SUMS OF HERMITIAN SQUARES DECOMPOSITION OF NON-COMMUTATIVE POLYNOMIALS IN NON-SYMMETRIC VARIABLES USING NCSOSTOOLS

KRISTIJAN CAFUTA[1]

Abstract. Numerous applied problems contain matrices as variables, and the formulas therefore involve polynomials in matrices. To handle such polynomials it is necessary to study non-commutative polynomials. In this paper we will present an algorithm and its implementation in the free Matlab package `NCSOStools` using semidefinite programming to check whether a given non-commutative polynomial in non-symmetric variables can be written as a sum of Hermitian squares.

1. Introduction

Optimization problems that involve positivity of polynomials in commuting variables, which is studied in classical real algebraic geometry, can be found in many areas, including operations research [Sho91, Nie09], probability and mathematical finance [Las09]. Non-commutative analogue of classical real algebraic geometry is free real algebraic geometry which studies positivity of polynomials in freely non-commuting variables. Such problems can have matrices as variables, and the formulas can involve polynomials in matrices. Free real algebraic geometry offers numerous applications as well: applications to control theory and system engineering [dHMP08], to quantum physics [PNA10], to mathematical physics and operator algebras [KS08a, KS08b], to investigation of PDEs and eigenvalues of polynomial partial differential operators [Cim10] to name just a few.

At VOCAL 2008 it was presented how to find a sum of Hermitian squares (SOHS) decomposition of a non-commutative polynomial (in symmetric variables with Real coefficients) using semidefinite programming, for which purpose we have developed a freely available open source Matlab toolbox called NCSOStools. This was the beginning of a series of publications on various problems of polynomial optimization problems with non-commuting *symmetric* variables, including trace optimization and constrained optimization, for example [KP10, CKP10, CKP11, CKP12, BCKP13a, BCKP13b, KP16]. All these algorithms have also been implemented in NCSOStools. In this paper we will set foundations for the generalization by using semidefinite programming,

Date: August 22, 2017.

2000 *Mathematics Subject Classification.* Primary 13J30, 90C22; Secondary 08B20, 11E25, 90C90.

Key words and phrases. noncommutative polynomial, sum of Hermitian squares, semidefinite programming, Matlab toolbox, NCSOStools.

[1]The author acknowledge the financial support from the Slovenian Research Agency (research core funding No. P1-0222 and project J1-8132).

which was presented at VOCAL 2016. We will replace symmetric variables with *non-symmetric* variables and therefore extend the theory of SOHS decompositions of non-commutative polynomials on symmetric matrices to non-symmetric matrices.

1.1. **Notation: NS polynomials.** We denote the sets of natural and real numbers with $\mathbb{N} := \{1, 2, \dots\}$ and $\mathbb{R}$. For a fixed $n \in \mathbb{N}$, let $\langle \underline{X}, \underline{X}^* \rangle$ consist of *words* in the $2n$ non-commuting letters $X_1, \dots, X_n, X_1^*, \dots, X_n^*$ (including the empty word denoted by 1), i.e., $\langle \underline{X}, \underline{X}^* \rangle$ is the monoid freely generated by letters $\underline{X} = (X_1, \dots, X_n)$ and $\underline{X}^* = (X_1^*, \dots, X_n^*)$. The set of all words from the monoid $\langle \underline{X}, \underline{X}^* \rangle$ length at most d is denoted by $\langle \underline{X}, \underline{X}^* \rangle_d$.

We write $\mathbb{R}\langle \underline{X}, \underline{X}^* \rangle = \mathbb{R}\langle X_1, \dots, X_n, X_1^*, \dots, X_n^* \rangle$ for the algebra of real polynomials in non-symmetric non-commuting variables $\underline{X} = (X_1, \dots, X_n)$ and $\underline{X}^* = (X_1^*, \dots, X_n^*)$. The elements of algebra $\mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ are linear combinations of words in the $2n$ letters $\underline{X}, \underline{X}^*$. They are called *NS polynomials*. The *degree* of f is the length of the longest word in an NS polynomial $f \in \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ and is denoted by $\deg f$.

A *monomial* is an element of the form aw where $0 \neq a \in \mathbb{R}$ and $w \in \langle \underline{X}, \underline{X}^* \rangle$ and a is its *coefficient*. Therefore words are monomials with coefficient 1.

We equip algebra $\mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ with the *involution* $f \mapsto f^*$, fixing $\mathbb{R}$ point-wise, sending $X_i \mapsto X_i^*$, $X_j^* \mapsto X_j$ and reversing words. Recall that an involution has the properties $(f + g)^* = f^* + g^*$, $(fg)^* = g^* f^*$ and $f^{**} = f$ for all NS polynomials $f, g \in \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$.

Example 1.1. Let $f = X_1 X_3 - 4(X_2^*)^2 X_3$. Then

$$\deg f = 3 \quad \text{and} \quad f^* = X_3^* X_1^* - 4 X_3^* X_2^2.$$

Therefore $\mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ is the $*$-algebra freely generated by $2n$ non-symmetric letters. The involution $*$ extends naturally to matrices over algebra $\mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$. For instance, if $V = (v_i)$ is a (column) vector of NS polynomials $v_i \in \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$, then V^* is the row vector with components v_i^*. We shall also use V^t to denote the row vector with components v_i.

The set of all *symmetric elements* in algebra $\mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ is denoted by $\operatorname{Sym} \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$, i.e.,

$$\operatorname{Sym} \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle = \{ f \in \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle \mid f = f^* \}.$$

1.2. **Semidefinite programming.** Semidefinite programming (SDP) is a generalization of linear programming, where nonnegativity constraints on real variables in linear programming are replaced by semidefiniteness constraints on matrix variables. It is a subfield of convex optimization dealing with the optimization of a linear objective function over the intersection of an affine subspace with the cone of positive semidefinite matrices. More precisely, given $s \times s$ self-adjoint matrices C, $A_1, \dots, A_m$ of the same size over $\mathbb{R}$ and a vector $b \in \mathbb{R}^m$, we formulate a *semidefinite program in standard primal form* as follows:

$$\text{(PSDP)} \qquad \begin{array}{rrcl} \inf & \langle C, G \rangle & & \\ \text{s.t.} & \langle A_i, G \rangle & = & b_i, \quad i = 1, \dots, m \\ & G & \succeq & 0. \end{array}$$

Here $\langle \cdot, \cdot \rangle$ stands for the standard inner product of matrices: $\langle A, B\rangle = \operatorname{tr}(B^*A)$, where tr denotes the trace, and $G \succeq 0$ means that G is positive semidefinite.

Many problems in control theory, system identification and signal processing can be formulated using SDPs [BGFB94, Par00, AL12]. Combinatorial optimization problems can often be modeled or approximated by SDPs [Goe97]. SDP plays an important role in real algebraic geometry, where it is used e.g. for finding sums of squares decompositions of polynomials or approximating the moment problem [Las01, Mar08, Lau09, Las09, HN12, Nie14].

The complexity of solving semidefinite programs is mainly determined by the order s of matrix variable G and the number of linear constraints m. Recently the applicability of semidefinite programming was spurred by the development of practically efficient methods to obtain optimal solutions. If the problem is of medium size (i.e., $s \leq 1000$ and $m \leq 10\ 000$), these packages are based on interior point methods, while packages for larger semidefinite programs use some variant of the first order methods (see [Mit] for a comprehensive list of state of the art SDP solvers). Given $\varepsilon > 0$, the interior point methods can find an ε-optimal solution with polynomially many iterations, where each iteration takes polynomially many real number operations, provided that (PSDP) and its dual both have non-empty interiors of feasible sets and we have good initial points. The variables appearing in these polynomial bounds are s, m and $\log \varepsilon$ (cf. [WSV00, Chapter 10.4.4]). Nevertheless, once $s \geq 3000$ or $m \geq 250\ 000$, the problem must share some special property, otherwise state of the art solvers will fail to solve it for complexity reasons. One way of reducing the size of an SDP is by using symmetries, cf. [BGSV12, GP04]. An alternative is to block diagonalize the constraint matrices A_i from (PSDP), i.e., study the matrix algebra $\mathcal{A}$ generated by $A_1, \ldots, A_m$ [Caf13].

2. Sums of Hermitian squares of NS polynomials

2.1. Positive semidefinite NS polynomials.

Motivation for the next definition is the fact that a symmetric matrix $A \in \mathbb{R}^{s\times s}$ is positive semidefinite if and only if it is of the form B^tB for some $B \in \mathbb{R}^{s\times s}$. Motivated by this, the following section introduces the notion of *sum of Hermitian squares* (SOHS) and explains its relation to semidefinite programming.

Definition 2.1. Non-commutative polynomial in non-symmetric variables of the form g^*g, where $g \in \mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$, is called a *Hermitian square* and the set of all sums of Hermitian squares is denoted by

$$\Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle = \Big\{ \sum_i g_i^* g_i \colon g_i \in \mathbb{R}\langle \underline{X}, \underline{X}^*\rangle \Big\} \subsetneq \operatorname{Sym}\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle.$$

An NS polynomial $f \in \mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$ is SOHS if it belongs to $\Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$. Clearly, $\Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle \subsetneq \operatorname{Sym}\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$.

Example 2.2.

$$X_1X_2 \notin \operatorname{Sym}\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle, \quad -X_1X_1^* \in \operatorname{Sym}\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle \setminus \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle,$$

$$\begin{aligned}(X_1^*)^2X_2 + 2X_1^*X_1 + X_2^*X_1X_1^*X_2 + X_2^*X_1^2 \\ = (X_1 + X_1^*X_2)^*(X_1 + X_1^*X_2) + X_1^*X_1 \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle.\end{aligned}$$

If NS polynomial $f \in \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ is SOHS and we substitute (not necessarily symmetric) matrices $A_1, \dots, A_n$ of the same size for the variables $\underline{X}$, then the resulting matrix $f(A_1, \dots, A_n, A_1^*, \dots, A_n^*)$ is positive semidefinite. Helton [Hel02] and McCullough [McC01] independently proved some kind of the converse of the above observation:

Theorem 2.3 (Helton-McCullough SOHS theorem)**.** *Let $f \in \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ and $f(A_1, \dots, A_n, A_1^*, \dots, A_n^*) \succeq 0$ for all n-tuples of matrices $\underline{A} = (A_1, \dots, A_n)$ of the same size $k \times k$ for any k. Then $f \in \Sigma^2 \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$.*

We refer the reader to [MP05] for an illustrative proof. Therefore we can say that $f \in \Sigma^2 \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ is a positive semidefinite NS polynomial.

The following proposition (cf. [Hel02, §2.2] or [MP05, Theorem 2.1]; see also [BKP16]) is the non-commutative version of the classical result due to Choi, Lam and Reznick ([CLR95, §2]; see also [Par03, PW98]).

Proposition 2.4. *Let $f \in \operatorname{Sym} \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ be of degree $\leq 2d$. Then $f \in \Sigma^2 \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ if and only if there exists a positive semidefinite (PSD) matrix G satisfying*

$$f = W_d^* G W_d = \sum_{i,j} G_{i,j} (W_d)_i^* (W_d)_j, \tag{1}$$

where W_d is a vector consisting of all words in $\langle \underline{X}, \underline{X}^* \rangle_d$.

The matrix G is called *a Gram matrix* for NS polynomial f.

Remark 2.5. Note that for a positive semidefinite matrix $G \in \mathbb{R}^{N \times N}$ of rank r satisfying (1) we can write $G = \sum_{i=1}^r G_i G_i^t$ for $G_i \in \mathbb{R}^{N \times 1}$ and defining $g_i := G_i^t W_d \in \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle_d$ yields $f = \sum_{i=1}^r g_i^* g_i \in \Sigma^2 \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$.

Remark 2.6. If we label columns and rows of a Gram matrix G for NS polynomial f with words from vector W_d, we can see, that for every product of words $w \in \{p^* q \mid p, q \in W_d\}$ the following must be true:

$$\sum_{\substack{p,q \in W_d \\ p^* q = w}} G_{p,q} = a_w, \tag{2}$$

where a_w is the coefficient of the word w in NS polynomial f ($a_w = 0$ if the word w does not appear in f).

Let us demonstrate the Proposition 2.4 with an example.

Example 2.7. Let

$$f = 1 + 2X_1^* X_2 X_2^* X_1 - X_1^* X_2^2 - X_2^* - (X_2^*)^2 X_1 + X_2^* X_2 - X_2 + X_2 X_2^*$$

and let V be the subvector $\begin{bmatrix} 1 & X_2 & X_2^* & X_2^* X_1 \end{bmatrix}^t$ of vector W_2. Then the Gram matrix for NS polynomial f with respect to the vector V is given by

$$G(a) := \begin{bmatrix} 1 & -a-1 & a & 0 \\ -a-1 & 1 & 0 & -1 \\ a & 0 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{bmatrix}.$$

That means $f = V^*G(a)V$ for all a. For SOHS decomposition we are looking for $a \in \mathbb{R}$ (which in general is not unique), such that $G(a)$ is a PSD matrix. The characteristic polynomial of the matrix $G(a)$ is

$$\lambda^4 - 5\lambda^3 + \lambda^2(-2a^2 - 2a + 7) + \lambda(6a^2 + 6a - 2) - 3a^2 - 4a - 1.$$

From the fact that matrix A is positive semidefinite if and only if coefficients in its characteristic polynomial alternate stricly in sign [HJ85, Cor. 7.2.4], it follows that the matrix $G(a)$ is positive semidefinite if and only if $-1 \leq a \leq -\frac{1}{3}$. If we choose $a = -\frac{2}{5}$, we can see $G\left(-\frac{2}{5}\right) = C^t_{-\frac{2}{5}} C_{-\frac{2}{5}}$ for

$$C_{-\frac{2}{5}} = \begin{bmatrix} 1 & -\frac{3}{5} & -\frac{2}{5} & 0 \\ 0 & \frac{4}{5} & -\frac{3}{10} & -\frac{5}{4} \\ 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{4} \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}.$$

From

$$C_{-\frac{2}{5}} V = \begin{bmatrix} 1 - \frac{3}{5}X_2 - \frac{2}{5}X_2^* & \frac{4}{5}X_2 - \frac{3}{10}X_2^* - \frac{5}{4}X_2^*X_1 & \frac{\sqrt{3}}{2}X_2^* - \frac{\sqrt{3}}{4}X_2^*X_1 & \frac{1}{2}X_2^*X_1 \end{bmatrix}^t$$

it follows that

$$\begin{aligned} f = &\left(1 - \frac{3}{5}X_2 - \frac{2}{5}X_2^*\right)^* \left(1 - \frac{3}{5}X_2 - \frac{2}{5}X_2^*\right) \\ &+ \left(\frac{4}{5}X_2 - \frac{3}{10}X_2^* - \frac{5}{4}X_2^*X_1\right)^* \left(\frac{4}{5}X_2 - \frac{3}{10}X_2^* - \frac{5}{4}X_2^*X_1\right) \\ &+ \left(\frac{\sqrt{3}}{2}X_2^* - \frac{\sqrt{3}}{4}X_2^*X_1\right)^* \left(\frac{\sqrt{3}}{2}X_2^* - \frac{\sqrt{3}}{4}X_2^*X_1\right) \\ &+ \left(\frac{1}{2}X_2^*X_1\right)^* \left(\frac{1}{2}X_2^*X_1\right) \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle. \end{aligned}$$

This is not the only SOHS decomposition of NS polynomial f. Let us also look at $a = -1$. Matrix $G(-1)$ has rank 3 (in contrast to the matrix $G\left(-\frac{2}{5}\right)$ which has rank 4). It follows that we get a shorter SOHS decomposition in this case. We can see that (for example) $G(-1) = C^t_{-1}C_{-1}$ for

$$C_{-1} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

From

$$C_{-1}V = \begin{bmatrix} 1 - X_2^* & X_2 - X_2^*X_1 & X_2^*X_1 \end{bmatrix}^t$$

it follows that

$$f = (1-X_2^*)^*(1-X_2^*) + (X_2 - X_2^*X_1)^*(X_2 - X_2^*X_1) + (X_2^*X_1)^*(X_2^*X_1) \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle.$$

At the end of this example let us note that because of the non-uniqueness of a decomposition $G(-1) = C^t_{-1}C_{-1}$ we could also take

$$C'_{-1} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & -\sqrt{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 & 0 \end{bmatrix},$$

which would again yield a different sum of (three) Hermitian squares.

2.2. **Sums of Hermitian squares and SDP.** In this subsection we present a basic algorithm (The Gram matrix method) for checking whether a given NS polynomial $f \in \operatorname{Sym}\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$ can be written as a sum of Hermitian squares. Following Proposition 2.4 we must determine whether there exists a positive semidefinite matrix G such that $f = W_d^* G W_d$. This is a special case of a semidefinite feasibility problem (PSDP) in matrix variable G, where the constraints $\langle A_i, G\rangle = b_i$ follow from the equation (2). Every NS polynomial $f \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$ is obviously symmetric and therefore $a_w = a_{w^*}$ for all words w. Consequently equations (2) can be rewritten as

$$\sum_{\substack{u,v\in W_d\\ u^*v=w}} G_{u,v} + \sum_{\substack{u,v\in W_d\\ u^*v=w^*}} G_{u,v} = a_w + a_{w^*} \quad \text{for all } w \in U_{2d}, \tag{3}$$

where U_{2d} stands for the subset of W_{2d}, where we take only one word from every pair of words (w, w^*). In other words

$$\langle A_w,\, G\rangle = a_w + a_{w^*} \quad \text{for all } w \in U_{2d}, \tag{4}$$

where A_w is the symmetric matrix defined by

$$(A_w)_{u,v} = \begin{cases} 2; & \text{if } u^*v \in \{w, w^*\},\ w^* = w, \\ 1; & \text{if } u^*v \in \{w, w^*\},\ w^* \neq w, \\ 0; & \text{otherwise.} \end{cases} \tag{5}$$

Using this notation our semidefinite program (PSDP) transforms to:

$$(\mathrm{SOHS_{SDP}}) \qquad \begin{array}{lrcl} \inf & \langle I, G\rangle & & \\ \text{s.t.} & \langle A_w, G\rangle & = & a_w + a_{w^*} \quad \text{for all } w \in U_{2d}, \\ & G & \succeq & 0, \end{array}$$

where U_{2d} stands for the subset of W_{2d}, where we take only one word from every pair of words (w, w^*).

As we are interested in an arbitrary positive semidefinite matrix $G = [G_{u,v}]_{u,v\in W}$ satisfying the constraints (4), we can choose the objective function freely. In practice one sometimes prefers solutions of small rank because this leads to shorter SOHS decompositions. Hence we minimize the trace, a commonly used heuristic for matrix rank minimization [RFP10] and therefore we choose $C = I$. On the other hand, one sometimes prefers solutions of a higher rank (for example when trying to compute a rational exact Gram matrix from numerical solution [CKP15]) so we choose $C = 0$ because under a strict feasibility assumption the interior point methods yield solutions in the relative interior of the optimal face, which is in our case the whole feasibility set. If strict complementarity is additionally provided, the interior point methods lead to the analytic center of the feasibility set [HdKR02].

Example 2.8. Let us return to Example 2.7:

$$f = 1 + 2X_1^*X_2X_2^*X_1 - X_1^*X_2^2 - X_2^* - (X_2^*)^2X_1 + X_2^*X_2 - X_2 + X_2X_2^*$$

with $V = \begin{bmatrix}1 & X_2 & X_2^* & X_2^* X_1\end{bmatrix}^t$ and look for some matrices (5) and constraints (4).

$$A_{X_2} = A_{X_2^*} = \begin{bmatrix} 0&1&1&0\\ 1&0&0&0\\ 1&0&0&0\\ 0&0&0&0 \end{bmatrix}, \quad A_{X_2X_2^*} = \begin{bmatrix} 0&0&0&0\\ 0&0&0&0\\ 0&0&2&0\\ 0&0&0&0 \end{bmatrix}.$$

Corresponding linear constraints from (4) are:

$$\begin{aligned} G_{1,X_2} + G_{1,X_2^*} + G_{X_2,1} + G_{X_2^*,1} &= \langle A_{X_2}, G\rangle \\ &= a_{X_2} + a_{X_2^*} = -2, \\ 2G_{X_2^*,X_2} = \langle A_{X_2X_2^*}, G\rangle &= 2a_{X_2X_2^*} = 2. \end{aligned}$$

INPUT: $f = \sum_{w\in\langle \underline{X},\underline{X}^*\rangle} a_w w$ *where* $f \in \operatorname{Sym}\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$ *and* $\deg f \le 2d$.
STEP 1: *Construct* W_d.
STEP 2: *Construct data* A_w, b, C *corresponding to the* (SOHS$_{\mathrm{SDP}}$).
STEP 3: *Solve the* (SOHS$_{\mathrm{SDP}}$) *to obtain* G.
If (SOHS$_{\mathrm{SDP}}$) *is not feasible then*
. $f \notin \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$; *STOP.*
end
STEP 4: *Compute a decomposition* $G = R^t R$.
OUTPUT: *SOHS decomposition:* $f = \sum_i g_i^* g_i$, *where* g_i *denotes the* i*-th component of* RW_d.

Algorithm 1: The Gram matrix method for finding SOHS decompositions

Remark 2.9. Let us look for a moment at the complexity of Algorithm 1. Let $f \in \mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$, where $\underline{X} = (X_1, \ldots, X_n)$. If $\deg f = 2d$, then W_d has length

$$N(n,d) := \sum_{k=0}^{d} (2n)^k = \frac{(2n)^{d+1} - 1}{2n - 1}.$$

This easily exceeds widely accepted manageable size by the state of the art SDP solvers, which is of order 1000. For example, Algorithm 1 can only handle NS polynomials in two variables if they are of degree < 5. So some reduction in the vector of words is needed. In the next section we will introduce Newton NS chip method, which replaces vector W_d in Algorithm 1 by a (usually much smaller) vector W with at most $\frac{kd}{2}$ words, where k is the number of symmetric monomials in NS polynomial f of degree $2d$.

Example 2.10. Let

$$f = X^*X - X^*X^5Y^{10}(X^*)^5 - X^5(Y^*)^{10}(X^*)^5X + X^5(Y^*)^{10}(X^*)^5X^5Y^{10}(X^*)^5,$$

where $X = X_1$ and $Y = X_2$. The size of vector of words W_d is $\frac{4^{21}-1}{3}$, which is too large for today's SDP solvers. But, it is easy to see that

$$f = \big(X - X^5Y^{10}(X^*)^5\big)^*\big(X - X^5Y^{10}(X^*)^5\big) \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle,$$

hence it is enough to use $W = [X \quad X^5Y^{10}(X^*)^5]$.

3. Newton NS chip method

In the following section we present a modification of Algorithm 1 (replacing vector W_d with a smaller vector W) by implementing non-commutative non-symmetric analogue of the classical Newton polytope method [Rez78] and non-symmetric analogue of the symmetric non-commutative Newton chip method [KP10].

Similar to the degree $\deg f$ (the length of the longest word in f) we can define min-degree denoted by $\operatorname{mindeg} f$ (the length of the shortest word in f). We can also consider the degree of f in some specific variable X_i denoted by $\deg_i f$ where we count repetitions of X_i and X_i^* and similarly we define $\operatorname{mindeg}_i f$. For example $\deg_1(X_1^2 X_2^3 X_1^*) = 3$.

Lemma 3.1. *We can replace vector W_d in Algorithm 1 with a (smaller) vector*

$$V := \left\{ w \in \langle \underline{X}, \underline{X}^* \rangle \;\middle|\; \frac{\operatorname{mindeg} f}{2} \leq \deg w \leq \frac{\deg f}{2}, \frac{\operatorname{mindeg}_i f}{2} \leq \deg_i w \leq \frac{\deg_i f}{2} \text{ for all } i \right\}.$$

Proof. Let $f = \sum_j g_j^* g_j$ be a SOHS decomposition. Since the lowest and the highest degree terms in this decomposition cannot cancel, it follows

$$\operatorname{mindeg} g_j \geq \frac{\operatorname{mindeg} f}{2} \quad \text{and} \quad \deg g_j \leq \frac{\deg f}{2} \quad \text{for all } j$$

so $\frac{\operatorname{mindeg} f}{2} \leq \deg w \leq \frac{\deg f}{2}$ can hold for needed words w.
Similarly, if we look at the degree in variables, it follows

$$\operatorname{mindeg}_i g_j \geq \frac{\operatorname{mindeg}_i f}{2} \quad \text{and} \quad \deg_i g_j \leq \frac{\deg_i f}{2} \quad \text{for all } i, j$$

so $\frac{\operatorname{mindeg}_i f}{2} \leq \deg_i w \leq \frac{\deg_i f}{2}$ can hold for all i and needed words w. ■

Below we will present a further reduction of the vector of needed words. As we will see the main role will be played by the above V and monomials in f of the form $a_w w^* w$.

Before we proceed, let us remind the reader of another possibility to perhaps reduce the size of the needed word vector. We will incorporate this in our algorithm later.

Lemma 3.2. *If there exists a constraint in* (SOHS_{SDP}) *of the form* $\langle A_{u^*u}, G \rangle = 0$ *and the matrix A_{u^*u} is diagonal, then we can delete word u from the vector of needed words.*

Proof. Since matrix G must be positive semidefinite and 0 on the diagonal of such matrix implies the whole corresponding row and column must be 0, it follows that we can delete word u from the vector of needed words. ■

For NS polynomial $f = \sum_w a_w w$ we will denote by $\mathcal{W}_f = \{w \in \langle \underline{X}, \underline{X}^* \rangle \mid a_w \neq 0\}$ the set of all words that appear in f.

Lemma 3.3. *Let $f = \sum_i g_i^* g_i$ and $\mathcal{W} = \cup_i \mathcal{W}_{g_i}$. Then for every word $w \in \mathcal{W}$ there exists word $v \in \langle \underline{X}, \underline{X}^* \rangle$, such that $u^* u \in \mathcal{W}_f$ for $u = vw$.*

Proof. Let $w \in \mathcal{W}$. Then $w \in \mathcal{W}_{g_i}$ for some i and therefore $w^* w \in \mathcal{W}_{g_i^* g_i}$. If $w^* w \in \mathcal{W}_f$, we conclude the proof with v=1. Otherwise, if $w^* w \notin \mathcal{W}_f$, then $w^* w$ canceled and therefore there exists some $v_1 \in \langle \underline{X}, \underline{X}^* \rangle$ such that $v_1 w \in \mathcal{W}$. Let

$w_1 := v_1 w$. If $w_1^* w_1 \in \mathcal{W}_f$, we conclude the proof with $v = w^* v_1^* v_1$, otherwise we repeat the procedure with w_1 instead of w. Eventually we get $w_k \in \mathcal{W}$ where $w_k^* w_k \in \mathcal{W}_f$ and conclude the proof with $v = w^* v_1^* \cdots v_k^* v_k \cdots v_1$. ∎

Following the idea from [KP10] we define the right chip function on words, which takes some variables from the right side, i.e., $\mathrm{rc} : \langle \underline{X}, \underline{X}^* \rangle \times \mathbb{N}_0 \to \langle \underline{X}, \underline{X}^* \rangle$ by

$$\mathrm{rc}(w_1 \cdots w_k, i) := \begin{cases} 1 & i = 0 \\ w_{k-i+1} \cdots w_k & 1 \leq i \leq k \\ w_1 \cdots w_k & \text{otherwise} \end{cases}$$

where $w_j \in \{X_1, \ldots, X_n, X_1^*, \ldots, X_n^*\}$ for all j.

INPUT: $f = \sum_{w \in \langle \underline{X}, \underline{X}^* \rangle} a_w w$ *where* $f \in \operatorname{Sym} \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$ *and* $\deg f \leq 2d$.
STEP 1: *Define vector V as in Lemma 3.1.*
STEP 2: *Set* $W := \varnothing$
STEP 3: *For every* $w^* w \in \mathcal{W}_f$:
. *For* $0 \leq i \leq \deg w$:
. *if* $\mathrm{rc}(w, i) \in V$ *then*
. $W := W \cup \{\mathrm{rc}(w, i)\}$
. *end if*
. *end for*
end for
STEP 4: *While there exists* $u \in W$ *such that* $a_{u^* u} = 0$ *and* $u^* u \neq v^* z$ *for every* $v, z \in W$, *where* $v \neq z$:
. *delete u from W*
end
STEP 5: *Construct data* A_w, b, C *corresponding to the* $(\mathrm{SOHS}_{\mathrm{SDP}})$ *with* W *as vector of needed words.*
STEP 6: *Solve the* $(\mathrm{SOHS}_{\mathrm{SDP}})$ *to obtain* G.
If $(\mathrm{SOHS}_{\mathrm{SDP}})$ *is not feasible then*
. $f \notin \Sigma^2 \mathbb{R}\langle \underline{X}, \underline{X}^* \rangle$; *STOP.*
end
STEP 7: *Compute a decomposition* $G = R^t R$.
OUTPUT: *SOHS decomposition:* $f = \sum_i g_i^* g_i$, *where* g_i *denotes the* i*-th component of* RW_d.

Algorithm 2: The Newton NS chip method

Remark 3.4. In Step 7 of Algorithm 2 we can take different decompositions, e.g. a Cholesky decomposition (which is not unique if G is not positive definite), the eigenvalue decomposition, etc.

Using Lemmas 3.1, 3.2 and 3.3 we formulate the next Theorem as an analogue of the Proposition 2.4 with a (usually quite) smaller vector of needed words and justify the use of Algorithm 2.

Theorem 3.5. *Let $f \in \operatorname{Sym}\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$. Then $f \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$ if and only if there exists a positive semidefinite (PSD) matrix G satisfying*

$$f = W^*GW,$$

where W is a vector of words constructed in Algorithm 2.

Proof. If $f = W^*GW$ for some positive semidefinite matrix G, then we can write $G = R^tR$ and $f = \sum_i g_i^* g_i$ where g_i denotes the i-th component of RW.

Conversely, if $f \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$, we first notice that using Lemma 3.1 we can restrict candidates $\operatorname{rc}(w, i)$ in Step 3 of Algorithm 2 to subvector of words $V \subseteq W_d$. Next we notice that using Lemma 3.2 we can do Step 4. Therefore for the rest of the proof we need to prove that for any SOHS decomposition $f = \sum_i g_i^* g_i$ we can write all the monoms in all g_i just with words from W. Using Lemma 3.3, for every word $w \in \mathcal{W} = \cup_i \mathcal{W}_{g_i}$ there exists word $v \in \langle \underline{X}, \underline{X}^*\rangle$ such that $(vw)^*vw \in \mathcal{W}_f$. So w is a right chip of some word of the form u^*u and therefore $w \in W$. ■

We implemented Algorithm 2 in `NCSOStools` with procedure `NSsos`.

Example 3.6. Let us return to Example 2.10

$$f = X^*X - X^*X^5Y^{10}(X^*)^5 - X^5(Y^*)^{10}(X^*)^5X + X^5(Y^*)^{10}(X^*)^5X^5Y^{10}(X^*)^5,$$

where we saw that W_d has $\frac{4^{21}-1}{3}$ elements and let us look at how we can considerably reduce this vector with Algorithm 2 (Newton NS chip method). The only words in NS polynomial f that are of the form w^*w are X^*X and $X^5(Y^*)^{10}(X^*)^5X^5Y^{10}(X^*)^5$. So we need to look at all right chips of words X and $X^5Y^{10}(X^*)^5$. They have 22 right chips and using Lemma 3.1 we delete 1 from this set of candidates. In Step 4 we realize that $a_{XX^*} = 0$ and that XX^* has a unique decomposition in W, so we can delete X^* from W. Repeating this observation on other words from W we delete all but X and $X^5Y^{10}(X^*)^5$ which leads us to exactly minimum needed vector of words

$$W = \begin{bmatrix} X & X^5Y^{10}(X^*)^5 \end{bmatrix}^t.$$

Solving (SOHS_{SDP}) returns the Gram matrix

$$G = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \end{bmatrix}^t \begin{bmatrix} 1 & -1 \end{bmatrix}$$

and therefore

$$f = \big(X - X^5Y^{10}(X^*)^5\big)^* \big(X - X^5Y^{10}(X^*)^5\big) \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle.$$

Look at Example 4.2 for use of `NCSOStools` on this NS polynomial f.

4. NCSOStools and NS polynomials

`NCSOStools` is an open source Matlab toolbox freely available at

`http://ncsostools.fis.unm.si/`

which we have developed for

(1) symbolic manipulation with polynomials in non-commuting variables;
(2) constructing and solving semidefinite programs for SOHS decomposition of polynomials in non-commuting variables.

Before version 1.8 only symmetric variables were supported. As an add-on to this paper a new version 1.8 was developed to handle non-symmetric variables too. At the time of writing to the best of our knowledge this is the only Matlab toolbox with the described functionality. For solving constructed semidefinite programs different SDP solvers are supported, like SeDuMi [Stu99], SDPT3 [TTT12] and SDPA [YFK03].

For a start let us do an example with some newly developed basic operations for symbolic manipulation of non-commutative polynomials in non-symmetric variables (NS polynomials).

Example 4.1. First we must define non-commutative non-symmetric variables. For this we use a command `NSvars`.

```
>> NSvars x y
```

As usually in Matlab we used ' for involution. NSvars created four variables X, x, Y, y, where $X = x^*$ and $Y = y^*$. Now we can form NS polynomials:

```
>> f = x*Y*x^2 + 3*X*y;
>> g = -2*y*X*Y;
```

Elementary operations are implemented in the standard Matlab manner.

```
>> f + g, f - g, f*g, -f, g^2, g'*g, f'
ans = 3*X*y + x*Y*x^2 - 2*y*X*Y
ans = 3*X*y + x*Y*x^2 + 2*y*X*Y
ans = -6*X*y^2*X*Y - 2*x*Y*x^2*y*X*Y
ans = -3*X*y - x*Y*x^2
ans = 4*y*X*Y*y*X*Y
ans = 4*y*x*Y*y*X*Y
ans = X^2*y*X + 3*Y*x
```

We can also define matrices of NS polynomials and then do elementary operations.

```
>> A = [x - y*X, x + 1; x^2 - x, 1 + Y]
>> B = [x*Y*x, 3*y*X]
>> B*A, A*A, A.*A, B', trace(A), diag(A), triu(A), [sum(A);B]
```

Throughout this paper the main question was how we can efficiently decide whether a given NS polynomial can be written as a SOHS. This can be answered using Algorithm 2 from this paper with the command `NSsos`.

Example 4.2. Let

$$f = X^*X - X^*X^5Y^{10}(X^*)^5 - X^5(Y^*)^{10}(X^*)^5X + X^5(Y^*)^{10}(X^*)^5X^5Y^{10}(X^*)^5$$

from the Example 2.10. Command `NSsos` has many optional parameters, for example with parameter `.precision` we can set the smallest value that is considered to be nonzero in numerical calculations.

```
>> NSvars x y
>> f=X*x-X*x^5*y^10*X^5-x^5*Y^10*X^5*x+x^5*Y^10*X^5*x^5*y^10*X^5;
>> params.precision = 1e-6;
>> [IsSohs,G,W,sohs,gsos] = NSsos(f,params)
```

Value

```
IsSohs = 1
```

means $f \in \Sigma^2\mathbb{R}\langle \underline{X}, \underline{X}^*\rangle$.

```
G = 1.0000   -1.0000
   -1.0000    1.0000
```

is a Gram matrix of a NS polynomial f for a vector of words

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

and

```
sohs = x-x^5*y^10*X^5
```

is a vector of NS polynomials g_i (in our case just one NS polynomial), for which

$$\texttt{gsos} := \sum_i g_i^* g_i = f$$

holds.

```
gsos = X*x-X*x^5*y^10*X^5-x^5*Y^10*X^5*x+x^5*Y^10*X^5*x^5*y^10*X^5
```

Example 4.3. Let us end with one more example where we present some of the output of the procedure NSsos.

```
>> NSvars x y
>> f = Y*x^5*X*x*X^5*y-Y*x^5*X*y*X-x*Y*x*X^5*y+x*Y*x*X*y*X+x*Y*y*X;
>> params.precision = 1e-6;
>> [IsSohs,G,W,sohs,gsos] = NSsos(f,params)

***** NCsostools: module NSsos started *****
Input polynomial has (max) degree 14 and min degree 4.
Detected 5 monomials in 2 nonsymmetric variables.
There are 357913941 monomials in 2 nonsymmetric variables of degree
   at most 14.
There are 357913856 monomials in 2 nonsymmetric variables of degree
   at most 14 and at least 4.
After Newton Chip Method keeping 3 monomials.
Number of linear constraints: 6.
Starting SDP solver ...
Computing SOHS decomposition ... done.
Found SOHS decomposition with 2 factors.

*************** Polynomial is SOHS ***************

IsSohs = 1
G =
    1.0000   -0.0000    0.0000
   -0.0000    1.0000   -1.0000
    0.0000   -1.0000    1.0000
W =
    'X*y*X'
    'x*X*X*X*X*X*y'
    'y*X'
```

```
sohs =
    X*y*X
    x*X^5*y-y*X
gsos = Y*x^5*X*x*X^5*y-Y*x^5*X*y*X-x*Y*x*X^5*y+x*Y*x*X*y*X+x*Y*y*X
```

5. Conclusion

In this paper we are dealing with a problem whether a given non-commutative polynomial in non-symmetric variables (NS polynomial) can be written as a sum of Hermitian squares. First we present a theoretical way to find such a decomposition - the Gram matrix method. This is a special case of a semidefinite feasibility problem, which can unfortunately easily exceed manageable size of SDP solvers. In Chapter 3 we therefore introduce a generalization of the augmented Newton chip method [KP10] from symmetric to non-symmetric variables, which replaces the vector of needed words for SDP solver by a usually much smaller vector. The proposed method was implemented and published in the new version of the open source Matlab toolbox `NCSOStools` which fills a rather large gap in the existing software (dealing with non-symmetric variables and matrices). We also include numerous examples throughout the paper to illustrate the theory and our results. We conclude with the demonstration of how to use non-symmetric variables (NSvars) and the proposed method (procedure NSsos) in the computer algebra system `NCSOStools`.

Acknowledgments. The author thanks both anonymous referees for helpful suggestions.

References

[AL12] M.F. Anjos and J.B. Lasserre. *Handbook of Semidefinite, Conic and Polynomial Optimization: Theory, Algorithms, Software and Applications*, volume 166 of *International Series in Operational Research and Management Science*. Springer, 2012.

[BCKP13a] S. Burgdorf, K. Cafuta, I. Klep, and J. Povh. Algorithmic aspects of sums of hermitian squares of noncommutative polynomials. *Comput. Optim. Appl.*, 55(1):137–153, 2013.

[BCKP13b] S. Burgdorf, K. Cafuta, I. Klep, and J. Povh. The tracial moment problem and trace-optimization of polynomials. *Math. Program.*, 137(1):557–578, 2013.

[BGFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Studies in Applied Mathematics. SIAM, 1994.

[BGSV12] C. Bachoc, D. C. Gijswijt, A. Schrijver, and F. Vallentin. Invariant semidefinite programs. In *Handbook on semidefinite, conic and polynomial optimization*, volume 166 of *Internat. Ser. Oper. Res. Management Sci.*, pages 219–269. Springer, New York, 2012.

[BKP16] S. Burgdorf, I. Klep, and J. Povh. *Optimization of polynomials in non-commuting variables*. SpringerBriefs in Mathematics. Springer, [Cham], 2016.

[Caf13] K. Cafuta. On matrix algebras associated to sum-of-squares semidefinite programs. *Linear and Multilinear Algebra*, 61(11):1496–1509, 2013.

[Cim10] J. Cimprič. A method for computing lowest eigenvalues of symmetric polynomial differential operators by semidefinite programming. *J. Math. Anal. Appl.*, 369(2):443–452, 2010.

[CKP10] K. Cafuta, I. Klep, and J. Povh. A note on the nonexistence of sum of squares certificates for the Bessis-Moussa-Villani conjecture. *J. Math. Phys.*, 51(8):083521, 10, 2010.

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

[CKP12] K. Cafuta, I. Klep, and J. Povh. Constrained polynomial optimization problems with noncommuting variables. *SIAM J. Optim.*, 22(2):363–383, 2012.

[CKP15] K. Cafuta, I. Klep, and J. Povh. Rational sums of hermitian squares of free noncommutative polynomials. *Ars Math. Contemp.*, 9(2):243–259, 2015.

[CLR95] M.D. Choi, T.Y. Lam, and B. Reznick. Sums of squares of real polynomials. In *K-theory and algebraic geometry: connections with quadratic forms and division algebras*, volume 58 of *Proc. Sympos. Pure Math.*, pages 103–126. AMS, Providence, RI, 1995.

[dHMP08] M.C. de Oliveira, J.W. Helton, S. McCullough, and M. Putinar. Engineering systems and free semi-algebraic geometry. In *Emerging applications of algebraic geometry*, volume 149 of *IMA Vol. Math. Appl.*, pages 17–61. Springer, New York, 2008.

[Goe97] M. X. Goemans. Semidefinite programming in combinatorial optimization. *Math. Program.*, 79(1-3, Ser. B):143–161, 1997.

[GP04] K. Gatermann and P.A. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *J. Pure Appl. Algebra*, 192(1-3):95–128, 2004.

[HdKR02] M. Halická, E. de Klerk, and C. Roos. On the convergence of the central path in semidefinite optimization. *SIAM J. Optim.*, 12(4):1090–1099, 2002.

[Hel02] J.W. Helton. "Positive" noncommutative polynomials are sums of squares. *Ann. of Math. (2)*, 156(2):675–694, 2002.

[HJ85] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, 1985.

[HN12] J.W. Helton and J. Nie. A semidefinite approach for truncated K-moment problems. *Found. Comput. Math.*, 12(6):851–881, 2012.

[KP10] I. Klep and J. Povh. Semidefinite programming and sums of hermitian squares of noncommutative polynomials. *J. Pure Appl. Algebra*, 214:740–749, 2010.

[KP16] I. Klep and J. Povh. Constrained trace-optimization of polynomials in freely noncommuting variables. *J. Global Optim.*, 64(2):325–348, 2016.

[KS08a] I. Klep and M. Schweighofer. Connes' embedding conjecture and sums of Hermitian squares. *Adv. Math.*, 217(4):1816–1837, 2008.

[KS08b] I. Klep and M. Schweighofer. Sums of Hermitian squares and the BMV conjecture. *J. Stat. Phys*, 133(4):739–760, 2008.

[Las01] J.B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.*, 11(3):796–817, 2000/01.

[Las09] J.B. Lasserre. *Moments, Positive Polynomials and Their Applications*, volume 1 of *Imperial College Press Optimization Series*. Imperial College Press, London, 2009.

[Lau09] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, volume 149 of *IMA Vol. Math. Appl.*, pages 157–270. Springer, New York, 2009.

[Mar08] M. Marshall. *Positive polynomials and sums of squares*, volume 146 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2008.

[McC01] S. McCullough. Factorization of operator-valued polynomials in several noncommuting variables. *Linear Algebra Appl.*, 326(1-3):193–203, 2001.

[Mit] H. Mittelman. Website. `http://plato.asu.edu/sub/pns.html`.

[MP05] S. McCullough and M. Putinar. Noncommutative sums of squares. *Pacific J. Math.*, 218(1):167–171, 2005.

[Nie09] J. Nie. Sum of squares method for sensor network localization. *Comput. Optim. Appl.*, 43(2):151–179, 2009.

[Nie14] J. Nie. Optimality conditions and finite convergence of Lasserre's hierarchy. *Math. Program.*, 146(1-2):97–121, 2014.

[Par00] P. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, California Institute of Technology, 2000.

[Par03] P.A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Program.*, 96(2, Ser. B):293–320, 2003.

[PNA10] S. Pironio, M. Navascués, and A. Acín. Convergent relaxations of polynomial optimization problems with noncommuting variables. *SIAM J. Optim.*, 20(5):2157–2180, 2010.

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

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

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

[Sho91] N. Z. Shor. Dual quadratic estimates in polynomial and boolean programming. *Ann. Oper. Res.*, 25(1-4):163–168, 1991.

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

[TTT12] K.C. Toh, M.J. Todd, and R.H. Tütüncü. On the implementation and usage of SDPT3—a Matlab software package for semidefinite-quadratic-linear programming, version 4.0. In *Handbook on semidefinite, conic and polynomial optimization*, volume 166 of *Internat. Ser. Oper. Res. Management Sci.*, pages 715–754. Springer, New York, 2012. `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`.

[WSV00] H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite Programming.* Kluwer, 2000.

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

Kristijan Cafuta, Univerza v Ljubljani, Fakulteta za elektrotehniko, Laboratorij za uporabno matematiko in statistiko, Tržaška 25, 1000 Ljubljana, Slovenia

E-mail address: `kristijan.cafuta@fe.uni-lj.si`

NOT FOR PUBLICATION

Contents

1. Introduction 1
1.1. Notation: NS polynomials 2
1.2. Semidefinite programming 2
2. Sums of Hermitian squares of NS polynomials 3
2.1. Positive semidefinite NS polynomials 3
2.2. Sums of Hermitian squares and SDP 6
3. Newton NS chip method 8
4. NCSOStools and NS polynomials 10
5. Conclusion 13
Acknowledgments 13
References 14
Index 17