)|_{\max}$. So

$$\|\triangle \mathbf{J}\|_\infty \le \max_{1\le i\le n} \sum_{j=1}^{n} |h_j^{(i)}(\mathbf{X}_0)|_{\max} \cdot \|\mathbf{y} - \mathbf{x}\|_\infty. \tag{11}$$

Therefore

$$K = \max_{1\le i\le n} \sum_{j=1}^{n} |h_j^{(i)}(\mathbf{X}_0)|_{\max} \tag{12}$$

is the Lipschitz constant w.r.t. $\mathbf{J}$.

Now we give an algorithm for computing initial intervals in Algorithm 1.

Algorithm 1 init_width

Input: Equation F; Approximation x_0; Number of variables n
Output: Initial interval's radius r

repeat
  $\mathbf{x}_0 = \mathbf{x}_0 - \mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{F}(\mathbf{x}_0)$;
  $B = \|\mathbf{J}^{-1}(\mathbf{x}_0)\|_\infty$; $\eta = \|\mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{F}(\mathbf{x}_0)\|_\infty$;
  $\omega = 2\eta$;
  $\mathbf{X}_0 = \text{midrad}(\mathbf{x}_0, \omega)$;
  $K = 0$;
  for $i = 1$ **to** n **do**
    Compute the Hessian matrix $H_i = (h_1^{(i)}, h_2^{(i)}, \ldots, h_n^{(i)})$ of $\mathbf{F}$ on $\mathbf{X}_0$;
    if $\sum_{j=1}^{n} |h_j^{(i)}(\mathbf{X}_0)|_{\max} > K$ **then**
      $K = \sum_{j=1}^{n} |h_j^{(i)}(\mathbf{X}_0)|_{\max}$;
    end if
  end for
  $h = BK\eta$;
until $h \le 1/2$
return $r = \frac{1-\sqrt{1-2h}}{h}\eta$

3.2 Empirical Estimate

As so far, we have established a rigorous method to construct initial intervals. This method takes a complex approximate root as input to obtain an initial box. But in practice we often find many approximations with "large" imaginary parts which strongly indicate that they are nonreal. A natural question is

Can we detect these nonreal roots without using interval arithmetic?

Let $\mathbf{z}$ be an approximation of the real root ξ. Because

$$\|\mathscr{R}e(\mathbf{z}) - \xi\| \leq \|\mathbf{z} - \xi\|,$$

then we can see the real part $\mathscr{R}e(\mathbf{z})$ is also an approximation of this root and is even closer. So we can simply replace $\mathbf{x_0}$ by $\mathscr{R}e(\mathbf{x_0})$ in Algorithm 1 to construct the initial box.

The other consideration is the efficiency of numerical computation. When we use Proposition 3, lots of interval matrix operations would be executed, which cost much more time than the point operations. So if we can find an empirical estimate radius, which can be computed much faster, but is still valid for most of the equations, then that will be a good choice in practice.

We now give one such empirical estimate.

For $\mathbf{F} = 0$, let $\mathbf{x}^*$ be an accurate root and $\mathbf{x}_0$ be its approximation. Although the mean value theorem is not valid in complex space, the Taylor expansion is still valid. And the polynomial systems considered here are all continuous, so we suppose the equation satisfies the mean value theorem approximately:

$$0 = \mathbf{F}(\mathbf{x}^*) \approx \mathbf{F}(\mathbf{x}_0) + \mathbf{J}(\xi)(\mathbf{x}^* - \mathbf{x}_0) \tag{13}$$

where ξ is between $\mathbf{x}^*$ and $\mathbf{x}_0$. So we have

$$\mathbf{x}^* - \mathbf{x}_0 \approx -\mathbf{J}^{-1}(\xi)\mathbf{F}(\mathbf{x}_0).$$

Let $\mathbf{J}(\xi) = \mathbf{J}(\mathbf{x}_0) + \triangle\mathbf{J}$, then

$$\begin{aligned} \mathbf{J}(\xi) &= \mathbf{J}(\mathbf{x}_0)(\mathbf{I} + \mathbf{J}^{-1}(\mathbf{x}_0)\triangle\mathbf{J}), \\ \mathbf{J}^{-1}(\xi) &= (\mathbf{I} + \mathbf{J}^{-1}(\mathbf{x}_0)\triangle\mathbf{J})^{-1}\mathbf{J}^{-1}(\mathbf{x}_0). \end{aligned} \tag{14}$$

For $\triangle\mathbf{J}$, we can get an estimate similar to Formula (10):

$$\|\triangle\mathbf{J}\|_\infty \leq \max_{1\leq i\leq n} \sum_{j=1}^{n} |h_j^{(i)}(\zeta_{ij})|_{\max} \cdot \|\mathbf{x}^* - \mathbf{x}_0\|_\infty. \tag{15}$$

From our hypothesis, $\mathbf{x}^*$ and $\mathbf{x}_0$ are very close, so are ζ_{ij} and $\mathbf{x}_0$. Thus, we approximate $\mathbf{x}_0$ with ζ_{ij}. Meanwhile, from $\mathbf{x}_0$, after a Newton iteration, we get $\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{F}(\mathbf{x}_0)$. Thus we may consider that the distance between $\mathbf{x}^*$ and $\mathbf{x}_0$ is more or less the same with that of $\mathbf{x}_0$ and $\mathbf{x}_1$, so we replace $\|\mathbf{x}^* - \mathbf{x}_0\|$ with $\|\mathbf{x}_1 - \mathbf{x}_0\| = \|\mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{F}(\mathbf{x}_0)\|$ for approximation.

So we get

$$\|\triangle\mathbf{J}\|_\infty \leq \max_{1\leq i\leq n} \sum_{j=1}^{n} |h_j^{(i)}(\mathbf{x}_0)|_{\max} \cdot \|\mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{F}(\mathbf{x}_0)\|_\infty. \tag{16}$$

Let $\lambda = \max_{1\leq i\leq n} \sum_{j=1}^{n} |h_j^{(i)}(\mathbf{x}_0)|_{\max}$, then

$$\|\mathbf{J}^{-1}(\mathbf{x}_0)\triangle\mathbf{J}\|_\infty \leq \lambda\|\mathbf{J}^{-1}(\mathbf{x}_0)\|_\infty^2\|\mathbf{F}(\mathbf{x}_0)\|_\infty.$$

Because $\|\mathbf{F}(\mathbf{x}_0)\|_\infty \ll 1$, the last formula is also much less than 1. So substitute that into Formula (14) we can get

$$\|\mathbf{J}^{-1}(\xi)\|_\infty \leq \frac{\|\mathbf{J}^{-1}(\mathbf{x}_0)\|_\infty}{1-\lambda\|\mathbf{J}^{-1}(\mathbf{x}_0)\|_\infty^2\|\mathbf{F}(\mathbf{x}_0)\|_\infty}. \tag{17}$$

Finally we obtain the empirical estimate

$$\begin{aligned}\|\mathbf{x}^* - \mathbf{x}_0\|_\infty &\approx \|\mathbf{J}^{-1}(\xi)\mathbf{F}(\mathbf{x}_0)\|_\infty \\ &\leq \frac{\|\mathbf{J}^{-1}(\mathbf{x}_0)\|_\infty\|\mathbf{F}(\mathbf{x}_0)\|_\infty}{1-\lambda\|\mathbf{J}^{-1}(\mathbf{x}_0)\|_\infty^2\|\mathbf{F}(\mathbf{x}_0)\|_\infty}.\end{aligned} \tag{18}$$

The inequality (18) is only an empirical estimate. As stated at the beginning of this subsection, an empirical estimate is only a way expected to speed-up computation in some cases and may not be valid for all examples. We test this empirical estimate by many numerical experiments and report the results in Sect. 4, which show that this empirical estimate performs very well on all those examples. An algorithm based on inequality (18) is described as Algorithm 2.

Algorithm 2 IsComplex

Input: Equation F; Approximation $\mathbf{z}$;
Output: `true` or `false`
1: Compute Formula (18), denote the result by r';
2: **if** `any(` $\mathscr{I}(\mathbf{z}) > r'$ `)` **then**
3: **return** `true`; // not a real root
4: **else**
5: **return** `false`; // may be a real root
6: **end if**

In Algorithm 2, `any()` is a default function in Matlab, which returns true if there is nonzero component in a vector.

3.3 Krawczyk–Moore Interval Iteration

We now discuss about the real root verification with a given interval. In Sect. 2.2.2, we have introduced the Krawczyk operator. With the properties in Proposition 1, we can determine whether an interval contains a real root by the relationship of the original interval and the one after the Krawczyk iteration.

However, in practice, we cannot expect the intervals to be entire inclusion or disjoint after just one iteration. Partly intersection is the most common cases that we

encounter. Since the real root is still in the interval after the Krawczyk iteration, a normal method is to let $\mathbf{X} \cap K(\mathbf{X})$ be the new iteration interval. So suppose $\mathbf{X}^{(0)}$ is the initial interval, the iteration rule is $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} \cap K(\mathbf{X}^{(k)})$, where $K(\mathbf{X}^{(k)})$ is defined by Formula (4). This update rule can make sure that the size of $\mathbf{X}^{(k)}$ is nonincreasing. But a problem is once we encounter $K(\mathbf{X}^{(k)}) \cap \mathbf{X}^{(k)} == \mathbf{X}^{(k)}$, the iteration will be trapped into endless loop. So we have to divide $\mathbf{X}^{(k)}$ if this happened.

Thus, we introduce a bisection function `divide()`. To speedup the convergence of our algorithm, we divide the longest dimension of an interval vector. This strategy may not be the optimal choice when the system's dimension is high. Greedy method or optimization algorithm will be studied in the future.

We now give a formal description of `divide` function in Algorithm 3 and the Krawczyk–Moore iteration process in Algorithm 4.

Algorithm 3 divide

Input: Interval vector $\mathbf{X}$
Output: $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, a decomposition of $\mathbf{X}$
1: Let $\mathbf{X}_i$ be the coordinate with the largest width in $\mathbf{X}$
2: $\mathbf{X}^{(1)} = \mathbf{X}$; $\mathbf{X}^{(2)} = \mathbf{X}$;
3: $\mathbf{X}_i^{(1)} = [\inf(\mathbf{X}_i), \mathrm{mid}(\mathbf{X}_i)]$;
4: $\mathbf{X}_i^{(2)} = [\mathrm{mid}(\mathbf{X}_i), \sup(\mathbf{X}_i)]$;
5: **return** $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$

Algorithm 4 Krawczyk

Input: F; initial box $\mathbf{X}$; isolation boxes $real\_roots$; number of real roots $nreal$
Output: symbol of whether there is a real root $flag$; $real\_roots$; $nreal$
1: $Y = \mathrm{mid}(\mathbf{F}'(\mathbf{X}))^{-1}$; $\mathbf{X}_t = K(\mathbf{X})$, where $K(\mathbf{X})$ is define by Formula (4);
2: **if** $\mathbf{X}_t \cap \mathbf{X} == \emptyset$ **then**
3: **return** $flag =$ `false`;
4: **end if**
5: **while not**($\mathbf{X}_t \subseteq \mathbf{X}$) **do**
6: **if** $\mathbf{X}_t \cap \mathbf{X} == \mathbf{X}$ **then**
7: $[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}] =$ `divide`($\mathbf{X}$);
8: $[f1, real\_roots, nreal] =$ `Krawczyk`($F, \mathbf{X}^{(1)}, real\_roots, nreal$);
9: **if** $f1 ==$ `false` **then**
10: $[f2, real\_roots, nreal] =$ `Krawczyk`($F, \mathbf{X}^{(2)}, real\_roots, nreal$);
11: **end if**
12: **return** $f1$ **or** $f2$;
13: **end if**
14: $\mathbf{X} = \mathbf{X}_t \cap \mathbf{X}$;
15: $Y = (\mathrm{mid}\mathbf{F}'(\mathbf{X}))^{-1}$.
16: $\mathbf{X}_t = K(\mathbf{X})$;
17: **if** $\mathbf{X}_t \cap \mathbf{X} == \emptyset$ **then**
18: **return** flag = `false`;
19: **end if**
20: **end while**
21: $nreal = nreal + 1$;
22: $real\_roots[nreal] = \mathbf{X}_t$
23: **return** $flag =$ `true`, $real\_roots, nreal$;

3.4 Verification and Refinement

After the Krawczyk iteration, we already have all the real root isolation intervals, but these are not the final results. Since we require an isolation of disjoint intervals, we have to check for possible overlaps.

On the other hand, some intervals may not be as small as required by users, so we can narrow them via bisection method until they match the requirement.

We discuss these details in this subsection.

3.4.1 Remove the Overlaps

There is a basic hypothesis: for nonsingular systems, each root has an approximation, and from this approximation, the iteration will end up in its corresponding accurate root, not any other root. So we only have to remove the overlaps, and the number of real roots will not change.

However, we want to expand our algorithm into multi-roots cases. And in that situation, it is possible that two isolated intervals contain the same real root. So whether or not the overlap part contains a real root, our algorithm has its corresponding processes. See Algorithm 5 for details.

Algorithm 5 disjoint_process

Input: Isolated intervals $real\_roots$; number of real roots $nreal$; F
Output: Checked isolated intervals $real\_roots$; $nreal$

$k = 0$;
for $i = 1$ **to** $nreal$ **do**
  $\mathbf{X} = real\_roots[i]$; $new\_root =$ `true`;
  for $j = 1$ **to** k **do**
    $\mathbf{Y} = real\_roots[j]$;
    $\mathbf{Z} = \mathbf{X} \cap \mathbf{Y}$;
    if $\mathbf{Z} == \emptyset$ **then**
      continue;
    end if
    $flag$ = `Krawczyk`(F,$\mathbf{Z}$);
    if $flag ==$ `true` **then**
      $new\_root =$ `false` ; **break**;
    else
      $\mathbf{X} = \mathbf{X} \setminus \mathbf{Z}$;
      $real\_roots[j] = real\_roots[j] \setminus \mathbf{Z}$;
    end if
  end for
  if $new\_root ==$ `true` **then**
    $k = k + 1$; $real\_roots[k] = \mathbf{X}$
  end if
end for
return $real\_roots$,$nreal = k$;

The function `Krawczyk()` in Algorithm 5 is a little bit different from that in the Krawczyk–Moore iteration. In the Krawczyk–Moore iteration, we have to store the information of isolated real root intervals, so the $real\_roots$ and $nreal$ are in the function arguments. However, we only need to know whether there is a real root here, so only the symbol variable $flag$ is returned. The situation is the same in Algorithm 6.

3.4.2 Narrow the Width of Intervals

The user may require that the width of isolation intervals be less than or equal to a prescribed number. Different from symbolic algorithms which can get any precision they want in theory, our floating point number calculation cannot beat the machine precision. In fact, in the Matlab environment that we implement our algorithm, the width cannot be smaller than the system zero threshold.[1]

Algorithm 6 narrowing

Input: Isolated intervals $real\_roots$; Number of real roots $nreal$; F; Threshold τ
Output: $real\_roots$ after bisection

```
for i = 1 to nreal do
  X = real_roots[i];
  while any(rad(X) > τ) do
    [Y(1), Y(2)] = divide(X);
    flag = Krawczyk(F, Y(1));
    if flag == true then
      X = Y(1);
    else
      X = Y(2)
    end if
  end while
  real_roots[i] = X;
end for
return real_roots
```

We also use bisection to do the narrowing job. Since there is only one root in the interval, we only have to continue dividing and checking the half that contains that root. Formal description of this procedure is in Algorithm 6.

3.5 Algorithm Description

Up to now, we have discussed all the parts of real root isolation algorithm in detail. We give the final main program in Algorithm 7.

[1] In Matlab2008b that we do the experiments, the zero threshold is 2.2204e−016.

Algorithm 7 real_root_isolate

Input: Equation $F(\mathbf{x})$; number of variables n; Threshold τ;
Output: Isolated intervals of $F(\mathbf{x}) = 0$ and number of real roots $nreal$

```
[complex_roots, ncomplex] = hom4ps(F,n);
Initialize real_roots to be empty; nreal = 0;
for i = 1 to ncomplex do
  z = complex_roots[i];
  if IsComplex(F, z) then
    continue;
  end if
  r = init_width[F,z,n];
  X0 = midrad(R(z), r);
  [flag, real_roots, nreal] = Krawczyk(F, X0, real_roots, nreal);
end for
[real_roots, nreal] = disjoint_process(real_roots, nreal, F);
real_roots = narrowing(real_roots, nreal, F, τ);
return real_roots, nreal;
```

4 Experiments

Now we apply our new method to some polynomial systems and do some comparison with some former algorithms.

All the experiments are undertaken in Matlab2008b, with Intlab [25] of Version 6. For arbitrarily high accuracy, we can call Matlab's vpa (variable precision arithmetic), but in fact all the real roots of the examples below are isolated by using Matlab's default double-precision floating point. We use Hom4ps-2.0 [19] as our homotopy continuation tool to obtain initial approximate roots.

4.1 Demo Example

We begin our illustration with a simple example.

Example 4 Consider the real root isolation of the system below.

$$\begin{cases} x^3y^2 + x + 3 = 0 \\ 4yz^5 + 8x^2y^4z^4 - 1 = 0 \\ x + y + z - 1 = 0 \end{cases}$$

The homotopy program tells us this system has 28 complex roots in total. And we get the following results after calling our `real_root_isolate` program.

```
intval =
[ - 0.94561016957416, - 0.94561016957415]
[ 1.55873837303161, 1.55873837303162]
[ 0.38687179654254, 0.38687179654255]
```

```
intval =
[ - 1.18134319868123, - 1.18134319868122]
[ - 1.05029487815439, - 1.05029487815438]
[ 3.23163807683560, 3.23163807683561]
intval =
[ - 2.99999838968782, - 2.99999838968781]
[ 0.00024421565895, 0.00024421565896]
[ 3.99975417402886, 3.99975417402887]
intval =
[ - 0.79151164911096, - 0.79151164911095]
[ 2.11038450699949, 2.11038450699950]
[ - 0.31887285788855, - 0.31887285788854]
The order of variables:
'x'
'y'
'z'
The number of real roots: 4
```

We verify the answers above with the DISCOVERER [33] package under Maple, which also return 4 isolated real roots. Here we show its output in floating point number format, i.e.

[[−2.999998391, −2.999998389], [0.0002442132, 0.0002442180], [3.999754090, 3.999754249]],
[[−1.181343199, −1.181343199], [−1.050294975, −1.050294818],[3.231637836, 3.231638372]],
[[−.9456101805, −.9456101656], [1.558738033, 1.558738728], [.3868716359, .3868719935]],
[[−.7915116549, −.7915116400], [2.110384024, 2.110385000], [−.3188729882, −.3188727498]].

And we can see the answers perfectly match the ones of our program.

We list some information during the calculation of our algorithm here for reference. Only the 4 real ones are given, and the other nonreal ones are all detected by our empirical estimate method. We mention that all the imaginary parts of complex roots are significant larger than the initial radius of our algorithm in order of magnitude in this example.

We give some remarks on Table 1. In the first row, *root1* to *root4* are refer to the 4 real roots mentioned above respectively. And B, K, η, h are exactly the same as they

Table 1 Key quantities comparison

	Root1	Root2	Root3	Root4
B	1.060227	1.192159	2.000864	0.874354
K	14.941946	7.198937e+003	4.095991e+003	16.988990
η	4.260422e−016	4.20807e−016	8.882333e−016	5.764449e−016
h	2.024791e−014	1.083446e−011	2.183861e−011	2.568823e−014
Estimate-rad	4.274976e−016	4.208067e−016	8.882344e−016	5.779921e−016
Empirical-rad	1.015249e−015	1.29164e−012	2.156138e−015	1.559270e−015

are defined in Algorithm 1. The *estimate-rad* are the radius obtained via Algorithm 1, while the *empirical-rad* are refer to the ones calculated by Formula (18).

We say a little more words about the *empirical-rad*. Firstly, although the empirical ones are basically larger than the rigorous error radius, they are still small enough, which hardly have any influence on the efficiency of interval iteration. We will see this in the comparison experiments later. But avoiding of interval matrix computation is very helpful to the algorithm. Secondly, the radius obtained from Algorithm 1 are so small that they are even comparable to the zero threshold of Matlab system.[2] And this could bring some uncertainty of floating point operation to our algorithm, such as misjudgement of interval inclusion in Intlab, etc. So we intend to use empirical estimate bound in next experiments.

For system *cyclic6*, the classic symbolic algorithm can do nothing due to the difficulty of triangularization. Meanwhile, we can easily get the 24 isolated real roots intervals with our `real_root_isolate` program.

4.2 Comparison Experiment

Many benchmarks have been checked with our `real_root_isolate` program. Since no results on the time complexity of our method have been obtained, we mainly focus on the isolation results and the program execution time.

We investigate over 130 benchmarks provided by Hom4ps [4], among which about 40 equations are nonsingular systems. Due to space limitation, we only list 11 of them in Tables 2, 4 and 5. Comparison to the symbolic tool DISCOVERER on the

Table 2 Comparison to symbolic method

Problem	Total roots	Real roots	DISCOVERER	Complex roots detected
barry	20	2	2	18
cyclic5	70	10	10	60
cyclic6	156	24	N/A	132
des18_3	46	6	N/A	40
eco7	32	8	8	24
eco8	64	8	N/A	56
geneig	10	10	N/A	0
kinema	40	8	N/A	32
reimer4	36	8	8	28
reimer5	144	24	N/A	120
virasoro	256	224	N/A	32

[2] As mentioned before, the zero threshold in Matlab2008b is 2.2204e−016, which is almost the same order of magnitude of those radiuses.

Table 3 Comparison to a hybrid method

	Verifyrealroot0(M)			Verifyrealroot0(H)			Real_root_isolate		
Problem	Time	Sol	Width	Time	Sol	Width	Time	Sol	Width
comb3000	1.56	1	2.0e−20	1.38	4	2.7e−20	1.11	4	5.3e−10
d1	52.3	2	1.7e−14	6.24	16	1.8e−14	6.19	16	7.3e−13
boon	27.6	1	5.1e−15	1.98	8	2.9e−15	1.14	8	1.8e−14
des22_24	1.79	1	2.5e−14	1.73	10	1.2e−08	4.46	10	1.5e−01
geneig	6.53	2	6.7e−15	4.63	10	2.7e−13	4.21	10	3.6e−13
heart	24.9	2	5.3e−15	1.40	2	4.9e−15	1.54	2	8.5e−15
kin1	52.3	2	1.8e−14	5.91	16	1.8e−14	6.44	16	1.2e−12
ku10	37.8	1	4.7e−14	0.96	2	6.7e−14	0.41	2	6.8e−13
noon3	1.88	1	1.6e−16	11.7	8	1.6e−15	0.94	7	2.6e−15
noon4	9.70	1	3.6e−15	30.2	22	3.9e−15	2.96	15	1.0e−05
puma	5.85	2	2.9e−14	3.99	16	1.8e−13	2.27	16	1.8e−13
quadfor2	1.48	2	5.6e−16	0.71	2	2.2e−16	1.11	2	1.2e−15
rbpl	5.59	1	2.6e−15	23.2	4	8.4e−14	7.78	4	4.1e−12
redeco5	0.95	1	8.3e−17	1.07	4	1.3e−15	0.48	4	1.0e−05
reimer5	26.7	3	8.4e−14	5.83	24	3.2e−13	10.5	24	7.8e−13

11 systems in Table 2 just tries to verify the results and to show the advantage of our hybrid method. Comparison to another hybrid tool in [36] on 15 zero-dimensional nonsingular systems is listed in Table 3.

The 11 systems in Table 2 were computed on a computer with OS: Windows Vista, CPU: Inter®Core 2 Duo T6500 2.10 GHz, Memory: 2G. The column *real roots* in Table 2 tells the number of intervals that our program isolated. Compared with the results of DISCOVERER, the new algorithm indeed works out all equations that are beyond the capability of classic symbolic algorithm. Moreover, the last column show that our empirical estimate method detects all the nonreal roots successfully.

The 15 problems in Table 3 are all from Table 1 of [36]. There are 20 problems in Table 1 of [36], of which 5 are either positive dimensional (e.g., cohn2) or singular (e.g., katsura5). They are beyond the scope of this paper and thus have not been listed here. The data of Columns 2–7 in Table 3 is copied from [36], which was obtained with Matlab(2011R) on a computer with Intel(R) Core(TM) at 2.6 GHz under Windows. The data of `real_root_isolate` (Columns 8–10) was obtained with Matlab(2008b) on a computer with intel(R) core(TM)i3 at 2.27 GHz under Windows 7.

From Table 3, one could see that our program is faster on most of the examples. The widths of the output intervals of our program are usually much larger than those of verifyrealroot0(H) and verifyrealroot0(M). It should be pointed out that our results on systems "noon3" and "noon4" are different from that of verifyrealroot0(H). On the other hand, verifyrealroot0 is a tool applicable also to positive dimensional systems but our tool is only effective on zero-dimensional nonsingular systems.

Table 4 Execution time comparison, unit:s

Problem	Total time	Homotopy time	Interval time
barry	0.421203	0.093601	0.327602
cyclic5	2.948419	0.218401	2.652017
cyclic6	9.984064	0.639604	9.063658
des18_3	4.180827	0.702004	3.385222
eco7	2.371215	0.265202	2.012413
eco8	3.946825	0.499203	3.354022
geneig	4.243227	0.249602	3.868825
kinema	3.946825	1.014006	2.808018
reimer4	2.480416	0.374402	2.059213
reimer5	12.963683	3.073220	9.578461
virasoro	137.124879	4.570829	109.996305

Table 4 shows that interval iterations consume more time than homotopy continuation. The reason is complicated and we enumerate some here:

1. The homotopy continuation focuses only on floating-point number, while the Krawczyk iteration cares about intervals;
2. Hom4ps-2.0 is a software complied from language C, which is much more efficient than the tool that we use to implement our algorithm, say Matlab.
3. The interval iteration time increases as roots number grows since we examine the approximate roots one by one. So the parallel computation of homotopy is much faster.

We believe that with efficient language such as C/C++, and parallel computation, the implementation of our algorithm will be much faster.

In order to verify our idea and see whether parallelization could help, we go into every approximate root's iteration process. Some critical data are recorded in Table 5. The *avg. rad. of ans* is the average radius of the final isolated intervals, while the *avg. rad. of init.* indicates the average radius of the initial intervals. The average time of each root's interval iteration is shown in column *avg. time of iteration* along with the max interval iteration time in *max time of iter*. We think the consumption for each root's process is acceptable.

From Table 5 we can see that the initial interval radii are extremely small, which leads to a nice process time for each iteration. We point out that almost all real root checks are done by just *one* Krawczyk iteration, and hardly any overlap is found after all the Krawczyk iteration processes due to the small initial intervals that we give. All of these save a great deal of executing time of our program.

Table 5 Detail data for each iteration, unit:s

Problem	avg. rad. of ans.	avg. rad. of init.	avg. time of iter.	max time of iter.
barry	3.552714e−015	1.377800e−014	0.054600	0.062400
cyclic5	1.614703e−009	7.142857e−007	0.113881	0.140401
cyclic6	4.440892e−016	2.137195e−015	0.183951	0.234002
des18_3	3.768247e−007	9.737288e−007	0.241802	0.296402
eco7	1.998401e−015	1.483754e−013	0.122851	0.156001
eco8	2.109424e−015	3.283379e−013	0.183301	0.218401
geneig	2.664535e−016	5.721530e−014	0.315122	0.436803
kinema	1.998401e−015	6.784427e−011	0.157951	0.218401
reimer4	1.110223e−016	1.258465e−014	0.122851	0.156001
reimer5	1.110223e−016	4.754080e−014	0.195001	0.421203
virasoro	9.472120e−009	2.265625e−006	0.387844	0.624004

5 Conclusion

In this paper, a new algorithm based on hybrid computation is provided for real root isolation of zero-dimensional nonsingular square polynomial systems. The algorithm first applies homotopy continuation to obtain all the approximate roots of the system. For each approximate root, an initial interval which contains the corresponding accurate root is constructed. Then the Krawczyk operator is called to verify all the initial intervals so as to get all the real root isolation boxes. Some necessary check and refinement work are done after that to ensure the boxes are pairwise disjoint and meet width requirement.

In the construction of initial intervals, we give a rigorous radius error bound based on a corollary of the Kantorovich theorem. Some constructive algorithms are presented for both real and complex approximate roots. Meanwhile, we introduce an empirical estimate radius, which has a nice performance in numerical experiments.

In the modification and implementation of the Krawczyk iteration algorithm, some problems of interval arithmetic are also discussed in this paper.

At last we utilize some existing tools to implement our algorithm under Matlab environment. Many benchmarks have been checked along with comparison and analysis.

We also mention some possible future work here. The construction of initial intervals is still too complicated and further optimization should be studied. Also the empirical estimate with more efficiency and accuracy is a considerable question. The strategy for dividing intervals in the Krawczyk iteration could also be improved, which may be helpful in the high dimension cases. It is important to study the cases that the systems are positive dimensional or singular.

In the aspect of implementation, replacing the Matlab implementation with C/C++ codes may improve the performance of our method. Parallel computation can be

another technique to speed-up the computation since the verification of all initial intervals can be obviously parallelled.

Acknowledgments The work is partly supported by the ANR-NSFC project EXACTA (ANR-09-BLAN-0371-01/60911130369), NSFC-11001040, NSFC-11271034 and the project SYSKF1207 from ISCAS. The authors especially thank professor Dongming Wang for the early discussion on this topic in 2010 and also thank professor T.Y. Li for his helpful suggestions and his team's work on Hom4ps2-Matlab interface. Thanks also go to Ting Gan who computed the 15 systems in Table 3 and provided us suggestion on possible improvements on our program. Thank Zhenyi Ji who shared with us his insights on our hybrid method. Thank the referees for their valuable constructive comments which help improve the presentation greatly.

References

1. Beltran, C.: A continuation method to solve polynomial systems, and its complexity. Numerische Mathematik. Online rst. doi:10.1007/s00211-010-0334-3
2. Beltran, C., Leykin, A.: Certified numerical homotopy tracking. Exp. Math. **21**(1), 69–83 (2012)
3. Beltran, C., Leykin, A.: Robust certified numerical homotopy tracking. Found. Comput. Math. **13**(2), 253–295 (2013)
4. Benchmarks: http://hom4ps.math.msu.edu/HOM4PS_soft_files/equations.zip
5. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer, New York (1997)
6. Boulier, F., Chen, C., Lemaire, F., Moreno Maza, M.: Real root isolation of regular chains. In: Proceedings of ASCM'2009, pp. 15–29 (2009)
7. Cheng, J.-S., Gao, X.-S., Guo, L.-L.: Root isolation of zero-dimensional polynomial systems with linear univariate representation. J. Symbolic Comput. **47**(7), 843–858 (2012)
8. Cheng, J.-S., Gao, X.-S., Li, J.: Root isolation for bivariate polynomial systems with local generic position method. In: Procedings of ISSAC'2009, pp. 103–110
9. Cheng, J.-S., Gao, X.-S., Yap, C.-K.: Complete numerical isolation of real zeros in zero-dimensional triangular systems. In: Proceedings of ISSAC'2007, pp. 92–99 (2007)
10. Collins, G.E., Akritas, A.G.: Polynomial real root isolation using Descartes' rule of signs. In: Proceedings of SYMSAC, pp. 272–275 (1976)
11. Collins, G.E., Loos, R.: Real zeros of polynomials. In: Buchberger, B., Collins, G.E., Loos, R. (eds.) Computer Algebra: Symbolic and Algebraic Computation, pp. 83–94. Springer, New York (1982)
12. Dayton, B., Li, T.Y., Zeng, Z.G.: Multiple zeros of nonlinear systems. Math. Comp. **80**, 2143–2168 (2011)
13. Emiris, I.Z., Mourrain, B., Tsigaridas, E.P.: The DMM bound: multivariate (aggregate) separation bounds. In: Proceedings of ISSAC'2010, pp. 243–250
14. Gragg, G.W., Tapia, R.A.: Optimal error bounds for the Newton-Kantorovich theorem. SIAM J. Numer. Anal. **11**(1), 10–13 (1974)
15. Hauenstein, J.D., Sottile, F.: Algorithm 921: alphaCertified: certifying solutions to polynomial systems. ACM Trans. Math. Softw.(TOMS) **38**(4), 28 (2012)
16. Leykin, A., Verschelde, J., Zhao, A.L.: Newton's method with deflation for isolated singularities of polynomial systems. Theoret. Comput. Sci. **359**, 111–122 (2006)
17. Leykin, A., Verschelde, J., Zhao, A.L.: Higher-order deflation for polynomial systems with isolated singular solutions. In: Dickenstein, A., Schreyer, F., Sommese, A. (eds.) Algorithms in Algeraic Geometry, pp. 79–97. Springer, New York (2008)
18. Li, T.Y.: Numerical solution of multivariate polynomial systems by homotopy continuation methods. Acta Numerica **6**, 399–436 (1997)

19. Li, T. Y.: HOM4PS-2.0. http://hom4ps.math.msu.edu/HOM4PS_soft.htm (2008)
20. Li, T.Y., Wang, X.S.: Solving real polynomial systems with real homotopies. Math. Comput. **60**(202), 669–680 (1993)
21. Mantzaflaris, A., Mourrain, B., Tsigaridas, E.P.: On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. Theor. Comput. Sci. (TCS) **412**(22), 2312–2330 (2011)
22. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. Society for Industrial and Applied Mathematics, Philadelphia (2009)
23. Mujica, J.: Complex Analysis in Banach Spaces. North-Holland Mathematics Studies. Elsevier, Amsterdam (1986)
24. Rouillier, F.: Solving zero-dimensional systems through the rational univariate representation. Appl. Algebra Eng. Commun. Comput. **9**, 433–461 (1999)
25. Rump, S.M.: INTLAB—INTerval LABoratory. In: Csendes, T. (ed.) Developments in Reliable Computing, pp. 77–104. Kluwer Academic Publishers. http://www.ti3.tu-harburg.de/rump/ (1999)
26. Rump, S.M.: Verification methods: rigorous results using floating-point arithmetic. Acta Numerica **19**, 287–449 (2010)
27. Shen, F.: The real roots isolation of polynomial system based on hybrid computation. Master degree thesis, Peking University (April 2012)
28. Sommese, A., Wampler, C.: The Numerical Solution of Systems of Polynomials: Arising in Engineering and Science. World Scientific, Singapore (2005)
29. Strzebonski, A.W., Tsigaridas, E.P.: Univariate real root isolation in multiple extension fields. In: Proceedings of ISSAC'2012, pp. 343–350
30. Verschelde, J.: PHCpack. http://homepages.math.uic.edu/jan/PHCpack/phcpack.html (1999)
31. Wampler, C.: HomLab. http://nd.edu/cwample1/HomLab/main.html
32. Wu, X., Zhi, L.: Computing the multiplicity structure from geometric involutive form. In: Proceedings of ISSAC'2008, pp. 325–332 (2008)
33. Xia, B.: DISCOVERER: a tool for solving semi-algebraic systems. ACM Commun. Comput. Algebra **41**(3), 102–103 (2007)
34. Xia, B., Zhang, T.: Real solution isolation using interval arithmetic. Comput. Math. Appl. **52**, 853–860 (2006)
35. Yang, L., Xia, B.: An algorithm for isolating the real solutions of semi-algebraic systems. J. Symbolic Comput. **34**, 461–477 (2002)
36. Yang, Z., Zhi, L., Zhu, Y.: Verified error bounds for real solutions of positive-dimensional polynomial systems. In: Proceedings of ISSAC'2013 (2013)
37. Zhang, T.: Isolating real roots of nonlinear polynomial. Master degree thesis, Peking University (2004)
38. Zhang, T., Xiao, R., Xia, B.: Real solution isolation based on interval Krawczyk operator. In: Sung-il, P., Park, H. (eds.) Proceedings of ASCM'2005, pp. 235–237 (2005)