ON DIFFERENCE-OF-SOS AND DIFFERENCE-OF-CONVEX-SOS DECOMPOSITIONS FOR POLYNOMIALS

YI-SHUAI NIU†

Abstract. In this paper, we are interested in difference-of-convex (DC) decompositions of polynomials. We investigate polynomial decomposition techniques for reformulating any multivariate polynomial into difference-of-sums-of-squares (DSOS) and difference-of-convex-sums-of-squares (DCSOS) polynomials. Firstly, we prove that the set of DSOS and DCSOS polynomials are vector spaces and equivalent to the set of real valued polynomials $\mathbb{R}[x]$. We also show that the problem of finding DSOS and DCSOS decompositions are equivalent to semidefinite programs (SDPs). Then, we focus on establishing several practical algorithms for DSOS and DCSOS decompositions with and without solving SDPs. Some examples illustrate how to use our methods.

Key words. Polynomial optimization, DC programming, SOS, DSOS, DCSOS

AMS subject classifications. 90C26, 90C30

1. Introduction. Let $\mathbb{N}^*$ be the set of natural numbers without zero. Let us denote $\mathbb{R}[x]$ for the vector space of real valued polynomials with variable $x \in \mathbb{R}^n$ for $n \in \mathbb{N}^*$ and with coefficients in $\mathbb{R}$, and $\mathbb{R}[x]_d$ for the subspace of $\mathbb{R}[x]$ with polynomials of degree up to $d \in \mathbb{N}$ whose dimension is $\binom{n+d}{n}$. We are interested in reformulation techniques for any polynomial $p \in \mathbb{R}[x]$ (or $p \in \mathbb{R}[x]_d$) in forms of difference-of-sums-of-squares (DSOS) and difference-of-convex-sums-of-squares (DCSOS). The later formulation can be used to transform a polynomial optimization into a difference-of-convex (DC) program which motivates our research in this paper.

Considering a convex constrained polynomial optimization defined by

$$(P) \quad \min \{f(x) : x \in \mathcal{D} \}$$

where $f$ is a polynomial in $\mathbb{R}[x]$ (or $\mathbb{R}[x]_d$) and $\mathcal{D} \subset \mathbb{R}^n$ is a nonempty closed convex set. The problem $(P)$ is equivalent to a DC program defined by

$$(P_{DC}) \quad \min \{f(x) = g(x) − h(x) : x \in \mathcal{D} \}$$

where $g$ and $h$ are both convex polynomials defined on $\mathcal{D}$. We call that $f$ is a DC function if it can be rewritten as $g − h$ with $g$ and $h$ being convex functions, the form $g − h$ is referred to as DC decomposition of $f$. The existence of a DC decomposition for any polynomial $f$ is a well-known result based on the fact that any $C^2(\mathcal{D}, \mathbb{R})$ function is a DC function [10, 14, 42]. Therefore, the polynomial optimization is indeed a DC program, and the later problem can be solved using an efficient DC Algorithm called DCA, which was introduced by D.T. Pham in 1985 and extensively developed by H.A. Le Thi and D.T. Pham since 1994. The reader can refer to www.lita.univ-lorraine.fr/~lethi and [21, 22, 34, 35] for more details on DCA. In this paper, we will only focus on DSOS and DCSOS decomposition techniques, the solution of polynomial optimization via DC programming approaches will be addressed in our future work.

Finding a difference of convex polynomial decompositions for a general multivariate polynomial is a hard problem although it is known that any polynomial is DC. Up
to now, there are few results in literature on DC decomposition techniques for general polynomials. H. Tuy has shown in [42] that determining whether a monomial is convex or not on $\mathbb{R}^n_+$ is easy. However, the problem becomes much more difficult when the variable $x$ in $\mathbb{R}^n$. Although one can always represent a variable $x \in \mathbb{R}^n$ as $y - z$ with two additional variables $y$ and $z$ in $\mathbb{R}^n_+$ in order to get a new polynomial with variables $(y, z)$ in $\mathbb{R}^{2n}_+$, but this representation increases twice number of variables which does not seem to be a good choice for large scale polynomial optimization. Y.S. Niu et al. investigated several DC decomposition techniques for general polynomial functions on $\mathbb{R}^n$. In [28], they proposed using the form $\rho^2 \|x\|^2 - (\rho^2 \|x\|^2 - f)$ for some suitable $\rho$ to establish DC decompositions. This decomposition requires computing an upper bound for the spectral radius of the Hessian matrix of $f$ over the convex set $D$, which can be easily computed for some special structured $D$ such as a standard simplex or a box. They applied this decomposition to higher moment portfolio optimization [36] and eigenvalue complementarity problems [29, 30, 31]. This approach provides an unified DC decomposition method for any real valued $C^2$ function with bounded spectral radius of its Hessian matrix. The difficulty lies in the estimation of an upper bound and particularly a tight upper bound is required for generating a better DC decomposition. They have also proposed in [28] a DC decomposition techniques based on an equivalent quadratic programming (QP) formulation for polynomial optimization, then applying DCA for QP requires solving a sequence of convex QPs which can be done efficiently using convex QP solvers CPLEX [15] and GUROBI [9]. Recently, A.A. Ahmadi et al. [1] investigated DC decompositions for polynomials with algebraic techniques based on the definition of SOS-convexity. They firstly proved that the convexity is not equivalent to the SOS-convexity [3] and found a counterexample in [4] that a convex polynomial is not a SOS-convex. Then they established three DC decompositions: diagonally-dominant-Sums-of-squares-convex, scaled-diagonally-dominant-sums-of-squares-convex and sos-convex decompositions for polynomials. Meanwhile, Y.S. Niu et al. proposed in [29] a DC decomposition in form of difference-of-convex-sos polynomials for quadratic eigenvalue complementarity problem. This new DC decomposition yields better numerical results in the quality of the computed solution and the computing time than the previous techniques in [30, 31]. This paper will generalize this kind of decomposition techniques.

The contributions in this paper consists of: (i) The establishment of new polynomial decomposition techniques convex-sums-of-squares (CSOS), DSOS and DCSOS based on the definition of SOS decomposition. We firstly give the definition of SOS, CSOS, DSOS and DCSOS polynomials, and investigate some properties on these sets. We prove that the set of CSOS polynomial is a full-dimensional convex cone, and the set of DSOS and DCSOS polynomials are vector spaces and equivalent to $\mathbb{R}[x]$. As a consequence, any polynomial can be rewritten in forms of DSOS and DCSOS whose decomposition can be computed in polynomial time by solving an SDP (semidefinite program). (ii) The development of several practical algorithms to generate these decompositions for any polynomial without solving SDP. We focus on establishing two kinds of decomposition algorithms (Parity decompositions and Spectral decompositions) for DSOS decompositions, as well as some practical DCSOS decomposition algorithms. We illustrate some examples to show the use our methods.

The paper is organized as follows. In section 2, we give the definitions of SOS, PSD, SOS-Convex, CSOS, DSOS and DCSOS polynomials, then investigate their relationships and algebraic properties. In sections 3 and 4, we focus respectively on the constructions of practical algorithms for DSOS and DCSOS decompositions,
and the comparisons on the degree of decompositions, the number of squares and the computational complexity of these algorithms are discussed. Finally, some conclusions and future works are presented in the last section.

2. Polynomial decompositions based on SOS. In this section, we firstly review some background knowledges on SOS polynomials and nonnegative polynomials, then we extend to the definitions of CSOS, DSOS, and DCSOS polynomials.

2.1. SOS polynomials and nonnegativity.

**Definition 2.1** (SOS polynomial). A polynomial \( p \) is called Sums-Of-Squares (SOS) if there exist polynomials \( q_1, \ldots, q_m \) such that \( p = \sum_{i=1}^{m} q_i^2 \). The set of all SOS polynomials of \( \mathbb{R}[x] \) is denoted by \( \text{SOS}_n \); and the subset of \( \text{SOS}_n \) in \( \mathbb{R}[x]_d \) is denoted by \( \text{SOS}_{n,d} \).

**Definition 2.2** (PSD polynomial). A polynomial \( p \) is called nonnegative or positive semidefinite (PSD) if \( p(x) \geq 0 \) for all \( x \in \mathbb{R}^n \). We denote the set of all PSD polynomials of \( \mathbb{R}[x] \) as \( \text{PSD}_n \) and the subset of \( \text{PSD}_n \) in \( \mathbb{R}[x]_d \) as \( \text{PSD}_{n,d} \).

Since a positive polynomial must be of even degree, thus we will use the notations \( \text{SOS}_{n,2d} \) and \( \text{PSD}_{n,2d} \) instead of \( \text{SOS}_{n,d} \) and \( \text{PSD}_{n,d} \).

**Theorem 2.3** (See, e.g., [38, 39]). \( \text{SOS}_n \) and \( \text{PSD}_n \) (resp. \( \text{SOS}_{n,2d} \) and \( \text{PSD}_{n,2d} \)) are proper cones \(^1\) in \( \mathbb{R}[x] \) (resp. \( \mathbb{R}[x]_{2d} \)).

**Proposition 2.4.** The set \( \text{SOS}_n \) (resp. \( \text{PSD}_n \)) verifies the properties

1. \( \text{SOS}_n \) (resp. \( \text{PSD}_n \)) is closed for positive combinations.
2. For any \( p \) and \( q \) polynomials in \( \text{SOS}_n \) (resp. \( \text{PSD}_n \)), we have \( |p|, p^+, p^- \) and \( p \times q \) are all polynomials in \( \text{SOS}_n \) (resp. \( \text{PSD}_n \)).
3. \( \text{SOS}_n \) (resp. \( \text{PSD}_n \)) is not closed for subtraction, division, sup and inf.

**Proof.** The closedness of positive combinations of SOS (resp. PSD) polynomials is due to the fact that \( \text{SOS}_n \) (resp. \( \text{PSD}_n \)) is a convex cone. For any SOS (resp. PSD) polynomial \( p \), we have \(|p| = p, p^+ = p \) and \( p^- = 0 \) being SOS (resp. PSD) polynomials. For \( p \times q \) with \( p \) and \( q \) in \( \text{PSD}_n \), it is clear that \( p \times q \in \text{PSD}_n \); While for \( p \) and \( q \) in \( \text{SOS}_n \), there exist \( m \) polynomials \( p_1, \ldots, p_m \) and \( m' \) polynomials \( q_1, \ldots, q_{m'} \) such that \( p = \sum_{i=1}^{m} p_i^2 \) and \( q = \sum_{j=1}^{m'} q_j^2 \), thus

\[
 p \times q = \sum_{i=1}^{m} p_i^2 \times \sum_{j=1}^{m'} q_j^2 = \sum_{i=1}^{m} \sum_{j=1}^{m'} (p_i q_j)^2 \in \text{SOS}_n.
\]

However, the subtraction of SOS (resp. PSD) polynomials is not always an SOS (resp. PSD) polynomial. E.g., \( (x_1^2) - (x_2^2) \) is neither SOS nor PSD. The division, sup and inf of two SOS (resp. PSD) polynomials may be even not a polynomial. E.g., \( \frac{x^2}{x}, \sup\{x^2, (x+1)^2\} \) and \( \inf\{x^2, (x+1)^2\} \) are not polynomials.

**Remark 2.5.** The properties given in Proposition 2.4 hold for \( \text{SOS}_{n,2d} \) and \( \text{PSD}_{n,2d} \) as well except for multiplication, since the multiplication of two \( \text{SOS}_{n,2d} \) (resp. \( \text{PSD}_{n,2d} \)) polynomials will belong to \( \text{SOS}_{n,4d} \) (resp. \( \text{PSD}_{n,4d} \)).

It is clear that any SOS polynomial is PSD, but the converse is not true. The question on the relationship between SOS and PSD polynomials seems appearing in 1885 when 23-year-old David Hilbert was one of the examiners in the Ph.D. defense of

\(^1\) A proper cone is a full-dimensional (solid), closed, pointed (without line) convex cone.
21-year-old Hermann Minkowski. During the examination, Minkowski claimed that there exist nonnegative polynomials that are not sums of squares, although he did not provide an example or a proof. Later, the problem is stated in Hilbert’s 17th problem as whether a nonnegative polynomial is a sum of squares of rational functions. The answer is also given by Hilbert in 1888 [13]. He showed with a non-constructive technique (based on Cayley-Bacharach theory) that a nonnegative polynomial in \( n \) variables and of degree \( d \) is an SOS polynomial if and only if \( n = 1 \) or \( d = 2 \), or \( (n, d) = (2, 4) \). In other cases, there exist counterexamples such that a nonnegative polynomial is not an SOS polynomial. E.g., the first counterexample is given by Motzkin seventy years later as \( x_1^4x_2^2 + x_1^2x_2^4 - 3x_1^2x_2^2x_3^2 + x_4^6 \) which is PSD but not SOS. See [38] for more counterexamples given by Robinson, Choi, Lax-Lax and Schmüdgen.

On the other hand, deciding nonnegativity of polynomials is known to be NP-hard [27]. However, checking whether a given polynomial admits an SOS decomposition is easy since the problem is equivalent to the feasibility of a semidefinite program (SDP)\(^2\), which is a convex optimization and can be solved efficiently in polynomial time using interior point methods [33]. One can use software packages YALMIP [23] and SOSTOOLS [37] to check whether a given polynomial is SOS or not, and to find an SOS decomposition if the polynomial is SOS.

2.2. CSOS polynomials, SOS-convex polynomials and nonnegative convex polynomials. Now, we extend SOS polynomial with convexity to get the definition of CSOS polynomial.

**Definition 2.6** (CSOS polynomial). A polynomial \( p \) is called Convex-Sums-Of-Squares (CSOS) if it is a convex SOS polynomial. The set of CSOS polynomials of \( \mathbb{R}[x] \) is denoted by \( \text{CSOS}_n \); and the subset of \( \text{CSOS}_n \) in \( \mathbb{R}[x]^d \) is denoted by \( \text{CSOS}_{n,d} \).

Note that CSOS is totally different to the definition of SOS-convexity formally given by Helton and Nie in [11] as

**Definition 2.7** (SOS-matrix and SOS-convex polynomial [11]). A symmetric polynomial matrix \( P(x) \in \mathbb{R}[x]^{m \times m} \) with \( x \in \mathbb{R}^n \) is called an SOS-matrix if there exists a polynomial matrix \( M(x) \in \mathbb{R}[x]^{s \times m} \) for some \( s \in \mathbb{N} \) such that

\[
P(x) = M^T(x) \cdot M(x).
\]

A polynomial \( p \) is called SOS-convex if its Hessian matrix is an SOS-matrix.

The original motivation of Helton and Nie for defining SOS-convexity was to characterize semidefinite representability of convex sets [11]. The SOS-convexity is a sufficient condition for convexity of polynomials based on SOS decomposition of the Hessian matrix. Thus, SOS-convexity can be considered as a second order SOS relaxation for convexity. In general, deciding convexity of polynomials appeared in the list of seven open questions in complexity theory for numerical optimization in 1992 [32]. Recently, it has been proved in [2] that this problem is strongly NP-hard even for polynomials of degree four. However, deciding SOS-convexity can be done in polynomial time based on the following lemma

**Lemma 2.8.** (See [16]) A polynomial matrix \( P(x) \in \mathbb{R}[x]^{m \times m} \) is an SOS-matrix if and only if the scalar polynomial \( y^T \cdot P(x) \cdot y \) is SOS in \( \mathbb{R}[[x, y]] \).

---

\(^2\)An SDP is an optimization problem defined by \( \min \{ f(X) : A(X) = 0, X \succeq 0 \} \) where \( X \succeq 0 \) means that \( X \) is a real symmetric positive semidefinite matrix, \( f \) is a linear form of \( X \), and \( A \) is an affine map of \( X \).

\(^3\)\( M^T \) stands for matrix transportation and the operator \( \cdot \) denotes for the matrix multiplication.
We can conclude that deciding SOS-convexity of a polynomial \( p \) amounts to checking its Hessian matrix \( \nabla^2 p(x) \) to be SOS-matrix, which is equivalent to verifying the scalar polynomial \( y^T \cdot \nabla^2 p(x) \cdot y \) to be \( \text{SOS}([x,y]) \) based on Lemma 2.8, and the last problem can be cast as the feasibility of an SDP, which can be solving efficiently in polynomial time using interior point methods.

Note that the sets of CSOS polynomials, SOS-convex polynomials and nonnegative convex polynomials are different. We summarize some relationships among them as follows. Firstly, nonnegative convex polynomials are not SOS-convex, which is proved by Ahmadi and Parrilo in [4]. They construct a counterexample of trivariate polynomial of degree 8 as

\[
p(x) = 32x_1^8 + 118x_1^6x_2^2 + 40x_1^6x_3^2 + 25x_1^4x_2^4 - 43x_1^4x_2^2x_3^2 \\
-35x_1^4x_2^4 + 3x_1^2x_2^2x_3^2 - 16x_1^2x_2^4x_3^2 + 24x_1^2x_3^6 \\
+16x_3^8 + 44x_2^4x_3^4 + 70x_2^4x_3^2 + 60x_2^2x_3^6 + 30x_3^8
\]

which is nonnegative and convex but not SOS-convex. The gap between SOS-convexity and convexity of polynomials is completely characterized in [3] as a convex polynomial in \( n \) variables and of degree \( d \) is SOS-convex if and only if \( n = 1 \) or \( d = 2 \), or \((n,d) = (2,4)\). It is interesting to observe that PSD is equivalent to SOS exactly in dimensions and degrees where convexity is equivalent to SOS-convexity, although it is still unclear whether there can be a deeper and more unifying reason explaining these observations.

Moreover, the polynomial (2.1) serves as well a counterexample to make difference between CSOS and SOS-convex since it has been shown in [4] that this polynomial is both SOS and convex (i.e., CSOS) but not SOS-convex. Conversely, a SOS-convex polynomial is neither CSOS nor nonnegative convex, since SOS-convexity does not apply nonnegativity. E.g., the univariate quadratic polynomial \( x^2 - x + 1 \) is SOS-convex (since its Hessian matrix is an SOS-matrix) but obviously not nonnegative.

However, Helton and Nie showed in [11] that a nonnegative SOS-convex polynomial is indeed a CSOS polynomial since if a nonnegative polynomial is SOS-convex, then it must be SOS.

Furthermore, any CSOS polynomial is nonnegative convex polynomial, but the converse is not true. The polynomial (2.1) does not serve as a counterexample since it is both nonnegative convex and CSOS. However, Blekherman proved in [7] that a nonnegative convex polynomial but not SOS does exist, although no explicit examples are known yet.

Concerning on the algebraic property of CSOS, SOS-convex and nonnegative convex polynomials, it has been shown in [1] that SOS-convex and nonnegative convex polynomials are proper cones. It is also true for CSOS polynomials.

**Theorem 2.9.** \( \text{CSOS}_n \) (resp. \( \text{CSOS}_{n,2d} \)) is a proper cone in \( \mathbb{R}[x] \) (resp. \( \mathbb{R}[x]_{2d} \)).

**Proof.** The set \( \text{CSOS}_n \) (resp. \( \text{CSOS}_{n,2d} \)) is pointed since it is a subset of the pointed set \( \text{SOS}_n \) (resp. \( \text{SOS}_{n,2d} \)). The closedness and convexity are immediate consequences of the recession cone theorem [6] since \( \text{CSOS}_n \) (resp. \( \text{CSOS}_{n,2d} \)) is a recession cone of the proper cone \( \text{SOS}_n \) (resp. \( \text{SOS}_{n,2d} \)), and any recession cone of nonempty closed convex set is closed and convex. To show that \( \text{CSOS}_n \) (resp. \( \text{CSOS}_{n,2d} \)) is full-dimensional in \( \mathbb{R}[x] \) (resp. \( \mathbb{R}[x]_{2d} \)), since the set of SOS-convex polynomials is a proper cone, the nonnegative SOS-convex polynomials is also a proper cone which is a subset of CSOS polynomials [11]. Finally, we conclude that \( \text{CSOS}_n \) (resp. \( \text{CSOS}_{n,2d} \)) is a proper cone in \( \text{CSOS}_n \) (resp. \( \text{CSOS}_{n,2d} \)). \( \blacksquare \)

**Proposition 2.10.** \( \text{CSOS}_n \) (resp. \( \text{CSOS}_{n,2d} \)) verifies the properties
1. $\text{CSOS}_n$ (resp. $\text{CSOS}_{n,2d}$) is closed for positive combinations.
2. For any $p$ and $q$ polynomials in $\text{CSOS}_n$ (resp. $\text{CSOS}_{n,2d}$), we have $|p|, p^+$ are $p^-$ are all polynomials in $\text{CSOS}_n$ (resp. $\text{CSOS}_{n,2d}$).
3. $\text{CSOS}_n$ (resp. $\text{CSOS}_{n,2d}$) is not closed for multiplication, subtraction, division, sup and inf.

Proof. The proofs are similar as in Proposition 2.4. The multiplication of two convex polynomials is not convex (e.g., $x_1^2x_2^2$ is not convex on $\mathbb{R}^2$ even $x_1^2$ and $x_2^2$ are both CSOS polynomials).

It is clear that the Proposition 2.10 holds for nonnegative convex polynomials, but not for SOS-convex polynomials due to the lack of nonnegativity, since we have for a SOS-convex polynomial $p$, the expressions $|p|, p^+$ and $p^-$ may not be polynomials.

Concerning on the certification of CSOS polynomials, since determining the convexity of a polynomial is NP-hard, it seems that the certification of CSOS polynomial is also NP-hard in general. This amounts to find a convex but not SOS-convex SOS polynomial which can not be checked in polynomial time. In our knowledge, no explicit examples are known yet. However, both SOS and SOS-convexity can be easily checked in polynomial time, we can then certificate any SOS-convex SOS polynomial (as a special CSOS polynomial) in polynomial time by solving an SDP.

2.3. DSOS polynomials and DCSOS polynomials. An interesting question is related to the representability of any polynomial as difference of SOS or CSOS polynomials. In this subsection we will show that this statement is true, and such decompositions can be constructed in polynomial time by solving SDPs.

Definition 2.11 (DSOS polynomial and DCSOS polynomial). A polynomial $p$ is called difference-of-sums-of-squares (DSOS) (resp. difference-of-convex-sums-of-squares (DCSOS)) if there exist SOS (resp. CSOS) polynomials $s_1$ and $s_2$ such that $p = s_1 - s_2$. The components $s_1$ and $s_2$ are called DSOS (resp. DCSOS) components of $p$. The set of DSOS (DCSOS) polynomials of $\mathbb{R}[x]$ is denoted by $\text{DSOS}_n$ (resp. $\text{DCSOS}_n$), and the subset of $\text{DSOS}_n$ (resp. $\text{DCSOS}_n$) in $\mathbb{R}[x]_d$ is denoted by $\text{DSOS}_{n,d}$ (resp. $\text{DCSOS}_{n,d}$).

Note that the degrees of DSOS and DCSOS polynomials could be either even or odd which is different to SOS and CSOS with only even degree components.

Theorem 2.12. $\text{DSOS}_n$ and $\text{DCSOS}_n$ (resp. $\text{DSOS}_{n,d}$ and $\text{DCSOS}_{n,d}$) are vector subspaces of $\mathbb{R}[x]$ (resp. $\mathbb{R}[x]_d$).

Proof. We will just prove the theorem for $\text{DSOS}_n$ (same to the other cases).

- $\text{DSOS}_n$ is nonempty since $0 \in \text{DSOS}_n$.
- For any $p$ and $q$ of $\text{DSOS}_n$ and $\lambda \in \mathbb{R}$, there exist $p_1, p_2$ and $q_1, q_2$ of $\text{SOS}_n$ such that $p = p_1 - p_2$ and $q = q_1 - q_2$. Then,

$$\lambda \times p + q = \begin{cases} 
(\lambda \times p_1 + q_1) - (\lambda \times p_2 + q_2) \in \text{DSOS}_n & \text{if } \lambda \geq 0 \\
(\lambda \times p_2 + q_1) - (\lambda \times p_1 + q_2) \in \text{DSOS}_n & \text{if } \lambda < 0
\end{cases}$$

which means $\lambda \times p + q \in \text{DSOS}_n$.

We conclude that $\text{DSOS}_n$ is a vector subspace of $\mathbb{R}[x]$.

Note that $\text{DSOS}_n$ (resp. $\text{DCSOS}_n$) is in fact a vector space spanned by $\text{SOS}_n$ (resp. $\text{CSOS}_n$) since $\text{DSOS}_n = \text{SOS}_n - \text{SOS}_n$ (resp. $\text{DCSOS}_n = \text{CSOS}_n - \text{CSOS}_n$). Moreover, $\text{DCSOS}_n \subset \text{DSOS}_n$ since $\text{CSOS}_n \subset \text{SOS}_n$. We have a stronger result as $\text{DCSOS}_n = \text{DSOS}_n$, whose proof will be given in Theorem 2.14.

The following proposition illustrates usual operations on $\text{DSOS}_n$ and $\text{DCSOS}_n$. 


**Proposition 2.13.** DSOS \(_n\) (resp. DCSOS \(_n\)) verifies the properties
1. DSOS \(_n\) (resp. DCSOS \(_n\)) is closed for linear combinations.
2. \(\forall (p,q) \in \text{DSOS}^n (\text{resp. DCSOS}^n)\), we have \(p \times q \in \text{DSOS} \_n \) (resp. \(\text{DCSOS} \_n\)), but \(|p|, p^+, p^-\), \(\text{sup}\{p,q\}\) and \(\text{inf}\{p,q\}\) are not in \(\text{DSOS} \_n \) (resp. \(\text{DCSOS} \_n\)).

**Proof.** The closedness of linear combinations is obviously true for any vector space. For proving the closedness of multiplication, let \(p\) and \(q\) possess the following DSOS (resp. DCSOS) decompositions

\[
p = \sum_{i=1}^{m} p_i \sum_{j=1}^{k} p_j^2, \quad q = \sum_{i'=1}^{m'} q_i' \sum_{j'=1}^{k'} q_j'^2.
\]

Then a DSOS decomposition of \(p \times q\) is easily computed by

\[
p \times q = \left( \sum_{i=1}^{m} p_i \sum_{i'=1}^{m'} (p_i \times q_{i'})^2 + \sum_{j=1}^{k} \sum_{j'=1}^{k'} (\hat{p}_j \times \hat{q}_j')^2 \right) \in \text{DSOS}_n
\]

\[
- \left( \sum_{i=1}^{m} \sum_{j'=1}^{k'} (p_i \times \hat{q}_j')^2 + \sum_{j=1}^{k} \sum_{i'=1}^{m'} (\hat{p}_j \times q_{i'})^2 \right) \in \text{DSOS}_n.
\]

However, this formulation is not DCSOS. It is not difficult to find a DCSOS decomposition for \(p \times q\) as

\[
p \times q = \frac{1}{2} \left[ \left( \sum_{i=1}^{m} p_i^2 + \sum_{i'=1}^{m'} q_{i'}^2 \right) \right]^2 + \left( \sum_{j=1}^{k} \sum_{j'=1}^{k'} \hat{p}_j^2 + \sum_{i'=1}^{m'} \hat{q}_{j'}^2 \right)^2 \in \text{CSOS}_n
\]

\[
- \frac{1}{2} \left[ \left( \sum_{i=1}^{m} p_i^2 + \sum_{j'=1}^{k'} q_{j'}^2 \right) \right]^2 + \left( \sum_{j=1}^{k} \sum_{i'=1}^{m'} \hat{p}_j^2 + \sum_{i'=1}^{m'} \hat{q}_{i'}^2 \right)^2 \in \text{CSOS}_n.
\]

Concerning on the operations \(|p|, p^+, p^-\), \(\text{sup}\{p,q\}\) and \(\text{inf}\{p,q\}\), they are not closed for DSOS and DCSOS since they are not always polynomials. E.g., let \(x \in \mathbb{R}\), \(p(x) = x^2 - x^4\) and \(q(x) = x^4\), we have \(p\) and \(q\) that are both DSOS and DCSOS polynomials, but \(|p|, p^+, p^-\), \(\text{sup}\{p,q\}\) and \(\text{inf}\{p,q\}\) are not polynomials.

Note that for DCSOS polynomials \(p\) and \(q\), although \(|p|, p^+, p^-\), \(\text{sup}\{p,q\}\) and \(\text{inf}\{p,q\}\) are not DCSOS polynomials, but they are still DC functions since these operations are closed for DC functions.

The next theorem reveals the relationships between the sets of SOS, CSOS, DSOS and DCSOS polynomials, and guarantees the representability of any polynomial as DSOS and DCSOS.

**Theorem 2.14.** For any \((n,d) \in \mathbb{N}^2\), we have
- \(\text{CSOS}_n \subset \text{SOS}_n\) (resp. \(\text{CSOS}_{n,2d} \subset \text{SOS}_{n,2d}\))
- \(\text{SOS}_{n,2d} \subset \text{SOS}_{n,2d+2}\) (resp. \(\text{CSOS}_{n,2d} \subset \text{CSOS}_{n,2d+2}\))
Theorem 2.9, there exists two real symmetric positive
Theorem 2.14, the set $\mathbb{R}(2.4)$
We have the following theorem.

Proof. The first three inclusions are obviously true.

To prove $\mathbb{R}[x]_{2d} = \mathbb{DCSOS}_{n,2d} = \mathbb{DSOS}_{n,2d}$, we just need to prove that $\mathbb{R}[x]_{2d} \subset \mathbb{DSOS}_{n,2d}$ and $\mathbb{R}[x]_{2d} \subset \mathbb{DCSOS}_{n,2d}$. For any polynomial $p \in \mathbb{R}[x]_{2d}$, the following two cases will be considered:

(i) If $p$ is in form of $\mathbb{DSOS}_{n,2d}$ (resp. $\mathbb{DCSOS}_{n,2d}$), then the result is true.
(ii) Otherwise, based on Theorem 2.9, the set $\mathbb{SOS}_{n,2d}$ (resp. $\mathbb{CSOS}_{n,2d}$) is a proper cone, so $\text{int}(\mathbb{SOS}_{n,2d})$ (resp. $\text{int}(\mathbb{CSOS}_{n,2d})$) is nonempty.

We can rewrite $p$ as difference of two vectors in $\text{int}(\mathbb{SOS}_{n,2d})$ (resp. $\text{int}(\mathbb{CSOS}_{n,2d})$), because for any $p_1 \in \text{int}(\mathbb{SOS}_{n,2d})$ (resp. $\text{int}(\mathbb{CSOS}_{n,2d})$) with $p_1 \neq p$, there exists an element $p_2$ in the segment $[p, p_1]$ such that $p_2 \in \mathbb{SOS}_{n,2d}$ (resp. $\mathbb{CSOS}_{n,2d}$). Thus, $\exists \lambda \in [0, 1]$ such that

$$p_2 = \lambda \cdot p + (1 - \lambda) \cdot p_1$$

which is rewritten as

$$p = \frac{1}{\lambda} \cdot p_2 - \frac{(1 - \lambda)}{\lambda} \cdot p_1$$

with $\frac{1}{\lambda} \cdot p_2$ and $\frac{(1 - \lambda)}{\lambda} \cdot p_1$ in $\mathbb{SOS}_{n,2d}$ (resp. $\mathbb{CSOS}_{n,2d}$).

Therefore, $\mathbb{R}[x]_{2d} \subset \mathbb{DSOS}_{n,2d}$ and $\mathbb{R}[x]_{2d} \subset \mathbb{DCSOS}_{n,2d}$ which yield

$$\mathbb{R}[x]_{2d} = \mathbb{DCSOS}_{n,2d} = \mathbb{DSOS}_{n,2d}.$$  

The inclusion $\mathbb{R}[x]_{2d+1} \subset \mathbb{DCSOS}_{n,2d+2}$ is due to the fact that $\mathbb{R}[x]_{2d+1} \subset \mathbb{R}[x]_{2d+2} = \mathbb{DCSOS}_{n,2d+2}$.

Since $\mathbb{R}[x]_{2d} = \mathbb{DCSOS}_{n,2d} = \mathbb{DSOS}_{n,2d}$ and $\mathbb{R}[x]_{2d+1} \subset \mathbb{DCSOS}_{n,2d+2}$ for any $(n, d) \in \mathbb{N}^2$, we get the equivalence $\mathbb{R}[x] = \mathbb{DSOS}_n = \mathbb{DCSOS}_n$. Therefore, the degree of the DSOS (resp. DCSOS) components for any polynomial $p \in \mathbb{R}[x]$ must be an even number and at least equals to $2 \left\lceil \frac{\deg(p)}{2} \right\rceil$.  

Concerning on the complexity for computing DSOS and DCSOS decompositions. We have the following theorem.

**Theorem 2.15.** A DSOS (resp. DCSOS) decomposition for any polynomial $p \in \mathbb{R}[x]_d$ can be computed in polynomial time by solving an SDP.

**Proof.** Firstly, for DSOS decomposition, let $b(x) = (1, x_1, \ldots, x_n, x_1^2, x_2^2, \ldots, x_n^2)^T$ be a vector of all monomials in variable $x \in \mathbb{R}^n$ and of degree up to $\lceil \frac{d}{2} \rceil$ (called full-basis of $\mathbb{R}[x]_d$), then based on Theorem 2.14, there exists two real symmetric positive semidefinite matrices $Q_1$ and $Q_2$ such that any $p \in \mathbb{R}[x]_d$ can be rewritten as

$$p = (b^T \cdot Q_1 \cdot b) - (b^T \cdot Q_2 \cdot b)^5.$$  

$^4$int$(A)$ stands for the interior of the set $A$.

$^5$A polynomial $f \in \mathbb{R}[x]_d$ is SOS if and only if there exists a real symmetric positive semidefinite matrix $Q$ such that $f = b^T \cdot Q \cdot b$. 


Then, all solutions of DSOS decompositions of $p$ are given in the set 

$$S(b) = \{(Q_1, Q_2) : p = b^T \cdot (Q_1 - Q_2) \cdot b, Q_1 \succeq 0, Q_2 \succeq 0\}$$

in which the constraint $p = b^T \cdot (Q_1 - Q_2) \cdot b$ implies a set of linear equations on the elements of $Q_1$ and $Q_2$. Therefore $S(b)$ is in fact an SDP which can be solved in polynomial time using interior point methods.

In a similar way, finding a DCSOS decomposition amounts to search in the set $S(b)$ a decomposition in form of (2.4) such that the components $b^T \cdot Q_1 \cdot b$ and $b^T \cdot Q_2 \cdot b$ are both convex. However, checking convexity for polynomials is NP-hard in general, but checking SOS-convexity can be done in polynomial time. Therefore, we can find a difference of SOS-convex SOS decomposition (a special DCSOS decomposition) for $p$ by solving the following feasibility problem:

$$\hat{S}(b) = \{(Q_1, Q_2) \in S(b) : \nabla^2 (b^T \cdot Q_1 \cdot b) \text{ and } \nabla^2 (b^T \cdot Q_2 \cdot b) \text{ are SOS-matrices}\}.$$ 

Base on Lemma 2.8, checking the polynomial matrix $\nabla^2 (b^T \cdot Q_i \cdot b)$, $i = 1, 2$ is SOS-matrix is equivalent to the feasibility of an SDP, so $\hat{S}(b)$ is indeed an SDP which can be solved in polynomial time using interior point method.

In practice, there exist some efficient SDP solvers such as MOSEK (MOSEK ApS)[26], SeDuMi (Jos F. Sturm) [40], SDPT3 (Kim-Chuan Toh, Michael J. Todd, and Reha H. Tutuncu) [41], CSDP (Helmberg, Rendl, Vanderbei, and Wolkowicz) [8], SDPA (Masakazu Kojima, Mituhiro Fukuda et al.) [17] and DSDP(Steve Benson, Yinyu Ye, and Xiong Zhang) [5]. We suggest using Yalmip [23] for rapid modeling and solving an SDP on MATLAB by calling one of the SDP solvers mentioned above.

Note that an SDP can be solved in polynomial time in theory, but in practice only small-scale and sparse SDP can be solved effectively in a reasonable time. The reader is recommended to the web of Hans D. Mittelmann http://plato.asu.edu/bench.html for some benchmark results on the performance of SDP solvers. In fact, using the full-basis $b(x)$ defined in the proof of Theorem 2.15 will often lead to large-scale and dense SDP which will become intractable very quickly when the degree and the density of the target polynomial increases. Therefore, in the rest of the paper, we will focus on establishing practical algorithms for both DSOS and DCSOS decompositions for any polynomial (especially for higher order dense polynomials) without solving SDP.

3. Practical DSOS decompositions. In this section, we will propose some practical algorithms to find DSOS decomposition without solving SDP.

Let $m$ be a monomial of $\mathbb{R}[x]$ defined by:

$$m(x) = c_{\alpha} x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} = c_{\alpha} x^{\alpha}$$

where $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ and $c_{\alpha} \in \mathbb{R}$ is the coefficient of the monomial $m$. The concise notation $x^{\alpha}$ stands for the monomial $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. The degree of the monomial $m$, denoted by $\deg(m)$, is equal to $\sum_{i=1}^n \alpha_i$ denoted by $|\alpha|$. Any polynomial $p \in \mathbb{R}[x]$ can be presented as canonical form as sum of its monomials as

$$p(x) = \sum_{\alpha} c_{\alpha} x^{\alpha}.$$ 

Obviously, if we can find a way to represent any monomial into DSOS polynomial, then based on Proposition 2.13, any polynomial as a linear combination of monomials (DSOS polynomials) is also a DSOS polynomial.
3.1. Parity DSOS decompositions. The first type of DSOS decomposition algorithms proposed here is based on parity separation of monomial which consists of separating \(x^\alpha\) into two parts: One part with odd degree’s variables (noted by \(o\)) and the other part with even degree’s variables (noted by \(e\)), thus

\[
x^\alpha = o(x)e^2(x).
\]

E.g., the monomial \(x_1^2x_2^3\) can be separated as \(o(x) = x_2, e^2(x) = (x_1x_2)^2\). It can be also rewritten as \(o(x) = x_1^3, e^2(x) = x_1^2\).

Clearly, the parity separations for a given monomial are not unique in general, but the number of all possible separations is finite.

3.1.1. Parity DSOS decomposition algorithm. The basic idea to get a DSOS decomposition for a monomial \(m\) is to find a DSOS decomposition for the part \(o\), then multiply the result by \(e^2\). The detailed method is described in Algorithm 3.1.

**Algorithm 3.1 Parity DSOS Decomposition**

**Input:** Monomial \(m(x)\); constant \(s > 0\).

**Output:** DSOS decomposition \(dsos(x)\).

**Step 1:** Extract \(x^\alpha\) and \(c_\alpha\) of \(m(x)\).

**Step 2:** Parity separate \(x^\alpha\) as \(o(x)e^2(x)\).

**Step 3:** Decompose the monomial \(o(x)\) as DSOS polynomial as:

\[
o(x) = \frac{1}{2s} (o(x) + s)^2 - \frac{1}{2s} (o(x)^2 + s^2).
\]

**Step 4:** A DSOS decomposition of \(m(x)\) is given by

\[
dsos(x) = \frac{c_\alpha}{2s} (e(x)o(x) + se(x))^2 - \frac{c_\alpha}{2s} (o^2(x)e^2(x) + s^2e^2(x)).
\]

**Example 3.1.** Applying Algorithm 3.1 to \(m(x) = -2x_1^3x_2^5\) with a chosen \(s > 0\): 

**Step 1:** \(c_\alpha = -2, x^\alpha = x_1^3x_2^5\).

**Step 2:** One possible parity septation of \(x_1^3x_2^5\) is \(o(x) = x_1^2x_2, e(x) = x_2^2\).

**Step 3:** Decompose \(o(x)\) as DSOS polynomial by formulation (3.1) as

\[
o(x) = x_1^3x_2 = \frac{1}{2s} (x_1^3x_2 + s)^2 - \frac{1}{2s} (x_1^2x_2^2 + s^2).
\]

**Step 4:** A DSOS decomposition of \(m(x)\) is given by

\[
dsos(x) = \frac{c_\alpha}{2s} (e(x)o(x) + se(x))^2 - \frac{c_\alpha}{2s} (o^2(x)e^2(x) + s^2e^2(x)).
\]

\[
= -\frac{1}{s} (x_1^3x_2^3 + sx_2)^2 + \frac{1}{s} (x_1^2x_2^2 + s^2x_2^2).
\]

Note that the formulation (3.1) can be applied directly to compute a DSOS decomposition for any polynomial \(p \in \mathbb{R}[x]\) as:

\[
p = \frac{1}{2s} (p + s)^2 - \frac{1}{2s} (p^2 + s^2).
\]
This confirmed the fact that any polynomial is DSOS. However, we would never use such DSOS decomposition in practice since its degree of DSOS components equals to $2 \deg(p)$, while Algorithm 3.1 will get some smaller degree decompositions. The next proposition proves the degree of DSOS components given by Algorithm 3.1.

**Proposition 3.2.** For any monomial $m \in \mathbb{R}[x]$ with a parity separation $oe^2$, the degree of DSOS components in Algorithm 3.1 is not greater than $\deg(m) + \deg(o)$.

**Proof.** In the expression of the output $dsos(x)$, it is easy to see that $\deg((eo + se)^2) = \deg(e^2o^2 + s^2e^2) = \deg(m) + \deg(o)$.

Therefore, the degree of DSOS components depends on $\deg(o)$. In order to reduce the degree of DSOS components, we are interested in parity separation with minimal degree for $o$ which yield the next improved parity DSOS decomposition.

### 3.1.2. Improved parity DSOS decomposition algorithm

Let us denote $O(m) = \{ i \in [1, n] : \alpha_i \text{ is odd} \}$ the index set for odd degree variables in monomial $m$. Then only two cases will be considered:

**Case 1:** If $O(m) = \emptyset$, then the part $o$ with minimal degree is $o(x) = 1$. E.g., $m(x) = x_1^2x_2^3$ with $o(x) = 1$ and $e(x) = x_1^2x_2$.

**Case 2:** Otherwise ($O(m) \neq \emptyset$), then the part $o$ with minimal degree must be $o(x) = \prod_{i \in O(m)} x_i$. E.g., $m(x) = x_1^3x_2^5x_3^2 = x_1x_2(x_1x_2x_3)^2$.

with $o(x) = x_1x_2$ and $e(x) = x_1x_2^2x_3$.

Once the part $o$ with minimal degree is computed, instead of applying the formulation (3.1) to get a DSOS decomposition for $o$ with degree $2 \deg(o)$, we propose **Procedure D** to find a DSOS decomposition for $o$ with minimal degree $2\lceil \frac{\deg(o)}{2} \rceil$.

**Procedure D: (DSOS Decomposition for $o(x)$)**

**Step 1:** Make pairs of variables in $o(x)$.
- If $|O(m)|$ is even, then we can pair all variables two by two.
  E.g., for $o(x) = x_1x_2x_3x_4$, we have pairs $\{(x_1, x_2), (x_3, x_4)\}$.
- Otherwise, we can pair one variable with 1, and make pairs of others.
  E.g., for $o(x) = x_1x_2x_3$, we have pairs $\{(x_1, x_2), (x_3, 1)\}$.

**Step 2:** Rewrite each pair of type $(x_i, x_j)$ or $(x_i, 1)$ as DSOS by:

$$(x_i, x_j) \rightarrow x_i x_j = \left(\frac{x_i + x_j}{2}\right)^2 - \left(\frac{x_i - x_j}{2}\right)^2 \in DSOS,$$

$$(x_i, 1) \rightarrow x_i = \left(\frac{x_i + 1}{2}\right)^2 - \left(\frac{x_i - 1}{2}\right)^2 \in DSOS.$$

**Step 3:** Apply the formulation (2.2) to get a DSOS decomposition for the multiplications of DSOS decompositions of all pairs. The degree of the resulting DSOS components equals to $2\lceil \frac{\deg(o)}{2} \rceil$.

The detailed improved parity DSOS decomposition algorithm is stated as follows:

---

$^6[a, b]$ stands for the set of integers included in the interval $[a, b]$. 

---

DSOS & DCSOS DECOMPOSITIONS FOR POLYNOMIALS
Algorithm 3.2 Improved Parity DSOS Decomposition

**Input:** Monomial \( m(x) \).

**Output:** DSOS decomposition \( \text{dsos}(x) \).

**Step 1:** Extract \( x^\alpha \) and \( c_\alpha \) for \( m(x) \).

**Step 2:** Compute \( O(m) \) to get a parity separation of \( x^\alpha \) with minimal degree of \( o \) as:

\[
o(x) = \prod_{i \in O(m)} x_i; \quad e^2(x) = m(x)/o(x).
\]

**Step 3:** Use Procedure D to get a DSOS decomposition for \( o(x) \) as:

\[
o(x) = o_1(x) - o_2(x).
\]

**Step 4:** A DSOS decomposition of \( m(x) \) is given by:

\[
\text{dsos}(x) = c_\alpha \left( o_1(x)e^2(x) - o_2(x)e^2(x) \right).
\]

return \( \text{dsos}(x) \).

**Example 3.3.** Applying Algorithm 3.2 to the same example \( m(x) = -2x^3_1x^5_2 \):

**Step 1:** \( c_\alpha = -2, x^\alpha = x^3_1x^5_2 \).

**Step 2:** The index set \( O(m) = \{1, 2\} \) and the parity separation of \( x^\alpha \) with minimal degree of \( o \) is

\[
o(x) = x_1x_2, \quad e^2(x) = x^3_1x^5_2.
\]

**Step 3:** Use Procedure D to get a DSOS decomposition of \( o(x) \) as

\[
o(x) = x_1x_2 = \left( \frac{x_1 + x_2}{2} \right)^2 - \left( \frac{x_1 - x_2}{2} \right)^2.
\]

**Step 4:** A DSOS decomposition of \( m(x) \) is

\[
\text{dsos}(x) = -2 \left[ x_1x^2_2 \left( \frac{x_1 + x_2}{2} \right) \right]^2 + 2 \left[ x_1x^2_2 \left( \frac{x_1 - x_2}{2} \right) \right]^2.
\]

The degree of DSOS components in Example 3.3 equals to 8 which is exactly equal to \( 2\left\lceil \frac{\deg(m)}{2} \right\rceil \). The next proposition guarantees the generation of a minimal degree DSOS decomposition by Algorithm 3.2.

**Proposition 3.4.** For any monomial \( m \in \mathbb{R}[x] \), the degree of the DSOS components generated by Algorithm 3.2 is equal to \( 2\left\lceil \frac{\deg(m)}{2} \right\rceil \).

**Proof.** Two cases will be considered

- If \( \deg(m) \) is even, then \( \deg(o) \) is also even (since \( \deg(m) = 2 \deg(e) + \deg(o) \)).
  
  We get \( \deg(o_1) = \deg(o_2) = \deg(o) \), thus the degree of DSOS components of \( m \) should be \( \deg(o_1e^2) \) and \( \deg(o_2e^2) \) which are both equal to \( \deg(o) + 2 \deg(e) = 2 \left\lceil \frac{\deg(m)}{2} \right\rceil \).

- If \( \deg(m) \) is odd, then \( \deg(o) \) is odd too. We get \( \deg(o_1) = \deg(o_2) = \deg(o) + 1 \), and \( \deg(o_1e^2) = \deg(o_2e^2) = \deg(o) + 2 \deg(e) = \deg(m) + 1 = 2 \left\lceil \frac{\deg(m)}{2} \right\rceil \). \( \square \)
3.2. Spectral DSOS decompositions. The second type of DSOS decomposition algorithms are based on the spectral decomposition of real symmetric matrix.

**Definition 3.5 (Valid basis).** For any polynomial $p \in \mathbb{R}[x]$, a valid basis of $p(x)$ is a set of monomials $\mathcal{B}$ written in column matrix form as $b(x)$ such that there exists a real symmetric matrix $Q$ (called Gram matrix) verifying

$$p(x) = b^T(x) \cdot Q \cdot b(x).$$

E.g., the full-basis $b(x) = (1, x_1, \ldots, x_n, x_1x_2, \ldots, x_n^{\lceil \frac{d}{2} \rceil})^T$ is obviously a valid basis for all polynomials of $\mathbb{R}[x]_d$.

Note that valid basis is not unique. E.g., we have two different valid bases for $p(x) = x_1 + x_2$ as

$$b(x) = \begin{bmatrix} 1 \\ x_1 \\ x_1^2x_2 \end{bmatrix}, Q = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix} \quad \text{and} \quad b(x) = \begin{bmatrix} 1 \\ x_1 \\ x_1x_2 \end{bmatrix}, Q = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 \end{bmatrix}.$$

For any valid basis $b(x)$, the Gram matrix $Q$ can be computed by solving a linear system of elements in $Q$ derived from the following identity:

$$p(x) - b^T(x) \cdot Q \cdot b(x) = 0, \forall x \in \mathbb{R}^n.

Once a Gram matrix $Q$ is computed, since $Q$ is a real symmetric matrix, it is diagonalizable over $\mathbb{R}$ with only real eigenvalues, thus the spectral decomposition for real symmetric matrix $Q$ is used for finding an orthogonal matrix $P$ and a diagonal matrix $\Lambda$ with diagonal entries being eigenvalues of $Q$ such that

$$Q = P \cdot \Lambda \cdot P^T.$$

Let $r$ be the length of $b(x)$, and $\lambda_1, \ldots, \lambda_r$ be all eigenvalues of $Q$. Then

$$p(x) = b^T(x) \cdot P \cdot \Lambda \cdot P^T \cdot b(x).$$

Denote $y(x) = P^T \cdot b(x)$ and $K = \{k \in [1, r] : \lambda_k \neq 0\}$, we get

$$p(x) = \sum_{i \in K} \lambda_i y_i^2(x) \in \mathcal{DSOS}.$$

The spectral DSOS decomposition algorithm is summarized as follows:

**Algorithm 3.3 Spectral DSOS Decomposition**

**Input:** Polynomial $p(x)$.

**Output:** DSOS decomposition $\text{dsos}(x)$.

**Step 1:** Compute a valid basis $b(x)$ and Gram matrix $Q$ for $p(x)$.

**Step 2:** Spectral decomposition on $Q$ to get $P$ and $\Lambda$ verifying

$$Q = P \cdot \Lambda \cdot P^T.$$

**Step 3:** Compute $y(x) = P^T \cdot b(x)$ and $K = \{k \in [1, r] : \lambda_k \neq 0\}$, a DSOS decomposition of $p(x)$ is given by

$$\text{dsos}(x) = \sum_{i \in K} \lambda_i y_i^2(x).$$

**return** $\text{dsos}(x)$. 
The following proposition holds for the degree of DSOS components in Algorithm 3.3:

**Proposition 3.6.** For polynomial \( p \in \mathbb{R}[x] \) with any valid basis \( b(x) \), the degree of DSOS components generated by Algorithm 3.3 is not greater than \( 2 \deg(b(x)) \).

**Proof.** Based on the formulation (3.3), the degree of DSOS components is smaller than \( \max_{i \in K} \deg(y^i) \) which is upper bounded by \( 2 \deg(y(x)) \). Then we get from \( y(x) = P^T \cdot b(x) \) that \( \deg(y(x)) = \deg(b(x)) \) which yields the required result. \( \square \)

Note that the flexibility in Algorithm 3.3 is related to the choice of valid basis \( b(x) \) and the corresponding Gram matrix \( Q \). Different valid bases and Gram matrices will lead to different DSOS decompositions. Next, we will investigate two forms of valid bases: direct basis and minimal basis.

### 3.2.1. Direct basis spectral DSOS decomposition.

**Definition 3.7 (Direct basis).** For any polynomial \( p \in \mathbb{R}[x] \) with canonical form \( p(x) = \sum_{\alpha} c_\alpha x^\alpha \), the direct basis of \( p(x) \) is a valid basis of \( p(x) \) as \( B = \{1, x^\alpha\} \) which consists of all monomials of \( p(x) \) with nonzero coefficients.

Note that the element 1 must in a direct basis even for a polynomial without constant part. E.g., the direct basis of \( 5 + x_1 x_2 \) is \( \{1, x_1 x_2\} \), and the direct basis of \( x_1 + x_2^2 \) is \( \{1, x_1, x_2^2\} \).

Moreover, the Gram matrix of direct basis is given by

\[
Q = \begin{bmatrix}
  c_{\alpha^0} & \frac{1}{2} c_{\alpha^1} & \cdots & \frac{1}{2} c_{\alpha^r}
  \\
  \frac{1}{2} c_{\alpha^1} & \ddots & & \\
  \vdots & & \ddots & \\
  \frac{1}{2} c_{\alpha^r} & & & 0
\end{bmatrix}
\]

where \( r + 1 \) is the length of the vector \( b(x) \) and \( (c_{\alpha^i})_{i \in [0, r]} \) is the list of coefficients of \( p \) in order of the direct basis \( b(x) \) with \( c_{\alpha^0} = 0 \) if no constant part exists in \( p \). The characteristic polynomial of \( Q \) is

\[
\chi_Q(\lambda) = (-1)^{r+1} \lambda^{-1} (\lambda^2 - c_{\alpha^0} \lambda - \frac{1}{4} \sum_{i=1}^{r} c_{\alpha^i}^2)
\]

which implies that there are only two possible non-zero eigenvalues

\[
\lambda^\pm = \frac{c_{\alpha^0} \pm \sqrt{\sum_{i=0}^{r} c_{\alpha^i}^2}}{2}
\]

with corresponding eigenvectors as

- For eigenvalue \( \lambda^+ = \frac{c_{\alpha^0} + \sqrt{\sum_{i=0}^{r} c_{\alpha^i}^2}}{2} \):

\[
v^+ = \begin{pmatrix}
  c_{\alpha^i} \left( \sqrt{\sum_{i=0}^{r} c_{\alpha^i}^2} - c_{\alpha^0} \right) \\
  \sum_{i=1}^{r} c_{\alpha^i}^2 \\
  \vdots \\
  \sum_{i=1}^{r} c_{\alpha^i}^2
\end{pmatrix}^T
\]

...
The DSOS decomposition, we have to use a valid basis with minimal degree which yields the minimal basis spectral DSOS decomposition.

In this case, the formulation (3.3) in Algorithm 3.3 is simplified as:

$$dsos(x) = \lambda^+ \frac{(b^T(x) \cdot v^+)^2}{\|v^+\|^2} + \lambda^- \frac{(b^T(x) \cdot v^-)^2}{\|v^-\|^2}.$$  

The Algorithm 3.3 with direct basis is then reduced to the direct basis spectral DSOS decomposition as follows:

**Algorithm 3.4** Direct Basis Spectral DSOS Decomposition

**Input:** Polynomial $p(x)$.

**Output:** DSOS decomposition $dsos(x)$.

**Step 1:** Get $b(x)$ as direct basis of $p(x)$.

**Step 2:** Computing $\lambda^\pm$ and $v^\pm$ via formulations (3.4)–(3.6).

**Step 3:** A DSOS decomposition of $p(x)$ is given by

$$dsos(x) = \lambda^+ \frac{(b^T(x) \cdot v^+)^2}{\|v^+\|^2} + \lambda^- \frac{(b^T(x) \cdot v^-)^2}{\|v^-\|^2}.$$  

return $dsos(x)$.

**Example 3.8.** Applying Algorithm 3.4 to $p(x) = 2 + 2x_1 + 2x_1^3 + 2x_2^2x_2$:

**Step 1:** The direct basis of $p(x)$ is $b(x) = (1, x_1, x_2, x_1^3x_2)^T$.

**Step 2:** We get $\lambda^+ = 3$, $\lambda^- = -1$, and $v^+ = (\sqrt{3}/2, \sqrt{3}/6, \sqrt{3}/6, \sqrt{3}/6)^T$, $v^- = (-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})^T$.

**Step 3:** A DSOS decomposition for $p(x)$ is

$$dsos(x) = 3\left(\frac{\sqrt{3}}{2} x_1 + \frac{\sqrt{3}}{6} x_2 + \frac{\sqrt{3}}{6} x_1^3x_2\right)^2 - \left(\frac{1}{2} x_1 + \frac{1}{2} x_2 + \frac{1}{2} x_1^3x_2\right)^2.$$  

We have the following proposition:

**Proposition 3.9.** For any polynomial $p \in \mathbb{R}[x]$ with direct basis $b(x)$, the degree of DSOS components generated by Algorithm 3.4 is not greater than $2 \deg(p)$.

**Proof.** Since $\deg(b) = \deg(p)$ for direct basis, we get from Proposition 3.6 that the degree of DSOS components is not greater than $2 \deg(p)$. \hfill $\Box$

Clearly, the computation in Algorithm 3.4 is very effective, but the degree of its DSOS components is very high as $2 \deg(p)$. In order to get a minimal degree spectral DSOS decomposition, we have to use a valid basis with minimal degree which yields the minimal basis spectral DSOS decomposition.
3.2.2. Minimal basis spectral DSOS decomposition.

Definition 3.10 (Minimal basis). For any polynomial \( p \in \mathbb{R}[x] \), a minimal basis of \( p \) is a valid basis whose degree is not greater than \( \left\lfloor \frac{\deg(p)}{2} \right\rfloor \).

E.g., the full-basis \( b(x) = (1, x, \ldots, x_n, x_1x_2, \ldots, x_1x_n)^T \) is a minimal basis for all polynomials in \( \mathbb{R}[x]_d \). However, this basis is too long with \( \left\lceil \frac{n+1}{2} \right\rceil \) elements. In fact, for most of polynomials, there exist some minimal bases as subsets of the full-basis. E.g., a minimal basis of \( x_1^2 \) is \( \{x_1\} \) and a minimal basis of \( x_1x_2 \) is \( \{x_1, x_2\} \).

We can generalize this idea to get a shorter minimal basis for any monomial \( m(x) \) with only two cases to be considered:

- Case 1: If \( \mathcal{O}(m) = \emptyset \), then a minimal basis of \( m(x) \) is \( \{x_1^\frac{n}{2}\} \).
- Case 2: Otherwise (\( \mathcal{O}(m) \neq \emptyset \)). Separate \( \mathcal{O}(m) \) into \( \mathcal{O}_1(m) \) and \( \mathcal{O}_2(m) \) verifying:
  - \( \mathcal{O}_1(m) \cap \mathcal{O}_2(m) = \emptyset \).
  - \( \mathcal{O}_1(m) \cup \mathcal{O}_2(m) = \mathcal{O}(m) \).
  - \( \mathcal{O}_1(m) \) and \( \mathcal{O}_2(m) \) belong to \( \{ \left\lceil \frac{|\mathcal{O}(m)|}{2} \right\rceil \} \cup \{ \left\lfloor \frac{|\mathcal{O}(m)|}{2} \right\rfloor \} \).

Then a minimal basis of \( m(x) \) is

\[
\left\{ x_1^\frac{n}{2} \right\} \bigcup_k x_k \bigcup_{k \in \mathcal{O}_1(m)} x_k \bigcup_{k \in \mathcal{O}_2(m)} x_k \bigcup \mathcal{O}_1(m) \bigcup \mathcal{O}_2(m)
\]

At last, a minimal basis of a polynomial is the union of minimal basis of its monomials.

E.g., for quadratic case \( p(x_1, x_2) = x_1^2 + x_2^2 - 3x_1x_2 \). We get the list of monomials of \( p \) as \( \{x_1^2, x_2^2, x_1x_2\} \). Then we get minimal basis for these monomials as: \( \{x_1\} \) for \( x_1^2 \), \( \{x_2\} \) for \( x_2^2 \), and \( \{x_1, x_2\} \) for \( x_1x_2 \). And a minimal basis of \( p \) is

\[
\{x_1\} \cup \{x_2\} \cup \{x_1, x_2\} = \{x_1, x_2\}.
\]

An example for higher-order polynomial \( p(x_1, x_2) = x_1^2x_2^6 - 2x_1^3x_2^{100} + 10 \): The list of monomials is \( \{x_1^2x_2^6, x_1^3x_2^{100}, 1\} \). We get minimal basis for these monomials as: \( \{x_1x_2\} \) for \( x_1^2x_2^6 \), \( \{x_1^2x_2^6, x_1x_2^{50}, x_1^3x_2^{50}\} \) for \( x_1^3x_2^{100} \), and \( \{1\} \) for \( 1 \). So a minimal basis of \( p \) is

\[
\{x_1x_2\} \cup \{x_1x_2^{50}, x_1^2x_2^{50}\} \cup \{1\} = \{1, x_1x_2, x_1x_2^{50}, x_1^2x_2^{50}\}.
\]

Replacing in Step 1 of Algorithm 3.3 a valid basis by a minimal basis, we will get a minimal basis spectral DSOS decomposition as follows:

**Algorithm 3.5 Minimal Basis Spectral DSOS Decomposition**

**Input:** Polynomial \( p(x) \).

**Output:** DSOS decomposition \( dsos(x) \).

**Step 1:** Compute a minimal basis \( b(x) \) and a Gram matrix \( Q \) for \( p(x) \).

**Step 2:** Spectral decomposition of \( Q \) to get \( P \) and \( \Lambda \) verifying

\[
Q = P \cdot \Lambda \cdot P^T.
\]

**Step 3:** Compute \( y(x) = P^T \cdot b(x) \) and \( \mathcal{K} = \{ k \in [1, r] : \lambda_k \neq 0 \} \), a DSOS decomposition of \( p(x) \) is given by

\[
dsos(x) = \sum_{i \in \mathcal{K}} \lambda_i y_i^2(x).
\]

**return** \( dsos(x) \).
Example 3.11. Applying Algorithm 3.5 to \( p(x) = 2 + 2x_1 + 2x_2^3 + 2x_1^2x_2 \):

**Step 1:** We get a minimal basis of \( p \) and its corresponding Gram matrix as

\[
\begin{bmatrix}
1 \\
x_2^2 \\
x_2 \\
x_1 \\
x_1x_2
\end{bmatrix},
\begin{bmatrix}
2 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}.
\]

**Step 2:** Spectral decomposition on \( Q \) to get

\[
P = \begin{bmatrix}
0.8877 & 0 & -0.3971 & 0 & -0.2332 \\
0 & 0.7071 & 0 & -0.7071 & 0 \\
0 & 0.7071 & 0 & 0.7071 & 0 \\
0.4271 & 0 & 0.5207 & 0 & 0.7392 \\
0.1721 & 0 & 0.7558 & 0 & -0.6318
\end{bmatrix},
\]

\( \Lambda = \text{diag}(2.4812, 1, 0.6889, -1, -1.1701) \).

**Step 3:** Set \( y(x) = P^T \cdot b(x) \) and \( K = [1, 5] \). A DSOS decomposition of \( p \) is

\[
dsos(x) = s_1(x) - s_2(x)
\]

with

\[
s_1(x) = (1.39821 + 0.6728x_1 + 0.2712x_1x_2)^2 + (0.7071x_2 + 0.7071x_2^2)^2 + (-0.32961 + 0.4321x_1 + 0.6273x_1x_2)^2,
\]

\[
s_2(x) = (0.7071x_2 - 0.7071x_2^2)^2 + (-0.25221 + 0.7996x_1 - 0.6834x_1x_2)^2.
\]

The degree of DSOS components equals to 4 which is exactly \( 2\lceil \frac{\deg(p)}{2} \rceil \). We can prove this result in the next proposition.

**Proposition 3.12.** For any polynomial \( p \in \mathbb{R}[x] \) with a minimal basis \( b(x) \), the degree of DSOS components generated by Algorithm 3.5 is equal to \( 2\lceil \frac{\deg(p)}{2} \rceil \).

**Proof.** Based on the definition of minimal basis, \( \deg(b) \leq \lceil \frac{\deg(p)}{2} \rceil \). Then the degree of DSOS components is not greater than \( 2\lceil \frac{\deg(p)}{2} \rceil \) based on the Proposition 3.6. However, according to Theorem 2.14, the degree of DSOS components should be greater than \( 2\lceil \frac{\deg(p)}{2} \rceil \). Therefore, we conclude that the degree of DSOS components equals to \( 2\lceil \frac{\deg(p)}{2} \rceil \).

Note that using minimal basis spectral DSOS decomposition for a quadratic form in \( x \) will get a diagonal form in \( y \) (i.e., spectral decomposition for quadratic form) which is both DSOS and DCSOS decompositions. Therefore, this decomposition can be considered as a generalization of spectral decomposition for general polynomials.

**3.3. Comparisons of DSOS decomposition algorithms.** In previous two subsections, we have proposed two types of practical DSOS decomposition algorithms: parity decompositions and spectral decompositions. In this section, we will compare different aspects of these algorithms to help readers make a suitable choice in practice.

Firstly, the degree of DSOS components generated by these algorithms are illustrated in Table 3.1.
Table 3.1
Degree of DSOS components with various practical DSOS decompositions

| DSOS Algorithms | Degree of DSOS components |
|-----------------|----------------------------|
| Parity          |                            |
| Algorithm 3.1   | $\deg(p) + \deg(o)$       |
| Algorithm 3.2   | $2\left\lceil \frac{\deg(p)}{2} \right\rceil$ |
| Spectral        |                            |
| Algorithm 3.3   | $2\deg(b)$                |
| Algorithm 3.4   | $2\deg(p)$                |
| Algorithm 3.5   | $2\left\lceil \frac{\deg(p)}{2} \right\rceil$ |

We observe that Algorithm 3.2 (improved parity decomposition) and Algorithm 3.5 (minimal basis spectral decomposition) will get the best degree DSOS decompositions, whereas the Algorithm 3.4 (direct basis spectral decomposition) leads to highest degree decompositions. In all other cases, the degree of DSOS components is bounded in $\left[2\left\lceil \frac{\deg(p)}{2} \right\rceil, 2\deg(p)\right]$.

Concerning on the number of squares in DSOS decompositions for a polynomial of degree $d$ with $n$ variables and the number of monomials is $J$.

1. For parity DSOS decomposition algorithms:
   - In Algorithm 3.1: We have 3 squares for each monomial, thus for a polynomial with $J$ monomials, we have totally $3J$ squares.
   - In Algorithm 3.2: For each monomial $m_i, i \in [1, J]$, the procedure D will construct $\left\lceil \frac{|O(m_i)|}{2} \right\rceil$ pairs which yields at most $2\left\lceil \frac{|O(m_i)|}{2} \right\rceil$ squares for monomial $m_i$. Therefore, we have at most $\sum_{i=1}^{J} 2\left\lceil \frac{|O(m_i)|}{2} \right\rceil$ squares for polynomial. Since $|O(m_i)| \leq n, i \in [1, J]$, the number of squares for polynomial is limited by $2\left\lceil \frac{n}{2} \right\rceil J$.

2. For spectral DSOS decomposition algorithms:
   - In Algorithm 3.3: Based on formulation (3.3), the number of squares is exactly equal to the number of non-zero eigenvalues of the Gram matrix $Q$, i.e., the number of squares is equal to $|K|$ which is limited by the length of valid basis. So $|K|$ is limited by the length of full-basis $\binom{n + \frac{d}{2}}{n}$.
   - In Algorithm 3.4: We have only 2 squares since only two simple non-zero eigenvalues exist.
   - In Algorithm 3.5: The number of squares is equal to $|K|$ which is limited by the length of the minimal basis. Based on the construction process of minimal basis, the length of minimal basis is not exceed either the length of full-basis nor twice of the number of monomials. Therefore, the number of squares for a polynomial is not greater than $\min\{2J, \binom{n + \frac{d}{2}}{n}\}$.

The complexity of these algorithms is highly depending on the number of squares and the complexity for computing each square. In practice, the Algorithm 3.5 (minimal basis spectral decomposition) seems to be the best choice since it could produce smallest degree DSOS decomposition with relatively less number of squares.

4. Practical DCSOS decompositions. The DSOS decompositions constructed in previous section are in general not DCSOS decompositions. We are going to investigate practical DCSOS decomposition technique without solving SDP in this section.

4.1. Parity DCSOS decomposition. Any monomial $x^\alpha$ can be decomposed by the multiplications of three elementary cases: $x_i x_j$, $x_i^{2k}$, $k \in \mathbb{N}$ and $p \times q$ with $(p, q) \in \mathbb{DCSOS}^2$ whose DCSOS decompositions can be computed by:

• For $x_i x_j$:

\begin{align}
(4.1) & \quad x_i x_j = \frac{1}{4}(x_i + x_j)^2 - \frac{1}{4}(x_i - x_j)^2. \\
(4.2) & \quad x_i x_j = \frac{1}{2}(x_i + x_j)^2 - \frac{1}{2}(x_i^2 + x_j^2).
\end{align}

A single variable $x_i$ is a special case of $x_i x_j$ with $x_j = 1$. The degree of DCSOS components is 2.

• For $x_i^{2k}, k \in \mathbb{N}$:

\begin{equation}
(4.3) \quad x_i^{2k} = \frac{x_i^{2k}}{\text{CSOS}} - \frac{0}{\text{CSOS}}.
\end{equation}

The degree of DCSOS components is $2k$.

• For $p \times q$ with $(p, q) \in \text{DCSOS}^2$: Let us denote the DCSOS decompositions of $p$ and $q$ as $p = p_1 - p_2$ and $q = q_1 - q_2$. We can apply the formulation (2.3) to get a DCSOS decomposition of $p \times q$ with DCSOS components of degree $2 \max\{\deg(p_1), \deg(p_2), \deg(q_1), \deg(q_2)\}$.

The idea of parity DCSOS decomposition consists of making pairs of variables in $x^\alpha$ as multiplications of elementary cases to get its DCSOS decomposition. Then any polynomial as a linear combination of monomials is also a DCSOS polynomial. The detailed algorithm is described as follows:

**Algorithm 4.1 Parity DCSOS Decomposition**

**Input:** Monomial $m(x)$.

**Output:** DCSOS decomposition $dcsos(x)$.

**Step 1:** Extract $x^\alpha$ and $c_\alpha$ for $m(x)$.

**Step 2:** Make pairs of $x^\alpha$ as multiplications of elementary cases.

**Step 3:** Use formulations (2.3) and (4.1)–(4.3) consecutively to get a DCSOS decomposition for $x^\alpha$ as

\[ x^\alpha = s_1(x) - s_2(x). \]

**Step 4:** A DCSOS decomposition of $m(x)$ is given by

\[ dcsos(x) = c_\alpha s_1(x) - c_\alpha s_2(x). \]

return $dcsos(x)$.

**Example 4.1.** Applying Algorithm 4.1 to $m(x) = 3x_1 x_2^2$:

**Step 1:** $c_\alpha = 3$ and $x^\alpha = x_1 x_2^2$.

**Step 2:** Separate $x^\alpha$ into elementary cases as

\[ x^\alpha = (x_1)(x_2^2). \]

**Step 3:** We use formulations (4.1) and (4.3) to get

\[ x^\alpha = \left(\frac{1}{4}(x_1 + 1)^2 - \frac{1}{4}(x_1 - 1)^2\right)(x_2^2 - 0). \]
Then using formulation (2.3), we obtain a DCSOS decomposition for $x^\alpha$ as

$$
\begin{align*}
\begin{array}{c}
x^\alpha = \frac{1}{2} \left[ \left( \frac{1}{4} (x_1 + 1)^2 + x_2^2 \right)^2 + \left( \frac{1}{4} (x_1 - 1)^2 \right)^2 \right] \\
- \frac{1}{2} \left[ \left( \frac{1}{4} (x_1 + 1)^2 \right)^2 + \left( x_2^2 + \frac{1}{4} (x_1 - 1)^2 \right)^2 \right].
\end{array}
\end{align*}
$$

**Step 4:** A DCSOS decomposition for $m(x)$ is

$$
\begin{align*}
\begin{array}{c}
dcsos(x) = \frac{3}{2} \left[ \left( \frac{1}{4} (x_1 + 1)^2 + x_2^2 \right)^2 + \left( \frac{1}{4} (x_1 - 1)^2 \right)^2 \right] \\
- \frac{3}{2} \left[ \left( \frac{1}{4} (x_1 + 1)^2 \right)^2 + \left( x_2^2 + \frac{1}{4} (x_1 - 1)^2 \right)^2 \right].
\end{array}
\end{align*}
$$

In Algorithm 4.1, the degree of DCSOS components is not easy to be determined since it depends on two important factors:

The separations of $x^\alpha$ into elementary cases. E.g., for $x^\alpha = x_1 x_2$, if we separate as $(x_1)(x_2)$, then applying Algorithm 4.1, a DCSOS decomposition is

$$
\begin{align*}
x_1 x_2 = \frac{1}{2} \left[ \left( \frac{(x_2 + 1)^2}{4} + \frac{(x_1 + 1)^2}{4} \right)^2 + \left( \frac{(x_2 - 1)^2}{4} + \frac{(x_1 - 1)^2}{4} \right)^2 \right] \\
- \frac{1}{2} \left[ \left( \frac{(x_2 + 1)^2}{4} \right)^2 + \left( \frac{(x_1 - 1)^2}{4} \right)^2 + \left( \frac{(x_2 - 1)^2}{4} \right)^2 + \left( \frac{(x_1 + 1)^2}{4} \right)^2 \right].
\end{align*}
$$

whose degree of DCSOS components is 4 greater than 2 (the degree of the DCSOS components given in formulation (4.1) and (4.2)).

Therefore, we propose an heuristic **Procedure S** to separate $x^\alpha$ as follows:

**Procedure S: (Separation of $x^\alpha$)**

- Compute the odd index set $O(x^\alpha)$ to get a parity separation of $x^\alpha$ as:

$$
o(x) = \prod_{i \in O(x^\alpha)} x_i; e^2(x) = x^\alpha / o(x).
$$

- In $o(x)$: We follow the Steps 1 and 2 of Procedure D to make pairs for $o(x)$, then using formulations (4.1) or (4.2) to get their DCSOS decompositions.

- In $e^2(x)$: We make pairs as $x_i^2$, then using formulation (4.3) to get their DCSOS decompositions.

The degree of DCSOS components for each pair in Procedure S is equal to 2 or 0.
The order of multiplications. E.g., for $x^\alpha = x_1^4x_2^2x_3^3$, if we multiply by the order \[
\left(\left(\frac{x_1^4x_2^2}{x_1^3}\right)\right),
\]
then we get a DCSOS decomposition as
\[
\left(\frac{x_3^2 + \left(x_2^2 + x_1^4\right)^2}{2}\right)^2 + \left(\frac{x_4^2 + x_3^8}{4}\right)^2 - \left(\frac{x_3^2 + x_4^2 + x_1^4}{2}\right)^2 + \left(\frac{x_2^2 + x_1^4}{4}\right)^4
\]
with DCSOS components of degree 16. Whereas, if we multiply by the order \[
\left(\left(\frac{x_1^4}{x_2^2x_3^3}\right)\right),
\]
then we get a DCSOS decomposition as
\[
\left(\frac{x_3^2 + \left(x_2^2 + x_3^4\right)^2}{2}\right)^2 + \left(\frac{x_4^2 + x_4^4}{4}\right)^2 - \left(\frac{x_3^2 + x_4^2 + x_4^4}{2}\right)^2 + \left(\frac{x_2^2 + x_3^4}{4}\right)^4
\]
with DCSOS components of degree 8.

Therefore, to get a minimal degree decomposition, we have to multiply from the smallest degree DCSOS components. Let $L = [s_1(x), \ldots, s_r(x)]$ be a list of $r$ DCSOS polynomials given by Procedure S. Then we propose the Procedure M for computing a DCSOS decomposition for \(\prod_{i=1}^r L(i)\) in the best order.

**Procedure M: (Order of Multiplications)**

**Step 1:** Sort the list $L$ in an increasing order by the degree of DCSOS components.

**Step 2:** If $|L| == 1$ Then STOP and Return $L(1)$.

Else

- Compute a DCSOS decomposition for $L(1) \times L(2)$ by formulation (2.3) to get $dcos(x)$.
- $L \leftarrow (L \setminus [L(1), L(2)]) \cup dcos(x)$.
- Goto Step 1.

Introducing Procedure S and Procedure M into Algorithm 4.1, we will get an improved parity DCSOS decomposition described in Algorithm 4.2 whose degree of DCSOS components is given in Proposition 4.2.

**Proposition 4.2.** For any monomial $m \in \mathbb{R}[x]$, the degree of DCSOS components generated by Algorithm 4.2 is not greater than
\[
\begin{cases}
0, & \text{if } \deg(m) = 0, \\
2, & \text{if } \deg(m) = 1, \\
2^{\lceil \log_2(\deg(m)) \rceil}, & \text{if } \deg(m) \geq 2.
\end{cases}
\]
Algorithm 4.2 Improved Parity DCSOS Decomposition

Input: Monomial $m(x)$.

Output: DCSOS decomposition $dc sos(x)$.

Step 1: Extract $x^\alpha$ and $c_\alpha$ for $m(x)$.

Step 2: Using Procedure $S$ to make pairs for $x^\alpha$ and get a list of DCSOS decompositions for each pairs as $L = [s_1(x), \ldots, s_\ell(x)]$.

Step 3: Using Procedure $M$ to get a DCSOS decomposition for $x^\alpha$ as

$$x^\alpha = \underbrace{p(x)}_{\text{cos}} - \underbrace{q(x)}_{\text{cos}}.$$

Step 4: A DCSOS decomposition of $m(x)$ is given by

$$dc sos(x) = c_\alpha p(x) - c_\alpha q(x).$$

return $dc sos(x)$.

Proof. The result is obviously true for $\deg(m) \in \{0,1\}$. Now, considering $\deg(m) \geq 2$, in Procedure $S$, we have a list of $\left\lceil \frac{\deg(m)}{2} \right\rceil$ DCSOS polynomials of degree 2. Then in Procedure $M$, we will firstly reduce to $\left\lceil \frac{\deg(m)}{2^2} \right\rceil$ DCSOS polynomials of degree 2, then reduce to $\left\lceil \frac{\deg(m)}{2^3} \right\rceil$ DCSOS polynomials of degree 2 etc. By recurrence, once $\left\lceil \frac{\deg(m)}{2^r} \right\rceil$ equals to 1, then $r = \lceil \log_2(\deg(m)) \rceil$ and we get the resulting DCSOS decomposition of degree $2^\lceil \log_2(\deg(m)) \rceil$.

Example 4.3. Applying Algorithm 4.2 to $m(x) = -2x_1^3x_2x_3^2$:

Step 1: $x^\alpha = x_1^3x_2x_3^2$ and $c_\alpha = -2$.

Step 2: Using Procedure $S$, we make pairs to $x^\alpha$ as $(x_1x_2)(x_1^2)(x_3^2)$. The list of DCSOS decompositions for all pairs is computed by formulations (4.1)-(4.3) as $L = [s_1, s_2, s_3]$ with $s_1 = \frac{1}{2}(x_1 + x_2)^2 - \frac{1}{4}(x_1 - x_2)^2, s_2 = x_1^2$ and $s_3 = x_3^2$.

Step 3: Using Procedure $M$, we get a DCSOS formulation of $x^\alpha$ as

$$dc sos(x) = \left[\left(x_3^2 + \left(\frac{(x_2 + x_1)^4}{16} + \frac{(x_1 - x_2)^2}{2} + \frac{x_1^2}{4}\right)^2\right) - \left(x_3^2 + \left(\frac{(x_2 + x_1)^4}{16} + \frac{(x_1 - x_2)^2}{2} + \frac{x_1^2}{4}\right)^2\right)\right].$$

Step 4: A DCSOS decomposition of $m(x)$ is

$$dc sos(x) = \left[\left(x_3^2 + \left(\frac{(x_2 + x_1)^4}{16} + \frac{(x_1 - x_2)^2}{2} + \frac{x_1^2}{4}\right)^2\right) - \left(x_3^2 + \left(\frac{(x_2 + x_1)^4}{16} + \frac{(x_1 - x_2)^2}{2} + \frac{x_1^2}{4}\right)^2\right)\right].$$
Clearly, \( \deg(m) = 6 \) and the degree of DCSOS components is \( 2^{\lceil \log_2(\deg(m)) \rceil} = 8 \).

We can see from Proposition 4.2 that the degree of DCSOS decomposition for monomial \( m \in \mathbb{R}[x] \) by using Algorithm 4.2 is greater than \( 2^{\lceil \deg(m) \rceil} \) if \( \deg(m) \notin 2^\mathbb{N} \). The gap of degrees is of order \( O(\deg(m)) \). The major reason of this defect lies in the use of the formulation (2.3) for computing a DCSOS decomposition for the multiplications of DCSOS polynomials. Since as we have explained in the third elementary case that the degree of DCSOS components of \( pq \) is \( 2 \max\{\deg(p_1), \deg(p_2), \deg(q_1), \deg(q_2)\} \) which is greater than \( 2^{\lceil \deg(\alpha) \rceil} \). In order to get a practical algorithm for generating a DCSOS decomposition with smallest degree. We should find a better formulation for multiplications of DCSOS polynomials instead of the formulation (2.3).

### 4.2. Practical DCSOS decomposition with minimal degree.

**Theorem 4.4.** Let \( x \in \mathbb{R}^n, \alpha \in \mathbb{N}^n \) and \( N = [1, n] \), for all positive integers \( m \) and \( n \), we have the identity

\[
\sum_{A \subseteq N} (-1)^{|A|} \left( \sum_{j \in A} x_j \right)^m = (-1)^n \sum_{|\alpha|=m, \alpha \in (N^n)} \binom{m}{\alpha} x^\alpha
\]

where \( |\alpha| = \sum_{i=1}^n \alpha_i \), \( x^\alpha = \prod_{i=1}^n x_i^{\alpha_i} \) and \( \binom{m}{\alpha} \) denotes the multinomial coefficient

\[
\binom{m}{\alpha} = \frac{m!}{\alpha_1!\alpha_2!\ldots \alpha_n!}
\]

under the convention that \( \binom{m}{\alpha} = 0 \) if \( \alpha \in \mathbb{N}^n \) with \( |\alpha| = m < n \).

**Proof.** Let us denote the left part of (4.4) as \( f(x) \) which is clearly a homogeneous polynomial of degree \( m \). We firstly prove that \( f(x) \) is a multiple of \( \prod_{i=1}^n x_i \). It is easy to see that if we set \( x_1 = 0 \) then \( f(x) = 0 \), because in each term \( (-1)^{|A|} \left( \sum_{j \in A \setminus \{1\}} x_j \right)^m \), if \( 1 \in A \), then \( x_1 = 0 \) will turn the term to \( (-1)^{|A|} \left( \sum_{j \in A \setminus \{1\}} x_j \right)^m \) which cancels out the same term in \( f(x) \) with opposite sign \( (-1)^{|A\setminus\{1\}|} \); otherwise, we can add \( x_1 \) to get \( (-1)^{|A|} \left( \sum_{j \in A \cup \{1\}} x_j \right)^m \) which cancels out the same term in \( f(x) \) with opposite sign \( (-1)^{|A\cup\{1\}|} \). So \( x_1 \) must be a divisor of \( f(x) \) and by the same argument also \( x_2, \ldots, x_n \). Hence \( f(x) \) is a multiple of \( \prod_{i=1}^n x_i \).

Next, we have three cases to prove the formulation (4.4).

(i) If \( m < n \), then \( \deg(f) = m < \deg(\prod_{i=1}^n x_i) = n \). An \( m \) degree polynomial \( f(x) \) is a multiple of a \( n \) degree polynomial \( \prod_{i=1}^n x_i \) with \( m < n \) implies

\[
\sum_{A \subseteq N} (-1)^{|A|} \left( \sum_{j \in A} x_j \right)^m = 0
\]

(ii) If \( m = n \), then \( \deg(f) = \deg(\prod_{i=1}^n x_i) = n \). In this case, \( f(x) \) as a multiple of \( \prod_{i=1}^n x_i \) must be of the form \( c \times \prod_{i=1}^n x_i \) with \( c \in \mathbb{Z} \). This term can be only provided by \( (-1)^{|N|}(\sum_{j \in N} x_j)^m \) whence \( c = (-1)^n m! \) and

\[
\sum_{A \subseteq N} (-1)^{|A|} \left( \sum_{j \in A} x_j \right)^n = (-1)^n m! \prod_{i=1}^N x_i.
\]
(iii) For \( m > n \), since \( f(x) \) is a homogeneous polynomial of degree \( m \) and a multiple of \( \prod_{i=1}^{n} x_i \), then its form must be \( \sum_{\alpha=\vec{m}, \alpha \in \{N_+\}^n} c_\alpha x^\alpha \). So \( x^\alpha \) is a monomial consists of all \( x_i, i \in N \) which can be only provided by \((-1)^{|N|}(\sum_{j \in N} x_j)^m\). Expanding this by multinomial theorem to get
\[
(-1)^{|N|}(\sum_{j \in N} x_j)^m = (-1)^n \sum_{|\alpha|=m} \binom{m}{\alpha} x^\alpha,
\]
we have the coefficient of \( x^\alpha \) as \((-1)^n \binom{m}{\alpha}\) which yields the identity
\[
\sum_{A \subseteq N} (-1)^{|A|} \left( \sum_{j \in A} x_j \right)^m = (-1)^n \sum_{|\alpha|=m, \alpha \in \{N_+\}^n} \binom{m}{\alpha} x^\alpha.
\]
From (i), (ii) and (iii), we conclude the same identity (4.4) for all \( m \) and \( n \).

Now, if we replace in identity (4.4) all \( x_i, i \in N \) by CSOS polynomials \( p_i, i \in N \) and let \( m = n \), then we get an expansion of \( \prod_{i=1}^{n} p_i \) as
\[
\prod_{i=1}^{n} p_i = \frac{1}{n!} \sum_{A \subseteq N} (-1)^{|A|+n} \left( \sum_{j \in A} p_j \right)^n.
\]
Clearly, \( \left( \sum_{j \in A} p_j \right)^n \) is also a CSOS polynomial, and the formulation (4.7) gives a DCSOS decomposition for \( \prod_{i=1}^{n} p_i \). Moreover, if \( \deg(p_i), i \in N \) are all the same, we get the degree of \( \left( \sum_{j \in A} p_j \right) \) equals to the degree of \( \prod_{i=1}^{n} p_i \). Then the formulation (4.7) gives a minimal degree DCSOS decomposition for \( \prod_{i=1}^{n} p_i \). Based on this fact, we propose a minimal degree DCSOS decomposition described in Algorithm 4.3 and the minimal degree of DCSOS components is proved in Proposition 4.5.

**Algorithm 4.3 Minimal Degree DCSOS Decomposition**

**Input:** Monomial \( m(x) \).

**Output:** DCSOS decomposition \( dcsos(x) \).

**Step 1:** Extract \( x^\alpha \) and \( c_\alpha \) for \( m(x) \).

**Step 2:** Use Procedure S to make pairs for \( x^\alpha \) and get a list of DCSOS decompositions for each pairs as \( L = [g_1(x) - h_1(x), \ldots, g_r(x) - h_r(x)] \).

**Step 3:** Set \( R = [1, r] \), then for each \( A \subseteq R \), using formulation (4.7) to get a DCSOS decomposition for
\[
\prod_{i \in R \setminus A} g_i \prod_{j \in A} h_j = \frac{\tilde{g}_A(x) - \tilde{h}_A(x)}{c_{\text{CSOS}}}
\]

**Step 4:** A DCSOS decomposition for \( m(x) \) is given by
\[
dcsos(x) = c_\alpha \sum_{A \subseteq R} (-1)^{|A|} (\tilde{g}_A(x) - \tilde{h}_A(x)).
\]

**return** \( dcsos(x) \).

**Proposition 4.5.** For any monomial \( m \in \mathbb{R}[x] \), the degree of DCSOS components generated by Algorithm 4.3 is equal to \( 2\left\lceil \frac{\deg(m)}{2} \right\rceil \).
Algorithm 4.3 that

Using formulation (4.7), we get a DCSOS decomposition for each \( \prod_{i \in R \setminus A} g_i \prod_{j \in A} h_j \) with at most 2r degree in DCSOS components, which proves that the degree of DCSOS components for m is 2r.

**Example 4.6.** Applying Algorithm 4.3 to monomial \( m(x) = -2x_1^2x_2x_3^2 \):

**Step 1:** We get \( x^\alpha = x_1^3x_2x_3^2 \) and \( c_\alpha = -2 \).

**Step 2:** Using Procedure S, we make pairs of \( x^\alpha \) as

\[ x^\alpha = (x_1x_2)(x_1^2)(x_3^2). \]

The list of DCSOS decompositions for all pairs is computed by formulations (4.1)--(4.3) as

\[ L = [s_1, s_2, s_3] \]

with \( s_1 = \frac{1}{3}(x_1 + x_2)^2 \), \( s_2 = \frac{1}{3}(x_1 - x_2)^2 \), \( s_3 = \frac{1}{3}x_1^2 \), and \( s_3 = \frac{1}{3}x_3^2 \).

**Step 3:** Note \( R = \{1, 2, 3\} \), we compute by formulation (4.7) that

\[ g_1g_2g_3 = \frac{1}{3!}(g_1 + g_2 + g_3)^3 + g_1^3 + g_2^3 + g_3^3 - \frac{1}{3!}(g_1 + g_2)^3 + (g_1 + g_3)^3 + (g_2 + g_3)^3 \]

\[ h_1g_2g_3 = \frac{1}{3!}(h_1 + g_2 + g_3)^3 + h_1^3 + g_2^3 + g_3^3 - \frac{1}{3!}(h_1 + g_2)^3 + (h_1 + g_3)^3 + (g_2 + g_3)^3 \]

and all other combinations are zeros since \( h_2 \) and \( h_3 \) are zeros.

**Step 4:** A DCSOS decomposition of \( m(x) \) is given by

\[ \text{dc sos}(x) = -2((-1)^0(\bar{g}_0 - \bar{h}_0) + (-1)^1(\bar{g}_1 - \bar{h}_1)) = 2(\bar{g}_1 + \bar{h}_0) - 2(\bar{g}_0 + \bar{h}_1) \]

with

\[ 2(\bar{g}_1 + \bar{h}_0) = \frac{(x_1 - x_2)^6}{192} + \frac{(x_1^2 + x_3^2)^3}{3} + \frac{(x_1 + x_2)^3}{4} \]

\[ + \frac{x_1^6 + x_3^6}{3} + \frac{(x_1 - x_2)^2 + x_1^2 + x_3^2}{3} \quad \in \text{CSOS}, \]

\[ 2(\bar{g}_0 + \bar{h}_1) = \frac{(x_1 - x_2)^2}{4} + \frac{x_1^2}{3} + \frac{(x_1^2 + x_3^2)^3}{3} \]

\[ + \frac{(x_1 + x_2)^3}{3} + \frac{(x_1^2 + x_3^2)^3}{3} \quad \in \text{CSOS} \].
Note that, based on formulation (4.7), it is also possible to get a DCSOS decomposition for any monomial \( x^\alpha \) without using Procedure S as follows: Let us denote \( x^\alpha \) as \( \prod_{i=1}^{2^{|\alpha|}} y_i \), defined by

\[
2^{|\alpha|} \prod_{i=1}^{2^{|\alpha|}} y_i = \begin{cases} 
  x_1 \times \cdots \times x_1 \times \cdots \times x_n , & \text{if } |\alpha| \text{ is even;} \\
  x_1 \times \cdots \times x_1 \times \cdots \times x_n \times 1 , & \text{otherwise.}
\end{cases}
\]

Then apply formulation (4.7) to \( \prod_{i=1}^{2^{|\alpha|}} y_i \), we get a DCSOS formulation for \( x^\alpha \) as

\[
x^\alpha = \frac{1}{(2^{|\alpha|})!} \sum_{A \subseteq N} (-1)^{|A| + 2^{|\alpha|}} \left( \sum_{j \in A} y_j \right) 2^{|\alpha|} \tag{4.8}
\]

where \( N = \{1, \ldots, 2^{|\alpha|}\} \). Clearly, the degree of the DCSOS decomposition is also equal to the smallest degree \( 2^{|\alpha|} \).

4.3. Comparisons of DCSOS decompositions. In this section, we have proposed several practical DCSOS decomposition techniques: Parity DCSOS decomposition (Algorithm 4.1), Improved Parity DCSOS decomposition (Algorithm 4.2), Minimal Degree DCSOS decompositions (Algorithm 4.3 and the variation using formulation (4.8)). Based on Propositions 4.2 and 4.5, Algorithm 4.3 and formulation (4.8) will generate best degree of DCSOS decompositions.

Concerning on the number of squares in DCSOS decompositions for a polynomial with \( J \) monomials.

- In Algorithms 4.1 and 4.2: For each monomial, since we use formulation (2.3) to compute the multiplications of DCSOS polynomials, therefore, we get 4 squares for each monomial in its DCSOS decomposition. And for a polynomial with \( J \) monomials, we will get \( 4J \) squares.
- In Algorithm 4.3: For a monomial \( m_i \), we have \( r = \frac{\deg(m_i)}{2} \) pairs in Step 2, so \( 2^r \) subsets \( A \) for the set \( R = [1, r] \). Based on formulation (4.7), we have at most \( 2^r \) squares for each subset \( A \) which yields \( 4^r \) squares for \( m_i \). And for a polynomial with \( J \) monomials \( m_1, \ldots, m_J \), we will have at most \( \sum_{i=1}^{J} 4^{\frac{\deg(m_i)}{2}} \) squares.
- In formulation (4.8): For a monomial \( m_i \), we have \( 2^{\frac{\deg(m_i)}{2}} \) subsets \( A \) for the set \( N = \{1, \ldots, 2^{\frac{\deg(m_i)}{2}}\} \) which yields \( 4^{\frac{\deg(m_i)}{2}} \) squares for \( m_i \). And for a polynomial with \( J \) monomials \( m_1, \ldots, m_J \), we have \( \sum_{i=1}^{J} 4^{\frac{\deg(m_i)}{2}} \) squares.

As in DSOS decompositions, the complexity for DCSOS algorithms is also depending on the number of squares and the complexity for computing each square. In practice, if the degree of DCSOS decomposition is crucial, we suggest using Algorithm 4.3 or formulation (4.8) as well as solving SDP for DCSOS decompositions. Otherwise, it is worth testing all of these algorithms in practice. E.g., when using DCA for solving polynomial optimization, it is still unclear how the different DC decompositions affect the convergence and the globality of the computed solution. Finding a best DC decomposition for a DC function is still an open question [21]. The deep-going analysis of these DCSOS decompositions combing with DCA for polynomial optimization and their performance in numerical simulations will be reported in our future work.
5. Conclusion and Perspectives. In this paper, we have proposed and investigated the sets of CSOS, DSOS and DCSOS polynomials, and discussed their relationships to the sets of SOS, SOS-convex and PSD polynomials. We proved that $\text{CSOS}_n$ is a proper cone of $\mathbb{R}[x]$, while $\text{DSOS}_n$ and $\text{DCSOS}_n$ are vector spaces of $\mathbb{R}[x]$ with equivalence as $\mathbb{R}[x] = \text{DSOS}_n = \text{DCSOS}_n$. As an important result, DSOS and DCSOS decompositions for any polynomial can be done in polynomial time by solving SDPs. We also proposed some practical algorithms without solving SDP for DSOS and DCSOS decompositions, the degree of each decomposition, the number of squares and the computational complexity are compared and discussed. Our codes are implemented in MATLAB [24] and in Maxima [25] which will be published soon, a C++ version is under development.

Our future works are related on various aspects. The Holy Grail which motivates the research in this paper is to solve polynomial optimization via DC programming approaches. As an application, we have applied in [29] our DCSOS decomposition Algorithm 4.2 to solve the quadratic eigenvalue complementarity problem (a fourth-ordered polynomial optimization) using DCA, the numerical results demonstrated a good performance of DCA on both the convergence rate and the quality of the computed solutions which outperformed many existing methods. The next step, we are going to develop DC programming algorithms for solving general polynomial optimization (involving convex constrained polynomial optimization and nonconvex polynomial constrained polynomial optimization). More applications arising in AI, Machine learning, engineering, optimal control and Big Data etc will be investigated. Moreover, we would like to compare the performance of our algorithm with respect to other exiting techniques such as Gloptipoly [12] (SOS-moment relaxation) and IPOPT [43] (Interior point method) etc. Concerning on the open questions, it is worth to prove the complexity for certificating CSOS polynomials, which amounts to prove the existence of a convex but not SOS-convex SOS polynomial that can not be checked in polynomial time. Another open question is how to find the best DC decomposition for DCA. We have understood by now in our recent work [29] that a best DC decomposition for DCA must be an undominated DC decomposition whose DC components are undominated convex functions. However, how to generate undominated DC decomposition for polynomials is still an open question which requires more investigations in future. Concerning on finding global optimal solutions for polynomial optimization, we are going to combine DCA (for upper bound solution) with global optimization techniques such as SDP relaxations (for lower bound and initial point estimation), Branch-and-Bound/Lasserre’s hierarchy (for global optimality) [18, 19, 20] etc. The local and global DC programming approaches for continuous polynomial optimization could be also extended to mixed-integer polynomial optimization. Moreover, solving optimization problems involving DSOS polynomials without convexity seems to be also an interesting problem which leads to a new type of mathematical programming problem with deep interactions with sums-of-squares. Researches in these directions will be reported subsequently in our future works.

Acknowledgments. This work is partially supported by the Natural Science Foundation of China (Grant No: 11601327) and by the Key Construction National “985” Program of China (Grant No: WF220426001).

REFERENCES

[1] A. A. Ahmadi and G. Hall, $Dc$ decomposition of nonconvex polynomials with algebraic tech-
28

Y. S. NIU

[2] A. A. AHMAD, A. OLSHEVSKY, P. A. PARRILLO, AND J. N. TSITSKILIS, Np-hardness of deciding convexity of quartic polynomials and related problems, Mathematical Programming, 137 (2013), pp. 453–476.

[3] A. A. AHMAD AND P. A. PARRILLO, A complete characterization of the gap between convexity and sos-convexity, SIAM J. Optim., 23 (2011), pp. 811–833.

[4] A. A. AHMAD AND P. A. PARRILLO, A convex polynomial that is not sos-convex, Math. Program., 135 (2012), pp. 275–292.

[5] S. J. BENSON, Y. YE, AND X. ZHANG, Solving large-scale sparse semidefinite programs for combinatorial optimization, SIAM Journal on Optimization, 10 (2000), pp. 443–461, http://www.mcs.anl.gov/hs/software/DSDP/.

[6] D. P. BERTSEKAS, Convex Optimization Theory, Athena Scientific, 1st ed., 2009.

[7] G. BLEKHERMAN, Convex forms that are not sums of squares, Mathematics, (2009).

[8] B. BORCHERS, Csdp, a c library for semidefinite programming, Optimization Methods & Software, 11 (1999), pp. 613–623, https://projects.coin-or.org/Csdp/.

[9] Gurobi, Gurobi 8.0, http://www.gurobi.com/.

[10] P. HARTMAN, On functions representable as a difference of convex functions, Pacific Journal of Mathematics, 9 (1959), pp. 707–713.

[11] J. W. HELTON AND J. NIE, Semidefinite representation of convex sets, Mathematical Programming, 122 (2010), pp. 21–64.

[12] D. HENRION, J. B. LASSEURRE, AND J. LOEFBERG, Gloptipoly 3: moments, optimization and semidefinite programming, Optimization Methods and Software, 24 (2009), pp. 761–779, http://homepages.laas.fr/henrion/software/gloptipoly/.

[13] D. HILBERT, Über die darstellung definiter formen als summe von formenquadraten, Math. Ann., 32 (1888), pp. 342–350.

[14] R. HORST AND N. V. THOM, Dc programming: Overview, Journal of Optimization Theory & Applications, 103 (1999), pp. 1–43.

[15] IBM, Ibm ilog cplex optimization studio 12.7, https://www.ibm.com/us-en/marketplace/ibm-ilog-plex.

[16] M. KOJIMA, Sums of squares relaxations of polynomial semidefinite programs, in Mathematical and computing sciences, Tokyo Institute of Technology, Meguro, Tokyo, 2014, pp. 152–8552.

[17] M. KOJIMA, K. FUJISAWA, K. NAKATA, AND M. YAMASHITA, Sdpa-m 7.3.9, http://sdpa.sourceforge.net/.

[18] J. B. LASSEURRE, Global optimization with polynomials and the problem of moments, Siam J Optim, 11 (2001), pp. 796–817.

[19] J. B. LASSEURRE, A semidefinite programming approach to the generalized problem of moments., Mathematical Programming, 112 (2008), pp. 65–92.

[20] J. B. LASSEURRE, Moments and sums of squares for polynomial optimization and related problems, Journal of Global Optimization, 45 (2009), pp. 39–61.

[21] H. A. LETHI AND D. T. PHAM, Dc programming and dca: thirty years of developments, Math. Program., Ser. B, (2018), pp. 1–64.

[22] H. A. LETHI, D. T. PHAM, AND V. HUYNH, Exact penalty and error bounds in dc programming, Journal of Global Optimization, 52 (2012), pp. 509–535.

[23] J. LOFBERG, Yalmip : a toolbox for modeling and optimization in matlab, Optimization, 2004 (2004), pp. 284 – 289.

[24] MathWorks, Matlab documentation, http://www.mathworks.com/help/matlab/.

[25] Maxima, Maxima: a computer algebra system, http://maxima.sourceforge.net.

[26] Mosek, Mosek modeling cookbook, https://www.mosek.com/.

[27] K. G. MURTY AND S. N. KABADI, Some np-complete problems in quadratic and nonlinear programming, Mathematical Programming, 39 (1987), pp. 117–129.

[28] Y. S. NIU, Programmation DC & DCA en Optimisation Combinatoire et Optimisation Polynômale via les Techniques de SDP, Ministère de l’enseignement supérieur et de la recherche, Institut National Des Sciences Appliquées de Rouen, France, 2010.

[29] Y. S. NIU, Improved dc programming approaches for solving quadratic eigenvalue complementarity problem, 2018, https://arxiv.org/abs/1712.00227.

[30] Y. S. NIU, J. J. JUDICE, H. A. LETHI, AND D. T. PHAM, Solving the quadratic eigenvalue complementarity problem by dc programming, Modelling, Computation and Optimization in Information Systems and Management Sciences, Advances in Intelligent Systems and Computing, 359 (2015), pp. 203–214.

[31] Y. S. NIU, H. A. LETHI, D. T. PHAM, AND J. J. JUDICE, Efficient dc programming approaches for the asymmetric eigenvalue complementarity problem, Optimization Methods and Soft-
ware, 28 (2013), pp. 812–829.

[32] P. M. Pardalos and S. A. Vavasis, Open questions in complexity theory for numerical optimization, Springer-Verlag New York, Inc., 1992.

[33] P. A. Parrilo, Semidefinite programming relaxations for semialgebraic problems, Mathematical Programming, 96 (2003), pp. 293–320.

[34] D. T. Pham and H. A. Le Thi, Dc optimization algorithms for solving the trust region subproblem, SIAM J. Optim., 8 (1998), pp. 476–507.

[35] D. T. Pham and H. A. Le Thi, The dc programming and dca revisited with dc models of real world nonconvex optimization problems, Annals of Operations Research, 133 (2005), pp. 23–46.

[36] D. T. Pham and Y. S. Niu, An efficient dc programming approach for portfolio decision with higher moments, Computational Optimization & Applications, 50 (2011), pp. 525–554.

[37] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, Sostools: sum of squares optimization toolbox for matlab, 2002-2005, http://www.mit.edu/~parrilo/sostools.

[38] B. Reznick, Some concrete aspects of hilbert’s 17th problem, Contemporary Mathematics, (1996), pp. 251–272.

[39] R. M. Robinson, Some definite polynomials which are not sums of squares of real polynomials, Selected questions of algebra and logic, (1973), pp. 264–282.

[40] J. F. Sturm, Sedumi 1.3, http://sedumi.ie.lehigh.edu/.

[41] K. C. Toh, M. J. Todd, and R. H. Tutuncu, Sdpt3 - a matlab software package for semidefinite programming, Optimization Methods & Software, 11 (1999), pp. 545–581, http://www.math.nus.edu.sg/~mattohkc/sdpt3.html.

[42] H. Tuy, Convex Analysis and Global Optimization, Springer, 2016.

[43] A. Wächter and L. T. Biegler, on the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming, Mathematical Programming, 106 (2006), pp. 25–57, https://projects.coin-or.org/Ipopt.