Chapter 2
Detecting Sums of Hermitian Squares

2.1 Introduction

The central question of this chapter is how to find out whether a given nc polynomial is a sum of hermitian squares (SOHS). We rely on Sect. 1.3, where we explained basic relations between SOHS polynomials and positive semidefinite Gram matrices. In this chapter we will enclose these results into the Gram matrix method and refine it with the Newton chip method.

2.2 The Gram Matrix Method

Recall from Sect. 1.3 that an nc polynomial $f \in \mathbb{R}\langle \underline{X} \rangle_{2d}$ is SOHS if and only if we can find a positive semidefinite Gram matrix associated with f, i.e., a positive semidefinite matrix G satisfying $\mathbf{W}_d^* G \mathbf{W}_d = f$, where $\mathbf{W}_d$ is the vector of all words of degree $\leq d$. This is a semidefinite feasibility problem in the matrix variable G. The constraints $\langle A_i \,|\, G \rangle = b_i$ are implied by the fact that for each monomial $w \in \mathbf{W}_{2d}$ we have

$$\sum_{\substack{u,v \in \mathbf{W}_d \\ u^* v = w}} G_{u,v} = a_w, \tag{2.1}$$

where a_w is the coefficient of w in f.

Problems like this can be (in theory) solved *exactly* using quantifier elimination [BPR06] as has been suggested in the commutative case by Powers and Wörmann [PW98]. However, this only works for problems of small size, so a *numerical* approach is needed in practice. Thus we turn to numerical methods to solve semidefinite programming problems.

© The Author(s) 2016

S. Burgdorf et al., *Optimization of Polynomials in Non-Commuting Variables*,
SpringerBriefs in Mathematics, DOI 10.1007/978-3-319-33338-0_2

Sums of hermitian squares are symmetric so we consider only $f \in \operatorname{Sym}\mathbb{R}\langle \underline{X}\rangle$. Two symmetric polynomials are equal if and only if all of their "symmetrized coefficients" (i.e., $a_w + a_{w^*}$) coincide, hence Eqs. (2.1) can be rewritten as

$$\sum_{\substack{u,v\in\mathbf{W}_d\\ u^*v=w}} G_{u,v} + \sum_{\substack{u,v\in\mathbf{W}_d\\ v^*u=w^*}} G_{v,u} = a_w + a_{w^*} \quad \forall w \in \mathbf{W}_{2d}, \tag{2.2}$$

or equivalently,

$$\langle A_w \,|\, G\rangle = a_w + a_{w^*} \quad \forall w \in \mathbf{W}_{2d}, \tag{2.3}$$

where A_w is the symmetric matrix defined by

$$(A_w)_{u,v} = \begin{cases} 2; \text{ if } u^*v = w,\ w^* = w, \\ 1; \text{ if } u^*v \in \{w, w^*\},\ w^* \neq w, \\ 0; \text{ otherwise.} \end{cases}$$

Note that in this formulation the constraints obtained from w and w^* are the same so we keep only one of them. As we are interested in an arbitrary positive semidefinite G satisfying constraints (2.3), we can choose the objective function freely. However, in practice one prefers solutions of small rank leading to shorter SOHS decompositions. Hence we minimize the trace, a commonly used heuristic for matrix rank minimization (cf. [RFP10]). Therefore our SDP in primal form is as follows:

$$\begin{aligned} \inf \quad & \langle I \,|\, G\rangle \\ \text{s.t. } & \langle A_w \,|\, G\rangle = a_w + a_{w^*} \ \forall w \in \mathbf{W}_{2d} \\ & G \succeq 0. \end{aligned} \tag{SOHS$_{\text{SDP}}$}$$

Summing up, the Gram matrix method can be presented in Algorithm 2.1.

Algorithm 2.1: The Gram matrix method for finding SOHS decompositions

Input: $f \in \operatorname{Sym}\mathbb{R}\langle \underline{X}\rangle$ with $\deg f \leq 2d$, $f = \sum_{w\in\langle \underline{X}\rangle} a_w w,$, where $a_w \in \mathbb{R}$;

1 $\mathscr{G} = \varnothing$;
2 Construct $\mathbf{W}_d$;
3 Construct data $A_w, \mathbf{b}, C$ corresponding to (SOHS$_{\text{SDP}}$);
4 Solve (SOHS$_{\text{SDP}}$);
5 **if** (SOHS$_{\text{SDP}}$) is not feasible **then**
6 $\quad f \notin \Sigma^2$. **Stop**;
7 **end**
8 Take an optimal solution G and compute the Cholesky decomposition $G = R^*R$;
9 $\mathscr{G} = \{g_i\}$, where g_i denotes the ith component of $R\mathbf{W}_d$;

Output: $\mathscr{G}$;

Remark 2.1. The order of G in (SOHS_{SDP}) is the length of $\mathbf{W}_d$, which is $\sigma = \frac{n^{d+1}-1}{n-1}$, as shown in Remark 1.12. Since $\sigma = \sigma(n,d)$ grows exponentially with the polynomial degree d it easily exceeds the size manageable by the state-of-the-art SDP solvers, which is widely accepted to be of order 1000. This implies, for example, that the above algorithm can only handle nc polynomials in two variables if they are of degree < 10. Therefore it is very important to find an improvement of the Gram matrix method which will be able to work with much larger nc polynomials. This will be done in the rest of the chapter.

Example 2.2. Let

$$f = X^2 - X^{10}Y^{20}X^{11} - X^{11}Y^{20}X^{10} + X^{10}Y^{20}X^{20}Y^{20}X^{10}. \tag{2.4}$$

The order of a Gram matrix G for f is $\sigma(10) = \sigma(2,10) = 2^{41} - 1$ and is too big for today's SDP solvers. Therefore any implementation of Algorithm 2.1 will get stuck. On the other hand, it is easy to see that

$$f = (X - X^{10}Y^{20}X^{10})^*(X - X^{10}Y^{20}X^{10}) \in \Sigma^2.$$

The polynomial f is sparse and an improved SDP for testing whether (sparse) polynomials are sums of hermitian squares will be given below.

The complexity of solving an SDP is also determined by the number of Eq. (2.3), which we denote by m. There are exactly

$$m = \operatorname{card}\{w \in \mathbf{W}_{2d} \mid w^* = w\} + \frac{1}{2}\operatorname{card}\{w \in \mathbf{W}_{2d} \mid w^* \neq w\}$$

such equations in (SOHS_{SDP}). Since $\mathbf{W}_d$ contains all words in $\langle \underline{X} \rangle$ of degree $\leq d$, we have $m > \frac{1}{2}\sigma(2d) = \frac{n^{2d+1}-1}{2(n-1)}$.

For each $w \in \mathbf{W}_{2d}$ there are t different pairs (u_i, v_i) such that $w = u_i^* v_i$, where $t = \deg w + 1$ if $\deg w \leq d$, and $t = 2d + 1 - \deg w$ if $\deg w \geq d+1$. Note that $t \leq d+1$. Therefore the matrices A_i defining constraints (2.3) have order $\sigma(d)$ and every matrix A_i has at most $d+1$ nonzero entries if it corresponds to a symmetric monomial of f, and has at most $2(d+1)$ nonzero entries otherwise. Hence the matrices A_i are sparse. They are also pairwise orthogonal with respect to the standard scalar product on matrices $\langle X \,|\, Y \rangle = \operatorname{tr} X^T Y$, and have disjoint supports, as we now proceed to show:

Theorem 2.3. *Let $\{A_i \mid i = 1, \ldots, m\}$ be the matrices constructed in Step 3 of Algorithm 2.1 [i.e., matrices satisfying (2.3)]. If $(A_i)_{u,v} \neq 0$, then $(A_j)_{u,v} = 0$ for all $j \neq i$. In particular, $\langle A_i \,|\, A_j \rangle = 0$ for $i \neq j$.*

Proof. The equations in the SDP underlying the SOHS decomposition represent the constraints that the monomials in $\mathbf{W}_{2d}$ must have coefficients prescribed by the polynomial f. Let us fix $i \neq j$. The matrices A_i and A_j correspond to some monomials $p_1^* q_1$ and $p_2^* q_2$ ($p_i, q_i \in \mathbf{W}_d$), respectively, and $p_1^* q_1 \neq p_2^* q_2$. If A_i and A_j both have a nonzero entry at position (u,v), then $p_1^* q_1 = u^* v = p_2^* q_2$, a contradiction. ∎

Remark 2.4. Sparsity and orthogonality of the constraints imply that the state-of-the-art SDP solvers can handle about 100,000 such constraints (see, e.g., [MPRW09]), if the order of the matrix variable is about 1000. The boundary point method introduced in [PRW06] and analyzed in [MPRW09] has turned out to perform best for semidefinite programs of this type. It is able to use the *orthogonality* of the matrices A_i (though *not* the disjointness of their supports). In the computationally most expensive steps—solving a linear system—the system matrix becomes diagonal, so solving the system amounts to dividing by the corresponding diagonal entries.

Since $\mathbf{W}_d$ contains all words in $\langle \underline{X} \rangle$ of degree $\leq d$, we have, e.g., for $n = 2$, $d = 10$ that $m = \sigma(20) = \sigma(2,20) = 2,097,150$ and this is clearly out of reach for *all* current SDP solvers. Nevertheless, we show in the sequel that one can replace the vector $\mathbf{W}_d$ in Step 2 of Algorithm 2.1 by a vector $\mathbf{W}$ which is usually much smaller and has at most kd words, where k is the number of symmetric monomials in f and $2d = \deg f$. Hence the order of the matrix variable G and the number of linear constraints m end up being much smaller in general.

2.3 Newton Chip Method

We present a modification of (Step 1 of) the Gram matrix method (Algorithm 2.1) by implementing the appropriate non-commutative analogue of the classical Newton polytope method [Rez78], which we call the *Newton chip method* and present it as Algorithm 2.2.

Definition 2.5. Let us define the *right chip function* $\mathrm{rc} : \langle \underline{X} \rangle \times \mathbb{N}_0 \to \langle \underline{X} \rangle$ by

$$\mathrm{rc}(w_1 \cdots w_n, i) := \begin{cases} w_{n-i+1} w_{n-i+2} \cdots w_n & \text{if } 1 \leq i \leq n; \\ w_1 \cdots w_n & \text{if } i > n; \\ 1 & \text{if } i = 0. \end{cases}$$

Example 2.6. Given the word $w = X_1 X_2 X_1 X_2^2 X_1 \in \langle \underline{X} \rangle$ we have $\mathrm{rc}(w,4) = X_1 X_2^2 X_1$, $\mathrm{rc}(w,6) = w$ and $\mathrm{rc}(w,0) = 1$.

We introduce the Newton chip method, presented as Algorithm 2.2. It substantially reduces the word vector needed in the Gram matrix method.

Theorem 2.7. *Suppose $f \in \mathrm{Sym}\,\mathbb{R}\langle \underline{X} \rangle$. Then $f \in \Sigma^2$ if and only if there exists a positive semidefinite matrix G satisfying*

$$f = \mathbf{W}^* G \mathbf{W},$$

where $\mathbf{W}$ is the output given by the Newton chip method (Algorithm 2.2).

Proof. Suppose $f \in \Sigma^2$. In every SOHS decomposition

$$f = \sum_i g_i^* g_i,$$

only words from $\mathscr{D}$ (constructed in Step 4) are used, i.e., $g_i \in \operatorname{span}\mathscr{D}$ for every i. This follows from the fact that the lowest and highest degree terms cannot cancel (cf. proof of Proposition 1.16). Let $\mathscr{W} := \bigcup_i \mathscr{W}_{g_i}$ be the union of the supports of the g_i. We shall prove that $\mathscr{W} \subseteq W$. For this, let us introduce a partial ordering on $\langle \underline{X} \rangle$:

$$w_1 \preceq w_2 \;\Leftrightarrow\; \exists i \in \mathbb{N}_0 : \operatorname{rc}(w_2, i) = w_1.$$

Note: $w_1 \preceq w_2$ if and only if there is a $v \in \langle \underline{X} \rangle$ with $w_2 = vw_1$.

CLAIM. For every $w \in \mathscr{W}$ there exists $u \in \langle \underline{X} \rangle$: $w \preceq u \preceq u^* u \in \mathscr{W}_f$.

Proof. Clearly, $w^* w$ is a word that appears in the representation of $g_i^* g_i$ which one naturally gets by multiplying out without simplifying, for some i. If $w^* w \notin \mathscr{W}_f$, then there are $w_1, w_2 \in \mathscr{W} \setminus \{w\}$ with $w_1^* w_2 = w^* w$ (appearing with a negative coefficient so as to cancel the $w^* w$ term). Then $w \preceq w_1$ or $w \preceq w_2$, without loss of generality, $w \preceq w_1$. Continuing the same line of reasoning, but starting with $w_1^* w_1$, we eventually arrive at $w_\ell \in \mathscr{W}$ with $w_\ell^* w_\ell \in \mathscr{W}_f$ and $w \preceq w_1 \preceq \cdots \preceq w_\ell$. Thus $w \preceq w_\ell \preceq w_\ell^* w_\ell \in \mathscr{W}_f$, concluding the proof of the claim.

The theorem follows now. Since $u^* u \in \mathscr{W}_f$ and w is a right chip of u we have $w \in \mathbf{W}$. ∎

Algorithm 2.2: The Newton chip method

Input: $f \in \operatorname{Sym}\mathbb{R}\langle \underline{X} \rangle$ with $\deg f \le 2d$, $f = \sum_{w \in \langle \underline{X} \rangle} a_w w$,, where $a_w \in \mathbb{R}$;

Define the *support* of f as $\mathscr{W}_f := \{w \in \langle \underline{X} \rangle \mid a_w \neq 0\}$;

$W := \varnothing$;

Let $m_i := \frac{\operatorname{mindeg}_i f}{2}$, $M_i := \frac{\deg_i f}{2}$, $m := \frac{\operatorname{mindeg} f}{2}$, $M := \frac{\deg f}{2}$;

The set of admissible words is defined as

$$\mathscr{D} := \{w \in \langle \underline{X} \rangle \mid m_i \le \deg_i w \le M_i \text{ for all } i,\ m \le \deg w \le M\};$$

for every $w^* w \in \mathscr{W}_f$ **do**
  for $0 \le i \le \deg w$ **do**
    if $\operatorname{rc}(w, i) \in \mathscr{D}$ **then**
      $W := W \cup \{\operatorname{rc}(w, i)\}$;
    end
  end
end

Sort W in a lexicographic order and transform it into the vector $\mathbf{W}$;

Output: $\mathbf{W}$;

Example 2.8 (Example 2.2 Continued). The polynomial f from Example 2.2 has two hermitian squares: X^2 and $X^{10}Y^{20}X^{20}Y^{20}X^{10}$. The first hermitian square contributes via the Newton chip method only one right chip: X; while the second hermitian square $X^{10}Y^{20}X^{20}Y^{20}X^{10}$ contributes to $\mathbf{W}$ the following words: X, $X^2, \ldots, X^{10}$ as well as $YX^{10}, Y^2X^{10}, \ldots, Y^{20}X^{10}, XY^{20}X^{10}, \ldots, X^{10}Y^{20}X^{10}$.

Applying the Newton chip method to f therefore yields $\mathbf{W}$ which is a vector in the lexicographic order and is equal to

$$\mathbf{W} = \begin{bmatrix} X & X^2 & \cdots & X^{10} & YX^{10} & \cdots & Y^{20}X^{10} & XY^{20}X^{10} & \cdots & X^{10}Y^{20}X^{10} \end{bmatrix}^T$$

of length 40. Problems of this size are easily handled by today's SDP solvers. Nevertheless we provide a further strengthening of our Newton chip method reducing the number of words needed in this example to 2 (see Sect. 2.4).

2.4 Augmented Newton Chip Method

The following simple observation is often crucial to reduce the size of $\mathbf{W}$ returned by the Newton chip method.

Lemma 2.9. *Suppose* $\mathbf{W}$ *is the vector of words returned by the Newton chip method. If there exists a word* $u \in \mathbf{W}$ *such that the constraint in* (SOHS$_{\text{SDP}}$) *corresponding to* u^*u *can be written as*

$$\langle A_{u^*u} \,|\, G \rangle = 0$$

and A_{u^*u} *is a diagonal matrix (i.e.,* $(A_{u^*u})_{u,u} = 2$ *and* A_{u^*u} *is* 0 *elsewhere), then we can eliminate* u *from* $\mathbf{W}$ *and likewise delete this equation from the semidefinite program.*

Proof. Indeed, such a constraint implies that $G_{u,u} = 0$ for the given $u \in \mathbf{W}$, hence the uth row and column of G must be zero, since G is positive semidefinite. So we can decrease the order of (SOHS$_{\text{SDP}}$) by deleting the uth row and column from G and by deleting this constraint. ■

Lemma 2.9 applies if and only if there exists a constraint $\langle A_w \,|\, G \rangle = 0$, where $w = u^*u$ for some $u \in \mathbf{W}$ and $w \neq v^*z$ for all $v, z \in \mathbf{W}$, $v \neq z$. Therefore we augment the Newton chip method (Algorithm 2.2) by new steps, as shown in Algorithm 2.3.

Algorithm 2.3: The Augmented Newton chip method

Input: $f \in \operatorname{Sym}\mathbb{R}\langle \underline{X} \rangle$ with $\deg f \leq 2d$, $f = \sum_{w \in \langle \underline{X} \rangle} a_w w,$, where $a_w \in \mathbb{R}$;

1 Compute $\mathbf{W}$ by the Newton chip method (Algorithm 2.2);
2 **while** *exists* $u \in \mathbf{W}$ *such that* $a_{u^*u} = 0$ *and* $u^*u \neq v^*z$ *for every pair* $v, z \in \mathbf{W}$, $v \neq z$ **do**
3 | delete u from $\mathbf{W}$;
4 **end**

Output: $\mathbf{W}$;

Note that in Step 2 there might exist some word $u \in \mathbf{W}$ which does not satisfy the condition initially but after deleting another u' from $\mathbf{W}$ it does. We demonstrate Algorithm 2.3 in the following example:

Example 2.10 (Example 2.2 Continued). By applying the Augmented Newton chip method to f from (2.4) we reduce the vector $\mathbf{W}$ significantly. Note that after Step 1, $\mathbf{W}$ also contains the words X^8, X^9, X^{10}. Although X^{18} does not appear in f, we cannot delete X^9 from $\mathbf{W}$ immediately since $X^{18} = (X^9)^* X^9 = (X^8)^* X^{10}$. But we can delete X^{10} since X^{20} does not appear in f and $(X^{10})^* X^{10}$ is the unique decomposition of X^{20} inside $\mathbf{W}$. After deleting X^{10} from $\mathbf{W}$ we realize that $(X^9)^* X^9$ becomes the unique decomposition of X^{18}, hence we can eliminate X^9 too. Eventually the Augmented Newton chip method returns

$$\mathbf{W} = \begin{bmatrix} X & X^{10} Y^{20} X^{10} \end{bmatrix}^T,$$

which is exactly the minimum vector needed for the SOHS decomposition of f.

2.5 Implementation

2.5.1 On the Gram Matrix Method

The Gram matrix method (Algorithm 2.1) consists of two main parts: (1) constructing the matrices corresponding to (SOHS_{SDP})—Step 3 and (2) solving the constructed SDP in Step 4. Step 3 is straightforward, running the Augmented Newton chip method (Algorithm 2.3) gives the desired vector of relevant words. There are no numerical problems, no convergence issues, Algorithm 2.3 always terminates with the desired vector $\mathbf{W}$.

The second main part is more subtle. Solving an instance of SDP in practice always involves algorithms that are highly numerical: algorithms to compute spectral decompositions, solutions of systems of linear equations, inverses of matrices, etc. Methods for solving SDP, especially interior point methods [dK02, Ter96, WSV00], but also some first order methods [MPRW09, PRW06], typically assume strictly feasible solutions on the primal and the dual side, which imply the strong duality property and the attainability of optimums on both sides. Moreover, this assumption also guarantees that most of the methods will converge to a primal–dual ε-optimal solution; see also Sect. 1.13.

As the following example demonstrates, the Slater condition is not necessarily satisfied on the primal side in our class of (SOHS_{SDP}) problems.

Example 2.11. Let $f = (XY + X^2)^*(XY + X^2)$. It is homogeneous, and the Augmented Newton chip method gives

$$\mathbf{W} = \begin{bmatrix} X^2 \\ XY \end{bmatrix}.$$

There exists a unique symmetric Gram matrix

$$G = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

for f such that $f = \mathbf{W}^* G \mathbf{W}$. Clearly G, a rank 1 matrix, is the only feasible solution of (SOHS_{SDP}), hence the corresponding SDP has no strictly feasible solution on the primal side.

If we take the objective function in our primal SDP (SOHS_{SDP}) to be equal to $\langle I \,|\, G \rangle$, then the pair $y = 0$, $Z = I$ is always strictly feasible for the dual problem of (SOHS_{SDP}) and thus we *do* have the strong duality property.

Hence, when the given nc polynomial is in Σ^2, the corresponding semidefinite program (SOHS_{SDP}) is feasible, and the optimal value is attained. If there is no strictly feasible solution, then numerical difficulties might arise but state-of-the-art SDP solvers such as SeDuMi [Stu99], SDPT3 [TTT99], SDPA [YFK03], or MOSEK [ApS15] are able to overcome them in most of the instances. When the given nc polynomial is not in Σ^2, then the semidefinite problem (SOHS_{SDP}) is infeasible and this might cause numerical problems as well. However, state-of-the-art SDP solvers are generally robust and can reliably detect infeasibility for most practical problems; for more details see [dKRT98, PT09].

2.5.2 *Software Package* `NCSOStools`

The software package `NCSOStools` [CKP11] was developed to help researchers working in the area of non-commutative polynomials. `NCSOStools` [CKP11] is an open source Matlab toolbox for solving SOHS related problems using semidefinite programming. It also implements symbolic computation with non-commuting variables in Matlab.

There is a small overlap in features with Helton's `NCAlgebra` package for Mathematica [HMdOS15]. However, `NCSOStools` [CKP11] performs basic manipulations with non-commuting variables and is mainly oriented to detect several variants of constrained and unconstrained positivity of nc polynomials, while `NCAlgebra` is a fully fledged add-on for symbolic computation with polynomials, matrices, and rational functions in non-commuting variables.

When we started writing `NCSOStools` we decided to use Matlab as a main framework since we solve the underlying SDP instances by existing open source solvers like SeDuMi [Stu99], SDPT3 [TTT99], or SDPA [YFK03] and these solvers can be very easily run within Matlab.

Readers interested in solving sums of squares problems for commuting polynomials are referred to one of the many great existing packages, such as SOSTOOLS [PPSP05], SparsePOP [WKK+09], GloptiPoly [HLL09], or YALMIP [Löf04].

Detecting sums of hermitian squares by the Gram matrix method and using the (Augmented) Newton chip method can be done within `NCSOStools` by calling `NCsos`.

Example 2.12 (Example 2.11 Continued). We declare the polynomial f that we started considering in Example 2.11 within NCSOStools by

```
NCvars x y
>> f=(x*y+x^2)'*(x*y+x^2)
```

By calling

```
>> [IsSohs,Gr,W,sohs,g,SDP_data,L] = NCsos(f)
```

we obtain that f is SOHS (IsSohs=1), the vector given by the Augmented Newton chip methods (W) and the corresponding Gram matrix Gr:

```
W =
    'x*x'
    'x*y'

Gr =
    1.0000    1.0000
    1.0000    1.0000
```

Likewise we obtain the SOHS decomposition of f

```
sohs =
       x^2+x*y
   2.2e-07*x*y
```

which means that the SOHS decomposition for f is

$$f = (X^2+XY)^*(X^2+XY) + (2.2\cdot 10^{-7}XY)^*(2.2\cdot 10^{-7}XY).$$

This is ε correct for $\varepsilon = 10^{-13}$, i.e., if we leave cut off all monomials with coefficients less than 10^{-13} we obtain f. We can control precision using the parameter pars.precision. All monomials in sohs having coefficient smaller than pars.precision are ignored. Therefore by running

```
>> pars.precision=1e-6;
>> [IsSohs,Gr,W,sohs,g,SDP_data,L] = NCsos(f,pars);
```

we obtain the exact value for a SOHS decomposition of f, i.e., f is exactly a SOHS of elements from sohs.

The data describing the semidefinite program (SOHS_{SDP}) is given in SDP_data while the optimal matrix for the dual problem to (SOHS_{SDP}) is given in L. In g we return sum of squares of entries from sohs with monomials having coefficient larger than 10^{-8} which is an internal parameter.

References

[BPR06] Basu, S., Pollack, R., Roy, M.-F.: Algorithms in Real Algebraic Geometry. Algorithms and Computation in Mathematics, vol. 10, 2nd edn. Springer, Berlin (2006)

[CKP11] Cafuta, K., Klep, I., Povh, J.: NCSOStools: a computer algebra system for symbolic and numerical computation with noncommutative polynomials. Optim. Methods Softw. **26**(3), 363–380 (2011). Available from http://ncsostools.fis.unm.si/

[dK02] de Klerk, E.: Aspects of Semidefinite Programming. Applied Optimization, vol. 65. Kluwer Academic, Dordrecht (2002)

[dKRT98] de Klerk, E., Roos, C., Terlaky, T.: Infeasible-start semidefinite programming algorithms via self-dual embeddings. In: Topics in Semidefinite and Interior-Point Methods (Toronto, ON, 1996). Fields Institute Communications, vol. 18, pp. 215–236. American Mathematical Society, Providence (1998)

[HMdOS15] Helton, J.W., Miller, R.L., de Oliveira, M.C., Stankus, M.: NCAlgebra: a mathematica package for doing non commuting algebra. Available from http://www.math.ucsd.edu/~ncalg/ (2015)

[HLL09] Henrion, D., Lasserre, J.B., Löfberg, J.: GloptiPoly 3: moments, optimization and semidefinite programming. Optim. Methods Softw. **24**(4–5), 761–779 (2009). Available from http://www.laas.fr/~henrion/software/gloptipoly3/

[Löf04] Löfberg, J.: YALMIP: a toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference, Taipei. Available from http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php (2004)

[MPRW09] Malick, J., Povh, J., Rendl, F., Wiegele, A.: Regularization methods for semidefinite programming. SIAM J. Optim. **20**(1), 336–356 (2009)

[ApS15] MOSEK ApS: The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 28) (2015)

[PT09] Pólik, I., Terlaky, T.: New stopping criteria for detecting infeasibility in conic optimization. Optim. Lett. **3**(2), 187–198 (2009)

[PRW06] Povh, J., Rendl, F., Wiegele, A.: A boundary point method to solve semidefinite programs. Computing **78**, 277–286 (2006)

[PW98] Powers, V., Wörmann, T.: An algorithm for sums of squares of real polynomials. J. Pure Appl. Algebra **127**(1), 99–104 (1998)

[PPSP05] Prajna, S., Papachristodoulou, A., Seiler, P., Parrilo, P.A.: SOSTOOLS and its control applications. In: Positive Polynomials in Control. Lecture Notes in Control and Information Science, vol. 312, pp. 273–292. Springer, Berlin (2005)

[RFP10] Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev. **52**(3), 471–501 (2010)

[Rez78] Reznick, B.: Extremal PSD forms with few terms. Duke Math. J. **45**(2), 363–374 (1978)

[Stu99] Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11/12**(1–4), 625–653 (1999). Available from http://sedumi.ie.lehigh.edu/

[Ter96] Terlaky, T. (ed.): Interior Point Methods of Mathematical Programming. Applied Optimization, vol. 5. Kluwer Academic, Dordrecht (1996)

[TTT99] Toh, K.C., Todd, M.J., Tütüncü, R.: SDPT3–a MATLAB software package for semidefinite programming, version 1.3. Optim. Methods Softw. **11/12**(1–4), 545–581 (1999). Available from http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

[WKK+09] Waki, H., Kim, S., Kojima, M., Muramatsu, M., Sugimoto, H.: Algorithm 883: sparsePOP—a sparse semidefinite programming relaxation of polynomial optimization problems. ACM Trans. Math. Softw. **35**(2), Art. 15, 13 (2009)

[WSV00] Wolkowicz, H., Saigal, R., Vandenberghe, L.: Handbook of Semidefinite Programming. Kluwer, Boston (2000)

[YFK03] Yamashita, M., Fujisawa, K., Kojima, M.: Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0). Optim. Methods Softw. **18**(4), 491–505 (2003). Available from http://sdpa.sourceforge.net/

http://www.springer.com/978-3-319-33336-6

Optimization of Polynomials in Non-Commuting Variables
Burgdorf, S.; Klep, I.; Povh, J.
2016, XV, 104 p. 2 illus. in color., Softcover
ISBN: 978-3-319-33336-6