VerifyRealRoots: A Matlab Package for Computing Verified Real Solutions of Polynomials Systems of Equations and Inequalities*

YANG Zhengfeng · ZHAO Hanrui · ZHI Lihong

DOI: 10.1007/s11424-023-1406-7
Received: 21 October 2021 / Revised: 26 December 2021

Abstract VerifyRealRoots is a Matlab package for computing and verifying real solutions of polynomial systems of equations and inequalities. It calls Bertini or MMCRSolver for finding approximate real solutions and then applies AINLSS to verify the existence of a regular solution of a polynomial system or applies AINLSS2 (AIVISS) to verify the existence of a double solution (a singular solution of an arbitrary multiplicity) of a slightly perturbed polynomial system.

Keywords Error bounds, polynomial system, real solutions, singular solutions, verification.

1 Introduction

VerifyRealRoots is a Matlab package for computing verified real solutions of polynomial systems of equations and inequalities. Let $f_1, \cdots, f_m, g_1, \cdots, g_s$ be polynomials in $\boldsymbol{R}[x_1, \cdots, x_n]$, and $\mathcal{S} \subset \boldsymbol{R}^n$ be the semi-algebraic set defined by $f_1, \cdots, f_m, g_1, \cdots, g_s$:

$$\begin{aligned}\mathcal{S} = \{&(a_1, \cdots, a_n) \in \boldsymbol{R}^n : f_i(a_1, \cdots, a_n) = 0,\ g_j(a_1, \cdots, a_n) > 0 \\ &\text{for } 1 \le i \le m, 1 \le j \le s\}.\end{aligned} \tag{1}$$

VerifyRealRoots aims to find at least one verified real solution for the semi-algebraic set $\mathcal{S}$ using hybrid symbolic and numeric methods.

YANG Zhengfeng · ZHAO Hanrui
Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, *China.* Email: zfyang@sei.ecnu.edu.cn; hrzhao@stu.ecnu.edu.cn.
ZHI Lihong
Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, *China; University of Chinese Academy of Sciences, Beijing* 100049, *China.* Email: lzhi@mmrc.iss.ac.cn.

*This research is supported by the National Key Research Project of China under Grant No. 2018YFA0306702, and the National Natural Science Foundation of China under Grant Nos. 12171159, 12071467 and 61772203, and Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

◇ *This paper was recommended for publication by Editor WU Wenyuan.*

Let $\boldsymbol{V} \subset \boldsymbol{C}^n$ be the algebraic variety defined by:

$$f_1(x_1, \cdots, x_n) = f_2(x_1, \cdots, x_n) = \cdots = f_m(x_1, \cdots, x_n) = 0, \tag{2}$$

and $F(\boldsymbol{x}) = [f_1, \cdots, f_m]^{\mathrm{T}}$. Suppose $I = \langle f_1, \cdots, f_m \rangle$ is a radical ideal and V is equidimensional, i.e., the irreducible components of V have same dimensions. Then a point $\widehat{\boldsymbol{x}} \in V$ is called a *regular point* of V if the rank of the Jacobian matrix $F_{\boldsymbol{x}}(\widehat{\boldsymbol{x}})$ satisfies

$$\dim V = n - \operatorname{rank}(F_{\boldsymbol{x}}(\widehat{\boldsymbol{x}})). \tag{3}$$

The set V_{reg} of regular points of V is called the *regular locus* of V. A point $\widehat{\boldsymbol{x}}$ is called *singular* at V if

$$\operatorname{rank}(F_{\boldsymbol{x}}(\widehat{\boldsymbol{x}})) < n - \dim V. \tag{4}$$

The set $V_{sing} := V \backslash V_{reg}$ is called the *singular locus* of V. If all points on V are regular, then V is called *smooth*. If $I = \langle f_1, \cdots, f_m \rangle$ is not a radical ideal, then a point $\widehat{\boldsymbol{x}} \in V$ is called a *regular point* of V if and only if the rank of the Jacobian matrix $G_{\boldsymbol{x}}(\widehat{\boldsymbol{x}})$ satisfies $\dim V = n - \operatorname{rank}(G_{\boldsymbol{x}}(\widehat{\boldsymbol{x}}))$, where $G(\boldsymbol{x}) = [g_1, \cdots, g_s]^{\mathrm{T}}$ is a polynomial basis of $\sqrt{I}$.

Computing sampling points of a semi-algebraic set $\mathcal{S}$ has many applications in robotics, computer vision[1–4]. There are symbolic methods based on cylindrical algebraic decomposition[5] and critical point methods[6–9]. There are several implementations of cylindrical algebraic decomposition: QEPCAD[10], QEPCAD B[11], REDLOG[12], SyNRAC[13], DISCOVERER[14, 15], RegularChains[16]. Algorithms proposed in [17–21] find at least one real point on each connected component of a smooth real algebraic set. RAGlib[22] is a Maple package which provides practical algorithms based on critical point methods for solving polynomial systems of equations and inequalities over the reals. There are also implementations based on homotopy methods and Smale's α-theorem[23–25]. Some hybrid algorithms based on homotopy continuation for computing real solutions are proposed in [26, 27].

A Square Zero-Dimensional Polynomial System Suppose $F(\boldsymbol{x})$ is a square and zero-dimensional polynomial system, i.e., $m = n$. Standard verification methods for nonlinear square systems are based on the following theorem[28–30].

Theorem 1.1 *Let $F(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^n$ be a polynomial system, and $\widetilde{\boldsymbol{x}} \in \mathbb{R}^n$. Let $\mathbb{IR}$ be the set of real intervals, $\mathbb{IR}^n$ and $\mathbb{IR}^{n\times n}$ be the set of real interval vectors and real interval matrices, respectively. Given $\boldsymbol{X} \in \mathbb{IR}^n$ with $\mathbf{0} \in \boldsymbol{X}$ and $M \in \mathbb{IR}^{n\times n}$ satisfies $\nabla f_i(\widetilde{\boldsymbol{x}} + \boldsymbol{X}) \subseteq M_{i,:}$, for $i = 1, 2, \cdots, n$, where $M_{i,:}$ is the i-th row of M. Denote by I_n the $n \times n$ identity matrix and assume*

$$-F_{\boldsymbol{x}}^{-1}(\widetilde{\boldsymbol{x}})F(\widetilde{\boldsymbol{x}}) + (I_n - F_{\boldsymbol{x}}^{-1}(\widetilde{\boldsymbol{x}})M)\boldsymbol{X} \subseteq \operatorname{int}(\boldsymbol{X}), \tag{5}$$

where $F_{\boldsymbol{x}}(\widetilde{\boldsymbol{x}})$ is the Jacobian matrix of $F(\boldsymbol{x})$ at $\widetilde{\boldsymbol{x}}$. Then there is a unique $\widehat{\boldsymbol{x}} \in \boldsymbol{X}$ with $F(\widehat{\boldsymbol{x}}) = 0$. Moreover, every matrix $\widetilde{M} \in M$ is nonsingular. In particular, the Jacobian matrix $F_{\boldsymbol{x}}(\widehat{\boldsymbol{x}})$ is nonsingular.

The non-singularity of the Jacobian matrix $F_{\boldsymbol{x}}(\widehat{\boldsymbol{x}})$ restricts the application of Theorem 1.1 to regular solutions of a square polynomial system. If $F_{\boldsymbol{x}}(\widehat{\boldsymbol{x}})$ is singular and $\widehat{\boldsymbol{x}}$ is an isolated singular solution of $F(\boldsymbol{x})$, in [31–33], by adding smoothing parameters properly to $F(\boldsymbol{x})$, an extended regular and square polynomial system is generated for computing verified error bounds, such that a slightly perturbed polynomial system of $F(\boldsymbol{x})$ is guaranteed to possess an isolated singular solution within the computed bounds. Methods in [34, 35] can also be used to verify isolated singular solutions.

There are two functions verifynlss and verifynlss2 in the INTLAB package implemented by Rump in Matlab[36]. The procedure verifynlss can be used to verify the existence of a simple root of a square and regular zero-dimensional polynomial system and verifynlss2 can be used to verify the existence of a double root of a slightly perturbed polynomial system of $F(\boldsymbol{x})$. If the polynomial system $F(\boldsymbol{x})$ has an isolated singular root with multiplicity larger than 2, then the function viss designed in [31, 32] and implemented by Li and Zhi in Matlab can be applied to obtain verified error bounds such that a slightly perturbed polynomial system of $F(\boldsymbol{x})$ is guaranteed to possess an isolated singular solution within the computed bounds.

An Overdetermined Zero-Dimensional Polynomial System Suppose $F(\boldsymbol{x})$ is an overdetermined zero-dimensional polynomial system, i.e., $m > n$. A natural procedure for obtaining a square polynomial system from $F(\boldsymbol{x})$ is to pick up a full rank random matrix $A \in \boldsymbol{Q}^{n\times m}$ and form a square polynomial system $A \cdot F(\boldsymbol{x})$. According to [4, Theorem 13.5.1], we have the following theorem.

Theorem 1.2 *There is a nonempty Zariski open subset $\mathcal{A} \in \boldsymbol{C}^{n\times m}$ such that for every $A \in \mathcal{A}$, a solution of $F(\boldsymbol{x})$ is regular if and only if it is a nonsingular solution of the square system $A \cdot F(\boldsymbol{x})$. Moreover, if $F(\boldsymbol{x})$ is a zero-dimensional system, then $A \cdot F(\boldsymbol{x})$ is also a zero-dimensional system.*

According to Theorem 1.2, we can compute verified regular solutions of the square polynomial system $A \cdot F(\boldsymbol{x})$ according to Theorem 1.1, and check whether the verified solution of $A \cdot F(\boldsymbol{x})$ is a solution of $F(\boldsymbol{x})$ by computing the residual of $F(\widehat{\boldsymbol{x}})$ as an additional test, see also [25, Lemma 3.1]. If the residue $F(\widehat{\boldsymbol{x}})$ is small, with high probability, the verified real solution of $A \cdot F(\boldsymbol{x})$ is a real solution of $F(\boldsymbol{x})$. Another interesting method based on hybrid symbolic-numeric methods was given in [37], which computed an exact rational univariate representation (RUR) of a component of the input system from approximate solutions for certifying that a given point is near an exact solution of an overdetermined polynomial system.

A Positive-Dimensional Polynomial System Suppose $F(\boldsymbol{x})$ is a positive-dimensional polynomial system. It is clear that an under-determined system $F(\boldsymbol{x})$ is a positive-dimensional system whose dimension is at least $n - m \geq 1$. A square polynomial system and an overdetermined system can also be positive-dimensional. In [38, 39], the authors transformed an underdetermined system into a regular square system by choosing m independent variables and setting $n - m$ remaining variables to be anchors, then they used a Krawczyk-type interval operator to verify the existence of the solutions of the transformed regular and square system. It is very impressive that they can verify a solution of a polynomial system with more than 10000 variables and 20000 equations with degrees as high as 100. More general methods using linear slices to reduce the underdetermined system to a square system were proposed in [4, 40, 41]. We notice that it is very important to choose independent variables and initial values for the dependent variables or linear slices. Especially, we might have a big chance to miss the real points because of the bad choice for values of some variables.

Consider the polynomial *Vor*2, which appears in a problem studying Voronoi Diagram of three lines in $\boldsymbol{R}^3$[42]. *Vor*2 is a polynomial in five variables with degree 18. It has an infinite number of real solutions. Let us set four variables as rational numbers chosen in the range $[-\frac{3000}{1000}, \frac{3000}{1000}]$, e.g.,

$$\widehat{x}_2 = \frac{177}{500}, \quad \widehat{x}_3 = \frac{423}{1000}, \quad \widehat{x}_4 = \frac{209}{1000}, \quad \widehat{x}_5 = \frac{143}{50},$$

the univariate polynomial $V(x_1) = \mathit{Vor}2\ (x_1, \widehat{x}_2, \widehat{x}_3, \widehat{x}_4, \widehat{x}_5) \in \boldsymbol{Q}[x_1]$ has no real solutions.

In [43], we proposed several different strategies to construct a square and zero-dimensional polynomial system for computing verified real solutions of positive-dimensional polynomial systems. In this paper, we extend these methods to verify the existence of real solutions of semi-algebraic sets defined by (1).

Structure of the Paper In Section 2, we introduce several known methods for computing verified real solutions for positive-dimensional polynomial systems of equations and extend these methods for verifying the existence of real solutions of semi-algebraic sets defined by (1). In Section 3, we present two algorithms for computing verified real solutions, the first one is based on low-rank moment matrix completion method and the second one is based on the critical point method and the homotopy continuation method. In Section 4, we demonstrate the efficiency of the algorithms for computing verified real solutions of a set of benchmark systems.

2 Brief Description of Two Methods

In this section, we briefly introduce two methods for verifying the existence of real solutions of semi-algebraic systems.

The Low-Rank Moment Matrix Completion Method Numerical semidefinite programming (SDP) has been used for characterizing and computing the real solutions of polynomial systems[44–46]. As pointed out in [46], the great benefit of using SDP techniques is that it exploits the real algebraic nature of the problem right from the beginning and avoids the computation of complex components. For example, if $V \cap \boldsymbol{R}^n$ is zero-dimensional, then the moment-matrix algorithm in [46] can compute all real solutions of $F(\boldsymbol{x})$ by solving a sequence of SDP problems. If the polynomial system (1) has an infinite number of real solutions, then the algorithm in [46] can not be used. Hence, in [45, 47], they replaced the constant object function by the trace of the moment matrix and showed that their software GloptiPoly is very efficient for finding a partial set of real solutions for a large set of polynomial systems (see [47, Table 6.3, 6.4]). Since the trace of a semidefinite moment matrix is equal to its nuclear norm defined as the sum of its singular values, the optimization problem can be transformed to the following nuclear norm minimization problem:

$$\begin{cases} \min & \|M_t(y)\|_* \\ \text{s.t.} & y_0 = 1, \\ & M_t(y) \succeq 0, \\ & M_{t-d_i}(f_i\ y) = 0, \quad i = 1, 2, \cdots, m, \\ & M_{t-d_j}(g_j\ y) \succeq 0, \quad j = 1, 2, \cdots, s. \end{cases} \tag{6}$$

In [48], an algorithm MMCRSolver based on accelerated fixed point continuation method and alternating direction method was presented to solve the minimization problem (6) for finding real solutions of (1), even when the number of real solutions of (1) is infinite. Although the method based on function values and gradient evaluations cannot yield as high accuracy as interior point methods, much larger problems can be solved since no second-order information needs to be computed and stored.

In [43], we proposed a method based on the low-rank moment matrix completion to verify the existence of real solutions for positive-dimensional polynomial systems. Suppose $\widetilde{\boldsymbol{x}}$ is an approximate real solution of (1) computed by MMCRSolver. If the rank of the Jacobian matrix $F_{\boldsymbol{x}}(\widetilde{\boldsymbol{x}})$ is less than $n-d$, then $\widetilde{\boldsymbol{x}}$ is a singular point on V. Stimulated by the deflation method used in [49] for constructing extended regular polynomial systems, we compute a normalized null vector $\boldsymbol{v}$ ($|\boldsymbol{v}|_2 = 1$) of $F_{\boldsymbol{x}}(\widetilde{\boldsymbol{x}})$ and generate a new polynomial system $\widetilde{F}(\boldsymbol{x}) = F(\boldsymbol{x}) \cup \{\boldsymbol{v}^{\mathrm{T}} \cdot (\boldsymbol{x} - \widetilde{\boldsymbol{x}})\}$. It is clear that $\widetilde{\boldsymbol{x}}$ is a real solution of

$$\begin{cases} F(\boldsymbol{x}) = \boldsymbol{0}, \\ \boldsymbol{v}^{\mathrm{T}} \cdot (\boldsymbol{x} - \widetilde{\boldsymbol{x}}) = \boldsymbol{0}. \end{cases} \tag{7}$$

It is possible that $\widetilde{\boldsymbol{x}}$ is still a singular solution of $\widetilde{F}$, then we can perform the similar deflations to the system $\widetilde{F}(\boldsymbol{x})$ again.

If the approximate solution $\widetilde{\boldsymbol{x}}$ is not a singular point on the variety $V \cap \boldsymbol{R}^n$, i.e., the rank of the Jacobian matrix $F_{\boldsymbol{x}}(\widetilde{\boldsymbol{x}})$ is $n-d$, then we choose a normalized random vector λ and construct a new polynomial system $\widetilde{F}(\boldsymbol{x}) = F(\boldsymbol{x}) \cup \{F_{\boldsymbol{x}}(\boldsymbol{x})\lambda - F_{\boldsymbol{x}}(\widetilde{\boldsymbol{x}})\lambda\}$. It is clear that $\widetilde{\boldsymbol{x}}$ is a solution of

$$\begin{cases} F(\boldsymbol{x}) = \boldsymbol{0}, \\ F_{\boldsymbol{x}}(\boldsymbol{x})\lambda - F_{\boldsymbol{x}}(\widetilde{\boldsymbol{x}})\lambda = \boldsymbol{0}. \end{cases} \tag{8}$$

Suppose $F(\boldsymbol{x})$ or its extended polynomial system $\widetilde{F}(\boldsymbol{x})$ is zero-dimensional. The procedure verifynlss can be used to compute the real root inclusion of X for $F(\boldsymbol{x})$ or $\widetilde{F}(\boldsymbol{x})$ if $\boldsymbol{x}$ is the approximate simple root of $F(\boldsymbol{x})$ or $\widetilde{F}(\boldsymbol{x})$; verifynlss2 or viss can be used to obtain the real root inclusion of a slightly perturbed polynomial system of $F(\boldsymbol{x})$ or $\widetilde{F}(\boldsymbol{x})$.

The verification algorithm based on using the null vector of $F_{\boldsymbol{x}}(\widetilde{\boldsymbol{x}})$ or a random vector to construct new polynomial systems can be more efficient since it avoids the computations of minors or the introduction of new variables. However, since we only use local information about the approximate real root $\widetilde{\boldsymbol{x}}$ of $F(\boldsymbol{x})$ in order to construct the new extended system, it is limited to verify the existence of a real root $\widehat{\boldsymbol{x}}$ in the neighborhood of $\widetilde{\boldsymbol{x}}$. For some interesting applications, it is enough to verify the existence of one real solution, e.g., for deciding reachability of the infimum of a multivariate polynomial[50], if we can verify the existence of one real solution for $f - f^*$, then we prove that f^* is a minimum which can be attained.

Suppose X is the real root inclusion of $F(\boldsymbol{x})$ or its slightly perturbed polynomial system. The remaining task is to verify whether X satisfies all inequalities in (1). We call the procedure eval in the C-XSC libraries for determining whether $g_i(X) > 0$ is true for $i = 1, 2, \cdots, s$. After the above steps, we obtain the real root inclusion X for verifying the existence of the real solution satisfying (1).

The Critical Point Method Considering the case the ideal I generated by $f_1(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})$ is radical and V is of dimension $d$and contains a regular point in $\boldsymbol{R}^n$. We can compute a regular real sample point on V by computing its critical points of a distance function to a generic point restricted to V. This method was proposed in [17, 20, 51], see also [19, 52, 53] for some recent results when $F(\boldsymbol{x})$ has real singular solutions.

For an arbitrary point $\boldsymbol{u} = (u_1, \cdots, u_n) \in \boldsymbol{R}^n$, let $g = \frac{1}{2}(x_1 - u_1)^2 + \frac{1}{2}(x_2 - u_2)^2 + \cdots + \frac{1}{2}(x_n - u_n)^2$ and

$$J_g(F) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} & \frac{\partial g}{\partial x_1} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} & \frac{\partial g}{\partial x_n} \end{bmatrix}. \tag{9}$$

We define the algebraic set:

$$C(V, \boldsymbol{u}) = \{\widehat{\boldsymbol{x}} \in V, \operatorname{rank}(J_g(F(\widehat{\boldsymbol{x}})) \le n - d\}. \tag{10}$$

Let $\Delta_{\boldsymbol{u},d}(F)$ be the set of all the minors of order $n - d + 1$ in the matrix $J_g(F)$ such that their last column contains the entries in the last column of $J_g(F)$.

According to [17, Theorem 2.3], for almost all $\boldsymbol{u}$, the dimension of the algebraic variety $C(V, \boldsymbol{u})$ of $\Delta_{\boldsymbol{u},d}(F) \cup F(\boldsymbol{x})$ will be strict less than the dimension of V. Therefore, inductively, we will obtain a zero-dimensional polynomial system which can be used to verify the existence of regular real solutions on V. As stated in [17], the main bottleneck for the critical points method is the computation of $\Delta_{\boldsymbol{u},d}$ since the number of elements in $\Delta_{\boldsymbol{u},d}$ is equal to $\binom{m}{n-d}\binom{n}{n-d+1}$ and the polynomials in $\Delta_{\boldsymbol{u},d}$ are usually dense and have large coefficients. An alternative way to avoid the computation of the minors is to introduce extra variables $\lambda_0, \lambda_1, \cdots, \lambda_{n-d}$ and pick up randomly $n-d$ real numbers $a_0, a_1, \cdots, a_{n-d}$ and polynomials in $F(\boldsymbol{x})$ such as $f_1, f_2, \cdots, f_{n-d}$, and replace the minors in $\Delta_{\boldsymbol{u},d}$ by polynomials defined below

$$p_i = \lambda_0 \frac{\partial g}{\partial x_i} + \lambda_1 \frac{\partial f_1}{\partial x_i} + \cdots + \lambda_{n-d} \frac{\partial f_{n-d}}{\partial x_i}, \quad \text{for } 1 \le i \le n,$$

$$p_{n+1} = a_0\lambda_0 + a_1\lambda_1 + \ldots + a_{n-d}\lambda_{n-d} - 1.$$

This is the way used in [54, Theorem 5] to generate solution paths leading to real solutions on V using the homotopy continuation method.

If V is compact and smooth, and the variables $x_1, x_2, \cdots, x_n$ are in a generic position with respect to $f_1, f_2, \cdots, f_m$, then as shown in [18, Theorem 10], one can change the distance function g to a coordinate function $g = x_i, 1 \leq i \leq n$ such that the dimension of the real variety of $\Delta_{\boldsymbol{u},d}(F) \cup F(\boldsymbol{x})$ will be zero and contains at least one real point on each connected component of $V \cap \boldsymbol{R}^n$. Moreover, in [21], they extended the result in [18] to deal with the case where $V \cap \boldsymbol{R}^n$ is non-compact.

If the ideal I generated by polynomials in $F(\boldsymbol{x})$ is not radical, it is difficult to verify the exact existence of real points on singular locus V_{sing} which might have the same dimension as V, see [43, Example 3]. According to [6, 20], one can add one or more infinitesimal deformations to polynomials in $F(\boldsymbol{x})$ and work over a non-archimedean real closed extension of the ground field. The computation could be quite expensive. Therefore, instead of proving the exact existence of real roots on the variety defined by a non-radical ideal, one can perturb the system by a tiny real number and show the existence of real roots of this slightly perturbed polynomial system[20, 54]. Notice here, the perturbed system is smooth, $C(V, \boldsymbol{u})$ is a zero-dimensional variety. However, it contains approximate singular solutions. Hence, it is necessary to apply the algorithm viss to verify the existence of real singular solutions of a slightly perturbed system. Moreover, since computations in Matlab have limited precisions, with or without tiny perturbations, we may get similar results.

There are many ways to extend the critical point method and the homotopy continuation method to compute approximate real solutions of semi-algebraic systems. Here we choose to introduce slack variables and transform the semi-algebraic system into the corresponding polynomial system. The system (1) can be written as the following polynomial equations with slack variables $y_1, y_2, \cdots, y_s$:

$$\widehat{F}(\boldsymbol{x}, \boldsymbol{y}) : \begin{cases} f_1(\boldsymbol{x}) = f_2(\boldsymbol{x}) = \cdots = f_m(\boldsymbol{x}) = 0, \\ g_1(\boldsymbol{x}) - y_1^2 = g_2(\boldsymbol{x}) - y_2^2 = \cdots = g_s(\boldsymbol{x}) - y_s^2 = 0, \end{cases} \tag{11}$$

here $y_1, y_2, \cdots, y_s$ are required to be nonzero.

We can apply the algorithm verifyrealroopc described in [43] for computing the verified solutions on each connected component on the algebraic variety defined by $\widehat{F}(\boldsymbol{x}, \boldsymbol{y})$. If our goal is to compute at least one verified real solution for $\widetilde{F}(\boldsymbol{x}, \boldsymbol{y})$, then instead of computing all minors of the Jacobian matrix, we can use the most possible canonical projections, i.e., fixing as many variables as possible to construct the zero-dimensional polynomial system. It is clear that X is also the real root inclusion of the original polynomial system $F(\boldsymbol{x})$. Similarly, the function eval is called to verify whether X satisfies all inequalities in (1).

3 Algorithms and Implementations

VerifyRealRoots is a Matlab package for computing at least one verified real root inclusion of a polynomial system of equations and inequalities. It is based on Bertini and C-XSC libraries. Bertini[55] solver is employed to compute approximate real solutions based on the homotopy continuation method, i.e., Line 14 in Algorithm 2. The procedure AINLSS in C-XSC is used to verify the existence of a square and regular zero-dimensional polynomial system. Following the methods provided in [33] and [31, 56], we also implement two procedures in the C-XSC library: AINLSS2 is used to verify the existence of a double root of a slightly perturbed polynomial system; AIVISS can be applied to verify the existence of an isolated singular root, with the multiplicity larger than 2, of a slightly perturbed polynomial system. We refer to the user manual for a detailed description of the options and an example to illustrate how to call the Matlab function VerifyRealRoots.

Remark 3.1 There are polynomial systems with infinite number of real solutions, e.g., Example 1 in the User's manual has four variables constrained by two equations and two inequalities and the Kissing [2, 2] has four variables constrained by two equations and one inequality. In order to speed up the computations, we add some additional constraints (some are chosen randomly) which may cause different outputs when you run the algorithm for finding real sample points of these polynomial systems. It should also be pointed out that our algorithm may fail to find sample points due to poorly chosen initial points, step sizes, tolerances, or relaxation orders. You may choose different options listed in Table 2 of the User's manual for solving your problems.

Remark 3.2 We would like to point out that unlike symbolic methods[17, 20, 21, 51], our algorithms can not be used to verify the nonexistence of real solutions in $\mathcal{S}$, i.e., the failure of our algorithms do not mean there exist no real solutions in $\mathcal{S}$.

Remark 3.3 The extended polynomial system $\widetilde{F}(\boldsymbol{x})$ generated in Line 17 of Algorithm 1 is not guaranteed to be of zero dimension. If $\widetilde{F}(\boldsymbol{x})$ is still of positive dimension and the verification step in Lines 20–24 fails to get the verified real root inclusion for $\boldsymbol{a}_i$, then we add more polynomials vanishing at $\boldsymbol{a}_i$ to $\widetilde{F}(\boldsymbol{x})$.

Remark 3.4 In Algorithm 1 and Algorithm 2, if $F(\boldsymbol{x})$ is underdetermined, i.e., $m < n$, our first choice of the random matrix A will always have the block structure $\left[\begin{smallmatrix} I_m & 0 \\ 0 & A_{sub} \end{smallmatrix}\right]$, where A_{sub} is chosen randomly. In this case, we do not need to compute the residue. The verified solution of $\widetilde{F}$ will also be a verified one of $F(\boldsymbol{x})$.

4 Experiments

In this section, we demonstrate on several examples that VerifyRealRoots can yield the verified real solutions of polynomial systems with equalities and inequalities. The installation instructions and user guide of VerifyRealRoots can be found at: `http://159.226.47.210:8080/`. The numerical experiments below were carried out on Intel(R) Core(TM) at 2.3GHz with Matlab (R2018a) under Mac OS X.

4.1 Kissing Number Problem

The kissing number problems seek the maximal number of unit n-dimensional spheres such that they can simultaneously touch the unit sphere in n-dimensional Euclidean space without

Algorithm 1 Computing verified real solutions based on low-rank moment matrix completion method

```
Input: A polynomial system $\mathcal{S}$ : $\{f_1(\boldsymbol{x}) = 0, f_2(\boldsymbol{x}) = 0, \cdots, f_m(\boldsymbol{x}) = 0, g_1(\boldsymbol{x}) \geq 0, g_2(\boldsymbol{x}) \geq 0, \cdots, g_s \geq 0\}$, a given small tolerance $\varepsilon \in \boldsymbol{R}_{>0}$.
Output: The set $L$ consisting of the verified inclusions of real solutions.
 1: Set $F(\boldsymbol{x}) = [f_1, \cdots, f_m]$, $G(\boldsymbol{x}) = [g_1, \cdots, g_s]$, and $L = \{\}$.
 2: Apply MMCRSolver to obtain approximate real solutions of $S$, and denote the solution set
    as $M = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k\}$, where $\boldsymbol{a}_i \in \boldsymbol{R}^n, i = 1, 2, \cdots, k$
 3: if $F(\boldsymbol{x}) = \{\,\}$ then
 4:     for $i = 1$ to $k$ do
 5:         Set the degenerated inclusion $X = [\boldsymbol{a}_i, \boldsymbol{a}_i]$
 6:         Compute the evaluation of $\boldsymbol{G}(\boldsymbol{x})$ at $X$
 7:         Let $\rho = \min\{g_1(X), \cdots, g_\ell(X)\}$
 8:         if $\rho \geq 0$ then
 9:             Set $L = L \cup X$;
10: else
11:     for $i = 1$ to $k$ do
12:         if $F_{\boldsymbol{x}}(\boldsymbol{a}_i)$ is singular then
13:             Compute normalized null vectors of $F_{\boldsymbol{x}}(\boldsymbol{a}_i)$, denoted by $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_t$
14:             $\widetilde{F}(\boldsymbol{x}) = F(\boldsymbol{x}) \cup \left\{\boldsymbol{v}_j^{\mathrm{T}} \cdot (\boldsymbol{x} - \boldsymbol{a}_i), j = 1, 2, \cdots, t\right\}$
15:         else
16:             Choose random normalized vectors $\lambda_1, \cdots, \lambda_t$
17:             $\widetilde{F}(\boldsymbol{x}) = F(\boldsymbol{x}) \cup \{F_{\boldsymbol{x}}(\boldsymbol{x})\,\lambda_j - F_{\boldsymbol{x}}(\boldsymbol{a}_i)\lambda_j, j = 1, 2, \cdots, t\}$
18:         Choose a random matrix $A = \boldsymbol{Q}^{n\times(m+t)}$, update $\widetilde{F}$ to be $A \cdot \widetilde{F}$
19:         if $\widetilde{F}_{\boldsymbol{x}}(\boldsymbol{a}_i)$ is regular then
20:             Call AINLSS to obtain the real root inclusion $X$ of $\widetilde{F}(\boldsymbol{x})$, set $b = 0$
21:         else
22:             Call AINLSS2 or AIVISS to obtain the error bound $b$ and the root inclusion $X$
23:         Compute the residue $\tau = F(X)$
24:         if $\tau < \varepsilon$ and $b < \varepsilon$ then
25:             Compute the evaluation of $\boldsymbol{G}(\boldsymbol{x})$ at $X$, and let $\rho = \min\{g_1(X), \cdots, g_s(X)\}$
26:             if $\rho \geq 0$ then
27:                 Set $L = L \cup X$.
28: return $L$
```

pairwise overlapping. The kissing numbers in dimensions one, two, three, four, eight, and twenty-four have been found, which are 2, 6, 12, 24, 240 and 196560, respectively. Gabriele Nebe and Neil Sloane maintained a table of the highest kissing numbers presently known `http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/kiss.html`. In [57], they presented a method based on SDP for computing upper bounds for kissing numbers.

Algorithm 2 Computing verified real solutions via the critical point method and the homotopy continuation

```
Input: A polynomial system $\mathcal{S}: \{f_1(\boldsymbol{x}) = 0, f_2(\boldsymbol{x}) = 0, \cdots, f_m(\boldsymbol{x}) = 0, g_1(\boldsymbol{x}) \geq 0, g_2(\boldsymbol{x}) \geq 0, \cdots, g_s \geq 0\}$, a given small tolerance $\varepsilon \in \boldsymbol{R}_{>0}$.
Output: The set $L$ consisting of the verified inclusions of real solutions.
 1: Set $F(\boldsymbol{x}) = [f_1, \cdots, f_m]$, $G(\boldsymbol{x}) = [g_1, \cdots, g_s]$, and $L = \{\}$;
 2: Introduce slack variables $y_1, \cdots, y_s$, and let $\boldsymbol{z} = [x_1, \cdots, x_n, y_1, \cdots, y_s]^{\mathrm{T}}$;
 3: Construct the new polynomial system $H(\boldsymbol{z}) = [f_1, \cdots, f_m, g_1 - y_1^2, \cdots, g_s - y_s^2]$;
 4: if $m < n$ then
 5:     Apply the critical point method to construct a zero-dimensional system $\widetilde{H}(\boldsymbol{z})$;
 6: else
 7:     Set $\widetilde{H}(\boldsymbol{z}) = H(\boldsymbol{z})$;
 8: Let $u$ be the number of polynomials in $\widetilde{H}(\boldsymbol{z})$;
 9: if $u > n + s$ then
10:     Choose a random matrix $A \in \boldsymbol{Q}^{(n+s)\times u}$;
11: else
12:     Set $A = I_{n+s}$;
13: Update $\widetilde{H}(\boldsymbol{z})$ to be $A \cdot \widetilde{H}(\boldsymbol{z})$;
14: Apply Bertini to obtain approximate real solutions of $\widetilde{H}(\boldsymbol{z})$, and denote the solution set as
    $M = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k\}$, where $\boldsymbol{a}_i \in \boldsymbol{R}^{n+s}, i = 1, 2, \cdots, k$;
15: if $F(\boldsymbol{x}) = \{\ \}$ then
16:     for $i = 1$ to $k$ do
17:         Let $\boldsymbol{a}_{i,1}$ be the subvector of $\boldsymbol{a}_i$corresponding to $\boldsymbol{x}$, i.e., $\boldsymbol{a}_{i,1} = \boldsymbol{a}_i(1 : n)$;
18:         Set the degenerated inclusion $X = [\boldsymbol{a}_{i,1}, \boldsymbol{a}_{i,1}]$;
19:         Compute the evaluation of $\boldsymbol{G}(\boldsymbol{x})$ at $X_i$;
20:         Let $\rho = \min\{g_1(X), \cdots, g_\ell(X)\}$;
21:         if $\rho \geq 0$ then
22:             Set $L = L \cup X$;
23: else
24:     for $i = 1$ to $k$ do
25:         if $\widetilde{H}_{\boldsymbol{z}}(\boldsymbol{a}_i)$ is regular then
26:             Call AINLSS to obtain the real root inclusion $Z$ of $\widetilde{H}(\boldsymbol{z})$, set $b = 0$;
27:         else
28:             Call AINLSS2 or AIVISS to obtain the error bound $b$ and the root inclusion $Z$;
29:         Let $X$ be the sub-inclusion of $Z$ corresponding to $\boldsymbol{x}$;
30:         Compute the residue $\tau = F(X)$;
31:         if $\tau < \varepsilon$ and $b < \varepsilon$ then
32:             Set $L = L \cup X$.
33: return $L$
```

The kissing number problems can be transformed as verifying the existence of real roots of the semi-algebraic systems. Now let us consider the 2-dimensional case. We show that there exist 6 unit circles which can touch the given unit circle without pairwise overlapping. Suppose the touching points are $T_i, 1 \leq i \leq 6$, whose coordinates are (x_i, y_i), respectively.

Observing Figure 1, the touching points must be on the circle, and the distance between any two different touching points is at least one, i.e.,

$$\begin{cases} x_i^2 + y_i^2 = 1, & 1 \leq i \leq 6, \\ (x_i - x_j)^2 + (y_i - y_j)^2 \geq 1, & 1 \leq i < j \leq 6. \end{cases} \tag{12}$$

It is noticed that the above semi-algebraic system (12) has 12 variables, 6 equations and 15 inequalities. After 1898 seconds computation, our algorithm can yield 13 verified real solutions of the system (12). Let $(x_{1,i}, x_{2,i}, \cdots, x_{n,i})^{\mathrm{T}}$ be the coordinates of the touching points. Suppose k unit hyperspheres in n-dimensions can touch the given unit hypersphere without any intersection, which means that each distance between two arbitrary touching points is ≥ 1. Therefore, the general case of the kissing number problem *kissing*$[n,k]$ is equivalent to verifying the existence of real roots of the following semi-algebraic system:

$$\begin{cases} x_{1,i}^2 + x_{2,i}^2 + \cdots + x_{n,i}^2 = 1, & 1 \leq i \leq k, \\ (x_{1,u} - x_{1,v})^2 + (x_{2,u} - x_{2,v})^2 + \cdots + (x_{n,u} - x_{n,v})^2 \geq 1, & 1 \leq u < v \leq k. \end{cases} \tag{13}$$

The semi-algebraic system (13) has $n \cdot k$ variables, k equations and $k \cdot (k-1)/2$ inequalities. It is noticed that all polynomials in (13) are quadratic.

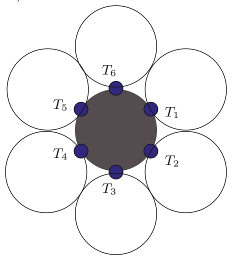

Figure 1 Kissing [2, 6]: 6 Unit 2-dimensional spheres

Comparison RAGlib[22] is a Maple package providing useful functionalities for the study of real solutions of polynomial systems of equations and inequalities. HasRealSolutions command in RAGlib is used to compute the sampling points in each connected component of their real solution set. Algorithm 1 and HasRealSolutions have been applied to solve the kissing number problems with different n and k. We report their performances in Table 1. Here *var* denotes the number of the variables of polynomials; $\sharp$ *eq* denotes the number of the equations; $\sharp$ *ineq* denotes the number of the inequalities; *sol* denotes the number of the verified solutions; *time*(s) is given in seconds for computing verified real solutions. Since the homotopy method in Algorithm 2 is not designed for solving polynomial systems with inequalities, we do not recommend it for solving kissing number problem.

Table 1 Algorithm performance on kissing number problems

Ex	*var*	$\sharp eq$	$\sharp ineq$	VerifyRealRoots		HasRealSolutions	
				time(s)	*sol*	*time*(s)	*sol*
kissing [2, 1]	2	1	0	1.935	2	0.088	2
kissing [2, 2]	4	2	1	6.130	6	0.161	4
kissing [2, 3]	6	3	3	11.520	8	0.263	2
kissing [2, 4]	8	4	6	21.982	4	13.729	1
kissing [2, 5]	10	5	10	117.345	19	—	—
kissing [2, 6]	12	6	15	1898.403	13	—	—
kissing [3, 1]	3	1	0	0.630	4	0.129	2
kissing [3, 2]	6	2	1	4.250	10	0.298	4
kissing [3, 3]	9	3	3	15.640	16	25.695	4
kissing [3, 4]	12	4	6	30.307	2	—	—
kissing [3, 5]	15	5	10	482.830	6	—	—
kissing [4, 1]	4	1	0	2.160	4	0.227	2
kissing [4, 2]	8	2	1	7.840	7	0.484	2
kissing [4, 3]	12	3	3	252.71	8	1194.98	2
kissing [4, 4]	16	4	6	100.52	10	—	—
kissing [4, 5]	20	5	10	1011.590	36	—	—
kissing [5, 1]	5	1	0	1.820	4	0.556	2
kissing [5, 2]	10	2	1	9.391	8	1.103	4
kissing [5, 3]	15	3	3	297.130	26	—	—
kissing [5, 4]	20	4	6	644.370	34	—	—
kissing [6, 1]	6	1	0	4.332	4	0.446	2
kissing [6, 2]	12	2	1	9.682	11	0.812	2
kissing [6, 3]	18	3	3	445.390	25	—	—
kissing [6, 4]	24	4	6	1401.440	34	—	—
kissing [7, 1]	7	1	0	3.310	7	0.802	2
kissing [7, 2]	14	2	1	24.023	18	1.186	2
kissing [7, 3]	21	3	3	684.789	29	—	—
kissing [8, 1]	8	1	0	0.885	2	1.603	2
kissing [8, 2]	16	2	1	15.354	15	1.800	2
kissing [8, 3]	24	3	3	1559.370	37	—	—

4.2 Satellite Trajectory Control

The objective of the satellite trajectory control is to keep a satellite in orbit. Illustrated in [58, 59], it can be transformed as the problem of computing dynamic output feedback laws of the linear control system. Pole placement method is used to find laws to feed the output to the input so that the closed-loop system has given eigenvalues (poles). In [58, 59], the state vector is $\boldsymbol{x} = [r, \dot{r}, \theta, \dot{\theta}]$, and the input is $[u_r, u_t]$, where r and θ are the deviations form the reference orbit and the reference attitude, u_r and u_t are the radial and tangential thrusters, respectively. The linearized state-space equations are defined by the following matrices:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3\omega_0^2 & 0 & 0 & 2\omega_0 r_0 \\ 0 & 0 & 0 & 1 \\ 0 & -2\dfrac{\omega_0}{r_0} & 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 \\ \dfrac{1}{v} & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{v\, r_0} \end{bmatrix}, \tag{14}$$

where r_0, ω_0, v are the radius of the orbit, the angular velocity, and the mass of the satellite respectively. They pick $C = \left[\begin{smallmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{smallmatrix}\right]$ such that the satellite is completely controllable.

The above matrices define the following linear system

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t). \end{cases} \tag{15}$$

Meanwhile, the control law can also be represented as a linear system

$$\begin{cases} \dot{z}(t) = Fx(t) + Ly(t), \\ u(t) = Hz(t) + My(t), \end{cases} \tag{16}$$

where

$$F = \begin{bmatrix} f \end{bmatrix}, \quad L = \begin{bmatrix} l_1 & l_2 \end{bmatrix}, \quad H = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix}.$$

According to (15) and (16), we can get the associated closed-loop system:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} A - BMC & -BH \\ LC & F \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}. \tag{17}$$

Our problem is to compute the real matrices F, L, H, M when the system matrices A, B, C and the eigenvalues (poles) λ_i's of the system (17) are given. We compute the characteristic polynomial $p(\lambda)$ of the system (17),

$$\begin{aligned} p(\lambda) = {} & \lambda^5 v r_0 + \lambda^3 v r_0 \omega_0^2 - \lambda^4 v r_0 f - \lambda^2 v r_0 \omega_0^2 f - 2\lambda\omega_0 h_1 l_1 \\ & + \lambda^2 h_2 l_1 - 3\omega_0^2 h_2 l_1 - 2\lambda^2 \omega_0 h_1 l_2 + \lambda^3 h_2 l_2 - 3\lambda\omega_0^2 h_2 l_2 \\ & - 2\lambda^2 \omega_0 m_{1,1} + 2\lambda\omega_0 f m_{1,1} - 2\lambda^3 \omega_0 m_{1,2} + 2\lambda^2 \omega_0 f m_{1,2} + \lambda^3 m_{2,1} \\ & - 3\lambda\omega_0^2 m_{2,1} - \lambda^2 f m_{2,1} + 3\omega_0^2 f m_{2,1} + \lambda^4 m_{2,2} - 3\lambda^2 \omega_0^2 m_{2,2} \\ & - \lambda^3 f m_{2,2} + 3\lambda\omega_0^2 f m_{2,2}. \end{aligned}$$

Following [58, 59], we may choose the eigenvalues as

$$\lambda_1 = \frac{-2+\mathrm{i}}{\sqrt{5}}, \quad \lambda_2 = \frac{-2-\mathrm{i}}{\sqrt{5}}, \quad \lambda_3 = -5, \quad \lambda_4 = -7, \quad \lambda_5 = -3,$$

and the parameters

$$v = 0.74564, \quad \omega_0 = 0.345354, \quad r_0 = 1.2342.$$

Substituting the eigenvalues λ_i's to $p(\lambda)$, we can get the positive-dimensional polynomial system

$$p(\lambda_1) = p(\lambda_2) = \cdots = p(\lambda_5) = 0,$$

with 9 variables: $f, l_1, l_2, h_1, h_2, m_{1,1}, m_{1,2}, m_{2,1}, m_{2,2}$. In practice, people are only interested in the real feedback laws, i.e., the real solutions of the positive-dimensional polynomial system. Hence, the problem of satellite trajectory control by the pole placement can be reformulated as computing real solutions of the positive-dimensional polynomial systems. By using of our algorithm, we can find one verified real solution:

f	h_1	h_2	l_1	l_2
$-18.07 \pm 2 \times 10^{-13}$	$-6.88 \pm 8 \times 10^{-12}$	$-21.14 \pm 3 \times 10^{-12}$	$15.58 \pm 1 \times 10^{-11}$	$-3.839 \pm 4 \times 10^{-13}$

and

$m_{1,1}$	$m_{1,2}$	$m_{2,1}$	$m_{2,2}$
$-14.96 \pm 1 \times 10^{-12}$	$-40.28 \pm 6 \times 10^{-12}$	$3.283 \pm 1 \times 10^{-11}$	$-1.184 \pm 6 \times 10^{-13}$

.

The total CPU time is around 612 seconds.

4.3 Some Benchmark Examples

Polynomial systems with equalities. We apply Algorithm 1 and Algorithm 2 for computing verified real solutions of polynomial systems, and report their performances in Table 2. All examples are taken from the homepage of Jan Verschelde `http://www.math.uic.edu/~jan`.

Table 2 Algorithm performance on polynomial systems

Ex	*var*	*♯eq*	*deg*	VerifyRealRoots (M)		VerifyRealRoots (H)	
				time(s)	*sol*	*time(s)*	*sol*
butcher	5	2	3	1.020	1	0.2000	2
birkhoff	4	1	10	3.000	1	0.4800	6
*cohn*2	4	4	5	4.050	1	16.040	1
*adjmin*22*e*4	6	2	2	1.010	1	0.970	1
*comb*3000	10	10	3	4.203	1	0.470	4
*noon*4	4	4	3	9.060	1	0.2000	2
*hairer*1	8	6	2	5.070	1	1.370	2
lanconelli	8	3	3	2.890	1	1.783	2
raksanyi	8	4	2	6.970	1	1.600	2
*bronestein*2	4	3	3	33.100	1	6.130	2
*i*1	10	10	3	3.400	1	9.830	6

Comparison alphaCertified is a C library based on α-theory for verifying the real solutions of polynomial equations[25]. Meanwhile, VerifyRealRoots is a Matlab package based on the Krawczyk-type interval operator for verifying the existence of real solutions of polynomial systems. In Table 3, we exhibit the performances of VerifyRealRoots and alphaCertified on some benchmark examples from the literature. Example *stewgou*40 in Table 3 is the polynomial system yielded from the Stewart-Gough platform[60]. The polynomial system *stewgou*40 and its 40 approximate real solutions are downloaded from the link `http://www.math.tamu.edu/~sottile/research/stories/alphaCertified/Stewart/index.html` given in [25]. The remaining examples are taken from Verschelde's test suite `http://www.math.uic.edu/~jan/`, and the associated input approximate real solutions yielded from Bertini solver. By using the same approximate solutions, we call VerifyRealRoots and alphaCertified to get the verified real solutions. Remark that for each problem we ran alphaCertified by using the default setting to compute the verified solutions. Here *var* and *deg* denote the number of the variables and the highest degree of polynomials, *approx-sol* denote the number of approximate real solutions yielded from Bertini solver. The columns VerifyRealRoots and alphaCertified contain the timing and the number of the verified real solutions by calling VerifyRealRoots and alphaCertified respectively.

Table 3 Comparison with alphaCertified on zero-dimensional polynomial systems

Ex	*var*	*deg*	*approx-sol*	VerifyRealRoots		alphaCertified	
				time(*s*)	*sol*	*time*(*s*)	*sol*
*stewgou*40	9	4	40	3.625	40	2.613	40
*cohn*3	4	6	23	0.484	7	0.518	2
*d*1	12	3	16	0.344	16	0.639	16
geneig	6	3	10	0.219	10	0.437	10
*katsura*5	6	2	12	0.109	11	0.352	12
*kin*1	12	3	16	0.359	16	0.664	16
rbpl	6	3	5	0.219	4	0.497	4
*geddes*2	5	6	7	0.281	7	0.364	2
*hairer*2	9	4	7	0.203	7	0.489	6
*assur*44	8	3	10	0.406	10	0.624	10
*chandra*6	6	2	31	0.344	31	0.712	31
*cyclic*9	9	9	234	18.203	234	37.34	216
*cyclic*10	10	10	680	80.797	680	675.092	627
*filter*9	9	5	134	2.797	128	5.472	128
*game*5*two*	5	4	10	0.266	10	0.486	10
*game*6*two*	6	5	36	2.813	35	2.014	34
*game*7*two*	7	6	111	27.453	100	37.253	100
*katsura*6	7	2	32	0.422	24	0.712	32
*katsura*7	8	2	44	0.953	42	1.107	44
*katsura*8	9	2	84	2.234	72	2.417	84
*pole*27*sys*	14	2	442	219.625	442	188.989	419
*rps*10	10	4	126	55.078	126	38.566	122
*tangents*2	6	2	24	0.359	24	0.651	24
utbikker	4	3	12	0.156	7	0.437	10

References

[1] Canny J, *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, USA, 1988.

[2] Fløystad G, Kileel J, Ottaviani G, The Chow form of the essential variety in computer vision, *Journal of Symbolic Computation*, 2017, **86**: 97–119.

[3] Petitjean S, Algebraic geometry and computer vision: Polynomial systems, real and complex roots, *J. Math. Imaging Vis.*, 1999, **10**(3): 191–220.

[4] Sommese A J and Wampler C W, *The Numerical Solution of Systems of Polynomials — Arising in Engineering and Science*, World Scientific, 2005, I-XXII: 1–401.

[5] Collins G E, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Lecture Notes in Computer Science*, 1975, **33**: 134–183.

[6] Basu S, Pollack R, and Roy M F, On the combinatorial and algebraic complexity of quantifier elimination, *Journal of ACM*, 1996, **43**(6): 1002–1045.

[7] Basu S, Pollack R, and Roy M F, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Springer Vienna, Vienna, Chapter A new algorithm to find a point in every cell defined by a family of polynomials, 1998, 341–350.

[8] Heintz J, Roy M F, and Solernó P, On the theoretical and practical complexity of the existential theory of reals, *The Computer Journal*, 1993, **36**(5): 427–431.

[9] Renegar J, On the computational complexity and geometry of the first-order theory of the reals. Part I: Introduction, Preliminaries, The geometry of semi-algebraic sets, The decision problem for the existential theory of the reals, *Journal of Symbolic Computation*, 1992, **13**(3): 255–299.

[10] Collins G E and Hong H, Partial cylindrical algebraic decomposition for quantifier elimination, *Journal of Symbolic Computation*, 1991, **12**(3): 299–328.

[11] Brown C W, QEPCAD B: A program for computing with semi-algebraic sets using CADs, *SIGSAM Bull.*, 2003, **37**(4): 97–108.

[12] Seidl A and Sturm T, A generic projection operator for partial cylindrical algebraic decomposition, *Proceedings of the 28th International Symposium on Symbolic and Algebraic Computation*, 2003, 240–247.

[13] Yanami H and Anai H, The maple package SyNRAC and its application to robust control design, *Future Generation Computer Systems*, 2007, **23**(5): 721–726.

[14] Xia B, DISCOVERER: A tool for solving semi-algebraic systems, *ACM Commun. Comput. Algebra*, 2007, **41**(3): 102–103.

[15] Xia B and Yang L, An algorithm for isolating the real solutions of semi-algebraic systems, *Journal of Symbolic Computation*, 2002, **34**(5): 461–477.

[16] Chen C and Maza M M, Quantifier elimination by cylindrical algebraic decomposition based on regular chains, *Journal of Symbolic Computation*, 2016, **75**: 74–93.

[17] Aubry P, Rouillier F, and Safey El Din M, Real solving for positive dimensional systems, *Journal of Symbolic Computation*, 2002, **34**(6): 543–560.

[18] Bank B, Giusti M, Heintz J, et al., Polar varieties and efficient real elimination, *Mathematische Zeitschrift*, 2001, **238**(1): 115–144.

[19] Bank B, Giusti M, Heintz J, et al., On the geometry of polar varieties, *Applicable Algebra in Engineering, Communication and Computing*, 2010, **21**(1): 33–83.

[20] Rouillier F, Roy M F, and Safey El Din M, Finding at least one point in each connected component of a real algebraic set defined by a single equation, *Journal of Complexity*, 2000, **16**(4): 716–750.

[21] Safey El Din M and Schost É, Polar varieties and computation of one point in each connected component of a smooth real algebraic set, *Proceedings of the* 28*th International Symposium on Symbolic and Algebraic Computation*, 2003, 224–231.

[22] Safey El Din M, Raglib (Real Algebraic Geometry Library), Maple Package, 2007.

[23] Beltrán C and Leykin A, Certified numerical homotopy tracking, *Experimental Mathematics*, 2012, **21**: 69–83.

[24] Beltrán C and Leykin A, Robust certified numerical homotopy tracking, *Foundations of Computational Mathematics*, 2013, **13**(2): 253–295.

[25] Hauenstein J D and Sottile F, Algorithm 921: AlphaCertified: Certifying solutions to polynomial systems, *ACM Trans. Math. Softw.*, 2012, **38**(4): 28:1–28:20.

[26] Shen F, Wu W, and Xia B, Real root isolation of polynomial equations based on hybrid computation, *Computer Mathematics: 9th Asian Symposium* (*ASCM* 2009), 2014, 375–396.

[27] Wang Y and Xia B, A Hybrid procedure for finding real points on a real algebraic set, *Journal of Systems Science and Complexity*, 2019, **32**(1): 185–204.

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

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

[30] Rump S M, Solving algebraic problems with high accuracy, *Proc. of the Symposium on a New Approach to Scientific Computation*, 1983, 51–120.

[31] Li N and Zhi L, Verified error bounds for isolated singular solutions of polynomial systems: Case of breadth one, *Theoretical Computer Science*, 2013, **479**: 163–173.

[32] Li N and Zhi L, Verified error bounds for isolated singular solutions of polynomial systems, *SIAM Journal on Numerical Analysis*, 2014, **52**(4): 1623–1640.

[33] Rump S M and Graillat S, Verified error bounds for multiple roots of systems of nonlinear equations, *Numerical Algorithms*, 2010, **54**(3): 359–377.

[34] Hauenstein J D, Mourrain B, and Szanto A, Certifying isolated singular points and their multiplicity structure, *Proceedings of the* 2015 *ACM on International Symposium on Symbolic and Algebraic Computation*, 2015, 213–220.

[35] Mantzaflaris A and Mourrain B, Deflation and certified isolation of singular zeros of polynomial systems, *Proceedings of the* 36*th International Symposium on Symbolic and Algebraic Computation*, 2011, 249–256.

[36] Rump S M, INTLAB - INTerval LABoratory, *Developments in Reliable Computing*, Springer, Dordrecht, 1999, 77–104.

[37] Akoglu T A, Hauenstein J D, and Szanto A, Certifying solutions to overdetermined and singular polynomial systems over Q, *Journal of Symbolic Computation*, 2018, **84**: 147–171.

[38] Chen X, Frommer A, and Lang B, computational existence proofs for spherical T-designs, *Numer. Math.*, 2011, **117**(2): 289–305.

[39] Chen X and Womersley R S, Existence of solutions to systems of underdetermined equations and spherical designs, *SIAM J. NUMER. ANAL.*, 2006, **44**(6): 2326–2341.

[40] Sommese A J and Verschelde J, Numerical homotopies to compute generic points on positive dimensional algebraic sets, *Journal of Complexity*, 2000, **16**(3): 572–602.

[41] Sommese A J, Verschelde J, and Wampler C W, Numerical algebraic geometry, *The Mathematics of Numerical Analysis* (*Park City, UT,* 1995), 1996, 749–763.

[42] Everett H, Lazard D, Lazard S, et al., The voronoi diagram of three lines, *Proceedings of the Twenty-Third Annual Symposium on Computational Geometry*, 2007, 255–264.

[43] Yang Z, Zhi Li, and Zhu Y, Verified error bounds for real solutions of positive-dimensional polynomial systems, *Proceedings of the* 38*th International Symposium on Symbolic and Algebraic Computation*, 2013, 371–378.

[44] Graziano C, Garulli A, Tesi A, et al., Characterizing the solution set of polynomial systems in terms of homogeneous forms: An LMI approach, *International Journal of Robust and Nonlinear Control*, 2003, **13**(13): 1239–1257.

[45] Henrion D and Lasserre J B, Detecting global optimality and extracting solutions in GloptiPoly, *Positive Polynomials in Control*, 2005, **312**: 293–310.

[46] Lasserre J B, Laurent M, and Rostalski P, Semidefinite characterization and computation of zero-dimensional real radical ideals, *Foundations of Computational Mathematics*, 2008, **8**: 607–647.

[47] Lasserre J B, *Moments, Positive Polynomials and Their Applications*, Imperial College Press, London, 2009.

[48] Ma Y and Zhi L, Computing real solutions of polynomial systems via low-rank moment matrix completion, *Proceedings of the* 37*th International Symposium on Symbolic and Algebraic Computation*, 2012, 249–256.

[49] Leykin A, Verschelde J, and Zhao A, Newton's method with deflation for isolated singularities of polynomial systems, *Theoretical Computer Science*, 2006, **359**(1): 111–122.

[50] Greuet A and Safey El Din M, Deciding reachability of the infimum of a multivariate polynomial, *Proceedings of the* 36*th International Symposium on Symbolic and Algebraic Computation*, 2011, 131–138.

[51] Rouillier F, Efficient algorithms based on critical points method, *Algorithmic and Quantitative Real Algebraic Geometry*, American Mathematical Society, Providenc, 2003, 123–138.

[52] Bank B, Giusti M, Heintz J, et al., Algorithms of intrinsic complexity for point searching in compact real singular hypersurfaces, *Foundations of Computational Mathematics*, 2012, **12**: 75–122.

[53] Mork H C and Piene R, Polars of real singular plane curves, *Algorithm in Algebraic Geometry*, Springer, New York, 2008, 99–115.

[54] Hauenstein J D, Numerically computing real points on algebraic sets, *Acta Applicandae Mathematicae*, 2013, **125**(1): 105–119.

[55] Bates D J, Hauenstein J D, Sommese A J, et al., Bertini: Software for Numerical Algebraic Geometry, `http://www.nd.edu/~sommese/bertini`, 2018.

[56] Li N and Zhi L, Computing isolated singular solutions of polynomial systems: Case of breadth one, *SIAM Journal on Numerical Analysis*, 2012, **50**(1): 354–372.

[57] Bachoc C and Vallentin F, New upper bounds for kissing numbers from semidefinite programming, *Journal of the American Mathematical Society*, 2008, **21**(3): 909–924.

[58] Palancz B, Application of Dixon resultant to satellite trajectory control by pole placement, *Journal of Symbolic Computation*, 2013, **50**: 79–99.

[59] Verschelde J and Wang Y, Computing dynamic output feedback laws, *IEEE Trans. Automatic Control*, 2004, **49**(8): 1393–1397.

[60] Dietmater P, The Stewart-Gough platform of general geometry can have 40 real postures, *Advances in Robot Kinematics: Analysis and Control*, 1998, 1–10.