

# Design of Canonical Higher Order Digital Filters

Jean H.F. Ritzerfeld

Technische Universiteit Eindhoven, Fac. Elektrotechniek

P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Phone: +31 (0)40 247 3252 Fax: +31 (0)40 246 6508

E-mail: [j.h.f.ritzerfeld@tue.nl](mailto:j.h.f.ritzerfeld@tue.nl)

**Abstract**—The study of stability tests for higher-order systems and the parallel treatment developed for the continuous and the discrete-time case [1] leads to a simple way to design higher-order digital filters. Using a bilinear transform on a canonical, direct form analog filter of arbitrary order, a digital filter can be found that is still canonical, but has low noise and low sensitivity. With a simple mapping of the coefficients of a discrete-time transfer function onto a canonical set of multipliers, the resulting filter can also be found directly in the digital domain.

**Keywords**—Filter Design, Higher Order, Canonical.

## I. INTRODUCTION

Higher order IIR filters are commonly implemented as a series or parallel connection of second-order sections having transfer functions with five degrees of freedom. These are then often realized as Direct Forms where the five degrees of freedom appear as multipliers in the signal flow graph. For those sections, however, that have poles close to the unit circle and with angles close to 0 (or  $\pi$ ), the coefficient sensitivity and the quantization noise become unacceptably high, specifically in fixed-point applications. Alternative solutions are then looked for [2], such as the Normal Form and the Optimal (i.e. minimum-noise) Form, both of which use extra degrees of freedom to achieve a better sensitivity/noise behavior at the cost of extra multipliers (i.e. they are non-canonical) and a less straightforward design, since the one-on-one relation between the values of the multipliers and the coefficients of the transfer function is lost.

The canonical Direct Wave Form that was introduced in [3] has low noise and sensitivity together with a very simple relation between its multipliers and the transfer coefficients. It is however still only of second order. In this paper we want to design an  $n$ -th order low noise structure directly, i.e. without having to expand into second-order sections. The resulting filter implementation is canonical and retains the simple mapping of the coefficients of the  $n$ -th order transfer function onto the canonical set of multipliers via a matrix-vector multiplication. The design starts out in Section II and III by exploiting the relationship between the stability tests for continuous versus discrete-time sys-

tems, studied in [1]. In Section IV the general  $n$ -th order scaled and canonical filter structure is presented, while in Section V its properties in terms of noise, sensitivity and stability are determined.

## II. BILINEAR MATRIX TRANSFORM

In [1] we considered the bilinear transform

$$s = \frac{z-1}{z+1}, \quad \text{or} \quad z = \frac{1+s}{1-s}, \quad (1)$$

to switch from the characteristic polynomial

$$b_0 s^n + b_1 s^{n-1} + \dots + b_{n-1} s + b_n \quad (2)$$

of an  $n$ -th order continuous-time system in the  $s$ -domain to the characteristic polynomial

$$a_0 z^n + a_1 z^{n-1} + \dots + a_{n-1} z + a_n \quad (3)$$

of an  $n$ -th order discrete-time system in the  $z$ -domain. This transform interrelates the coefficients  $a_i$  and  $b_{n-i}$  via an  $(n+1) \times (n+1)$  transformation matrix  $\mathbf{P}_n$  with columns indexed from  $j = 0$  to  $n$  found from the coefficients of  $(z+1)^{n-j}(z-1)^j$ . This matrix is involutory (it is its own inverse) apart from a factor  $2^{-n}$ . For example for  $n = 6$ ,

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 6 & 4 & 2 & 0 & -2 & -4 & -6 \\ 15 & 5 & -1 & -3 & -1 & 5 & 15 \\ 20 & 0 & -4 & 0 & 4 & 0 & -20 \\ 15 & -5 & -1 & 3 & -1 & -5 & 15 \\ 6 & -4 & 2 & 0 & -2 & 4 & -6 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix}. \quad (4)$$

Note that the entries satisfy  $p_{ij} = p_{i-1,j} + p_{i-1,j+1} + p_{i,j+1}$  so the matrix can be built from its zeroth row and its  $n$ -th column starting in the upper right corner. We will adopt the following notation:

$$\hat{\mathbf{a}} = lr(\mathbf{P}_n) \hat{\mathbf{b}} \quad \text{and} \quad \hat{\mathbf{b}} = 2^{-n} ud(\mathbf{P}_n) \hat{\mathbf{a}}, \quad (5)$$

where the coefficient vectors  $\hat{\mathbf{a}}$  and  $\hat{\mathbf{b}}$  are indexed from 0 to  $n$ , and  $lr(\mathbf{P}_n)$  denotes the matrix  $\mathbf{P}_n$  with its columns

flipped in the left-right direction, while  $ud(\mathbf{P}_n)$  indicates  $\mathbf{P}_n$  with its rows flipped up-down.

The polynomials of (2) and (3) are also the characteristic polynomials associated with the direct form state matrices

$$\mathbf{A}_c = \begin{pmatrix} -b_1/b_0 & -b_2/b_0 & \cdots & -b_{n-1}/b_0 & -b_n/b_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}, \quad (6)$$

$$\mathbf{A}_d = \begin{bmatrix} -a_1/a_0 & -a_2/a_0 & \cdots & -a_{n-1}/a_0 & -a_n/a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix},$$

the state equations (with state  $\mathbf{x}$  and input  $u$ ) of the corresponding  $n$ -th order linear systems being, respectively

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}_c \mathbf{x}(t) + \mathbf{B}_c u(t), \quad \mathbf{x}[k+1] = \mathbf{A}_d \mathbf{x}[k] + \mathbf{B}_d u[k], \quad (7)$$

where  $\mathbf{B}_c = (1/b_0, 0, \dots, 0)^T$  and  $\mathbf{B}_d = [1/a_0, 0, \dots, 0]^T$ . Note that we use square parentheses for  $\mathbf{A}_d$  and  $\mathbf{B}_d$  to remind us that we are dealing with a discrete-time system. Now we would like to switch between state descriptions of these systems, not just their characteristic polynomials. Using the *bilinear matrix transform*, known from system theory, the continuous direct form  $\mathbf{A}_c, \mathbf{B}_c$  is mapped to the state description of a discrete-time system with

$$\mathbf{A}_w = (\mathbf{I} - \mathbf{A}_c)^{-1}(\mathbf{I} + \mathbf{A}_c), \quad \mathbf{B}_w = \sqrt{2}(\mathbf{I} - \mathbf{A}_c)^{-1} \mathbf{B}_c, \quad (8)$$

where  $\mathbf{A}_w$  is not a direct form matrix, but has the same characteristic polynomial (3) associated with it as  $\mathbf{A}_d$ .

As an example, consider the third-order direct form filter with state space description

$$\mathbf{A}_c = \begin{pmatrix} -b_1/b_0 & -b_2/b_0 & -b_3/b_0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{B}_c = \begin{pmatrix} 1/b_0 \\ 0 \\ 0 \end{pmatrix}.$$

The bilinear state-space transform (8) yields the discrete-time system with state matrix

$$\mathbf{A}_w = \frac{\begin{bmatrix} b_0 - b_1 - b_2 - b_3 & -2b_2 - 2b_3 & -2b_3 \\ 2b_0 & b_0 + b_1 - b_2 - b_3 & -2b_3 \\ 2b_0 & 2b_0 + 2b_1 & b_0 + b_1 + b_2 - b_3 \end{bmatrix}}{b_0 + b_1 + b_2 + b_3} \quad (9)$$

$$= \frac{1}{c_0} \begin{bmatrix} c_1 & c_2 - c_0 & c_3 - c_0 \\ c_1 + c_0 & c_2 & c_3 - c_0 \\ c_1 + c_0 & c_2 + c_0 & c_3 \end{bmatrix},$$

where

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}. \quad (10)$$

Looking at the matrix  $\mathbf{A}_w$  in terms of the  $c$ -parameters, we note that the system (like the direct form  $\mathbf{A}_d$  with  $a_0 = 1$ ) is canonical, i.e. it can be implemented with only  $n = 3$  multipliers if we let  $c_0 = 1$ . To see this, just introduce the summation node  $s[k] = c_1 x_1[k] + c_2 x_2[k] + c_3 x_3[k]$  in the implementation using three multipliers and note that the new state  $\mathbf{x}[k+1]$  can be found from  $s[k]$  without extra multiplications, e.g.  $x_1[k+1] = s[k] - x_2[k] - x_3[k]$ . Of course, we could also let  $c_0 = \sqrt{2}$  and use the three multipliers  $c_1/c_0, c_2/c_0, c_3/c_0$  for the summation node  $s[n]$ . This has the added advantage that the matrix  $\mathbf{B}_w$ , which from (8) is found to be

$$\mathbf{B}_w = \frac{\sqrt{2}}{b_0 + b_1 + b_2 + b_3} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \frac{\sqrt{2}}{c_0} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad (11)$$

uses no extra multiplier, the same as for  $\mathbf{B}_d$  with  $a_0 = 1$ .

Alternatively, we could also drop the factor  $\sqrt{2}$  in (8) which is just a scaling factor that can be incorporated in the matrix  $\mathbf{C}_w$  of the system output equation

$$y[k] = \mathbf{C}_w \mathbf{x}[k] + \mathbf{D}_w u[k]. \quad (12)$$

The reason why this factor is introduced in system theory is the fact that with it, the continuous-time system  $\mathbf{A}_c, \mathbf{B}_c$  will have the same covariance  $E(\mathbf{x}\mathbf{x}^T)$  as the discrete-time system  $\mathbf{A}_w, \mathbf{B}_w$ . Given that the input variance  $E(u^2)$  is unity, the covariance  $\mathbf{K} = E(\mathbf{x}\mathbf{x}^T)$  of a continuous-time system is found from the Lyapunov equation [1]

$$-\mathbf{A}\mathbf{K} - \mathbf{K}\mathbf{A}^T = \mathbf{B}\mathbf{B}^T, \quad (13)$$

whereas the covariance matrix of a discrete-time system is found from the Lyapunov equation [1]

$$\mathbf{K} - \mathbf{A}\mathbf{K}\mathbf{A}^T = \mathbf{B}\mathbf{B}^T. \quad (14)$$

Entering the system  $\mathbf{A}_w, \mathbf{B}_w$  into (14) and substituting the relations (8) results in the continuous Lyapunov equation (13), as can easily be checked. Hence  $\mathbf{K}_w = \mathbf{K}_c$  or, dropping the factor  $\sqrt{2}$  in (8),  $\mathbf{K}_w = \frac{1}{2}\mathbf{K}_c$ .

The covariance matrix  $\mathbf{K}_w$  can be computed explicitly. In [1] it is stated that for the two direct forms (6) there are only  $n$  distinct entries in the covariance matrix, since  $\mathbf{K}_d$  has equal entries on its diagonals and  $\mathbf{K}_c$  has alternating

equal entries on its odd cross-diagonals and zeros on its even cross-diagonals. Specifically, for  $n = 3$ ,

$$\mathbf{K}_c = \begin{pmatrix} k_1 & 0 & -k_2 \\ 0 & k_2 & 0 \\ -k_2 & 0 & k_3 \end{pmatrix}, \quad \mathbf{K}_d = \begin{bmatrix} l_1 & l_2 & l_3 \\ l_2 & l_1 & l_2 \\ l_3 & l_2 & l_1 \end{bmatrix}. \quad (15)$$

From [1] we know that the entries of  $\mathbf{K}_c$  are given by

$$k_j = \frac{|\mathbf{H}_{1j}|}{2b_0 \det \mathbf{H}}, \quad \text{where } \mathbf{H} = \begin{pmatrix} b_1 & b_3 & 0 \\ b_0 & b_2 & 0 \\ 0 & b_1 & b_3 \end{pmatrix} \quad (16)$$

and  $|\mathbf{H}_{1j}|$  are the minors of the first row of the Hurwitz matrix  $\mathbf{H}$ , so with  $\mathbf{B}_w = (1/c_0)[1, 1, 1]^T$  we have

$$k_j = \frac{|\mathbf{H}_{1j}|}{4b_0 \det \mathbf{H}}, \quad \mathbf{K}_w = \frac{\begin{bmatrix} b_2 b_3 & 0 & -b_0 b_3 \\ 0 & b_0 b_3 & 0 \\ -b_0 b_3 & 0 & b_0 b_1 \end{bmatrix}}{4b_0 b_3 (b_1 b_2 - b_0 b_3)}. \quad (17)$$

Incidentally, the Hurwitz matrix associated with the polynomial (2) of arbitrary order is an  $n \times n$  matrix with alternating rows of the odd-indexed and the even-indexed coefficients, such that the principal diagonal is  $b_1, \dots, b_n$ .

The question now is, what are the properties of the system  $\mathbf{A}_w, \mathbf{B}_w$  as opposed to the direct form  $\mathbf{A}_d, \mathbf{B}_d$  and also, can we design it directly in the digital domain for any order  $n$ , e.g. with a similarity transform or a simple mapping of the coefficients  $a_i$  onto the parameters  $c_i$  in (10)?

### III. DESIGN IN THE DIGITAL DOMAIN

We want to design an  $n$ -th order digital filter that has the polynomial (3) as denominator of its transfer function, so

$$H(z) = \frac{d_0 z^n + d_1 z^{n-1} + \dots + d_{n-1} z + d_n}{a_0 z^n + a_1 z^{n-1} + \dots + a_{n-1} z + a_n}. \quad (18)$$

In a canonical implementation, only  $2n + 1$  multipliers are needed, since we can let  $a_0 = 1$  without loss of generality. For the direct form we would take the state matrix  $\mathbf{A}_d$  as in (6) with  $a_0 = 1$ ,  $\mathbf{B}_d$  as  $[1, 0, \dots, 0]^T$ , and the matrices  $\mathbf{C}_d$  and  $\mathbf{D}_d$  of the output equation as

$$\mathbf{C}_d = \mathbf{d}^T - d_0 \mathbf{a}^T, \quad \mathbf{D}_d = [d_0], \quad (19)$$

where  $\mathbf{d}$  and  $\mathbf{a}$  denote the numerator and denominator coefficient vectors indexed from 1 to  $n$ . Remember that we use the ‘hat’-notation  $\hat{\mathbf{a}}, \hat{\mathbf{d}}$  if the index 0 is included.

Now we can write down the  $n$ -th order state matrix of the alternative canonical system  $\mathbf{A}_w, \mathbf{B}_w, \mathbf{C}_w, \mathbf{D}_w$  as

$$\mathbf{A}_w = \frac{1}{c_0} \begin{bmatrix} c_1 & c_2 & \dots & c_n \\ c_1 & c_2 & \dots & c_n \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \dots & c_n \end{bmatrix} + \begin{bmatrix} 0 & -1 & \dots & -1 \\ 1 & 0 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 0 \end{bmatrix} \quad (20)$$

equivalent to (9), where the vector  $\hat{\mathbf{c}}$  is found from the the polynomial coefficients  $\hat{\mathbf{b}}$  of a continuous-time system via an  $(n + 1) \times (n + 1)$  matrix  ${}_c \mathbf{M}_b$  as in (10). But we want to relate the  $c$ -parameters to the denominator coefficients  $\hat{\mathbf{a}}$  of the discrete-time system in (18). Using (5) we have

$$\hat{\mathbf{c}} = {}_c \mathbf{M}_b 2^{-n} ud(\mathbf{P}_n) \hat{\mathbf{a}} = {}_c \mathbf{M}_a \hat{\mathbf{a}}, \quad (21)$$

where the entries  $m_{ij}$  ( $i, j = 0, \dots, n$ ) of  ${}_c \mathbf{M}_b$  are  $-1$  for  $i \leq j$  and  $i, j \neq 0$ , and 1 otherwise. The matrix  ${}_c \mathbf{M}_a$  has a simple structure, e.g. expanding (21) for  $n = 3$  yields

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 4 & 0 & 0 & 0 \\ -3 & -1 & 1 & -1 \\ 0 & -2 & 0 & 2 \\ 3 & -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}. \quad (22)$$

For any order  $n$  we have  $c_0 = a_0$  with

$${}_c \mathbf{M}_a = \frac{1}{2^{n-1}} \begin{pmatrix} 2^{n-1} & \mathbf{0}_n^T \\ \gamma & -ud(\mathbf{P}_{n-1}) \end{pmatrix}. \quad (23)$$

Letting  $a_0 = 1$ , the parameters  $c_i$  for  $i \neq 0$  are given by

$$\mathbf{c} = 2^{-n+1} \{ \gamma - ud(\mathbf{P}_{n-1}) \mathbf{a} \}. \quad (24)$$

The vector  $\gamma$  in this expression can be found by taking  $\mathbf{a} = \mathbf{0}_n$ , or equivalently  $\hat{\mathbf{a}} = (1, 0, \dots, 0)^T$  in (21) to yield

$$\hat{\gamma} = \frac{1}{2} {}_c \mathbf{M}_b \mathbf{p}_n, \quad (25)$$

where  $\mathbf{p}_n$  is a (column) vector containing the  $n$ -th row of Pascal’s triangle.

With (24) we have obtained a simple mapping of the denominator coefficients  $\mathbf{a}$  onto the parameters  $\mathbf{c}$  of the alternative canonical matrix  $\mathbf{A}_w$ . Seeing that the first row of  $\mathbf{A}_d$ , i.e.  $-\mathbf{a}^T$ , is mapped to the first row  $\mathbf{c}^T$  of  $\mathbf{A}_w$  by right multiplication with a matrix  $\mathbf{T} = 2^{-n+1} ud(\mathbf{P}_{n-1})^T$ , we suspect that we can link the state descriptions of the direct and alternative forms through a state transformation  $\mathbf{x}_d = \mathbf{T} \mathbf{x}_w$ . Indeed, further inspection reveals that  $\mathbf{A}_w$  can be found from  $\mathbf{A}_d$  through a similarity transform

$$\mathbf{A}_w = \mathbf{T}^{-1} \mathbf{A}_d \mathbf{T}, \quad (26)$$

and hence

$$\mathbf{B}_w = \mathbf{T}^{-1} \mathbf{B}_d, \quad \mathbf{C}_w = \mathbf{C}_d \mathbf{T}, \quad \mathbf{D}_w = \mathbf{D}_d, \quad (27)$$

where

$$\mathbf{T} = 2^{-n+1} ud(\mathbf{P}_{n-1})^T \quad \text{and} \quad \mathbf{T}^{-1} = lr(\mathbf{P}_{n-1})^T. \quad (28)$$

Note that the first column of  $\mathbf{T}^{-1}$  is a vector of ones, so  $\mathbf{B}_w$  is the unity vector  $\mathbf{1}_n$  as indeed it should be, cf. the third-order example in the previous section. Furthermore,

since we now know the entries of  $\mathbf{C}_w$  from the transfer function (18) via (19) and (27), we can draw the canonical implementation using  $2n + 1$  multipliers, i.e.  $n$  for  $\mathbf{A}_w$ ,  $n$  for  $\mathbf{C}_w$  and one for  $\mathbf{D}_w = [d_0]$ . As mentioned, we need to introduce the summation node  $s[k] = \sum_{j=1}^n c_j x_j[k] + u[k]$  which is then fed back to all new states  $x_i[k + 1]$ . We can choose to find  $\mathbf{A}_w$  either from the similarity transform (26) or from  $\mathbf{1}_n \cdot \mathbf{c}^T + \mathbf{E}$ , where  $\mathbf{c}$  is given by (24) and the matrix  $\mathbf{E}$  has entries  $e_{ij} = \text{sign}[i - j]$ , cf. (20), so e.g.  $x_1[k + 1]$  is given by  $s[k] - \sum_{j=2}^n x_j[k]$ . The resulting structure is still unscaled; to scale, we need to normalize the variances  $k_j$  of the  $n$  state variables, given by (17), to unity.

Before doing so, let us look at the allowed range of the  $c$ -parameters. The system (18) is stable if its continuous-time counterpart with denominator polynomial (2) has all poles with negative real parts. This is the case iff the Hurwitz matrix (16) is positive definite (given that  $b_0 > 0$ ), or equivalently, iff its principal minors are positive. A necessary condition, though not sufficient, is that all entries on the principal diagonal are positive, so  $b_i > 0$  for  $i = 0, \dots, n$ . Expressing  $\hat{\mathbf{b}}$  in terms of  $\hat{\mathbf{c}}$ , via  $\hat{\mathbf{b}} = \mathbf{b}\mathbf{M}_c \hat{\mathbf{c}}$ , we find

$$\mathbf{b}\mathbf{M}_c = \frac{1}{2} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \end{pmatrix} \quad (29)$$

as the inverse of  $\mathbf{c}\mathbf{M}_b$  in (10), or in general

$$b_i = \frac{1}{2}(c_{i+1} - c_i) \quad \text{for } i = 1, \dots, n-1 \quad \text{and} \\ b_0 = \frac{1}{2}(c_0 + c_1), \quad b_n = \frac{1}{2}(c_0 - c_n). \quad (30)$$

This then means that, given  $c_0 = 1$ ,

$$-1 < c_1 < c_2 < \dots < c_n < 1, \quad (31)$$

so the  $c$ -parameters are distinct in an increasing sequence and, more importantly, are absolutely bounded by unity. To conclude, we determine the inverse of  $\mathbf{c}\mathbf{M}_a$  in (22), i.e.

$$\mathbf{a}\mathbf{M}_c = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 \\ 3 & 2 & 0 & -2 \\ 0 & -1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{0}_n^T \\ \boldsymbol{\alpha} & -lr(\mathbf{P}_{n-1}) \end{pmatrix}, \quad (32)$$

so

$$\mathbf{a} = \boldsymbol{\alpha} - lr(\mathbf{P}_{n-1}) \mathbf{c}.$$

The vector  $\boldsymbol{\alpha}$  in this expression can be found by taking  $\mathbf{c} = \mathbf{0}_n$ , or equivalently  $\hat{\mathbf{c}} = (1, 0, \dots, 0)^T$  in the relation  $\hat{\mathbf{a}} = \mathbf{a}\mathbf{M}_c \hat{\mathbf{c}} = lr(\mathbf{P}_n) \mathbf{b}\mathbf{M}_c \hat{\mathbf{c}}$  to yield

$$\hat{\boldsymbol{\alpha}} = lr(\mathbf{P}_n) \frac{1}{2}(1, 0, \dots, 0, 1)^T = \frac{1}{2}(\mathbf{p}_n^* + \mathbf{p}_n), \quad (33)$$

where  $\mathbf{p}_n^*$  contains the  $n$ -th row of Pascal's triangle with alternating signs. In other words,  $\alpha_i = 0$  for odd  $i$  and equal to  $[\mathbf{p}_n]_i$  for even  $i$ . So, since  $\boldsymbol{\alpha}$  is so easy to determine, we can find  $\boldsymbol{\gamma}$  in (24) more readily with

$$\boldsymbol{\gamma} = ud(\mathbf{P}_{n-1}) \boldsymbol{\alpha}, \quad (34)$$

instead of with the earlier expression (25), to yield

$$\mathbf{c} = 2^{-n+1} ud(\mathbf{P}_{n-1}) (\boldsymbol{\alpha} - \mathbf{a}) \quad (35)$$

as the quickest way to determine the  $c$ -parameters.

#### IV. SCALING OF THE N-TH ORDER DESIGN

Scaling is a state transformation with a diagonal matrix  $\mathbf{S} = \text{diag}\{\sqrt{k_1}, \dots, \sqrt{k_n}\}$ , where  $k_j$  are the diagonal entries of the  $\mathbf{K}_w$ , given by (17). This means that the state matrix undergoes a similarity transform  $\mathbf{S}^{-1} \mathbf{A}_w \mathbf{S}$  and the covariance matrix is transformed into  $\mathbf{S}^{-1} \mathbf{K}_w \mathbf{S}^{-T}$ , the diagonal entries of which are unity. Of course we want to avoid using extra multipliers, so in practice scaling multipliers are rounded to powers of two such that the variances of the  $n$  state variables are less than (but close to) unity. Power-of-two scalars are simple shift operations. Still, the number of scalars in an  $n$ -th order design is  $n^2$ , since  $\mathbf{B}_w$  has  $n$  entries to be scaled and all  $n^2 - n$  off-diagonal entries of  $\mathbf{A}_w$  are changed by a diagonal similarity transform.

To be specific,  $\mathbf{B}'_w = \mathbf{S}^{-1} \mathbf{B}_w = [1/\sigma_1, \dots, 1/\sigma_n]^T$ , where  $\sigma_i = 2^p$  with  $p = \lceil \log_2 \sqrt{k_i} \rceil$ , and the entries  $a'_{ij}$  of  $\mathbf{A}'_w = \mathbf{S}^{-1} \mathbf{A}_w \mathbf{S}$  are

$$a'_{ij} = (c_j + \text{sign}[i - j]) \sigma_j / \sigma_i. \quad (36)$$

Thus, introducing the summation node

$$s'[k] = \sum_{j=1}^n \sigma_j c_j x'_j[k] + u[k], \quad (37)$$

the scaled new state variables  $x'_i[k + 1]$  can be found with

$$x'_i[k + 1] = s'[k] / \sigma_i + \sum_{j=1}^n \text{sign}[i - j] \sigma_j x'_j[k] / \sigma_i, \quad (38)$$

necessitating one scalar  $\sigma_i$  and  $n - 1$  scalars  $\sigma_j / \sigma_i$  for each state variable, hence  $n^2$  shift operations for the total implementation. Note that the scalars for  $s'[k]$  and the output node  $y[k] = \mathbf{C}'_w \mathbf{x}'[k] + \mathbf{D}_w u[k]$ , where  $\mathbf{C}'_w = \mathbf{C}_w \mathbf{S}$ , can be integrated with the multipliers necessary to realize these two linear combinations of the state variables.

We can, however, significantly reduce the number of scalars from  $n^2$  to  $n$ , if we rewrite the state equations as

$$x_1[k + 1] = c_1 x_1[k] + \sum_{j=2}^n (c_j - 1) x_j[k] + u[k] \quad \text{and}$$

$$x_i[k+1] = x_{i-1}[k+1] + x_{i-1}[k] + x_i[k] \quad \text{for } i \neq 1, \quad (39)$$

which after scaling read as

$$x'_1[k+1] = c_1 x'_1[k] + \sum_{j=2}^n \sigma_j (c_j - 1) x'_j[k] / \sigma_1 + u[k] / \sigma_1, \quad (40)$$

$$x'_i[k+1] = \sigma_{i-1} (x'_{i-1}[k+1] + x'_{i-1}[k]) / \sigma_i + x'_i[k].$$

The resulting structure is depicted in Fig. 1, where indeed only a single shift operation is used for every state variable. The output summation to the right is realized with multipliers  $\sigma_j t_j$  which are the scaled entries  $t_j$  of  $\mathbf{C}_w^T$  given by

$$\mathbf{t} = 2^{-n+1} \text{ud}(\mathbf{P}_{n-1}) (\mathbf{d} - d_0 \mathbf{a}). \quad (41)$$

In total, the number of multipliers is  $2n+1$ , the number of delays is  $n$ , the number of scalars is  $n$  and the number of adders is  $2n$ , of which  $n-1$  are used to add the current and new state variables. The quantization points are right in front of the delays, so all adders are double precision.

In Matlab the design can be written as a simple function in an m-file named `canonic` that carries out a similarity transform on a direct form state space description generated by the Matlab function `tf2ss` (which stands for ‘transfer function to state space’). The input arguments ( $\mathbf{b}$ ,  $\mathbf{a}$ ) on the first line are the numerator and denominator coefficient vectors  $\hat{\mathbf{d}}$  and  $\hat{\mathbf{a}}$  of the transfer function (18). The output argument  $s$  on the last line is the number of bits to shift over for each of the scalars  $\sigma_j$ . Note that the Matlab function `dgram(A, B)` calculates the covariance matrix which in system theory is sometimes called the (discrete-time) controllability gramian. The function `pascalm` determines  $\mathbf{P}_n$  which is a Pascal-type matrix with columns of binomial coefficients.

```
function [A,B,C,D,s]=canonic(b,a)
[A,B,C,D]=tf2ss(b,a);
n=length(a)-1; P=pascalm(n-1);
T=flipud(P)'/2^(n-1); Ti=flipplr(P)';
A=Ti*A*T; B=Ti*B; C=C*T; D=D;
s=ceil(log2(sqrt(diag(dgram(A,B)))));
```

```
function P=pascalm(n)
P=zeros(n+1,n+1);
for k=0:n,
    bino1=binopdf(0:n-k,n-k,.5);
    bino2=binopdf(0:k,k,.5).*(-1).^([0:k]);
    P(:,k+1)=round(2^n*conv(bino1,bino2))';
end
```

As a matter of interest, we determine the unscaled transfer functions  $F_j(z)$  from input  $u[k]$  to the state variables

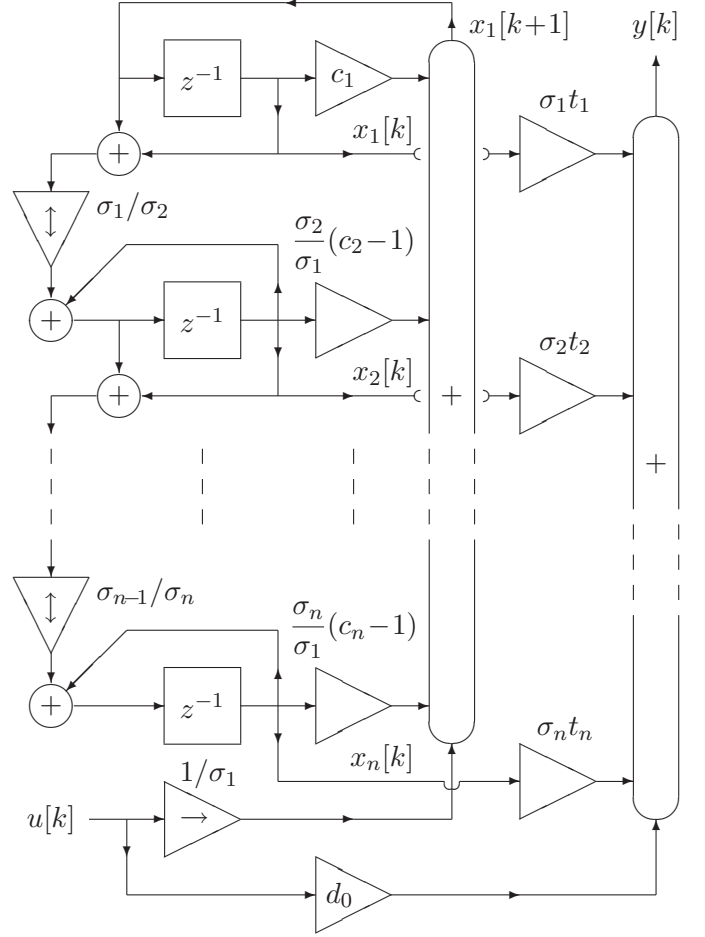


Fig. 1. Scaled canonical  $n$ -th order digital filter structure

$x_j[k]$ , of which the diagonal entries  $k_j$  of the covariance matrix are the square  $L_2$ -norms  $\|F_j\|_2^2$ . We find

$$F_j(z) = \frac{(z-1)^{n-j}(z+1)^{j-1}}{z^n + a_1 z^{n-1} + \dots + a_{n-1} z + a_n} \quad (42)$$

for  $j = 1$  to  $n$ , so the state variables can be used as high-pass, bandpass and lowpass outputs with numerators of order  $n-1$ . An  $n$ -th order HP numerator  $(z-1)^n$  is found with  $x_1[k+1] - x_1[k]$ , while an  $n$ -th order LP numerator  $(z+1)^n$  is made with  $x_n[k+1] + x_n[k]$ .

## V. PROPERTIES OF THE N-TH ORDER DESIGN

Similar to  $\|F_j\|_2^2$ , the square  $L_2$ -norms  $\|G_j\|_2^2$  of the transfer functions  $G_j(z)$  from the state variables to the output  $y[k]$  are on the diagonal of the observability gramian  $\mathbf{W}$  which is the companion matrix to the controllability gramian  $\mathbf{K}$ , to be computed with `dgram(A', C')` in Matlab. It is the solution of the Lyapunov equation [2]

$$\mathbf{W} - \mathbf{A}^T \mathbf{W} \mathbf{A} = \mathbf{C}^T \mathbf{C}, \quad (43)$$

comparable to (14). We need to determine this matrix in order to calculate the total output quantization noise variance

which we will then use as a quality measure for our  $n$ -th order design. Assuming that there are quantizers at all (new) state nodes, representing  $n$  uncorrelated error sources with variance  $q^2/12$ , the output noise of the scaled system is

$$N = \frac{q^2}{12} \sum_{j=1}^n w'_{jj} = \frac{q^2}{12} \sum_{j=1}^n k_j w_{jj} = \frac{q^2}{12} G, \quad (44)$$

where  $q$  is the quantization step size, and  $w'_{jj}$  and  $w_{jj}$  are the diagonal entries of the scaled and unscaled observability gramian. So, the noise gain  $G$  of any given implementation, i.e. the factor with which to multiply  $q^2/12$ , depends on the diagonals of both  $\mathbf{K}$  and  $\mathbf{W}$ , in Matlab reading as:  $G = \text{diag}(\text{dgram}(\mathbf{A}, \mathbf{B}))' * \text{diag}(\text{dgram}(\mathbf{A}', \mathbf{C}'))$ .

As an example, let us implement a sixth-order narrow-band elliptic filter with a passband ripple of 1 dB, a stopband attenuation of 90 dB and a cut-off frequency of  $\pi/10$ :  $[b, a] = \text{ellip}(6, 1, 90, 0.1)$ ;  $\text{zplane}(b, a)$ . Fig. 2 shows the corresponding pole-zero plot. Comparing

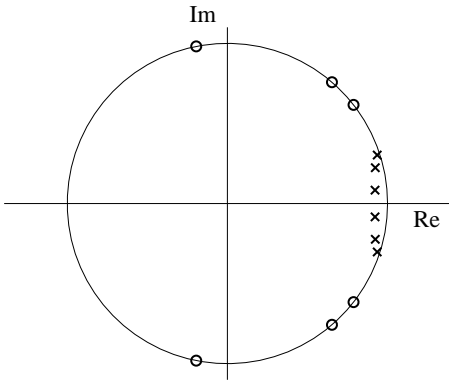


Fig. 2. Pole-zero plot of sixth-order elliptic filter

implementations, we look at the noise gain  $G_d$  of the sixth-order direct form, the noise gain  $G_{sd}$  of the series connection of three second-order direct form sections, the noise gain  $G_w$  of the canonical alternative form and the noise gain  $G_{so}$  of the sectional optimal design, i.e. the series connection of three optimal sections. We find:  $G_d = 3.1512 \cdot 10^6$ ,  $G_{sd} = 3.1060 \cdot 10^4$ ,  $G_w = 45.2406$  and  $G_{so} = 21.9429$ . So we would lose 11 bits of dynamic range in a sixth-order direct form implementation (which is not very practical) and still 7.5 bits in an implementation with second-order direct form sections. The canonical alternative form on the other hand loses less than 3 bits and differs only half a bit with the sectional optimal design (which uses 27 multipliers as compared to 13 for the canonical design). Note that the noise gain of a direct form is given by  $k_{d,11} w_{d,11}$ , since there is a single quantizer necessary for the first state variable only. The noise gain of an optimal section depends on the eigenvalues of  $\mathbf{KW}$  [2].

An important feature of a good design is the fact that its noise gain is independent of the cut-off frequency, which indeed  $G_w$  is. If the above elliptic filter is made extremely narrow-band with a cut-off frequency  $\pi/100$ , the resulting  $G_w$  and  $G_{so}$  are unchanged, but the two direct form implementations become useless with noise gains  $G_d = 2.3283 \cdot 10^{16}$  and  $G_{sd} = 2.3264 \cdot 10^{10}$ , corresponding to 27 and 17 bits loss of dynamic range respectively. Given its low noise gain for any cut-off frequency, the scaled canonical structure can be implemented with 16 bit registers for the state variables as well as for the multipliers, since a low noise design inherently has a low coefficient sensitivity. To prove the point, Fig. 3 shows the effect of the use of 16 bit multipliers on the amplitude response (with  $\theta_c = \pi/100$ , the frequency axis is drawn only up to  $\pi/10$ ).

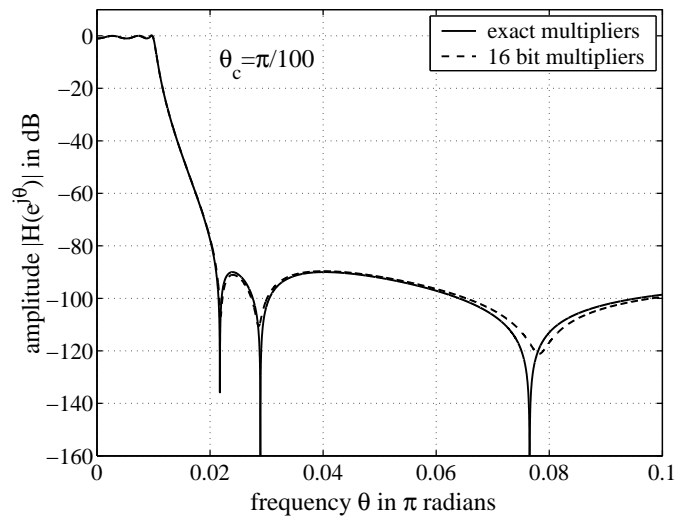


Fig. 3. Amplitude response of sixth-order elliptic filter

Looking at the necessary scaling operations, the output argument  $s$  of the Matlab function `canonic` is found to be  $(5, 11, 17, 23, 29, 36)^T$ , so there are 6 bit shifts (to the right) between the state variables and a 5 bit right shift  $1/\sigma_1$  for  $x_1$ . Note that a 36 bit shift in one go would be impossible in a 16 bit implementation, whereas 6 steps of 6 bits present no problem. If the vector  $s$  were reversed however, we would need start with  $x_n$  and work backwards to  $x_1$  in terms of the necessary shifts between states, i.e.

$$x'_n[k+1] = \sum_{j=1}^{n-1} \sigma_j (c_j + 1) x'_j[k] / \sigma_n + c_n x'_n[k] + u[k] / \sigma_n, \quad (45)$$

$$x'_{i-1}[k+1] = \sigma_i (x'_i[k+1] - x'_i[k]) / \sigma_{i-1} - x'_{i-1}[k],$$

to replace (40). Can this be avoided by reversing the order of the state variables? A decreasing shift vector  $s$  is typically associated with systems that have poles with negative real parts, as is the case for filters with  $\theta_c > \pi/2$ .

For example, if we design the elliptic high pass filter  $[b, a] = \text{ellip}(6, 1, 90, .9, 'high')$  in Matlab, the pole-zero plot is as Fig. 2 mirrored with respect to the imaginary axis. The resulting shift vector  $s$  is found to be  $(18, 15, 12, 9, 6, 4)^T$ , i.e. the reverse of the corresponding low pass case. Looking at the state description, the matrices  $A$  and  $C$  are negated and rotated (over 180 degrees) versions of the corresponding low pass matrices, where the negation is caused by the mirroring of the pole-zero plot (low pass to high pass transform  $z := -z$ ) and the rotation is associated with a reversal of the state order. This then means that if we were to dispense with the state reversal, we could still use the implementation of Fig. 1, albeit with a state matrix that has a decreasing diagonal

$$1 > c_1 > c_2 > \dots > c_n > -1 \quad (46)$$

instead of an increasing one, cf. (31). The simplest way to achieve this non-reversal of the state variables is to take

$$\mathbf{T} = 2^{-n+1} \mathbf{P}_{n-1}^T, \quad \text{with inverse} \quad \mathbf{T}^{-1} = \mathbf{P}_{n-1}^T, \quad (47)$$

in the similarity transform (26), (27), in other words to dispense with the flipping up-down and left-right of the Pascal matrix. We can make this choice of  $\mathbf{T}$  dependent on the pole locations by changing the fourth line in the function `canonic` into the `if` statement

```
if a(2)>0, T=P'/2^(n-1); Ti=P'; else
T=flipud(P)'/2^(n-1); Ti=fliplr(P)'; end
```

since the second denominator coefficient  $a(2)$  is equal to minus the sum of the poles. Alternatively, we can also add an inverter to all delay elements, i.e. replace  $z^{-1}$  by  $-z^{-1}$ , in the original low pass filter to implement a high pass filter with a mirrored pole-zero plot. The resulting structure is identical to the above implementation, since negating  $A$  and  $C$  is equivalent to the substitution  $z := -z$ .

Note that the basic building block in Fig. 1 is a delay element with a feedback of unity, while the sum of its input and output is fed forward. This building block has a transfer function  $(z+1)/(z-1)$  which is a bilinear integrator  $1/s$ . Replacing  $z^{-1}$  by  $-z^{-1}$  changes the basic building block into a bilinear differentiator  $s = (z-1)/(z+1)$ , which is the logical thing to do if the poles are grouped around  $z = -1$  instead of around  $z = +1$ . Indeed, the transfer functions (42) confirm that each state variable is made by integrating the previous or differentiating the next (which is the previous after state reversal). Note also that the sum of input and output of a delay is fed forward to the next state variable, whereas only its output is fed back to the input  $x_1[k+1]$  of the line of bilinear integrators. Using the sum in the feedback would be impossible, since this would introduce delay-free loops.

Next, we address the question of whether we may use single precision for the  $n-1$  adders that are needed to add the current and new state variables, i.e. the adders right in front of the shifts between states. If we quantize the new state right after the summation nodes  $x_i[k+1]$ , the corresponding quantization errors  $e_i[k]$  are introduced at all state variable nodes  $x_j[k+1]$  with index  $j \geq i$  instead of just at  $x_i[k+1]$ , thereby increasing the noise gain. Careful examination of the effect of the use of these single precision adders leads to a new total output noise of the scaled system to replace (44):

$$N = \frac{q^2}{12} \sum_{j=1}^n k_j \sum_{p=j}^n \sum_{q=j}^n w_{pq}. \quad (48)$$

Although the difference in noise gain is small for narrow-band filters, in practice it is preferable to use double precision for all adders. In our sixth-order example the noise increases 0.1 dB ( $G_w = 46.2790$ ) for  $\theta_c = \pi/100$  and 1 dB ( $G_w = 57.8156$ ) for  $\theta_c = \pi/10$ . Note also that the sum of the new and current state variables may overflow and thus that the pertinent adders should have one overflow bit. (The amplitude is doubled if the state is highly correlated in time as is the case for narrow-band filters.) Of course we can do without these adders altogether if the new and the current state are shifted separately (though at the cost of doubling the number of shift operations). Even so, the new state must be shifted before quantization for (44) to hold.

As a final property of our  $n$ -th order design, let us look at the stability region in the coefficient space  $\{c_i\}$ . From [1] we know that the stability region of the direct form in  $\hat{a}$ -space is a subspace of a hyper-pyramid with  $n+1$  angular points given by the columns of the Pascal matrix  $\mathbf{P}_n$ , or the columns minus the zeroth row of  $\mathbf{P}_n$  in  $\hat{a}$ -space. Using (21) to go from  $\hat{a}$  to  $\hat{c}$ , we see that in  $\hat{c}$ -space the stability region is a hyper-pyramid with angular points given by the columns of  ${}_c\mathbf{M}_b$ , or its columns minus the zeroth row in  $\hat{c}$ -space. For example for  $n = 3$  we find a stability region that is delimited by the four points

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \quad (49)$$

in the three-dimensional coefficient space  $\{c_i\}$ , cf. (10). The pyramid with these four angular points has two triangular side planes  $c_1 = -1$  and  $c_3 = 1$  that coincide with the boundary of stability. In the coefficient space  $\{a_i\}$  these correspond to  $1 - a_1 + a_2 - a_3 = 0$  and  $1 + a_1 + a_2 + a_3 = 0$  respectively, and from [1] we know that a positive sum and alternating sum of  $a_i$  are always two linear conditions for stability. For any  $n$  we can say that the stability region has

two flat sides  $c_1 = -1$  and  $c_n = 1$ , the first containing all angular points except the first, and the latter containing all angular points except the last. Moreover, the stability region always fits within the unit hyper-cube  $|c_i| < 1$  and its two flat sides are nicely parallel to the axes as parts of this unit cube. Compare this with the stability region in  $\mathbf{a}$ -space which has two ‘outliers’ (the first and last column of  $\mathbf{P}_n$ ) while its flat sides are not parallel to the axes. This difference in size, shape and orientation of the stability region explains the dramatic increase in noise gain for the direct form if  $n$  is large and the poles are close to  $z = 1$  or  $z = -1$  (equivalent to being close to an outlier).

Apart from the two critical, linear conditions  $c_1 > -1$  and  $c_n < 1$ , there are  $n - 1$  other conditions for stability following from the positivity of the principal minors of the Hurwitz matrix (16). Only one of these is critical, i.e.  $|H_{nn}| > 0$ , meaning that if we start in the stable point  $\mathbf{a} = \mathbf{0}_n$  (all poles in the origin) and vary the parameters, we need only check these three critical constraints for the system to remain stable. Note that all poles are in the origin if  $\hat{c} = 2^{-n+1}\hat{\gamma} = 2^{-n}{}_c\mathbf{M}_b\mathbf{p}_n$ , cf. (24) and (25). For the corresponding continuous-time system this means that all poles are at  $s = -1$  with  $\hat{\mathbf{b}} = 2^{-n}\mathbf{p}_n$ . Note also that the condition  $|H_{nn}| > 0$  can be expressed in terms of the  $c$ -parameters using (30), so for  $n = 3$  we find

$$(c_2 - c_1)(c_3 - c_2) - (1 + c_1)(1 - c_3) > 0 \quad (50)$$

as the third critical constraint for stability.

To conclude, let us look at the second-order case. The stability region is the triangle  $-1 < c_1 < c_2 < 1$  which is a rotated version of the standard stability triangle for the second-order direct form. Indeed, the transformation matrix  $\mathbf{T} = 2^{-n+1}ud(\mathbf{P}_{n-1})^T$  for  $n = 2$  is a rotator

$$\mathbf{T} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad \text{with} \quad \mathbf{T}^{-1} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \quad (51)$$

and the resulting structure is denoted ‘wave digital form’ in [2]. This explains the index  $w$  that is used for the matrix description  $\mathbf{A}_w, \mathbf{B}_w, \mathbf{C}_w, \mathbf{D}_w$ , though for order  $n > 2$  the structure is not a wave digital filter but does have similar properties. In wave digital filter design one starts with an analog (canonical) reference filter that uses the basic building blocks  $(1 - s)/(1 + s)$  and  $-(1 - s)/(1 + s)$ , which are lossless reflectors in wave scattering. As with the canonical design presented here, this analog filter is then translated to a digital filter with the bilinear transform leading to  $z^{-1}$  and  $-z^{-1}$  as basic building blocks in the digital domain. The similarity transform  $\mathbf{A}_w = \mathbf{T}^{-1}\mathbf{A}_d\mathbf{T}$  could be said to translate a direct form to a wave scattering form, since it could also be used on the analog direct form matrix  $\mathbf{A}_c$ . In

this sense the canonical structure of Fig. 1 could be designated a wave digital filter, though it is far easier to design.

Determining the noise gain for  $n = 2$ , we find that it cannot exceed the optimal second-order gain  $G_o$  by more than a factor 2. Since the covariance  $\mathbf{K}_w$  is diagonal, the noise gain  $\sum_{j=1}^n k_j w_{jj}$  is equal to  $\text{trace}(\mathbf{K}\mathbf{W})$ , which in turn is the sum of the positive, real eigenvalues  $\mu_1^2$  and  $\mu_2^2$  of the product matrix  $\mathbf{K}\mathbf{W}$ . These so-called second-order modes determine the optimal gain with  $G_o = \frac{1}{2}(\mu_1 + \mu_2)^2$ , so  $G_w < 2G_o$ . Ideally, we would like  $\mathbf{K}_w$  to be diagonal for any order  $n$ , because then  $G_w$  would be upper-bounded by  $nG_o = (\sum \mu_j)^2$ . From (15) we know that this is not the case however, though  $\mathbf{K}_w$  is clearly ‘more diagonal’ than  $\mathbf{K}_d$  which intuitively explains why the wave form is far superior to the direct form in terms of noise and sensitivity.

## VI. CONCLUSION

We have presented an easy way to design canonical  $n$ -th order digital filters that have low coefficient sensitivity and low quantization noise, the scaled implementation being drawn in Fig. 1. The number of multipliers used is  $2n + 1$ , the number of delays is  $n$ , the number of (power-of-two) scalers is  $2n - 1$  and the number of (double precision) adders is  $n + 1$ . In the latter it is assumed that the current and the new state variables in Fig. 1 are scaled separately in order to save  $n - 1$  adders, thereby increasing the number of data shifts (i.e. scalers) by the same amount.

## REFERENCES

- [1] J.H.F. Ritzerfeld, “On stability tests for continuous and discrete-time linear systems,” *Proc. ProRISC 2005, 16th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, Netherlands, 17-18 November 2005, ISBN 90-73461-50-2; STW, Technology Foundation, Utrecht, The Netherlands, pp. 668-673, 2005.
- [2] J.H.F. Ritzerfeld, “Noise gain expressions for low noise second-order digital filter structures,” *IEEE Trans. Circuits Syst. II - Analog Digit. Signal Process.*, vol. 52, no. 4, pp. 223-227, 2005.
- [3] J.H.F. Ritzerfeld, “The direct wave form: an easy alternative for the direct form,” *Proc. ProRISC 2004, 15th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, Netherlands, 25-26 November 2004, ISBN 90-73461-43-X; STW, Technology Foundation, Utrecht, The Netherlands, pp. 133-137, 2004.