Certified Numerical Real Root Isolation of Zero-dimensional Multivariate Real Nonlinear Systems*

Jin-San Cheng† Junyi Wen
KLMM, Institute of Systems Science, Academy of Mathematics and Systems Science, CAS
School of Mathematical Sciences, University of Chinese Academy of Sciences

Abstract

Using the local geometrical properties of a given zero-dimensional square multivariate nonlinear system inside a box, we provide a simple but effective and new criterion for the uniqueness and the existence of a real simple zero of the system inside the box. Based on the result, we design an algorithm based on subdivision and interval arithmetics to isolate all the real zeros of a general real nonlinear system inside a given box. Our method is complete for systems with only finite isolated simple real zeros inside a box. A termination precision is given for general zero-dimensional systems. Multiple zeros of the system are output in bounded boxes. A variety of benchmarks show the effectiveness and efficiency of our implementation (in C++). It works for polynomial systems with Bezout bound more than 100 million. It also works for non-polynomial nonlinear systems. We also discuss the limitations of our method.

Keywords: Real root isolation; real nonlinear system; opposite monotone system; subdivision method; uniqueness and existence.

Mathematics Subject Classification: 13P15 · 14Q30 · 65H10 · 65G40

1 Introduction

Real root isolation of equation systems is a fundamental problem in mathematics and engineering applications. There are many famous symbolic computation methods to solve the problem: the Gröbner basis method [6, 16], the Ritt-Wu characteristic set method [55], the cylindrical algebraic decomposition (CAD) [2] and the resultant method [26] and so on. Though the size of the polynomial systems that can be solved is limited, symbolic methods can get the algebraic representation(s) of all the complex solutions, even including the multiplicity of the solutions. Numerical methods, such as the homotopy continuation method [36], can get all the isolated complex solutions of square polynomial systems (systems with $n$ variables and $n$ polynomials) with large sizes, even for over-determined systems or positive dimensional systems [3]. Though the solutions with traditional homotopy continuation method are lack of certification, the certified homotopy method was proposed [4, 5, 22, 53, 50] in some literatures to overcome the shortcoming based on the famous $\alpha$-theory. To ensure that the output results are reliable, certification with interval Newton’s

*This work was partially supported by the National Key Research and Development Program of China grant 2022YFC3802102.
†Corresponding author: jcheng@amss.ac.cn
method (see \[34, 41, 47\]) for possible real roots after homotopy continuation method is applied in \[48\]. Subdivision method is also used to get the real solutions of the given systems, which are not limited to polynomial case \[7, 8, 25, 31, 27, 32, 38, 44, 45, 49, 57\]. There are many methods to exclude the domains without solutions \[25, 44, 49, 29, 14\]. More requirements are needed to ensure that the boxes are isolating boxes. There are three main methods for certifying a real root: Miranda theorem with Jacobian test, the interval Newton method, and the \(\alpha\)-theory.

Miranda theorem \[39\] and its practical version MK test \[25, 33, 42\] is used for checking the existence of a real zero of a system inside a box. The MK test works well for linear systems \[42\]. Jacobian test \[1\] is used for certifying that a system has at most one real zero inside a box, so does the methods in \[27\]. Thus, the termination of the subdivision method based on the MK test and Jacobian test is guaranteed in a theoretical sense for square systems with simple roots. When MK test is used for systems with more than two variables, it seldom succeeds which can also be found in our experiments. We also analyze the reason in the experiments section. Besides that, the interval Newton method \[34, 41, 47\] and \(\alpha\)-theory \[50\] can work for testing the uniqueness and existence of the complex (or real) zeros. The interval Newton method can verify that a box is an isolating box. In \[31, 32, 15, 28\], the authors use the interval Newton method for real root finding. But the termination of the method is not ensured for certifying a box containing a simple zero or not by successive subdivision. Since an isolating box of a square system may not satisfy the existence and uniqueness condition, the termination condition of subdivision is absent. Thus the theory for root isolation of equation systems based on interval Newton method is not complete even for systems with only simple roots. The \(\alpha\)-theory is slightly different from the other two methods, which computes derivatives with high orders, and the verified domain of \(\alpha\)-theory is a ball, not a box. Similar as the interval Newton method, the termination of the \(\alpha\)-theory method for root isolation of a square system is also not complete. In \[38\], the authors present the concept of the \(\alpha\)-inclusion box and use it for seeking the real roots of a square system. In \[14\], we presented a new method which was based on the geometrical property, the so-called orthogonal monotonicity inside a box for a bivariate polynomial system to certify the existence and uniqueness of a real root inside the box. We used bounding polynomials to exclude the regions which contained no roots. The termination of the method is guaranteed. Thus it can be used for real root isolation inside a box containing only simple real zeros. The method was extended to bivariate nonlinear systems in the journal version \[15\]. We extend the method to general zero-dimensional equation systems in this paper.

In this paper, we present a new existence criterion of a simple real root of a zero-dimensional square system inside a box, which is much easier to succeed than Miranda based criterion. Based on the new criterion, we propose an algorithm to isolate the real roots of a zero-dimensional real nonlinear square system \(F = (f_1, \ldots, f_n)\) inside a given box \(B = [a_1, b_1] \times \ldots \times [a_n, b_n]\), where \(f_i\) and \(\frac{\partial f_i}{\partial x_j}\) are well defined in \(B\) for \(1 \leq i, j \leq n\). In order to analyze the roots of the system locally inside a box, we give the concept of the opposite monotone system (O-M system for short) in an \(n-D\) box and give a criterion to check the existence and uniqueness of a simple real zero of the given system in the given box based on properties of the opposite monotone system. Though the O-M system is firstly presented in \[14\] for 2-D case (It is called orthogonal monotone in \[14\]), the O-M condition for \(n-D\) systems are more complicated than that for 2-D systems.
Simply speaking, for a system $F$ and an $n$-D box $B$, we transform locally the original system $F$ into a new system $G = (g_1, \ldots, g_n)$ with the same zeros as $F$ such that the curve $S = \mathbb{V}(g_1, \ldots, g_{n-1})$ is monotone in $B$ and $S$ intersects transversally with the hypersurface $\mathbb{V}(g_n)$ in $B$ (see Definition 3). The evaluation of the functions of the tangent vector of $S$ on $B$ does not contain zeros ensures its monotonicity inside $B$. The direction of the tangent vector of $S$ and the normal vector of $\mathbb{V}(g_n)$ are almost identical or opposite when evaluating on $B$, which ensures the uniqueness of the root inside $B$. The existence of a real zero of $G$ inside the box $B$ can be determined by the change of the signs of the evaluations of $g_n$ on the two endpoints of $S$ which are the intersection of $S$ and the boundaries of $B$. Some local transformation techniques in MK test [33, 42] are modified and used in our method. The new system in MK test is $J_F(m_B)^{-1}F^T$, here $J_F(m_B)^{-1}$ is the inverse of the Jacobian matrix of $F$ at the middle point of $B$ and $F^T$ is the transpose of $F$. In our method, $G = U J_F(m_B)^{-1}F^T$, where $U$ is an invertible matrix under some requirements. We prove that the termination of the subdivision process for finding all simple real zeros of a system inside a box. So our method is complete for real root isolation of a square nonlinear system inside a bounded box.

Since the existence condition based on the opposite monotone method is used recursively, we revise the original conditions for the opposite monotone system and propose the concept of the strong monotone (S-M) system to avoid constructing opposite monotone systems repeatedly. For boxes which contain multiple real zeros of systems, our method is invalid, thus we give a terminate precision for subdivision process. Therefore, we may get some suspected boxes which reach the terminate precision and do not satisfy the conditions of our method. We give a heuristic verification method to deal with those suspected boxes. Based on our theory, we design an algorithm to isolate the real zeros of a multivariate equation system. We also analyze the complexity of our algorithm. We implement our algorithm in C++. Our experiments show the effectivity and efficiency of our method. We compare our method with some existing methods and analyze some aspects of the methods. Notice that our method can be used for complex root isolation since a complex nonlinear system can be transformed into a real nonlinear system.

The rest of this paper is organized as follows. We introduce some notations and preliminaries in the next section. In Section 3, we give the concepts of O-M system and S-M system in a box and prove the uniqueness and existence theorem, then we show how to match the uniqueness and existence conditions for a given system and a given box. The algorithm of the method is also given in this section. The complexity analysis is also given there. In Section 4, some experiment results are given and an analysis based on the results is shown. We draw a conclusion in the last section.

## 2 Notations and Preliminaries

In this section, we will give some notations, definitions and basic results.

### 2.1 Notations

Let $C^i(\Omega)$ denote the set of all $i$-order continuous differentiable functions defined in $\Omega$, where $\Omega \subset \mathbb{R}^n$ and $\mathbb{R}$ is the field of real numbers. Let $B = [a_1, b_1] \times \ldots \times [a_n, b_n]$ be an $n$-D box in $\mathbb{R}^n$. Let $F_i^l(B) = \{(p_1, \ldots, p_n) \in B | p_i = a_i\}, i = 1, \ldots, n,$
We call \( F^r_i(B) \) or \( F^s_i(B) \) a face of \( B \) for any \( 1 \leq i \leq n \). Let \( m(B) = (\frac{a_i+b_i}{2}, \ldots, \frac{a_n+b_n}{2}) \) be the middle point of the box \( B \) and \( w(B) = \max\{b_1-a_1, \ldots, b_n-a_n\} \) be the width of \( B \). Let \( \partial B = \bigcup_{i=1}^n F_i^1(B) \cup \bigcup_{i=1}^n F_i^r(B) \) be the set of the vertexes of the box and \( \nu(B) = \{(p_1, \ldots, p_n) \in B|p_i = a_i \text{ or } b_i, i = 1, \ldots, n\} \) be the set of the boundaries of the box and \( \nu(B) \) be the set of the vertexes of \( B \).

Let \( F = (f_1(X), \cdots, f_n(X)) \) be a function system, where \( X = \{x_1, \ldots, x_n\} \) are variables and \( f_i(X) \in C^1(B) \). We denote \( F = 0 \) as the equation system \( \{f_1 = 0, \ldots, f_n = 0\} \). If a point \( p \in \mathbb{R}^n \) satisfies \( f_1(p) = \cdots = f_n(p) = 0 \), then we call \( p \) a real root of the function system \( F \). We denote \( \mathbb{I}R \) as the enclosure of the range of a function over a box. A real function \( f(X) \) is usually difficult to be computed exactly. Interval analysis \([40]\) is a useful tool to compute \( f(X) \). By interval analysis, the basic arithmetic operations over intervals are as follows.

\[
\begin{aligned}
I_1 + I_2 &= [a + c, b + d], \\
I_1 - I_2 &= [a - d, b - c], \\
I_1 \times I_2 &= \{\min\{a \ast c, a \ast d, b \ast c, b \ast d\}, \max\{a \ast c, a \ast d, b \ast c, b \ast d\}\}, \\
I_1/I_2 &= [a, b] \ast [1/d, 1/c], 0 \notin I_2.
\end{aligned}
\]

Let \( \Box f \) denote the interval function of \( f \), it has two properties \([37]\):

1. \( f(B) \subset \Box f(B) \),
2. \( \lim_{i \to \infty} \Box f(B_i) = f(\lim_{i \to \infty} B_i) \),

where \( B, B_i \subset \mathbb{R}^n \) and \( \lim B_i = p \), \( p \) is a point in \( \mathbb{R}^n \). There are many different forms of \( \Box f \), for polynomial case, a simple way is just using interval arithmetic \([40]\). For example,

\[
g = x^2 - x, B = [0, 1], \text{ then } \Box g(B) = [0, 1] \ast [0, 1] = [0, 1] - [0, 1] = [-1, 1].
\]

Since \( g(B) = \frac{-1}{4} \), we can find that \( \Box g(B) \) is much bigger than \( g(B) \). Notice that most of the real functions (such as exp, sin, cos, etc.) are also easy to be extended to interval functions.

Since we usually cannot get the exact representation of \( f(B) \), we use it to represent \( \Box f(B) \) for simplicity if there is no doubt in the rest of the paper.
3 Uniqueness and Existence

We will give the uniqueness and existence conditions of a square system containing a simple zero inside a box in this section.

Let \( p \in \mathbb{R}^n \) be an isolated zero of a system \( F \). We always assume that the functions in this section are \( C^1 \) inside the domain we consider. We call \( p \in \mathbb{V}(F) \) a simple zero of \( F \) if \( \det(J_F(p)) \neq 0 \), otherwise we say \( p \) is a singular or multiple zero of \( F \).

3.1 An opposite monotone (O-M) system in a box

The concept of the opposite monotone system inside a box for 2-D case is first presented in [14]. We extend it to \( n \)-D case which is much more complicated. Let \( G = (g_1, \ldots, g_n) \) be a nonlinear system and \( G' = (g_1, \ldots, g_{n-1}) \), our O-M condition is based on the geometric properties of \( \mathbb{V}(G') \). Generally speaking, we know that \( \mathbb{V}(G') \) is a one-dimensional curve in \( \mathbb{R}^n \) if it exists. We denote the tangent vector of \( \mathbb{V}(G') \) at \( p \in \mathbb{V}(G') \) as below:

\[
T_p = (\det(T_1(p)), \ldots, (-1)^{i+1} \det(T_i(p)), \ldots, (-1)^{n+1} \det(T_n(p))),
\]

where \( X_i = X \setminus x_i = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\} \) and \( T_i = J^{X_i}_{G'}, i = 1, \ldots, n \).

We introduce some definitions and lemmas below and then give the concept of the monotonicity of \( S = \mathbb{V}(G') \) in \( B \).

Definition 1. Let \( U = (u_i) \in \mathbb{R}^n \) and \( V = (v_i) \in \mathbb{R}^n \) be \( n \)-dimensional interval vectors. We say \( U \) and \( V \) are matched, if

1. for any \( i \in \{1, \ldots, n\} \), \( 0 \notin u_i \) and \( 0 \notin v_i \).
2. for any \( i, j \in \{1, \ldots, n\} \), \( u_i \cdot v_i \) and \( u_j \cdot v_j \) have the same signs.

For example, \( ([1, 2], [2, 3], [-2, -1]) \) and \( ([-4, -3], [-2, -1], [1, 2]) \) are matched. Let \( U_1, U_2, U_3 \) be interval vectors. It is easily to see that the following properties hold.

1. If \( U_1 \) and \( U_2 \) are matched, \( U_2 \) and \( U_3 \) are matched, then \( U_1 \) and \( U_3 \) are matched.
2. If \( U_1 \) and \( U_2 \) are matched, then \( U_1 \) and \( -U_2 \) are matched.

The following lemma is well-known and can be found in many text books (see Chap. 2 Part. III in [23] for example).

Lemma 1. (Mean Value Theorem) Let \( \Omega \subset \mathbb{R}^n \) be a convex set, \( f = f(x_1, \ldots, x_n) \) is a differentiable function defined on \( \Omega \). Then \( \forall p_1, p_2 \in \Omega, p_1 \neq p_2, \exists \theta \in (0, 1) \) s.t.

\[
f(p_1) - f(p_2) = \nabla f(p_2 + \theta(p_1 - p_2))(p_1 - p_2).
\]

Definition 2. Let \( G' = (g_1, \ldots, g_{n-1}) \) and \( S = \mathbb{V}(G') = \{p \in \mathbb{R}^n | g_1(p) = \cdots = g_{n-1}(p) = 0\} \). We say \( S \) is strong monotonous in \( B \) if \( 0 \notin \det(T_i(B)), i = 1, \ldots, n \) (see [7] for \( T_i \)).

Example 1. \( G' = (x^2/2 + y^2 - 2z^2, x^2/2 + y^2 - 2 - z^2/2 - 1/2) \) and \( B = [0.10, 0.11] \times [0.10, 0.11] \times [0.10, 0.11] \). We have

\[
T_1 = J^{y,z}_{G'} = \begin{pmatrix} 2y & -4z & 0 \\ y & -4z & 0 \end{pmatrix},
T_2 = J^{x,z}_{G'} = \begin{pmatrix} x & -4z & 0 \\ x & -z & 0 \end{pmatrix},
T_3 = J^{x,y}_{G'} = \begin{pmatrix} x & 2y & 0 \\ x & y & 0 \end{pmatrix}.
\]
Thus,
\[
\det(T_1(B)) = \det\left(\begin{bmatrix}
2 & [0.10, 0.11] & -4 \cdot [0.10, 0.11] \\
[0.10, 0.11] & -1 \cdot [0.10, 0.11]
\end{bmatrix}\right)
\]
\[
= -2 \cdot [0.10, 0.11] \cdot [0.10, 0.11] + 4 \cdot [0.10, 0.11] \cdot [0.10, 0.11]
\]
\[
= -2 \cdot [0.01, 0.0121] + 4 \cdot [0.01, 0.0121]
\]
\[
= [-0.0242, -0.02] + [0.04, 0.0484]
\]
\[
= [0.0158, 0.0284].
\]

Similarly, we have \(\det(T_2(B)) = [0.0279, 0.0384]\), \(\det(T_3(B)) = [-0.0142, -0.0079]\). So \(S =\n\)
\(\mathcal{V}(\mathcal{G}')\) is strong monotonous in \(B\) since \(0 \not\in \det(T_i(B))\) for \(i = 1, 2, 3\).

Then, we will prove some nice properties of \(S\) if it is strong monotonous in \(B\). Let \(p = (p_1, \cdots, p_n) \in \mathbb{R}^n\), we define \(\Pi_i(p) = p_i\), for any \(i = 1, \ldots, n\).

**Lemma 2.** Let \(G' = (g_1, \cdots, g_{n-1})\) and \(S = \mathcal{V}(\mathcal{G}')\). If \(S\) is strong monotonous in \(B = [a_1, b_1] \times \cdots \times [a_n, b_n]\), then we have:

(a) \(\forall x_i \in [a_i, b_i], i = 1, \ldots, n\), the hyperplane \(x_i = \hat{x}_i\) intersects \(S\) at most once in \(B\).

Moreover, the hyperplane and \(S\) are not tangent.

(b) \(S\) cannot be a loop in \(B\).

(c) \(\forall p, q \in S \cap B\) and \(p \neq q\), \((\Pi_1(p - q))_{1 \leq i \leq n}\) and \((-1)^{i+1} \det(T_i(B))_{1 \leq i \leq n}\) are matched.

**Proof.** (a) We prove only that \(\forall x_1 \in I_1\), the hyperplane \(x_1 = \hat{x}_1\) intersects \(S\) at most once in \(B\). The case \(i = 2, \ldots, n\) can be proved similarly. Assume that the hyperplane \(x_1 = \hat{x}_1\) intersects \(S\) at two points \(p, p'\) in \(B\). Let \(p - p' = \Delta x = (\Delta x_1, \Delta x_2, \ldots, \Delta x_n)\). Using mean value theorem for every \(g_j\), we have that there exists a point \(q_j \in B\) s.t.

\[
\sum_{i=1}^{n} \frac{\partial g_j}{\partial x_i}(q_j) \Delta x_i = g_j(p) - g_j(p') = 0, j = 1, \ldots, n - 1.
\]

Since \(\Delta x_1 = 0\), thus we have the following equation

\[
\begin{pmatrix}
1 & 0 & \cdots & 0 \\
\frac{\partial g_1}{\partial x_1}(q_1) & \frac{\partial g_1}{\partial x_2}(q_1) & \cdots & \frac{\partial g_1}{\partial x_n}(q_1) \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial g_{n-1}}{\partial x_1}(q_{n-1}) & \frac{\partial g_{n-1}}{\partial x_2}(q_{n-1}) & \cdots & \frac{\partial g_{n-1}}{\partial x_n}(q_{n-1})
\end{pmatrix}
\begin{pmatrix}
\Delta x_1 \\
\Delta x_2 \\
\vdots \\
\Delta x_n
\end{pmatrix}
= 0.
\]

Let \(A\) denote the above matrix, i.e., \(A \Delta x^T = 0\). Since \(\Delta x\) is nonzero, it implies \(\det(A) = 0\). However, since \(S\) is strong monotonous in \(B\), we have \(0 \not\in \det(T_1(B))\), then we have \(\det(A) = 1 \det(M_1) \neq 0\), where \(M_1 \in T_1(B)\). It is a contradiction. Moreover, if \(x_1 = \hat{x}_1\) and \(S\) are tangent at point \(p\) in \(B\), then \(T_1(p) = 0\). It is also a contradiction since \(S\) is strong monotonous in \(B\).

(b) If \(S\) is a loop in \(B\), then \(\exists \hat{x}_i \in I_i\), the hyperplane \(x_i = \hat{x}_i\) must intersect \(S\) at two points in \(B\). It is a contradiction with (a).

(c) For any two point \(p, q\), let \(p - q = \Delta x = (\Delta x_1, \Delta x_2, \ldots, \Delta x_n)\). By Lemma 2(a), we know that \(\Delta x_i \neq 0, \forall i = 1, \ldots, n\), then we need only to prove that for \(i = 2, \ldots, n\), \(\Delta x_1 \det(T_1(B))\) and \(\Delta x_i(-1)^i+1 \det(T_i(B))\) have the same signs.
We consider the following equation system

\[
\begin{pmatrix}
\Delta x_i \\
\frac{\partial g_1}{\partial x_1}(q_1) & \cdots & 0 & \cdots & 0 \\
\vdots & \cdots & \vdots & \cdots & \vdots \\
\frac{\partial g_{n-1}}{\partial x_1}(q_{n-1}) & \cdots & 0 & \cdots & 0 \\
\end{pmatrix}
\begin{pmatrix}
\frac{\partial g_1}{\partial x_1}(q_1) & \cdots & \frac{\partial g_1}{\partial x_n}(q_1) \\
\vdots & \cdots & \vdots \\
\frac{\partial g_{n-1}}{\partial x_1}(q_{n-1}) & \cdots & \frac{\partial g_{n-1}}{\partial x_n}(q_{n-1}) \\
\end{pmatrix}
\begin{pmatrix}
\Delta x_1 \\
\Delta x_2 \\
\vdots \\
\Delta x_n \\
\end{pmatrix} = 0.
\]

Let \( A' \) denote the above matrix, whose last \( n - 1 \) rows are the same as \( A \), we have

\[
\det(A') = \Delta x_i \det(M_1) - \Delta x_1 (-1)^{i+1} \det(M_i),
\]
where \( M_1 \in T_1(B), M_i \in T_i(B) \). If

\[
\Delta x_1 \det(T_1(B)) \cdot \Delta x_1 (-1)^{i+1} \det(T_i(B)) < 0,
\]
we have \( \Delta x_i \det(M_1) \) and \( \Delta x_1 (-1)^{i+1} \det(M_i) \) have different signs, then \( \det(A') \neq 0 \), and \( \Delta x = 0 \). It is a contradiction with \( p, q \) are two different points. Therefore,

\[
\Delta x_1 \det(T_1(B)) \cdot \Delta x_1 (-1)^{i+1} \det(T_i(B)) > 0.
\]

Notice that \( \Delta x_i \) and \( \det(M_i) \) are all nonzero for \( i = 1, \ldots, n \). Then we know that for any \( j = 1, \ldots, n \), \( \Delta x_j (-1)^{j+1} \det(T_j(B)) \) have the same signs, i.e., \( (\Pi_i(p - q))_{1 \leq i \leq n} \) and \( ((-1)^{i+1} \det(T_i(B)))_{1 \leq i \leq n} \) are matched.

Now, we will give the definition of an O-M system in a box \( B \).

**Definition 3.** Let \( M \in \mathbb{R}^{n \times n} \). For any \( i = 1, \ldots, n \), we denote \( M_{(n,i)} \in \mathbb{R}^{(n-1) \times (n-1)} \) as a sub-matrix of \( M \) by deleting the \( n \)-th row and \( i \)-th column and we denote \( M_{n,i} \) as \( n \)-th row of \( M \). We say \( M \) is an O-M matrix if \( ((-1)^{i+n} \det(M_{(n,i)}))_{1 \leq i \leq n} \) and \( M_{n,i} \) are matched.

**Remark 1.** If \( M \in \mathbb{R}^{n \times n} \) is a matrix and satisfies the above conditions, we also regard \( M \) as an O-M matrix.

**Example 2.** Let \( M = \begin{pmatrix} 1,2 & 3,4 & -1,1 \\ 3,4 & -1,1 & 5,6 \\ 1,2 & -2,1 & -2,-1 \end{pmatrix} \). We have

\[
(-1)^{1+3} \det(M_{(3,1)}) = [14, 25],
\]

\[
(-1)^{2+3} \det(M_{(3,2)}) = [-16, -1],
\]

\[
(-1)^{3+3} \det(M_{(3,3)}) = [-18, -7],
\]

and

\[
M_{3,:} = ([1,2], [-2, -1], [-2, -1]).
\]

Therefore, \( M \) is an O-M matrix.

**Definition 4.** Let \( F = (f_1, \ldots, f_n) \), \( F' = (f_1, \ldots, f_{n-1}) \) and \( S = \forall(F') \). We say \( F \) is an O-M system in \( B \) if

1. \( S \) is strong monotonous in \( B \).

2. \( ((-1)^{i+n}T_i(B))_{1 \leq i \leq n} \) and \( (\frac{\partial f_i}{\partial x_i}(B))_{1 \leq i \leq n} \) are matched, where \( T_i = J_{F_i}^{X_i} \) (see [1] for more details).
i.e., $J^N_F(B)$ is an O-M matrix.

We can use interval evaluation to get $J^N_F(B)$ and check whether it is an O-M matrix or not.

Theorem 3. Let $F = (f_1, \ldots, f_n)$ and $B$ a box. If $F$ is an O-M system in $B$, then $F$ has at most one zero in $B$.

Proof. Assume that $F$ has two different real zeros $p = (p_1, \ldots, p_n), q = (q_1, \ldots, q_n)$ in $B$. Using Mean Value Theorem for $f_n$, there exists a point $p' \in B$ such that

$$\sum_{i=1}^{n} \frac{\partial f_n}{\partial x_i}(p') \cdot \Pi_i(p - q) = f_n(p) - f_n(q) = 0.$$ 

However, by Lemma 2, since $S$ is strong monotonous in $B$, we have $(\Pi_i(p - q))_{1 \leq i \leq n}$ and $((-1)^{i+1} \det(T_i(B)))_{1 \leq i \leq n}$ are matched. By the definition of O-M system, we have $((-1)^{i+1} \det(T_i(B)))_{1 \leq i \leq n}$ and $(\frac{\partial f_i}{\partial x_i}(B))_{1 \leq i \leq n}$ are matched. Therefore, $(\Pi_i(p - q))_{1 \leq i \leq n}$ and $(\frac{\partial f_i}{\partial x_i}(B))_{1 \leq i \leq n}$ are matched. Then, we know that

$$\forall q' \in B, \sum_{i=1}^{n} \frac{\partial f_n}{\partial x_i}(q') \cdot \Pi_i(p - q) \neq 0.$$ 

It is a contradiction. Hence $F$ has at most one zero in $B$.

3.2 Preconditioner

In this subsection, we will find an equivalent system of the original system inside a box, that is, two systems have the same solution(s) inside the box, such that the new system satisfies the O-M condition inside the box.

We give the following lemma first. Though it is clear, we give the proof below.

Lemma 4. Let $F = (f_1, \ldots, f_n)$ and $M \in \mathbb{R}^{n \times n}$ be an $n \times n$ invertible matrix. Then $\nabla(F) = \nabla(MF^T)$, where $F^T$ is the transpose of $F$.

Proof. On one hand, $\forall p \in \nabla(F)$, we have $F(p) = 0$, then $MF^T(p) = 0$ i.e., $p \in \nabla(MF^T)$. On the other hand, $\forall p \in \nabla(MF^T)$, we have $MF^T(p) = 0$. Since $M$ is an invertible matrix, then we have $F^T(p) = M^{-1}MF^T(p) = 0$ i.e., $p \in \nabla(F)$. Thus, we have $\nabla(F) = \nabla(MF^T)$.

We give the preconditioner which transforms locally a square system $F$ into a new system $UJ_F^{-1}(p) \cdot F^T$, where $U \in \mathbb{R}^{n \times n}$ is an O-M matrix (In the rest of the paper, $U$ always denotes an O-M matrix) and $p \in B$. In general, we choose $p$ as $m(B)$. The idea of multiplying $J_F^{-1}(p)$ to the original system originates from [33] and used by [21, 42, 44]. They just transform locally the system $F$ into $J_F^{-1}(p) \cdot F^T = (f_1, \ldots, f_n)^T$ s.t. $\forall (f_i)$ are almost orthogonal to each other in the neighborhood of $p$. Then the Miranda theorem can be used to check the existence of a real zero of the system. But we further do a rotation on $J_F^{-1}(p) \cdot F^T$ by multiplying the matrix $U$ to make the system $UJ_F^{-1}(p) \cdot F^T$ becoming an O-M system in $B$. See the example below for illustration. A similar example to illustrate the same problem can be found in [44].
Example 3. Let $F = (2y^2 - z^2 + 2x + 0.16, x^2 + x + 3y - z - 0.02, 3x^2 - 5.08x - 0.2492 - 4y^2 - 6z + 3z^2 + 3y)$, $B = [-0.09, -0.04] \times [0.01, 0.06] \times [0.01, 0.06]$, $U = \begin{pmatrix} 3 & 1 & 1 \\ 1 & -3 & 1 \\ 1 & 1 & -3 \end{pmatrix}$.

$F$ has a unique zero $(-0.080966, 0.049827, 0.055071)$ in $B$. However, we can find that $F$ is not an O-M system in $B$. The preconditioner in $\mathbb{R}^3$ produces $G' = F^{-1}(m(B)) \cdot F^T = \begin{pmatrix} 1.09864y^2 - 0.570758z^2 + 1.11937x + 0.0864214 - 0.0673992x^2 + 0.131059z - 0.0715789y \\ -2.42325y^2 + 1.62329z^2 - 2.93426x - 0.163387 + 1.1658x^2 - 2.40357z + 1.03577y \\ -2.02830y^2 + 1.22491z^2 - 2.45787x - 0.143304 + 0.430319x^2 - 1.0658z + 0.026427y \end{pmatrix}$, but our preconditioner generates $G = UJ_F^{-1}(m(B)) \cdot F^T = \begin{pmatrix} -1.15563y^2 + 1.13593z^2 - 2.03402x - 0.047427 + 1.39868x^2 - 3.07297z + 0.847460y \\ 6.34009y^2 - 4.21572z^2 + 7.46428x + 0.433278 - 3.14216x^2 + 6.27919z - 3.15246y \\ 4.76029y^2 - 2.62220z^2 + 5.55872x + 0.352946 - 0.18912x^2 + 0.91523z + 0.884910y \end{pmatrix}$. We can find that both $F'$ and $G'$ do not satisfy the condition of Miranda theorem. But it is easy to check that $G$ is an O-M system in $B$.

Figure 1: The left figure is of $F = 0$, the middle one is of $G' = 0$, the right one is of $G = 0$.

Remark: We have many choices for the O-M matrix $U$. For example, when $n = 2$, \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} and \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix} are both O-M matrices. When $n = 3$, \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & -3 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & -3 \end{pmatrix} and \begin{pmatrix} 1 & -3 & 1 \\ 1 & 1 & -3 \end{pmatrix} are all O-M matrices. We will discuss how to select a better rotation matrix later.

For each simple zero $p^*$ of $F$, we will prove that there always exists a small box $B$ containing $p^*$ s.t. $UJ_F^{-1}(m(B)) \cdot F^T$ is an O-M system in $B$.

For a point $p \in \mathbb{R}^n$ and a positive number $\delta > 0$, we define a set of box as $B(p, \delta) = \{B|B$ is a box and $p \in B, w(B) < \delta\}$. Then, we have the following theorem:

Theorem 5. Let $F = (f_1, \ldots, f_n)$, $f_i \in C^1(B), 1 \leq i \leq n$ and $p^*$ a simple zero of $F$. Then, $\exists \delta > 0$ s.t. $\forall B \in B(p^*, \delta)$, $UJ_F^{-1}(m(B)) \cdot F^T$ is an O-M system in $B$.

Proof. Let $G = UJ_F^{-1}(m(B)) \cdot F^T$, when $B \rightarrow p^*$, we have \begin{align*} \lim_{B \rightarrow p^*} J_G(B) &= \lim_{B \rightarrow p^*} UJ_F^{-1}(m(B)) \cdot J_F(B) = UJ_F^{-1}(p^*) \cdot J_F(p^*) = U. \end{align*} Since $U$ is an O-M matrix, we prove the theorem.
Remark: If \( p^* \) is a singular root, then for any box \( B \) containing \( p^* \), we can not transform \( F \) into an O-M system in \( B \) since \( 0 = \det(J_F(p^*)) \in \det(J_F(B)) \). Hence, our method is invalid for singular roots.

For an O-M system in \( B \), it has at most one real zero in \( B \) by Theorem 3. We will discuss how to determine whether \( F \) does have a zero in \( B \) in the next subsection.

### 3.3 Existence

In this subsection, we will give a method to determine whether an O-M system has one zero inside a box or not.

**Lemma 6.** Let \( G' = (g_1, \cdots, g_{n-1}) \) and \( S = \mathcal{V}(G') \). If \( S \) is strong monotonous in \( B = [a_1, b_1] \times \cdots \times [a_n, b_n] \), then we have that \( B \) contains at most one connected component of \( S \).

**Proof.** Assume that \( B \) contains two connected components of \( S: L, L' \). By Lemma 2 (a) and (b), we know that \( L \) intersects the boundaries of \( B \) at most twice. We prove the lemma in the following two cases:

1. If \( L \) intersects the boundaries of \( B \) twice, there are two cases:
   1.1. \( L \) passes \( B \) from \( F_i^l(B) \) to \( F_i^r(B) \). For a point \( q \in L' \), we have \( \pi_i(q) \in [a_i, b_i] \). Since \( L \) is a connected component, \( \exists p \in L \) such that \( \Pi_i(p) = \Pi_i(q) \). It is a contradiction with Lemma 2.
   1.2. \( L \) passes \( B \) from \( F_i^l(B) \) to \( F_j^r(B) \), \( i \neq j \). Let \( L \cap F_i^l(B) = \{p_1\} \) and \( L \cap F_j^r(B) = \{p_2\} \), then we have \( \Pi_i(p_1) = a_i \) and \( \Pi_j(p_2) = b_j \). For a point \( q \in L' \), since \( L \) is a connected component and Lemma 2 (a), we have \( \Pi_i(q) \notin [a_i, \Pi_i(p_2)] \) i.e., \( \Pi_i(q) > \Pi_i(p_2) \) and obviously \( \Pi_j(q) < b_j = \Pi_j(p_2) \). Then we have

\[
\Delta x_i = \Pi_i(p_1) - \Pi_i(p_2) < 0, \Delta x_j = \Pi_j(p_1) - \Pi_j(p_2) < 0,
\]

\[
\Delta x_i' = \Pi_i(q) - \Pi_i(p_2) > 0, \Delta x_j' = \Pi_j(q) - \Pi_j(p_2) < 0.
\]

Thus we have \( \Delta x_i \cdot \Delta x_j > 0 \) and \( \Delta x_i' \cdot \Delta x_j' < 0 \). However, by Lemma 2 (c), \( \forall p \neq q \in S \cap B, \Pi_i(p - q)(-1)^{i+1} \det(T_i(B)) \Pi_j(p - q)\det(T_j(B)) > 0 \). Since the sign of \( \Pi_i(p - q)\Pi_j(p - q) \) is unchanged, we know the sign of \( \Pi_i(p - q)\Pi_j(p - q) \) is unchanged too. Therefore we get a contradiction. Notice that \( L \) passes \( B \) from \( F_i^l(B) \) to \( F_j^r(B) \) (\( F_i^l(B) \) to \( F_j^r(B) \)) can be proved in the similar way.

2. If \( L \) and \( L' \) both intersect the boundaries of \( B \) only once, we know that the intersection points must be vertexes of \( B \). Let \( L \cap \partial B = \{p\}, L' \cap \partial B = \{p'\} \). Without loss of generality, assume that \( p = (a_1, \cdots, a_n) \), then by Lemma 2 (a), \( p' \) must be \( (b_1, \cdots, b_n) \). Considering the tangent vector of \( L \) at \( p \), by Lemma 2 (c), we have \( \Pi_i(p - p') \) and \( \Pi_i(p - q) \) are matched, therefore the sign of the tangent vector \( (\text{Sign}((-1)^{i+1} \det(T_i(B)))) = (1, 1, \cdots, 1) \) or \( (-1, -1, \cdots, -1) \). That is to say \( L \) will go forward to the inside of \( B \). Then \( L \) must go out of \( B \) and intersects the boundaries of \( B \) at another point. It is contradiction with the assumption. Notice that if \( p \) is another vertex, we can prove it in a similar way. Thus, we prove the lemma.

By Lemma 2, we know that if \( S \) is strong monotonous in \( B \), then \( S \) intersects the boundaries of the box \( \partial B \) at most twice, that is to say \( \#(S \cap \partial B) \leq 2 \), where \( \#(A) \) denotes the number of elements of a set \( A \). Next, we determine whether \( F \) has a root or not in \( B \) based on the three cases: \( \#(S \cap \partial B) = 0, 1, 2 \).
If \( \#(S \cap \partial B) = 0 \), since \( S \) can not be a loop in \( B \), then we know that \( S \) dose not pass the box \( B \). If \( \#(S \cap \partial B) = 1 \), we know that \( S \) intersects only the box at a point and this point must be a vertex of \( B \). Next, we will analyze the last case: \( \#(S \cap \partial B) = 2 \).

**Lemma 7.** Let \( F = (f_1, \ldots, f_n) \) be an O-M system in \( B \). Let \( F' = (f_1, \ldots, f_{n-1}) \) and \( S = \nabla(F') \). Assume that \( S \cap \partial B = \{p_1, p_2\} \), we have:

1. If \( f_n(p_1)f_n(p_2) \leq 0 \), \( F = 0 \) has a unique root in \( B \).
2. If \( f_n(p_1)f_n(p_2) > 0 \), \( F = 0 \) has no root in \( B \).

**Proof.** Since \( B \) contains a unique component of \( S \), we can parameterize it as \( \eta(t) = (x_1(t), \ldots, x_n(t)) \), \( t_1 \leq t \leq t_2 \), where \( \eta(t_i) = p_i, i = 1, 2 \). Since \( S \) is strong monotone inside \( B \), \( \eta(t)(t_1 \leq t \leq t_2) \) is strictly contained inside \( B \). Consider the univariate function \( g(t) = f_n(x_1(t), \ldots, x_n(t)) \). If \( f_n(p_1)f_n(p_2) \leq 0 \), i.e. \( g(t_1)g(t_2) \leq 0 \), then \( \exists t' \in [t_1, t_2] \) s.t. \( g(t') = 0 \). Thus \( (x_1(t'), \ldots, x_n(t')) \) is a root of \( F = 0 \). Since \( F \) is an O-M system in \( B \), we know that \( F = 0 \) has a unique root in \( B \). If \( f_n(p_1)f_n(p_2) > 0 \), i.e. \( g(t_1)g(t_2) > 0 \), then there are even number roots (counting multiplicity) of \( g(t) = 0 \) in \([t_1, t_2] \). Since \( F = 0 \) has most one root in \( B \), we can know that \( F = 0 \) has no root in \( B \).

**3.4 Checking the existence**

We will show how to check the existence of a real zero of a system in a box in this subsection.

Let \( F = (f_1, \ldots, f_n) \) be an O-M system in \( B \) and \( F' = (f_1, \ldots, f_{n-1}) \). By the definition of the O-M system, we know that \( S = \nabla(f_1, \ldots, f_{n-1}) \) is strong monotone in \( B \). Then by Lemma 7, in order to check the existence, we need to compute the intersection points \( p_1, p_2 \) of \( S \) and \( \partial B \). However, it is not easy and unnecessary to get the points exactly, we only need to get two \((n - 1)\)-boxes containing \( p_1, p_2 \), say \( B_{p_1}, B_{p_2} \). Then by computing \( \text{Sign}(f_n(B_{p_1})), \text{Sign}(f_n(B_{p_2})) \), we can get \( \text{Sign}(f_n(p_1)), \text{Sign}(f_n(p_2)) \). The signs of \( f_n(B_{p_1}), f_n(B_{p_2}) \) can be obtained by interval computation, since \( 0 \notin \partial_x f_n(B_{p_1}, B_{p_2}) \), we can use the following lemma to compute the signs in an easier way.

**Lemma 8.** Let \( f(x_1, \ldots, x_n) \in C^1(B) \) and \( B = [a_1, b_1] \times \cdots \times [a_n, b_n] \) a box. If \( 0 \notin \partial_x f_n(B), i = 1, \ldots, n \) and \( f(v(B)) > 0(0) \), then \( f(B) > 0(0) \).

**Proof.** WLOG, we assume that \( f(v(B)) > 0 \) since \( f(v(B)) < 0 \) can be proved similarly. We prove the lemma with mathematical induction:

- For \( n = 1 \), i.e., \( B = [a_1, b_1] \): Since \( \partial_x f_n(B) > 0 \), we know that \( f \) is a monotonous univariate function, then we have \( f(B) > 0 \) since \( f(a_1) > 0 \) and \( f(b_1) > 0 \).

- Next we assume that the lemma is proved for \( n = k - 1 \) and we are going to prove the case \( n = k \), where \( k \) is a positive integer. Let \( B_i = [a_1, b_1] \times \cdots \times [a_{i-1}, b_{i-1}] \times [a_{i+1}, b_{i+1}] \times \cdots \times [a_n, b_n] \subset \mathbb{R}^{n-1}, i = 1, \ldots, n \) and \( t_i = a_i \) or \( b_i \), \( 1 \leq i \leq n \), since \( f|_{x_{i} = t_i}(v(B_i)) \subset f(v(B)) \), we have \( f|_{x_{i} = t_i}(v(B_i)) > 0 \). Then by the assumption we have \( f|_{x_{i} = t_i}(B_i) > 0 \) since the function \( f|_{x_{i} = t_i} \) is with \( k - 1 \) variables. In summary, we have \( f(\partial(B)) > 0 \). \( \forall p_i = [a_1, b_1] \), we have \( f|_{x_{i} = p_i}(v(B)) \subset f(\partial(B)) \), thus \( f|_{x_{i} = p_i}(v(B)) > 0 \). Then by the assumption we have \( f|_{x_{i} = p_i}(B_i) > 0 \). By the arbitrariness of \( p_i \), we have \( f(B) > 0 \).

By Lemma 8, we need only to compute the signs of \( f_n \) at the vertexes of \( B_{p_1}, B_{p_2} \). If \( f_n(v(B_{p_1})), f_n(v(B_{p_2})) \) are all positive or negative, then we know the signs of \( f_n(B_{p_1}), f_n(B_{p_2}) \). So do the signs of \( f_n(p_1), f_n(p_2) \).
Next, we will show how to check the existence of a real zero of the system in $B$. Let $B = [a_1, b_1] \times \cdots \times [a_n, b_n]$ and $B_i = [a_i, b_i] \times \cdots \times [a_{i-1}, b_{i-1}] \times [a_{i+1}, b_{i+1}] \times \cdots \times [a_n, b_n] \subseteq \mathbb{R}^{n-1}, i = 1, \ldots, n$. For all $p_i \in [a_i, b_i]$, let $F'|_{x_i=p_i} = (f_1|_{x_i=p_i}, \ldots, f_{n-1}|_{x_i=p_i}) = (f_1(x_1, \ldots, p_i, \ldots, x_n), \ldots, f_{n-1}(x_1, \ldots, p_i, \ldots, x_n))$. For convenience, we define the index set of the box $B$ with $2n$ elements as follows

$$\text{Ind}(B) = \{(i, a_i), (i, b_i) | i = 1, \ldots, n\}.$$ 

If $S \cap \partial B = \{p_1, p_2\}$, it is easy to see that there exist $(i_1, t_1), (i_2, t_2) \in \text{Ind}(B)$ such that $F'|_{x_{i_1}=t_1}, F'|_{x_{i_2}=t_2}$ both have a unique zero in $B_{i_1}, B_{i_2}$ respectively. Therefore, for $(i, t) \in \text{Ind}(B)$, we need to know the square system $F'|_{x_i=t}$ has a zero in $B_i$ or not. We can solve those systems by our method recursively. Finally, we need to isolate some bivariate systems, which is solved in [14]. Notice that for non-polynomial case, the method is also valid. We give the algorithm structure of checking existence as following:

1. Solve systems $F'|_{x_i=t}$ in $B_i$ where $(i, t) \in \text{Ind}(B)$.
2. If $\forall (i, t) \in \text{Ind}(B), F'|_{x_i=t}$ has no zero in $B_i$, it means that $S$ does not pass the box $B$ and $F$ has no zero in $B$.
3. Else if $\exists (i_1, t_1), (i_2, t_2) \in \text{Ind}(B)$ such that both $F'|_{x_{i_1}=t_1}$ and $F'|_{x_{i_2}=t_2}$ have a unique zero in $B_{i_1}, B_{i_2}$ respectively. Then we compute the signs of $f|_{x_{i_1}=t_1}(v(B_{i_1}))$ and $f|_{x_{i_2}=t_2}(v(B_{i_2}))$.
   (a) If $\text{Sign}(f|_{x_{i_1}=t_1}(v(B_{i_1}))) \text{Sign}(f|_{x_{i_2}=t_2}(v(B_{i_2}))) < 0$, $F$ has a unique zero in $B$.
   (b) Else, if $\text{Sign}(f|_{x_{i_1}=t_1}(v(B_{i_1}))) \text{Sign}(f|_{x_{i_2}=t_2}(v(B_{i_2}))) > 0$, $F$ has no zero in $B$.
   (c) Else if we cannot determine the sign of $f|_{x_{i_1}=t_1}(v(B_{i_1}))$ or $f|_{x_{i_2}=t_2}(v(B_{i_2}))$, we need to refine the boxes $B_{i_1}, B_{i_2}$ and check the signs again.
4. Else, we cannot determine if $F$ has a real zero in $B$ or not. We need to subdivide the box and check again.

In Step(1), we will solve $2n$ systems with $n - 1$ equations and $n - 1$ variables in the worst case, that is to say, we need to solve $2n \times 2(n - 1) \times \cdots \times 2 \cdot 2 = 2^n - 1 \cdot n!$ univariate functions recursively in the worst case.

In Step(3-c), if we cannot determine the signs of $f|_{x_{ij}=t_j}(v(B_{ij})), j = 1, 2$, we will refine the box $B_{ij}$. That is to say, we will subdivide $B_{ij}$ and find the unique box containing the zero of $F'|_{x_{ij}=t_j}$ in those sub-boxes. The worst case is that the width of refined box is very small, but we still cannot determine the signs. This case happens when the zero $p^*$ of $F$ is on (or very close to) the boundaries of $B$, we will discuss the bad situation later.

Based on the above discussion, if a system and a box satisfy our O-M and existence conditions, we can determine that the box is an isolating box of the system. We can design an algorithm for real root isolation based on the O-M condition and the subdivision method. However, from the discussion above, we know that a recursive solving is required in our existence checking step. We need to ensure that the related system in the related box is an O-M system. In order to avoid constructing an O-M system for each one by one, we prefer to construct all these systems in their related boxes together. In doing so, we introduce the Jacobian test. One can find that the O-M condition implies the famous Jacobian test based on the following theorem. One can find it in many places, see the corollary of Theorem 12.1 in [1].
**Theorem 9.** Let $G = (g_1, \ldots, g_n), g_i \in C^1(B)$. If $0 \notin \det(J_G^X(B))$, then $G = 0$ has at most one real root in $B$.

It is clear that the Jacobian test is weaker than our conditions in Definition 3. The Condition (1) in Definition 3 is for existence checking. The Condition (2) in Definition 3 is for uniqueness checking, but it is not necessary. We can replace it with a weaker condition as the Jacobian test. In fact, when we check the existence, for $(i, t) \in Ind(B)$, we need to know whether the system $F'(x_t)$ has a zero in $B$ or not. In some cases, the system $F'(x_t)$ may not be an O-M system in the related box, thus we cannot use the existence condition directly and need an $(n-1)$-D O-M matrix $U$ to help construct O-M system again. This operation may take much time, which is not necessary. In order to improve the situation, we present the Strong Monotone (S-M) system revised from O-M system.

### 3.5 Strong Monotone system (S-M)

Giving an interval matrix $M \in \mathbb{IR}^{n \times n}$, we denote $M_i \in \mathbb{IR}^{i \times n}(1 \leq i \leq n)$ as the first $i$ rows of $M$ and denote $S(M_i)$ as the set of all $i$-order sub-matrices of $M_i$. We give the following definition:

**Definition 5.** Let $M \in \mathbb{IR}^{n \times n}$ and $M = (m_{ij})_{1 \leq i,j \leq n}$. We say $M$ is an S-M matrix if

- $0 \notin \det(A), \forall A \in S(M_i), 1 \leq i \leq n$.

**Remark 2.** If $M \in \mathbb{IR}^{n \times n}$ is a matrix and satisfy the above conditions, we also regard $M$ as an S-M matrix.

**Example 4.** Let $M = (m_{ij}) = \begin{pmatrix} 3, 4 & 1, 2 & 1, 2 \\ 1, 2 & 3, 4 & [-2, -1] \\ 1, 2 & [-2, -1] & [-2, -1] \end{pmatrix}$. We can verify that $M$ is an S-M matrix. First, we have $0 \notin m_{1j}, 1 \leq j \leq 3$. Then for $i = 2$, we have that $M_2 = \begin{pmatrix} 3, 4 & 1, 2 & 1, 2 \\ 1, 2 & 3, 4 & [-2, -1] \\ 1, 2 & [-2, -1] & [-2, -1] \end{pmatrix}$ and $S(M_2) = \{A_1, A_2, A_3\} = \{\begin{pmatrix} 1, 2 & 1, 2 \\ 3, 4 & [-2, -1] \end{pmatrix}, \begin{pmatrix} 3, 4 & 1, 2 \\ 1, 2 & [-2, -1] \end{pmatrix}, \begin{pmatrix} 3, 4 & 1, 2 \\ 1, 2 & 3, 4 \end{pmatrix}\}$. After computing, we have $\det(A_1) = [-12, -4], \det(A_2) = [-12, -4]$ and $\det(A_3) = [5, 15]$. For $i = 3$, we have $M_3 = M$, $S(M_3) = \{M\}$ and $\det(M) = [-72, -16]$. Therefore, $M$ is an S-M matrix.

In the following, we always denote $V$ as an S-M matrix, then we will introduce the definition of the S-M system.

**Definition 6.** Let $F = (f_1, \ldots, f_n), f_i \in C^1(B), 1 \leq i \leq n$ and $B$ a box. We call $F$ an S-M system in $B$ if $J_F^X(B)$ is an S-M matrix.

**Theorem 10.** Let $F = (f_1, \ldots, f_n), f_i \in C^1(B), 1 \leq i \leq n$. If $F$ is an S-M system in $B$, then $F = 0$ has at most one root in $B$.

**Proof.** Since $F$ is an S-M system in $B$, we have $0 \notin \det(J_F^X(B))$. Then by Theorem 9 we know that $F = 0$ has at most one root in $B$.

For simplification, we set $A = \text{IsSMSys}(F, X, B)$ as the algorithm to check whether $J_F^X(B)$ is an S-M system or not. Interval evaluation can be used here to get $J_F^X(B)$.

Similar as Theorem 5, we have the following result.
Theorem 11. Let $F = (f_1, \ldots, f_n)$, $f_i \in C^1(B)$, $1 \leq i \leq n$, and $p^*$ a simple zero of $F$. Then, $\exists \delta > 0$ s.t. $\forall B \in B(p^*, \delta)$, $VJ^{-1}_F(m(B)) \cdot F^T$ is an S-M system in $B$, where $V$ is a S-M matrix.

Now, we will show how to check the existence for an S-M system inside a box. We give the following lemma first.

Lemma 12. Let $F = (f_1, \ldots, f_n)$, $f_i \in C^1(B)$, $1 \leq i \leq n$, $B = I_1 \times \cdots \times I_n$, $F' = (f_1, \ldots, f_{n-1})$. If $F$ is an S-M system in $B$, then for any $\tilde{x}_i \in I_i$, $i = 1, \ldots, n$, $F'|_{x_i = \tilde{x}_i}$ is still an S-M system in $I_1 \times \cdots \times I_{i-1} \times I_{i+1} \times \cdots \times I_n$.

Proof. It can be derived from Definition 6 directly.

Let $F$ be an S-M system in $B$ and $S = \mathbb{V}(F')$. By the definition of S-M system, we know that $S$ is strong monotone in $B$, meanwhile, we know that $F = 0$ has at most one root in $B$. Thus, in Lemma 7 if we replace the O-M system by S-M system, the lemma is still hold:

Lemma 13. Let $F = (f_1, \ldots, f_n)$ be an S-M system. Let $F' = (f_1, \ldots, f_{n-1})$ and $S = \mathbb{V}(F')$. Assume that $S \cap \partial B = \{p_1, p_2\}$, we have:

1. If $f_n(p_1)f_n(p_2) \leq 0$, $F = 0$ has a unique root in $B$.
2. If $f_n(p_1)f_n(p_2) > 0$, $F = 0$ has no root in $B$.

Therefore, the existence condition for an O-M system can be applied for an S-M system. We will still consider the index set:

$$Ind(B) = \{(i, a_i), (i, b_i) | i = 1, \ldots, n\}.$$ 

For $(i, t) \in Ind(B)$, we also need to solve $F'|_{x_i = t}$ in $B_i$. Compared with the O-M system, the advantages of the S-M system are as follows:

1. By Lemma 12, $F'|_{x_i = t}$ is still S-M system in $B_i$.
2. Let $F'|_{x_i = t} = (f'_1, \ldots, f'_{n-1})$. If $f'_j(v(B_i)) > 0$ or $< 0$ for some $j \in \{1, \ldots, n - 1\}$, then by Lemma 8 we have $f'_j(B_i) > 0$ or $< 0$. Thus $F'|_{x_i = t}$ has no zero in $B_i$, we do not need to solve $F'|_{x_i = t}$.

We give the following example to show how to check the existence.

Example 5. Let $F = (f_1, f_2, f_3) = (x - y + z, y^2 + x + y + 2z, x^2 + yz - 3x - y + z)$ and $B = [-0.1, 0.1] \times [-0.1, 0.1] \times [-0.1, 0.1]$, $B_1, B_2, B_3$ are all $[-0.1, 0.1] \times [-0.1, 0.1]$. One can verify that $F$ is an S-M system in $B$. Next we check the existence condition. Let $F' = (f_1, f_2)$ and other notations are as above, we need to check whether these systems $F'|_{x_i = t_i}$ have a real zero in $B_i$ or not, where $x_1 = x, x_2 = y, x_3 = z, t_i = -0.1, 0.1$. (Some systems can be checked easily, for example, let $g = f_2|_{x=0.1} = y^2 + x + y + 0.2$, we have $g(0.1, 0.1) > 0, g(-0.1, -0.1) > 0$ and $g(-0.1, 0.1) > 0$, i.e., $g(v(B_3)) > 0$.

By Lemma 8 we know that $g(B_3) > 0$, hence $F'|_{x=0.1}$ has no zero in $B_3$ and we do not need to solve the system. Finally we find both the two systems $F'|_{x=0.1}$ and $F'|_{x=-0.1}$ have a unique zero in $B_1$, see the left and right figures in Figure 2. Then we check the condition of Lemma 13 by simple evaluation, we have $f_3|_{x=0.1}(v(B_1)) < 0$ and $f_3|_{x=-0.1}(v(B_1)) > 0$. Therefore we know that $F = 0$ has a unique zero in $B$. 

14
Based on the discussion above, we give an algorithm \textbf{Existence} below.

\textbf{Algorithm 1} \quad A = \text{Existence}(F, B, \epsilon_b) :

1. \textbf{Input:} An S-M system $F = (f_1, \ldots, f_n)$, a box $B$, a given precision $\epsilon_b > 0$ for refinement, where $X = \{x_1, \ldots, x_n\}$ are variables of $F$.

2. \textbf{Output:} 1,0,or \textbf{Unknown}.

1. $F' \leftarrow (f_1, \ldots, f_{n-1})$, $T \leftarrow \text{Ind}(B)$, $num \leftarrow 0$, $S = \{}$

2. While $T \neq \emptyset$ and $num < 2$

   a. Take $(i, t)$ from $T$ and $T = T \setminus \{(i, t)\}$.
   b. $b \leftarrow \text{Existence}(F'|_{x_i=t}, B_i, \epsilon_b)$.
   c. If $b = \text{Unknown}$, return \textbf{Unknown}.
   d. Else if $b = 1$, $num = num + b$, $S \leftarrow S \cup \{(i, t)\}$.

3. If $num = 0$, return 0.
4. Else, $\{(i, t), (j, t')\} \leftarrow S$

   a. $t_1 \leftarrow \text{Sign}(f_n|_{x_i=t}(v(B_i)))$, $t_2 \leftarrow \text{Sign}(f_n|_{x_j=t'}(v(B_j)))$.
   b. While $w(B_i) > \epsilon_b$ and $t_1 = 0$ do
      a. Refine $B_i$ w.r.t. $F'|_{x_i=t}$, and still denote the refined boxes as $B_i$.
      b. $t_1 \leftarrow \text{Sign}(f_n|_{x_i=t}(v(B_i)))$.
   c. While $w(B_j) > \epsilon_b$ and $t_2 = 0$ do
      a. Refine $B_j$ w.r.t. $F'|_{x_j=t'}$, and still denote the refined boxes as $B_j$.
      b. $t_2 \leftarrow \text{Sign}(f_n|_{x_j=t'}(v(B_j)))$.
   d. If $t_1t_2 < 0$, return 1.
   e. Else if $t_1t_2 > 0$, return 0.
   f. Else, return \textbf{Unknown}.

\textbf{Remarks for the algorithm:}
1. $A = 1$ means that the system $F$ has a unique zero in $B$, $A = 0$ means that the system $F$ has no zero in $B$.

2. When the algorithm returns Unknown, it means that we do not find two systems which have a unique zero in their corresponding boxes or can not determine the sign of $f_n|_{x_i=t}(v(B_i)) \cdot f_n|_{x_j=t'}(v(B_j))$. This case happens when the zero is on (or very close to) a $k$-dimensional ($k \leq n-2$) the boundaries of $B$, so that our precision $\epsilon$ can not handle this case. This also explains why there are at most two $(n-1)$-dimensional faces of $B$ intersects $S$ in Step 4. We can avoid this case by changing the length of the box or combining some adjacent boxes with the same output “Unknown” to form a new box. Subdividing the new box and checking the conditions again, one usually succeeds in finding the results. The Existence for bivariate systems are presented in [14].

3. In Steps 3.b, 3.c, refining the boxes $B_i, B_j$ means to refine the root of the systems $F'$ for $x_i = t, x_j = t'$ in the boxes to get smaller boxes. Notice that we know there exists and only exists one unique real zero in the related boxes. We can use interval-Newton method for the refinement. $B_i(B_j)$ or part of it can be set as the original box for the iteration.

4. The correctness and termination of the algorithm is based on theories before.

3.6 Choosing a proper S-M matrix $V$

A better S-M matrix helps us avoid some unnecessary computation. In this subsection, we will discuss how to choose a “better” S-M matrix $V$. As mentioned before, we may meet some bad cases: although both $F'|_{x_i=t}$ and $F'|_{x_j=t'}$ have a unique zero in $B_i$ and $B_j$, we can not determine the signs of $f_n|_{x_i=t}(v(B_i))$ and $f_n|_{x_j=t'}(v(B_j))$. Thus, we need to refine the boxes $B_i$ and $B_j$. Therefore, we want to select some “nice” S-M matrices $V$ to avoid bad cases as much as possible. See the following example first.

**Example 6.** Consider the example in [14]. Let $F = (y - x^2, x - 2y)$, $B = [-0.1, 0.1] \times [-0.1, 0.1]$. Let $V_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and $V_2 = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$. We have $G_1 = V_1J^{-1}(m(B)) \cdot F^T = (-3x^2 + x + y, x^2 + x - y)^T = (g_1^{(1)}, g_2^{(1)})^T$ and $G_2 = V_2J^{-1}(m(B)) \cdot F^T = (-5x^2 + 2x + y, x - 2y)^T = (g_1^{(2)}, g_2^{(2)})^T$. It is easy to check that $G_1$ and $G_2$ are both S-M systems in $B$. Next we consider the existence condition. For the system $G_1$, $g_1^{(1)}|_{x=0.1}$ has a unique zero in $B_1 = [-0.1, 0.1]$, however, $g_2^{(1)}|_{x=0.1}(v(B_1))$ are not all positive(negative), then we need to refine $B_1$, see the left figure in Figure 3. For the system $G_2$, both $g_1^{(2)}|_{y=0.1}$ and $g_2^{(2)}|_{y=-0.1}$ have a unique zero in $B_2 = [-0.1, 0.1]$, then we can get $g_2^{(2)}|_{y=0.1}(v(B_2)) < 0$ and $g_2^{(2)}|_{y=-0.1}(v(B_2)) > 0$ immediately, thus we know the existence without refinement, see the right figure in Figure 3. It means that $V_2$ is “better” than $V_1$.

Give a system $F = (f_1, \ldots, f_n)$ and a box $B$, we assume that $B$ is an isolating box of $F$ (i.e., $B$ contains only one simple zero $p$ of $F$). Let $G = VJ^{-1}(m(B)) \cdot F^T = (g_1, \ldots, g_n)^T$ and $G' = (g_1, \ldots, g_{n-1})$. Our goal is to choose an S-M matrix $V$ such that $S = \forall (G')$ goes through $F_n^a(B)$ and $F_n^{b}(B)$, i.e., both $G'|_{x_n=a}$ and $G'|_{x_n=b}$ have a unique zero in $B_n$, meanwhile, $g_n|_{x_n=a}(v(B_n)) \cdot g_n|_{x_n=b}(v(B_n)) < 0$. Then we immediately know $B$ is an isolating box of $F$. Notice that when $w(B)$ is small, i.e., $m(B)$ is close to the simple
zero \( p = (p_1, p_2, \ldots, p_n) \), we know that \( J_F^{-1}(m(B)) \cdot F_T \approx (x_1 - p_1, \ldots, x_n - p_n)^T \) and \( G \approx V \cdot (x_1 - p_1, \ldots, x_n - p_n)^T \) in \( B \). That is to say \( g_i \) is approximately a hyperplane in \( B \) and \( \nabla g_i(m(B)) \approx V_i \cdot \) for \( i = 1, \ldots, n \).

Based on the analysis above, we assume that \( p = 0 \), \( F = (x_1 + h_1, \ldots, x_n + h_n) \) and \( B = [-1, 1]^n \) is a unit box, where \( h_i \in C^1(B) \) and the Taylor expansion of \( h_i \) at \( p = 0 \) has only terms with degree greater than 1, \( i = 1, \ldots, n \). Then we show how to choose \( V \) such that the system \( G = (g_1, \ldots, g_n)^T = V \cdot F_T \) satisfying the following conditions:

1. \( V \) is an S-M matrix.
2. \((g_1, \ldots, g_n-1)|_{x_n=a_n} \) and \((g_1, \ldots, g_n-1)|_{x_n=b_n} \) have a unique zero in \( F^n_l(B) \) and \( F^n_r(B) \).
3. \( \text{Sign}(g_n(F^n_l(B))) \cdot \text{Sign}(g_n(F^n_r(B))) < 0 \).

For example, \( V_{2d} = \begin{pmatrix} N & 1 \\ 1 & -N \end{pmatrix} \) for \( N = 2 \) or a larger positive integer and \( V_{3d} = \begin{pmatrix} N & 1 & 1 \\ 1 & -N & 1 \\ 1 & 1 & N \end{pmatrix} \) for \( N = 3 \) or a larger positive integer satisfy our conditions, see Figure 4.

![Figure 3](image1.png)

Figure 3: The influence of the different S-M matrixes to the same system.

![Figure 4](image2.png)

Figure 4: Examples for choosing \( V \).

For general cases, 

\[
V_{nd} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{pmatrix},
\]
Let $\alpha_{ii} = (-1)^{i+1}N$ and $|\alpha_{ij}| \leq 1, i \neq j$, by generating $\alpha_{ij}$ randomly, we can find an S-M matrix $V_{nd}$ satisfies the above conditions with high probability, where $N$ can be $n$ or another integer larger than $n$. We can also choose $\alpha_{ii}$ for different sign(s) for more choices. Usually, it is easier to get an S-M matrix for $\partial G / \partial X (B)$ when $N$ is larger. This phenomenon can be obtained by observing the matrix of $\partial G / \partial X (B)$. We can also check whether it is a matrix we want by Definition 5 and we ensure that it is an S-M matrix.

### 3.7 Algorithm

In this subsection, we give the main algorithm for real zero isolation of real nonlinear systems. Our method is a subdivision method, we need the exclusion test which is based on the following famous box predicate $C_0(f, B)$:

$$C_0(f, B) := 0 \notin \Box f(B).$$  \hspace{1cm} (2)

Given a function $f$ and a box $B$, we say $C_0(f, B)$ is true if $0 \notin \Box f(B)$. Based on the box predicate, we can write it as an algorithm $A = \text{Exclusion}(F, B)$. Obviously, if the algorithm $\text{Exclusion}$ returns 1, $F = 0$ must have no root in $B$.

If the given system is a polynomial system, we can isolate the real zeros of bounding polynomials to exclude boxes [14]. We rewrite a multivariate polynomial as below.

$$f_i(x_1, \ldots, x_n) = \sum_{j=0}^{d_i} t_{i,j}(x_1, \ldots, x_{n-1})x_n^j.$$  

Let $B = I_1 \times \cdots \times I_n = B_1 \times I_n$. We split $B_1$ into small boxes with a given length. For each these small $(n-1)$-D box $b$, we evaluate $f_i(x_1, \ldots, x_n)$ on $b$ to get a sleeve polynomial [12].

$$f_i(b, x_n) = \sum_{j=0}^{d_i} t_{i,j}(b)x_n^j = \sum_{j=0}^{d_i} [a_{i,j}, b_{i,j}]x_n^j.$$  

Isolating the real zeros of $f_1(b, x_n)$ in $I_n$ (see [14] for details), we can get a list of intervals, say $J_1, \ldots, J_m$. Continuing to isolate the real zeros of $f_2(b, x_n)$ in $J_1, \ldots, J_m$, we can get a list of intervals, say $J_1', \ldots, J_{m'}$, or an empty set. If we get an empty set, the box $b \times I_n$ can be thrown away. Else, doing so for $f_3, \ldots, f_n$, we can get a list of intervals, say $J_1', \ldots, J_{m''}$ or an empty set. Then we can get candidate boxes $b \times J_k (1 \leq k \leq m''$) or throwing away the box $b \times I_n$. Thus we exclude some sub-boxes of $B$. We can recursively do so on the boxes to exclude some sub-domain of $B$. When $I_n$ is large, this method may be more efficient to compute the possible candidate boxes than the method in [2]. In fact, this method also works for non-polynomial systems. But the way to construct the upper (lower) bounding function is a little more complicated than polynomial case and we need to isolate the real zeros of non-polynomial univariate equations. We denote it as an algorithm $A = \text{Candidate}(F, B)$. Our implementation uses mainly this method for polynomial systems.

Based on the discussion above, we have the main algorithm $\text{Realrootfinding}$. 

Algorithm 2 \textbf{R, SR = Realrootfinding}(F, B_0, \epsilon) :

1. **Input:** A system $F = (f_1, \ldots, f_n)$, an initial box $B_0$, a termination precision $\epsilon > 0$.

2. **Output:** An isolating box set $R$ and a suspected root box set $SR$.

1. $R \leftarrow \emptyset, SR \leftarrow \emptyset$ and $BS \leftarrow \{B_0\}$.
2. While $BS \neq \emptyset$
   
   (1) Take $B \in BS$, $BS \leftarrow BS \setminus \{B\}$, $A \leftarrow \text{Exclusion}(F, B)$.
   
   (2) If $A = 0$,
      
      (a) $G \leftarrow V_{nd} J_{F}^{-1}(m(B)) \cdot F^T$.
      (b) $A' \leftarrow \text{IsSMSys}(G, B)$.
      (c) If $A' = 1$,
         
         • $A'' \leftarrow \text{Existence}(G, B, \epsilon)$.
         • If $A'' = 1$, $R \leftarrow R \cup \{B\}$.
         • If $A'' = \text{Unknown}$, $SR \leftarrow SR \cup \{B\}$.
      (d) Else if $w(B) > \epsilon$,
         
         • Split $B$ into two similar parts and add them into $BS$.
      (e) Else, $SR \leftarrow SR \cup \{B\}$.

3. Return $R, SR$.

The correctness of the algorithm \textbf{Realrootfinding} is guaranteed by Theorem 6 and Lemma 13. The termination of the algorithm is guaranteed by Theorem 11 and the given $\epsilon$.

**Remarks for the Realrootfinding algorithm:**

1. For a system $F$ with only simple zeros, we can always get all the isolating boxes of all the zeros of $F$ in $B_0$ by recursively subdividing those suspected root boxes. Our theories ensure the termination of the algorithm.

2. If $F$ has singular zeros, we can not determine whether a box contains only a singular zero or not. Thus, we give the termination precision $\epsilon > 0$ and repeat subdividing the boxes until the widths of the obtained boxes are less than $\epsilon$. Finally we get some suspected root boxes. Each suspected root box may contain several zeros (counting the multiplicities of the zeros) or no zero.

3. In order to get all the real zeros of a given system in a general given real box, we consider the coordinate transformation: $x_i \rightarrow \frac{1}{x_i}$. We map the interval $[-b, -1]$ to $[-1, -1/b]$, and the interval $[1, b]$ to $[1/b, 1]$, where $b > 1$. Hence, we need only to consider finding real zeros in $[-1, 1]^n$. Doing this way, we take only interval computation inside $[-1, 1]^n$ which need less interval evaluation. If the given system has only finite real zeros in a bounded box, we can get all its real zeros in the whole real space. For example, for a bivariate system $F = (f_1(x, y), f_2(x, y))$, we can get the isolating boxes or suspected boxes of real zeros of the original system in $[1, b] \times [-1, 1]$ by isolating the real zeros of the system $(f_1(\frac{1}{x}, y), f_2(\frac{1}{x}, y))$ in $[1/b, 1] \times [-1, 1]$. If $F$ is a polynomial system, then we can get all the real zeros of $F$ if we take $b$ as its root bound.

4. In Step 2.(2) (d), we can find that some regions in $B$ may be computed for several
times which waste much computing time. Thus we can split the given box $B_0$ into many smaller boxes at first.

5. In Step 2.(1), our aim is to find the candidate regions which may contain real zeros of the given system. For the case that the given system is a polynomial system, we can compute candidate regions as below (see more details in [12, 13, 14]). Given a system $F = (f_1, \ldots, f_n)$ and a box $B = I_1 \times \cdots \times I_n$, let $J = I_2 \times \cdots \times I_n \subset \mathbb{R}^{n-1}$. We denote $T_i$ as the set of the real (interval) zeros of the interval polynomial $f_i(x_1, J)$ for $i = 1, \ldots, n$, and let $\{t_1 \cap \cdots \cap t_n | t_i \in T_i, i = 1, \ldots, n\} = \{I'_1, \ldots, I'_m\}$. We call $\{I'_1 \times J, \ldots, I'_m \times J\}$ the candidates of $F$ in $B$. It is obvious that all the real zeros of $F$ in $B$ are in the candidates. If the width $w(J)$ is large, we can split $J$ into $(n-1)^k$ parts equally (by splitting $I_i$, $i = 2, \ldots, n$ into $k$ parts equally) and compute the candidates separately.

6. For each suspect box we got, it may contain no zero, a simple zero, a multiple zero, several simple zeros or, one or more simple zero(s) together with one or more multiple zero(s) of the given system. This happens because we set a termination precision for the subdivision process since we are not sure if the given system has multiple real zeros or not. For some examples, there may be so many suspect boxes and most of them contain no roots. We need to remove the redundant boxes which contains no roots. For a given system, there may exist one or several cluster(s) of boxes. Each cluster of boxes may contain one (or several) multiple (or simple) root(s), or no roots. If there is a root, the Newton’s method will converge to the root if the start point is chosen well, that is, the start point is in the basins of attraction of the system for the root [52]. The convergent region for the root will intersect some the boxes inside the cluster of the boxes. If we choose properly some point(s) in each suspect box in the cluster of boxes as start point(s) for Newton’s method for the system, we may get the root(s) inside the cluster of boxes and remove the redundant boxes. For the derived box(es) after computing with Newton’s method, we can do only a heuristic verification of a suspect box by deflation methods (see [11, 21, 35] and the methods mentioned therein). Notice that we may miss some root(s) or get more roots with this operation. For example, a root on (or very close to) the boundaries of a suspected box, the root may be missed or counted twice because of numerical computation. But it usually works well. We will show experiments for illustration with this step.

3.8 Complexity analysis

We analyze the bit complexity of isolating the real roots of a zero-dimensional polynomial system $\Sigma$ in this subsection. We assume that there are $n$ variables and the degrees of the polynomials of the system are bounded by $d$, the number of their terms are bounded by $m$ and the bit sizes of their coefficients are bounded by $\tau$.

The complexity analysis for subdivision based algorithm of a single polynomial was considered by [9, 17]. The complexity analysis for subdivision based algorithm for a polynomial system was given in [38]. Different from their work to find exact results, our analysis is for a given terminating precision without assuming that the system has only simple real roots.

We would like to mention that the condition number is an important parameter for the complexity analysis of subdivision based methods for root finding. There are a series of works about real root counting of polynomial systems with probabilistic numeric methods and related analysis based on condition number [18, 19, 20].

In this paper, $O$ means the bit complexity, $\tilde{O}(\cdot)$ indicates that we omit poly-logarithmic factors.

Consider the real roots of $\Sigma$ inside $B = [-1, 1]^n$. Notice that if we want to get all the
real zeros of $\Sigma$, we can transform the original system into new system by replacing $x_i$ with $1/x_i$ and removing the denominators of the whole polynomials. We can get $2^n$ this kind of systems in $[-1,1]^n$. We assume that the termination precision for the boxes is $\rho$, that is, we stop subdividing the boxes when their lengths are less than $\rho$, where $\rho$ is a rational number such that $0 < \rho < 1$. So the number of the boxes in $B$ is bounded by $(\frac{2}{\rho})^n$. The bitsize of the endpoints of the boxes is bounded by $-\log(\rho)$.

For each box, we take one exclusion test and one our existence test at most. We will analyse these two operations one by one.

**Lemma 14** (Multivariate polynomial interval evaluation). Let $g \in \mathbb{Z}[x_1, \ldots, x_n]$ be of magnitude $(d, \tau)$ and $m$ terms. $I_1, \ldots, I_n$ are intervals whose endpoints are rational numbers with bitsize $\sigma$, then evaluating $g(I_1, \ldots, I_n)$ has a bit complexity of $\tilde{O}(dm\sigma + m\tau)$, and the bitsize of the endpoints of $g(I_1, \ldots, I_n)$ is $O(d\sigma + \tau)$.

**Proof.** Consider one term of $g$ at first. The operations here include the multiplications of $d$ intervals at most and one multiplication between the coefficient and the product of the intervals. We can divide $d$ intervals into $\frac{d}{2}$ pairs. Each pair usually contains two intervals. If $d$ is odd, the last pair contains only one interval. To get the product of them includes at most 4 multiplications. The total bit complexity for computing all the pairs is $\frac{d}{2} + 4\sigma = 2d\sigma$. The bitsizes of the products are $2\sigma$. For these $\frac{d}{2}$ products, we divide them into $\frac{d}{4}$ pairs. Similarly, the total bit complexity for computing the products of all the pairs is $\frac{d}{4} + 4\sigma = 2d\sigma$. The bitsizes of the products are $4\sigma$. Doing so in a similar way until the $k$ step such that $d = 2^k$, that is, $k = \log(d)$, we can get the product of all the intervals. So to get the product of all the intervals, we have the total bit complexity $2d\sigma \cdot \log(d)$ and the bitsize of the final product is $2^\log(d)\sigma = d\sigma$. Considering the coefficient into the product, we have the total bit complexity of evaluating one term is $2d\sigma \cdot \log(d) + d\sigma + \tau = \tilde{O}(d\sigma + \tau)$. The bitsize of the final product for one term is $d\sigma + \tau$. So evaluating $g(I_1, \ldots, I_n)$ has a complexity of $\tilde{O}(dm\sigma + m\tau)$, and the bitsize of the endpoints of $g(I_1, \ldots, I_n)$ is $O(d\sigma + \tau)$.  

**Lemma 15.** [51] Let $A = (a_{i,j}) \in \mathbb{Z}^{n \times n}$ be nonsingular. We denote by $\|A\| := \max |a_{i,j}|$ the maximum magnitude of entries in $A$, and by $\kappa(A) := \|A\||A^{-1}\|$ the condition number of the matrix with respect to the max norm. We give a Las Vegas probabilistic algorithm that has expected running time $\tilde{O}(n^3(\log\|A\| + \log\kappa(A)))$ bit operations to compute the exact inverse of $A$. Thus, for a well conditioned $A$, with $\kappa(A)$ bounded by a polynomial function of $n\log\|A\|$, this cost estimate becomes $\tilde{O}(n^3\log\|A\|)$.

**Lemma 16.** We can compute the least common multiple of $n$ integers with bitsize $\sigma$ by $\tilde{O}(n\sigma)$ bit complexity and the bit size of the least common multiple is $n\sigma$.

**Proof.** We can divide $n$ intervals into $\frac{n}{2}$ pairs. Each pair is two integers. Note that if $n$ is odd, the last one can be regarded as a pair. For each pair, we compute the least common multiple of the two integers $a, b$. It is $a/b \gcd(a, b)$. The bit complexity is $2\tilde{O}(\sigma)$. So the bit complexity of computing all the pairs is $n\tilde{O}(\sigma)$. The bit sizes of the results are all $2\sigma$. We continue to divide the results into $\frac{n}{4}$ pairs. For each pair we compute its least common multiple with bit complexity $4\tilde{O}(\sigma)$. So the bit complexity of computing all the pairs is $n\tilde{O}(\sigma)$. And the bit sizes of the results are $4\sigma$. Doing so, until we get the least common multiple of the $n$ integers, which we need $\log(n)$ steps. So the total bit complexity is $\log(n)\tilde{O}(n\sigma) = \tilde{O}(n\sigma)$. The bit size of the last least common multiple is $2^{\log(n)}\sigma = n\sigma$.  

21
For a nonsingular matrix $M$ with rational entries such that the bit sizes of the entries are bounded by $\sigma$, we can rewrite $M = M_1M_2$, where $M_1$ is a diagonal matrix and $M_2$ is a matrix with integer entries. We can compute the least common multiple of the denominators of the elements of each row. Set its inverse as the element of the related row of $M_1$. Each of the related element of $M$ of the row multiplies the least common multiple and set them as the related elements of $M_2$. The bit size of each element of $M_2$ is $n\sigma$, so as $M_1$. By Lemmas 15, 16, we can find that the bit complexity of computing the inverse of $M$ is $\tilde{O}(n^4\sigma + n^3\log(\kappa(\|M_2\|)))$ (a well conditioned one is $\tilde{O}(n^4\sigma)$) and the bit sizes of the elements of $M^{-1} = M_2^{-1}M_1^{-1}$ is $2n\sigma$.

Now we consider the complexity to check the existence and the uniqueness of a real root inside a box. When we compute the intersection between the space curve formed by the $n - 1$ functions and the boundaries of a box. We need to check each face of the box to intersect the space curve. There are $2n$ faces. Each face is related to a zero-dimensional system with $n - 1$ functions and $n - 1$ variables. Recursively, we will do root finding of $2^{n-1}n!$ univariate polynomials. In order to get an approximating root, we can bisect the interval a fixed number times, say 10 times, if there is a root. We also need to multiply two square matrices with order $n$: One is $V$, the other is $J_F^{-1}(m(B))$.

**Lemma 17.** Let $F = (f_1, \ldots, f_n) \subset \mathbb{Z}[x_1, \ldots, x_n]$ and each $f_i$ be of magnitude $(d, \tau)$, and $m$ terms. If $B = I_1 \times \cdots \times I_m$ is a box with rational entries such that the endpoints of $I_1, \ldots, I_m$ all with bit size $\sigma$, then checking whether $F$ contains a unique real root in $B$ has a bit complexity of $\tilde{O}(2^n n^{n+1} m (d\sigma + \tau))$.

**Proof.** Denote the Jacobian matrix of $F$ w.r.t. $x_1, \ldots, x_n$ as $J_F$. The bit size of the middle point of $B$, say $\mathbf{P}$, is $\sigma$. It is clear that the bit size of each element of $J_F(\mathbf{P})$ is $\mathcal{O}(d\sigma + \tau)$. The bit complexity to compute $J_F(\mathbf{P})$ is $\mathcal{O}(n^2 m \sigma + n^2 m \tau)$ by Lemma 14. Thus the bit size of the elements of the inverse of $J_F^{-1}(\mathbf{P})$ is $\mathcal{O}(n d\sigma + n \tau)$ and the bit complexity to compute $J_F^{-1}(\mathbf{P})$ is $\mathcal{O}(n^3 \sigma + n^4 \tau)$ by the analysis below Lemma 16. The bit complexity to compute $V J_F^{-1}(\mathbf{P})$ is $n^3 \mathcal{O}(n d\sigma + n \tau) = \mathcal{O}(n^4 d\sigma + n^4 \tau)$, where $V$ is the S-M matrix we mentioned before whose elements are with $\mathcal{O}(1)$ and the bit size of the elements of $V J_F^{-1}(\mathbf{P})$ is $\mathcal{O}(n d\sigma + n \tau)$.

Let $G = (g_1, \ldots, g_n) = V J_F^{-1}(\mathbf{P})F^T$. Then $J_G = V J_F^{-1}(\mathbf{P})J_F$. After we evaluate $J_F$ on $B$, denoted as $J_F(B)$, we have that each element of the matrix has a bit size of $\mathcal{O}(d\sigma + \tau)$ by Lemma 14. Notice that $f_i$ and $\frac{\partial f_i}{\partial x_j}$ have the same bit size after evaluating on $B$. In order to check whether the matrix $J_G(B) = V J_F^{-1}(\mathbf{P})J_F(B)$ is strong monotonous over $B$, we compute the minors of the determinant of $J_G(B)$ step by step. We compute the order $i + 1$ minors with the result of the order $i$ minors until we get the determinant of $J_G(B)$, where changes from 1 to $n - 1$. Totally, we can consider computing $n!$ products among $n$ intervals. From the way we check the strong monotonous matrix condition, we compute each product by multiplying the $n$ intervals one by one. There are $4(n - 1)$ multiplications. Notice that the bit size of $J_G(B)$ is $\mathcal{O}(n d\sigma + n \tau)$. The total complexity to get one product is

$$4 \cdot \mathcal{O}(n d\sigma + n \tau) + 4 \cdot 2 \mathcal{O}(n d\sigma + n \tau) + \ldots + 4 \cdot (n - 1) \mathcal{O}(n d\sigma + n \tau) = 2 \cdot n \cdot (n - 1) \mathcal{O}(n d\sigma + n \tau) = \mathcal{O}(n^3 d\sigma + n^3 \tau).$$

By Stirling’s approximation, the total bit complexity to check the strong monotonous condition is $\mathcal{O}(n^6 d\sigma + n^3 \tau) = \mathcal{O}(n^{n+3} d\sigma + n^{n+3} \tau)$. Notice that the uniqueness condition is already checked by the determinant of $J_G(B)$. To check the number of intersections between the space curve formed by $g_1, \ldots, g_{n-1}$ and the faces of $B$, we need to check whether $2^n n!$ univariate polynomials with magnitude $(d, n d\sigma + n \tau)$ and at most $nm$ terms have solutions in the related intervals. So for one root isolation on one interval, the bit complexity
is $O(nm(d\sigma + nd\sigma + n\tau)) = O(nm(d\sigma + m\tau))$ by Lemma 14. So the total bit complexity for the root isolation is $2^n n! O(nm d\sigma + n m\tau) = O(2^n n^{n+1} m (d\sigma + \tau))$. It is not difficult to find that this is the main part for the whole bit complexity when compared all the steps. Thus the total bit complexity of checking whether $F$ contains a unique real root in $B$ is $O(2^n n^{n+1} m (d\sigma + \tau))$.

We want to mention that there exist more efficient algorithm to compute the determinant of an interval matrix, see [30]. But it does not change the total bit complexity. The analysis is based on our implementation.

From the analysis above, we can directly deduce the following result.

**Theorem 18.** Isolating the real roots of a square system of polynomials with magnitude $(d, \tau)$ and $m$ terms with the algorithm mentioned above, one takes a bit complexity of $O((\frac{8}{\rho})^n n^{n+1} m (-d \log(\rho) + \tau))$.

### 4 Experiments

We implement our algorithm in C++, and we do some experiments under Linux with a computer of 64 AMD 3990X 2.90GHz CPU and 64 GB RAM. We also realize the parallel computation with MPI. Our code can be downloaded from [http://mmrc.iss.ac.cn/~jcheng/pai/RootFinding.tar.gz](http://mmrc.iss.ac.cn/~jcheng/pai/RootFinding.tar.gz) as well as a simple user guider from [http://mmrc.iss.ac.cn/~jcheng/pai/Pai User Guide.pdf](http://mmrc.iss.ac.cn/~jcheng/pai/Pai User Guide.pdf). We compare our algorithm with MK test [25, 33, 42] and Bertini [3]. There are two other famous homotopy continuation softwares: PHCpack [54] and Hom4ps [10]. We choose Bertini to compare just because it derives real roots directly.

For systems with small sizes, we check their roots with symbolic methods such as [13] and we will point out if the results of some methods are not correct.

We use NiDj to represent the systems formed by polynomials with $i$ variables and degree $j$ in the tables. NiDjE means we scaling back the coordinates of the variables. For example, the polynomials in N2D5E case is got by substituting variables $(x_1, x_2)$ with $(100 * x_1, 100 * x_2)$, and then we can just compute the roots in $[-1, 1]^2$ to get the roots in $[-100, 100]^2$ of the original system, which is more efficient in some cases. In the tables, Case means the type of systems. Terms means the number of terms of each polynomial in the system. Coef means the maximal absolute value of the coefficients of the polynomials in the systems. Range means the box to search the real roots for our method and MK test. But Bertini finds all the complex roots of the given systems, including the real roots out of the given box. Roots means the number of real roots of the systems. For our method, there are three parameters: The first one is the number of certified real roots of the system inside the box; The second one is the number of the suspected boxes of the system inside the box; The last one is the number of possible real roots of the system in all the suspected boxes with the method mentioned in Remark 6 of Algorithm 2. MK contains only the first two parameters for their roots. Bertini shows only the real roots among all its complex roots, including the real roots out of the box. Times means the computing times for the related examples in seconds. We take 5 systems for each case to get their average computing time. “~” means we do not compute the examples. For Bertini, it means the example is out of its computing ability or the computing time is larger than 10000 seconds or their cases are NiDjE. For MK, there are no certified real roots, thus we do not compute the examples. In the examples, we usually set the precision $\rho = 10^{-6}$. See the data in Table 1.
Table 1: Comparing our method with MK and Bertini for polynomial systems.

Remarks for Table 1:

1. In case N2D51, one example has 7 roots and we get 5 of them and miss finding the other 2 from the suspected boxes. In the related case N2D51E, we directly find 7 certified roots for the example but another example miss one root from the suspected boxes.

2. In case N3D9, the number of the total average roots is 13.4. Bertini misses one root, one of whose coordinate is around 3,000,000. Our code finds all the real roots of the systems inside the box $[-100, 100]^3$. There are totally 3 roots out of the box.

3. In case N3D101, the computing times for 3 of the 5 systems are less than 100 seconds. But one example takes 20157 seconds and there are 11 certified solutions and no suspected boxes. We find that several roots are very close to each other. One example has 17 certified roots and 162 suspected boxes, which takes 112.398 seconds.

4. In Table 1, the examples above N3D51 does not use parallel computing and the results are proved by symbolic computation. From N3D51, we use MPI parallel computing with 30 cores and the results are without proof with symbolic computation.

Table 2: The influence of the number of terms of the input polynomials to our method. The first system $[g_1, g_2, g_3]$, $g_i =$

| System $[g_1, g_2, g_3]$, $g_i =$ | Terms | Times(second) | Roots |
|-----------------------------------|-------|---------------|-------|
| $f_i$                             | 5     | 1.536         | 4.2/0/0 |
| $f_i + (x_1^2+1)$                 | 10    | 3.964         | 4.2/1.6/0 |
| $f_i + (x_1^2 + x_2^2 + x_3^2 + 1)$ | 20    | 8.470         | 4.2/1.6/0 |
| $f_i + ((x_1 + x_2)^2 + (x_1 + x_3)^2 + 1))$ | 30    | 355.113       | 4.2/68.8/0 |
| $f_i + ((x_1 + x_2 + 1)^2 + (x_1 + x_3 - 1)^2 + 1)$ | 40    | 326.391       | 4.2/166.6/0 |
| $f_i + ((x_1 + x_2 + x_3 + 1)^2 + 1)$ | 50    | 4229.643      | 4.2/4882.8/0 |

Table 2: The influence of the number of terms of the input polynomials to our method. The first system $[f_1, f_2, f_3]$ is with randomly generated polynomials of degree 11 and 5 terms. All the systems are expanded before computing. We consider the roots inside $[-1, 1]^3$. 

24
In order to analyse the influence of the number of the terms to our method, we design the examples in Table 2. The systems all have the same solutions but the number of the terms of the polynomials are different. We can find that our method is sensitive to the number of terms of the polynomials in the systems. When the terms of the polynomials increase, the computing times increase and the suspected roots increase. This claim matches the complexity analysis in Lemma 17.

The examples in Table 3 shows the influence of the bit size of the coefficients of the input polynomials to our method. Increasing the bit size of the coefficients does not influence a lot to our method or to MK method but does influence a lot to Bertini since the trick step in Bertini is sensitive to the bit size of the coefficients. This phenomenon matches the bit complexity analysis of our algorithm, see Theorem 18. We can also find that Bertini can not work for systems with large coefficients.

| System \([g_1, g_2, g_3]\), \(g_i = \ldots\) | Times(second) | Roots |
|-----------------|--------------|-------|
|                 | Ours | MK | Bertini | Ours | MK | Bertini |
| \(f_i\)         | 47.093 | 49.262 | 0.213 | 4.8/138.2/0 | 0/262.4 | 5.0 |
| \(f_i \ast 2^{10}\) | 45.397 | 48.985 | 0.636 | 4.8/138.2/0 | 0/262.4 | 5.0 |
| \(f_i \ast 2^{50}\) | 45.362 | 49.254 | \(-\) | 4.8/138.2/0 | 0/262.4 | \(-\) |
| \(f_i \ast 2^{100}\) | 45.464 | 49.302 | \(-\) | 4.8/138.2/0 | 0/262.4 | \(-\) |
| \(f_i \ast 2^{200}\) | 45.484 | 49.009 | \(-\) | 4.8/138.2/0 | 0/262.4 | \(-\) |

Table 3: The influence of the bitsize of the coefficients of the input polynomials to the methods. The first system \([f_1, f_2, f_3]\) is with randomly generated polynomials of degree 5 and 10 terms. The other systems are formed as shown in the table. The systems are expanded before computing. We consider the roots inside \([-100, 100]^3\). There is 1 root, one of whose coordinate is out of \([-100, 100]\) that is why the number of our roots is 4.8 but that of Bertini is 5.0. Bertini cannot work for systems with large coefficients.

We also check the influence of the number of real roots inside a box to our method, see Table 4. If a system has more real roots inside a box, then our methods will take more times. It is reasonable since more roots mean that there are more boxes need to do the existence checking which is time-consuming. The number of real roots of a system almost does not influence the computing times of Bertini since the number of the total complex roots is unchanged under the situation.

We also check the systems with multiple zeros, see Table 5. Usually, systems with multiple real zeros will take more computing time since near the multiple zeros there exist many suspected boxes. To exclude the ones without real roots with the method mentioned in Remark 6 of Algorithm 2 is time-consuming. And in Case multiN3D12, when excluding suspected boxes, one root is counted twice since it is very close to the boundaries of two suspected boxes. Some multiple roots of the systems may be computed as several roots by Bertini, so the numbers of roots in Cases multiN2D6 and multiN2D12 are not exactly the numbers of the exact roots of the systems.

We test the five systems N2D9 in Table 1 with our method for different precisions, see Table 6. We can find that the higher precision we use, the less suspected boxes we get and the more computing times we need.

We check our implementation of parallel computing with MPI on N4D9 in Table 1. We use 4, 8, 16, 32, 60 cores to compute the 5 systems and get the average computing time. The related data is given in Table 7. It shows the speedup of the parallel computing of our
Table 4: The influence of the number of roots of the input polynomials to the methods. The systems are \( [f - a_i, g - b_i, h - c_i] \) such that they have 8, 16, 24, 32, 40, 48 real roots respectively, where \([a_i, b_i, c_i]\) are \([62, 61, 63], [5, 61, 61], [32, 31, 37], [22, 21, 17], [2, 63, 7], [10, 10, 10]\) respectively and \(f, g, h\) are the expansions of \((25 x^2 - 2) \ (25 y^2 - 11) \ (25 z^2 - 5), (25 x^2 - 11) \ (25 y^2 - 5) \ (25 z^2 - 3), (25 x^2 - 5) \ (25 y^2 - 2) \ (25 z^2 - 11)\) respectively. All the roots are inside \([-1, 1]^3\).

Table 5: We test systems with multiple real zeros with our method and Bertini. For cases multiN2D6 and multiN2D12, we first randomly generate two polynomials \(f\) and \(g\) with 4 terms, then the polynomial system \(\{f_1, f_2\}\) is got by: \(f_1 = f \ast g\) and \(f_2 = \frac{\partial f_1}{\partial x_2}\). For cases multiN3D6 and multiN3D12, we first randomly generate two polynomials \(f\) and \(g\) with 4 terms, then the polynomial system \(\{f_1, f_2, f_3\}\) is got by: \(f_1 = f \ast g, f_2 = \frac{\partial f_1}{\partial x_1}\) and \(f_3\) is a randomly generated polynomial.

Table 6: The influence of the termination precisions to the 5 systems in N2D9 in Table 1.
From Table 1, we can find that our method is sensitive to the number of variables and the degree of the polynomials also influence the computing times, this suits for the complexity analysis in Theorem 18. For some examples whose roots are distributed in a very bad position, our method may take much time.

Compared to the three related methods, our method is complete for square nonlinear systems with only simple roots, as the MK method. But the interval Newton’s method and the α-theory method are not complete. Since the exclusion test of MK method and our method are the same, the computing times of two methods are very close if we do not exclude the suspected roots. But we have more certified roots. The existence checking method of MK seldom works for systems with more than two variables. But our method works for systems with 6 variables as shown in Table 1. The reason is that Miranda theorem works for evaluating the functions of the systems directly on some of the boundaries of the boxes to require that the result interval does not contain zero. For certifying a simple root, the MK method, our method and the interval Newton’s method need only to compute the evaluations of the functions in the system and their order-one derivatives. The α-theory method needs compute higher order derivatives of each function in the system. The existence criteria of the interval Newton method, the α-theory method and our method, all need the information of order-one derivatives of the functions. But the Miranda based method needs only the evaluation of the functions. When the box which contains a zero of the system becomes small, the functions are very close to zero. When evaluated on the boundaries of the box, which are also a box (or an interval), the intervals derived from the functions contain zero with high probability, especially for functions with more than two variables. The deeper subdivision does not change the situation and even makes it worse in practice because of the interval calculus. This is the reason why the Miranda based method works only for less variables systems with lower degrees. Our experiments support the claim. Our method avoids using functions directly but using order-one derivatives of the functions for the existence of a zero. It works for systems with more variables and high degrees.

Compared to Bertini, our method works well for polynomial systems with larger Bézout bound, higher degrees and less variables. Bertini works well for systems with many variables but not so larger Bézout bound. Our method can find roots of non-polynomial systems but Bertini can not. There are two other advantages of our method: One is that it can find roots of a system in a fixed local region. The other is that each of our certified root box contains exactly one real root of the system. Currently, our method can work only for systems with only several variables. To overcome this shortcoming is our future work.

We will use our code to solve two problems in applications.

Example 7. The equations of this example is from robotics and describes the inverse kinematics of an elbow manipulator. One can find the problem in [28, 29]. We solve directly the original one without transforming it to an algebraic system. Thus we have only 6 variables

| Number of cores | 4   | 8   | 16  | 32  | 60  |
|-----------------|-----|-----|-----|-----|-----|
| Computing times | 1299.0 | 803.2 | 533.4 | 364.8 | 250.8 |

Table 7: The influence of the number of cores of parallel computing with our method for N4D9 in Table 1.
but there are 12 variables in \([28, 29]\). The following is the system given in \([28, 29]\).
\[
\begin{align*}
&\begin{array}{ll}
&s2c5s6 - s3c5s6 - s4c5s6 + c2c6 + c3c6 + c4c6 - A077, \\
&c1c2s5 + c1c3s5 + c1c4s5 + s1c5 - 1.9115, \\
&s2s5 + s3s5 + s4s5 - 1.9791, \\
&3c1c2 + 2c1c3 + c1c4 - 4.0616, \\
&3s1c2 + 2s1c3 + s1c4 - 1.7172, \\
&s2 + 2s3 + s4 - 3.9701, \\
&s_i^2 + c_i^2 - 1, i = 1, \ldots, 6.
\end{array}
\end{align*}
\]

We replace \(s_i, c_i\) with \(\sin(6.3x_i), \cos(6.3x_i)\) for \(i = 1, \ldots, 6\) in \([23]\). Thus we can consider \(x_i\) in \([0, 1]\) such that \(6.3x_i\) covers \([0, 2\pi]\). Doing so, we get all the solutions of the system.
Computing the real roots of the new system in \([0, 1]\) with precision \(\rho = 10^{-3}\) and 32 cores with MPI parallel computing, we get 106 real roots:

Furthermore, we get 4116 suspected boxes and find none roots from them. We miss 6 real roots.

If we set \(\rho = 10^{-6}\), we get exactly 16 certified roots and no suspected boxes which takes 2133.05 seconds:

Example 8. The following problem is the inverse position problem for a six-revolute-joint problem in mechanics, one can find it in \([28, 43]\). The defining equations in the reference are as below.
\[
\begin{align*}
&\begin{array}{l}
&\begin{array}{l}
a_{i3}x_2x_3 + a_{i4}x_2x_4 + a_{i4}x_1x_3 + a_{i2}x_1x_4 + a_{i5}x_5x_7 + a_{i6}x_5x_8 + a_{i7}x_6x_7 + a_{i8}x_6x_8
\end{array}, \\
\end{array}
\end{align*}
\]

There are 8 equations and 8 variables. The values of \(a_{ij}, j = 1, \ldots, 17\) can be found in \([28]\). We replace \(x_i, x_{i+1}\) with \(\sin(6.3y_{i+1}), \cos(6.3y_{i+1})\) for \(i = 1, 3, 5, 7\) in the first equation.
above. Then we can get a system with 4 equations and 4 variables. Computing its real roots
in $[0, 1]^4$ with precision $\rho = 10^{-3}$ and 32 cores with MPI parallel computing, we get 9 certified
roots:

\[
\begin{align*}
&[0.199219, 0.203125], [0.195312, 0.199219], [0.703125, 0.707031], [0.167969, 0.171875], \\
&[0.17781, 0.177734], [0.226516, 0.230469], [0.375, 0.376953], [0.609375, 0.611328], \\
&[0.189453, 0.191406], [0.871094, 0.873047], [0.0703125, 0.0722656], [0.496094, 0.498047], \\
&[0.917969, 0.919922], [0.152344, 0.154297], [0.355469, 0.357422], [0.630859, 0.632812], \\
&[0.902344, 0.904297], [0.0566406, 0.0585938], [0.662109, 0.664062], [0.179688, 0.181641], \\
&[0.859375, 0.861328], [0.345703, 0.347656], [0.748047, 0.75], [0.148438, 0.150391], \\
&[0.718891, 0.718444], [0.726516, 0.730469], [0.746094, 0.748047], [0.257812, 0.259766], \\
&[0.0722656, 0.0732422], [0.199219, 0.203125], [0.969938, 0.961914], [0.0576172, 0.0585938], [0.513672, 0.514648].
\end{align*}
\]

and 2143 suspected boxes. We did not find roots from the suspected boxes. It takes 322.695
seconds.

If we set $\rho = 10^{-6}$, we get exactly 12 certified roots and no suspected boxes which takes
902.642 seconds:

\[
\begin{align*}
&[0.199219, 0.203125], [0.195312, 0.199219], [0.703125, 0.707031], [0.167969, 0.171875], \\
&[0.17781, 0.177734], [0.226516, 0.230469], [0.375, 0.376953], [0.609375, 0.611328], \\
&[0.189453, 0.191406], [0.871094, 0.873047], [0.0703125, 0.0722656], [0.496094, 0.498047], \\
&[0.917969, 0.919922], [0.152344, 0.154297], [0.355469, 0.357422], [0.630859, 0.632812], \\
&[0.902344, 0.904297], [0.0566406, 0.0585938], [0.662109, 0.664062], [0.179688, 0.181641], \\
&[0.859375, 0.861328], [0.345703, 0.347656], [0.748047, 0.75], [0.148438, 0.150391], \\
&[0.718891, 0.718444], [0.726516, 0.730469], [0.746094, 0.748047], [0.257812, 0.259766], \\
&[0.0722656, 0.0732422], [0.199219, 0.203125], [0.969938, 0.961914], [0.0576172, 0.0585938], [0.513672, 0.514648].
\end{align*}
\]

This two examples show that our method works for non-polynomial systems and gives
certified solutions.

5 Conclusion

In this paper, we propose a numerical method to isolate real zeros of a zero-dimensional
multivariate square nonlinear system. We present the concept of the O-M system and the
S-M system for a multivariate nonlinear system in an n-D box. Based on that, a new
existence criterion of a real zero of a system inside a box which is different from Miranda
based method is presented, it is much easier to be satisfied than Miranda based method.
The uniqueness of a real zero of a system inside a box presented in the paper is related to
the existence condition and it contains the traditional Jacobian test. For nonlinear systems,
we use the exclusion test to get candidate boxes and for polynomial systems, we can use
the bounding polynomials to get the candidate boxes. Then we check the uniqueness and
existence conditions for each candidate box. If it succeeds, we get an isolating box of the
system. If not, we split these candidate boxes until they satisfy the conditions or their
widths reach a given precision. Our method is complete for systems with only finite simple
real roots inside a box. We implemented the presented algorithms which shows it works
well. The shortcoming of our current method is that we can not solving systems with so
many variables. In the future, we will overcome this shortcoming and consider real zero
isolating of high dimensional systems.
References

[1] O. Aberth. *Introduction to precise numerical methods*. Elsevier/Academic Press, Amsterdam, 2nd edition, 2007.

[2] D.S. Arnon, G.E. Collins, and S. McCallum. Cylindrical algebraic decomposition i: The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, 1984.

[3] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. *Numerically solving polynomial systems with Bertini*. Software, Environments, and Tools 25, SIAM, 2013.

[4] C. Beltran and A. Leykin. Certified numerical homotopy tracking. *Experimental Mathematics*, 21(1):69–83, 2012.

[5] C. Beltran and A. Leykin. Robust certified numerical homotopy tracking. *Found. Comput. Math.*, 13:253–295, 2013.

[6] B. Buchberger. Bruno buchberger’s phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3-4):475–511, 2006.

[7] M. Burr, S. W. Choi, B. Galehouse, and C. Yap. Complete subdivision algorithms, ii: Isotopic meshing of singular algebraic curves. *Journal of Symbolic Computation*, 47(2):131–152, 2012.

[8] M. Burr, S. Gao, and E. Tsigaridas. The complexity of an adaptive subdivision method for approximating real curves. In *Proceedings of ISSAC’2017*, pages 61–68. ACM, 2017.

[9] M. A. Burr, S. Gao, and E. P. Tsigaridas. The complexity of an adaptive subdivision method for approximating real curves. In *ISSAC ’17*, pages 61–68. ACM, 2017.

[10] T. Chen, T.-L. Lee, and T.-Y. Li. Hom4ps-3: A parallel numerical solver for systems of polynomial equations based on polyhedral homotopy continuation methods. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 183–190, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[11] J. S. Cheng, X. Dou, and J. Wen. A new deflation method for verifying the isolated singular zeros of polynomial systems. *Journal of Computational and Applied Mathematics*, 376:112825, 2020.

[12] J. S. Cheng, X. S. Gao, and C. K. Yap. Complete numerical isolation of real roots in zero-dimensional triangular systems. *Journal of Symbolic Computation*, 44(7):768–785, 2009.

[13] J. S. Cheng and K. Jin. A generic position based method for real root isolation of zero-dimensional polynomial systems. *Journal of Symbolic Computation*, 68:204–224, 2015.

[14] J. S. Cheng and J. Wen. Certified numerical real root isolation for bivariate polynomial systems. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*, ISSAC ’19, pages 90–97, New York, NY, USA, 2019. ACM.
[15] J.-S. Cheng, J. Wen, and B. Zhang. Certified numerical real root isolation for bivariate nonlinear systems. *Journal of Symbolic Computation*, 114:149–171, 2023.

[16] D. Cox, J. Little, and D. O’Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, 1996.

[17] F. Cucker, E. Alperen A., and J. Tonelli-Cueto. Plantinga-vegter algorithm takes average polynomial time. Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC, pages 114–121. ACM, July 2019.

[18] F. Cucker, T. Krick, G. Malajovich, and M. Wschebor. A numerical algorithm for zero counting, i: Complexity and accuracy. *Journal of Complexity*, 24(5):582–605, 2008.

[19] F. Cucker, T. Krick, G. Malajovich, and M. Wschebor. A numerical algorithm for zero counting, ii: Distance to ill-posedness and smoothed analysis. *J. Fixed Point Theory Appl.*, 6:285–294, 2009.

[20] F. Cucker, T. Krick, G. Malajovich, and M. Wschebor. A numerical algorithm for zero counting. iii: Randomization and condition. *Advances in Applied Mathematics*, 48(1):215–248, 2012.

[21] B. H. Dayton, T. Y. Li, and Z. Zeng. Multiple zeros of nonlinear systems. *Mathematics of Computation*, 80(276):2143–2168, 2011.

[22] J. Van der Hoeven. Reliable homotopy continuation. *hal-00589948v3*, 2011.

[23] JR. C.H. Edwards. *Advanced Calculus of Several Variables*. Academic Press, 1973.

[24] J. Garloff and A.P. Smith. Investigation of a subdivision based algorithm for solving systems of polynomial equations. *Nonlinear Analysis: Theory, Methods and Applications*, 47(1):167 – 178, 2001.

[25] J. Garloff and A.P. Smith. Solution of systems of polynomial equations by using bernstein expansion. In *Symbolic Algebraic Methods and Verification Methods*, pages 87–97. Springer, 2001.

[26] I. Gelfand, M. Kapranov, and A. Zelevinsky. *Discriminants, resultants, and multidimensional determinants*. Springer Science & Business Media, 2008.

[27] Iddo Hanniel and Gershon Elber. Subdivision termination criteria in subdivision multivariate solvers using dual hyperplanes representations. *Computer-Aided Design*, 39(5):369–378, 2007. Geometric Modeling and Processing 2006.

[28] P. Van Hentenryck, D. Mcallester, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM Journal on Numerical Analysis*, 34:797–827, 1997.

[29] H. Hong and V. Stahl. Safe starting regions by fixed points and tightening. *Computing*, 53(3-4):323–335, 1994.

[30] J. Horacek, M. Hladik, and J. Matejka. Determinants of interval matrices. *Electronic Journal of Linear Algebra*, 33:99–112, 2018.
31. R. Imbach. A subdivision solver for systems of large dense polynomials. arXiv:1603.07916, 2016.

32. R. Baker Kearfott. Rigorous Global Search: Continuous Problems. Springer, Boston, MA, 1996.

33. J. B. Kioustelidis. Algorithmic error estimation for approximate solutions of nonlinear systems of equations. Computing, 19(4):313–320, 1978.

34. R. Krawczyk. Newton-algorithmen zur bestimmung von nullstellen mit fehlerschranken. Computing, 4(3):187–201, 1969.

35. Anton Leykin, Jan Verschelde, and Ailing Zhao. Newton’s method with deflation for isolated singularities of polynomial systems. Theor. Comput. Sci., 359:111–122, 2006.

36. T.-Y. Li, T. Sauer, and J. A. Yorke. Numerically determining solutions of systems of polynomial equations. Bulletin of the American Mathematical Society, 18(2):173–177, 1988.

37. J. M. Lien, V. Sharma, G. Vegter, and C. Yap. Isotopic arrangement of simple curves: An exact numerical approach based on subdivision. In ICMS 2014, pages 277–282. Springer, 2014. LNCS No. 8592. Download from http://cs.nyu.edu/exact/papers/ for version with Appendices and details on MK Test.

38. A. Mantzaflaris, B. Mourrain, and E. Tsigaridas. On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. Theoretical Computer Science, 412(22):2312–2330, 2011.

39. C. Miranda. Un’osservazione su un teorema di Brouwer. Consiglio Nazionale delle Ricerche, 1940.

40. R. E. Moore. Interval analysis, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.

41. R. E. Moore. A test for existence of solutions to nonlinear systems. SIAM Journal on Numerical Analysis, 14(4):611–615, 1977.

42. R. E. Moore and J. B. Kioustelidis. A simple test for accuracy of approximate solutions to nonlinear (or linear) systems. SIAM Journal on Numerical Analysis, 17(4):521–529, 1980.

43. A. P. MORGAN. Computing all solutions to polynomial systems using homotopy continuation. Appl. Math. Comput., 24:115–138, 1987.

44. B. Mourrain and J. P. Pavone. Subdivision methods for solving polynomial equations. Journal of Symbolic Computation, 44(3):292–306, 2009.

45. A. Neumaier. Interval Methods for Systems of Equations. Cambridge University Press, 1991.

46. S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 245–254. ACM, 2004.
[47] S. M. Rump. Solving algebraic problems with high accuracy. In A new approach to scientific computation, pages 51–120. Elsevier, 1983.

[48] F. Shen, W. Wu, and B. Xia. Real root isolation of polynomial equations based on hybrid computation. In Ruyong Feng, Wen-shin Lee, and Yosuke Sato, editors, Computer Mathematics, pages 375–396, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[49] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. Computer Aided Geometric Design, 10(5):379–405, 1993.

[50] S. Smale. Newton’s method estimates from data at one point. In The merging of disciplines: new directions in pure, applied, and computational mathematics, pages 185–196. Springer, 1986.

[51] A. Storjohann. On the complexity of inverting integer and polynomial matrices. Comput. Complex., 24:777–821, 2015.

[52] H. Susanto and N. Karjanto. Newton’s method’s basins of attraction revisited. Applied Mathematics and Computation, 215(3):1084–1090, 2009.

[53] S. Telen, M. Van Barel, and J. Verschelde. A robust numerical path tracking algorithm for polynomial homotopy continuation. SIAM Journal on Scientific Computing, 42(6):A3610–A3637, 2020.

[54] J. Verschelde. Algorithm 795: Phcpack: a general-purpose solver for polynomial systems by homotopy continuation. ACM Transactions on Mathematical Software, 25:251–276, 1999.

[55] W. Wu. Mathematics mechanization: mechanical geometry theorem-proving, mechanical geometry problem-solving, and polynomial equations-solving. Science Press Beijing, 2000.

[56] J. Xu, M. Burr, and C. Yap. An approach for certifying homotopy continuation paths: Univariate case. ISSAC ’18, pages 399–406. ACM, 2018.

[57] J. Xu and C. Yap. Effective subdivision algorithm for isolating zeros of real systems of equations, with complexity analysis. ISSAC ’19, pages 355–362. ACM, 2019.