Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 15, No. 1, pp. 157 – 171, 2019

FINDING ALL REAL SOLUTIONS OF POLYNOMIAL EQUATIONS*

Xin Zhao and Jinyan Fan†

Abstract: In this paper, we study how to find all real solutions of polynomial equations, if there are finitely many ones. We propose a semidefinite relaxation algorithm for computing them sequentially. Each solution can be obtained by solving a hierarchy of semidefinite relaxations. Convergence properties of the algorithm are also discussed.

Key words: *polynomial equations, polynomial optimization, semidefinite relaxation*

Mathematics Subject Classification: *49M20, 65H04, 90C22*

1 Introduction

In this paper, we consider the system of equations

$$h_i(x) = 0, \quad i = 1, \ldots, m, \tag{1.1}$$

where $h_i : \mathbb{R}^n \to \mathbb{R} (i = 1, \ldots, m)$ are polynomials. Solving polynomial equations is a classic problem in mathematics. They arise naturally in many areas including optimization, coding theory, economics, graph theory, computational statistics, cryptography, etc. (cf. [4, 17, 23]).

Current methods for polynomial equations can be classified into symbolic, geometric and numeric methods. Symbolic methods, based on resultants and Gröbner bases, can eliminate variables, thereby reduce the problem to finding the solutions of univariate polynomials. Geometric formulations of a problem have been used to develop several algorithms, for example, subdivision-based algorithms for intersections and ray tracing of curves and surfaces. Numeric techniques for solving polynomial equations are based on either iterative or homotopy methods (cf. [8, 18]). For example, the Newton method, quasi-Newton methods, etc. However, these methods usually can not find all solutions nor to prove that there is no solution.

In many practical applications, people are only interested in real solutions. Computing real solutions to a system of polynomial equations is a challenging problem, particularly verifying that all solutions have been computed. For a polynomial equation of degree d over the complex domain, the Fundamental Theorem of Algebra tells us that there are d

*The authors are partially supported by NSFC 11571234.
†Corresponding author

solutions, assuming that the solutions are counted by multiplicity. Wright [28] extended the result to a system that has as many equations as complex variables. It was shown that the number of solutions to such a system is the product of the degrees of the equations, if the solutions are counted by multiplicity.

Brake, Hauenstein and Liddell [3] combined numerical algebraic geometry and sums of squares programming for computing all real solutions of (1.1). They used sums of squares programming to validate that a "complete" real solution set has been computed, that is, the Zariski closure of the given set is equal to the Zariski closure of the set of all real solutions. The method combines numerical and symbolic methods. Lasserre, Laurent and Rostalski [15] used numerical linear algebra and semidefinite optimization techniques to compute all (finitely many) solutions of the polynomial equations. The algorithm is based on moment relaxations and, in contrast to other existing methods, it exploits the real algebraic nature and avoides the computation of complex components. They also gave a real algebraic symbolic-numeric algorithm, assuming the solution set is finite [14].

Recently, Cui, Dai and Nie [6] studied $\mathcal{B}$-eigenvalues of symmetric tensors. They proposed a semidefinite relaxation approach for computing all real $\mathcal{B}$-eigenvalues sequentially, from the largest to the smallest. Each eigenvalue can be computed by solving a finite hierarchy of semidefinite relaxations. The approach was originally proposed by Nie [22] for computing the hierarchy of local minimums for polynomial optimization.

In this paper, we follow the approach in [6, 22] to compute all real solutions of polynomial equations (1.1), if there are finitely many ones. We formulate it as a sequence of polynomial optimization problems. The real solutions are computed in order, by choosing a random objective. Each of them can be obtained by solving a hierarchy of semidefinite relaxations.

This paper is organized as follows. In Section 2, we review some basics in polynomial optimization. In Section 3, we show how to formulate (1.1) as a sequence of polynomial optimization problems. All real solutions are computed sequentially. A semidefinite relaxation algorithm is proposed. The convergence properties of the algorithm are also discussed. In Section 4, some numerical experiments are given. Finally, we conclude the paper in Section 5.

2 Preliminaries

Notations. The symbol $\mathbb{R}$ (resp., $\mathbb{N}$) denotes the set of real number (resp., nonnegative integers). $\mathbb{R}^n$ (resp., $\mathbb{N}^n$) denotes the set of all real (resp., nonnegative integers) n-dimensional vectors. The symbol $\mathbb{R}[x] = \mathbb{R}[x_1, \ldots, x_n]$ denotes the ring of polynomials in $x := (x_1, \ldots, x_n)$ over the real field. The cardinality of a set S is denoted as $|S|$. For $k \in \mathbb{R}$, $\lceil k \rceil$ denotes the smallest integer not smaller than k. For a symmetric matrix X, $X \succeq 0$ means X is positive semidefinite. For a vector y, $\|y\|$ denotes its standard Euclidean norm.

In this section, we review some basics in polynomial optimization. We refer to [12, 13, 16] for more details.

Given $h_1, \ldots, h_m \in \mathbb{R}[x]$ and a tuple $h = (h_1, \ldots, h_m)$. The *ideal* generated by h is

$$I(h) = h_1 \cdot \mathbb{R}[x] + \cdots + h_m \cdot \mathbb{R}[x].$$

The k-th truncation of $I(h)$, denoted as $I_k(h)$, is the set

$$I_k(h) = h_1 \cdot \mathbb{R}[x]_{k-deg(h_1)} + \cdots + h_m \cdot \mathbb{R}[x]_{k-deg(h_m)},$$

where $\mathbb{R}[x]_k$ is the set of polynomials in $\mathbb{R}[x]$ with degree at most k.

A polynomial $f \in \mathbb{R}[x]$ is said to be *sum of squares (SOS)* if $f = \sigma_1^2 + \cdots + \sigma_t^2$ for some $\sigma_1, \ldots, \sigma_t \in \mathbb{R}[x]$. The set of all SOS polynomials in x is denoted as $\Sigma[x]$. The k-th truncation of $\Sigma[x]$ is denoted as $\Sigma[x]_k = \Sigma[x] \cap \mathbb{R}[x]_k$. The *quadratic module* of a tuple $g = (g_1, \ldots, g_t)$ is the set

$$Q(g) = \Sigma[x] + g_1 \cdot \Sigma[x] + \cdots + g_t \cdot \Sigma[x].$$

The k-th truncation of the quadratic module $Q(g)$, denoted as $Q(k)_g$, is the set

$$Q(k)_g = \Sigma[x]_{2k} + g_1 \cdot \Sigma[x]_{2k-deg(g_1)} + \cdots + g_t \cdot \Sigma[x]_{2k-deg(g_t)}.$$

The set $I(h) + Q(g)$ is said to be *archimedean* if there exists $N > 0$ such that $N - \|x\|^2 \in I(h) + Q(g)$. For the tuples h and g as above, denote

$$E(h) := \{x \in \mathbb{R}^n : h(x) = 0\}, \quad S(g) := \{x \in \mathbb{R}^n : g(x) \geq 0\}.$$

Clearly, if $I(h) + Q(g)$ is archimedean, then $E(h) \cap S(g)$ is compact. On the other hand, if $E(h) \cap S(g)$ is compact, then $I(h) + Q(g)$ can be forced to be archimedean by adding the polynomial $R - \|x\|^2$ to the tuple g, for R sufficiently large. If $f \in I(h) + Q(g)$, then $f \geq 0$ on the set $E(h) \cap S(g)$. Conversely, if $f > 0$ on $E(h) \cap S(g)$ and $I(h) + Q(g)$ is archimedean, then $f \in I(h) + Q(g)$. This is called Putinar's Positivstellensatz (cf. [24]) in the literature.

For $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, denote $|\alpha| = \alpha_1 + \cdots + \alpha_n$. Let

$$\mathbb{N}^n_d = \{\alpha \in \mathbb{N}^n : |\alpha| \leq d\}.$$

Let $\mathbb{R}^{\mathbb{N}^n_d}$ be the space of real vectors indexed by $\alpha \in \mathbb{N}^n_d$. A vector in $\mathbb{R}^{\mathbb{N}^n_d}$ is called a *truncated moment sequence (tms)* of degree d. For $y \in \mathbb{R}^{\mathbb{N}^n_d}$, define a Riesz function $\mathcal{F}_y$ acting on $\mathbb{R}[x]_d$ as

$$\mathcal{F}_y(q) := \sum_{\alpha \in \mathbb{N}^n_d} q_\alpha y_\alpha, \quad \text{for all } q(x) = \sum_{\alpha \in \mathbb{N}^n_d} q_\alpha x^\alpha,$$

where $x^\alpha = x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ and (q_α) denotes the coefficient vector of the polynomial q. For convenience, we also denote $\langle q, y \rangle := \mathcal{F}_y(q)$. We say that y admits a representing measure supported in a set T if there exists a Borel measure μ such that its support $supp(\mu)$ is contained in T and

$$y_\alpha = \int_T x^\alpha d\mu, \quad \forall \alpha \in \mathbb{N}^n_d.$$

Denote

$$[x]_d = [1, x_1, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_1^d, \ldots, x_n^d]^T.$$

Define the symmetric matrices $A_\alpha^{(k)}$ such that

$$q(x)[x]_d[x]_d^T = \sum_{\alpha \in \mathbb{N}^n_{2k}} A_\alpha^{(k)} x^\alpha,$$

where $q \in \mathbb{R}[x]$ with $\deg(q) \leq 2k$ and $d = k - \lceil deg(q)/2 \rceil$. The *k-th order localizing matrix* of q, generated by $y \in \mathbb{R}^{\mathbb{N}^n_{2k}}$, is the symmetric matrix $L_q^{(k)}(y)$ satisfying

$$L_q^{(k)}(y) = \sum_{\alpha \in \mathbb{N}^n_{2k}} A_\alpha^{(k)} y_\alpha. \tag{2.1}$$

When $q = 1$, we call it a *moment matrix*, denoted as $M_k(y)$. We also denote

$$L_h^{(k)}(y) = diag(L_{h_1}^{(k)}(y), \ldots, L_{h_m}^{(k)}(y)), \tag{2.2}$$

for the tuple $h = (h_1, \ldots, h_m)$.

For the polynomial tuples h and g given as above, people are often interested in whether or not a tms $y \in \mathbb{R}^{\mathbb{N}^n_{2k}}$ admits a representing measure whose support is contained in $E(h) \cap S(g)$. Indeed, if

$$L_h^{(k)}(y) = 0, \quad M_k(y) \succeq 0, \quad L_g^{(k)}(y) \succeq 0, \tag{2.3}$$

moreover, if y also satisfies the rank condition

$$\text{rank } M_{k-d'}(y) = \text{rank } M_k(y), \tag{2.4}$$

where $d' = \max\{1, \lceil deg(h)/2 \rceil, \lceil deg(g)/2 \rceil\}$, then y admits a measure supported in $E(h) \cap S(g)$ (cf. [7, 10]). In such case, y admits a unique finitely atomic measure on $E(h) \cap S(g)$. We call that y is *flat* with respect to $h = 0$ and $g \geq 0$ if both (2.3) and (2.4) are satisfied.

For $y \in \mathbb{R}^{\mathbb{N}^n_d}$ and $t \leq d$, denote the truncation of y as

$$y|_t = (y_\alpha)_{\alpha \in \mathbb{N}^n_t}.$$

For two tms' $y \in \mathbb{R}^{\mathbb{N}^n_k}$ and $z \in \mathbb{R}^{\mathbb{N}^n_l}$ with $k < l$, we say that y is a truncation of z (equivalently, z is an extension of y), if $y = z|_k$. For such case, y is called a *flat truncation* of z if y is flat, and z is a *flat extension* of y if z is flat. Flat extensions and flat truncations are useful in solving polynomial optimization and truncated moment problems (cf. [20, 21]).

3 Finding All Real Solutions of Polynomial Equations

Denote the set of all real solutions of the polynomial equations (1.1) by

$$V_{\mathbb{R}}(h) = \{x \in \mathbb{R}^n : h_i(x) = 0, i = 1, \ldots, m\}.$$

We assume that $V_{\mathbb{R}}(h)$ is finite throughout the paper. In this section, we study how to compute all elements of $V_{\mathbb{R}}(h)$. We propose a semidefinite algorithm for computing them sequentially. Each of them can be obtained by solving a hierarchy of semidefinite relaxations.

Let f be a random polynomial in $\mathbb{R}[x]$. Consider the optimization problem:

$$\begin{array}{ll} \min & f(x) \\ s.t. & h_i(x) = 0, \quad i = 1, \ldots, m. \end{array} \tag{3.1}$$

Clearly, $x \in V_{\mathbb{R}}(h)$ if and only if x is feasible for (3.1). When it is generically chosen, f achieves different values at different $x \in V_{\mathbb{R}}(h)$. We order them monotonically as

$$f_1 < f_2 < \cdots < f_s.$$

Let

$$V_j = \{x \in V_{\mathbb{R}}(h) : f(x) = f_j\}, \quad j = 1, \ldots, s. \tag{3.2}$$

Then,

$$V_{\mathbb{R}}(h) = V_1 \cup \cdots \cup V_s. \tag{3.3}$$

3.1 The first set V_1

By (3.2), the value f_1 is equal to the optimal value of the optimization problem

$$\begin{array}{lll} f_1 = \min & f(x) \\ & s.t. & h_i(x) = 0, \quad i = 1, \ldots, m. \end{array} \tag{3.4}$$

We apply Lasserre type semidefinite relaxations (cf. [12]) to solve (3.4). The k-th order semidefinite relaxation of (3.4) is

$$\begin{array}{rll} r_1^{(k)} = \min & \langle f, y\rangle & \\ s.t. & \langle 1, y\rangle = 1, & \\ & L_h^{(k)}(y) = 0, & \\ & M_k(y) \succeq 0, y \in \mathbb{R}^{\mathbb{N}^n_{2k}}, & \end{array} \tag{3.5}$$

for $k = k_0, k_0+1, \ldots$, where $k_0 = \max\{\lceil deg(f)/2\rceil, \lceil deg(h_1)/2\rceil, \ldots, \lceil deg(h_m)/2\rceil\}$. In (3.5), $\langle 1, y\rangle = 1$ means that the first entry of y is one, and the matrices $M_k(y), L_h^{(k)}(y)$ are defined as in (2.1)–(2.2).

The dual problem of (3.5) is

$$\begin{array}{rll} d_1^{(k)} = \max & \gamma & \\ s.t. & f - \gamma \in I_{2k}(h). & \end{array} \tag{3.6}$$

Suppose $y^{(1,k)}$ is a minimizer of (3.5). By weak duality, $r_1^{(k)} \geq d_1^{(k)}$ for all k. If, for some $t \in [k_0, k]$, the truncation $\hat{y} = y^{(1,k)}|_{2t}$ satisfies

$$\text{rank } M_{t-k_0}(\hat{y}) = \text{rank } M_t(\hat{y}), \tag{3.7}$$

then $r_1^{(k)} = f_1$ and we can obtain γ_1 global minimizers of (3.4), where $\gamma_1 = \text{rank } M_t(\hat{y})$ (cf. [21]). Thus, we obtain V_1.

Theorem 3.1. *Suppose that $V_{\mathbb{R}}(h)$ is finite. For problems* (3.4)–(3.6)*, we have the following results:*

(i) *If the problem* (3.5) *is infeasible for some k, then $V_{\mathbb{R}}(h) = \varnothing$.*

(ii) *If $V_{\mathbb{R}}(h) = \varnothing$, then* (3.5) *must be infeasible for some k.*

(iii) *If $V_{\mathbb{R}}(h)$ is nonempty, then for all k sufficiently large,*

$$r_1^{(k)} = d_1^{(k)} = f_1$$

and the condition (3.7) *must be satisfied.*

Proof. (i) Note that (3.5) is a relaxation of (3.4). If (3.5) is infeasible, then (3.4) is infeasible. So, (i) is true.

(ii) Since $V_{\mathbb{R}}(h) = \varnothing$, by the Positivstellensatz (cf. [2, Theorem 4.4.2]), there exists $\phi = -1 \in I(h)$ where $I(h)$ is the ideal generated by the tuple h. Hence, we have $\phi \in I_{2k}(h)$ for all k sufficiently large. This implies that (3.6) has an improving direction (that is, the direction is feasible and can make the objective function increase along it) and it is unbounded from the above. By weak duality, the relaxation (3.5) must be infeasible for k sufficiently large.

(iii) Since $V_{\mathbb{R}}(h)$ is finite and nonempty, we have $|V_{\mathbb{R}}(h)| < \infty$ and $V_{\mathbb{R}}(h) \neq \varnothing$. Let $d_h = \max_{j=1,\ldots,m}\{deg(h_j)/2\}$. By [15, Proposition 4.6], for k sufficiently large, there exists $s \in [d_h, k]$ such that

$$\text{rank } M_s(y) = \text{rank } M_{s-d_h}(y)$$

for all $y \in \Omega_k$, where

$$\Omega_k = \{y \in \mathbb{R}^{\mathbb{N}^n_{2k}} : \langle 1, y\rangle = 1, M_k(y) \succeq 0, L_{h_i}^{(k)}(y) = 0, i = 1, \ldots, m\}.$$

Since $V_{\mathbb{R}}(h)$ is finite, by [20, Theorem 1.1], we have $d_1^{(k)} = f_1$ for all k sufficiently large. So, when the relaxation order k is sufficiently large, we have

$$r_1^{(k)} = d_1^{(k)} = f_1$$

and the rank condition (3.7) must be satisfied. □

3.2 The second and other sets V_j

Suppose f_j and V_j are already known. We investigate how to compute f_{j+1} and V_{j+1} for $j = 1, \ldots, s-1$. Consider the optimization problem

$$\begin{array}{rll} f_j^+ = \min & f(x) & \\ s.t. & h_i(x) = 0, & i = 1, 2, \ldots, m, \\ & f(x) \geq f_j + \delta. & \end{array} \tag{3.8}$$

Clearly, $f_{j+1} = f_j^+$ if

$$0 < \delta < f_{j+1} - f_j. \tag{3.9}$$

Similarly, the k-th order Lasserre relaxation of (3.8) is

$$\begin{array}{rl} r_{j+1}^{(k)} = \min & \langle f, y \rangle \\ s.t. & \langle 1, y \rangle = 1, \\ & L_h^{(k)}(y) = 0, \\ & M_k(y) \succeq 0, \\ & L_{f-f_j-\delta}^{(k)}(y) \succeq 0, y \in \mathbb{R}^{\mathbb{N}_{2k}^n}, \end{array} \tag{3.10}$$

for $k = k_0, k_0 + 1, \ldots$. The dual problem of (3.10) is

$$\begin{array}{rl} d_{j+1}^{(k)} = \max & \gamma \\ s.t. & f - \gamma \in I_{2k}(h) + Q_k(f - f_j - \delta). \end{array} \tag{3.11}$$

By weak duality, $r_{j+1}^{(k)} \geq d_{j+1}^{(k)}$ for all k. Suppose $y^{(j+1,k)}$ is a minimizer of (3.10). If, for some $t \in [k_0, k]$, the truncation $\hat{y} = y^{(j+1,k)}|_{2t}$ satisfies the rank condition (3.7), then $r_{j+1}^{(k)} = f_{j+1}$ and we can obtain γ_{j+1} global minimizers of (3.8), where $\gamma_{j+1} = \operatorname{rank} M_t(\hat{y})$ (cf. [21]). Thus, we obtain V_{j+1}.

In practice, we typically do not know whether f_{j+1} exists. Even if it exists, its value is typically not available. So, we need to determine the value of δ that satisfies $0 < \delta < f_{j+1} - f_j$. Consider the optimization problem:

$$\begin{array}{rll} f_j^- = \max & f(x) & \\ s.t. & h_i(x) = 0, & i = 1, 2, \ldots, m, \\ & f(x) \leq f_j + \delta. & \end{array} \tag{3.12}$$

Its optimal value f_j^- can be computed by the semidefinite relaxations similar to (3.10) and (3.11).

Lemma 3.2. *Suppose that $V_{\mathbb{R}}(h)$ is finite. Let f_j be the value of $f(x)$ on V_j. For all $j > 1$ and $\delta > 0$, we have*

(i) *If f_{j+1} exists, then $f_j^- = f_j$ if and only if δ satisfies* (3.9).

(ii) *If $f_j^- = f_j$ for some $\delta > 0$ and* (3.10) *is infeasible for some k, then f_{j+1} does not exist, that is, f_j is the maximum value of f on $V_{\mathbb{R}}(h)$.*

(iii) *If f_j is the maximum value of f on $V_{\mathbb{R}}(h)$, then $f_j^- = f_j$ for any $\delta > 0$.*

Proof. (i) If $0 < \delta < f_{j+1} - f_j$, by (3.12), the result is obvious. Conversely, if $f_j^- = f_j$, then the maximum value of f on $V_{\mathbb{R}}(h)$, which is not larger than $f_j + \delta$, is still f_j. So, if f_{j+1} exists, we must have $f_{j+1} > f_j + \delta$.

(ii) If (3.10) is infeasible for some k, then (3.8) is infeasible. This implies that, for any x satisfying $h_i(x) = 0$ $(i = 1, \ldots, m)$, $f(x) < f_j + \delta$. By $f_j^- = f_j$, we know that f_{j+1} does not exist. So, f_j is the maximum value of f on $V_{\mathbb{R}}(h)$.

(iii) If f_j is the maximum value of f on $V_{\mathbb{R}}(h)$, then, for any x satisfying $h_i(x) = 0$ $(i = 1, \ldots, m)$, $f(x) \leq f_j$. Hence, $f_j^- = f_j$ for any $\delta > 0$. □

3.3 An algorithm for computing $V_{\mathbb{R}}(h)$

We propose a semidefinite relaxation algorithm to compute $V_{\mathbb{R}}(h)$. Firstly, we compute V_1 by solving (3.4), if $V_{\mathbb{R}}(h)$ is nonempty. After obtaining V_1, we solve (3.8). If f_2 does not exist, then V_1 is the solution set of (1.1), and we stop. Otherwise, we determine f_2 and V_2. Repeating this procedure, we can obtain the solution set $V_{\mathbb{R}}(h) = V_1 \cup \cdots \cup V_s$.

The algorithm is presented as follows.

Algorithm 3.3.

Step 0. Choose a random polynomial $f(x)$. Let $j := 0$ and $k := k_0$.

Step 1. If (3.5) is infeasible, then $V_{\mathbb{R}}(h) = \varnothing$ and stop. Otherwise, solve (3.5) to obtain a minimizer $y^{(1,k)}$ and the optimal value $r_1^{(k)}$.

Step 2. If (3.7) is satisfied for some $t \in [k_0, k]$, then let $V := V_1$, where V_1 is the set of minimizers of (3.4). Let $F = \{r_1^{(k)}\}$, $k := k_0$, $j := j + 1$, and go to Step 3. If such t does not exist, let $k := k + 1$ and go to Step 1.

Step 3. Let $\delta = 0.05$. Compute the optimal value f_j^- of (3.12). If $f_j^- > f_j$, let $\delta = \delta/2$ and compute (3.12) again. Repeat this procedure until we obtain $f_j^- = f_j$.

Step 4. Solve (3.10). If it is infeasible, then $h(x) = 0$ has no more real solutions, let $V_{\mathbb{R}}(h) = V$ and stop. Otherwise, compute a minimizer $y^{(j+1,k)}$ and the optimal value $r_{j+1}^{(k)}$.

Step 5. If (3.7) is satisfied for some $t \in [k_0, k]$, then update $V := V \cup V_{j+1}$, where V_{j+1} is the set of minimizers of (3.8). Let $F := F \cup \{r_{j+1}^{(k)}\}$, $k := k_0$, $j := j + 1$, and go to Step 3. If such t does not exist, let $k := k + 1$ and go to Step 4.

Algorithm 3.3 has the following nice convergence properties.

Theorem 3.4. *Suppose $V_{\mathbb{R}}(h)$ is finite and nonempty. Let f_j be the value of $f(x)$ on $V_j (j \geq 1)$. Then, we have:*

(i) *If, for any $\delta > 0$,* (3.10) *is infeasible for some k, then the maximum value on $V_{\mathbb{R}}(h)$ is f_j.*

(ii) *If the maximum value on $V_{\mathbb{R}}(h)$ is f_j, then* (3.10) *must be infeasible for some k, for any $\delta > 0$.*

(iii) *Suppose f_{j+1} exists and $0 < \delta < f_{j+1} - f_j$, then for all k sufficiently large,*

$$r_{j+1}^{(k)} = d_{j+1}^{(k)} = f_{j+1}$$

and the condition (3.7) *must be satisfied.*

Proof. (i) Note that (3.10) is a relaxation of (3.8). If (3.10) is infeasible, then (3.8) is infeasible. So, (i) is true.

(ii) If the maximum value of $f(x)$ on $V_{\mathbb{R}}(h)$ is f_j, then the feasible set of (3.8) is empty for any $\delta > 0$. So, by the Positivstellensatz (cf. [2] Theorem 4.4.2), there exist $\phi \in I(h)$ and $\psi \in Pr(f - f_j - \delta)$ such that

$$\phi + \psi = -2,$$

where $I(h)$ is the ideal generated by the tuple h and $Pr(f - f_j - \delta)$ denotes the preordering generated by the polynomial $(f - f_j - \delta)$. (We refer to [2] for preorderings). It is a convex cone of $\mathbb{R}[x]$.

Since $V_{\mathbb{R}}(h)$ is finite, $-\|h\|^2 \in I(h)$ and $\{x \in \mathbb{R}^n : -\|h\|^2 \geq 0\}$ is compact. So, $I(h) + Q(f - f_j - \delta)$ is archimedean. Note that, $1 + \psi$ is strictly positive on the set $\{x \in \mathbb{R}^n : h = 0, f - f_j - \delta \geq 0\}$, then, by Putinar's Positivstellensatz (cf. [24]),

$$1 + \psi \in I(h) + Q(f - f_j - \delta),$$

that is, there exist $\sigma = 1 + \psi \in Q_k(f - f_j - \delta)$ and $\phi \in I_{2k}(h)$ such that

$$\phi + \sigma = -1$$

for all k sufficiently large. This implies that (3.11) has an improving direction and it is unbounded from the above. By weak duality, the relaxation (3.10) must be infeasible for k sufficiently large.

(iii) It is clear that if $0 < \delta < f_{j+1} - f_j$ holds, the optimal value of (3.8) is equal to f_{j+1}, if f_{j+1} exists.

When $V_{\mathbb{R}}(h)$ is finite, the results can be derived from [15, Proposition 4.6] and [20, Theorem 1.1]. The proof is similar to that of Theorem 3.1 (iii). □

Remark 3.5. In theory, the objective function f can be any random polynomial in $\mathbb{R}[x]$. However, in practice, we may avoid those f with high degree. As shown in (3.5) and (3.10), we use the Lasserre's SDP relaxation to solve the polynomial optimization problem. The high degree of f produces a higher order relaxation problem, which may not be favorable. So, a random f with low degree is preferred.

4 Numerical Experiments

In this section, we present some numerical experiments for solving polynomial equations (1.1) by Algorithm 3.3. We use the software GloptiPoly 3 [11] and SeDuMi [26] to solve (3.5) and (3.10). The experiments are implemented on a laptop with an Intel Core i7 CPU (2.2GHz) and 16GB of RAM, using Matlab R2015a. We display 4 decimal digits for numerical numbers.

Example 4.1 ([5]). Consider the polynomial equations:

$$\begin{aligned} h_1 &= x_2^4 x_1 + 3x_1^3 - x_2^4 - 3x_1^2, \\ h_2 &= x_1^2 x_2 - 2x_1^2, \\ h_3 &= 2x_2^4 x_1 - x_1^3 - 2x_2^4 + x_1^2. \end{aligned}$$

We choose the objective function as $f = x_1^2 + x_2^2$. By Algorithm 3.3, we obtain two real solutions:

$$\begin{aligned} v_1 &= (-0.0000, 0.0000)^T, \\ v_2 &= (1.0000, 2.0000)^T. \end{aligned}$$

The computation takes about 1.1 seconds.

Example 4.2 ([1]). Consider the polynomial equations:

$$\begin{aligned} h_1 &= 5x_1^9 - 6x_1^5 x_2 + x_1 x_2^4 + 2x_1 x_3, \\ h_2 &= -2x_1^6 x_2 + 2x_1^2 x_2^3 + 2x_2 x_3, \\ h_3 &= x_1^2 + x_2^2 - 0.265625. \end{aligned}$$

They have 20 complex solutions, among which eight are real.

We choose the objective function as $f = x_1 + x_2 + x_3$. By Algorithm 3.3, we obtain all real solutions:

$$\begin{aligned} v_1 &= (-0.5153, -0.0001, -0.0124)^T, \\ v_2 &= (-0.0002, -0.5152, -0.0000)^T, \\ v_3 &= (-0.5016, 0.1185, 0.0124)^T, \\ v_4 &= (-0.2619, 0.4439, -0.0132)^T, \\ v_5 &= (0.5153, 0.0001, -0.0124)^T, \\ v_6 &= (0.0000, 0.5154, -0.0000)^T, \\ v_7 &= (0.5016, 0.1185, 0.0124)^T, \\ v_8 &= (0.2619, 0.4439, -0.0132)^T. \end{aligned}$$

The computation takes about 59 seconds.

Example 4.3 ([27]). Consider the polynomial equations:

$$\begin{aligned} h_1 &= x_1 + x_2 - 2, \\ h_2 &= x_1 x_3 + x_2 x_4, \\ h_3 &= x_1 x_3^2 + x_2 x_4^2 - 2/3, \\ h_4 &= x_1 x_3^3 + x_2 x_4^3, \end{aligned}$$

which represents a Gaussian quadrature formula with two weights and two knots.

We choose the objective function as $f = x_1^2 + x_2^2 + x_3^2 + x_4^2$. By Algorithm 3.3, we obtain two real solutions:

$$\begin{aligned} v_1 &= (1.0000, 1.0000, -0.5774, 0.5774)^T, \\ v_2 &= (1.0000, 1.0000, 0.5774, -0.5774)^T. \end{aligned}$$

The computation takes about 3.6 seconds.

Example 4.4 ([30])**.** Consider the polynomial equations Katsura5:

$$\begin{aligned}
h_1 &= 2x_6^2 + 2x_5^2 + 2x_4^2 + 2x_3^2 + 2x_2^2 + x_1^2 - x_1,\\
h_2 &= x_6x_5 + x_5x_4 + 2x_4x_3 + 2x_3x_2 + 2x_2x_1 - x_2,\\
h_3 &= 2x_6x_4 + 2x_5x_3 + 2x_4x_2 + x_2^2 + 2x_3x_1 - x_3,\\
h_4 &= 2x_6x_3 + 2x_5x_2 + 2x_3x_2 + 2x_4x_1 - x_4,\\
h_5 &= x_3^2 + 2x_6x_1 + 2x_5x_1 + 2x_4x_1 - x_5,\\
h_6 &= 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + x_1 - 1.
\end{aligned}$$

We can choose the objective function as $f = x_1^2 + \cdots + x_6^2$. By Algorithm 3.3, we obtain twelve real solutions:

$$\begin{aligned}
v_1 &= (0.1362, 0.0428, 0.0417, 0.0404, 0.0964, 0.2106)^T,\\
v_2 &= (0.2386, 0.0608, -0.0622, -0.0233, 0.1862, 0.2192)^T,\\
v_3 &= (0.2772, 0.2259, 0.1621, 0.0858, 0.0115, -0.1240)^T,\\
v_4 &= (0.2919, -0.1011, 0.1805, -0.0591, 0.1929, 0.1409)^T,\\
v_5 &= (0.4086, -0.0732, 0.0657, -0.1266, 0.2521, 0.1777)^T,\\
v_6 &= (0.4411, 0.1515, 0.0226, 0.2192, 0.0935, -0.2073)^T,\\
v_7 &= (0.4616, 0.3087, 0.0553, -0.1020, -0.0844, 0.0917)^T,\\
v_8 &= (0.5903, 0.0422, 0.3274, -0.0642, -0.0874, -0.0132)^T,\\
v_9 &= (0.6798, 0.2657, -0.1541, 0.0323, 0.0896, -0.0735)^T,\\
v_{10} &= (0.7263, -0.0503, 0.1220, 0.1636, 0.1095, -0.2079)^T,\\
v_{11} &= (0.7534, 0.0532, 0.1909, -0.1144, -0.1456, 0.1391)^T,\\
v_{12} &= (1.0000, 0.0000, 0.0000, -0.0000, -0.0000, -0.0000)^T.
\end{aligned}$$

The computation takes about 25 seconds.

Example 4.5 ([15])**.** Consider the polynomial equations:

$$\begin{aligned}
h_1 &= x_1^2 + x_2 + x_3 + 1,\\
h_2 &= x_1 + x_2^2 + x_3 + 1,\\
h_3 &= x_1 + x_2 + x_3^2 + 1,
\end{aligned}$$

which admit seven complex solutions, among which the real solution $(-1, -1, -1)^T$ has multiplicity two.

We choose the objective function as $f = x_1^2 + x_2^2 + x_3^2$. By Algorithm 3.3, we obtain the real solutions

$$\begin{aligned}
v_1 &= (-0.9999, -0.9999, -0.9999)^T,\\
v_2 &= (-1.0090, -1.0090, -1.0090)^T.
\end{aligned}$$

The computation takes about 0.8 seconds.

Example 4.6 ([5])**.** Consider the polynomial equations:

$$h_1 = x_1^2 - 2x_1x_3 + 5,$$

$$h_2 = x_1x_2^2 + x_2x_3 + 1,$$
$$h_3 = 3x_2^2 - 8x_1x_3,$$

which have eight complex solutions, among which two are real.

We choose the objective function as $f = x_1^2 + x_2^2 + x_3^2$. By Algorithm 3.3, we obtain the real solutions:

$$v_1 = (-1.1010, -2.8780, -2.8212)^T,$$
$$v_2 = (0.9657, -2.8125, 3.0716)^T.$$

The computation takes about 1.7 seconds.

Example 4.7 ([3])**.** Consider the intersection of a circle and a bivariate cubic, namely

$$h_1 = x_1^2 + x_2^2 - 2,$$
$$h_2 = 2x_1x_2^2 - x_1 + 1,$$

which have six solutions, all of which are real.

We choose the objective function as $f = x_1^2 + x_2^2$. By Algorithm 3.3, we obtain all real solutions:

$$v_1 = (-1.0000, -1.0000)^T,$$
$$v_2 = (-1.0000, 1.0000)^T,$$
$$v_3 = (-0.3660, -1.3660)^T,$$
$$v_4 = (-0.3660, 1.3660)^T,$$
$$v_5 = (1.3660, -0.3660)^T,$$
$$v_6 = (1.3660, 0.3660)^T.$$

The computation takes about 1.0 seconds.

Example 4.8 ([3])**.** Consider the polynomial equations:

$$h_1 = x_1^2 + x_2^2 + x_3^2 - 1,$$
$$h_2 = x_1^2 + x_2^2 + x_3 - 1,$$
$$h_3 = x_1.$$

We choose the objective function as $f = x_1^2 + x_2^2 + x_3^2$. By Algorithm 3.3, we obtain three real solutions:

$$v_1 = (0, -1.0000, 0.0000)^T,$$
$$v_2 = (0, 0.0000, 1.0000)^T,$$
$$v_3 = (0, 1.0000, 0.0000)^T.$$

The computation takes about 0.6 seconds.

Example 4.9 ([3])**.** This example aims to compute the real critical points of the energy landscape of the two-dimensional nearest-neighbor $\varphi 4$ model on a 3×3 grid as in [9, 19]. We label the nodes $1, \ldots, 9$ with the following figure showing the coupling between the nodes.

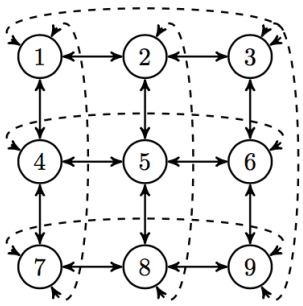

Figure 1: Nearest-neighbor coupling for a 3×3 grid of nodes

Let $N(i)$ denote the four nearest neighbors of node i, that is,

$$\begin{aligned} &N(1) = \{2,3,4,7\}, \quad N(2) = \{1,3,5,8\}, \quad N(3) = \{1,2,6,9\},\\ &N(4) = \{1,5,6,7\}, \quad N(5) = \{2,4,6,8\}, \quad N(6) = \{3,4,5,9\},\\ &N(7) = \{1,4,8,9\}, \quad N(8) = \{2,5,7,9\}, \quad N(9) = \{3,6,7,8\}. \end{aligned}$$

After selecting various parameters for this model, we consider the potential energy

$$V(x) = \sum_{i=1}^{9} \left[\frac{1}{40} x_i^4 - x_i^2 + \frac{1}{4} \sum_{j \in N(i)} (x_i - x_j)^2 \right].$$

The system defining the critical points is $h = \nabla V$ so that

$$h_i = \frac{1}{10} x_i^3 - 2x_i + \sum_{j \in N(i)} (x_i - x_j), i = 1, \ldots, 9.$$

It has 3 solutions

$$(0,0,0,0,0,0,0,0,0), \qquad \mp(\omega,\omega,\omega,\omega,\omega,\omega,\omega,\omega,\omega),$$

where $\omega = \sqrt{20} \approx 4.4721$.

We choose the objective function as $f = x_1^2 + \cdots + x_9^2$. By Algorithm 3.3, we obtain all real solutions

$$\begin{aligned} v_1 =& (0.00000.0000, 0.0000, 0.0000, 0.0000, 0.0000, -0.0000, -0.0000, -0.0000)^T,\\ v_2 =& (-4.4721, -4.4721, -4.4721, -4.4721, -4.4721, -4.4721, -4.4721, -4.4721, -4.4721)^T,\\ v_3 =& (4.4721, 4.4721, 4.4721, 4.4721, 4.4721, 4.4721, 4.4721, 4.4721, 4.4721)^T \end{aligned}$$

in 452 seconds.

5 Conclusions and Discussions

Solving polynomial equations is a classic problem in mathematics. In this paper, we studied how to find all real solutions of polynomial equations, if there are finitely many ones. We proposed a semidefinite relaxation algorithm for computing them sequentially. The convergence properties of the algorithm are also given.

Indeed, the polynomial inequality system $g_i(x) \geq 0$ $(i = 1, \ldots, s)$ can also be solved by a similar method proposed in this paper. Consider the optimization problem

$$\begin{array}{ll} \min & f(x) \\ s.t. & g_i(x) \geq 0, \quad i = 1, \ldots, s, \end{array} \tag{5.1}$$

where f is a random polynomial in $\mathbb{R}[x]$. Suppose $g_i(x) \geq 0$ $(i = 1, \ldots, s)$ has finitely many solutions. Then we can compute them sequentially, similar to the way in Section 3. Each solution can be obtained by solving a hierarchy of semidefinite relaxations.

References

[1] D. Bini and B. Mourrain, Polynomial test suite, Available from http://www-sop.inria.fr/saga/POL, 1996.

[2] J. Bochnak, M. Coste and M. F. Roy, Real algebraic geometry, Springer, 1998.

[3] D.A. Brake, J.D. Hauenstein and A.C. Liddell, Numerically validating the completeness of the real solution set of a system of polynomial equations, arXiv preprint arXiv:1602.00700, (2016).

[4] N. Courtois, A. Klimov, J. Patarin and A. Shamir, Efficient algorithms for solving overdefined systems of multivariate polynomial equations, International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2000, pp. 392–407.

[5] D.A. Cox, J. Little and D. O'Shea, Using Algebraic Geometry, Springer Science and Business Media, 2006.

[6] C.F. Cui, Y.H. Dai and J. Nie, All real eigenvalues of symmetric tensors, *SIAM Journal on Matrix Analysis and Applications* 35 (2014) 1582–1601.

[7] R.E. Curto and L.A. Fialkow, Truncated K-moment problems in several variables, *Journal of Operator Theory* (2005), 189–226.

[8] A. Dickenstein and I. Z. Emiris (editors), Solving polynomial equations: Foundations, algorithms, and applications, Algorithms and Computation in Mathematics, vol.14, Springer, 2005.

[9] R. Franzosi, L. Casetti, L. Spinelli and M. Pettini, Topological aspects of geometrical signatures of phase transitions, *Physical Review E* **60** (1999).

[10] J.W. Helton and J. Nie, A semidefinite approach for truncated K-moment problems, *Foundations of Computational Mathematics* 12 (2012) 851–881.

[11] D. Henrion, J.B. Lasserre and J. Löfberg, GloptiPoly 3: moments, optimization and semidefinite programming, *Optimization Methods and Software* 24 (2009) 761–779.

[12] J.B. Lasserre, Global optimization with polynomials and the problem of moments, *SIAM Journal on Optimization* 11 (2001) 796–817.

[13] J.B. Lasserre, Moments, Positive polynomials and their applications, World Scientific, 2009.

[14] J.B. Lasserre, M. Laurent and P. Rostalski, Computing the real variety of an ideal: A real algebraic and symbolic-numeric algorithm, in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008, 1845–1846.

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

[16] M. Laurent, Sums of squares, moment matrices and optimization over polynomials, *Emerging Applications of Algebraic Geometry* (2009), 157–270.

[17] S.L. Lauritzen, *Graphical Models*, Clarendon Press, 1996.

[18] D. Manocha, Solving systems of polynomial equations, *IEEE Computer Graphics and Applications* 14 (1994) 46–55.

[19] D. Mehta, J.D. Hauenstein and M. Kastner, Energy-landscape analysis of the two-dimensional nearest-neighbor $\varphi 4$ model, *Physical Review E* 85 (2012).

[20] J. Nie, Polynomial optimization with real varieties, *SIAM Journal On Optimization* 23 (2013) 1634–1646.

[21] J. Nie, Certifying convergence of Lasserre's hierarchy via flat truncation, *Mathematical Programming* 142 (2013) 485–510.

[22] J. Nie, The Hierarchy of Local Minimums in Polynomial Optimization, *Mathematical Programming* 151 (2015) 555–583.

[23] G. Pistone, E. Riccomagno and H.P. Wynn, *Algebraic Statistics: Computational Commutative Algebra in Statistics*, CRC Press, 2000.

[24] M. Putinar, Positive polynomials on compact semi-algebraic sets, *Indiana University Mathematics Journal* 42 (1993) 969–984.

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

[26] J.F. Sturm, Using SeDuMi 1.02: A MATLAB toolbox for optimization over symmetric cones, *Optimization Methods and Software* 11 (1999) 625–653.

[27] J. Verschelde and K. Gatermann, Symmetric Newton polytopes for solving sparse polynomial systems, *Advances in Applied Mathematics* 16 (1995) 95–127.

[28] A.H. Wright, Finding all solutions to a system of polynomial equations, *Mathematics of Computation* 44 (1985) 125–133.

[29] L. Zhi and G. Reid, Solving nonlinear polynomial system via symbolic-numeric elimination method, in *Proceedings of the International Conference on Polynomial System Solving*, 2004, pp. 50–53.

[30] http://www.mat.univie.ac.at/ neum/glopt/coconut/Benchmark/Library3/katsura5.mod.

Manuscript received 19 March 2017
revised 26 July 2017, 24 August 2017
accepted for publication 25 August 2017

Xin Zhao
Department of Applied Mathematics, Donghua University 2999 North Renmin Road 201620, P.R. China
E-mail address: xinzhao@dhu.edu.cn

Jinyan Fan
School of Mathematical Sciences, and MOE-LSC
Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai 200240, P.R. China E-mail address: jyfan@sjtu.edu.cn