Filter Design With Low Complexity Coefficients

Joëlle Skaf and Stephen P. Boyd, *Fellow, IEEE*

***Abstract*—We introduce a heuristic for designing filters that have low complexity coefficients, as measured by the total number of nonzeros digits in the binary or canonic signed digit (CSD) representations of the filter coefficients, while still meeting a set of design specifications, such as limits on frequency response magnitude, phase, and group delay. Numerical examples show that the method is able to attain very low complexity designs with only modest relaxation of the specifications.**

***Index Terms*—Coefficient truncation, filter design.**

I. Introduction

We consider a discrete-time filter defined by its transfer function $H_\theta : \mathbf{C} \rightarrow \mathbf{C}$, where $\mathbf{C}$ is the set of complex numbers, and $\theta \in \mathbf{R}^p$ is the vector of p (real) coefficients used to parametrize the transfer function. We refer to (the coefficients of) θ as the *design parameters* or *coefficients* in the filter. For example, when the filter is an infinite impulse response (IIR) filter implemented in direct form, H_θ is a rational function with real coefficients, and θ_i are the coefficients in its numerator and denominator.

We are given a *nominal filter design*, described by the coefficient vector $\theta^{\rm nom}$, and a set of acceptable filter designs $\mathcal{H}$, which is the set of transfer functions that satisfy our design or performance requirements. We define

$$\mathcal{C} = \{\theta \in \mathbf{R}^p | H_\theta \in \mathcal{H}\}$$

as the set of coefficient vectors that correspond to acceptable designs. We assume that the nominal filter meets the performance specifications, i.e., $\theta^{\rm nom} \in \mathcal{C}$. Our goal is to find $\theta \in \mathcal{C}$ that has lowest (or at least, low) complexity.

The *complexity* of a vector of filter coefficients θ is measured by the function $\Phi : \mathbf{R}^p \rightarrow \mathbf{R}$

$$\Phi(\theta) = \sum_{i=1}^{p} \phi(\theta_i)$$

where $\phi(\theta_i)$ gives the complexity of the ith coefficient of θ. In this paper, we will focus on two complexity measures, even though the algorithms we describe are more general. In the first measure, we take $\phi(\theta_i)$ to be $\phi_{\rm bin}(\theta_i)$, the number of 1 s in the binary expansion of the coefficient θ_i. In this case, $\Phi(\theta)$ gives the total number of 1 s in the filter coefficients, and will be denoted $\Phi_{\rm bin}(\theta)$. In the second complexity measure, we take $\phi(\theta_i)$ to be $\phi_{\rm csd}(\theta_i)$, the number of nonzero digits in the *canonical signed digit* (CSD) representation of θ_i [1], [2], which we will describe in more detail in Section II-A. In this case, we denote the complexity measure as $\Phi_{\rm csd}(\theta)$.

We can pose our filter design problem as the optimization problem

$$\begin{array}{ll} \text{minimize} & \Phi(\theta) \\ \text{subject to} & \theta \in \mathcal{C} \end{array} \qquad (1)$$

with variable $\theta \in \mathbf{R}^p$. The filter design problem (1) is in general very difficult to solve. With the complexity measure $\Phi_{\rm bin}$, it can be cast as a combinatorial optimization problem, with the binary expansions of the coefficients as Boolean (i.e., {0,1}) variables. In the case of CSD complexity, it can be cast as a combinatorial optimization problem with the CSD digits taking ternary values in $\{-1, 0, 1\}$ as discrete variables. In general it is very difficult to solve this problem exactly (i.e., globally), even for a relatively small number of coefficients. But finding the globally optimal solution is not crucial; it is enough to find a set of filter coefficients with low (if not lowest) complexity.

In this paper we describe a greedy randomized heuristic algorithm for the filter design problem (1). Our method starts from the nominal design and greedily truncates individual coefficients sequentially, in random order, while guaranteeing performance, i.e., maintaining $\theta \in \mathcal{C}$. We run this algorithm a few times, taking the best filter coefficients found (i.e., θ with least value of $\Phi(\theta)$) as our final design. Examples show that our method typically produces aggressively truncated filter designs, with far lower complexity than the nominal design.

The idea of truncating or simplifying filter coefficients in return for a small degradation in performance goes back a long way, at least to [3], [4]. Coefficient truncation subsequently appeared in other fields like speech processing [5] and control [6]. Several methods have been proposed for coefficient truncation: exhaustive search over possible truncated coefficients [3], successive truncation of coefficients and reoptimization over remaining ones [4], [7], local bivariate search around the scaled and truncated coefficients [8], tree-traversal techniques for truncated coefficients organized in a tree according to their complexity [9], [10], coefficient quantization using information-theoretic bounds [11], weighted least-squares [12], simulated annealing [13], [14], genetic algorithms [15]–[17], Tabu search [18], design of optimal filter realizations that minimize coefficient complexity [13]. Other approaches have formulated the problem as a nonlinear discrete optimization problem [19], or

Manuscript received November 30, 2007; revised January 16, 2008. This work was supported in part by the Focus Center Research Program Center for Circuit and System Solutions Award #2003-CT-888, by JPL Award #I291856, by Army Award #W911NF-07-1-0029, by NSF Award #ECS-0423905, by NSF Award #0529426, by DARPA Award #N66001-06-C-2021, by NASA Award #NNX07AEIIA, by AFOSR Award #FA9550-06-1-0514, and by AFOSR Award #FA9550-06-1-0312. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Manuel Davy.

The authors are with the Information Systems Lab, Electrical Engineering Department, Stanford University, Stanford, CA 94305 USA (e-mail: jskaf@stanford.edu; boyd@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2008.919386

have used integer programming techniques [20]–[22]. Related research explores filter realization forms that are relatively insensitive to (small) changes in the coefficients (i.e., as occurs when they are truncated); see, e.g., [23]. A survey of methods for quantizing lifting coefficients for wavelet filters can be found in [24].

We should mention one difference between our approach and essentially all of the work cited above. In the work cited above, the designer starts with a *reference design*, i.e., a set of coefficients, of infinite complexity, that gives the desired performance. Then some budget of total complexity is decided on. Then the designer searches for a set of coefficients, over the grid of coefficients that satisfy the total complexity budget, that minimizes some deviation from the reference design. In contrast, we maintain a design that satisfies the performance specifications; its complexity decreases as the algorithm proceeds.

II. Coefficient Representations and Complexity Measures

In this section, we describe two number representation systems, binary and CSD, and the associated complexity measures.

A. Binary and CSD Representation

We will consider numbers $z \in \mathbf{R}$ that can be represented in the form

$$z = s \sum_{i=-L}^{R} b_i 2^{-i}$$

where L and R are nonnegative integers. In the case of the binary representation, s is the sign of z, with $s \in \{-1, 0, 1\}$ ($s = 0$ when $z = 0$); $b_i \in \{0, 1\}$ are the bits in the binary expansion of z. In this case, $L+1$ and R are the number of bits in the integer and fractional part of z, respectively.

In the case of CSD representation, we have $s = 1$, and $b_i \in \{-1, 0, 1\}$, with $b_i b_{i+1} = 0$ for $i = -L, \ldots, R-1$ (i.e., the representation does not contain any consecutive nonzero digits.) In this case, $L+1$ and R are the number of digits in the integer and fractional part of z, respectively. The CSD representation can be derived recursively from the binary representation of a number, by noting that k ones in succession, in a binary expansion, is equal to $2^k - 1$, which can be written using two nonzero digits and $k-1$ zeros in the CSD representation. As a simple example, we have $3.75 = 2^1 + 2^0 + 2^{-1} + 2^{-2}$ (in its binary representation), which is equal to $2^2 - 2^{-2}$, its CSD representation.

We will assume that the nominal filter coefficients can be represented in binary or CSD form, with a given value of R.

B. Complexity Measures

While it is possible to consider a variety of complexity measures, we will focus on two: The number of nonzero bits (digits) in the binary (CSD) representation of the coefficients. We define

$$\phi_{\mathrm{bin}}(z) = \sum_{i=-L}^{R} b_i$$

where b_i are the bits in the binary expansion of z. This complexity measure is the number of adders needed to implement multiplication by z using a shift and add method. We will denote the associated complexity measure of the full coefficient vector as $\Phi_{\mathrm{bin}}(\theta)$, and refer to it as the *binary complexity* of θ.

The corresponding complexity measure for the CSD representation is the number of nonzero digits in the CSD representation of z,

$$\phi_{\mathrm{csd}}(z) = \sum_{i=-L}^{R} |b_i|$$

where b_i are the digits in the CSD representation of z. The associated complexity measure of the full coefficient vector will be denoted $\Phi_{\mathrm{csd}}(\theta)$. We will refer to this as the *CSD complexity* of θ.

III. Filter Parametrization

We assume that the transfer function $H_\theta(z)$ and its derivative with respect to z, $H'_\theta(z)$, are continuous functions of θ for z on the unit circle. (In almost all cases of interest, $H_\theta(z)$ has a much more special form: it is typically rational in z and θ, and for each θ_i, bilinear in θ_i. But we will not exploit these properties.) While it is often the case, we do *not* assume that the parametrization is unique, i.e., that different filter coefficient vectors give rise to different transfer functions. In other words, we allow the filter to be over parametrized.

We give some standard examples below; for more details on each of these filter structures, see, e.g., [25].

FIR filter: A finite impulse response (FIR) filter has transfer function

$$H_\theta(z) = \sum_{n=0}^{N} b_n z^{-n}.$$

Here we have filter coefficient vector $\theta = (b_0, \ldots, b_N) \in \mathbf{R}^p$, with $p = N+1$.

IIR filter in direct form I or II: The transfer function of an IIR filter is

$$H_\theta(z) = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{1 + \sum_{n=1}^{N-1} a_n z^{-n}}$$

Here we have $\theta = (b_0, \ldots, b_{M-1}, a_1, \ldots, a_{N-1}) \in \mathbf{R}^p$, with $p = M + N - 1$.

IIR filter in cascade form: In cascade form, the transfer function is written as a product of second-order sections, i.e.

$$H_\theta(z) = \prod_{l=1}^{L} \frac{b_{0L} + b_{1l} z^{-1} + b_{2l} z^{-2}}{1 + a_{1l} z^{-1} + a_{2l} z^{-2}}.$$

Here we have

$$\theta = (b_{01}, b_{11}, b_{21}, a_{11}, a_{21}, \ldots, b_{0L}, b_{1L}, b_{2L}, a_{1L}, a_{2L})$$

and $\theta \in \mathbf{R}^p$ with $p = 5L$.

IIR filter in parallel form: In this form the transfer function is written as a sum of second-order sections

$$H_\theta(z) = \sum_{l=1}^{L} \frac{b_{0l} + b_{1l}z^{-1}}{1 + a_{1l}z^{-1} + a_{2l}z^{-2}}.$$

Here we have

$$\theta = (b_{01}, b_{11}, b_{21}, a_{11}, a_{21}, \dots, b_{0L}, b_{1L}, b_{2L}, a_{1L}, a_{2L})$$

and $\theta \in \mathbf{R}^p$ where $p = 4L$.

IIR filter in lattice-ladder form: An IIR filter in lattice-ladder form, with lattice coefficients k_n, $n = 1, \dots, N$, and ladder coefficients c_m, $m = 0, \dots, M$, where $M \le N$, has the state-space form

$$x(t) = Ax(t-1) + bu(t), \qquad y(t) = cx(t),$$

where $u(t) \in \mathbf{R}$ is the input, $y(t) \in \mathbf{R}$ is the output, $x(t) \in \mathbf{R}^{N+1}$ is the state and

$$A = \begin{bmatrix} -k_1 & -k_2 & \cdots & \cdot & -k_N & 0 \\ (1-k_1)^2 & -k_1k_2 & -k_1k_3 & \cdot & -k_1k_N & 0 \\ 0 & \left(1-k_2^2\right) & -k_2k_3 & \cdot & -k_2k_N & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \left(1-k_N^2\right) & 0 \end{bmatrix},$$

$$b = \begin{bmatrix} 1 \\ k_1 \\ k_2 \\ \vdots \\ k_N \end{bmatrix}, c^T = \begin{bmatrix} c_0 \\ \vdots \\ c_M \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The transfer function is $H_\theta(z) = c(zI - A)^{-1}b$. Here we have $\theta = (k_1, k_2, \dots, k_N, c_0, c_1, \dots, c_M) \in \mathbf{R}^p$, with $p = M + N + 1$.

IV. Filter Specifications

Our algorithm requires only the ability to check whether a given filter coefficient vector θ is acceptable, i.e., whether $\theta \in \mathcal{C}$; the details of this procedure do not matter.

We list some simple examples of specifications, and methods for checking that they hold. One basic requirement is that H_θ should be stable, which is easily checked by computing its poles, or the eigenvalues of the dynamics matrix in a state-space realization.

Performance specifications are typically given as a collection of constraints on the transfer function $H_\theta(z)$ and $H'_\theta(z)$ (its derivative with respect to z), evaluated on the unit circle, i.e., $z = e^{-i\omega}$ with $\omega \in [0, 2\pi]$. Typical specifications are lower and upper bounds on the magnitude, phase, and group delay

$$\begin{aligned} L_{\text{mag}}(\omega) &\le \left|H_\theta(e^{-i\omega})\right| \le U_{\text{mag}}(\omega), \\ L_{\text{ph}}(\omega) &\le \angle H_\theta(e^{-i\omega}) \le U_{\text{ph}}(\omega), \\ L_{\text{gd}}(\omega) &\le G_\theta(\omega) \le U_{\text{gd}}(\omega) \end{aligned} \qquad (2)$$

where G_θ is the group delay of the filter

$$\begin{aligned} G(\omega) &= -\frac{d}{d\omega}\angle H_\theta(e^{-i\omega}) \\ &= -\frac{d}{d\omega}\Im \log\left(H_\theta(e^{-i\omega})\right) \\ &= \Re\left\{e^{-i\omega}\frac{H'(e^{-i\omega})}{H(e^{-i\omega})}\right\}. \end{aligned}$$

The lower and upper bounds L_{mag}, U_{mag}, L_{ph}, U_{ph}, L_{gd}, and U_{gd} are given functions from $[0, \pi]$ into $\mathbf{R}$. The inequalities in (2) must hold for all $\omega \in [0, \pi]$. They can be verified in practice by appropriately fine sampling, i.e., by checking that they hold (possibly with some margin) at a finite number of frequencies.

V. The Algorithm

A. High Level Algorithm

Our algorithm is a greedy randomized heuristic, which is initialized with the nominal design, which we assume has finite (but possibly large) complexity. (This can be ensured simply rounding the coefficients of the nominal design to, say, $R = 40$ bits.) At each step an index $i \in \{1, \dots, p\}$ is chosen, and all filter coefficients except θ_i are fixed. We use the procedure **adjust** (described below) to find a value of θ_i with (possibly) lower complexity, while still satisfying the design specifications. We have experimented with various methods for choosing the index i in each step, and found the best results by organizing the algorithm into passes, each of which involves updating each coefficient once; in each pass, the ordering of the indices is chosen randomly. The algorithm stops when the filter coefficient vector θ does not change over one pass.

At the highest level, the algorithm has the following form:

```
$\theta := \theta^{\text{nom}}$
repeat
    $\theta^{\text{prev}} := \theta$
    choose a permutation $\pi$ of $(1, \dots, p)$
    for $i = 1$ to $p$
        $j := \pi(i)$
        $\theta_j :=$ adjust$(\theta, j)$
until $\Phi(\theta) = \Phi(\theta^{\text{prev}})$
```

Since the algorithm is random, it can and does converge to different points in different runs. It can be run several times, taking the best filter coefficient vector found as our final choice.

The algorithm is guaranteed to converge in a finite number of steps, since in each pass for which $\theta \ne \theta_{\text{prev}}$ (i.e., we actually change some coefficient), the total complexity decreases: $\Phi(\theta) < \Phi(\theta_{\text{prev}})$. (Since Φ takes on nonnegative integer values, it cannot decrease more than $\Phi(\theta^{\text{nom}})$ times.)

We now describe the procedure **adjust**, which takes as input a coefficient vector $\theta \in \mathcal{C}$ and an index $i \in \{1, \dots, p\}$, and returns a value $\hat{\theta}_i$ with $\phi(\hat{\theta}_i) \le \phi(\theta_i)$ that satisfies $\hat{\theta} \in \mathcal{C}$, where

$\hat{\theta} = (\theta_1, \dots, \theta_{i-1}, \hat{\theta}_i, \theta_{i+1}, \dots, \theta_p)$. The procedure $\mathbf{adjust}$ is as follows.

```
**if** $\theta_i = 0$,
    $\hat{\theta}_i := \theta_i$; **exit**
$l := -2|\theta_i|, u := 2|\theta_i|$
**repeat**
    $\hat{\theta}_i := \mathbf{trunc}(\theta_i, l, u)$
    **if** $\hat{\theta} \in \mathcal{C}$, **exit**
    **else**
        **if** $\hat{\theta}_i > \theta_i, u := \hat{\theta}_i$
        **else** $l := \hat{\theta}_i$
```

If $\theta_i = 0$, it already has the minimum possible complexity (zero) so we leave it as it is. Otherwise, we maintain an open interval (l, u), which will contain $\hat{\theta}$, our tentative guess at a suitable value, and θ_i (the current value, which satisfies $\theta \in \mathcal{C}$). The procedure $\mathbf{trunc}$ returns a number in the open interval (l, u) whose complexity is less than or equal to $\phi(\theta_i)$; it will be described in detail later. By definition, we have $\phi(\mathbf{trunc}(\theta_i, l, u)) \leq \phi(\theta_i)$. We check if $\hat{\theta}_i \in \mathcal{C}$, and if so, we quit. Otherwise we change either l or u to be $\hat{\theta}_i$, so we once again have an open interval that contains θ_i. This loop can execute only a finite number of times; one simple upper bound is the number of numbers with precision 2^{-R} in $(-2|\theta_i|, 2|\theta_i|)$ that satisfy $\phi(z) \leq \phi(\theta_i)$.

B. Truncation Methods

We describe how to compute $z = \mathbf{trunc}(x, l, u)$ for the complexity measures ϕ_{bin} and ϕ_{csd}. Here $x, l, u \in \mathbf{R}$ have precision 2^{-R} and satisfy $l < x < u$.

Given a number y written as $s_y \sum_{i=-L}^{R} y_i 2^{-i}$, we introduce the notation $[y]_k$ to denote

$$[y]_k = s_y \sum_{i=-L}^{k} y_i 2^{-i}$$

where k is an integer satisfying $-L \leq k \leq R$. (Informally, $[y]_k$ is y, truncated to k bits or digits.) It is easy to show that, for j, k integers satisfying $-L \leq j \leq k \leq R$,

$$\phi_{\text{bin}}([y]_j) \leq \phi_{\text{bin}}([y]_k), \qquad \phi_{\text{csd}}([y]_j) \leq \phi_{\text{csd}}([y]_k).$$

It is also obvious that $[y]_R = y$.

We assume that l and u are represented as

$$l = s_l \sum_{i=-L}^{R} l_i 2^{-i}, \qquad u = s_u \sum_{i=-L}^{R} u_i 2^{-i},$$

respectively.

We describe a general procedure $\mathbf{trunc}$ that returns a number in (l, u) with complexity less than or equal to $\phi(x)$. This procedure can be used if the complexity measure is ϕ_{bin} or ϕ. The number $z = \mathbf{trunc}(x, l, u)$ can be found as follows.

```
**if** $s_u s_l = -1$
    $z := 0$, **exit**
**for** $i = -L$ **to** $R - 1$
    $z := ([l]_i + [u]_i)/2$
    **if** $l < z < u$ **and** $\phi(z) \leq \phi(x)$
        **exit**
```

The procedure $\mathbf{trunc}$ starts by examining whether l and u have opposite signs: if they do, it returns 0, which is obviously the number in (l, u) with smallest complexity. Otherwise, for $i = -L, \dots, R$, it computes the numbers $[l]_i$ and $[u]_i$ and set z to be their average. Note that z is the number with the smallest number of 1 s in $([l]_i, [u]_i)$. The procedure $\mathbf{trunc}$ then exits if z lies in (l, u) and has complexity less than or equal to that of x. If $i = R - 1$ is reached, $\mathbf{trunc}$ must (and does) return x since, in that case, it is the only number with precision 2^{-R} in (l, u) and therefore the one with smallest complexity in that interval. (Note that we cannot have $[l]_i = [u]_i$ for $i = -L, \dots, R - 1$ because that would imply that $u - l = 2^{-R}$ and, therefore, that $x = l$ or $x = u$, which is not allowed).

To illustrate how the $\mathbf{trunc}$ procedure works, we consider a numerical example. Let $x = 1.248046875$, $l = 1.224609375$, and $u = 1.27734375$. We take $L = 0$ and $R = 9$. In the binary representation case, we have

$$\begin{aligned} l &= 2^0 + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} \\ x &= 2^0 + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-8} + 2^{-9} \\ u &= 2^0 + 2^{-2} + 2^{-6} + 2^{-7} + 2^{-8}. \end{aligned}$$

Since l and u are of the same sign, $\mathbf{trunc}$ does not return 0. It instead calculates the following quantities shown in the equation at the bottom of the page. For $i = 0,1,2,3,4$, z does not lie in (l, u). However, for $i = 5$, it does. Since $\phi_{\text{bin}}(z) = 5 < \phi_{\text{bin}}(x) = 8$, $\mathbf{trunc}$ returns $z = 2^0 + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} = 1.234375$.

i	$[l]_i$	$[u]_i$	z
0	2^0	2^0	2^0
1	2^0	2^0	2^0
2	2^0	$2^0 + 2^{-2}$	$2^0 + 2^{-3}$
3	$2^0 + 2^{-3}$	$2^0 + 2^{-2}$	$2^0 + 2^{-3} + 2^{-4}$
4	$2^0 + 2^{-3} + 2^4$	$2^0 + 2^{-2}$	$2^0 + 2^{-3} + 2^{-4} + 2^{-5}$
5	$2^0 + 2^{-3} + 2^{-4} + 2^{-5}$	$2^0 + 2^{-2}$	$2^0 + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6}$

In the CSD representation case, we have

$$\begin{aligned} l &= 2^0 + 2^{-2} - 2^{-5} + 2^{-7} - 2^{-9} \\ x &= 2^0 + 2^{-2} - 2^{-9} \\ u &= 2^0 + 2^{-2} + 2^{-5} - 2^{-8}. \end{aligned}$$

Since l and u are of the same sign, **trunc** does not return 0. It instead calculates the following quantities:

i	$[l]_i$	$[u]_i$	z
0	2^0	2^0	2^0
1	2^0	2^0	2^0
2	$2^0 + 2^{-2}$	$2^0 + 2^{-2}$	$2^0 + 2^{-2}$

.

For $i = 0, 1$, z does not lie in (l, u). However, for $i = 2$, it does. Since $\phi_{\text{csd}}(z) = 2 < \phi_{\text{csd}}(x) = 3$, **trunc** returns $z = 2^0 + 2^{-2} = 1.25$.

C. Complexity Analysis

We can give a worst-case complexity analysis of our algorithm for the cases where the binary complexity measure ϕ_{bin} is used and the CSD complexity measure ϕ_{csd} is used. As aforementioned, the main loop can be executed at most $\Phi(\theta^{\text{nom}})$ times. Each pass over the main loop calls **adjust** p times. We will show that, when ϕ_{bin} is used, the number of passes over the loop in **adjust** is at most $(K+1)^2/2$, where $K = L_{\max} + R_{\max} + 1$. Here $L_{\max} + 1$ is the maximum number of bits in the integer part of the initial coefficients, and $R_{\max}$ is the maximum number of bits in the fractional part of the initial coefficients. In case ϕ_{csd} is used, the number of passes over the loop in **adjust** is at most $(K+1)^2$. This means that the oracle for testing membership in $\mathcal{C}$ (i.e., testing that a given filter design satisfies the specifications) is called at most $p\Phi_{\text{bin}}(\theta^{\text{nom}})(K+1)^2/2$ when the binary complexity measure is used and at most $p\Phi_{\text{csd}}(\theta^{\text{nom}})(K+1)^2$ when the CSD complexity measure is used. In both cases, our proposed algorithm therefore has polynomial-time complexity in the problem data (i.e., the filter coefficients).

We will now derive the upper bound on the number of passes over the loop in **adjust** in the case where ϕ_{bin} is the complexity measure. Without loss of generality, let **adjust** be called on a coefficient $\theta_i > 0$ which can be written as $\theta_i = s_i \sum_{j=-L}^{R} b_{ij} 2^{-j}$, where $b_i \in \{0, 1\}$. Since $\theta_i > 0$, we have $R + L > 0$. The worst-case scenario occurs when $\textbf{adjust}(\theta, i) = \theta_i$, i.e., when **adjust** fails to find a number of lesser complexity than θ_i, in the interval $(-2|\theta_i|, 2|\theta_i|)$, that corresponds to an acceptable filter design. The following would occur: in the first pass over the loop in **adjust**, 0 is returned by **trunc** and fails to induce an acceptable filter design. Next, the algorithm would go over all numbers z in $(0, 2\theta_i)$ with $\phi_{\text{bin}}(z) = 1$, and fail to find an acceptable filter design: there are $K = L + R + 1$ such numbers in $(0, 2\theta_i)$. Let z_1 be the largest number in $(0, \theta_i)$ with $\phi_{\text{bin}}(z_1) = 1$. Note that z_1 and θ_i share a 1 in the Lth position of their binary expansion. This implies that there are $K - 1$ numbers z in $(z_1, 2z_1)$ with $\phi_{\text{bin}}(z) = 2$. Let z_2 be the largest number in (z_1, θ_i) with $\phi_{\text{bin}}(z_2) = 2$. The same analysis can be carried out to show that the algorithm will then look at $K - 2$ numbers of complexity 3 in $(z_2, 2z_2)$, and so on until the algorithm stops. This happens when it looks for the lowest complexity coefficient in $(z_{K-1}, 2z_{K-1})$: this number is θ_i. We can therefore conclude that the number of passes over the loop in **adjust** is no more than W where

$$\begin{aligned} W &= 1 + K + (K-1) + (K-2) + \cdots + 1 \\ &= 1 + K(K+1)/2 \\ &\leq (K+1)^2/2 \end{aligned}$$

(since $K > 1$).

We follow a similar methodology to derive the upper bound on the number of passes over the loop in **adjust** in the case where ϕ_{csd} is the complexity measure. Without loss of generality, let **adjust** be called on a coefficient $\theta_i > 0$ which can be written as $\theta_i = s_i \sum_{j=-L}^{R} b_{ij} 2^{-j}$, where $b_i \in \{-1, 0, 1\}$ and $b_i b_{i+1} = 0$. The worst-case scenario occurs when $\textbf{adjust}(\theta, i) = \theta_i$, i.e., when **adjust** fails to find a number of lesser complexity than θ_i, in the interval $(-2|\theta_i|, 2|\theta_i|)$, that corresponds to an acceptable filter design. The following would occur: in the first pass over the loop in **adjust**, 0 is returned by **trunc** and fails to induce an acceptable filter design. Next, the algorithm would go over all numbers z in $(0, 2\theta_i)$ with $\phi_{\text{csd}}(z) = 1$, and fail to find an acceptable filter design: there are $K = L + R + 1$ such numbers in $(0, 2\theta_i)$. Let $z_1 = [\theta_i]_{-L}$. Note that θ_i and z_1 share the same digit b_{-L}. The coefficient θ_i belongs to the interval $(z_1, 2z_1)$ if $z_1 < \theta_i$, or to the interval $(1/2z_1, z_1)$ otherwise. Let I_1 be the interval in which θ_i falls. There are at most $2K - 3$ numbers z in I_1 with $\phi_{\text{csd}}(z) = 2$. Let $z_2 = [\theta_i]_{-L+1}$. The same analysis is carried out to show that the algorithm will then look at at most $2(K-1) - 3$ numbers of complexity 3 in I_2. (Here I_2 is either $(z_2, 2z_2)$ or $(1/2z_2, z_2)$). This continues until the algorithm stops, which happens when it looks for the lowest complexity coefficient in I_{R-1}: this number is θ_i. We can, therefore, conclude that the number of passes over the loop in **adjust** is no more than V where

$$\begin{aligned} V &= 1 + K + 2K - 3 + 2(K-1) - 3 + \ldots + 1 \\ &= 1 + K + (K-1)^2 \\ &\leq (K+1)^2 \end{aligned}$$

(Since $K > 1$).

In numerical experiments, we have found that the actual number of steps required by our algorithm is always far smaller than the upper bounds derived in this section, i.e., $p\Phi_{\text{bin}}(\theta^{\text{nom}})(K+1)^2/2$ when ϕ_{bin} is used and $p\Phi_{\text{csd}}(\theta^{\text{nom}})(K+1)^2$ when ϕ_{csd} is used.

VI. Examples

A. IIR Filter Design

For each of our examples we use the same set of filter specifications, with frequency response magnitude bounds

$$L_{\text{mag}}(\omega) = \begin{cases} 10^{-1/20} & \omega \in [0, 0.12\pi] \\ 0 & \text{otherwise} \end{cases}$$

$$U_{\text{mag}}(\omega) = \begin{cases} 10^{1/20} & \omega \in [0, 0.12\pi] \\ 10^{-35/20} & \omega \in [0.24\pi, \pi] \\ \infty & \text{otherwise.} \end{cases}$$

TABLE I
NOMINAL AND TRUNCATED FILTER COEFFICIENTS FOR DIRECT FORM IMPLEMENTATION

	Nominal	Binary truncated	CSD truncated
b_0	0.018814	$2^{-6}+2^{-8}$	$2^{-6}+2^{-8}$ -2^{-11}
b_1	-0.013956	$-(2^{-7}+2^{-8}+2^{-9})$	$-2^{-6}+2^{-9}$
b_2	0.004505	2^{-8}	2^{-8}
b_3	0.010579	$2^{-7}+2^{-9}+2^{-11}$	$2^{-6}-2^{-8}$ -2^{-10}
a_1	-2.602456	$-(2^{1}+2^{-1}+2^{-4}$ $+2^{-5}+2^{-7})$	$-2^{1}-2^{-1}-2^{-3}$ $+2^{-5}-2^{-7}$
a_2	2.361682	$2^{1}+2^{-2}+2^{-4}$ $+2^{-5}+2^{-6}+2^{-9}$	$2^{1}+2^{-1}-2^{-3}$ $-2^{-6}+2^{-9}$
a_3	-0.741026	$-(2^{-1}+2^{-3}+2^{-4}$ $+2^{-5}+2^{-6}+2^{-8}$ $+2^{-9}+2^{-10})$	$-2^{0}+2^{-2}+2^{-7}$ $+2^{-10}$

This corresponds to a low-pass filter, with pass band $[0, 0.12\pi]$, stopband $[0.24\pi, \pi]$, maximum passband ripple ± 1 dB, and minimum stopband attenuation 35 dB.

The nominal filter has the form

$$H_{\rm nom}(z) = \frac{b_0 + b_1 z + b_2 z^2 + b_3 z^3}{1 + a_1 z + a_2 z^2 + a_3 z^3},$$

with $\theta = (b_0, b_1, b_2, b_3, a_1, a_2, a_3)$. The nominal filter coefficients are found using the spectral factorization method [26], [27], by maximizing the minimum stopband attenuation subject to maximum passband ripple ± 0.8 dB. This problem can be cast as a convex optimization problem, so we can easily obtain the globally optimal solution [28]. This globally optimal solution achieves a minimum stopband attenuation of 37.5 dB. Thus, our specifications involve relaxing the passband ripple from ± 0.8 to ± 1 dB, and relaxing the stopband attenuation from 37.5 to 35 dB. In particular, our set $\mathcal{H}$ is quite small, since our specifications are quite close to globally optimal.

To find the nominal coefficients for each of our examples, we first realize our nominal filter in the given form, and take $R = 40$ bits (i.e., round the resulting coefficients to 40 bits in their fractional part).

IIR filter in direct form: The nominal filter coefficients have binary complexity $\Phi_{\rm bin}(\theta_{\rm nom}) = 179$. We run our algorithm 100 times. The best result has binary complexity $\Phi_{\rm bin}(\theta) = 28$, which is around 4.3 1 s per coefficient; see Table I. The average binary complexity of the designs returned by the algorithm over these 100 random runs is around 32 1 s.

The nominal filter coefficients have CSD complexity $\Phi_{\rm csd}(\theta_{\rm nom}) = 99$. We run our algorithm 100 times. The best result has complexity $\Phi_{\rm csd}(\theta) = 23$, which is around 3.3 nonzero digits per coefficient; see Table I. The average CSD complexity of the designs returned by the algorithm over these 100 random runs is around 27 nonzero digits.

Fig. 1 shows the frequency response magnitude of the nominal filter and the filters with binary and CSD truncated coefficients.

IIR filter in cascade form: The nominal transfer function is expressed in cascade form with $L = 2$ second-order sections

$$H(z) = \prod_{l=1}^{2} \frac{b_{0l} + b_{1l} z^{-1} + b_{2l} z^{-2}}{1 + a_{1l} z^{-1} + a_{2l} z^{-2}}$$

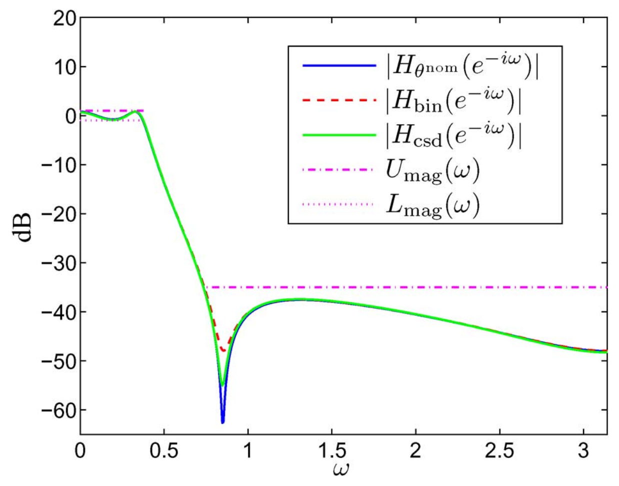

Fig. 1. Magnitude responses of the nominal filter (blue), the filter with coefficients truncated according to $\phi_{\rm bin}$ (red), and the filter with coefficients truncated according to $\phi_{\rm csd}$ (green), for direct form implementation.

TABLE II
NOMINAL AND TRUNCATED FILTER COEFFICIENTS FOR CASCADE FORM IMPLEMENTATION

	Nominal	Binary truncated	CSD truncated
b_{01}	0.01881	$2^{-6}+2^{-9}$	$2^{-6}+2^{-9}$
b_{11}	0.01072	$2^{-7}+2^{-9}$	$2^{-6}-2^{-8}$
b_{21}	0	0	0
a_{11}	-0.84757	$-(2^{-1}+2^{-2}+2^{-4}$ $+2^{-5})$	$-2^{0}+2^{-3}+2^{-5}$
a_{21}	0	0	0
b_{02}	1	2^{0}	2^{0}
b_{12}	-1.13115	$-(2^{0}+2^{-2}+2^{-4})$	$-2^{0}-2^{-2}-2^{-4}$
b_{22}	0.98682	2^{0}	2^{0}
a_{12}	-1.75488	$-(2^{0}+2^{-1}+2^{-2}$ $+2^{-8})$	$-2^{1}+2^{-2}-2^{-8}$
a_{22}	0.87429	$2^{-1}+2^{-2}+2^{-3}$	$2^{0}-2^{-3}$

with $\theta = (b_{01}, b_{11}, b_{21}, a_{11}, a_{21}, b_{02}, b_{12}, b_{22}, a_{12}, a_{22})$. (This cascade form is over parametrized; there are many choices of coefficients to realize the nominal transfer function. We simply choose one reasonable realization.)

The binary complexity of our nominal cascade design is $\Phi_{\rm bin}(\theta^{\rm nom}) = 185$. The best design obtained after 100 random runs of our algorithm achieves a binary complexity of $\Phi_{\rm bin}(\theta) = 20$, around 2 1 s per coefficient; see Table II. The average binary complexity of the designs returned by the algorithm over these 100 random runs is around 32 1 s.

The CSD complexity of our nominal cascade design is $\Phi_{\rm csd}(\theta^{\rm nom}) = 93$. The best design obtained after 100 random runs of our algorithm achieves a complexity of $\Phi_{\rm csd}(\theta) = 17$, around 2 nonzeros digits per coefficient; see Table II. The average CSD complexity of the designs returned by the algorithm over these 100 random runs is around 23 nonzero digits.

IIR filter in lattice-ladder form: The nominal transfer function is expressed in lattice-ladder form, with lattice coefficients k_1, k_2, k_3, and ladder coefficients c_0, c_1, c_2, c_3, as described in Section III, i.e., $\theta = (k_1, k_2, k_3, c_0, c_1, c_2, c_3)$.

The binary complexity of our nominal lattice-ladder design is $\Phi_{\rm bin}(\theta^{\rm nom}) = 194$. The best design obtained after 100 random runs of our algorithm achieves a binary complexity of $\Phi_{\rm bin}(\theta) = 21$, around 3 1 s per coefficient; see Table III.

TABLE III
NOMINAL AND TRUNCATED FILTER COEFFICIENTS FOR LATTICE-LADDER IMPLEMENTATION

	Nominal	Binary truncated	CSD truncated
k_1	-0.96415	$-(2^{-1}+2^{-2}+2^{-3}+2^{-4}+2^{-6}+2^{-7}+2^{-9})$	$-2^{0}+2^{-5}+2^{-8}+2^{-10}$
k_2	0.96077	$2^{-1}+2^{-2}+2^{-3}+2^{-4}+2^{-6}+2^{-7}$	$2^{0}-2^{-5}-2^{-7}$
k_3	-0.74102	$-(2^{-1}+2^{-2})$	$-2^{0}+2^{-2}$
c_0	0.01672	$2^{-6}+2^{-10}$	$2^{-6}+2^{-10}$
c_1	0.02162	$2^{-6}+2^{-7}$	$2^{-6}+2^{-8}$
c_2	0.03203	2^{-5}	2^{-5}
c_3	0.01057	2^{-6}	0

The average binary complexity of the designs returned by the algorithm over these 100 random runs is around 24 1 s.

The CSD complexity of our nominal lattice-ladder design is $\Phi_{\text{csd}}(\theta^{\text{nom}}) = 93$. the best design obtained after 100 random runs of our algorithm achieves a CSD complexity of $\Phi_{\text{csd}}(\theta) = 14$, i.e., 2 nonzero digits per coefficient; see Table III. The average CSD complexity of the designs returned by the algorithm over these 100 random runs is around 15.5 nonzero digits.

Comparison with simple truncation: We can compare the results obtained above with the simple approach of truncating the binary expansions of the coefficients to precision 2^{-q} (i.e., with q bits in their fractional part), choosing q as small as possible while still maintaining $\theta \in \mathcal{C}$. For the direct form implementation, the largest value of q for which $\theta \in \mathcal{C}$ is 14, and the binary complexity of the resulting design is $\Phi_{\text{bin}}(\theta) = 44$; its CSD complexity is $\Phi_{\text{csd}}(\theta) = 36$. For the cascade form implementation, the largest value of q with $\theta \in \mathcal{C}$ is 12, and which gives $\Phi_{\text{bin}}(\theta) = 46$ and $\Phi_{\text{csd}}(\theta) = 30$. For the lattice-ladder implementation, we have $q = 11$, with $\Phi_{\text{bin}}(\theta) = 36$ 1 s, and $\Phi_{\text{csd}}(\theta) = 22$. In each case, our method finds a filter coefficient vector with far smaller complexity.

B. FIR Filter Design

This example is based on the low-pass filter specifications given in Example 2 of [8]: a 59-tap linear FIR filter, with passband $[0, 0.042\pi]$, stopband $[0.14\pi, \pi]$, maximum passband ripple ± 0.2 dB, and minimum stopband attenuation 60 dB. The nominal filter coefficients are found using the spectral factorization method [26], [27], by maximizing the minimum stopband attenuation subject to maximum passband ripple ± 0.15 dB. The globally optimal solution achieves a minimum stopband attenuation of 71.2 dB. Thus, our specifications involve relaxing the passband ripple from ± 0.15 to ± 0.2 dB, and relaxing the stopband attenuation from 71.2 to 60 dB.

With the nominal coefficients rounded to 40 bits, the nominal filter coefficients have CSD complexity $\Phi_{\text{csd}}(\theta_{\text{nom}}) = 335$. We run our algorithm 100 times. The best result has complexity $\Phi_{\text{csd}}(\theta) = 71$, which is around 1.2 nonzero digits per coefficient. In constrast, the procedure described in [8] failed to find a 59-tap FIR filter, with quantized coefficients, that satisfies the specifications. They repeated their procedure with a (longer) 60-tap filter, however, and found a filter that satisfies the specifications, with complexity $\Phi_{\text{csd}}(\theta) = 87$, which is around 1.5 nonzero digits per coefficient.

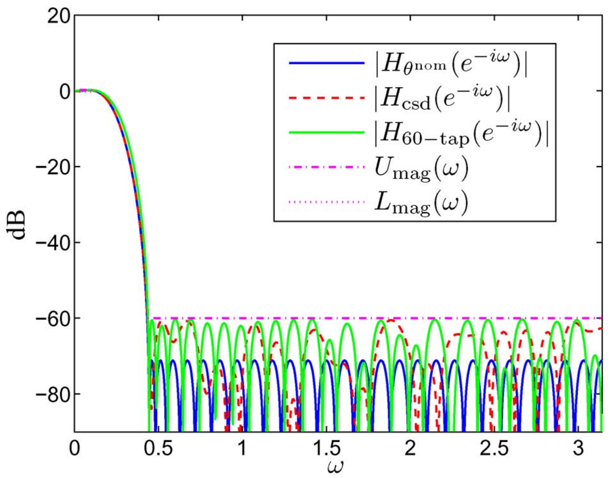

Fig. 2. Magnitude responses of the 59-tap nominal filter (blue), the 59-tap filter with coefficients truncated according to ϕ_{csd} (red), and the 60-tap filter obtained in [8] (green).

Fig. 2 shows the frequency response magnitude of the 59-tap nominal filter, the 59-tap filter that our truncation procedure returns, and the 60-tap filter obtained in Example 2 of [8].

VII. CONCLUSION

We have presented an algorithm for designing filters with low coefficient complexity that meet a set of design specifications. We have shown, through numerical examples, that the method presented is able to return designs with aggressively truncated coefficients even with very modest relaxation of the specifications. More complex specifications, such as absence of overflow induced instability, can be handled, for example using a Lyapunov-based nonlinear stability certificate; see, e.g., [29].

We mention that when the method is applied to overparametrized, or otherwise nonminimal nominal realizations, the results can be a final filter design with lower complexity than if a lower order, or uniquely parametrized, form is used. For example, when we apply the method with a sixth-order nominal design (say), we can end up with a sixth-order filter with fewer 1 s in its coefficients than when we start with a 4th order filter, with the same specifications in each case.

Our final comment concerns the difference between binary and CSD complexity measures. Although we have described a customized method for CSD coefficient truncation, which performs well, we should mention that a simpler approach leads to filter designs with nearly as small CSD complexity: We first design the filter for low binary complexity, and then simply express the resulting coefficients in CSD form.

ACKNOWLEDGMENT

The authors would like to thank several anonymous reviewers for helpful suggestions.

REFERENCES

[1] S. Arno and F. Wheeler, “Signed digit representations of minimial Hamming weight,” *IEEE Trans. Computers*, vol. 42, no. 8, pp. 1007–1010, 1993.

[2] F. Xu, C. Chang, and C. Jong, “Hamming weight pyramid: A new insight into canonical signed digit representation and its applications,” *Comput. Elect. Eng.*, vol. 33, no. 3, pp. 195–207, 2007.

[3] E. Avenhaus, "On the design of digital filters with coefficients of limited word length," *IEEE Trans. Audio Electroacoust.*, vol. 20, no. 3, pp. 206–212, 1972.

[4] F. Brglez, "Digital filter design with short word-length coefficients," *IEEE Trans. Circuits Syst.*, vol. 25, no. 12, pp. 1044–1050, 1978.

[5] A. Gray and J. Markel, "Quantization and bit allocation in speech processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 24, no. 6, pp. 459–473, 1976.

[6] R. Rink and H. Chong, "Performance of state regulator systems with floating point computation," *IEEE Trans. Autom. Control*, vol. 14, pp. 411–412, 1979.

[7] Y. Lim and Y. Yu, "A successive reoptimization approach for the design of discrete coefficient perfect reconstruction lattice filter bank," in *IEEE Int. Symp. Cicuits Syst.*, 2000, vol. 2, pp. 69–72.

[8] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, no. 7, pp. 1044–1047, 1989.

[9] J. Lee, C. Chen, and Y. Lim, "Design of discrete coefficient FIR digital filters with arbitrary amplitude and phase response," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, no. 7, pp. 444–448, 1993.

[10] Y. Lim and Y. Yu, "A width-recursive depth-first tree search approach for the design of discrete coefficient perfect reconstruction lattice filter bank," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 50, pp. 257–266, 2003.

[11] P. Frossard, P. Vandergheynst, R. Figueras, and M. Kunt, "A posteriori quantization of progressive matching pursuit streams," *IEEE Trans. Signal Process.*, vol. 52, no. 2, pp. 525–535, 2004.

[12] Y. Lim and S. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 30, no. 10, pp. 723–739, 1983.

[13] S. Chen and J. Wu, "The determination of optimal finite precision controller realizations using a global optimization strategy: A pole-sensitivity approach," in *Digital Controller Implementation and Fragility: A Modern Perpective*, R. Istepanian and J. Whidborne, Eds. New York: Springer-Verlag, 2001, ch. 6, pp. 87–104.

[14] N. Benvenuto, M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini, "Finite wordlength digital filter design using an annealing algorithm," in *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 1989, pp. 861–864.

[15] A. Fuller, B. Nowrouzian, and F. Ashrafzadeh, "Optimization of FIR digital filters over the canonical signed-digit coefficient space using genetic algorithms," in *Proc. 1998 Midwest Symp. Circuits Syst.*, 1998, pp. 456–459.

[16] P. Gentili, F. Piazza, and A. Uncini, "Efficient genetic algorithm design for power-of-two FIR filters," in *Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 1995, pp. 1268–1271.

[17] R. Istepanian and J. Whidborne, "Multi-objective design of finite wordlength controller structures," in *Proc. Congress on Evolut. Comput.* , 1999, vol. 1, pp. 61–68.

[18] S. Traferro, F. Capparelli, F. Piazza, and A. Uncini, "Efficient allocation of power-of-two terms in FIR digital filter design using Tabu search," in *Int. Symp. Circuits Syst. (ISCAS)*, 1999, pp. III-411–III-414.

[19] B. Wah, Y. Shang, and Z. Wu, "Discrete lagrangian methods for optimizing the design of multiplierless QMF banks," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 9, pp. 1179–1191, 1999.

[20] D. Kodek, "Design of optimal finite wordlength FIR digital filters using integer programming techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, no. 3, pp. 304–308, 1980.

[21] Y. Lim and S. Parker, "FIR filter design over a discrete powers-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 31, no. 3, pp. 583–591, 1983.

[22] Q. Zhao and Y. Tadokoro, "A simple design of FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 35, no. 5, pp. 566–570, 1988.

[23] T. Hilaire, P. Chevrel, and J. Whidborne, "A unifying framework for finite wordlength realizations," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 8, 2007.

[24] S. Barua, K. Kotteri, A. Bell, and J. Carletta, "Optimal quantized lifting coefficients for the 9/7 wavelet," in *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2004, vol. 5, pp. V-193–V-196.

[25] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[26] L. Rabiner, N. Graham, and H. Helms, "Linear programming design of IIR digital filters with arbitrary magnitude function," *IEEE Trans. Signal Process.*, vol. 22, no. 2, pp. 117–123, 1974.

[27] S. Wu, S. Boyd, and L. Vandenberghe, "FIR filter design via semidefinite programming and spectral factorization," in *IEEE Conf. Decision and Control*, 1996, pp. 271–276.

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge , U.K.: Cambridge University Press, 2004.

[29] J. Skaf and S. Boyd, "Controller coefficient truncation using Lyapunov performance certificate," in *Proc. Europ. Control Conf.*, 2007, pp. 4699–4706.

Joëlle Skaf received the Bachelor of Engineering degree in computer and communications engineering from the American University of Beirut in 2003, and the Master of Science degree in electrical engineering from Stanford University, Stanford, CA, in 2005.

She is a graduate student pursuing the Ph.D. degree in electrical engineering at Stanford University. Her current interests include convex optimization and its applications in control theory, machine learning, and computational finance.

Stephen P. Boyd (S'82–M'85–SM'97–F'99) received the A.B. degree in mathematics (*summa cum laude*) from Harvard University, Cambridge, MA, in 1980, and the Ph.D. degree in electrical engineer and computer science from the University of California, Berkeley, in 1985.

He is the Samsung Professor of Engineering in the Information Systems Laboratory, Electrical Engineering Department, Stanford University, Stanford, CA. His current interests include convex programming applications in control, signal processing, and circuit design. He is the author of *Linear Controller Design: Limits of Performance* (with C. Barratt, 1991), *Linear Matrix Inequalities in System and Control Theory* (with L. El Ghaoui, E. Feron, and V. Balakrishnan, 1994), and *Convex Optimization* (with L. Vandenberghe, 2004).

Dr. Boyd received an ONR Young Investigator Award, a Presidential Young Investigator Award, and the 1992 AACC Donald P. Eckman Award. He has received the Perrin Award for Outstanding Undergraduate Teaching in the School of Engineering, and an ASSU Graduate Teaching Award. In 2003, he received the AACC Ragazzini Education award. He is a a Distinguished Lecturer of the IEEE Control Systems Society.