SeDuMi Pol—a Preprocessor for Optimization with Positive Polynomials ⋆

Bogdan C. Şicleru * **Bogdan Dumitrescu** *,**

* *Dept. of Automatic Control and Computers, "Politehnica" University of Bucharest, 313 Spl. Independenţei, 060042, Bucharest, Romania (e-mail: bogdan_sicleru@yahoo.com)*
** *Tampere Int. Center for Signal Processing, Tampere University of Technology, P.O. Box 553, SF-33101, Tampere, Finland (e-mail: bogdand@cs.tut.fi)*

Abstract: We present a new Matlab library, SeDuMi Pol, able to solve convex optimization problems, whose constraints may involve also positive polynomials. The way in which positive polynomials are introduced in optimization problems as variables is described. An example of using the library is also provided.

Keywords: positive polynomials, convex optimization, Matlab library, SeDuMi Pol

1. INTRODUCTION

In the last two decades there has been a high interest in the analysis of multidimensional systems using constraints that require the positivity of polynomials, once with the appearance of the semidefinite programming (SDP) methods. This also brings the need of a collection of programs capable of transforming a problem containing positive polynomials as constraints into a standard semidefinite-quadratic-linear programming (SQLP) problem. Precisely this is what we propose here: a library able to solve convex optimization problems whose constraints involve positive polynomials, named SeDuMi Pol, being the first of its kind. It assures the transparency of the parameterization of positive polynomials, basically making the user able to solve problems with positive polynomials without knowing what is needed for their characterization. (Remember that positive polynomials are also sum-of-squares). Although there are toolboxes which can handle sum-of-squares , see [12], [6], none of them has considered an approach like we propose here. SeDuMi Pol is written in Matlab [1] and uses SeDuMi [14] toolbox for optimization over symmetric cones.

Outline The remainder of this paper is organized as follows. We present in Section 2 a first view on how the SeDuMi Pol library works. The way positive polynomials are inserted in an optimization problem is treated in Section 3. Next, in Sections 4, we discuss the Matlab structure that describes a SeDuMi Pol problem. An example of usage of the library is provided in Section 5. We conclude in Section 6.

Notations The notation is standard. Multivariate entities (vectors, matrices) are denoted by bold characters. $\boldsymbol{M}^T$ is the transposed of matrix $\boldsymbol{M}$ and $\boldsymbol{M}^H$ is the transposed and complex conjugated of $\boldsymbol{M}$. We denote by $\mathbf{Tr}\,\boldsymbol{M}$ the trace of the matrix $\boldsymbol{M}$. For $a \in \mathbb{C}$, a^* is the complex conjugated of a.

2. SEDUMI POL OVERVIEW

Let us see now, at a first glance, how does SeDuMi Pol work. For this consider an SQLP problem written in the equality form

$$\begin{array}{l} \min\ \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \in \mathbb{K} \end{array} \tag{1}$$

Here $\mathbb{K}$ is a symmetric cone which is a cartesian product of a nonnegative orthant, quadratic cones and cones of semidefinite matrices; SeDuMi also allows the use of free (unrestricted) variables.

What we do is to extend the problem (1) by inserting positive polynomials constraints. Thus, (1) becomes:

$$\begin{array}{l} \min\ \tilde{\boldsymbol{c}}^T\tilde{\boldsymbol{x}} \\ \text{s.t.}\ \ \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \boldsymbol{b} \end{array} \tag{2}$$

The variable vector $\tilde{\boldsymbol{x}}$ is the vector $\boldsymbol{x}$ in which we insert the *coefficients* of the polynomials.

Having built $\tilde{\boldsymbol{A}}$, $\boldsymbol{b}$, $\tilde{\boldsymbol{c}}$ from (2), the `sedumi_pol` function (the main function of the SeDuMi Pol library) can be used to solve the problem (2). A typical call for this function is

```
[x,y,info] = sedumi_pol(AsP,bsP,csP,KsP,pars);
```

where `AsP`, `bsP`, `csP` are $\tilde{\boldsymbol{A}}$, $\boldsymbol{b}$, $\tilde{\boldsymbol{c}}$ from (2), respectively. `KsP` is a structure that describes the cone $\mathbb{K}$ from (1) and `pars` (which the user is not obliged to use) is a structure which can change the parameter settings for SeDuMi. The output parameters `x` and `y` are the solutions for the dual forms of the optimization problem involving positive polynomials, while `info` offers information about the solution obtained.

Support The current version is 1.02 and supports the following:

⋆ This work was supported by Romanian CNCSIS grant IDEI 309/2007.

[1] Matlab is a registered trademark of The MathWorks, Inc.

- positive univariate polynomials
- trigonometric or real variable
- real/complex coefficients
- scalar/matrix coefficients
- positivity intervals

3. COEFFICIENT DESCRIPTION

We present next the coefficients chosen to represent the positive polynomials, coefficients which become variables in a SeDuMi Pol problem. More precisely we define the variables that are added in the the $\tilde{\boldsymbol{x}}$ term in (2) for every type of polynomial.

We consider complex coefficients for trigonometric and hybrid polynomials. For real polynomials we have only real coefficients.

3.1 Scalar coefficients

- *Trigonometric polynomial.* Let us take a (Hermitian) trigonometric polynomial of degree $\boldsymbol{n}$ with complex coefficients

$$R(\boldsymbol{z}) = \sum_{\boldsymbol{k}=-\boldsymbol{n}}^{\boldsymbol{n}} r_{\boldsymbol{k}} \boldsymbol{z}^{-\boldsymbol{k}}, \qquad r_{-\boldsymbol{k}} = r_{\boldsymbol{k}}^*, \tag{3}$$

$\boldsymbol{k} \in \mathbb{Z}^d$, $\boldsymbol{n} \in \mathbb{N}^d$, $\boldsymbol{z} \in \mathbb{T}^d$, $r_{\boldsymbol{k}} \in \mathbb{C}$, where $\mathbb{T}$ is the unit circle. We must retain all the coefficients that belong to a halfspace $\mathcal{H}_d \in \mathbb{Z}^d$. Remember that in $\mathbb{Z}^d$ we say that $\boldsymbol{k} \in \mathcal{H}_d$ if $(k_d > 0)$ or $(k_d = 0$ and $(k_1, \ldots, k_{d-1}) \in \mathcal{H}_{d-1})$. Therefore SeDuMi Pol needs the parameters

$$\begin{aligned}\{&r_{0,\ldots,0}, \quad r_{1,0,\ldots,0}, \quad \ldots, \quad r_{n_1,0,\ldots,0},\\ &r_{-n_1,1,0,\ldots,0}, \quad \ldots, \quad r_{-n_1,n_2,\ldots,n_d}, \quad \ldots,\\ &r_{n_1,n_2,\ldots,n_d}\}.\end{aligned} \tag{4}$$

The total number of coefficients in (4) is

$$M = \frac{1 + \prod_{i=1}^{d}(2n_i + 1)}{2}. \tag{5}$$

Trigonometric polynomials have been discussed in [5, 1, 8, 2, 3].

- *Real Polynomial.* Let $P \in \mathbb{R}_{\boldsymbol{n}}[\boldsymbol{t}]$,

$$P(\boldsymbol{t}) = \sum_{\boldsymbol{k}=\boldsymbol{0}}^{\boldsymbol{n}} p_{\boldsymbol{k}} \boldsymbol{t}^{\boldsymbol{k}}, \tag{6}$$

be a real polynomial. We need all the coefficients of the polynomial. Hence the coefficients are

$$\begin{aligned}\{&p_{0,\ldots,0}, \quad \ldots, \quad p_{n_1,0,\ldots,0}, \; p_{0,1,0,\ldots,0},\\ &\ldots, \quad p_{n_1,1,0,\ldots,0}, \quad \ldots, \quad p_{0,n_2,\ldots,n_d},\\ &\ldots, \quad p_{n_1,n_2,\ldots,n_d}\}.\end{aligned} \tag{7}$$

See [11, 13, 3].

- *Hybrid polynomial.* Consider a hybrid polynomial [4]

$$\begin{aligned}H(z_1, \ldots, z_\ell, t_1, \ldots, t_m) = &\sum_{i_1=-n_1}^{n_1} \cdots \sum_{i_\ell=-n_\ell}^{n_\ell} \sum_{i_{\ell+1}=0}^{n_{\ell+1}} \cdots\\ \cdots &\sum_{i_d=0}^{n_d} h_{i_1,\ldots,i_\ell,i_{\ell+1},\ldots,i_d} z_1^{i_1} \cdots z_\ell^{i_\ell} t_{\ell+1}^{i_{\ell+1}} \cdots t_d^{i_d},\end{aligned} \tag{8}$$

with $z_i \in \mathbb{T}$, $t_j \in \mathbb{R}$ and $h_{i_1,\ldots,i_\ell,i_{\ell+1},\ldots,i_d} \in \mathbb{C}$, $\forall i = 1:\ell$, $j = 1:m$. The relation

$$\begin{aligned}&h_{i_1,\ldots,i_{u-1},-i_u,\ldots,-i_v,i_{v+1},\ldots,i_\ell,\alpha_1,\ldots,\alpha_m} =\\ &h^*_{-i_1,\ldots,-i_{u-1},i_u,\ldots,i_v,-i_{v+1},\ldots,-i_\ell,\alpha_1,\ldots,\alpha_m}\end{aligned} \tag{9}$$

$\forall u, v = 1:\ell$, $u \le v$, $i_i = -n_i : n_i$, $\forall i = 1:\ell$, $\alpha_j = 0 : n_j$, $\forall j = 1:m$, implies that the polynomial (8) takes real values on $\mathbb{T}^\ell \times \mathbb{R}^m$.

The coefficients we need are the ones that correspond to an ℓ-tuple from $\mathcal{H}_\ell$. As said, we choose

$$\{h_{i_1,\ldots,i_\ell,i_{\ell+1},\ldots,i_d}\}, \qquad (i_1, i_2, \ldots, i_\ell) \in H_\ell. \tag{10}$$

The elements of the set from (10) are ordered as follows: for every $(i_{\ell+1}, \ldots, i_d)$ $(d-\ell)$-tuple, covered like in (7), we choose all the $(i_1, \ldots, i_\ell)$ possible ℓ-tuples, the order being just like in (4).

3.2 Matrix coefficients

Next, we extend the results to polynomials with matrix coefficients. Likewise the case with scalar coefficients the order in which we consider the coefficients is the same. The difference is that now we have matrices, instead of scalars. So, we vectorize the matrices.

- *Trigonometric polynomial.* We consider a trigonometric polynomial with matrix coefficients having the form

$$\boldsymbol{R}(\boldsymbol{z}) = \sum_{\boldsymbol{k}=-\boldsymbol{n}}^{\boldsymbol{n}} \boldsymbol{R}_{\boldsymbol{k}} \boldsymbol{z}^{-\boldsymbol{k}}, \qquad \boldsymbol{R}_{-\boldsymbol{k}} = \boldsymbol{R}_{\boldsymbol{k}}^H, \tag{11}$$

$\boldsymbol{R}_{\boldsymbol{k}} \in \mathbb{C}^{\kappa\times\kappa}$. Let vec() be the function that transforms a matrix into a vector by stacking its columns. The symmetry relation from (11) tells that $\boldsymbol{R}_{\boldsymbol{0}}$ is Hermitian, hence only half of its elements are needed. Let vecs() be the function that transforms the lower part of a Hermitian matrix into a vector, by stacking the relevant part of its columns. Taking (4) into account the coefficients of (11) are described by the vector

$$\begin{aligned}\{&\mathrm{vecs}(\boldsymbol{R}_{0,\ldots,0}), \quad \mathrm{vec}(\boldsymbol{R}_{1,0,\ldots,0}), \quad \ldots,\\ &\mathrm{vec}(\boldsymbol{R}_{n_1,0,\ldots,0}), \quad \mathrm{vec}(\boldsymbol{R}_{-n_1,1,0,\ldots,0}), \; \ldots,\\ &\mathrm{vec}(\boldsymbol{R}_{n_1,n_2,\ldots,n_d})\}.\end{aligned} \tag{12}$$

- *Real polynomial.* For a real polynomial with matrix coefficients,

$$\boldsymbol{P}(\boldsymbol{t}) = \sum_{\boldsymbol{k}=\boldsymbol{0}}^{\boldsymbol{n}} \boldsymbol{P}_{\boldsymbol{k}} \boldsymbol{t}^{\boldsymbol{k}}, \qquad \boldsymbol{P}_{\boldsymbol{k}} = \boldsymbol{P}_{\boldsymbol{k}}^T, \tag{13}$$

$\boldsymbol{P}_{\boldsymbol{k}} \in \mathbb{R}^{\kappa\times\kappa}$, the elements that we choose for describing the coefficients are

$$\begin{aligned}\{&\mathrm{vecs}(\boldsymbol{P}_{0,\ldots,0}), \; \mathrm{vecs}(\boldsymbol{P}_{1,0,\ldots,0}), \quad \ldots,\\ &\mathrm{vecs}(\boldsymbol{P}_{n_1,0,\ldots,0}), \quad \ldots, \quad \mathrm{vecs}(\boldsymbol{P}_{n_1,\ldots,n_d})\}.\end{aligned} \tag{14}$$

(See (7).)

- *Hybrid polynomial.* Let

$$\begin{aligned}\boldsymbol{H}(z_1, \ldots, z_\ell, t_1, \ldots, t_m) = &\sum_{i_1=-n_1}^{n_1} \cdots \sum_{i_\ell=-n_\ell}^{n_\ell} \sum_{i_{\ell+1}=0}^{n_{\ell+1}} \cdots\\ \cdots &\sum_{i_d=0}^{n_d} \boldsymbol{H}_{i_1,\ldots,i_\ell,i_{\ell+1},\ldots,i_d} z_1^{i_1} \cdots z_\ell^{i_\ell} t_{\ell+1}^{i_{\ell+1}} \cdots t_d^{i_d},\end{aligned} \tag{15}$$

with $z_i \in \mathbb{T}$ and $t_j \in \mathbb{R}$, $i = 1:\ell$, $j = 1:m$, be a multivariate hybrid polynomial with real matrix coefficients. The corresponding symmetry relation to (9) is

$$\begin{aligned}&\boldsymbol{H}_{i_1,\ldots,i_{u-1},-i_u,\ldots,-i_v,i_{v+1},\ldots,i_\ell,\alpha_1,\ldots,\alpha_m} =\\ &\boldsymbol{H}^H_{-i_1,\ldots,-i_{u-1},i_u,\ldots,i_v,-i_{v+1},\ldots,-i_\ell,\alpha_1,\ldots,\alpha_m},\end{aligned} \tag{16}$$

$\forall u, v = 1 : \ell,\ u \le v,\ i_i = -n_i : n_i,\ \forall i = 1 : \ell,\ \alpha_j = 0 : n_j,\ \forall j = 1 : m$. The values that we choose are the matrix coefficients vectorized by applying the vec() function to all of them but $\boldsymbol{H_0}$, to which we apply vecs(). The order in which we take the coefficients is the same as in (10).

4. STRUCTURE OF SEDUMI POL

In the structure `KsP` that describes the cone, we introduce two new fields for describing the positive polynomials: `ptype` and `p`.

If the optimization problem has N positive polynomials among its variables, then `KsP.ptype` is an N-by-1 cell array, each cell denoting the type of a polynomial. To be able to describe the polynomials these cells have several (sub)fields.

There are three types of polynomials: trigonometric, real and hybrid. In order to specify the type of polynomial the user must explicitly give the *number* of trigonometric and/or real variables; for this, each cell of `KsP.ptype` has two fields: `trigonometric` and `real`. If the polynomial does not have real variables the field `real` can be omitted. By default, if none of the fields `trigonometric` and `real` appears, it is considered that the polynomial is trigonometric; its number of variables is given by `KsP.p`, see (17).

The coefficients of the polynomials can be real or complex. To change between these two types one of the fields `real_coef` and `complex_coef` can be set in `ptype`. By default, real coefficients are considered to work with, so only in the complex case you need to set one of them, the `complex_coef` field, to 1. Remember in the complex case also to set `KsP.ycomplex`, as requested by SeDuMi.

Finally, if global positivity is not desired, `ptype` can have one more field, for the description of the positivity domain of the polynomials: `int`. Currently we have support only for *univariate* polynomials, so the field `int` can be an interval or a reunion of two intervals. See Table 1 for values of field `int`; other cases than those listed in the table do not exist. The positivity on intervals of real positive polynomials was discussed in [9, 10, 3] and of trigonometric polynomials in [1, 7, 3].

Table 1. Positivity intervals types.

Case	Domain	Matlab input	Type of polynomial
1	$[a, b] \in (-\pi, \pi)$	[a b]	trig. pol. complex coef.
2	$[a, \pi] \in (-\pi, \pi)$	[a pi]	trig. pol. complex coef.
3	$[-\pi, b] \in (-\pi, \pi)$	[-pi b]	trig. pol. complex coef.
4	$[a, b] \in [0, \pi]$	[a b]	trig. pol. real coef.
5	$[-\pi, a] \cup [b, \pi]$	[-pi a b pi]	trig. pol. complex coef.
6	$[0, a] \cup [b, \pi]$	[0 a b pi]	trig. pol. real coef.
7	$[a, b] \in (-\infty, \infty)$	[a b]	real pol.
8	$[a, \infty)$	[a Inf]	real pol.
9	$(-\infty, b]$	[-Inf b]	real pol.
10	$(-\infty, a] \cup [b, \infty)$	[-Inf a b Inf]	real pol. (even degree)

The field `p`, in the structure `KsP`, is also a cell array, having the same number of elements as `KsP.ptype`, one for each polynomial. `KsP.p` holds the degree of a d-variate polynomial, $\boldsymbol{n} = (n_1, n_2, \ldots, n_d) \in \mathbb{Z}^d$, and the size of the $\kappa \times \kappa$ matrix polynomial coefficients. (Scalar coefficients are of size 1×1). Hence, `KsP.p` has the following structure:

$$[\ n_1\ n_2\ \ldots n_d\ \kappa\ ]. \tag{17}$$

If `KsP.p` has only one element, n, the polynomial is univariate with scalar coefficients and degree n.

Figure 1 gives an overview of the fields *added* in the SeDuMi structure to describe the polynomials in SeDuMi Pol.

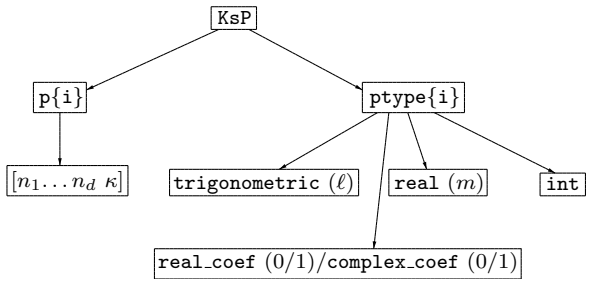

Fig. 1. `KsP`—the structure of SeDuMi Pol.

5. EXAMPLE OF USAGE

We show next an example of solving optimization problems involving positive polynomials using SeDuMi Pol. Let us consider the problem of designing linear-phase FIR filters of even order $n = 2\tilde{n}$, using optimization, problem which can be found in [3, Section 5.1.1]. We optimize only the magnitude, therefore we can work with zero-phase filters

$$\tilde{H}(\omega) = \sum_{k=-\tilde{n}}^{\tilde{n}} \tilde{h}_k z^{-k}, \qquad \tilde{h}_{-k} = \tilde{h}_k. \tag{18}$$

A peak constrained least-squares (PCLS) problem can be formulated as

$$\begin{aligned} \min_{\tilde{H} \in \mathbb{R}_{\tilde{n}}} \ & E_s \\ \text{s.t.} \ & 1 + \gamma_p - \tilde{H}(\omega) \ge 0, \forall \omega \in [0, \omega_p] \\ & \tilde{H}(\omega) - 1 + \gamma_p \ge 0, \forall \omega \in [0, \omega_p] \\ & \gamma_s - \tilde{H}(\omega) \ge 0, \forall \omega \in [\omega_s, \pi] \\ & \tilde{H}(\omega) + \gamma_s \ge 0, \forall \omega \in [\omega_s, \pi] \end{aligned} \tag{19}$$

where E_s is the stopband energy of the FIR filter and γ_p, γ_s are the passband and stopband given error bounds, respectively. We have now a problem with polynomials that are nonnegative on given intervals.

Let us see the expression of the stopband energy. Considering $\tilde{\boldsymbol{h}} = [\tilde{h}_0\ \tilde{h}_1 \ldots \tilde{h}_{\tilde{n}}]^T$ we have

$$E_s = \tilde{\boldsymbol{h}}^T \tilde{\boldsymbol{C}} \tilde{\boldsymbol{h}}, \text{ with } \tilde{\boldsymbol{C}} = \boldsymbol{P}^T \boldsymbol{C} \boldsymbol{P} \succeq \boldsymbol{0} \tag{20}$$

where

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{J}_{\tilde{n}} \\ 1 & 0 \\ \boldsymbol{0} & \boldsymbol{I}_{\tilde{n}} \end{bmatrix} \tag{21}$$

and $\boldsymbol{C} = \text{Toep}(c_0, c_1, \ldots, c_n) \succeq \boldsymbol{0}$ with

$$c_k = \begin{cases} 1 - \omega_s/\pi, & \text{if } k = 0, \\ -\dfrac{\sin k\omega_s}{k\pi}, & \text{if } k > 0, \end{cases} \tag{22}$$

and Toep() is the symmetric Toeplitz matrix having its first row the scalar input arguments. ($\boldsymbol{J}_{\tilde{n}}$ is the counteridentity matrix of size $\tilde{n} \times \tilde{n}$.)

Denoting

$$\begin{aligned} S_0(\omega) &= 1+\gamma_p-\tilde{H}(\omega) \\ S_1(\omega) &= \tilde{H}(\omega)-1+\gamma_p \\ S_2(\omega) &= \gamma_s-\tilde{H}(\omega) \\ S_3(\omega) &= \tilde{H}(\omega)+\gamma_s \end{aligned} \tag{23}$$

and using (20) we can transform the problem (19) into a SeDuMi Pol problem

$$\begin{aligned} &\min\ \varepsilon \\ &\text{s.t. (23)} \\ &\quad \boldsymbol{y}=\tilde{\boldsymbol{C}}^{\frac{1}{2}}\tilde{\boldsymbol{h}} \\ &\quad ||y||\le\varepsilon \\ &\quad S_0(\omega)\ge 0, \forall\omega\in[0,\omega_p] \\ &\quad S_1(\omega)\ge 0, \forall\omega\in[0,\omega_p] \\ &\quad S_2(\omega)\ge 0, \forall\omega\in[\omega_s,\pi] \\ &\quad S_3(\omega)\ge 0, \forall\omega\in[\omega_s,\pi] \end{aligned} \tag{24}$$

where the variables are $\tilde{\boldsymbol{h}}$, ε, $\boldsymbol{y}$ and the coefficients of the causal parts of the polynomials S_i, $i=0:3$. The following system characterizes the equality constraints of the problem (24):

$$\left[\begin{array}{ccccccc} \boldsymbol{I}_{\tilde{n}+1} & 0 & \boldsymbol{0} & \boldsymbol{I}_{\tilde{n}+1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \hline -\boldsymbol{I}_{\tilde{n}+1} & 0 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\tilde{n}+1} & \boldsymbol{0} & \boldsymbol{0} \\ \hline \boldsymbol{I}_{\tilde{n}+1} & 0 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\tilde{n}+1} & \boldsymbol{0} \\ \hline -\boldsymbol{I}_{\tilde{n}+1} & 0 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\tilde{n}+1} \\ \hline -\tilde{\boldsymbol{C}}^{\frac{1}{2}} & 0 & \boldsymbol{I}_{\tilde{n}+1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{array}\right] \left[\begin{array}{c} \tilde{\boldsymbol{h}} \\ \varepsilon \\ \boldsymbol{y} \\ \boldsymbol{s}_0 \\ \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \boldsymbol{s}_3 \end{array}\right] = \left[\begin{array}{c|c|c|c|c} 1+\gamma_p & \boldsymbol{0} & -1+\gamma_p & \boldsymbol{0} & \gamma_s \quad \boldsymbol{0} \end{array}\right]^T, \tag{25}$$

where $\boldsymbol{s}_i$, $i=0:3$ are the vectors of the distinct coefficients of the S_i polynomials.

We name the function that designs the linear-phase filters according to (24), `lpf`. To design linear-phase FIR filters, if one would choose for instance $n=50$, $\omega_p=0.2\pi$, $\omega_s=0.25\pi$, $\gamma_p=0.1$, $\gamma_s=0.0158$, then would enter the following command in Matlab:

```
>> th = lpf(25,0.2*pi,0.25*pi,0.1,0.0158);
```

(The value given above to γ_s corresponds to a stopband attenuation of 36 dB.)

To plot the frequency response of the filter (shown in Figure 2) one can use the command

```
>> fvtool([flipud(th(2:end));th]);
```

The listing of the function `lpf` is in the Table 2. Let us comment on the code of the function.

Comments. We define `wtn`, `wp`, `ws`, `gp`, `gs` input parameters for the `lpf` function, corresponding to $\tilde{n}$, ω_p, ω_s, γ_p, γ_s, respectively. In line 4 we establish the number of equality constraints and in line 5 the number of variables of the SeDuMi Pol problem; according to these values we initialize the `AsP`, `bsP` and `csP` variables in lines 8–10. We set the objective function in line 12. The `bsP` term is set in lines 14–17 corresponding to the vector from the right part of equality (25). Then, also according to (25), the matrix that describes the equality constraints is set in lines 19–29 and 44–45. The SeDuMi Pol problem (24) has $\tilde{n}+1$ unrestricted variables and one constraint belonging to a second-order cone (SOC), as described in the lines 47 and 48, respectively. The lines 50–53 state that the problem has four trigonometric polynomials depending each on one variable. In the lines 55–58 we set the positivity interval for each polynomial. Each polynomial has degree $\tilde{n}$ and scalar coefficients, as set in lines 60–63. All being set, in line 65, the `sedumi_pol` function is used to solve the optimization problem. Finally, the distinct coefficients of the linear-phase FIR filter—the vector $\tilde{\boldsymbol{h}}$—are returned in the line 67. ■

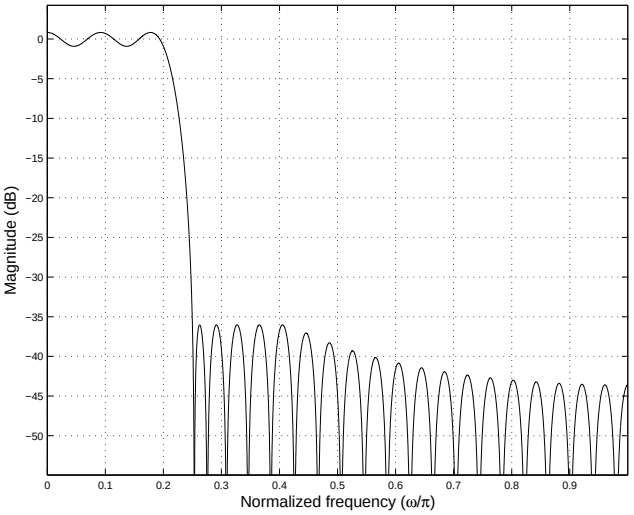

Fig. 2. Frequency response of linear phase FIR filter.

6. CONCLUSIONS

We have presented the SeDuMi Pol Matlab library capable of solving convex optimization problems having positive polynomials constraints also. The library transforms a problem with positive polynomials into one SQLP problem, which is then passed to SeDuMi, a Matlab toolbox for solving optimization problems over symmetric cones. The SeDuMi Pol library greatly reduces the effort of solving convex optimization problems with positive polynomials constraints. It is available at http://www.schur.pub.ro/Idei2007/SeDuMi_Pol_102.zip.

Further work will be devoted to extend our library to multivariate polynomials.

Appendix A. PARAMETERIZATION OF SUM-OF-SQUARES POLYNOMIALS

We present here the parameterizations used to characterize 2-D sum-of-squares polynomials with scalar coefficients, the generalization to more than two variables being straightforward. The case of polynomials with matrix coefficients can be found in [3].

A.1 Trigonometric polynomials

Considering $d=2$ in (3) we obtain a 2-D trigonometric polynomial

$$R(z_1,z_2)=\sum_{k_1=-n_1}^{n_1}\sum_{k_2=-n_2}^{n_2} r_{k_1,k_2}z_1^{-k_1}z_2^{-k_2}, \tag{A.1}$$

Table 2. SeDuMi Pol program for solving the problem (24).

```
function wth = lpf(wtn,wp,ws,gp,gs)

wtn1 = wtn+1;
nConstr = 5*wtn1;
nVar = nConstr+wtn1+1;
I = speye(wtn1);

AsP = sparse(nConstr,nVar);
bsP = sparse(nConstr,1);
csP = sparse(1,nVar);

csP(wtn1+1) = 1;

bsP(1) = 1+gp;
bsP(1+wtn1) = -1+gp;
bsP(1+2*wtn1) = gs;
bsP(1+3*wtn1) = gs;

AsP(1:wtn1,1:wtn1) = I;
AsP(1:wtn1,2*wtn1+2:3*wtn1+1) = I;

AsP(wtn1+1:2*wtn1,1:wtn1) = -I;
AsP(wtn1+1:2*wtn1,3*wtn1+2:4*wtn1+1) = I;

AsP(2*wtn1+1:3*wtn1,1:wtn1) = I;
AsP(2*wtn1+1:3*wtn1,4*wtn1+2:5*wtn1+1) = I;

AsP(3*wtn1+1:4*wtn1,1:wtn1) = -I;
AsP(3*wtn1+1:4*wtn1,5*wtn1+2:6*wtn1+1) = I;

wtn21 = 2*wtn+1;
c = zeros(wtn21,1);
c(1) = 1-ws/pi;
for i = 2:wtn21
    c(i) = -sin((i-1)*ws)/((i-1)*pi);
end
C = toeplitz(c);
P = [zeros(wtn,1) fliplr(eye(wtn));...
          1         zeros(1,wtn)  ;...
     zeros(wtn,1)     eye(wtn)];
wtC = P'*C*P;
wtC12 = real(sqrtm(wtC));

AsP(4*wtn1+1:5*wtn1,1:wtn1) = -wtC12;
AsP(4*wtn1+1:5*wtn1,wtn1+2:2*wtn1+1) = I;

KsP.f = wtn1;
KsP.q = wtn1+1;

KsP.ptype{1}.trigonometric = 1;
KsP.ptype{2}.trigonometric = 1;
KsP.ptype{3}.trigonometric = 1;
KsP.ptype{4}.trigonometric = 1;

KsP.ptype{1}.int = [0 wp];
KsP.ptype{2}.int = [0 wp];
KsP.ptype{3}.int = [ws pi];
KsP.ptype{4}.int = [ws pi];

KsP.p{1} = [wtn 1];
KsP.p{2} = [wtn 1];
KsP.p{3} = [wtn 1];
KsP.p{4} = [wtn 1];

x = sedumi_pol(AsP,bsP,csP,KsP);

wth = x(1:wtn1);
```

with $r_{-k_1,-k_2} = r^*_{k_1,k_2}$, where $k_i \in \mathbb{Z}$, $n_i \in \mathbb{N}$, $z_i \in \mathbb{T}$, $i = 1:2$, $r_{k_1,k_2} \in \mathbb{C}$.

Theorem 1. The polynomial $R(z_1, z_2)$ is sum-of-squares if and only if there exists a positive semidefinite matrix $\boldsymbol{Q} \in \mathbb{C}^{N_r \times N_r}$ such that

$$r_{k_1,k_2} = \mathbf{Tr}[\boldsymbol{\Theta}_{k_2} \otimes \boldsymbol{\Theta}_{k_1} \cdot \boldsymbol{Q}], \tag{A.2}$$

where $\boldsymbol{\Theta}_k$ is the elementary Toeplitz matrix with ones on the k-th diagonal and $N_r = (n_1 + 1)(n_2 + 1)$. The matrix $\boldsymbol{Q}$ is called a *Gram* matrix associated with the polynomial (A.1). ■

A.2 Real polynomials

We take now a polynomial $P \in \mathbb{R}_{n_1,n_2}[\boldsymbol{t}]$

$$P(t_1, t_2) = \sum_{k_1=0}^{n_1} \sum_{k_2=0}^{n_2} p_{k_1,k_2} t_1^{k_1} t_2^{k_2}, \tag{A.3}$$

where $k_i, n_i \in \mathbb{N}$, $t_i \in \mathbb{R}$, $i = 1:2$, $p_{k_1,k_2} \in \mathbb{R}$.

Theorem 2. The polynomial $P(t_1, t_2)$ is sum-of-squares if and only if there exists a positive semidefinite matrix $\boldsymbol{Q} \in \mathbb{C}^{N_t \times N_t}$ such that

$$p_{k_1,k_2} = \mathbf{Tr}[\boldsymbol{\Upsilon}_{k_2} \otimes \boldsymbol{\Upsilon}_{k_1} \cdot \boldsymbol{Q}], \tag{A.4}$$

where $\boldsymbol{\Upsilon}_k$ is the elementary Hankel matrix with ones on the k-th antidiagonal and $N_t = (n_1/2 + 1)(n_2/2 + 1)$. ■

A.3 Hybrid polynomials

Taking $\ell = m = 1$ in (8) we have a 2-D hybrid polynomial

$$H(z, t) = \sum_{k_1=-n_1}^{n_1} \sum_{k_2=0}^{n_2} h_{k_1,k_2} z^{-k_1} t^{k_2}, \tag{A.5}$$

with $r_{-k_1,k_2} = r^*_{k_1,k_2}$, where $k_1 \in \mathbb{Z}$, $k_2, n_i \in \mathbb{N}$, $i = 1:2$, $z \in \mathbb{T}$, $t \in \mathbb{R}$.

Theorem 3. The polynomial $H(z, t)$ is sum-of-squares if and only if there exists a positive semidefinite matrix $\boldsymbol{Q} \in \mathbb{C}^{N_h \times N_h}$ such that

$$h_{k_1,k_2} = \mathbf{Tr}[\boldsymbol{\Theta}_{k_1} \otimes \boldsymbol{\Upsilon}_{k_2} \cdot \boldsymbol{Q}], \tag{A.6}$$

where $N_h = (n_1 + 1)(n_2/2 + 1)$. ■

REFERENCES

[1] B. Alkire and L. Vandenberghe. Convex optimization problems involving finite autocorrelation sequences. *Math. Progr. ser. A*, 93(3):331–359, 2002.

[2] M.A. Dritschel. On Factorization of Trigonometric Polynomials. *Integr. Equ. Oper. Theory*, 49:1142, 2004.

[3] B. Dumitrescu. *Positive Trigonometric Polynomials and Signal Processing Applications.* Springer-Verlag, 2007.

[4] B. Dumitrescu and R. Ştefan. Positive Hybrid Real-Trigonometric Polynomials and Applications. In *Math. Theory of Networks and Systems.* Blacksburg, Virginia, 2008.

[5] Y. Genin, Y. Hachez, Yu. Nesterov, and P. Van Dooren. Optimization Problems over Positive Pseudopolynomial Matrices. SIAM J. Matrix Anal. Appl., 25(1):57–79, 2003.

[6] D. Henrion, J. B. Lasserre. GloptiPoly: Global optimization over polynomials with Matlab

and SeDuMi. *ACM Transactions on Mathematical Software*, 29(2): 165–194, June 2003. http://www.laas.fr/∼henrion/software/gloptipoly/.

[7] J.W. McLean and H.J.Woerdeman. Spectral Factorizations and Sums of Squares Representations via Semidefinite Programming. *SIAM J. Matrix Anal. Appl.*, 23(3):646–655, 2002.

[8] A. Megretski. Positivity of Trigonometric Polynomials. In *Proc. 42nd IEEE Conf. Decision Control (CDC)*, 3:3814–3817, Hawaii, USA, Dec. 2003.

[9] G. Pólya and G. Szegö. *Problems and Theorems in Analysis II.* Springer-Verlag, New York, 1976.

[10] V. Powers and B. Reznick. *Polynomials That Are Positive on an Interval. Trans. Amer. Math. Soc.*, 352:4677–4692, 2000.

[11] A. Prestel and C.N. Delzell. *Positive Polynomials: From Hilberts 17th Problem to Real Algebra.* Springer Monographs in Mathematics, Berlin, 2001.

[12] S. Prajna, A. Papachristodoulou, P. Seiler and P.A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2004. http://www.cds.caltech.edu/sostools, http://www.mit.edu/∼parrilo/sostools.

[13] B. Reznick. Uniform Denominators in Hilberts 17th Problem. *Math. Z.*, 220:7598, 1995.

[14] J.F. Sturm. Using SeDuMi, a Matlab Toolbox for Optimization over Symmetric Cones. *Optimization Methods and Software*, 11-12:625–653, 1999. http://sedumi.ie.lehigh.edu.