FIR FILTER OPTIMIZATION WITH POS3POLY IN CVX

Bogdan C. Şicleru[1], *Bogdan Dumitrescu*[1,2]

[1]Dept. of Automatic Control and Computers
"Politehnica" University of Bucharest
313 Spl. Independenţei, 060042 Bucharest, Romania

[2]Tampere Int. Center for Signal Processing
Tampere University of Technology
P.O.Box 553, SF-33101 Tampere, Finland

e-mail: bogdan.sicleru@schur.pub.ro, bogdan.dumitrescu@tut.fi

ABSTRACT

POS3POLY is a new library for solving convex optimization problems with positive polynomials, designed and written by the current authors. POS3POLY allows simple descriptions of three types of positive multivariate polynomials, namely trigonometric, real and hybrid, sparing the user of the task of parameterizing them with linear matrix inequalities. Positivity on semialgebraic domains and a polynomial Bounded Real Lemma are also treated. In this paper, we present the features of POS3POLY that allow the manipulation of positive trigonometric polynomials in CVX and are useful in FIR filter optimization. Three fully detailed examples are given, for designing peak-constrained least-squares linear-phase filters, sparse 2D linear-phase filters and 2D approximately linear-phase filters.

1. INTRODUCTION

FIR filter design is one of the main beneficiaries of the developments in convex optimization [3]. Many recent methods for optimizing one- or multi-dimensional FIR filters appeal to linear, quadratic and semidefinite programming (SQLP) as a basic tool. The design is facilitated by several software libraries appeared in the last decade as a practical outcome of the theoretical progress of convex optimization methods. For SQLP, leading examples are SeDuMi [11] and SDPT3 [12], in which the SQLP problem has to be described in an elementary standard form. A further step in easing designer's task was taken with CVX [6], which allows modeling convex problems, including but not limited to SQLP, using a natural language similar to the mathematical description of an optimization problem; the transformation to an equivalent SQLP problem is hidden to the user. YALMIP [8] is another modeling tool in the same vein, dealing also with some instances of nonconvex optimization.

Some of the FIR filter design methods appeal to positive polynomials and their parameterization via linear matrix inequalities [5]. Relatively simple for 1D globally positive polynomials, the parameterizations become cumbersome for polynomials that are positive on intervals or domains, in the multidimensional case. The library POS3POLY (www.schur.pub.ro/pos3poly) is an extension of SeDuMi and, through it, of other SQLP libraries, which allows a simplified description of convex optimization problems that include positive polynomials. POS3POLY solves the problem

$$\begin{array}{ll} \min & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x}=\mathbf{b}, \quad \mathbf{x}\in\mathbb{K}\times\mathbb{P} \end{array} \tag{1}$$

where $\mathbb{K}$ is the SQLP cone (cartesian product of nonnegative orthant, second order cone and semidefinite matrices cone) and $\mathbb{P}$ is a cartesian product of diverse cones of positive polynomials. POS3POLY transforms (1) into the standard SQLP problem

$$\begin{array}{ll} \min & \tilde{\mathbf{c}}^T\tilde{\mathbf{x}} \\ \text{s.t.} & \tilde{\mathbf{A}}\tilde{\mathbf{x}}=\tilde{\mathbf{b}}, \quad \tilde{\mathbf{x}}\in\mathbb{K} \end{array} \tag{2}$$

by taking advantage of the available parameterizations of sum-of-squares polynomials via semidefinite (Gram) matrices. For multivariate polynomials, (2) is a sum-of-squares relaxation of the original problem (1), hence may not have the same solution. However, as revealed by empirical evidence, the solution of (2) in practical problems is usually identical or very close to the solution of (1).

POS3POLY can also be used for creating the set of positive polynomials inside CVX. In CVX, the linear constraints need not be expressed as a single system $\mathbf{A}\mathbf{x}=\mathbf{b}$ and equalities can be mixed with inequalities. However, after internal transformations, CVX solves a problem of form (2). Hence, POS3POLY offers support for both the high-end user who needs only recognize and summarily describe the convex optimization problem and the more optimization-knowledgeable user, who is familiar with SeDuMi but does not have to bother with the intricacies of positive polynomials parameterizations.

POS3POLY allows working with three types of polynomials (hence the '3' in the name): trigonometric, real and hybrid (real-trigonometric). The polynomials can be univariate or multivariate, having real or complex, scalar or matrix coefficients. The positivity can be global or only on semialgebraic domains (intervals in the univariate case). A Bounded Real Lemma (BRL) for polynomials is also implemented.

Other libraries for sum-of-squares optimization like SOSTOOLS [10] or GloptiPoly [7] are dedicated to real polynomials and cannot efficiently integrate SQLP constraints. CVX contains a module for positive univariate polynomials and YALMIP for real sum-of-squares. Compared to these predecessors, POS3POLY comes not only with a novel approach, but with a more complete treatment of polynomial positivity.

This paper shows the use of POS3POLY in the design of FIR filters, hence the presentation will be restrained to trigonometric polynomials (fully implemented only in POS3POLY; real and hybrid polynomials are treated similarly, see POS3POLY's guide on the site.) Due to space

This work was supported by CNCSIS-UEFISCSU, project PNII – IDEI 309/2007, Tekes FiDiPro – Finland Distinguished Professor Programme and the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/6/1.5/S/16.

constraints, we only describe POS3POLY use within CVX, since it leads to more compact programs. We solve several 1D and 2D FIR filter design problems. Proof to how much POS3POLY and CVX simplify the programming burden is the fact that the paper contains the full code for these problems. The following sections are each dedicated to a feature of POS3POLY; the FIR filter optimization problems are presented as examples.

2. POSITIVE TRIGONOMETRIC POLYNOMIALS

A symmetric (zero-phase) d-variate trigonometric polynomial has the form

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

where $\mathbf{z} = (z_1, z_2, \ldots, z_d)$, $\mathbf{z}^{\mathbf{k}} = z_1^{k_1} \ldots z_d^{k_d}$ and * denotes complex conjugation; the sum in (3) goes for all $\mathbf{k} \in \mathbb{N}^d$ such that $-\mathbf{n} \le \mathbf{k} \le \mathbf{n}$. The polynomial has real values on the unit d-circle $\mathbb{T}^d$. Due to symmetry, the coefficients defining the polynomial belong to a halfspace; for convenience, we use the halfspace $\mathscr{H}_d \in \mathbb{Z}^d$ defined by: $\mathbf{k} \in \mathscr{H}_d$ if $(k_d > 0)$ or $(k_d = 0$ and $(k_1, \ldots, k_{d-1}) \in \mathscr{H}_{d-1})$. In POS3POLY, the polynomial is described by the vector

$$\begin{array}{l}[r_{(0,\ldots,0)}, r_{(1,0,\ldots,0)}, \ldots, r_{(n_1,0,\ldots,0)}, r_{(-n_1,1,0,\ldots,0)}, \\ \ldots, r_{(-n_1,n_2,\ldots,n_d)}, \ldots, r_{(n_1,n_2,\ldots,n_d)}]^T.\end{array} \tag{4}$$

The length of this vector is $\left[1 + \prod_{i=1}^{d} (2n_i + 1)\right]/2$.

For example, the 1D polynomial (or linear-phase FIR filter)

$$R(z) = 3 + 2(z^{-1} + z) + (z^{-2} + z^2)$$

is represented through the vector

```
r = [3 2 1]';
```

containing the coefficients of its causal part.

The 2D polynomial of degree $(1,1)$

$$\begin{array}{rcl} R(z_1, z_2) & = & 5 + 4(z_1^{-1} + z_1) + 3(z_1 z_2^{-1} + z_1^{-1} z_2) \\ & & + 2(z_2^{-1} + z_2) + (z_1^{-1} z_2^{-1} + z_1 z_2) \end{array}$$

is represented through the 5-elements vector

```
r = [5 4 3 2 1]';
```

As the simplest example of POS3POLY use, let us compute the minimum value of the above 2D polymomial on the unit bicircle. This is equivalent to finding the largest μ such that $R(\mathbf{z}) - \mu \ge 0$. The corresponding CVX program is

```
cvx_begin
  variable mu;
  maximize mu
  subject to
    r - mu*eye(5,1) == sos_pol([1 1 1]);
cvx_end
```

The function `sos_pol` creates variables that are positive polynomials. (By default, they are trigonometric polynomials.) The first argument of `sos_pol` has the form $[\mathbf{n}\ \kappa]$, i.e. contains the degree of the polynomial and the size of its coefficients ($\kappa = 1$ for scalar coefficients). If only this argument is given, then the polynomial is globally positive. Since POS3POLY uses sum-of-squares relaxation, the polynomial is actually sum-of-squares of degree $\mathbf{n}$. Running the above program gives $\mu = -7$, which is the true minimum value.

In the case of matrix coefficients, where $\mathbf{R}_{\mathbf{k}} \in \mathbb{C}^{\kappa \times \kappa}$ replaces $r_{\mathbf{k}}$ in (3), the symmetry relation writes $\mathbf{R}_{-\mathbf{k}} = \mathbf{R}_{\mathbf{k}}^H$ (where the index ^{H} means transposition and complex conjugation). The polynomial is represented by a vector in the style of (4), with $r_{\mathbf{k}}$ replaced by $\mathrm{vec}(\mathbf{R}_{\mathbf{k}})$ (for $\mathbf{k} \ne \mathbf{0}$) where the operator vec concatenates the columns of $\mathbf{R}_{\mathbf{k}}$. Since $\mathbf{R}_{\mathbf{0}}$ is Hermitian, it is described by vectorizing (on columns) its lower triangular part.

3. POSITIVITY ON INTERVALS

More generally, POS3POLY handles polynomials that are positive on certain sets. In this section we present the case of 1D polynomials, where one can describe positivity on intervals or unions of disjoint intervals. Let such a union be

$$\mathscr{I} = \bigcup_{\ell=1}^{L} [\omega_{1\ell}\ \omega_{2\ell}]. \tag{5}$$

To describe a polynomial that is positive on $\mathscr{I}$ (and other possible properties), the function `sos_pol` needs a second argument, call it `ptype`, which is a structure. In this case, only one field of the structure is used, namely `ptype.int`, having the value

$$[\omega_{11}\ \omega_{21}\ \omega_{12}\ \omega_{22}\ \ldots\ \omega_{1L}\ \omega_{2L}].$$

(We will see later the use of other fields of `ptype`.)

As an introductory filter design example, let us consider the peak-constrained least-square (PCLS) optimization [1] of the linear phase filter of order $2n$ given by

$$z^{-n} H(z) = z^{-n} \sum_{k=-n}^{n} h_k z^{-k}, \;\; h_{-k} = h_k.$$

Assuming for simplicity that a lowpass filter is desired, the passband and stopband edges ω_p and ω_s are given, together with the passband and stopband error bounds γ_p and γ_s, respectively. The PCLS problem consists of minimizing the stopband energy E_s subject to peak constraints in passband and stopband:

$$\begin{array}{ll} \min & E_s \\ \text{s.t.} & |1 - H(e^{j\omega})| \le \gamma_p, \;\; \forall \omega \in [0, \omega_p] \\ & |H(e^{j\omega})| \le \gamma_s, \;\; \forall \omega \in [\omega_s, \pi] \end{array} \tag{6}$$

Denoting $\mathbf{h} = [h_0\ h_1 \ldots h_n]^T$, the stopband energy of the filter is

$$E_s = \mathbf{h}^T \tilde{\mathbf{C}} \mathbf{h}, \;\; \text{with } \tilde{\mathbf{C}} = \mathbf{P}^T \mathbf{C} \mathbf{P} \succeq 0, \tag{7}$$

where

$$\mathbf{P} = \begin{bmatrix} 0 & \mathbf{J}_n \\ 1 & 0 \\ 0 & \mathbf{I}_n \end{bmatrix}$$

($\mathbf{J}_n$ is the counteridentity matrix of size $n \times n$) and $\mathbf{C} = \text{Toeplitz}(c_0, c_1, \ldots, c_n) \succeq 0$, with

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

By showing explicitly the positive polynomials, extending to upper passband bound to the whole frequency domain and expressing the minimization of stopband energy as a second-order cone constraint, the problem (6) becomes

$$\begin{array}{ll} \min & \varepsilon \\ \text{s.t.} & \|\tilde{\mathbf{C}}^{\frac{1}{2}}\mathbf{h}\| \le \varepsilon \\ & 1+\gamma_p - H(e^{j\omega}) \ge 0, \ \ \forall \omega \\ & H(e^{j\omega}) - 1 + \gamma_p \ge 0, \ \ \forall \omega \in [0, \omega_p] \\ & \gamma_s - H(e^{j\omega}) \ge 0, \ \ \forall \omega \in [\omega_s, \pi] \\ & H(e^{j\omega}) + \gamma_s \ge 0, \ \ \forall \omega \in [\omega_s, \pi] \end{array} \tag{8}$$

Let us design a PCLS filter with $n = 34$, $\omega_p = 0.3\pi$, $\omega_s = 0.36\pi$, $\gamma_p = \gamma_s = 0.01$. We start by introducing the data:

```
n = 34;
wp = 0.3*pi;
ws = 0.36*pi;
gp = 0.01;
gs = 0.01;
```

Then, we build the matrix describing the stopband energy criterion

```
N = 2*n + 1;
c = zeros( N, 1 );
c( 1 ) = 1 - ws / pi;
c(2:N) = -sin((1:N-1)*ws) ./ (1:N-1)/pi;
P = [ zeros(n,1)  fliplr(eye(n)); ...
         1        zeros(1,n)    ; ...
      zeros(n,1)  eye(n)        ];
C = real( sqrtm(P' * toeplitz(c) * P) );
```

Next, we define the structures that characterize positivity in passband and stopband

```
ptypep.int = [ 0  wp ];
ptypes.int = [ ws pi ];
```

We are now ready to describe problem (8), which is simply

```
p1 = eye(n+1,1);
cvx_begin
 variable e;
 variable h(n+1);
 minimize e;
 subject to
  norm(C * h) <= e;
  (1+gp)*p1 - h == sos_pol([n 1]);
  h - (1-gp)*p1 == sos_pol([n 1],ptypep);
  gs*p1 - h     == sos_pol([n 1],ptypes);
  h + gs*p1     == sos_pol([n 1],ptypes);
cvx_end
```

The code is practically self-explanatory ! There is no need to know anything about the parameterization of trigonometric polynomials that are positive on an interval. Finally, we compose the full vector of coefficients of the linear-phase filter by

```
H = [h(end:-1:1); h(2:end)];
```

The frequency response of the designed filter is shown in Figure 1.

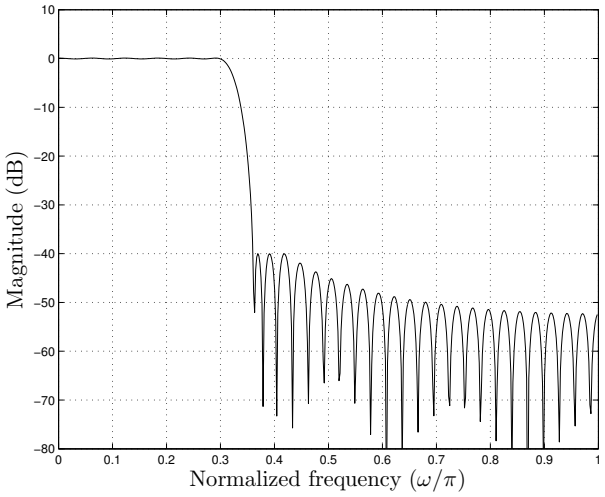

Figure 1: Frequency response of PCLS designed filter.

4. POSITIVITY ON DOMAINS

For multidimensional polynomials, POS3POLY allows the creation in CVX of polynomial variables that are positive on domains of the form

$$\mathscr{D} = \bigcup_{\ell=1}^{L} \mathscr{D}_\ell, \ \ \mathscr{D}_\ell = \left\{ \mathbf{z} \in \mathbb{T}^d \mid D_{i\ell}(\mathbf{z}) \ge 0, \ i = 1 : I_\ell \right\}, \tag{9}$$

where $D_{i\ell}(\mathbf{z})$ are given trigonometric polynomials. To characterize such domains, the structure `ptype` is again employed as second argument of `sos_pol`. Positivity domains are described by the subfields of `ptype.dom`, called `nunion`, `deg`, `coef` and `nc`.

The field `nunion` is the vector $[I_1\ I_2\ \ldots I_L]$, i.e. contains the *number* of polynomials defining each domain $\mathscr{D}_\ell$ appearing in the union (9). The other fields contain concatenated information on the degrees and coefficients of the polynomials $D_{i\ell}(\mathbf{z})$. There are two possibilities of describing the polynomials $D_{i\ell}(\mathbf{z})$. In the first, the field `deg` is a matrix with $I = I_1 + \ldots + I_L$ rows, each row containing the degree of a polynomial $D_{i\ell}(\mathbf{z})$. The field `coef` is a vector containing the concatenated coefficients of the polynomials, enumerated as in (4).

The second possibility assumes that the polynomials are sparse and so it is more efficient to give their nonzero coefficients. The field `nc` is a vector of length I, whose elements are the number of nonzero coefficients of the polynomials $D_{i\ell}(\mathbf{z})$; `deg` is a matrix whose rows contain the degrees of the monomials with nonzero coefficients and `coef` is a vector containing these coefficients.

As an illustration to the above, let us consider the problem of designing linear-phase 2-D FIR filters [5, Sect. 5.2] with diamond passband and stopband

$$\begin{array}{l} \mathscr{D}_p = \{\omega \in [-\pi,\pi]^2 \mid |\omega_1| + |\omega_2| \le \omega_p\}, \\ \mathscr{D}_s = \{\omega \in [-\pi,\pi]^2 \mid |\omega_1| + |\omega_2| \ge \omega_s\}, \end{array} \tag{10}$$

where $0 < \omega_p < \omega_s$ are given. The passband and stopband

can be described by polynomial positivity through

$$\begin{array}{l}\mathscr{D}_p = \left\{\mathbf{z}\in\mathbb{T}^2 \mid D_{p_1}(\mathbf{z})\geq 0,\ D_{p_2}(\mathbf{z})\geq 0,\ D_{p_3}(\mathbf{z})\geq 0\right\}\\ \mathscr{D}_s = \mathscr{D}_{s_1}\cup\mathscr{D}_{s_2}\cup\mathscr{D}_{s_3}\\ \mathscr{D}_{s_1} = \left\{\mathbf{z}\in\mathbb{T}^2 \mid D_{s_1}(\mathbf{z})\geq 0\right\}\\ \mathscr{D}_{s_2} = \left\{\mathbf{z}\in\mathbb{T}^2 \mid D_{s_2}(\mathbf{z})\geq 0\right\}\\ \mathscr{D}_{s_3} = \left\{\mathbf{z}\in\mathbb{T}^2 \mid D_{s_3}(\mathbf{z})\geq 0\right\}\end{array}$$

Denoting $a_p = \cos\omega_p$, $a_s = \cos\omega_s$, the polynomials appearing above are

$$\begin{array}{rcl}D_{p_1}(z_1,z_2) &=& (z_1z_2+z_1^{-1}z_2^{-1})/2-a_p\\ D_{p_2}(z_1,z_2) &=& (z_1^{-1}z_2+z_1z_2^{-1})/2-a_p\\ D_{p_3}(z_1,z_2) &=& (z_1+z_1^{-1})/2+(z_2+z_2^{-1})/2\\ D_{s_1}(z_1,z_2) &=& -(z_1z_2+z_1^{-1}z_2^{-1})/2+a_s\\ D_{s_2}(z_1,z_2) &=& -(z_1^{-1}z_2+z_1z_2^{-1})/2+a_s\\ D_{s_3}(z_1,z_2) &=& -(z_1+z_1^{-1})/2-(z_2+z_2^{-1})/2\end{array} \qquad (11)$$

For a polynomial that is positive on $\mathscr{D}_p$, the field `ptype` is described for POS3POLY via the following lines (`ap` stands for a_p)

```
ptypep.dom.deg = [1 1; 1 1; 1 1];
ptypep.dom.coef = [-ap 0   0   0   0.5 ...
                   -ap 0   0.5 0   0 ...
                    0  0.5 0   0.5 0];
```

Since there is no field `nunion`, it is assumed that $L=1$ in (9). The field `deg` says that there are $I_1=3$ polynomials whose positivity characterizes $\mathscr{D}_p$, all of degree $(1,1)$. The field `coef` contains the vector of concatenated coefficients of the first three polynomials from (11).

Similarly, for $\mathscr{D}_s$, the description is

```
ptypes.dom.nunion = [1 1 1];
ptypes.dom.deg  = [1 1; 1 1; 1 1];
ptypes.dom.coef = [as 0   0   0  -0.5 ...
                   as 0  -0.5 0   0 ...
                   0 -0.5 0  -0.5 0];
```

Now, the field `nunion` tells that $L=3$ and $I_1=I_2=I_3=1$ in (9). The fields `deg` and `coef` are filled using the degrees and coefficients of the last three polynomials from (11).

Assume that we want to design sparse 2D filters as in [9] (see an alternative approach in the 1D case in [2]), starting from the solution of the minimax optimization problem

$$\begin{array}{ll}\min & \gamma+\lambda\|\mathbf{r}\|_1\\ \text{s.t.} & R(\mathbf{z})\leq 1+\gamma,\ \forall\mathbf{z}\in\mathbb{T}^2\\ & R(\mathbf{z})\geq 1-\gamma,\ \forall\mathbf{z}\in\mathscr{D}_p\\ & R(\mathbf{z})\leq \gamma,\ \forall\mathbf{z}\in\mathscr{D}_s\\ & R(\mathbf{z})\geq -\gamma,\ \forall\mathbf{z}\in\mathscr{D}_s\end{array} \qquad (12)$$

The optimization criterion mixes the maximum error bound γ (with respect to the ideal response) and the 1-norm of the vector of coefficients of the filter, with the aim of promoting sparsity. The positive constant λ is a trade-off weight. Assume that we want to design a filter (3) with $\mathbf{n}=(8,8)$ (having a total of 289 coefficients, of which 145 distinct, this being the length of the vector $\mathbf{r}$), taking $\omega_p=0.5\pi$, $\omega_s=0.9\pi$, $\lambda=0.001$. We set first the design data

```
n   = [8 8];
ap  = cos(0.5*pi);
as  = cos(0.9*pi);
lam = 0.001;
```

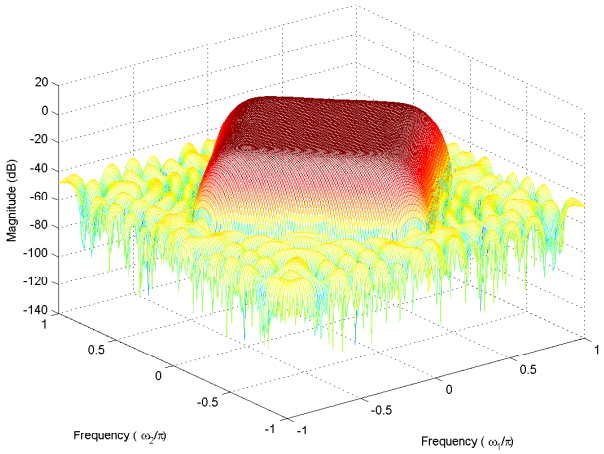

Figure 2: Frequency response of designed sparse 2-D filter.

Then, we set `ptypep` and `ptypes` as described above. For solving (12), we use the following sequence of instructions

```
M  = (1 + prod(2*n+1)) / 2;
p1 = eye(M,1);
cvx_begin
 variable g;
 variable r(M);
 minimize g + lam*norm(r,1);
 subject to
  (1+g)*p1 - r == sos_pol([n 1]);
  r - (1-g)*p1 == sos_pol([n 1], ptypep);
  g*p1 - r     == sos_pol([n 1], ptypes);
  r + g*p1     == sos_pol([n 1], ptypes);
cvx_end
```

Notice the similarity (in form) with the 1D problem from the previous section. The sum-of-squares parameterization [4] that replaces (and relaxes) the polynomial inequalities from (12) is hidden.

The solution of (12) can be used for further optimization. Assume that we want a filter with only N nonzero elements. We select the N largest coefficients of the solution of (12) and solve again (12) with $\lambda=0$ and forcing to zero the other coefficients; the above code can be easily modified to this purpose, so we skip the programming details. Hence, the optimal values of the selected coefficients are obtained.

For example, by taking $N=193$ (remind that the full filter has 289 coefficients), the optimal error is $\gamma=0.00559$. The frequency response is shown in Figure 2. By reoptimizing using the largest N coefficients of the optimal full filter (obtained by solving (12) with $\lambda=0$), we get $\gamma=0.00672$. So, it indeed makes sense to introduce the 1-norm term in the criterion, as shown also in [9], where all optimization is based on discretization and linear programming.

5. BOUNDED REAL LEMMA

A causal trigonometric polynomial with matrix coefficients has the form

$$\mathbf{H}(\mathbf{z}) = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{n}} \mathbf{H}_{\mathbf{k}}\mathbf{z}^{-\mathbf{k}}, \qquad (13)$$

with $\mathbf{H_k} \in \mathbb{C}^{\kappa_1 \times \kappa_2}$. In POS3POLY, it is described by the vector

$$\begin{array}{l}[\mathrm{vec}(\mathbf{H}_{0,\ldots,0}); \mathrm{vec}(\mathbf{H}_{1,0,\ldots,0}); \ldots; \mathrm{vec}(\mathbf{H}_{n_1,0,\ldots,0}); \\ \mathrm{vec}(\mathbf{H}_{0,1,\ldots,0}); \ldots; \mathrm{vec}(\mathbf{H}_{n_1,\ldots,n_d})].\end{array} \quad (14)$$

A Bounded Real Lemma (BRL) is an inequality

$$\|\mathbf{H}(\mathbf{z})\|_\infty \leq |A(\mathbf{z})|, \ \ \forall \mathbf{z} \in \mathscr{D}, \quad (15)$$

where $A(\mathbf{z})$ is a causal polynomial with degree $\mathbf{n}$, but with scalar coefficients, and $\mathscr{D}$ is defined as in (9). Such a BRL can be described in POS3POLY through a single vector variable that concatenates the vectors (14) for $\mathbf{H}(\mathbf{z})$ and (4) for the symmetric polynomial $R(\mathbf{z}) = A(\mathbf{z})A(\mathbf{z}^{-1})$. That such a variable corresponds to a BRL is marked by the presence of the field `ptype.brl`. The only subfield needed for working with CVX is `hsize`, which should contain the vector $[\kappa_1 \ \kappa_2]$ giving the size of the matrix coefficients.

For illustration, let us consider the minimax design of approximately linear phase filters via the optimization problem

$$\begin{array}{ll}\min & \gamma \\ \text{s.t.} & |H(\mathbf{z}) - z_1^{-\tau_1} z_2^{-\tau_2}| \leq \gamma, \ \ \forall \mathbf{z} \in \mathscr{D}_p \\ & |H(\mathbf{z})| \leq \gamma, \ \ \forall \mathbf{z} \in \mathscr{D}_s\end{array} \quad (16)$$

where the passband and stopband domains are defined as in (10) and $\tau_1, \tau_2 \in \mathbb{N}$. To solve this problem with POS3POLY, we first set `n`, `tau1` and `tau2` to the desired values of $\mathbf{n}$, τ_1, τ_2, respectively. We define `ptypep` and `ptypes` like in the previous section for describing the diamond-shaped domains and add

```
ptypep.brl.hsize = [1 1];
ptypes.brl.hsize = [1 1];
```

for specifying that we deal with BLR inequalities. Next, we define a few auxiliary variables:

```
Mh = prod(n+1);
Mr = (1 + prod(2*n+1)) / 2;
t  = zeros(Mh+Mr,1);
t(tau2*(n(1)+1)+tau1+1 ) = 1;
```

`Mh` and `Mr` are the lengths of the vectors of coefficients of $H(\mathbf{z})$ and $R(\mathbf{z}) = \gamma^2$; note that although in this case $R(\mathbf{z})$ is a constant, the corresponding coefficient vector is as long as for a polynomial of degree $\mathbf{n}$. The vector `t` represents the monomial $z_1^{-\tau_1} z_2^{-\tau_2}$. Denoting γ^2 by `g`, the problem (16) is solved by the code

```
cvx_begin
  variable g;
  variable hr(Mh+Mr);
  minimize g;
  subject to
    hr - t == sos_pol([n 1], ptypep);
    hr     == sos_pol([n 1], ptypes);
    hr(Mh+1:end) == g * eye(Mr,1);
cvx_end
```

The first two equality constraints correspond to the BRL inequalities from (16), while the third represents the equality $R(\mathbf{z}) = \gamma^2$.

6. CONCLUSIONS

The library POS3POLY lightens the programming burden in convex optimization with positive polynomials, as we hope that this paper made clear. The interested reader should go to www.schur.pub.ro/pos3poly for more information and the software itself. The user's guide contains many design examples, including the SeDuMi version of the FIR filter design problems treated here (also combining minimax and least-squares optimization), but also regarding the design of matrix filters, adjustable FIR filters, MIMO filters. Further work will be directed to implementing faster alternative parameterizations.

REFERENCES

[1] J.W. Adams and J.L. Sullivan. Peak-Constrained Least-Squares Optimization. *IEEE Trans. Sign. Proc.*, 46(2):306–321, Feb. 1998.

[2] T. Baran, D. Wei, and A.V. Oppenheim. Linear Programming Algorithms for Sparse Filter Design. *IEEE Trans. Signal Proc.*, 58(3):1605–1617, March 2010.

[3] T.N. Davidson. Enriching the Art of FIR Filter Design via Convex Optimization. *IEEE Signal Proc. Mag.*, 27(3):89–101, May 2010.

[4] B. Dumitrescu. Trigonometric Polynomials Positive on Frequency Domains and Applications to 2-D FIR Filter Design. *IEEE Trans. Signal Proc.*, 54(11):4282–4292, Nov. 2006.

[5] B. Dumitrescu. *Positive trigonometric polynomials and signal processing applications*. Springer, 2007.

[6] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming. http://cvxr.com/cvx, Sept. 2010.

[7] D. Henrion, J.B. Lasserre, and J. Löfberg. GloptiPoly 3: moments, optimization and semidefinite programming. *Optim. Meth. Software*, 24(4-5):761–779, 2009.

[8] J. Löfberg. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proc. CASCD*, Taipei, Taiwan, 2004.

[9] W.-S. Lu and T. Hinamoto. Two-dimensional digital filters with sparse coefficients. *Multidim. Syst. Sign. Process.*, 2010. to appear (online at http://dx.doi.org/10.1007/s11045-010-0129-9).

[10] S. Prajna, A. Papachristodoulou, and P.A. Parrilo. SOSTOOLS: Sum of Squares Optimization Toolbox for Matlab, 2002. available from http://www.cds.caltech.edu/sostools.

[11] J.F. Sturm. Using SeDuMi 1.02, a Matlab Toolbox for Optimization over Symmetric Cones. *Optimization Methods and Software*, 11:625–653, 1999. http://sedumi.ie.lehigh.edu.

[12] K.C. Toh, M.J. Todd, and R.H. Tütüncü. SDPT3 – a Matlab Software Package for Semidefinite Programming. http://www.math.cmu.edu/∼reha/sdpt3.html.