

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 effectivity 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, 56] in some literatures to overcome the shortcoming based on the famous α -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

30 method (see [34, 41, 47]) for possible real roots after homotopy continuation method is
31 applied in [48].

32 Subdivision method is also used to get the real solutions of the given systems, which
33 are not limited to polynomial case [7, 8, 25, 31, 27, 32, 38, 44, 45, 49, 57]. There are many
34 methods to exclude the domains without solutions [25, 44, 49, 29, 14]. More requirements
35 are needed to ensure that the boxes are isolating boxes. There are three main methods for
36 certifying a real root: Miranda theorem with Jacobian test, the interval Newton method,
37 and the α -theory.

38 Miranda theorem [39] and its practical version MK test [25, 33, 42] is used for checking
39 the existence of a real zero of a system inside a box. The MK test works well for linear
40 systems [42]. Jacobian test [1] is used for certifying that a system has at most one real
41 zero inside a box, so does the methods in [27]. Thus, the termination of the subdivision
42 method based on the MK test and Jacobian test is guaranteed in a theoretical sense for
43 square systems with simple roots. When MK test is used for systems with more than
44 two variables, it seldom succeeds which can also be found in our experiments. We also
45 analyze the reason in the experiments section. Besides that, the interval Newton method
46 [34, 41, 47] and α -theory [50] can work for testing the uniqueness and existence of the
47 complex (or real) zeros. The interval Newton method can verify that a box is an isolating
48 box. In [31, 32, 45, 28], the authors use the interval Newton method for real root finding.
49 But the termination of the method is not ensured for certifying a box containing a simple
50 zero or not by successive subdivision. Since an isolating box of a square system may not
51 satisfy the existence and uniqueness condition, the termination condition of subdivision is
52 absent. Thus the theory for root isolation of equation systems based on interval Newton
53 method is not complete even for systems with only simple roots. The α -theory is slightly
54 different from the other two methods, which computes derivatives with high orders, and the
55 verified domain of α -theory is a ball, not a box. Similar as the interval Newton method,
56 the termination of the α -theory method for root isolation of a square system is also not
57 complete. In [38], the authors present the concept of the α -inclusion box and use it for
58 seeking the real roots of a square system. In [14], we presented a new method which was
59 based on the geometrical property, the so-called orthogonal monotonicity inside a box for a
60 bivariate polynomial system to certify the existence and uniqueness of a real root inside the
61 box. We used bounding polynomials to exclude the regions which contained no roots. The
62 termination of the method is guaranteed. Thus it can be used for real root isolation inside
63 a box containing only simple real zeros. The method was extended to bivariate nonlinear
64 systems in the journal version [15]. We extend the method to general zero-dimensional
65 equation systems in this paper.

66 In this paper, we present a new existence criterion of a simple real root of a zero-
67 dimensional square system inside a box, which is much easier to succeed than Miranda
68 based criterion. Based on the new criterion, we propose an algorithm to isolate the real
69 roots of a zero-dimensional real nonlinear square system $F = (f_1, \dots, f_n)$ inside a given box
70 $B = [a_1, b_1] \times \dots \times [a_n, b_n]$, 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
71 to analyze the roots of the system locally inside a box, we give the concept of the opposite
72 monotone system (O-M system for short) in an n -D box and give a criterion to check the
73 existence and uniqueness of a simple real zero of the given system in the given box based on
74 properties of the opposite monotone system. Though the O-M system is firstly presented
75 in [14] for 2-D case (It is called orthogonal monotone in [14]), the O-M condition for n -D
76 systems are more complicated than that for 2-D systems.

77 Simply speaking, for a system F and an n -D box B , we transform locally the original
78 system F into a new system $G = (g_1, \dots, g_n)$ with the same zeros as F such that the curve
79 $S = \mathbb{V}(g_1, \dots, g_{n-1})$ is monotone in B and S intersects transversally with the hypersurface
80 $\mathbb{V}(g_n)$ in B (see Definition 3). The evaluation of the functions of the tangent vector of S on
81 B does not contain zeros ensures its monotonicity inside B . The direction of the tangent
82 vector of S and the normal vector of $\mathbb{V}(g_n)$ are almost identical or opposite when evaluating
83 on B , which ensures the uniqueness of the root inside B . The existence of a real zero of
84 G inside the box B can be determined by the change of the signs of the evaluations of g_n
85 on the two endpoints of S which are the intersection of S and the boundaries of B . Some
86 local transformation techniques in MK test [33, 42] are modified and used in our method.
87 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
88 matrix of F at the middle point of B and F^T is the transpose of F . In our method,
89 $G = U J_F(m_B)^{-1}F^T$, where U is an invertible matrix under some requirements. We prove
90 that the termination of the subdivision process for finding all simple real zeros of a system
91 inside a box. So our method is complete for real root isolation of a square nonlinear system
92 inside a bounded box.

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

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

113 2 Notations and Preliminaries

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

115 2.1 Notations

Let $C^i(\Omega)$ denote the set of all i -order continuous differentiable functions defined in Ω ,
where $\Omega \subset \mathbb{R}^n$ and \mathbb{R} is the field of real numbers. Let $B = [a_1, b_1] \times \dots \times [a_n, b_n]$ be an n -D
box in \mathbb{R}^n . Let

$$F_i^l(B) = \{(p_1, \dots, p_n) \in B | p_i = a_i\}, i = 1, \dots, n,$$

$$F_i^r(B) = \{(p_1, \dots, p_n) \in B | p_i = b_i\}, i = 1, \dots, n.$$

116 We call $F_i^l(B)$ or $F_i^r(B)$ a **face** of B for any $1 \leq i \leq n$. Let $m(B) = (\frac{a_1+b_1}{2}, \dots, \frac{a_n+b_n}{2})$ be
 117 the middle point of the box B and $w(B) = \max\{b_1 - a_1, \dots, b_n - a_n\}$ be the width of B . Let
 118 $\partial B = \bigcup_{i=1}^n F_i^l(B) \cup \bigcup_{i=1}^n F_i^r(B)$ be the boundaries of the box and $v(B) = \{(p_1, \dots, p_n) \in$
 119 $B | p_i = a_i \text{ or } b_i, i = 1, \dots, n\}$ be the set of the vertexes of B .

120 Let $F = (f_1(X), \dots, f_n(X))$ be a function system, where $X = \{x_1, \dots, x_n\}$ are variables
 121 and $f_i(X) \in C^1(B)$. We denote $F = 0$ as the equation system $\{f_1 = 0, \dots, f_n = 0\}$. If
 122 a point $\mathbf{p} \in \mathbb{R}^n$ satisfies $f_1(\mathbf{p}) = \dots = f_n(\mathbf{p}) = 0$, then we call \mathbf{p} is a real zeros of the
 123 function system F or a real root of equation system $F = 0$. We denote all the real zeros of
 124 F as $\mathbb{V}(F)$. Let J_F^X be the Jacobian matrix of F with respect to X (simply for J_F without
 125 misunderstanding). Denote $\mathbb{I}\mathbb{R}, \mathbb{I}\mathbb{R}^n$ and $\mathbb{I}\mathbb{R}^{n \times n}$ by the set of real intervals, n -D interval
 126 vectors and $n \times n$ interval matrices, respectively. For a (an interval) matrix $M \in \mathbb{R}^{n \times n}$
 127 ($\mathbb{I}\mathbb{R}^{n \times n}$), we let $M_{i,:}$ ($M_{:,i}$) denote the i -th row (column) of M .

Let $f(X) \in C^1(B)$ be a real function and B an n -D box. For a subset $B' \subset B$, we
 denote $f(B') = \{f(\mathbf{p}) | \mathbf{p} \in B'\}$ and we say $f(B') > 0 (< 0)$ if $\forall \mathbf{p} \in B', f(\mathbf{p}) > 0 (< 0)$.
 Similarly, for an interval $I \subset \mathbb{R}$, we say $I > 0 (< 0)$ if $\forall a \in I, a > 0 (< 0)$. We define a sign
 function of $f(B')$ as following:

$$\text{Sign}(f(B')) = \begin{cases} 1, & \text{if } f(B') > 0, \\ -1, & \text{if } f(B') < 0, \\ 0, & \text{otherwise.} \end{cases}$$

128 2.2 Interval Analysis

129 Using our method, we need to compute the evaluation of a function $f(x_1, \dots, x_n) \in$
 130 $C^1(B)$ on a box $B = I_1 \times \dots \times I_n \subset \mathbb{R}^n$: $f(B) = \{f(\mathbf{p}) | \mathbf{p} \in B\}$. However, $f(B)$ is
 131 usually difficult to be computed exactly. Interval analysis [40] is a useful tool to compute
 132 the enclosure of the range of a function over a box. A real function f can be extended to
 133 an **interval function** by interval analysis. The basic arithmetic operations over intervals
 134 are as below. Let $I_1 = [a, b] \subset \mathbb{R}, I_2 = [c, d] \subset \mathbb{R}$.

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

135 Let $\square f$ denote the interval function of f , it has two properties [37]:

- 136 1. $f(B) \subset \square f(B)$,
- 137 2. $\lim_{i \rightarrow \infty} \square f(B_i) = f(\lim_{i \rightarrow \infty} B_i)$,

138 where $B, B_i \subset \mathbb{R}^n$ and $\lim_{i \rightarrow \infty} B_i = \mathbf{p}, \mathbf{p}$ is a point in \mathbb{R}^n . There are many different forms of
 139 $\square f$, for polynomial case, a simple way is just using interval arithmetic [40]. Fox example,
 140 $g = x^2 - x, B = [0, 1]$, then $\square g(B) = [0, 1] \cdot [0, 1] - [0, 1] = [0, 1] - [0, 1] = [-1, 1]$. Since
 141 $g(B) = [-\frac{1}{4}, 0]$, we can find that $\square g(B)$ is much bigger than $g(B)$. Notice that most of the
 142 real functions (such as exp, sin, cos, etc.) are also easy to be extended to interval functions.

143 Since we usually can not get the exact representation of $f(B)$, we use it to represent
 144 $\square f(B)$ for simplicity if there is no doubt in the rest of the paper.

145 3 Uniqueness and Existence

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

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

151 3.1 An opposite monotone (O-M) system in a box

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

$$\mathbf{T}_{\mathbf{p}} = (\det(T_1(\mathbf{p})), \dots, (-1)^{i+1} \det(T_i(\mathbf{p})), \dots, (-1)^{n+1} \det(T_n(\mathbf{p}))), \quad (1)$$

157 where $X_i = X \setminus x_i = \{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n\}$ and $T_i = J_{G'}^{X_i}, i = 1, \dots, n$.

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

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

162 (1) for any $i \in \{1, \dots, n\}$, $0 \notin u_i$ and $0 \notin v_i$.

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

164 For example, $([1, 2], [2, 3], [-2, -1])$ and $([-4, -3], [-2, -1], [1, 2])$ are matched. Let
165 $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$ be interval vectors. It is easily to see that the following properties hold.

166 1. If \mathbf{U}_1 and \mathbf{U}_2 are matched, \mathbf{U}_2 and \mathbf{U}_3 are matched, then \mathbf{U}_1 and \mathbf{U}_3 are matched.

167 2. If \mathbf{U}_1 and \mathbf{U}_2 are matched, then \mathbf{U}_1 and $-\mathbf{U}_2$ are matched.

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

Lemma 1. (Mean Value Theorem) Let $\Omega \subset \mathbb{R}^n$ be a convex set, $f = f(x_1, \dots, x_n)$ is a
differentiable function defined on Ω . Then $\forall p_1, p_2 \in \Omega, \mathbf{p}_1 \neq \mathbf{p}_2, \exists \theta \in (0, 1)$ s.t.

$$f(\mathbf{p}_1) - f(\mathbf{p}_2) = \nabla f(\mathbf{p}_2 + \theta(\mathbf{p}_1 - \mathbf{p}_2))(\mathbf{p}_1 - \mathbf{p}_2).$$

170 **Definition 2.** Let $G' = (g_1, \dots, g_{n-1})$ and $S = \mathbb{V}(G') = \{\mathbf{p} \in \mathbb{R}^n | g_1(\mathbf{p}) = \dots = g_{n-1}(\mathbf{p}) =$
171 $0\}$. We say S is **strong monotonous** in B if $0 \notin \det(T_i(B)), i = 1, \dots, n$ (see (1) for T_i).

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

$$T_1 = J_{G'}^{\{y,z\}} = \begin{pmatrix} 2 * y & -4 * z \\ y & -z \end{pmatrix}, T_2 = J_{G'}^{\{x,z\}} = \begin{pmatrix} x & -4 * z \\ x & -z \end{pmatrix}, T_3 = J_{G'}^{\{x,y\}} = \begin{pmatrix} x & 2 * y \\ x & y \end{pmatrix}.$$

172 Thus,

$$\begin{aligned}
\det(T_1(B)) &= \det\left(\begin{pmatrix} 2 * [0.10, 0.11] & -4 * [0.10, 0.11] \\ [0.10, 0.11] & -1 * [0.10, 0.11] \end{pmatrix}\right) \\
&= -2 * [0.10, 0.11] * [0.10, 0.11] + 4 * [0.10, 0.11] * [0.10, 0.11] \\
&= -2 * [0.01, 0.0121] + 4 * [0.01, 0.0121] \\
&= [-0.0242, -0.02] + [0.04, 0.0484] \\
&= [0.0158, 0.0284].
\end{aligned}$$

173 Similarly, we have $\det(T_2(B)) = [0.0279, 0.0384]$, $\det(T_3(B)) = [-0.0142, -0.0079]$. So $S =$
174 $\mathbb{V}(G')$ is strong monotonous in B since $0 \notin \det(T_i(B))$ for $i = 1, 2, 3$.

175 Then, we will prove some nice properties of S if it is strong monotonous in B . Let
176 $\mathbf{p} = (p_1, \dots, p_n) \in \mathbb{R}^n$, we define $\Pi_i(\mathbf{p}) = p_i$, for any $i = 1, \dots, n$.

177 **Lemma 2.** Let $G' = (g_1, \dots, g_{n-1})$ and $S = \mathbb{V}(G')$. If S is strong monotonous in $B =$
178 $[a_1, b_1] \times \dots \times [a_n, b_n]$, then we have:

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

180 Moreover, the hyperplane and S are not tangent.

181 (b) S can not be a loop in B .

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

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

$$\sum_{i=1}^n \frac{\partial g_j}{\partial x_i}(\mathbf{q}_j) \Delta x_i = g_j(\mathbf{p}) - g_j(\mathbf{p}') = 0, j = 1, \dots, n-1.$$

183 Since $\Delta x_1 = 0$, thus we have the following equation

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ \frac{\partial g_1}{\partial x_1}(\mathbf{q}_1) & \frac{\partial g_1}{\partial x_2}(\mathbf{q}_1) & \dots & \frac{\partial g_1}{\partial x_n}(\mathbf{q}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{n-1}}{\partial x_1}(\mathbf{q}_{n-1}) & \frac{\partial g_{n-1}}{\partial x_2}(\mathbf{q}_{n-1}) & \dots & \frac{\partial g_{n-1}}{\partial x_n}(\mathbf{q}_{n-1}) \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{pmatrix} = \mathbf{0}.$$

184 Let A denote the above matrix, i.e., $A\Delta x^T = \mathbf{0}$. Since Δx is nonzero, it implies $\det(A) =$
185 0 . However, since S is strong monotonous in B , we have $0 \notin \det(T_1(B))$, then we have
186 $\det(A) = 1 \det(M_1) \neq 0$, where $M_1 \in T_1(B)$. It is a contradiction. Moreover, if $x_1 = \hat{x}_1$
187 and S are tangent at point \mathbf{p} in B , then $T_1(\mathbf{p}) = 0$. It is also a contradiction since S is
188 strong monotonous in B .

189 (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
190 points in B . It is a contradiction with (a).

191 (c) For any two point \mathbf{p}, \mathbf{q} , let $\mathbf{p} - \mathbf{q} = \Delta x = (\Delta x_1, \Delta x_2, \dots, \Delta x_n)$. By Lemma 2 (a),
192 we know that $\Delta x_i \neq 0, \forall i = 1, \dots, n$. Then, we need only to prove that for $i = 2, \dots, n$,
193 $\Delta x_1 \det(T_1(B))$ and $\Delta x_i (-1)^{i+1} \det(T_i(B))$ have the same signs.

194 We consider the following equation system

$$\begin{pmatrix} \Delta x_i & 0 & \cdots & 0 & -\Delta x_1 & 0 & \cdots & 0 \\ \frac{\partial g_1}{\partial x_1}(\mathbf{q}_1) & \cdots & \cdots & \cdots & \frac{\partial g_1}{\partial x_i}(\mathbf{q}_1) & \cdots & \cdots & \frac{\partial g_1}{\partial x_n}(\mathbf{q}_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial g_{n-1}}{\partial x_1}(\mathbf{q}_{n-1}) & \cdots & \cdots & \cdots & \frac{\partial g_{n-1}}{\partial x_i}(\mathbf{q}_{n-1}) & \cdots & \cdots & \frac{\partial g_{n-1}}{\partial x_n}(\mathbf{q}_{n-1}) \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{pmatrix} = \mathbf{0}.$$

195 Let A' denote the above matrix, whose last $n - 1$ rows are the same as A , we have

$$\det(A') = \Delta x_i \det(M_1) - \Delta x_1 (-1)^{i+1} \det(M_i),$$

196 where $M_1 \in T_1(B)$, $M_i \in T_i(B)$. If

$$\Delta x_1 \det(T_1(B)) \cdot \Delta x_i (-1)^{i+1} \det(T_i(B)) < 0,$$

197 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
198 $\Delta x = \mathbf{0}$. It is a contradiction with \mathbf{p}, \mathbf{q} are two different points. Therefore,

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

199 Notice that Δx_i and $\det(M_i)$ are all nonzero for $i = 1, \dots, n$. Then we know that for
200 any $j = 1, \dots, n$, $\Delta x_j (-1)^{j+1} \det(T_j(B))$ have the same signs, i.e., $(\Pi_i(\mathbf{p} - \mathbf{q}))_{1 \leq i \leq n}$ and
201 $((-1)^{i+1} \det(T_i(B)))_{1 \leq i \leq n}$ are matched. ■

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

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

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

208 **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

$$\begin{aligned} (-1)^{1+3} \det(M_{\{3,1\}}) &= [14, 25], \\ (-1)^{2+3} \det(M_{\{3,2\}}) &= [-16, -1], \\ (-1)^{3+3} \det(M_{\{3,3\}}) &= [-18, -7], \end{aligned}$$

209 and

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

210 Therefore, M is an O-M matrix.

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

213 (1) S is strong monotonous in B .

214 (2) $((-1)^{i+n} T_i(B))_{1 \leq i \leq n}$ and $(\frac{\partial f_n}{\partial x_i}(B))_{1 \leq i \leq n}$ are matched, where $T_i = J_{F'}^{X_i}$ (see (1) for more
215 details).

216 i.e., $J_F^X(B)$ is an O-M matrix.

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

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

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

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

223 However, by Lemma 2, since S is strong monotonous in B , we have $(\Pi_i(\mathbf{p} - \mathbf{q}))_{1 \leq i \leq n}$
 224 and $((-1)^{i+1} \det(T_i(B)))_{1 \leq i \leq n}$ are matched. By the definition of O-M system, we have
 225 $((-1)^{i+n} T_i(B))_{1 \leq i \leq n}$ and $(\frac{\partial f_n}{\partial x_i}(B))_{1 \leq i \leq n}$ are matched. Therefore, $(\Pi_i(\mathbf{p} - \mathbf{q}))_{1 \leq i \leq n}$ and
 226 $(\frac{\partial f_n}{\partial x_i}(B))_{1 \leq i \leq n}$ are matched. Then, we know that

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

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

228 3.2 Preconditioner

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

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

233 **Lemma 4.** Let $F = (f_1, \dots, f_n)$ and $M \in \mathbb{R}^{n \times n}$ be an $n \times n$ invertible matrix. Then
 234 $\mathbb{V}(F) = \mathbb{V}(MF^T)$, where F^T is the transpose of F .

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

238 ■

239 We give the preconditioner which transforms locally a square system F into a new
 240 system $UJ_F^{-1}(\mathbf{p}) \cdot F^T$, where $U \in \mathbb{R}^{n \times n}$ is an O-M matrix (In the rest of the paper, U
 241 always denotes an O-M matrix) and $\mathbf{p} \in B$. In general, we choose \mathbf{p} as $m(B)$. The idea
 242 of multiplying $J_F^{-1}(\mathbf{p})$ to the original system originates from [33] and used by [24, 42, 44].
 243 They just transform locally the system F into $J_F^{-1}(\mathbf{p}) \cdot F^T = (\tilde{f}_1, \dots, \tilde{f}_n)^T$ s.t. $\mathbb{V}(\tilde{f}_i)$ are
 244 almost orthogonal to each other in the neighborhood of \mathbf{p} . Then the Miranda theorem can
 245 be used to check the existence of a real zero of the system. But we further do a rotation on
 246 $J_F^{-1}(\mathbf{p}) \cdot F^T$ by multiplying the matrix U to make the system $UJ_F^{-1}(\mathbf{p}) \cdot F^T$ becoming an
 247 O-M system in B . See the example below for illustration. A similar example to illustrate
 248 the same problem can be found in [14].

249 **Example 3.** Let $F = (2y^2 - z^2 + 2x + 0.16, x^2 + x + 3y - z - 0.02, 3x^2 - 5.08x - 0.2492 -$
250 $4y^2 - 6z + 3z^2 + 3y)$, $B = [-0.09, -0.04] \times [0.01, 0.06] \times [0.01, 0.06]$, $U = \begin{pmatrix} 3 & 1 & 1 \\ 1 & -3 & 1 \\ 1 & 1 & -3 \end{pmatrix}$.
251 F has a unique zero $(-0.080966, 0.049827, 0.055071)$ in B . However, we can find that F
252 is not an O-M system in B . The preconditioner in [33] produces $G' = J_F^{-1}(m(B)) \cdot F^T =$
253 $\begin{bmatrix} 1.09864y^2 - 0.570758z^2 + 1.11937x + 0.0864214 - 0.0667392x^2 + 0.131059z - 0.0715789y \\ -2.42325y^2 + 1.62329z^2 - 2.93426x - 0.163387 + 1.16858x^2 - 2.40357z + 1.03577y \\ -2.02830y^2 + 1.22491z^2 - 2.45787x - 0.143304 + 0.430319x^2 - 1.06258z + 0.026427y \end{bmatrix}$, but
254 our preconditioner generates $G = UJ_F^{-1}(m(B)) \cdot F^T =$
255 $\begin{bmatrix} -1.15563y^2 + 1.13593z^2 - 2.03402x - 0.047427 + 1.39868x^2 - 3.07297z + 0.847460y \\ 6.34009y^2 - 4.21572z^2 + 7.46428x + 0.433278 - 3.14216x^2 + 6.27919z - 3.15246y \\ 4.76029y^2 - 2.62220z^2 + 5.55872x + 0.352946 - 0.18912x^2 + 0.91523z + 0.884910y \end{bmatrix}$. We can
256 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 .

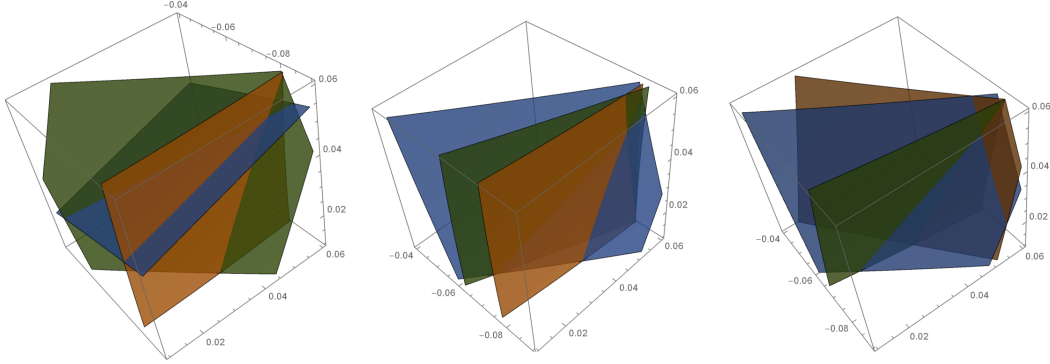


Figure 1: The left figure is of $F = 0$, the middle one is of $G' = 0$, the right one is of $G = 0$.

257

258 **Remark:** We have many choices for the O-M matrix U . For example, when $n = 2$,
259 $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and $\begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$ are both O-M matrices. When $n = 3$, $\begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & -1 & 2 \end{pmatrix}$, $\begin{pmatrix} 3 & 1 & 1 \\ 1 & -3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$
260 and $\begin{pmatrix} 3 & 1 & 1 \\ 1 & -3 & 1 \\ 1 & 1 & -3 \end{pmatrix}$ are all O-M matrices. We will discuss how to select a better rotation
261 matrix later.

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

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

266 **Theorem 5.** Let $F = (f_1, \dots, f_n)$, $f_i \in C^1(B)$, $1 \leq i \leq n$ and \mathbf{p}^* a simple zero of F . Then,
267 $\exists \delta > 0$ s.t. $\forall B \in B(\mathbf{p}^*, \delta)$, $UJ_F^{-1}(m(B)) \cdot F^T$ is an O-M system in B .

268 *Proof.* Let $G = UJ_F^{-1}(m(B)) \cdot F^T$, when $B \rightarrow \mathbf{p}^*$, we have $\lim_{B \rightarrow \mathbf{p}^*} J_G(B) = \lim_{B \rightarrow \mathbf{p}^*} UJ_F^{-1}(m(B)) \cdot$
269 $J_F(B) = UJ_F^{-1}(\mathbf{p}^*) \cdot J_F(\mathbf{p}^*) = U$. Since U is an O-M matrix, we prove the theorem. \blacksquare

270 **Remark:** If \mathbf{p}^* is a singular root, then for any box B containing \mathbf{p}^* , we can not transform
 271 F into an O-M system in B since $0 = \det(J_F(\mathbf{p}^*)) \in \det(J_F(B))$. Hence, our method is
 272 invalid for singular roots.

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

275 3.3 Existence

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

278 **Lemma 6.** *Let $G' = (g_1, \dots, g_{n-1})$ and $S = \mathbb{V}(G')$. If S is strong monotonous in $B =$
 279 $[a_1, b_1] \times \dots \times [a_n, b_n]$, then we have that B contains at most one connected component of
 280 S .*

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

284 1. If L intersects the boundaries of B twice, there are two cases:

285 (1.1) L passes B from $F_i^l(B)$ to $F_i^r(B)$. For a point $\mathbf{q} \in L'$, we have $\pi_i(\mathbf{q}) \in [a_i, b_i]$.
 286 Since L is a connected component, $\exists \mathbf{p} \in L$ such that $\Pi_i(\mathbf{p}) = \Pi_i(\mathbf{q})$. It is a contradiction
 287 with Lemma 2.

(1.2) L passes B from $F_i^l(B)$ to $F_j^r(B)$, $i \neq j$. Let $L \cap F_i^l(B) = \{\mathbf{p}_1\}$ and $L \cap F_j^r(B) =$
 $\{\mathbf{p}_2\}$, then we have $\Pi_i(\mathbf{p}_1) = a_i$ and $\Pi_j(\mathbf{p}_2) = b_j$. For a point $\mathbf{q} \in L'$, since L is a
 connected component and Lemma 2 (a), we have $\Pi_i(\mathbf{q}) \notin [a_i, \Pi_i(\mathbf{p}_2)]$ i.e., $\Pi_i(\mathbf{q}) > \Pi_i(\mathbf{p}_2)$
 and obviously $\Pi_j(\mathbf{q}) < b_j = \Pi_j(\mathbf{p}_2)$. Then we have

$$\Delta x_i = \Pi_i(\mathbf{p}_1) - \Pi_i(\mathbf{p}_2) < 0, \Delta x_j = \Pi_j(\mathbf{p}_1) - \Pi_j(\mathbf{p}_2) < 0,$$

$$\Delta x'_i = \Pi_i(\mathbf{q}) - \Pi_i(\mathbf{p}_2) > 0, \Delta x'_j = \Pi_j(\mathbf{q}) - \Pi_j(\mathbf{p}_2) < 0.$$

288 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), $\forall \mathbf{p} \neq$
 289 $\mathbf{q} \in S \cap B$, $\Pi_i(\mathbf{p} - \mathbf{q})(-1)^{i+1} \det(T_i(B)) \Pi_j(\mathbf{p} - \mathbf{q})(-1)^{j+1} \det(T_j(B)) > 0$. Since the sign of
 290 $(-1)^{j+1} \det(T_i(B)) (-1)^{j+1} \det(T_j(B))$ is unchanged, we know the sign of $\Pi_i(\mathbf{p} - \mathbf{q}) \Pi_j(\mathbf{p} - \mathbf{q})$
 291 is unchanged too. Therefore we get a contradiction. Notice that L passes B from $F_i^l(B)$ to
 292 $F_j^l(B)$ ($F_i^l(B)$ to $F_j^l(B)$, $F_i^r(B)$ to $F_j^r(B)$) can be proved in a similar way.

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

302 By Lemma 2, we know that if S is strong monotonous in B , then S intersects the
 303 boundaries of the box ∂B at most twice, that is to say $\#(S \cap \partial B) \leq 2$, where $\#(A)$ denotes
 304 the number of elements of a set A . Next, we determine whether F has a root or not in B
 305 based on the three cases: $\#(S \cap \partial B) = 0, 1, 2$.

306 If $\#(S \cap \partial B) = 0$, since S can not be a loop in B , then we know that S dose not pass
 307 the box B . If $\#(S \cap \partial B) = 1$, we know that S intersects only the box at a point and this
 308 point must be a vertex of B . Next, we will analyze the last case: $\#(S \cap \partial B) = 2$.

309 **Lemma 7.** *Let $F = (f_1, \dots, f_n)$ be an O-M system in B . Let $F' = (f_1, \dots, f_{n-1})$ and
 310 $S = \mathbb{V}(F')$. Assume that $S \cap \partial B = \{\mathbf{p}_1, \mathbf{p}_2\}$, we have:*

311 (1) *If $f_n(\mathbf{p}_1)f_n(\mathbf{p}_2) \leq 0$, $F = 0$ has a unique root in B .*

312 (2) *If $f_n(\mathbf{p}_1)f_n(\mathbf{p}_2) > 0$, $F = 0$ has no root in B .*

313 *Proof.* Since B contains a unique component of S , we can parameterize it as $\eta(t) =$
 314 $(x_1(t), \dots, x_n(t))$, $t_1 \leq t \leq t_2$, where $\eta(t_i) = \mathbf{p}_i, i = 1, 2$. Since S is strong monotone
 315 inside B , $\eta(t)(t_1 \leq t \leq t_2)$ is strictly contained inside B . Consider the univariate function
 316 $g(t) = f_n(x_1(t), \dots, x_n(t))$. If $f_n(\mathbf{p}_1)f_n(\mathbf{p}_2) \leq 0$, i.e. $g(t_1)g(t_2) \leq 0$, then $\exists t' \in [t_1, t_2]$ s.t.
 317 $g(t') = 0$. Thus $(x_1(t'), \dots, x_n(t'))$ is a root of $F = 0$. Since F is an O-M system in B , we
 318 know that $F = 0$ has a unique root in B . If $f_n(\mathbf{p}_1)f_n(\mathbf{p}_2) > 0$, i.e., $g(t_1)g(t_2) > 0$, then
 319 there are even number roots (counting multiplicity) of $g(t) = 0$ in $[t_1, t_2]$. Since $F = 0$ has
 320 most one root in B , we can know that $F = 0$ has no root in B . ■

321 3.4 Checking the existence

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

324 Let $F = (f_1, \dots, f_n)$ be an O-M system in B and $F' = (f_1, \dots, f_{n-1})$. By the defini-
 325 tion of the O-M system, we know that $S = \mathbb{V}(f_1, \dots, f_{n-1})$ is strong monotonous in B .
 326 Then by Lemma 7, in order to check the existence, we need to compute the intersection
 327 points $\mathbf{p}_1, \mathbf{p}_2$ of S and ∂B . However, it is not easy and unnecessary to get the points
 328 exactly, we need only to get two $(n - 1)$ -boxes containing $\mathbf{p}_1, \mathbf{p}_2$, say $B_{\mathbf{p}_1}, B_{\mathbf{p}_2}$. Then by
 329 computing $\text{Sign}(f_n(B_{\mathbf{p}_1})), \text{Sign}(f_n(B_{\mathbf{p}_2}))$, we can get $\text{Sign}(f_n(\mathbf{p}_1)), \text{Sign}(f_n(\mathbf{p}_2))$. The signs
 330 of $f_n(B_{\mathbf{p}_1}), f_n(B_{\mathbf{p}_2})$ can be obtained by interval computation, since $0 \notin \frac{\partial f_n}{\partial x_i}(B)$, we can use
 331 the following lemma to compute the signs in an easier way.

332 **Lemma 8.** *Let $f(x_1, \dots, x_n) \in C^1(B)$ and $B = [a_1, b_1] \times \dots \times [a_n, b_n]$ a box. If $0 \notin$
 333 $\frac{\partial f}{\partial x_i}(B), i = 1, \dots, n$ and $f(v(B)) > 0 (< 0)$, then $f(B) > 0 (< 0)$.*

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

336 We first prove the case $n = 1$, i.e., $B = [a_1, b_1]$: Since $\frac{\partial f}{\partial x_1}(B) > 0$, we know that f is a
 337 monotonous univariate function, then we have $f(B) > 0$ since $f(a_1) > 0$ and $f(b_1) > 0$.

338 Next we assume that the lemma is proved for $n = k - 1$ and we are going to prove the case
 339 $n = k$, where k is a positive integer. Let $B_i = [a_1, b_1] \times \dots \times [a_{i-1}, b_{i-1}] \times [a_{i+1}, b_{i+1}] \times \dots \times$
 340 $[a_n, b_n] \subset \mathbb{R}^{n-1}, i = 1, \dots, n$ and $t_i = a_i$ or b_i . $\forall 1 \leq i \leq n$, since $f|_{x_i=t_i}(v(B_i)) \subset f(v(B))$,
 341 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
 342 function $f|_{x_i=t_i}$ is with $k - 1$ variables. In summary, we have $f(\partial(B)) > 0$. $\forall p_1 \in [a_1, b_1]$,
 343 we have $f|_{x_1=p_1}(v(B_1)) \subset f(\partial(B))$, thus $f|_{x_1=p_1}(v(B_1)) > 0$. Then by the assumption we
 344 have $f|_{x_1=p_1}(B_1) > 0$. By the arbitrariness of p_1 , we have $f(B) > 0$. ■

345 By Lemma 8, we need only to compute the signs of f_n at the vertexes of $B_{\mathbf{p}_1}, B_{\mathbf{p}_2}$. If
 346 $f_n(v(B_{\mathbf{p}_1})), f_n(v(B_{\mathbf{p}_2}))$ are all positive or negative, then we know the signs of $f_n(B_{\mathbf{p}_1}), f_n(B_{\mathbf{p}_2})$.
 347 So do the signs of $f_n(\mathbf{p}_1), f_n(\mathbf{p}_2)$.

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 $B_i = [a_1, b_1] \times \cdots \times [a_{i-1}, b_{i-1}] \times [a_{i+1}, b_{i+1}] \times \cdots \times [a_n, b_n] \subset \mathbb{R}^{n-1}$, $i = 1, \dots, n$. For all $p_i \in [a_i, b_i]$, let $F'|_{x_i=p_i} = (f_1|_{x_i=p_i}, \dots, f_{n-1}|_{x_i=p_i}) = (f_1(x_1, \dots, p_i, \dots, x_n), \dots, f_{n-1}(x_1, \dots, p_i, \dots, x_n))$. For convenience, we define the index set of the box B with $2n$ elements as follows

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

348 If $S \cap \partial B = \{\mathbf{p}_1, \mathbf{p}_2\}$, it is easy to see that there exist $(i_1, t_1), (i_2, t_2) \in Ind(B)$ such
 349 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
 350 $(i, t) \in Ind(B)$, we need to know the square system $F'|_{x_i=t}$ has a zero in B_i or not. We can
 351 solve those systems by our method recursively. Finally we need to isolate some bivariate
 352 systems, which is solved in [14]. Notice that for non-polynomial case, the method is also
 353 valid. We give the algorithm structure of checking existence as following:

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

366 In Step(1), we will solve $2n$ systems with $n - 1$ equations and $n - 1$ variables in the
 367 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
 368 functions recursively in the worst case.

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

374 Based on the above discussion, if a system and a box satisfy our O-M and existence
 375 conditions, we can determine that the box is an isolating box of the system. We can
 376 design an algorithm for real root isolation based on the O-M condition and the subdivision
 377 method. However, from the discussion above, we know that a recursive solving is required
 378 in our existence checking step. We need to ensure that the related system in the related
 379 box is an O-M system. In order to avoid constructing an O-M system for each one by
 380 one, we prefer to construct all these systems in their related boxes together. In doing so,
 381 we introduce the Jacobian test. One can find that the O-M condition implies the famous
 382 Jacobian test based on the following theorem. One can find it in many places, see the
 383 corollary of Theorem 12.1 in [1].

384 **Theorem 9.** Let $G = (g_1, \dots, g_n), g_i \in C^1(B)$. If $0 \notin \det(J_G^X(B))$, then $G = 0$ has at most
385 one real root in B .

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

395 3.5 Strong Monotone system (S-M)

396 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
397 i rows of M and denote $S(M_i)$ as the set of all i -order sub-matrices of M_i . We give the
398 following definition:

399 **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

- 400 • $0 \notin \det(A), \forall A \in S(M_i), 1 \leq i \leq n$.

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

403 **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
404 an S-M matrix. First, we have $0 \notin m_{1j}, 1 \leq j \leq 3$. Then for $i = 2$, we have that $M_2 =$
405 $\begin{pmatrix} [3, 4] & [1, 2] & [1, 2] \\ [1, 2] & [3, 4] & [-2, -1] \end{pmatrix}$ and $S(M_2) = \{A_1, A_2, A_3\} = \left\{ \begin{pmatrix} [1, 2] & [1, 2] \\ [3, 4] & [-2, -1] \end{pmatrix}, \begin{pmatrix} [3, 4] & [1, 2] \\ [1, 2] & [-2, -1] \end{pmatrix}, \right.$
406 $\left. \begin{pmatrix} [3, 4] & [1, 2] \\ [1, 2] & [3, 4] \end{pmatrix} \right\}$. After computing, we have $\det(A_1) = [-12, -4], \det(A_2) = [-12, -4]$ and
407 $\det(A_3) = [5, 15]$. For $i = 3$, we have $M_3 = M, S(M_3) = \{M\}$ and $\det(M) = [-72, -16]$.
408 Therefore, M is an S-M matrix.

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

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

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

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

417 For simplification, we set $A = \mathbf{IsSMSys}(F, X, B)$ as the algorithm to check whether
418 $J_F^X(B)$ is an S-M system or not. Interval evaluation can be used here to get $J_F^X(B)$.

419 Similar as Theorem 5, we have the following result.

420 **Theorem 11.** Let $F = (f_1, \dots, f_n), f_i \in C^1(B), 1 \leq i \leq n$ and \mathbf{p}^* a simple zero of F .
421 Then, $\exists \delta > 0$ s.t. $\forall B \in B(\mathbf{p}^*, \delta), VJ_F^{-1}(m(B)) \cdot F^T$ is an S-M system in B , where V is a
422 S-M matrix.

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

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

428 *Proof.* It can be derived from Definition 6 directly. ■

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

433 **Lemma 13.** Let $F = (f_1, \dots, f_n)$ be an S-M system. Let $F' = (f_1, \dots, f_{n-1})$ and $S =$
434 $\mathbb{V}(F')$. Assume that $S \cap \partial B = \{\mathbf{p}_1, \mathbf{p}_2\}$, we have:

435 (1) If $f_n(\mathbf{p}_1)f_n(\mathbf{p}_2) \leq 0, F = 0$ has a unique root in B .

436 (2) If $f_n(\mathbf{p}_1)f_n(\mathbf{p}_2) > 0, F = 0$ has no root in B .

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

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

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

- 439 1. By Lemma 12, $F'|_{x_i=t}$ is still S-M system in B_i .
- 440 2. Let $F'|_{x_i=t} = (f'_1, \dots, f'_{n-1})$. If $f'_j(v(B_i)) > 0$ or < 0 for some $j \in \{1, \dots, n-1\}$, then
441 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
442 need to solve $F'|_{x_i=t}$.

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

444 **Example 5.** Let $F = (f_1, f_2, f_3) = (x - y + z, y^2 + x + y + 2z, x^2 + yz - 3x - y + z)$
445 and $B = [-0.1, 0.1] \times [-0.1, 0.1] \times [-0.1, 0.1]$, B_1, B_2, B_3 are all $[-0.1, 0.1] \times [-0.1, 0.1]$.
446 One can verify that F is an S-M system in B . Next we check the existence condition. Let
447 $F' = (f_1, f_2)$ and other notations are as above, we need to check whether these systems
448 $F'|_{x_i=t_i}$ 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
449 systems can be checked easily, for example, let $g = f_2|_{z=0.1} = y^2 + x + y + 0.2$, we have
450 $g(0.1, 0.1) > 0, g(0.1, -0.1) > 0, g(-0.1, 0.1) > 0$ and $g(-0.1, -0.1) > 0$, i.e., $g(v(B_3)) > 0$.
451 By Lemma 8, we know that $g(B_3) > 0$, hence $F'|_{z=0.1}$ has no zero in B_3 and we do not
452 need to solve the system). Finally we find both the two systems $F'|_{x=0.1}$ and $F'|_{x=-0.1}$ have
453 a unique zero in B_1 , see the left and right figures in Figure 2. Then we check the condition
454 of Lemma 13, by simple evaluation, we have $f_3|_{x=0.1}(v(B_1)) < 0$ and $f_3|_{x=-0.1}(v(B_1)) > 0$.
455 Therefore we know that $F = 0$ has a unique zero in B .

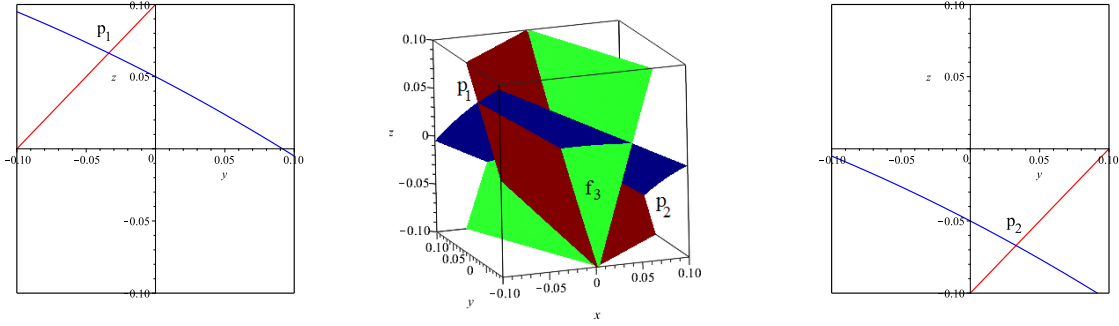


Figure 2: The figures for Example 5

456

Based on the discussion above, we give an algorithm **Existence** below.

Algorithm 1 $A = \mathbf{Existence}(F, B, \epsilon_b)$:

1. **Input:** An S-M system $F = (f_1, \dots, f_n)$, a box B , a given precision $\epsilon_b > 0$ for refinement, where $X = \{x_1, \dots, x_n\}$ are variables of F .
 2. **Output:** 1,0,or **Unknown**.
1. $F' \leftarrow (f_1, \dots, f_{n-1}), T \leftarrow \text{Ind}(B), num \leftarrow 0, S = \{\}$.
 2. While $T \neq \emptyset$ and $num < 2$
 - Take (i, t) from T and $T = T \setminus \{(i, t)\}$.
 - $b \leftarrow \mathbf{Existence}(F'|_{x_i=t}, B_i, \epsilon_b)$.
 - If $b = \mathbf{Unknown}$, return **Unknown**.
 - 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
 - Refine B_i w.r.t. $F'|_{x_i=t}$, and still denote the refined boxes as B_i .
 - $t_1 \leftarrow \text{Sign}(f_n|_{x_i=t}(v(B_i)))$.
 - c. While $w(B_j) > \epsilon_b$ and $t_2 = 0$ do
 - Refine B_j w.r.t. $F'|_{x_j=t'}$, and still denote the refined boxes as B_j .
 - $t_2 \leftarrow \text{Sign}(f_n|_{x_j=t'}(v(B_j)))$.
 - d. If $t_1 t_2 < 0$, return 1.
 - e. Else if $t_1 t_2 > 0$, return 0.
 - f. Else, return **Unknown**.
-

457 **Remarks for the algorithm:**

- 458 1. $A = 1$ means that the system F has a unique zero in B , $A = 0$ means that the system
459 F has no zero in B .
- 460 2. When the algorithm returns Unknown, it means that we do not find two systems
461 which have a unique zero in their corresponding boxes or can not determine the sign
462 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
463 to) a k -dimensional ($k \leq n - 2$) the boundaries of B , so that our precision ϵ_b can not
464 handle this case. This also explains why there are at most two $(n - 1)$ -dimensional
465 faces of B intersects S in Step 4. We can avoid this case by changing the length of
466 the box or combining some adjacent boxes with the same output “Unknown” to form
467 a new box. Subdividing the new box and checking the conditions again, one usually
468 succeeds in finding the results. The **Existence** for bivariate systems are presented in
469 [14].
- 470 3. In Steps 3.b, 3.c, refining the boxes B_i, B_j means to refine the root of the systems F'
471 for $x_i = t, x_j = t'$ in the boxes to get smaller boxes. Notice that we know there exists
472 and only exists one unique real zero in the related boxes. We can use interval-Newton
473 method for the refinement. $B_i(B_j)$ or part of it can be set as the original box for the
474 iteration.
- 475 4. The correctness and termination of the algorithm is based on theories before.

476 3.6 Choosing a proper S-M matrix V

477 A better S-M matrix helps us reducing some unnecessary computation. In this subsec-
478 tion, we will discuss how to choose a “better” S-M matrix V . As mentioned before, we
479 may meet some bad cases: although both $F'|_{x_i=t}$ and $F'|_{x_j=t'}$ have a unique zero in B_i and
480 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
481 refine the boxes B_i and B_j . Therefore, we want to select some “nice” S-M matrices V to
482 avoid bad cases as much as possible. See the following example first.

483 **Example 6.** Consider the example in [14]. Let $F = (y - x^2, x - 2y)$, $B = [-0.1, 0.1] \times$
484 $[-0.1, 0.1]$. Let $V_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and $V_2 = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$. We have $G_1 = V_1 J_F^{-1}(m(B)) \cdot F^T =$
485 $(-3x^2 + x + y, -x^2 + x - y)^T = (g_1^{(1)}, g_2^{(1)})^T$ and $G_2 = V_2 J_F^{-1}(m(B)) \cdot F^T = (-5x^2 + 2x +$
486 $y, x - 2y)^T = (g_1^{(2)}, g_2^{(2)})^T$. It is easy to check that G_1 and G_2 are both S-M systems in
487 B . Next we consider the existence condition. For the system G_1 , $g_1^{(1)}|_{x=0.1}$ has a unique
488 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
489 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
490 $g_1^{(2)}|_{y=-0.1}$ 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
491 $g_2^{(2)}|_{y=-0.1}(v(B_2)) > 0$ immediately, thus we know the existence without refinement, see the
492 right figure in Figure 3. It means that V_2 is “better” than V_1 .

493 Give a system $F = (f_1, \dots, f_n)$ and a box B , we assume that B is an isolating box of F
494 (i.e., B contains only one simple zero \mathbf{p} of F). Let $G = V J_F^{-1}(m(B)) \cdot F^T = (g_1, \dots, g_n)^T$
495 and $G' = (g_1, \dots, g_{n-1})$. Our goal is to choose an S-M matrix V such that $S = \mathbb{V}(G')$
496 goes through $F_n^l(B)$ and $F_n^r(B)$, i.e., both $G'|_{x_n=a_n}$ and $G'|_{x_n=b_n}$ have a unique zero in
497 B_n , meanwhile, $g_n|_{x_n=a_n}(v(B_n)) \cdot g_n|_{x_n=b_n}(v(B_n)) < 0$. Then we immediately know B is
498 an isolating box of F . Notice that when $w(B)$ is small, i.e., $m(B)$ is close to the simple

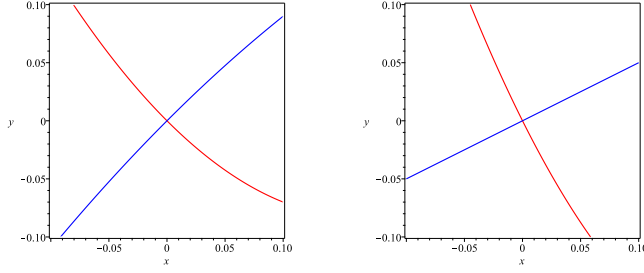


Figure 3: The influence of the different S-M matrixes to the same system.

499 zero $\mathbf{p} = (p_1, p_2, \dots, p_n)$, we know that $J_F^{-1}(m(B)) \cdot F^T \approx (x_1 - p_1, \dots, x_n - p_n)^T$ and
 500 $G \approx V \cdot (x_1 - p_1, \dots, x_n - p_n)^T$ in B . That is to say g_i is approximately a hyperplane in B
 501 and $\nabla g_i(m(B)) \approx V_{i,\cdot}$ for $i = 1, \dots, n$.

502 Based on the analysis above, we assume that $\mathbf{p} = \mathbf{0}$, $F = (x_1 + h_1, \dots, x_n + h_n)$ and
 503 $B = [-1, 1]^n$ is a unit box, where $h_i \in C^1(B)$ and the Taylor expansion of h_i at $\mathbf{p} = \mathbf{0}$ has
 504 only terms with degree greater than 1, $i = 1, \dots, n$. Then we show how to choose V such
 505 that the system $G = (g_1, \dots, g_n)^T = V \cdot F^T$ satisfying the following conditions:

- 506 (1) V is an S-M matrix.
 507 (2) $(g_1, \dots, g_{n-1})|_{x_n=a_n}$ and $(g_1, \dots, g_{n-1})|_{x_n=b_n}$ have a unique zero in $F_n^l(B)$ and $F_n^r(B)$.
 508 (3) $\text{Sign}(g_n(F_n^l(B))) \cdot \text{Sign}(g_n(F_n^r(B))) < 0$.

509 For example, $V_{2d} = \begin{pmatrix} N & 1 \\ 1 & -N \end{pmatrix}$ for $N = 2$ or a larger positive integer and $V_{3d} =$
 510 $\begin{pmatrix} N & 1 & 1 \\ 1 & -N & 1 \\ 1 & 1 & N \end{pmatrix}$ for $N = 3$ or a larger positive integer satisfy our conditions, see Figure 4.

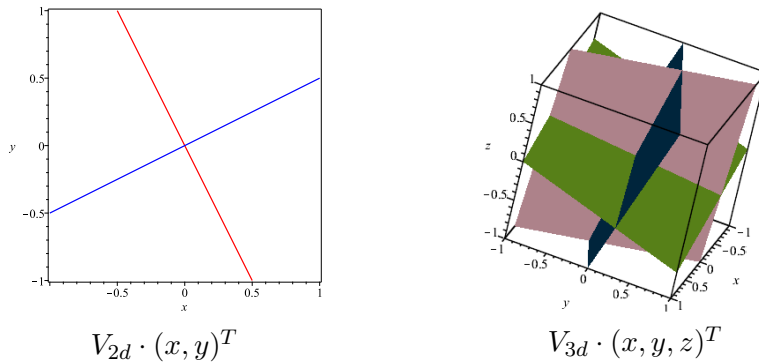


Figure 4: Examples for choosing V .

511 For general cases,

$$V_{nd} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{pmatrix}.$$

512 Let $\alpha_{ii} = (-1)^{i+1}N$ and $|\alpha_{ij}| \leq 1, i \neq j$, by generating α_{ij} randomly, we can find an S-M
513 matrix V_{nd} satisfies the above conditions with high probability, where N can be n or another
514 integer larger than n . We can also choose α_{ii} for different sign(s) for more choices. Usually,
515 it is easier to get an S-M matrix for $\frac{\partial G}{\partial X}(B)$ when N is larger. This phenomenon can be
516 obtained by observing the matrix of $\frac{\partial G}{\partial X}(B)$. We can also check whether it is a matrix we
517 want by Definition 5 and we ensure that it is an S-M matrix.

518 3.7 Algorithm

519 In this subsection, we give the main algorithm for real zero isolation of real nonlinear
520 systems. Our method is a subdivision method, we need the exclusion test which is based
521 on the following famous box predicate $C_0(f, B)$ [46]:

$$C_0(f, B) := 0 \notin \square f(B). \quad (2)$$

522 Given a function f and a box B , we say $C_0(f, B)$ is true if $0 \notin \square f(B)$. Based on the box
523 predicate, we can write it as an algorithm $A = \mathbf{Exclusion}(F, B)$. Obviously, if the algorithm
524 **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.

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

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

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

525 Isolating the real zeros of $f_1(\mathbf{b}, x_n)$ in I_n (see [14] for details), we can get a list of intervals,
526 say J_1, \dots, J_m . Continuing to isolate the real zeros of $f_2(\mathbf{b}, x_n)$ in J_1, \dots, J_m , we can
527 get a list of intervals, say $J'_1, \dots, J'_{m'}$, or an empty set. If we get an empty set, the box
528 $\mathbf{b} \times I_n$ can be thrown away. Else, doing so for f_3, \dots, f_n , we can get a list of intervals, say
529 $\bar{J}_1, \dots, \bar{J}_{m''}$ or an empty set. Then we can get candidate boxes $\mathbf{b} \times \bar{J}_k (1 \leq k \leq m'')$ or
530 throwing away the box $\mathbf{b} \times I_n$. Thus we exclude some sub-boxes of B . We can recursively
531 do so on the boxes to exclude some sub-domain of B . When I_n is large, this method may
532 be more efficient to compute the possible candidate boxes than the method in (2). In
533 fact, this method also works for non-polynomial systems. But the way to construct the
534 upper (lower) bounding function is a little more complicated than polynomial case and
535 we need to isolate the real zeros of non-polynomial univariate equations. We denote it
536 as an algorithm $A = \mathbf{Candidate}(F, B)$. Our implementation uses mainly this method for
537 polynomial systems.

538 Based on the discussion above, we have the main algorithm **Realrootfinding**.

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

1. **Input:** A system $F = (f_1, \dots, f_n)$, an initial box B_0 , a termination precision $\epsilon > 0$.
 2. **Output:** An isolating box set \mathbf{R} and a suspected root box set \mathbf{SR} .
1. $\mathbf{R} \leftarrow \emptyset, \mathbf{SR} \leftarrow \emptyset$ and $\mathbf{BS} \leftarrow \{B_0\}$.
 2. While $\mathbf{BS} \neq \emptyset$
 - (1) Take $B \in \mathbf{BS}$, $\mathbf{BS} \leftarrow \mathbf{BS} \setminus \{B\}$, $A \leftarrow \mathbf{Exclusion}(F, B)$.
 - (2) If $A = 0$,
 - (a) $G \leftarrow V_{nd} J_F^{-1}(m(B)) \cdot F^T$.
 - (b) $A' \leftarrow \mathbf{IsSMSys}(G, B)$.
 - (c) If $A' = 1$,
 - $A'' \leftarrow \mathbf{Existence}(G, B, \epsilon)$.
 - If $A'' = 1$, $\mathbf{R} \leftarrow \mathbf{R} \cup \{B\}$.
 - If $A'' = \mathbf{Unknown}$, $\mathbf{SR} \leftarrow \mathbf{SR} \cup \{B\}$.
 - (d) Else if $w(B) > \epsilon$,
 - Split B into two similar parts and add them into \mathbf{BS} .
 - (e) Else, $\mathbf{SR} \leftarrow \mathbf{SR} \cup \{B\}$.
 3. Return $\mathbf{R, SR}$.
-

539 The correctness of the algorithm **Realrootfinding** is guaranteed by Theorem 6 and
540 Lemma 13. The termination of the algorithm is guaranteed by Theorem 11 and the given
541 ϵ .

Remarks for the Realrootfinding algorithm:

542
543 1. For a system F with only simple zeros, we can always get all the isolating boxes of
544 all the zeros of F in B_0 by recursively subdividing those suspected root boxes. Our theories
545 ensure the termination of the algorithm.

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

551 3. In order to get all the real zeros of a given system in a general given real box,
552 we consider the coordinate transformation: $x_i \rightarrow \frac{1}{x_i}$. We map the interval $[-b, -1]$ to
553 $[-1, -1/b]$, and the interval $[1, b]$ to $[1/b, 1]$, where $b > 1$. Hence, we need only to consider
554 finding real zeros in $[-1, 1]^n$. Doing this way, we take only interval computation inside
555 $[-1, 1]^n$ which need less interval evaluation. If the given system has only finite real zeros
556 in a bounded box, we can get all its real zeros in the whole real space. For example, for a
557 bivariate system $F = (f_1(x, y), f_2(x, y))$, we can get the isolating boxes or suspected boxes
558 of real zeros of the original system in $[1, b] \times [-1, 1]$ by isolating the real zeros of the system
559 $(f_1(\frac{1}{x}, y), f_2(\frac{1}{x}, y))$ in $[1/b, 1] \times [-1, 1]$. If F is a polynomial system, then we can get all the
560 real zeros of F if we take b as its root bound.

561 4. In Step 2.(2) (d), we can find that some regions in B may be computed for several

562 times which waste much computing time. Thus we can split the given box B_0 into many
 563 smaller boxes at first.

564 5. In Step 2.(1), our aim is to find the candidate regions which may contain real zeros
 565 of the given system. For the case that the give system is a polynomial system, we can
 566 compute candidate regions as below (see more details in [12, 13, 14]). Given a system
 567 $F = (f_1, \dots, f_n)$ and a box $B = I_1 \times \dots \times I_n$, let $J = I_2 \times \dots \times I_n \subset \mathbb{R}^{n-1}$. We denote T_i as
 568 the set of the real (interval) zeros of the interval polynomial $f_i(x_1, J)$ for $i = 1, \dots, n$, and
 569 let $\{t_1 \cap \dots \cap t_n \cap I_1 | t_i \in T_i, i = 1, \dots, n\} = \{I'_1, \dots, I'_m\}$. We call $\{I'_1 \times J, \dots, I'_m \times J\}$ the
 570 **candidates** of F in B . It is obvious that all the real zeros of F in B are in the candidates. If
 571 the width $w(J)$ is large, we can split J into $(n-1)^k$ parts equally (by splitting $I_i, i = 2, \dots, n$
 572 into k parts equally) and compute the candidates separately.

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

592 3.8 Complexity analysis

593 We analyze the bit complexity of isolating the real roots of a zero-dimensional polynomial
 594 system Σ in this subsection. We assume that there are n variables and the degrees of the
 595 polynomials of the system are bounded by d , the number of their terms are bounded by m
 596 and the bit sizes of their coefficients are bounded by τ .

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

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

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

607 Consider the real roots of Σ inside $B = [-1, 1]^n$. Notice that if we want to get all the

608 real zeros of Σ , we can transform the original system into new system by replacing x_i with
609 $1/x_i$ and removing the denominators of the whole polynomials. We can get 2^n this kind of
610 systems in $[-1, 1]^n$. We assume that the termination precision for the boxes is ρ , that is, we
611 stop subdividing the boxes when their lengths are less than ρ , where ρ is a rational number
612 such that $0 < \rho < 1$. So the number of the boxes in B is bounded by $(\frac{2}{\rho})^n$. The bitsize of
613 the endpoints of the boxes is bounded by $-\log(\rho)$.

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

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

620 *Proof.* Consider one term of g at first. The operations here include the multiplications of
621 d intervals at most and one multiplication between the coefficient and the product of the
622 intervals. We can divide d intervals into $\frac{d}{2}$ pairs. Each pair usually contains two intervals. If
623 d is odd, the last pair contains only one interval. To get the product of them includes at most
624 4 multiplications. The total bit complexity for computing all the pairs is $\frac{d}{2} * 4 * \sigma = 2d\sigma$. The
625 bitsizes of the products are 2σ . For these $\frac{d}{2}$ products, we divide them into $\frac{d}{4}$ pairs. Similarly,
626 the total bit complexity for computing the products of all the pairs is $\frac{d}{4} * 4 * 2\sigma = 2d\sigma$. The
627 bitsizes of the products are 4σ . Doing so in a similar way until the k step such that $d = 2^k$,
628 that is, $k = \log(d)$, we can get the product of all the intervals. So to get the product of
629 all the intervals, we have the total bit complexity $2d\sigma * \log(d)$ and the bitsize of the final
630 product is $2^{\log(d)}\sigma = d\sigma$. Considering the coefficient into the product, we have the total
631 bit complexity of evaluating one term is $2d\sigma * \log(d) + d\sigma + \tau = \tilde{O}(dm\sigma + m\tau)$. The bitsize
632 of the final product for one term is $d\sigma + \tau$. So evaluating $g(I_1, \dots, I_n)$ has a complexity of
633 $\tilde{O}(dm\sigma + m\tau)$, and the bitsize of the endpoints of $g(I_1, \dots, I_n)$ is $\mathcal{O}(d\sigma + \tau)$. ■

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

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

642 *Proof.* We can divide n intervals into $\frac{n}{2}$ pairs. Each pair is two integers. Note that if n is
643 odd, the last one can be regarded as a pair. For each pair, we compute the least common
644 multiple of the two integers a, b . It is $ab/\gcd(a, b)$. The bit complexity is $2\tilde{O}(\sigma)$. So the
645 bit complexity of computing all the pairs is $n\tilde{O}(\sigma)$. The bit sizes of the results are all
646 2σ . We continue to divide the results into $\frac{n}{4}$ pairs. For each pair we compute its least
647 common multiple with bit complexity $4\tilde{O}(\sigma)$. So the bit complexity of computing all the
648 pairs is $n\tilde{O}(\sigma)$. And the bit sizes of the results are 4σ . Doing so, until we get the least
649 common multiple of the n integers, which we need $\log(n)$ steps. So the total bit complexity
650 is $\log(n)\tilde{O}(n\sigma) = \tilde{O}(n\sigma)$. The bit size of the last least common multiple is $2^{\log(n)}\sigma = n\sigma$. ■

651 For a nonsingular matrix M with rational entries such that the bit sizes of the entries
652 are bounded by σ , we can rewrite $M = M_1 M_2$, where M_1 is a diagonal matrix and M_2 is a
653 matrix with integer entries. We can compute the least common multiple of the denominators
654 of the elements of each row. Set its inverse as the element of the related row of M_1 . Each
655 of the related element of M of the row multiplies the least common multiple and set them
656 as the related elements of M_2 . The bit size of each element of M_2 is $n\sigma$, so as M_1 . By
657 Lemmas 15, 16, we can find that the bit complexity of computing the inverse of M is
658 $\tilde{\mathcal{O}}(n^4\sigma + n^3 \log(\kappa(\|M_2\|)))$ (a well conditioned one is $\tilde{\mathcal{O}}(n^4\sigma)$) and the bit sizes of the
659 elements of $M^{-1} = M_2^{-1}M_1^{-1}$ is $2n\sigma$.

660 Now we consider the complexity to check the existence and the uniqueness of a real root
661 inside a box. When we compute the intersection between the space curve formed by the
662 $n - 1$ functions and the boundaries of a box. We need to check each face of the box to
663 intersect the space curve. There are $2n$ faces. Each face is related to a zero-dimensional
664 system with $n - 1$ functions and $n - 1$ variables. Recursively, we will do root finding of
665 $2^{n-1} n!$ univariate polynomials. In order to get an approximating root, we can bisect the
666 interval a fixed number times, say 10 times, if there is a root. We also need to multiply two
667 square matrices with order n : One is V , the other is $J_F^{-1}(m(B))$.

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

672 *Proof.* Denote the Jacobian matrix of F w.r.t. x_1, \dots, x_n as J_F . The bitsize of the middle
673 point of B , say \mathbf{P} , is σ . It is clear that the bitsize of each element of $J_F(\mathbf{P})$ is $\mathcal{O}(d\sigma + \tau)$.
674 The bit complexity to compute $J_F(\mathbf{P})$ is $\mathcal{O}(n^2 m d\sigma + n^2 m \tau)$ by Lemma 14. Thus the
675 bitsize 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
676 compute $J_F^{-1}(\mathbf{P})$ is $\mathcal{O}(n^5\sigma + n^4\tau)$ by the analysis below Lemma 16. The bit complexity to
677 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
678 mentioned before whose elements are with bitsize $\mathcal{O}(1)$. And the bitsize of the elements of
679 $V J_F^{-1}(\mathbf{P})$ is $\mathcal{O}(n d\sigma + n \tau)$.

Let $G = (g_1, \dots, g_n) = V J_F^{-1}(\mathbf{P}) F^T$. 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 bitsize of $\mathcal{O}(d\sigma + \tau)$
by Lemma 14. Notice that f_i and $\frac{\partial f_i}{\partial x_j}$ have the same bitsize 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

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

680 By Stirling's approximation, the total bit complexity to check the strong monotonous con-
681 dition is $\mathcal{O}(n^n)\mathcal{O}(n^3 d\sigma + n^3 \tau) = \mathcal{O}(n^{n+3} d\sigma + n^{n+3} \tau)$. Notice that the uniqueness condition
682 is already checked by the determinant of $J_G(B)$. To check the number of intersections be-
683 tween the space curve formed by g_1, \dots, g_{n-1} and the faces of B , we need to check whether
684 $2^n n!$ univariate polynomials with magnitude $(d, n d\sigma + n \tau)$ and at most $n m$ terms have so-
685 lutions in the related intervals. So for one root isolation on one interval, the bit complexity

686 is $\mathcal{O}(nm(d\sigma + nd\sigma + n\tau)) = \mathcal{O}(nm d\sigma + nm\tau)$ by Lemma 14. So the total bit complexity
 687 for the root isolation is $2^n n! \mathcal{O}(nm d\sigma + nm\tau) = \mathcal{O}(2^n n^{n+1} m (d\sigma + \tau))$. It is not difficult
 688 to find that this is the main part for the whole bit complexity when compared all the steps.
 689 Thus the total bit complexity of checking whether F contains a unique real root in B is
 690 $\mathcal{O}(2^n n^{n+1} m (d\sigma + \tau))$. ■

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

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

695 **Theorem 18.** *Isolating the real roots of a square system of polynomials with magnitude*
 696 *(d, τ) and m terms with the algorithm mentioned above, one takes a bit complexity of*
 697 $\mathcal{O}((\frac{8}{\rho})^n n^{n+1} m (-d \log(\rho) + \tau))$.

698 4 Experiments

699 We implement our algorithm in C++, and we do some experiments under Linux with
 700 a computer of 64 AMD 3990X 2.90GHz CPU and 64 GB RAM. We also realize the par-
 701 allel computation with MPI. Our code can be downloaded from [http://mmrc.iss.ac.cn/
 702 ~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/
 703 ~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
 704 Bertini [3]. There are two other famous homotopy continuation softwares: PHCpack [54]
 705 and Hom4ps [10]. We choose Bertini to compare just because it derives real roots directly.

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

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

Case	Terms	Coeff	Range	Roots			Times(seconds)		
				Ours	MK	Bertini	Ours	MK	Bertini
N2D5	10	10	-100~100	3.0/0/0	1.2/11.6	3.0	0.391	0.633	0.012
N2D5E	10	10	-1~1	3.0/0/0	1.0/10.6	-	0.212	0.324	-
N2D9	10	10	-100~100	6.2/49.8/0.2	1.8/77.4	6.4	7.148	8.232	0.222
N2D9E	10	10	-1~1	6.4/139.4/0	1.8/190.2	-	13.193	14.227	-
N2D51	10	10	-100~100	7.4/15.6/0	0.4/53.4	7.8	20.862	17.265	1140.89
N2D51E	10	10	-1~1	7.6/128.4/0	0.4/154.0	-	38.208	38.065	-
N2D101	10	10	-100~100	6.0/390.0/3.0	0.4/421.0	-	137.141	140.961	>10000
N2D101E	10	10	-1~1	5.6/274.4/1.2	0.2/290.2	-	91.37	97.727	>10000
N3D9	10	10	-100~100	12.6/1493.4/0.2	0/2348.6	13.2	721.023	798.912	5.891
N3D9E	10	10	-1~1	12.8/519.0/0	0/1067.2	-	173.859	207.422	-
N3D51	10	10	-2~2	5.2/89.4/0	-	-	45.913	-	>10000
N3D101	10	10	-2~2	8.0/44.2/0	-	-	4079.618	-	>10000
N4D9	10	10	-2~2	9.8/0/0	-	-	338.795	-	>10000
N4D51	10	10	-1~1	2.8/0/0	-	-	131.885	-	>10000
N4D101	10	10	-1~1	1.0/0/0	-	-	72.618	-	>10000
N5D11	10	10	-1~1	1.2/0/0	-	-	322.770	-	>10000
N6D11	10	10	-1~1	1.6/0/0	-	-	8071.526	-	>10000

Table 1: Comparing our method with MK and Bertini for polynomial systems.

729 Remarks for Table 1:

- 730 1. In case N2D51, one example has 7 roots and we get 5 of them and miss finding the
731 other 2 from the suspected boxes. In the related case N2D51E, we directly find 7
732 certified roots for the example but another example miss one root from the suspected
733 boxes.
- 734 2. In case N3D9, the number of the total average roots is 13.4. Bertini misses one root,
735 one of whose coordinate is around 3,000,000. Our code finds all the real roots of the
736 systems inside the box $[-100, 100]^3$. There are totally 3 roots out of the box.
- 737 3. In case N3D101, the computing times for 3 of the 5 systems are less than 100 seconds.
738 But one example takes 20157 seconds and there are 11 certified solutions and no
739 suspected boxes. We find that several roots are very close to each other. One example
740 has 17 certified roots and 162 suspected boxes, which takes 112.398 seconds.
- 741 4. In Table 1, the examples above N3D51 does not use parallel computing and the results
742 are proved by symbolic computation. From N3D51, we use MPI parallel computing
743 with 30 cores and the results are without proof with symbolic computation.

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 $[-1, 1]^3$.

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

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

System $[g_1, g_2, g_3], g_i =$	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
$f_i * 2^{10}$	45.397	48.985	0.636	4.8/138.2/0	0/262.4	5.0
$f_i * 2^{50}$	45.362	49.254	~	4.8/138.2/0	0/262.4	~
$f_i * 2^{100}$	45.464	49.302	~	4.8/138.2/0	0/262.4	~
$f_i * 2^{200}$	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.

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

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

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

773 We check our implementation of parallel computing with MPI on N4D9 in Table 1. We
 774 use 4, 8, 16, 32, 60 cores to compute the 5 systems and get the average computing time.
 775 The related data is given in Table 7. It shows the speedup of the parallel computing of our

# roots	Times		Roots	
	Bertini	our method	Bertini	our method
8	1.100	42.839	8	8/0/0
16	0.900	61.049	16	16/0/0
24	1.050	114.557	24	24/0/0
32	1.180	301.210	32	32/0/0
40	0.900	1082.670	40	40/728/0
48	1.280	982.576	48	48/536/0

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 $(25x^2 - 2)(25y^2 - 11)(25z^2 - 5)$, $(25x^2 - 11)(25y^2 - 5)(25z^2 - 3)$, $(25x^2 - 5)(25y^2 - 2)(25z^2 - 11)$ respectively. All the roots are inside $[-1, 1]^3$.

Case	Exact roots	Roots		Times	
		Ours	Bertini	Ours	Bertini
multiN2D6	3.8	2.4/8665/1.4	6.8	334.389	0.09
multiN2D12	6.4	2.6/16535/3.8	9.8	6149.520	1.518
multiN3D6	6.4	2.2/6744.2/4.2	6.4	264.546	2.446
multiN3D12	6.6	1.0/16779.2/5.8	6.6	3115.749	1929.558

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: $f_1 = f * g$ 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: $f_1 = f * g$, $f_2 = \frac{\partial f_1}{\partial x_1}$ and f_3 is a randomly generated polynomial.

ρ	Certified roots	Suspected boxes	Refined roots	Times
10^{-2}	1/3/2/3/1	31/106/91/87/44	2/6/5/2/7	1.735/4.346/4.671/3.916/3.192
10^{-4}	3/8/6/4/8	14/57/128/106/8	0/1/1/1/0	2.737/8.640/8.849/9.847/4.345
10^{-6}	3/9/6/5/8	0/26/123/100/0	0/0/1/0/0	3.014/11.083/14.595/16.017/4.446
10^{-8}	3/9/7/5/8	0/0/3/0/0	0/0/0/0/0	3.031/11.389/18.164/17.942/4.335
10^{-10}	3/9/7/5/8	0/0/0/0/0	0/0/0/0/0	3.049/11.390/18.332/18.153/4.429

Table 6: The influence of the termination precisions to the 5 systems in N2D9 in Table 1.

code.

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.

776

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

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

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

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

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

815 but there are 12 variables in [28, 29]. The following is the system given in [28, 29].

$$\begin{aligned}
& s2 c5 s6 - s3 c5 s6 - s4 c5 s6 + c2 c6 + c3 c6 + c4 c6 - .4077, \\
& c1 c2 s5 + c1 c3 s5 + c1 c4 s5 + s1 c5 - 1.9115, \\
& s2 s5 + s3 s5 + s4 s5 - 1.9791, \\
& 3 c1 c2 + 2 c1 c3 + c1 c4 - 4.0616, \\
& 3 s1 c2 + 2 s1 c3 + s1 c4 - 1.7172, \\
& 3 s2 + 2 s3 + s4 - 3.9701, \\
& si^2 + ci^2 - 1, i = 1, \dots, 6.
\end{aligned} \tag{3}$$

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

[[0.0625, 0.0644531], [0.0957031, 0.0976562], [0.148438, 0.150391], [0.107422, 0.109375], [0.277344, 0.279297], [0.230469, 0.232422]],
[[0.0625, 0.0644531], [0.128906, 0.130859], [0.0820312, 0.0839844], [0.140625, 0.142578], [0.275391, 0.277344], [0.224609, 0.226562]],
[[0.0625, 0.0644531], [0.0957031, 0.0976562], [0.148438, 0.150391], [0.107422, 0.109375], [0.277344, 0.279297], [0.787109, 0.789062]],
[[0.0625, 0.0644531], [0.128906, 0.130859], [0.0820312, 0.0839844], [0.140625, 0.142578], [0.275391, 0.277344], [0.783203, 0.785156]],
[[0.560547, 0.5625], [0.400391, 0.402344], [0.347656, 0.349609], [0.390625, 0.392578], [0.220703, 0.222656], [0.289062, 0.291016]],
[[0.560547, 0.5625], [0.400391, 0.402344], [0.347656, 0.349609], [0.390625, 0.392578], [0.220703, 0.222656], [0.728516, 0.730469]],
[[0.561523, 0.5625], [0.368164, 0.369141], [0.415039, 0.416016], [0.357422, 0.358398], [0.22168, 0.222656], [0.285156, 0.286133]],
[[0.561523, 0.5625], [0.390625, 0.391602], [0.394531, 0.395508], [0.329102, 0.330078], [0.304688, 0.305664], [0.257812, 0.258789]],
[[0.561523, 0.5625], [0.368164, 0.369141], [0.415039, 0.416016], [0.357422, 0.358398], [0.22168, 0.222656], [0.724609, 0.725586]],
[[0.561523, 0.5625], [0.390625, 0.391602], [0.394531, 0.395508], [0.329102, 0.330078], [0.304688, 0.305664], [0.693359, 0.694336]].

820 Furthermore, we get 4116 suspected boxes and find none roots from them. We miss 6 real
821 roots. The computing time is 843.994 seconds.

822 If we set $\rho = 10^{-6}$, we get exactly 16 certified roots and no suspected boxes which takes
823 2133.05 seconds:

[[0.0625, 0.0644531], [0.0957031, 0.0976562], [0.148438, 0.150391], [0.107422, 0.109375], [0.277344, 0.279297], [0.230469, 0.232422]],
[[0.0625, 0.0644531], [0.128906, 0.130859], [0.0820312, 0.0839844], [0.140625, 0.142578], [0.275391, 0.277344], [0.224609, 0.226562]],
[[0.0625, 0.0644531], [0.0957031, 0.0976562], [0.148438, 0.150391], [0.107422, 0.109375], [0.277344, 0.279297], [0.787109, 0.789062]],
[[0.0625, 0.0644531], [0.128906, 0.130859], [0.0820312, 0.0839844], [0.140625, 0.142578], [0.275391, 0.277344], [0.783203, 0.785156]],
[[0.560547, 0.5625], [0.400391, 0.402344], [0.347656, 0.349609], [0.390625, 0.392578], [0.220703, 0.222656], [0.289062, 0.291016]],
[[0.560547, 0.5625], [0.400391, 0.402344], [0.347656, 0.349609], [0.390625, 0.392578], [0.220703, 0.222656], [0.728516, 0.730469]],
[[0.561523, 0.5625], [0.368164, 0.369141], [0.415039, 0.416016], [0.357422, 0.358398], [0.22168, 0.222656], [0.285156, 0.286133]],
[[0.561523, 0.5625], [0.390625, 0.391602], [0.394531, 0.395508], [0.329102, 0.330078], [0.304688, 0.305664], [0.257812, 0.258789]],
[[0.561523, 0.5625], [0.368164, 0.369141], [0.415039, 0.416016], [0.357422, 0.358398], [0.22168, 0.222656], [0.724609, 0.725586]],
[[0.561523, 0.5625], [0.390625, 0.391602], [0.394531, 0.395508], [0.329102, 0.330078], [0.304688, 0.305664], [0.693359, 0.694336]],
[[0.562012, 0.5625], [0.40332, 0.403809], [0.371582, 0.37207], [0.339844, 0.340332], [0.308105, 0.308594], [0.25293, 0.253418]],
[[0.562012, 0.5625], [0.40332, 0.403809], [0.371582, 0.37207], [0.339844, 0.340332], [0.308105, 0.308594], [0.688965, 0.689453]],
[[0.0634766, 0.0637207], [0.107178, 0.107422], [0.103516, 0.10376], [0.169189, 0.169434], [0.193115, 0.193359], [0.195068, 0.195312]],
[[0.0634766, 0.0637207], [0.0952148, 0.095459], [0.126953, 0.127197], [0.158691, 0.158936], [0.19043, 0.190674], [0.190674, 0.190918]],
[[0.0634766, 0.0637207], [0.107178, 0.107422], [0.103516, 0.10376], [0.169189, 0.169434], [0.193115, 0.193359], [0.756348, 0.756592]],
[[0.0634766, 0.0637207], [0.0952148, 0.095459], [0.126953, 0.127197], [0.158691, 0.158936], [0.19043, 0.190674], [0.751465, 0.751709]].

824

825 **Example 8.** The following problem is the inverse position problem for a six-revolute-joint
826 problem in mechanics, one can find it in [28, 43]. The defining equations in the reference
827 are as below.

$$\begin{aligned}
& a_{i3}x_2 x_3 + a_{i4}x_2 x_4 + a_{i1}x_1 x_3 + a_{i2}x_1 x_4 + a_{i5}x_5 x_7 + a_{i6}x_5 x_8 + a_{i7}x_6 x_7 + a_{i8}x_6 x_8 \\
& + a_{i9}x_1 + a_{i10}x_2 + a_{i11}x_3 + a_{i12}x_4 + a_{i13}x_5 + a_{i14}x_6 + a_{i15}x_7 + a_{i16}x_8 + a_{i17}, \\
& x_i^2 + x_{i+1}^2 - 1, i = 1, 3, 5, 7.
\end{aligned}$$

828 There are 8 equations and 8 variables. The values of $a_{i,j}, j = 1, \dots, 17$ can be found in
829 [28]. We replace x_i, x_{i+1} with $\sin(6.3 y_{\frac{i+1}{2}}), \cos(6.3 y_{\frac{i+1}{2}})$ for $i = 1, 3, 5, 7$ in the first equation

830 *above. Then we can get a system with 4 equations and 4 variables. Computing its real roots*
 831 *in $[0, 1]^4$ with precision $\rho = 10^{-3}$ and 32 cores with MPI parallel computing, we get 9 certified*
 832 *roots:*

[[0.199219, 0.203125], [0.195312, 0.199219], [0.703125, 0.707031], [0.167969, 0.171875]],
 [[0.175781, 0.177734], [0.228516, 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]],
 [[0.712891, 0.714844], [0.728516, 0.730469], [0.746094, 0.748047], [0.257812, 0.259766]],
 [[0.0722656, 0.0732422], [0.379883, 0.380859], [0.566406, 0.567383], [0.387695, 0.388672]],
 [[0.21875, 0.219727], [0.960938, 0.961914], [0.0576172, 0.0585938], [0.513672, 0.514648]].

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

835 *If we set $\rho = 10^{-6}$, we get exactly 12 certified roots and no suspected boxes which takes*
 836 *902.642 seconds:*

[[0.199219, 0.203125], [0.195312, 0.199219], [0.703125, 0.707031], [0.167969, 0.171875]],
 [[0.175781, 0.177734], [0.228516, 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]],
 [[0.712891, 0.714844], [0.728516, 0.730469], [0.746094, 0.748047], [0.257812, 0.259766]],
 [[0.0722656, 0.0732422], [0.379883, 0.380859], [0.566406, 0.567383], [0.387695, 0.388672]],
 [[0.21875, 0.219727], [0.960938, 0.961914], [0.0576172, 0.0585938], [0.513672, 0.514648]],
 [[0.0625, 0.0629883], [0.385742, 0.38623], [0.567871, 0.568359], [0.391602, 0.39209]],
 [[0.0993652, 0.0996094], [0.712646, 0.712891], [0.292236, 0.29248], [0.515625, 0.515869]],
 [[0.656242, 0.65625], [0.773285, 0.773293], [0.321335, 0.321342], [0.549934, 0.549942]].

837

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

840 5 Conclusion

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

References

- 857
- 858 [1] O. Aberth. *Introduction to precise numerical methods*. Elsevier/Academic Press, Am-
859 sterdam, 2nd edition, 2007.
- 860 [2] D.S. Arnon, G.E. Collins, and S. McCallum. Cylindrical algebraic decomposition i:
861 The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, 1984.
- 862 [3] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. *Numerically solving*
863 *polynomial systems with Bertini*. Software, Environments, and Tools 25, SIAM, 2013.
- 864 [4] C. Beltran and A. Leykin. Certified numerical homotopy tracking. *Experimental Math-*
865 *ematics*, 21(1):69–83, 2012.
- 866 [5] C. Beltran and A. Leykin. Robust certified numerical homotopy tracking. *Found.*
867 *Comput. Math.*, 13:253–295, 2013.
- 868 [6] B. Buchberger. Bruno buchberger’s phd thesis 1965: An algorithm for finding the basis
869 elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of*
870 *Symbolic Computation*, 41(3-4):475–511, 2006.
- 871 [7] M. Burr, S. W. Choi, B. Galehouse, and C. Yap. Complete subdivision algorithms,
872 ii: Isotopic meshing of singular algebraic curves. *Journal of Symbolic Computation*,
873 47(2):131–152, 2012.
- 874 [8] M. Burr, S. Gao, and E. Tsigaridas. The complexity of an adaptive subdivision method
875 for approximating real curves. In *Proceedings of ISSAC’2017*, pages 61–68. ACM, 2017.
- 876 [9] M. A. Burr, S. Gao, and E. P. Tsigaridas. The complexity of an adaptive subdivision
877 method for approximating real curves. In *ISSAC ’17*, pages 61–68. ACM, 2017.
- 878 [10] T. Chen, T.-L. Lee, and T.-Y. Li. Hom4ps-3: A parallel numerical solver for systems of
879 polynomial equations based on polyhedral homotopy continuation methods. In Hoon
880 Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 183–190,
881 Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- 882 [11] J. S. Cheng, X. Dou, and J. Wen. A new deflation method for verifying the isolated sin-
883 gular zeros of polynomial systems. *Journal of Computational and Applied Mathematics*,
884 376:112825, 2020.
- 885 [12] J. S. Cheng, X. S. Gao, and C. K. Yap. Complete numerical isolation of real roots in
886 zero-dimensional triangular systems. *Journal of Symbolic Computation*, 44(7):768–785,
887 2009.
- 888 [13] J. S. Cheng and K. Jin. A generic position based method for real root isolation of
889 zero-dimensional polynomial systems. *Journal of Symbolic Computation*, 68:204–224,
890 2015.
- 891 [14] J. S. Cheng and J. Wen. Certified numerical real root isolation for bivariate polynomial
892 systems. In *Proceedings of the 2019 on International Symposium on Symbolic and*
893 *Algebraic Computation*, ISSAC ’19, pages 90–97, New York, NY, USA, 2019. ACM.

- 894 [15] J.-S. Cheng, J. Wen, and B. Zhang. Certified numerical real root isolation for bivariate
895 nonlinear systems. *Journal of Symbolic Computation*, 114:149–171, 2023.
- 896 [16] D. Cox, J. Little, and D. O’Shea. *Ideals, Varieties, and Algorithms: An Introduction to*
897 *Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts
898 in Mathematics. Springer, 1996.
- 899 [17] F. Cucker, E. Alperen A., and J. Tonelli-Cueto. Plantinga-vegter algorithm takes
900 average polynomial time. Proceedings of the International Symposium on Symbolic
901 and Algebraic Computation, ISSAC, pages 114–121. ACM, July 2019.
- 902 [18] F. Cucker, T. Krick, G. Malajovich, and M. Wschebor. A numerical algorithm for zero
903 counting, i: Complexity and accuracy. *Journal of Complexity*, 24(5):582–605, 2008.
- 904 [19] F. Cucker, T. Krick, G. Malajovich, and M. Wschebor. A numerical algorithm for zero
905 counting. ii: Distance to ill-posedness and smoothed analysis. *J. Fixed Point Theory*
906 *Appl.*, 6:285–294, 2009.
- 907 [20] F. Cucker, T. Krick, G. Malajovich, and M. Wschebor. A numerical algorithm for
908 zero counting. iii: Randomization and condition. *Advances in Applied Mathematics*,
909 48(1):215–248, 2012.
- 910 [21] B. H. Dayton, T. Y. Li, and Z. Zeng. Multiple zeros of nonlinear systems. *Mathematics*
911 *of Computation*, 80(276):2143–2168, 2011.
- 912 [22] J. Van der Hoeven. Reliable homotopy continuation. *hal-00589948v3*, 2011.
- 913 [23] JR. C.H. Edwards. *Advanced Calculus of Several Variables*. Academic Press, 1973.
- 914 [24] J. Garloff and A.P. Smith. Investigation of a subdivision based algorithm for solving
915 systems of polynomial equations. *Nonlinear Analysis: Theory, Methods and Applica-*
916 *tions*, 47(1):167 – 178, 2001.
- 917 [25] J. Garloff and A.P. Smith. Solution of systems of polynomial equations by using
918 bernstein expansion. In *Symbolic Algebraic Methods and Verification Methods*, pages
919 87–97. Springer, 2001.
- 920 [26] I. Gelfand, M. Kapranov, and A. Zelevinsky. *Discriminants, resultants, and multidi-*
921 *mensional determinants*. Springer Science & Business Media, 2008.
- 922 [27] Iddo Hanneil and Gershon Elber. Subdivision termination criteria in subdivision
923 multivariate solvers using dual hyperplanes representations. *Computer-Aided Design*,
924 39(5):369–378, 2007. Geometric Modeling and Processing 2006.
- 925 [28] P. Van Hentenryck, D. Mcallester, and D. Kapur. Solving polynomial systems using a
926 branch and prune approach. *SIAM Journal on Numerical Analysis*, 34:797–827, 1997.
- 927 [29] H. Hong and V. Stahl. Safe starting regions by fixed points and tightening. *Computing*,
928 53(3-4):323–335, 1994.
- 929 [30] J. Horacek, M. Hladik, and J. Matejka. Determinants of interval matrices. *Electronic*
930 *Journal of Linear Algebra*, 33:99–112, 2018.

- 931 [31] R. Imbach. A subdivision solver for systems of large dense polynomials. *arX-*
932 *iv:1603.07916*, 2016.
- 933 [32] R. Baker Kearfott. *Rigorous Global Search: Continuous Problems*. Springer, Boston,
934 MA, 1996.
- 935 [33] J. B. Kioustelidis. Algorithmic error estimation for approximate solutions of nonlinear
936 systems of equations. *Computing*, 19(4):313–320, 1978.
- 937 [34] R. Krawczyk. Newton-algorithmen zur bestimmung von nullstellen mit fehlerschranken.
938 *Computing*, 4(3):187–201, 1969.
- 939 [35] Anton Leykin, Jan Verschelde, and Ailing Zhao. Newton’s method with deflation for
940 isolated singularities of polynomial systems. *Theor. Comput. Sci.*, 359:111–122, 2006.
- 941 [36] T.-Y. Li, T. Sauer, and J. A. Yorke. Numerically determining solutions of systems of
942 polynomial equations. *Bulletin of the American Mathematical Society*, 18(2):173–177,
943 1988.
- 944 [37] J. M. Lien, V. Sharma, G. Vegter, and C. Yap. Isotopic arrangement of simple curves:
945 An exact numerical approach based on subdivision. In *ICMS 2014*, pages 277–282.
946 Springer, 2014. LNCS No. 8592. Download from <http://cs.nyu.edu/exact/papers/> for
947 version with Appendices and details on MK Test.
- 948 [38] A. Mantzaflaris, B. Mourrain, and E. Tsigaridas. On continued fraction expansion
949 of real roots of polynomial systems, complexity and condition numbers. *Theoretical*
950 *Computer Science*, 412(22):2312–2330, 2011.
- 951 [39] C. Miranda. *Un’osservazione su un teorema di Brouwer*. Consiglio Nazionale delle
952 Ricerche, 1940.
- 953 [40] R. E. Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.
- 954 [41] R. E. Moore. A test for existence of solutions to nonlinear systems. *SIAM Journal on*
955 *Numerical Analysis*, 14(4):611–615, 1977.
- 956 [42] R. E. Moore and J. B. Kioustelidis. A simple test for accuracy of approximate solutions
957 to nonlinear (or linear) systems. *SIAM Journal on Numerical Analysis*, 17(4):521–529,
958 1980.
- 959 [43] A. P. MORGAN. Computing all solutions to polynomial systems using homotopy
960 continuation. *Appl. Math. Comput.*, 24:115–138, 1987.
- 961 [44] B. Mourrain and J. P. Pavone. Subdivision methods for solving polynomial equations.
962 *Journal of Symbolic Computation*, 44(3):292–306, 2009.
- 963 [45] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press,
964 1991.
- 965 [46] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces.
966 In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry*
967 *processing*, pages 245–254. ACM, 2004.

- 968 [47] S. M. Rump. Solving algebraic problems with high accuracy. In *A new approach to*
969 *scientific computation*, pages 51–120. Elsevier, 1983.
- 970 [48] F. Shen, W. Wu, and B. Xia. Real root isolation of polynomial equations based on hy-
971 brid computation. In Ruyong Feng, Wen-shin Lee, and Yosuke Sato, editors, *Computer*
972 *Mathematics*, pages 375–396, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- 973 [49] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear
974 polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, 1993.
- 975 [50] S. Smale. Newton’s method estimates from data at one point. In *The merging of*
976 *disciplines: new directions in pure, applied, and computational mathematics*, pages
977 185–196. Springer, 1986.
- 978 [51] A. Storjohann. On the complexity of inverting integer and polynomial matrices. *Com-*
979 *put. Complex.*, 24:777–821, 2015.
- 980 [52] H. Susanto and N. Karjanto. Newton’s method’s basins of attraction revisited. *Ap-*
981 *plied Mathematics and Computation*, 215(3):1084–1090, 2009.
- 982 [53] S. Telen, M. Van Barel, and J. Verschelde. A robust numerical path tracking algo-
983 rithm for polynomial homotopy continuation. *SIAM Journal on Scientific Computing*,
984 42(6):A3610–A3637, 2020.
- 985 [54] J. Verschelde. Algorithm 795: Phcpack: a general-purpose solver for polynomial sys-
986 tems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25:251–
987 276, 1999.
- 988 [55] W. Wu. *Mathematics mechanization: mechanical geometry theorem-proving, mechani-*
989 *cal geometry problem-solving, and polynomial equations-solving*. Science Press Beijing,
990 2000.
- 991 [56] J. Xu, M. Burr, and C. Yap. An approach for certifying homotopy continuation paths:
992 Univariate case. ISSAC ’18, pages 399–406. ACM, 2018.
- 993 [57] J. Xu and C. Yap. Effective subdivision algorithm for isolating zeros of real systems of
994 equations, with complexity analysis. ISSAC ’19, pages 355–362. ACM, 2019.