IEEE International Symposium on Circuits and Systems (ISCAS 2018)
Florence, Italy, 27–30 May, 2018

Equiripple-Stopband Multiplierless FIR Filters by Chebyshev Sharpening of Two-Sample Averaging

Jeffrey O. Coleman
IEEE Senior Member
http://alum.mit.edu/www/jeffc

Naval Research Laboratory (retired)
Radar Division
Washington DC, USA

***Abstract*—Odd-length linear-phase prototype FIR filters were recently sharpened with scaled Chebyshev polynomials. Here the approach is extended to even-length prototypes and applied to a two-tap averaging filter, and passband normalization in either case is greatly improved for wide-stopband designs. This paper also presents an alternate sharpened-filter structure suitable for decimation or interpolation filtering or for antenna-array weighting. Either structure can generally be made multiplierless.**

I. Introduction

Copies of a prototype filter with zero-phase response $H(f)$ can be incorporated into a sharpening structure based on a polynomial $P(x)$ to obtain a zero-phase response $P(H(f))$. Early polynomial sharpening [1] sharpened substantial prototypes with polynomials of degree two or three. More recently, this author sharpened smallish prototypes with medium-degree Chebyshev polynomials [2], [3] to obtain equiripple stopbands.

This paper extends [3] in various ways. It shows how to sharpen even-length prototypes, which was not addressed by [3]. It also refines prototype scaling to correct a passband droop that was sometimes fatal, leaving no passband at all, and adapts simple passband flattening from [3] to the new scaling. This paper also explores the simplest prototype, a two-sample averager, using higher-order sharpening. This yields low-complexity multiplierless sharpened filters.

Most importantly, this paper presents an alternate sharpening structure that, while using more adders, is better suited to decimation/interpolation filtering or antenna-array element weighting. This new structure can also be made multiplierless.

This paper works with the ideas of [3] but is self-contained.

II. Sharpening

A. Frequency-Response Behaviors

1) Scaled Chebyshev Polynomials: Let us use polynomials

$$\begin{aligned} &P_0(x) = 2, \quad P_1(x) = x, \\ &P_n(x) = xP_{n-1}(x) - \alpha P_{n-2}(x) \text{ for } n > 1. \end{aligned} \tag{1}$$

These are derived in [3] by using functions of scaling parameter α, where $0 < \alpha < 1$, to scale the Chebyshev polynomials of the first kind along both axes. As the Fig. 1 examples illustrate, these polynomials have small-amplitude ripples for small x.

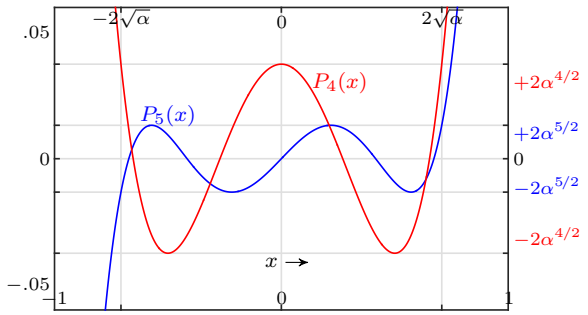


Fig. 1. Polynomials $P_4(x)$ and $P_5(x)$ for $\alpha = \frac{1}{8}$. Polynomial $P_n(x)$ ripples between $\pm 2\alpha^{n/2}$ for $|x| \le 2\sqrt{\alpha}$ and is even or odd according as n is.

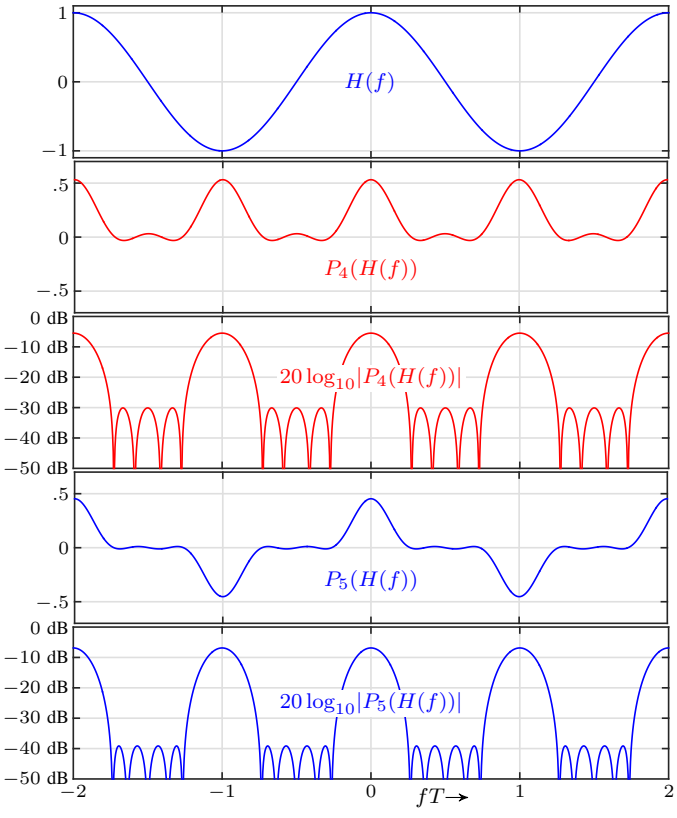


Fig. 2. Prototype response $H(f)$ and example sharpened responses $P_4(H(f))$ and $P_5(H(f))$ with $\alpha = \frac{1}{8}$ on linear and log vertical scales.

2) Sharpen the Simplest Even-Length FIR Prototype: The polynomials of (1) behave well only for real arguments, so a zero-phase prototype is required. This paper uses the simplest:

$$h(t) = \tfrac{1}{2}\big(\delta(t+T/2) + \delta(t-T/2)\big) \qquad H(f) = \cos(\pi f T). \tag{2}$$

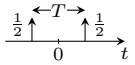


Fig. 2 shows this $H(f)$ and, on linear and dB scales, the result of sharpening it with each of the polynomials of Fig. 1.

3) Design Sequence: Polynomial $P_n(x)$ is even or odd, so $|P_n(H(f))|$ depends only on $|H(f)|$ and not on the sign of $H(f)$. In Fig. 3 then, polynomial arguments and magnitudes are shown in decibels. Varying n with constant α (left) varies $P_n(x)$ ripple amplitude $2\alpha^{n/2}$ with constant ripple cutoff $|x| = 2\sqrt{\alpha}$, while varying α with constant n (right) varies both ripple quantities. So, design first sets α to fix the ripple bandwidth of $|P_n(H(f))|$ and then fixes n to set stopband depth.

4) Optional Improved Passband Normalization: In [3] designs relied on $|P_n(\pm 1)| \approx 1$ to map prototype passbands with $|H(f)| \approx 1$ to sharpened passbands with $|P_n(H(f))| \approx 1$. But $|P_n(\pm 1)| \approx 1$ is a visibly loose approximation in Fig. 3, unless α is very small. In Fig. 2 this weakness appears as peak sharpened transfer-function magnitudes around -6 or -7 dB.

The fix is simple. An inductive proof on n using (1) shows

$$P_n(1+\alpha) = 1 + \alpha^n \tag{3}$$

for $n \geq 1$ or, for practical n and α, just $P_n(1+\alpha) \approx 1$. If we replace $H(f)$ with $(1+\alpha)H(f)$ in the math and in the sharpened filter structure (discussed below), prototype passbands where $|H(f)| \approx 1$ will map to sharpened passbands with $|P_n((1+\alpha)H(f))| = 1 + \alpha^n \approx 1$. This is illustrated in the Fig. 4 example, where without the new $1+\alpha$ factor, the very large α would have eliminated the passband completely. The sharpened response would have been all stopband!

Scaling the prototype by $1+\alpha$ to improve passband normalization is omitted below unless explicitly stated otherwise.

5) Optional Passband Flattening: Given $1+\alpha$ prototype scaling, let us flatten the sharpened passband by flattening the sharpening polynomial assuming $H(0) = 1$ and $H'(0) = 0$. Differentiating (1),

$$P_0'(x) = 0, \quad P_1'(x) = 1, \qquad P_n'(x) = xP_{n-1}'(x) + P_{n-1}(x) - \alpha P_{n-2}'(x) \text{ for } n > 1. \tag{4}$$

The latter at $x = 1+\alpha$ becomes, using (3),

$$P_n'(1+\alpha) = (1+\alpha)P_{n-1}'(1+\alpha) + 1 + \alpha^{n-1} - \alpha P_{n-2}'(1+\alpha).$$

This with (4), induction on n, and some tedious algebra proves

$$P_n'(1+\alpha) = n\,\frac{1-\alpha^n}{1-\alpha} \quad \text{for } n \geq 0 \tag{5}$$

using that $|\alpha| \neq 1$. Define new blended sharpening polynomial

$$Q_{n,m}(x) \triangleq uP_m(x) + vP_n(x) \tag{6}$$

and set u and v with $Q_{n,m}(1+\alpha) \approx 1$ and $Q_{n,m}'(1+\alpha) = 0$. At $f = 0$ then, sharpened response $Q_{n,m}((1+\alpha)H(f)) \approx 1$ with two zero derivatives, one more than $P_n(H(f))$.

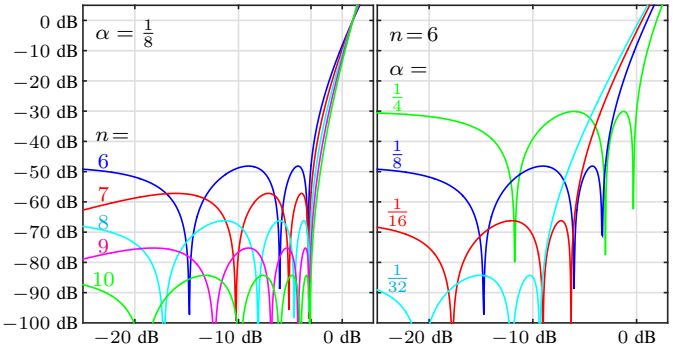


Fig. 3. Magnitude $20\log_{10}|P_n(x)|$ versus $20\log_{10}|x|$ for select n and α.

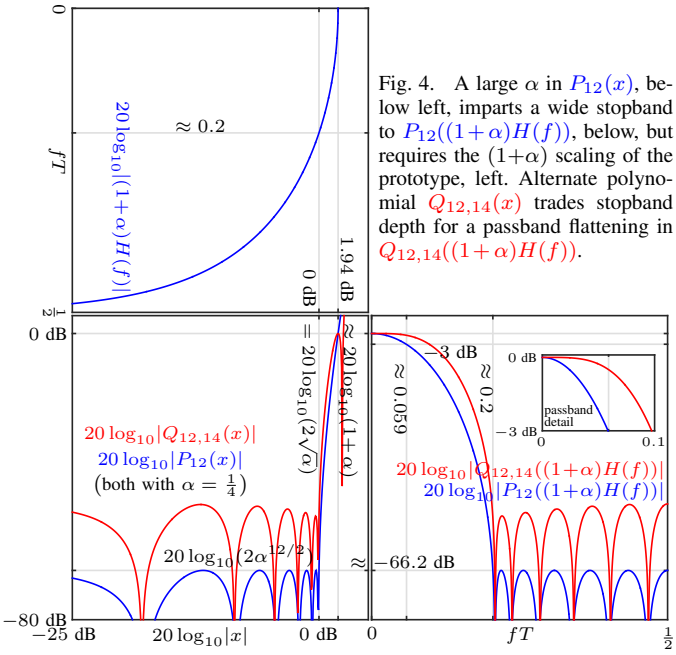


Fig. 4. A large α in $P_{12}(x)$, below left, imparts a wide stopband to $P_{12}((1+\alpha)H(f))$, below, but requires the $(1+\alpha)$ scaling of the prototype, left. Alternate polynomial $Q_{12,14}(x)$ trades stopband depth for a passband flattening in $Q_{12,14}((1+\alpha)H(f))$.

First set $u+v=1$ so that $Q_{n,m}(1+\alpha) = 1 + u\alpha^m + v\alpha^n \approx 1$ by (3). Then differentiate (6), use (5) and $u+v=1$, and solve:

$$u = \frac{1}{1 - \dfrac{m(1-\alpha^m)}{n(1-\alpha^n)}}, \qquad v = \frac{1}{1 - \dfrac{n(1-\alpha^n)}{m(1-\alpha^m)}}. \tag{7}$$

(To get one from the other: $u+v=1$.) Suppose $m > n$. Then $u < 0$ and $v > 0$ with the stopband of $Q_{n,m}(x)$ largely that of $vP_n(x)$, which dominates $uP_m(x)$ there. When $\alpha^n \ll 1$, use $\alpha = 0$ in (7) to get the simpler weights of [3], which work better with prototype scaling $1+\alpha$ here than without it there.

Realization will favor $m-n$ being small and even (for even-length prototypes), so in examples $m - n = 2$. In the Fig. 4 example, $m = 14$ and $n = 12$. As an empirical observation, the slope in stopband peaks can be tuned out by replacing the 2 initializing polynomial recursion (1) with a smaller number, say 1.4, at the cost of a slight increase in passband height.

B. Leapfrog Computational Structures

These structures from [3] are discussed in more detail here.

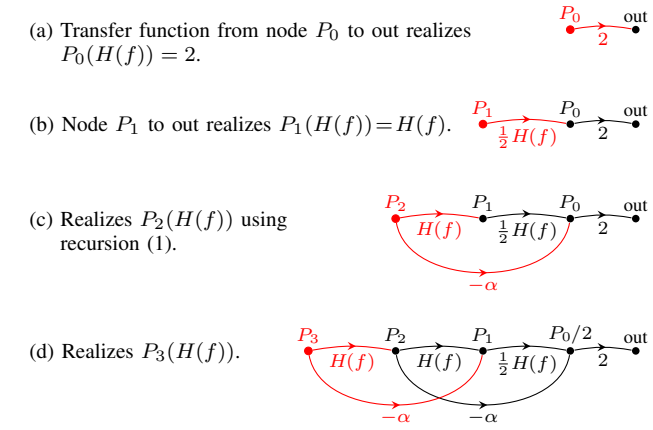


Fig. 5. Steps (a) to (d) each add material from defining recursion (1) and can be continued to construct a zero-phase realization of $P_n(H(f))$ for any n.

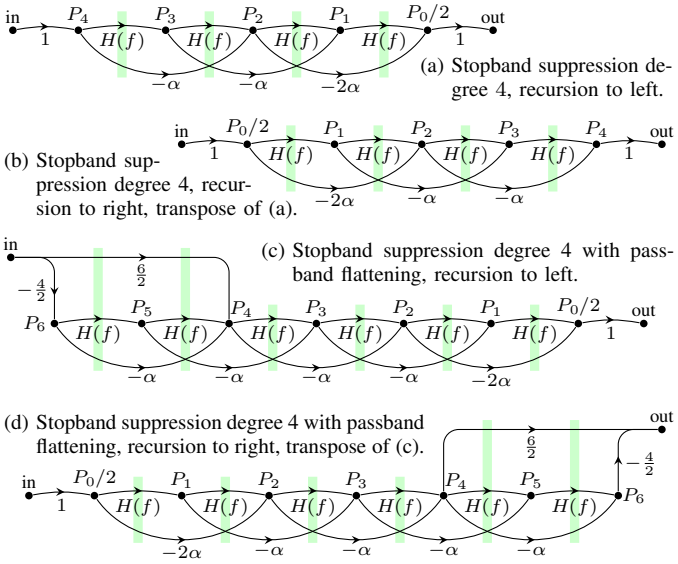


Fig. 6. Systems (a) and (c) from [3] are transposed in (b) and (d). Where recursion is to left (resp. to right), the gain to the output (resp. from the input) from the node labeled P_i is $P_i(H(f))$. In realization make zero-phase $H(f)$ branches causal with shimming delays where paths cross vertical bars.

1) Recursive Construction: Zero-phase response $P_n(H(f))$ is realized through leftward recursion in Fig. 5. Doubling node P_0 amplitude yields Fig. 6(a), where node label $P_0/2$ reflects halving its gain to the output. Transpose structure (a) to obtain (b) or construct (b) using rightward recursion to add to the output side at each step. Systems (c) and (d) use the last two recursion steps to realize optional passband flattening when α is so small that improved passband normalization can be omitted. Structure (c) matches the Fig. 5(c) example of [3].

2) Causality Shimming: Inserting identical shimming delays of arbitrary amount D wherever a particular shaded vertical bar crosses a Fig. 6 graph edge only delays the output by D. This is causality shimming if D barely makes the prototype path causal. Given our prototype, causality shimming is by $D = T/2$ on all bars and yields Fig. 7 once implicit rate $1/T$ output sampling is moved to the input with a noble identity.

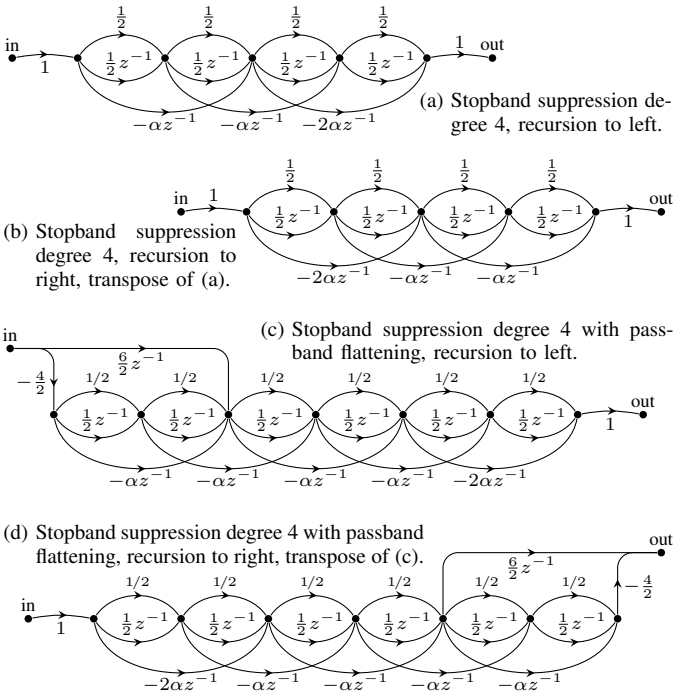


Fig. 7. The Fig. 6 systems realized with this paper's two-sample-average prototype $H(f)$, causality shimming by $T/2$, and z^{-1} denoting delay by T.

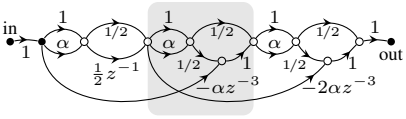


Fig. 8. Causality and pipeline shimming and improved passband normalization. Hollow nodes include z^{-1}. Use $n-2$ copies of the shaded section.

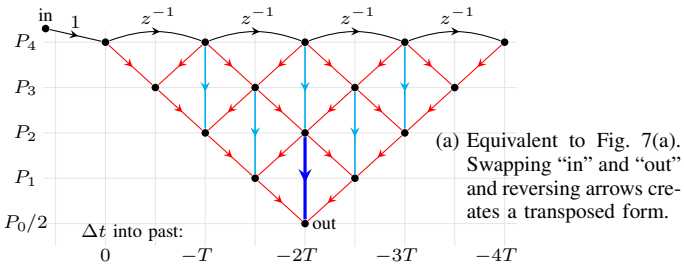
Bars cross the lower feedforward paths in pairs in zero-phase systems as in Fig. 6, so causality shimming delays paths by integral multiples of T for prototypes of even or odd lengths.

3) Multiplierless filtering: Simple prototype impulse response (2) has coefficients $\frac{1}{2}$. Other multiplies are all by α or 2α or $1+\alpha$. But setting $\alpha = 2^{-k}$ or $\alpha = 1-\beta = 1-2^{-k}$, where integer $k \geq 1$, makes those multiplies into sign-extended shifts (of two's-complement signals) or sums of two such terms. We can fairly call such filters multiplierless.

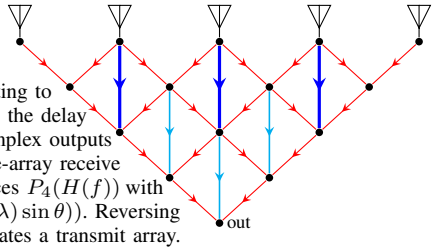
4) Arithmetic Pipelining: To shim Fig. 6(a) for arithmetic pipelining as well, further increase D by T and add signals in pairs with each adder followed by z^{-1} to register its output. If optional passband flattening is used, as in the Fig. 8 example, increase D by $2T$ instead. In Fig. 8 worst-case implementation cost for a sharpening-polynomial degree and sharpened filter order of n with $\alpha = 2^{-k}$ is seen to be $3n-1$ registered adds and $n-1$ triple delays. Using $\alpha = 1-\beta$ requires an additional registered add per leapfrog feedforward path, but $1+\alpha = 2-\beta$ in the forward path still uses a two-term structure.

C. Tree-like Computational Structures

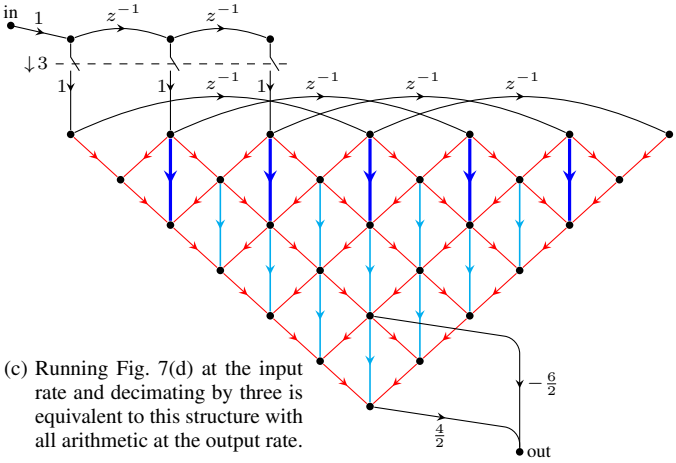
In the Fig. 9(a) tree, the node at coordinates $(\Delta t, P)$ holds the value of the Fig. 6(a) leapfrog structure's node P delayed by Δt. Ideally the delay line would take input in the center and output P_4 delayed by $-2T, -T, 0, T, 2T$, but included



(a) Equivalent to Fig. 7(a). Swapping "in" and "out" and reversing arrows creates a transposed form.



(b) Using Fig. 7(b) instead moves the -2α weighting to the first row. Replacing the delay line in (a) with the complex outputs from elements of a line-array receive antenna, as here, replaces $P_4(H(f))$ with array factor $P_4(H((d/\lambda)\sin\theta))$. Reversing arrows to transpose creates a transmit array.



(c) Running Fig. 7(d) at the input rate and decimating by three is equivalent to this structure with all arithmetic at the output rate.

Fig. 9. Tree structures based on the first three systems of Fig. 7. Delay shimming to obtain causality is not needed. Diagonal edges have weights of $\frac{1}{2}$ with each converging pair realizing $H(f)$. Light and heavy vertical edges have weights $-\alpha$ and -2α respectively.

causality shimming puts the input at the left and delays by $0, T, 2T, 3T, 4T$. The values of P_3 delayed by $\frac{1}{2}T, \frac{3}{2}T, \frac{5}{2}T, \frac{7}{2}T$ are then computed by averaging time-adjacent samples of P_4, effectively realizing the leftmost prototype $H(f)$ of Fig. 6(a) using (2). Calculation of P_2 delayed by $T, 2T, 3T$ is then by averaging time-adjacent P_3 values and adding $-\alpha$ of P_4 at the three output delays to represent the leftmost feedforward path of Fig. 6(a). Two values of P_1 and one of output $P_0/2$ are similarly computed, with the feedforward gain changed to -2α in the final computation. The output value computed is delayed by $2T$ because of the delay-line causality shimming.

The leapfrog structure's transposed form does not map to the tree structure's transposed forms. If we construct the tree as before but following the transposed leapfrog structure of Fig. 6(b), the result is Fig. 9(b) if we imagine a delay line rather than antennas (dealt with below). The trees of Figs. 9(a) and (b) then have transposes with delay lines at the output, so transposition has netted us four possible structures, not two.

The trees shown are for prototype $h(t)$ of (2), but it should be clear how to modify them for other prototypes. However, a tree based on even this small prototype has many more adders than does the nearest equivalent leapfrog structure, so trees like Figs. 9(a) and (b) and their transposes will rarely be of value using simple delay lines. Their value is in special applications.

For example, the tree structure is easily adapted to decimation and interpolation filtering. The Fig. 9(c) delay line shown incorporates decimation by three so that a tree structure computing samples at the decimator's output rate is driven by consecutive samples at the decimator's input rate. For interpolation filtering, transpose the structure and load the top delay line at the low rate but clock it out at the high rate.

As another example, the Fig. 9(b) delay line can be replaced, as shown, with complex-baseband outputs of antenna elements spaced along a line at intervals of d, which is classically fixed at half of wavelength λ. The output then has the directional pattern of a single element, embedded in the array, but scaled by array factor $P_4\big(H\big((d/\lambda)\sin\theta\big)\big)$, with angle θ from the array line's normal plane to the direction of arrival. This can be derived using [4] or any graduate antenna text. The transposed signal-flow graph gives the same pattern to a transmit array, given conversion to an amplified RF signal at each element.

III. Suggestions for Future Work

Example frequency responses here used $\alpha = 2^{-k}$ and modest n. For even wider stopbands, use $\alpha = 1 - 2^{-k}$ with optional improved passband normalization. With such large α, stopband depth increases more slowly with n, so n must be larger. (Try $\alpha = 1 - \frac{1}{8}$ with $n = 150$.) Computational complexity will increase, as even a leapfrog structure like Fig. 8 will require an extra add per section. Also, one can vary the 2 that initializes polynomial recursion (1) and appears in -2α in realization. Lowering it tapers the stopband, which can improve receive SNR or transmit power efficiency in array antennas.

As the last paragraph suggests, this paper does not systematically explore the design space and offers no comparison with other approaches, so there is much to do. Most importantly, might there be more computationally efficient sharpening structures for, say, decimation filtering, since the Fig. 9 tree structures use many adders for nontrivial filters?

References

[1] J. Kaiser and R. Hamming, "Sharpening the response of a symmetric nonrecursive filter by multiple use of the same filter," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 5, pp. 415–422, Oct. 1977.

[2] J. O. Coleman, "Chebyshev Stopbands for CIC Decimation Filters and CIC-Implemented Array Tapers in 1D and 2D," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 59, no. 12, pp. 2956–2968, Dec. 2012.

[3] ——, "Integer-coefficient FIR filter sharpening for equiripple stopbands and maximally flat passbands," in *2014 IEEE Int'l Symp. on Circuits and Systems (ISCAS 2014)*, Jun. 2014, pp. 1604–1607.

[4] ——, "Planar arrays on lattices and their FFT steering, a primer," Naval Research Laboratory, Washington DC, USA, NRL Formal Report NRL/FR/5320--11-10,207, 29 April 2011. [Online]. Available: http://www.dtic.mil/docs/citations/ADA544059