CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS
NOVEMBER 1994

DEPARTMENT OF MUSIC
REPORT NO. STAN-M-90

ON FAST FIR FILTERS IMPLEMENTED AS TAIL-CANCELING IIR FILTERS

AVERY LI-CHUN WANG AND JULIUS O. SMITH III

RESEACH SPONSORED BY
THE NATIONAL SCIENCE FOUNDATION
FELLOWSHIP PROGRAM

CCRMA
DEPARTMENT OF MUSIC
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-8180

COPYRIGHT © 1994
AVERY LI-CHUN WANG AND JULIUS O. SMITH III

On Fast FIR Filters Implemented as Tail-Canceling IIR Filters

Avery Wang, Julius O. Smith III *

Center for Computer Research in Music and Acoustics
Stanford University, Stanford, CA 94305

1 Introduction

Infinite impulse response (IIR) recursive linear digital filters are widely used because of their low computational cost and low storage overhead requirements. Finite impulse response (FIR) filters, on the other hand, allow the possibility of implementing linear-phase linear digital filters which have constant group delay across all frequencies. The tradeoff is that to achieve similar magnitude transfer functions, FIR filters usually require much larger filter orders than their IIR counterparts. For example, a general N-th order FIR filter requires $N+1$ multiplies and N adds. In certain cases, however, FIR filters may be designed which have an operation count comparable to that of an IIR filter while maintaining the linear phase property.

The set of finite impulse responses which may be efficiently implemented includes those which are truncated IIR (TIIR) sequences of low order. Although the following results were derived independently of Saramäki and Fam [1, 2], the idea of using truncated IIR filters to generate linear-phase filters was originally introduced by Fam [1]. Fam and Saramäki [1, 2] deal with unstable hidden modes due to pole-zero cancellations outside of the unit circle by employing a switching and resetting algorithm to reduce the effects of quantization error buildup. We introduce a slightly more efficient version of this idea, as well as an error analysis.

In this paper, we describe a similar algorithm for the efficient implementation of certain classes of FIR filters. We introduce an extension of the TIIR algorithm which allows the truncation of arbitrary IIR filter tails. Our algorithm allows the possibility of implementing polynomial impulse responses. Additionally, we present an analysis of the effects of limited numerical precision and provide design guidelines for designing systems with acceptable noise tolerance.

2 Definitions

A general causal N-th order FIR filter consists of a tapped delay line N elements long and a table of $N+1$ impulse response coefficients $\{h_0, \ldots, h_N\}$ such that at each time step n the output

$$y[n] = \sum_{k=0}^{N} h_k x[n-k] \tag{1}$$

is formed. The transfer function has the simple form

$$\begin{aligned} H_{\text{FIR}}(z) &\triangleq h_0 + h_1 z^{-1} + \ldots + h_N z^{-N} && (2)\\ &= z^{-N} C(z), && (3)\end{aligned}$$

*This work was supported in part by a National Science Foundation Graduate Fellowship

where $C(z)$ is the N-th degree polynomial formed by the h_k. On the other hand, a causal P-th order IIR filter has the relation

$$y[n] = -\sum_{k=1}^{P} a_k y[n-k] + \sum_{\ell=0}^{P} b_\ell x[n-\ell]. \tag{4}$$

The corresponding transfer function is

$$\begin{aligned} H_{\text{IIR}}(z) &\triangleq \frac{b_0 + b_1 z^{-1} + \ldots + b_P z^{-P}}{1 + a_1 z^{-1} + \ldots + a_P z^{-P}} && (5)\\ &= \frac{b_0 z^P + b_1 z^{P-1} + \ldots + b_P}{z^P + a_1 z^{P-1} + \ldots + a_P} && (6)\\ &= \frac{B(z)}{A(z)} && (7)\\ &= h_0 + h_1 z^{-1} + h_2 z^{-2} + \ldots, && (8) \end{aligned}$$

where both

$$A(z) = z^P + a_1 z^{P-1} + \ldots + a_P \tag{9}$$

and

$$B(z) = b_0 z^P + b_1 z^{P-1} + \ldots + b_P \tag{10}$$

may be assumed to be relatively prime P-th degree polynomials in z, and $A(z)$ is monic by construction.[1] We use the forms in Eqns. (5) and (6) interchangeably. Eqn. (8) is the expansion of the transfer function in terms of the impulse response $\{h_0, h_1, \ldots\}$. $A(z)$ is also known as the *characteristic polynomial* of the filter, and its roots, i.e., the poles, determine the filter dynamics.

Group delay [3] is defined by

$$\delta(\omega) \triangleq -\frac{d \arg\{H(e^{j\omega})\}}{d\omega}. \tag{11}$$

The group delay at normalized frequency $\omega = 2\pi f / f_s$, where f_s is the sampling frequency, is the number of samples of delay experienced by the amplitude envelope of a narrow-band input signal centered at ω.

A *linear-phase filter* is one such that the phase response at a given frequency is a linear function of frequency, i.e. $\arg\{H(e^{j\omega})\} = K_1\omega + K_2$ for some constants K_1 and K_2. From this property we see immediately that the group delay is constant for all frequencies. Filters with linear phase response are often desirable because they have no frequency-dependent temporal distortion. A stable IIR filter with non-zero poles cannot have linear phase. However, an FIR filter with coefficients $\{h_0, \ldots, h_N\}$ has linear phase if there exists a ψ such that for all $k \in \{0, \ldots, N\}$

$$h_{N-k} = e^{j\psi} h_k^*, \tag{12}$$

i.e., if the reversed coefficients are the complex conjugates of the forward sequence plus a constant phase shift. This may be seen by considering

$$\begin{aligned} H(e^{j\omega}) &= \sum_{k=0}^{N} h_k e^{-jk\omega} \\ &= \frac{1}{2}\sum_{k=0}^{N} \left(h_k e^{-jk\omega} + h_{N-k} e^{-j(N-k)\omega} \right) \end{aligned}$$

[1] A *monic* polynomial of degree N has 1 as the coefficient of the highest degree term z^N.

$$= \frac{e^{-jN\omega/2}}{2} \sum_{k=0}^{N} \left(h_k e^{-j(2k-N)\omega/2} + e^{j\psi} h_k^* e^{j(2k-N)\omega/2} \right)$$
$$= \frac{e^{-j(N\omega-\psi)/2}}{2} |h_k| \sum_{k=0}^{N} \left(e^{-j((2k-N)\omega/2-\theta_k+\psi/2)} + e^{j((2k-N)\omega/2-\theta_k+\psi/2)} \right)$$
$$= e^{-j(N\omega-\psi)/2} \sum_{k=0}^{N} |h_k| \cos\left(\frac{(2k-N)\omega + 2\theta_k + \psi}{2} \right), \qquad (13)$$

where $\theta_k \triangleq \arg(h_k)$. Thus,

$$\arg\left(H\left(e^{j\omega}\right)\right) = \frac{-N\omega + \psi}{2}, \qquad (14)$$

and the group delay is

$$\delta(f) = \frac{N}{2}. \qquad (15)$$

3 Truncated IIR (TIIR) Filters

Consider an FIR filter having a truncated geometric sequence $\{h_0, h_0p, \ldots, h_0p^N\}$ as an impulse response. This filter has the same impulse response for the first $N+1$ terms as the one-pole IIR filter with transfer function

$$H_{\mathrm{IIR}}(z) = \frac{h_0}{1 - pz^{-1}}. \qquad (16)$$

If we subtract off the tail of the impulse response we obtain

$$H_{\mathrm{FIR}}(z) = h_0 + h_0pz^{-1} + \ldots + h_0p^N z^{-N} \qquad (17)$$
$$= \left\{h_0 + h_0pz^{-1} + \ldots\right\} \qquad (18)$$
$$- \left\{h_0p^{N+1}z^{-(N+1)} + h_0p^{(N+2)}z^{-(N+2)} + \ldots\right\}$$
$$= \frac{h_0}{1-pz^{-1}} - \frac{h_0p^{N+1}z^{-(N+1)}}{1-pz^{-1}} \qquad (19)$$
$$= h_0\frac{1-p^{N+1}z^{-(N+1)}}{1-pz^{-1}}. \qquad (20)$$

The output relation for this filter is

$$y[n] = \sum_{k=0}^{N} h_0p^k x[n-k] \qquad (21)$$
$$= py[n-1] + h_0\left(x[n] - p^{N+1}x[n-(N+1)]\right). \qquad (22)$$

We see that the first formulation, Eqn. (21), requires $N+1$ multiplies and N adds to implement directly, whereas the second formulation, Eqn. (22), requires only 3 multiplies and 2 adds, independent of N. Thus we see that if we can represent an FIR sequence as a truncated exponential sequence, a tremendous savings in computation can be achieved. Note that the $x[n-(N+1)]$ term in Eqn. (22) still requires a delay line to be maintained, and thus there are no savings in storage. With modern digital signal processing (DSP) chips, however, ring buffers may be implemented with virtually no computational overhead and the full savings in computational cost may be achieved.

Notice that there is a pole-zero cancellation in the representation given by Eqn. (20). If $|p| < 1$ there is no problem since the system is inherently stable. If $|p| \geq 1$, however, then there is a potential problem due to the hidden mode. We will deal with this in Section 4 where we will see how to run TIIR filters with unstable modes.

The idea of this section was used by Fam [1] where a partial fraction expansion of a transfer function is taken and each mode is truncated separately using Eqn. (20). This method works only for cases in which the multiplicity of each pole is one, so that the each mode exhibits a simple exponential decay.

3.1 Extension to Higher-Order TIIR Sequences

We may extend the idea of the previous section for computing the TIIR sequence of any rational $H(z)$. The general procedure is to find the "tail" IIR transfer function

$$\begin{aligned} H'_{\mathrm{IIR}}(z) &= h'_0 z^{-1} + h'_1 z^{-2} + \ldots \\ &\triangleq h_{N+1} z^{-1} + h_{N+2} z^{-2} + \ldots \end{aligned} \tag{23}$$

whose impulse response, except for a time shift of N steps, matches the tail of the transfer function $H_{\mathrm{IIR}}(z)$ which we would like to truncate after time step N.

We multiply Eqn. (8) by z^N and obtain

$$z^N H_{\mathrm{IIR}}(z) = h_0 z^N + \ldots + h_{N-1} z + h_N \tag{24}$$
$$+ h_{N+1} z^{-1} + h_{N+2} z^{-2} + \ldots$$
$$= C(z) + H'_{\mathrm{IIR}}(z) \tag{25}$$
$$= \frac{z^N B(z)}{A(z)} \tag{26}$$
$$= C(z) + \frac{B'(z)}{A(z)}, \tag{27}$$

where $\mathrm{Deg}\{B'(z)\} < \mathrm{Deg}\{A(z)\} = P$. We may assume that $\mathrm{Deg}\{B'(z)\} = P-1$. $B'(z)$ is unique and may be obtained by performing synthetic division on $z^N B(z)$ by $A(z)$ and finding the remainder. Thus, $z^N B(z) \equiv B'(z) \pmod{A(z)}$.

Once we have obtained $B'(z)$ we have $H'_{\mathrm{IIR}}(z) = B'(z)/A(z)$ and we may write

$$H_{\mathrm{FIR}}(z) = H_{\mathrm{IIR}}(z) - z^{-N} H'_{\mathrm{IIR}}(z) \tag{28}$$
$$= \frac{B(z) - z^{-N} B'(z)}{A(z)} \tag{29}$$

The corresponding system is

$$\begin{aligned} y[n] = &-\sum_{k=1}^{P} a_k y[n-k] + \sum_{\ell=0}^{P} b_\ell x[n-\ell] \\ &- \sum_{m=0}^{P-1} b'_m x[n-m-(N+1)], \end{aligned} \tag{30}$$

using the representation in Eqn. (29).

The fact that the denominators of the transfer functions $H_{\mathrm{IIR}}(z)$ and $H'_{\mathrm{IIR}}(z)$ are the same allows additional savings in computational cost due to the fact that the original IIR and tail IIR dynamics are the same and do not need to be performed twice. The term $z^{-(N+1)} z^{-(P-1)} B'(z)$ serves to zero out the dynamics at the end of the delay line and requires only an additional P multiplies and $P-1$ adds. The computational cost of this general truncated P-th order IIR system is $3P+1$ multiplies and $3P-2$ adds, independent of N. Thus, a net computational savings with this class of FIR filters is achieved if $N > 3P$.

The storage costs for this filter are P output samples for the IIR feedback dynamics, N input samples of the FIR filter, and an additional P input samples for the tail-cancellation dynamics, yielding $N+P$ input delay samples, of which only the first and last P are used, and P output delay samples. Thus, the fast FIR algorithm actually requires $2P$ more storage samples than a direct FIR implementation.

As in the previous section, we observe cautiously that the effect of subtracting the tail IIR response $z^{-N}H'_{\text{IIR}}(z)$ from $H_{\text{IIR}}(z)$ is to cancel all the poles in Eqn. (29), which is to be expected since an FIR filter is an all-zero filter.

3.2 Other Architectures

The direct implementation specified by Eqn. (30) may not be desirable for various reasons. For example, one may choose to use a factored structure such as the cascaded biquad or the parallel partial fraction form.

The basic biquad factorization for a $P = 2Q$-th degree transfer function is given by

$$H(z) = b_0 \prod_{k=1}^{Q} \frac{1 + b_{1,k}z^{-1} + b_{2,k}z^{-2}}{1 + a_{1,k}z^{-1} + a_{2,k}z^{-2}}. \tag{31}$$

Each sample is calculated in a cascade

$$x_0[n] = b_0 x[n], \tag{32}$$

$$\begin{aligned} x_k[n] &= x_{k-1}[n] - a_{1,k}x_k[n-1] - a_{2,k}x_k[n-2] \\ &\quad + b_{1,k}x_{k-1}[n-1] + b_{2,k}x_{k-1}[n-2], \quad k \geq 1, \text{ and} \end{aligned} \tag{33}$$

$$y[n] = x_Q[n], \tag{34}$$

thus requiring $4Q+1 = 2P+1$ multiplies and $4Q = 2P$ adds per sample. Biquad structures are especially suited for the efficient and numerically stable computation of large-order IIR filters which often have large coefficients when left unfactored. Additionally, complex-conjugate pole pairs may be conveniently grouped together. Since the TIIR filter structure does not lend itself to biquad factorization, the form in Eqn. (28) must be taken and the recursive dynamics must be computed twice, resulting in a computational cost of $8Q+1 = 4P+1$ multiplies and $8Q+1 = 4P+1$ adds per sample. Thus, this realization is somewhat more computationally expensive than the direct realization.

We may also factor $H(z)$ modally into a partial fraction expansion:

$$H(z) = \sum_{k=1}^{N_p} \sum_{\ell=1}^{M_k} \frac{C_{k,\ell}}{(1 - p_k z^{-1})^{\ell}}, \tag{35}$$

where N_p is the number of distinct poles, and M_k is the multiplicity of the k-th pole. The (k,ℓ) partial fraction gives rise to a filter which has as impulse response

$$h_{k,\ell,n} = C_{k,\ell} \binom{n+\ell-1}{\ell-1} p_k^n. \tag{36}$$

To form the TIIR filter, a tail IIR filter is derived for each partial fraction using synthetic division as outlined in Section 3.1. Each TIIR response is calculated separately, and the results are added together to form the complete response. The factorization need not be as complete as outlined in Eqn. (35). one may choose an intermediate level of factorization, leaving some factors lumped together and others separated from each other. For example, one may want to group complex-conjugate pairs together, thus avoiding complex arithmetic.

Alternatively, one may wish to leave terms with the same poles together as in

$$H(z) = \sum_{k=1}^{N_p} \frac{B_k(z)}{(z-p_k)^{M_k}}, \tag{37}$$

since calculating the tail IIR response for each n-th order multiplicity term yields a degree $n-1$ polynomial numerator anyway. The impulse response of the k-th partial fraction in this case is

$$h_{k,n} = \sum_{\ell=0}^{M_k} b_{k,\ell} \binom{n-\ell+M_k-1}{M_k-1} p_k^{n-\ell} \tag{38}$$

where the $b_{k,\ell}$, $\ell = 0, \ldots, M_k$ are the coefficients of $B_k(z)$.

Another example is to group together the stable factors, i.e., with poles p_k such that $|p_k| < 1$, and implement the unstable poles with $|p_k| < 1$ separately. These strategies will become useful in Section 5.

4 Unstable Hidden Modes

$n \equiv 0 \pmod N$	$n \not\equiv 0 \pmod N$
Primary TIIR Filter	
$\begin{aligned} w_1[N+k] &\leftarrow 0, \quad k=1,\ldots,P-1 \\ w_1[k] &\leftarrow w_1[k-1], \quad k=N,\ldots,1 \\ w_1[0] &\leftarrow x[n] \\ q_1[k] &\leftarrow q_2[k-1], \quad k=P,\ldots,1 \\ q_1[0] &\leftarrow -\sum_{k=1}^{P} a_k q_1[k] + \sum_{\ell=0}^{P} b_\ell w_1[\ell] \\ y[n] &\leftarrow q_1[0] \end{aligned}$	$\begin{aligned} w_1[k] &\leftarrow w_1[k-1], \quad k=N+P,\ldots,1 \\ w_1[0] &\leftarrow x[n] \\ q_1[k] &\leftarrow q_1[k-1], \quad k=P,\ldots,1 \\ q_1[0] &\leftarrow -\sum_{k=1}^{P} a_k q_1[k] + \sum_{\ell=0}^{P} b_\ell w_1[\ell] \\ &\quad -\sum_{m=0}^{P-1} b'_m w_1[m+N+1] \\ y[n] &\leftarrow q_1[0] \end{aligned}$
Auxiliary TIIR Filter	
$\begin{aligned} w_2[k] &\leftarrow 0, \quad k=1,\ldots,P-1 \\ w_2[0] &\leftarrow x[n] \\ q_2[k] &\leftarrow 0, \quad k=1,\ldots,P-1 \\ q_2[0] &\leftarrow b_0 w_2[0] \end{aligned}$	$\begin{aligned} w_2[k] &\leftarrow w_2[k-1], \quad k=P,\ldots,1 \\ w_2[0] &\leftarrow x[n] \\ q_2[k] &\leftarrow q_2[k-1], \quad k=P,\ldots,1 \\ q_2[0] &\leftarrow -\sum_{k=1}^{P} a_k q_2[k] + \sum_{\ell=0}^{P} b_\ell w_2[\ell] \end{aligned}$

Table 1: Fast FIR algorithm for an unstable Truncated IIR (TIIR) filter.

We now address the issue of pole-zero cancellation and the resulting hidden modes in the fast FIR algorithm. Although the naïve Fast FIR algorithm of Eqn. (30) works in theory there is the practical matter of quantization error due to finite register lengths when dealing with truncated unstable IIR systems, i.e., those with poles p_k such that $|p_k| > 1$. Consider the system in equation (22). We may

$n \equiv 0 \pmod{N}$	$n \not\equiv 0 \pmod{N}$
Primary TIIR Filter	
$\begin{aligned} w_1[N+k] &\leftarrow 0, \quad k = 1, \ldots, P-1 \\ w_1[k] &\leftarrow w_1[k-1], \quad k = N, \ldots, 1 \\ w_1[0] &\leftarrow x[n] \\ q_1[k] &\leftarrow q_2[k-1], \quad k = P, \ldots, 1 \\ q_1[0] &\leftarrow -\sum_{k=1}^{P-1} \frac{a_k^* q_1[P-k]}{a_P^*} - \frac{q_1[P]}{a_P^*} \\ &\qquad + \sum_{m=0}^{P-1} \frac{b_m'^* w_1[P-1-m]}{a_P^*} \end{aligned}$	$\begin{aligned} w_1[k] &\leftarrow w_1[k-1], \quad k = N+P, \ldots, 1 \\ w_1[0] &\leftarrow x[n] \\ q_1[k] &\leftarrow q_1[k-1], \quad k = P, \ldots, 1 \\ q_1[0] &\leftarrow -\sum_{k=1}^{P-1} \frac{a_k^* q_1[P-k]}{a_P^*} - \frac{q_1[P]}{a_P^*} \\ &\qquad + \sum_{m=0}^{P-1} \frac{b_m'^* w_1[P-1-m]}{a_P^*} \\ &\qquad - \sum_{\ell=0}^{P} \frac{b_\ell^* w_1[N+P-\ell]}{a_P^*} \\ y[n] &\leftarrow q_1[0] \end{aligned}$
Auxiliary TIIR Filter	
$\begin{aligned} w_2[k] &\leftarrow 0, \quad k = 1, \ldots, P-1 \\ w_2[0] &\leftarrow x[n] \\ q_2[k] &\leftarrow 0, \quad k = 1, \ldots, P-1 \\ q_2[0] &\leftarrow \frac{b_{P-1}'^* w_2[0]}{a_P^*} \end{aligned}$	$\begin{aligned} w_2[k] &\leftarrow w_2[k-1], \quad k = P, \ldots, 1 \\ w_2[0] &\leftarrow x[n] \\ q_2[k] &\leftarrow q_2[k-1], \quad k = P, \ldots, 1 \\ q_2[0] &\leftarrow -\sum_{k=1}^{P-1} \frac{a_k^* q_2[P-k]}{a_P^*} - \frac{q_2[P]}{a_P^*} \\ &\qquad + \sum_{m=0}^{P-1} \frac{b_m'^* w_2[P-1-m]}{a_P^*} \end{aligned}$

Table 2: Fast FIR algorithm for a reversed unstable Truncated IIR (TIIR) filter derived from a stable TIIR filter.

model quantization error as an independent, identically distributed (IID) noise signal $\nu[n]$ with zero mean and variance σ_ν^2 input to the system of Eqn. (22).[2] The output equation is then

$$y[n] = py[n-1] + h_0\left(x[n] - p^{N+1}x[n-(N+1)]\right) + \nu[n] \tag{39}$$

$$= h_0 \sum_{k=0}^{N-1} p^k x[n-k] + \sum_{k=0}^{n} p^k \nu[n-k]. \tag{40}$$

Thus, the accumulated noise has zero mean, but its variance grows as

$$\sigma_{\text{acc}}^2[n] = \sum_{k=0}^{n} |p|^{2k}\sigma_\nu^2 \tag{41}$$

$$= \sigma_\nu^2 \frac{1-|p|^{2(n+1)}}{1-|p|^2} \tag{42}$$

so that

$$\lim_{n\to\infty} \sigma_{\text{acc}}^2[n] = \begin{cases} +\infty, & |p| \geq 1 \\ \dfrac{\sigma_\nu^2}{1-|p|^2}, & |p| < 1. \end{cases} \tag{43}$$

For a higher-order system, as described in Section 3.1, this analysis gives an estimate of the noise accumulation behavior if p is the largest-magnitude pole since its dynamics dominates the behavior of the system. A direct implementation of an FIR system, such as in equation (21) does not have noise accumulation problems because of its finite memory. We trade computational cost for noise sensitivity in TIIR systems.

Recalling the discussion in Section 3.2, we may factor the unstable modes of a system apart from the transfer function so that

$$H(z) = H^{\downarrow}(z) + H^{\uparrow}(z) \tag{44}$$

$$= \frac{B^{\downarrow}(z)}{A^{\downarrow}(z)} + \frac{B^{\uparrow}(z)}{A^{\uparrow}(z)}, \tag{45}$$

where the "$\downarrow$" and "$\uparrow$" superscripts denote "stable hidden modes" and "unstable hidden modes," respectively. Thus, we may implement the two filters of orders $P^{\downarrow}$ and $P^{\uparrow}$, where $P^{\downarrow} + P^{\uparrow} = P$, and add together the outputs. $H^{\downarrow}(z)$ may be implemented as described in Section 3.1, but $H^{\uparrow}(z)$ must be handled carefully.

Alternatively, we may factor the transfer function into stable and unstable parts

$$H(z) = H^{\downarrow}(z)H^{\uparrow}(z) \tag{46}$$

$$= \frac{B^{\downarrow}(z)}{A^{\downarrow}(z)}\frac{B^{\uparrow}(z)}{A^{\uparrow}(z)}, \tag{47}$$

in which case the stable and unstable filters are cascaded. Note that in this factorization $B^{\downarrow}(z)$ and $B^{\uparrow}(z)$ are not uniquely determined.

A possible algorithm for stabilizing an unstable TIIR filter using two parallel sets of state variables is given in Table 1. The algorithm cycles with a period of N time steps. Given a cycle number k, at time step $n = kN$ the auxiliary filter's input delay line $w_2[\cdot]$ of length P and P state variables $q_2[\cdot]$

[2]Finite-register effects may be modeled as the sum of q uniformly distributed IID $[-\epsilon, +\epsilon]$, where $\epsilon > 0$ is the smallest quantity such that $x + \epsilon \neq x$ in machine arithmetic, and q is the number of additive terms. If q is relatively large, the total noise may be modeled as a Gaussian distribution. For fixed-point arithmetic ϵ is constant $\forall x$. Floating point arithmetic presents difficulties since the effective ϵ varies proportionally to the magnitude of x. Assuming independence and fixed-point arithmetic, we may model quantization noise as a Gaussian random variable with variance $\sigma_\nu^2 = q\epsilon^2/3$. A thorough analysis of quantization noise statistics is beyond the scope of this paper.

are cleared to zero. Otherwise the primary and auxiliary systems evolve with the same dynamics. We have taken into account the fact that during the first N time steps the tail canceling terms due to $z^{-N}B'(z)$ in the auxiliary filter are identically zero, hence those terms are not included. Since the FIR impulse response has length $N+1$ we see that at time $n = kN + N$ the two systems have identical output values, except for the amount of accumulated noise. The state variables $q_1[\cdot]$ and $q_2[\cdot]$ are not identical because the input histories are slightly different. However, since we are only concerned with the output, this does not matter. We then copy the auxiliary system's state variables to the primary system and repeat the cycle. This algorithm may be optimized by realizing that the values for $w_2[m]$ are zero for $m > N$ during each cycle. Thus the last summation in the dynamics for the auxiliary system shown in Table 1 is omitted for the cases where $n \equiv 1, \ldots, N \pmod N$.

Under this scheme the amount of time that quantization errors can accumulate varies from a minimum of N time steps to a maximum of $2N$, depending on when each error occurs. The minimum is due to the amount of time necessary to allow the auxiliary system to evolve to having the same output value as the primary system. For noise accumulation analysis in this system, we may assume that the unstable system dynamics are dominated by the mode with the largest eigenvalue $p_{\max}$ such that $|p_{\max}| > 1$. We assume for the present analysis that the multiplicity of $p_{\max}$ is one. Thus we must reckon for tolerable performance with

$$\sigma^2_{\max} \geq \sigma^2_\nu \frac{1 - |p_{\max}|^{4N}}{1 - |p_{\max}|^2}. \tag{48}$$

If the maximum tolerable noise variance is σ^2_{tol} then we must have $\sigma^2_{\max} < \sigma^2_{\text{tol}}$ so that, approximately,

$$N < \frac{\log\left(1 + \frac{(|p_{\max}|^2 - 1)\sigma^2_{\text{tol}}}{\sigma^2_\nu}\right)}{4\log(|p_{\max}|)}. \tag{49}$$

Thus, given $p_{\max}$, σ^2_{tol}, and quantization noise variance σ^2_ν, we have a fundamental limitation on the length of the effective impulse response. A refinement of this bound is given below in Section 5.3.

The computational cost for implementing an unstable TIIR filter is at most twice the cost of a stable TIIR filter. The extra cost is due to the cost of implementing the auxiliary TIIR filter, which costs about $2P$ operations per sample. Thus the computational cost of the unstable TIIR filter is approximately $5P$ multiples per input sample. The number of adds is about the same. If we take into account the fact that the feed-forward terms due to $B(z)$ are the same (except for cases where $w_2[k]$ has been set to zero) in both the primary and auxiliary systems, we may optimize by avoiding duplicate calculation of these terms and thus reduce the total number of multiplies and adds to about $4P$ per input sample. The shifts of the state variables and input delay lines may be simulated by using pointer arithmetic, and need not actually be performed.

5 Fast Linear-Phase FIR Filters

The implementation of filters as TIIR systems is rather pointless unless there is an advantage that is unavailable to untruncated IIR filters. That advantage is that it is possible to implement exactly linear-phase filters. We present two basic strategies for designing filters based on TIIR filters. The first uses the factorization given in Eqn. (44), and the second uses the factorization used in Eqn. (46).

5.1 Additive Factorization Design Method

We saw in Section 2 that an FIR filter has linear phase if Eqn. (12) holds. It is possible to attain such a relation using TIIR filters. Let $H^+_{\mathrm{FIR}}(z)$ be a TIIR transfer function such that

$$H^+_{\mathrm{FIR}}(z) = \sum_{k=0}^{N} h^+_k z^{-k} \tag{50}$$

$$= \frac{B^+(z) - z^{-N} B'^+(z)}{A^+(z)}. \tag{51}$$

We form the time-reversed truncated transfer function

$$H^-_{\mathrm{FIR}}(z) = \sum_{k=0}^{N} h^-_k z^{-k} \tag{52}$$

$$= \sum_{k=0}^{N} h^{+*}_k z^{k-N} \tag{53}$$

$$= z^{-N} \left\{ H^+_{\mathrm{FIR}}(1/z^*) \right\}^* \tag{54}$$

$$= \frac{z^{-N} \left\{ B^+(1/z^*) \right\}^* - \left\{ B'^+(1/z^*) \right\}^*}{\left\{ A^+(1/z^*) \right\}^*} \tag{55}$$

$$= \frac{z^{-N} z^{-P} B^-(z) - z^{-(P-1)} B'^-(z)}{z^{-P} A^-(z)} \tag{56}$$

$$= \frac{-z B'^-(z) + z^{-N} B^-(z)}{A^-(z)}, \tag{57}$$

where the "+" and "−" superscripts denote "forward" and "reverse-conjugated" filter respectively, and the "*" superscript denotes complex conjugation, as usual. Thus, comparing with Eqns. (9) and (10), we have

$$A^-(z) = 1 + a_1^* z + \ldots + a_P^* z^P, \tag{58}$$

$$B^-(z) = b_0^* + b_1^* z + \ldots + b_P^* z^P, \tag{59}$$

and

$$B'^-(z) = b_0'^* + b_1'^* z + \ldots + b_{P-1}'^* z^{P-1}. \tag{60}$$

We have assumed that $B^+(z)$ and $B'^-(z)$ have degrees P and $P-1$, respectively. If we assume that $H^+_{\mathrm{FIR}}(z)$ is a stable TIIR filter then $H^-_{\mathrm{FIR}}(z)$ is an unstable TIIR filter whose hidden modes are conjugate-reciprocals of those of $H^+_{\mathrm{FIR}}(z)$. An example implementation for $H^-_{\mathrm{FIR}}(z)$ is given in Table 2. There we have normalized the numerator and denominator of the filter by dividing through by a_P^*.

Using the arbitrary time shift $M \geq 0$ and phase shift ψ we define the filter

$$H_{\mathrm{LPFIR}}(z) \triangleq H^+_{\mathrm{FIR}}(z) + e^{j\psi} z^{-M} H^-_{\mathrm{FIR}}(z) \tag{61}$$

$$= \sum_{k=0}^{N} h^+_k z^{-k} + e^{j\psi} \sum_{k=0}^{N} h^{+*}_k z^{k-N-M} \tag{62}$$

We note that this new FIR filter of length $M + N + 1$ is invariant with respect to reversing the order, conjugating the coefficients, and multiplying by the phase factor $\exp(j\psi)$. Eqn. (12) holds, and thus $H_{\mathrm{LPFIR}}(z)$ is a linear-phase FIR filter. We may see the linear-phase property more directly by first noticing that, on the unit circle,

$$H^-_{\mathrm{FIR}}\left(e^{j\omega}\right) = e^{-jN\omega} \left\{ H^+_{\mathrm{FIR}}\left(e^{j\omega}\right) \right\}^*, \tag{63}$$

so that

$$H_{\text{LPFIR}}\left(e^{j\omega}\right) = H_{\text{FIR}}^{+}\left(e^{j\omega}\right) + e^{j\psi}e^{-jM\omega}H_{\text{FIR}}^{-}\left(e^{j\omega}\right) \tag{64}$$

$$= H_{\text{FIR}}^{+}\left(e^{j\omega}\right) + e^{j\psi}e^{-j(N+M)\omega}\left\{H_{\text{FIR}}^{+}\left(e^{j\omega}\right)\right\}^{*} \tag{65}$$

$$= e^{-j((N+M)\omega+\psi)/2}\text{Re}\left\{e^{j((N+M)\omega+\psi)/2}H_{\text{FIR}}^{+}\left(e^{j\omega}\right)\right\}. \tag{66}$$

Eqn. (15) gives us the result

$$\delta = \frac{M+N}{2}. \tag{67}$$

It is rather difficult to design fast FIR filters to a given set of specifications using this additive factorization design method because it is not intuitively obvious how to control the magnitude $\left|H_{\text{LPFIR}}\left(e^{j\omega}\right)\right|$ due to Eqn. (66). Nonetheless, for certain impulse response waveforms which are well characterized, this technique provides a useful tool for designing corresponding fast FIR filters to realize them. Some examples are given in Section 6.2.

5.2 Magnitude-Squared Design Method

We begin by choosing a desired non-negative real-valued transfer function $H^2(z) > 0$ for which there exists a stable transfer function $H(z)$ such that

$$H^2\left(e^{j\omega}\right) \triangleq \left|H\left(e^{j\omega}\right)\right|^2 \tag{68}$$

$$= H\left(e^{j\omega}\right)^{*}H\left(e^{j\omega}\right) \tag{69}$$

and polynomials $A^+(z)$ and $B^+(z)$ such that

$$H(z) = \sum_{k=0}^{\infty} h_k z^{-k} \tag{70}$$

$$= \frac{B^+(z)}{A^+(z)}. \tag{71}$$

If the coefficients of $A^+(z)$ and $B^+(z)$ are real then Eqn. (68) yields

$$H^2\left(e^{j\omega}\right) = H\left(e^{j\omega}\right)H\left(e^{-j\omega}\right). \tag{72}$$

We form the TIIR transfer function

$$H_{\text{FIR}}^{+}(z) = \sum_{k=0}^{N} h_k z^{-k} \tag{73}$$

$$= \frac{B^+(z) - z^{-N}B'^+(z)}{A^+(z)}, \tag{74}$$

as in Eqn. (51). Similarly, we form $H_{\text{FIR}}^{-}(z)$, as in Eqn. (57). We see immediately, in light of Eqn. (63) that the filter

$$H_{\text{FIR}}^{2}(z) \triangleq H_{\text{FIR}}^{+}(z)H_{\text{FIR}}^{-}(z) \tag{75}$$

has the property that

$$H_{\text{FIR}}^{2}\left(e^{j\omega}\right) = e^{-jN\omega}\left|H_{\text{FIR}}^{+}\left(e^{j\omega}\right)\right|^2, \tag{76}$$

and this is obviously a linear-phase filter with group delay

$$\delta = N. \tag{77}$$

This filter may be simply implemented as a cascade of $H^+_{\mathrm{FIR}}(z)$ and $H^-_{\mathrm{FIR}}(z)$, which may be implemented by Eqn. (30) and Table 2, respectively.

The relationship between $H^2_{\mathrm{FIR}}(z)$ and $H^2(z)$ is seen by considering the cyclic convolution [4]

$$H^+_{\mathrm{FIR}}\left(e^{j\omega}\right) = \sum_{k=0}^{\infty} w_k h_k e^{-ik\omega} \tag{78}$$

$$= \frac{1}{2\pi}\int_0^{2\pi} H\left(e^{j\theta}\right) W_N\left(e^{j(\omega-\theta)}\right) d\theta \tag{79}$$

where

$$w_k = \begin{cases} 1, & -N \le k \le N \\ 0, & \text{otherwise,} \end{cases} \tag{80}$$

and

$$W_N(z) = \sum_{k=-N}^{N} z^k \tag{81}$$

$$= \frac{z^{-N} - z^{N+1}}{1-z} \tag{82}$$

so that

$$W_N\left(e^{j\omega}\right) = \frac{\sin\{(N+1/2)\omega\}}{\sin(\omega/2)} \tag{83}$$

The extension of the window to negative values does not affect the result and is made to simplify the periodic convolution by eliminating the phase modulation of the window transfer function $W_N(z)$ as well as narrowing its main lobe by a factor of two. Thus,

$$H^2_{\mathrm{FIR}}\left(e^{j\omega}\right) = e^{-jN\omega}\left|(W_N * H)\left(e^{j\omega}\right)\right|^2, \tag{84}$$

where the "$*$" is understood to be periodic convolution as in Eqn. (79).

We notice that the phase of the filter $H(z)$ utilized in the design of $H^2_{\mathrm{FIR}}(z)$ is irrelevant, and thus, IIR filters which are usually considered to have excessive phase distortion near sharp cutoffs may be used. Additionally, the fact that the magnitude response for $H^2(z)$ is twice the distance from 0 dB compared to the magnitude response of $H(z)$ implies that constraints in the stop band of $H(z)$ need only be half the desired dB design specification, whereas, on the other hand, the ripples in the passband of $H(z)$ must be better than half the desired dB design specification. Thus, any stable discrete-time IIR filter design such as elliptic, Chebyshev, and Butterworth filters may be chosen as $H(z)$ in Eqn. (68) and transformed into a magnitude-squared filter with linear phase. The filter length N for $H^+_{\mathrm{FIR}}(z)$ and $H^-_{\mathrm{FIR}}(z)$ must be chosen long enough so that the blurring induced by the periodic convolution of $H\left(e^{j(\omega-\theta)}\right)$ by $W_N\left(e^{j(\omega-\theta)}\right)$ does not induce too much distortion in the frequency response. Indeed, disregarding quantization noise for the moment, as N grows to infinity, $W_N\left(e^{j\omega}\right)$ approaches an impulse centered at $\omega = 0$, and $H^+_{\mathrm{FIR}}(z)$ converges to $H(z)$. There are constraints due to Eqn. (49) on the filter length which are considered in the next section.

5.3 A Refined Truncation Algorithm

In both the additive and multiplicative designs of the previous two sections, the length N is constrained by the growth of quantization error in the unstable, reverse-conjugated filter $H^-_{\mathrm{FIR}}(z)$. This bound is given in Eqn. (49). We note that it is the most *stable* hidden mode of $H^{+}_{\mathrm{FIR}}(z)$ which gives rise to the most unstable hidden mode of $H^{-}_{\mathrm{FIR}}(z)$ due to the conjugate-reciprocal correspondence between their modes. These problematic conjugate-reciprocal modes may restrict the direct implementation in Table 2 to undesirably short lengths. One way around this problem is to set a significance floor λ_S above the floor for numerical precision λ_P so that errors with magnitude less than λ_S are insignificant, and quantization errors are bounded by λ_P. Let $H(z)$ be the untruncated IIR impulse response associated with $H^{+}_{\mathrm{FIR}}(z)$ as in Eqn. (71). Since $H(z)$ is stable, we have

$$|p_k| < 1, \tag{85}$$

where the p_k's are the poles of $H(z)$, and consequently the hidden modes of $H^{+}_{\mathrm{FIR}}(z)$. The hidden modes of $H^{-}_{\mathrm{FIR}}(z)$ are thus $1/p_k^*$ and are unstable. We perform a partial fraction expansion on $H(z)$ as in Eqns. (37) and (38). We then observe the impulse response for each partial fraction. We set the cutoff point N_k for the k-th partial fraction response to be the smallest time after which the maximum impulse response becomes insignificant, i.e., $\forall n > N_k$, $|\mu h_{k,n}| \leq \lambda_S$, where μ is the largest magnitude input. We may solve

$$|h_{k,N_k}| = \lambda_S/\mu \tag{86}$$

numerically for N_k by using Eqn. (38) or by using the approximation

$$\begin{aligned} h_{k,n} &\approx \sum_{\ell=0}^{M_k} \frac{b_{k,\ell}(n-\ell)^{M_k-1}p_k^{n-\ell}}{(M_k-1)!} && (87)\\ &\approx n^{M_k-1}p_k^n \sum_{\ell=0}^{M_k} \frac{b_{k,\ell}p_k^{-\ell}}{(M_k-1)!} && (88)\\ &= B_k n^{M_k-1}p_k^n && (89)\end{aligned}$$

for large n, where B_k is implicitly defined by Eqns. (88) and (89). For $M_k = 1$, we have exactly

$$N_k = \frac{\log\left(\left|\frac{\lambda_S}{\mu B_k}\right|\right)}{\log(|p_k|)}. \tag{90}$$

Thus, if $N_k < N$ we may truncate the k-th partial fraction response at N_k instead of N without losing significance. However, since $H(z)$ is stable and responses due to the k-th partial fraction beyond the N_k-th time step are below the significance floor, this does not make any difference. The refinement comes from implementing $H^{-}_{\mathrm{FIR}}(z)$ as a sum of reversed partial fractions based on Eqn. (37) and Eqn. (57) but with the k-th partial fraction truncated after $n = N_k$ samples instead of $n = N$ if $N_k < N$. Thus, the unstable mode of $H^{-}_{\mathrm{FIR}}(z)$ due to the pole at $1/p_k^*$ only needs to grow from having an initial magnitude above the significance floor λ_S and has less time to accumulate exponentially growing quantization noise. We have

$$H^{-}_{\mathrm{FIR}}(z) = \sum_{k=1}^{N_p} \frac{-zB_k'^{-}(z) + z^{-N_k'}B_k^{-}(z, N_k')}{(1-p_k^* z)^{M_k}}, \tag{91}$$

where

$$N_k' = \begin{cases} N_k, & N_k \leq N \\ N, & N_k > N. \end{cases} \tag{92}$$

Eqn. (48) indicates how much quantization noise will accumulate due to the truncated response of length $N = N_k$ of the k-th partial fraction of $H^{-}_{\mathrm{FIR}}(z)$. Assuming that quantization error occurs on the order of the precision floor λ_P, we have

$$\sigma_v^2 \approx \lambda_P^2. \tag{93}$$

Also, the significance floor sets a convenient level of noise tolerance, so that may set

$$\sigma_{\text{tol}}^2 = \lambda_S^2. \tag{94}$$

We may recast Eqn. (48) as

$$\lambda_S^2 \geq \lambda_P^2 \frac{1 - |p_k|^{-4N_k'}}{1 - |p_k|^{-2}} \tag{95}$$

$$\approx \lambda_P^2 \frac{|p_k|^{-4N_k'}}{|p_k|^{-2} - 1} \tag{96}$$

and thus

$$\lambda_P \leq \lambda_S |p_k|^{2N_k'-1} \sqrt{1 - |p_k|^2}. \tag{97}$$

Combining Eqns. (97) and (90), assuming $M_k = 1$, we arrive at

$$\lambda_P \leq \frac{\lambda_S^3 \sqrt{1 - |p_k|^2}}{\mu^2 |p_k| |B_k|^2}. \tag{98}$$

We see that the precision, in bits, for the state variables must be about 3 times greater than the precision of the data in order to prevent significant noise accumulation. This result is intuitive since the significant part of the impulse response must span the dynamic range specified by the largest magnitude output (which may be assumed to be normalized to 1) and λ_S over a period of N_k' samples, whereas the noise in $H_{\text{FIR}}^-(z)$ may grow over a period of up to $2N_k'$ samples using the same dynamics. The preceding analysis is not generally applicable when dealing with floating point quantities because the range of significance varies with the exponent. Assuming a constant-energy input, we see that the analysis still holds since the significance and precision floors are approximately constant at steady-state.

5.3.1 Unit-Magnitude Mode TIIR filters

A third class of linear-phase filters consisting of pure TIIR responses is available. If we choose $H_u(z)$ to be a TIIR filter of length N such that all of its (hidden) modes are on the unit circle and each mode has multiplicity one, then we are assured that $H_u(z)$ will have linear phase. This may be seen by observing that such a filter must satisfy Eqn. (12). If there are modes with multiplicity greater than one, then Eqn. (12) is not generally satisfied. However, care may be taken to place the zeros so that Eqn. (12) holds.

Applying the analysis of the previous section, we see from Eqn. (90) that the N_k's for this type filter will all be infinity. In this case, it is not necessary to reset the state variables very often because the growth of quantization error will be additive, and not exponential.

From a worst-case point of view, the number of samples in which error in the least significant bit may accumulate before reaching the significance floor λ_s is approximately

$$N = \lambda_s / \lambda_p, \tag{99}$$

assuming that there is one bit's worth of error in the LSB per sample processed. If the noise is zero-mean, then the accumulated error is a random walk whose standard error grows proportionally to the square root of the number of samples processed. We would like the standard error to be some distance away from the least significant bit of the output signal. For an error accumulation such that the 3-sigma error is below the significance floor, we have the relation

$$3\lambda_p \sqrt{N} < \lambda_s, \tag{100}$$

giving

$$N < (\lambda_s/3\lambda_p)^2 \tag{101}$$
$$\approx 4^G/10, \tag{102}$$

where G is the number of guard bits. Hence, we see that TIIR filters with hidden modes only on the unit circle have desirable stability properties over the filters described in the previous two sections.

Since unit-magnitude mode TIIR filters are quasi-stable and do not need to be reset every N cycles, the number of computations necessary to stabilize them may be reduced by performing the state-variable reset infrequently. Since an FIR filter has finite memory, it is sufficient to run two parallel filters starting only N steps before the state-variable transfer.

The impulse responses of TIIR filters with unit-magnitude hidden modes are sums of modal responses of the form

$$h_k[n] = P_k(n)p_k^n \tag{103}$$
$$= P_k(n)exp(j\omega_k n) \tag{104}$$

for $n = 0, \ldots, N$, where p_k is the k-th unique pole and $P_k(n)$ is an $m_k - 1$-th degree polynomial, where m_k is the multiplicity of the k-th pole; these are truncated polynomials times complex exponentials.

Some examples of TIIR filters of the unit-magnitude-modes class are given in Section 6.2.

5.4 Time-Varying TIIR Filters

The ability to change filtering characteristics in real-time may be a desirable characteristic for discrete-time filters. FIR filter coefficients in conventional tapped delay-line implementations may be adjusted dynamically without much regard to stability; such FIR systems are intrinsically stable. One simply needs to take care not to change the coefficients so quickly that transients are introduced into the filter output.

Stable IIR systems may also have their coefficients changed on-line, if the changes are slow enough. It is possible to inject energy into an IIR system by changing the filter coefficients in such a way as to make the filter output unbounded, even though the filter coefficients at each instant in time may correspond to a stable fixed-coefficient IIR system. If the coefficient changes are slow enough with respect to the filter time constant then mismatches between the state variables and coefficients are absorbed by the exponential decay of the filter.

Since the TIIR formulation depends on the careful cancellation between the impulse responses of two IIR systems, changes in filter coefficients of a TIIR system would result in a very complicated mixture of modes which are difficult to cancel out exactly. The behavior of such non-steady state operation of TIIR filters is beyond the scope of this paper. Hence, it may seem at first glance that TIIR filters would not be suited for deployment in situations where adjustable filters are necessary.

There do exist several possibilities for time-varying TIIR implementations, however. One way is to change the coefficients of the TIIR system dynamically as we would change the coefficients of a stable IIR system, taking care to match the tail-canceling IIR filter according to the impulse-generating filter using synthetic division. In this case, we must depend on the resetting algorithm given in Section 4 used to stabilize unstable TIIR systems. The use of this algorithm induces a finite memory into the state variables of the system; after $2N$ samples are processed, the system will again have the normally expected steady-state behavior. However, the period between the coefficient change and the state-variable refresh gives plenty of time for an unstable TIIR system to accumulate significant, exponentially growing errors. The result would then be quite unacceptable.

A better solution is to leave the filters invariant in time, thus avoiding the difficult analysis. We simply need to calculate the desired target TIIR filter of length N, say $H_2(z)$, with its states initialized to zero. At the appropriate time, the input stream into $H_2(z)$ is turned on and the filter begins operating. After N time steps, the filter's input delay line is full and its output has no startup transients. Given that $H_1(z)$ is the starting filter, we may then transition to $H_2(z)$ by cross-fading. Let $\alpha[n]$ be some kind of ramping function such that

$$\alpha[n] = \begin{cases} 0, & n = 0 \\ 1, & n = T, \end{cases} \tag{105}$$

and has smoothly changing values in between, and where T is the transition time. For example, $\alpha[n]$ could be a linear ramp or some kind of exponential decay. T is assumed to be sufficiently long that any transients induced by the transition are negligible. Then, the time-varying filter response may be given by the cross-faded output

$$y[n] = \begin{cases} h_1 * x[n], & n \leq n_0 \\ (1 - \alpha[n - n_0])h_1 * x[n] + \alpha[n - n_0]h_2 * x[n], & n_0 < n < n_0 + T \\ h_2 * x[n], & n \geq n_0 + T, \end{cases} \tag{106}$$

where n_0 is the time step at which the transition begins. It is assumed that $H_2(z)$ has been properly initialized, starting at least N time steps before the transition begins. At time $n_0 + T$ the filter output is determined by $H_2(z)$ so that $H_1(z)$ may be turned off and its computational and storage resources reclaimed.

Such a set-up may be used where $H_1(z)$ and $H_2(z)$ are linear-phase FFIR filters. In order to preserve phase alignment, $H_1(z)$ and $H_2(z)$ should have the same FIR length of N.

Hence, we see that the TIIR and linear-phase FFIR algorithms may be used where time-varying filters are required.

5.5 Computational Cost

Assuming that $H^+_{\mathrm{FIR}}(z)$ is a stable filter, we may implement it with $3P + 1$ multiplies and $3P$ adds per sample. $H^-_{\mathrm{FIR}}(z)$ is then unstable and must be calculated with parallel state variables as outlined in Section 4, Table 2, and the previous section. Table 2 reflects a few changes from the algorithm outlined in Table 1 which are necessary to implement the reversed filter $\tilde{H}^+_{\mathrm{FIR}}(z)$. The cost for $H^-_{\mathrm{FIR}}(z)$ is about $4P$ adds and multiplies per sample, as discussed in Section 4. We see, then, that the number of operations is about $7P$ multiplies and adds per sample.

It is interesting to note that implementing a partial fraction (parallel) structure for a transfer function as opposed to a direct implementation, as in Eqn. (30) does not significantly alter the number of multiply-accumulates used as long as the characteristic polynomials are monic and the numerators have degree one less than the denominator. However, each separate factor requires an additional input delay line of length $N_k + M_k$, which might be avoided by clever indexing. Also, the algorithmic overhead may increase, not to mention a greater amount of design effort. The advantage is that greater numerical stability is achieved, and N may then be chosen arbitrarily large. Also, each mode is present only where it is above the significance threshold λ_S. Of course, a full factorization is not necessary: it is sufficient to group together modes with the same magnitude, or to implement only the troublesome modes separately.

The unit-magnitude-mode, linear-phase filters described in Section 5.3.1 are particularly computationally efficient. Since such filters are intrinsically self-hermitian, there is no need to implement a mirror filter as in the additive and magnitude-squared design methods. Additionally, the resetting only needs to be implemented infrequently, and may be absorbed into the overhead cost. Hence, the computational cost should be approximately $3P$ operations per sample, plus the refresh overhead.

6 Simulations and Examples

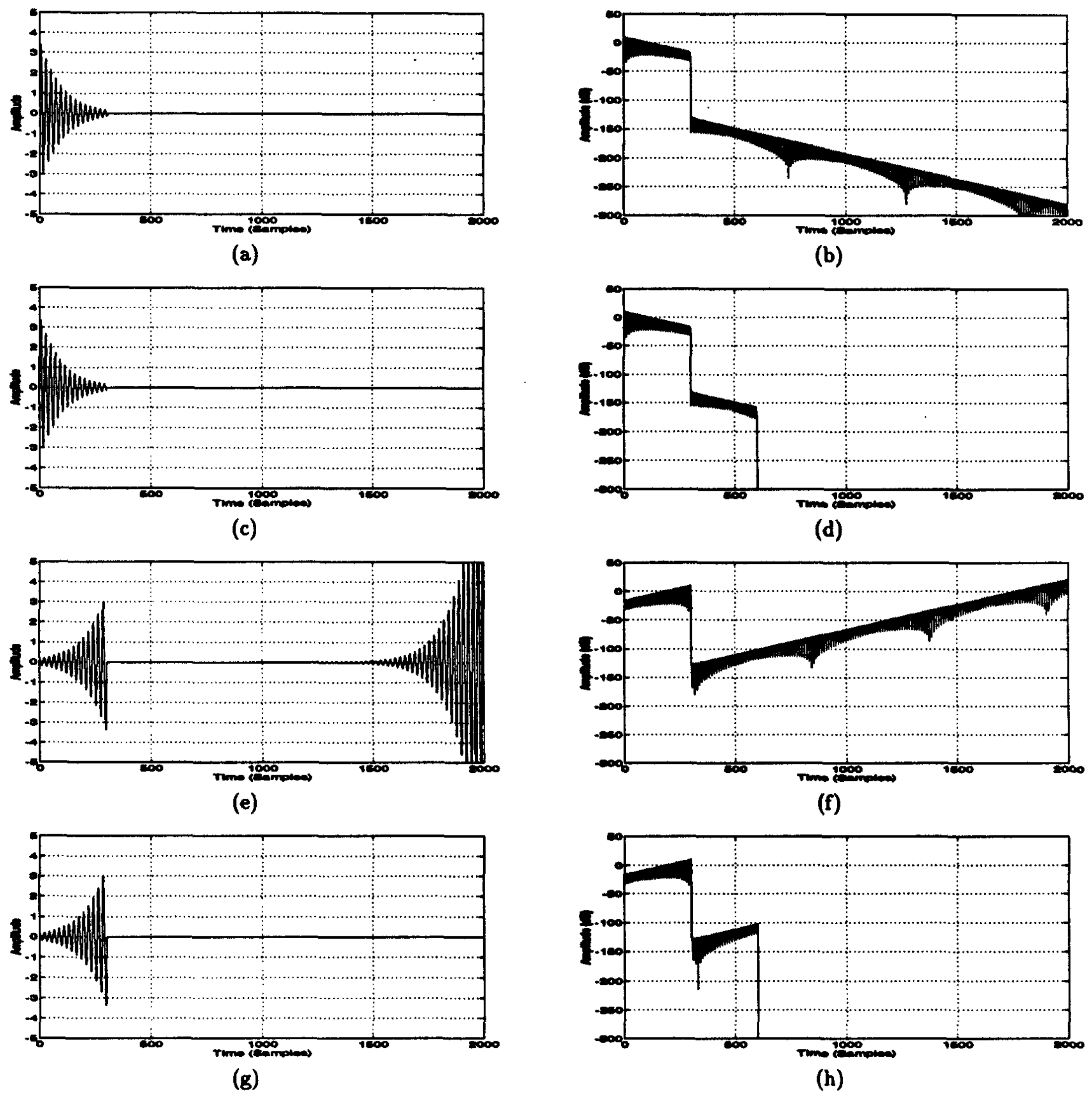

Figure 1: (a) Impulse response of TIIR system of Eqn. (107) truncated after 300 steps using stable implementation. (c) Same as (a) except using unstable implementation–state variables refreshed every 300 steps; thus the system has finite memory. (e) TIIR impulse response of reversed system of Eqn. (112) using the stable implementation. Note that the quantization error grows without bound. (g) Same as (e) except using unstable implementation. Again, state variables refreshed every 300 steps. (b,d,f, and h) Same as (a,c,e, and g), respectively, except on a decibel scale. The arithmetic was done using double-precision floating point, but with 32-bit single-precision floating-point state variables.

To illustrate the basic idea of a truncated IIR (TIIR) system, we examine the system

$$H(z) = \frac{z^2}{z^2 - 1.9z + 0.98} \tag{107}$$

$$= \frac{B^+(z)}{A^+(z)} \tag{108}$$

which we wish to truncate after $N = 300$ samples; thus we wish to have an FIR response of 301 steps. We first form the tail-canceling polynomial as in Eqn. (29). We perform synthetic division on $z^{300}B(z)$ by $A(z)$ and obtain the remainder

$$B'^+(z) = -0.162126z + 0.139770 \tag{109}$$

so that $H^+_{\mathrm{FIR}}(z)$ of Eqn. (51) is defined. We plot the impulse response of the direct implementation given in Eqn. (30) in Figures 1(a) and 1(b). We see that at step $n = 301$ the tail of the response has been subtracted off and the response magnitude drops by about 115 dB. Due to quantization errors, there is a residual response. Figures 1(c) and 1(d) show the same system, but implemented using the algorithm presented in Section 4 and Table 1. Here, the system is forced to have a finite memory and thus the residual response is completely canceled at time step $n = 600$ because the state variables are refreshed every $N = 300$ time steps.

We next form the reverse-conjugated system $H^-_{\mathrm{FIR}}(z)$ as described by Eqn. (57) and arrive at

$$H^-_{\mathrm{FIR}}(z) = \frac{-zB'^-(z) + z^{-N}B^-(z)}{A^-(z)} \tag{110}$$

$$= \frac{-0.139770z^2 + 0.162126z - z^{-N}}{0.98z^2 - 1.9z + 1} \tag{111}$$

$$= \frac{-0.142622z^2 + 0.165435z - 1.020408z^{-N}}{z^2 - 1.938776z + 1.020408}, \tag{112}$$

where, in the last equation, we have normalized by 0.98 in order to have a monic characteristic polynomial. This system has unstable hidden modes and its impulse response is plotted in Figures 1(e) and 1(f). The tail is canceled with about 125 dB attenuation, but the quantization noise grows without bound to overwhelm the signal eventually. Figures 1(g) and 1(h) show how the state variable refresh technique completely cancels the quantization noise before it becomes significant.

6.1 Elliptic-Filter-Based FFIR Design Example

Elliptic Filter Coefficients			
a_0	1.00000000	b_0	0.05149489
a_1	−5.20086294	b_1	−0.25706694
a_2	11.46455205	b_2	0.57645267
a_3	−13.68525876	b_3	−0.74102189
a_4	9.32002688	b_4	0.57645267
a_5	−3.43103178	b_5	−0.25706694
a_6	0.53331689	b_6	0.05149489

Table 3: Elliptic filter coefficients for the example in Section 6.1

We illustrate here the design of a low-pass linear-phase FFIR filter using the magnitude-squared technique outlined in Section 5.2. We wish to design a filter which meets the following criteria:

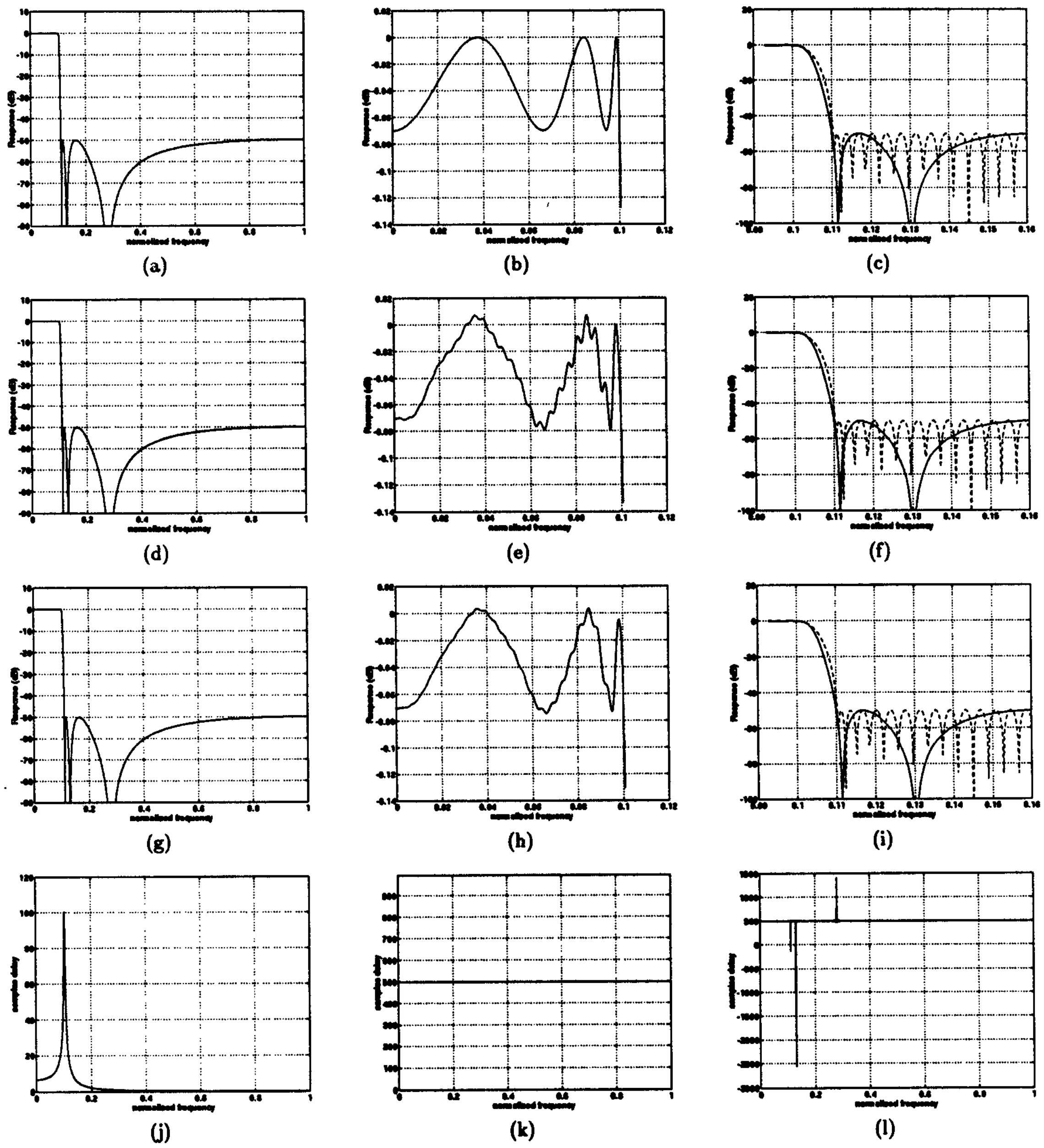

Figure 2: Filter responses from the linear-phase elliptic filter design example in Section 6.1. (a-c) Views of the untruncated response of $H^2\left(e^{j\omega}\right)$. (d-f) Views of the response of $H^2_{\mathrm{FIR}}\left(e^{j\omega}\right)$, with truncation at $N = 497$. Note the ringing in the passband in (e) due to Gibb's effect. (g-i) Views of the hybrid filter $H\left(e^{j\omega}\right) H^-_{\mathrm{FIR}}\left(e^{j\omega}\right)$. Views (c), (f), and (i) have portions of an order 500 FIR filter designed by the Parks-McLellan algorithm to have the same specs as the target filter, superposed for comparison. (j) Group delay of the forward filter $H(z)$ used as the design basis of $H^2_{\mathrm{FIR}}(z)$. (k) Group delay of the $H^2_{\mathrm{FIR}}\left(e^{j\omega}\right)$. It is exactly 497 time steps. (l) Group delay of the hybrid filter $H\left(e^{j\omega}\right) H^-_{\mathrm{FIR}}\left(e^{j\omega}\right)$.

k	Poles	Residue	N_k	$\lambda_P(dB)$
1	$0.93560805 + j0.31706231$	$0.00710587 + j0.01100914$	497	−211.6943
2	$0.93560805 - j0.31706231$	$0.00710587 - j0.01100914$	497	−211.6943
3	$0.88941945 + j0.28954430$	$-0.06609826 + j0.02150355$	116	−233.0556
4	$0.88941945 - j0.28954430$	$-0.06609826 - j0.02150356$	116	−233.0556
5	$0.77540397 + j0.15290736$	$0.06436785 - j0.21546997$	38	−247.2166
6	$0.77540397 - j0.15290736$	$0.06436785 + j0.21546997$	38	−247.2166
Direct term: $h_0 = 0.05149489$				

Table 4: Elliptic filter poles, magnitudes and decay times to significance level λ_S for the example in Section 6.1. The N_k's are calculated using Eqn. (90) with $\lambda_S = 2^{-15}$ and maximum input amplitude $\mu = 1$. The λ_P's denote the precision floor necessary to implement this filter.

	$\ell = 1$	$\ell = 2$
$a_{1,\ell}$	−1.87121609	0.97589092
$b_{1,\ell}$	0.01421174	−0.02027779
$b'_{1,\ell}$	6.08847078e−05	−5.74152962e−05
$a_{2,\ell}$	−1.77883890	0.87490286
$b_{2,\ell}$	−0.13219652	0.10512570
$b'_{2,\ell}$	−3.88583695e−06	−1.38234153e−05
$a_{3,\ell}$	−1.55080795	0.62463198
$b_{3,\ell}$	0.12873570	−0.03392829
$b'_{3,\ell}$	5.80618310e−05	−4.35418795e−05

Table 5: Coefficients for the modal decomposition of the elliptic filter in Eqn. (113) and Eqn. (114). The $b'_{k,\ell}$ are calculated using synthetic division as outlined in Table 8 and N_k from Table 4.

(1) Passband (0.00, 0.10), in fractions of $f_s/2$ with at most 0.080dB maximum peak-to-peak ripple, where f_s is the sampling frequency
(2) Stopband (0.11, 1.00) with at least 50 dB minimum attenuation
(3) Linear phase
(4) 15 bits of fixed point significance in the mantissa.
(5) Maximum input amplitude is $\mu = 1.0$

Because of the sharp transition band in the interval (0.10, 0.11), we select an elliptic filter as the basis for our design [5]. We realize that our choice of $H(z)$ only needs to meet half the specifications of (1) and (2) above, and that (3) is irrelevant. Using Matlab, we find that an order 6 filter suffices to give us a filter $H(z)$ with the desired properties such that $H^2(z)$ satisfies the above criteria. The arguments to the **ellip** function are

- $N = 6$
- Passband ripple = 0.035 dB
- Stopband attenuation = 25 dB
- Cutoff frequency = 0.10 (normalized units).

We have allowed an additional 0.005 dB margin in the passband to anticipate Gibbs ringing due to sequence truncation expressed in Eqn. (78). For the form given in Eqn. (5) the coefficients are given in Table 3. Since the multiplicity of each pole is one, we may use Eqn. (90) to calculate the decay times N_k to the significance floor. We round up in each case. The poles are given in Table 4 as well as the magnitude and N_k for each pole. Because the coefficients come in complex-conjugate pairs we may rearrange $H(z)$ into three 2nd-order real terms so that

$$\begin{aligned} H(z) = h_0 &+ \frac{b_{1,1}z^{-1} + b_{1,2}z^{-2}}{1 + a_{1,1}z^{-1} + a_{1,2}z^{-2}} \\ &+ \frac{b_{2,1}z^{-1} + b_{2,2}z^{-2}}{1 + a_{2,1}z^{-1} + a_{2,2}z^{-2}} + \frac{b_{3,1}z^{-1} + b_{3,2}z^{-2}}{1 + a_{3,1}z^{-1} + a_{3,2}z^{-2}}, \end{aligned} \tag{113}$$

where the coefficients are given in Table 5. We then implement each term separately in the truncated form given in Eqn. (30) to form $H^{+}_{\text{FIR}}(z)$ using each term's decay-to-significance times N_1, N_3, and N_5 for the cutoffs. The coefficients are calculated by using synthetic division as explained in Table 8 and are listed in Table 5. We choose $N = 497$ as the cutoff point for truncating the filter since this is the largest N_k. Thus we may implement the forward filter as

$$\begin{aligned} H^{+}_{\text{FIR}}(z) = h_0 &+ \frac{b_{1,1}z^{-1} + b_{1,2}z^{-2} - b'_{1,1}z^{-(N_1+1)} - b'_{1,2}z^{-(N_1+2)}}{1 + a_{1,1}z^{-1} + a_{1,2}z^{-2}} \\ &+ \frac{b_{2,1}z^{-1} + b_{2,2}z^{-2} - b'_{2,1}z^{-(N_3+1)} - b'_{2,2}z^{-(N_3+2)}}{1 + a_{2,1}z^{-1} + a_{2,2}z^{-2}} \\ &+ \frac{b_{3,1}z^{-1} + b_{3,2}z^{-2} - b'_{3,1}z^{-(N_5+1)} - b'_{3,2}z^{-(N_5+2)}}{1 + a_{3,1}z^{-1} + a_{3,2}z^{-2}}. \end{aligned} \tag{114}$$

Using Eqn. (57) we form

$$\begin{aligned} H^{-}_{\text{FIR}}(z) = h_0 &+ z^{-N-N_1}\frac{-b'_{1,2} - b'_{1,1}z^{-1} + b_{1,2}z^{-N_1} + b_{1,1}z^{-(N_1+1)}}{a_{1,2} + a_{1,1}z^{-1} + z^{-2}} \\ &+ z^{-N-N_3}\frac{-b'_{2,2} - b'_{2,1}z^{-1} + b_{2,2}z^{-N_3} + b_{2,1}z^{-(N_3+1)}}{a_{2,2} + a_{2,1}z^{-1} + z^{-2}} \\ &+ z^{-N-N_5}\frac{-b'_{3,2} - b'_{3,1}z^{-1} + b_{3,2}z^{-N_5} + b_{3,1}z^{-(N_5+1)}}{a_{3,2} + a_{3,1}z^{-1} + z^{-2}} \end{aligned} \tag{115}$$

from the corresponding parts in Eqn. (114). Notice that each term is delayed by an appropriate $N - N_k$ steps in order to align the phases of the response properly. To implement this filter we use the algorithm of Table 2 and Section 4 due to the unstable hidden modes of each dynamic term. We may wish to normalize the coefficients by $a_{k,2}$ so that the characteristic polynomials are monic in each term. At this point we form $H^2_{\mathrm{FIR}}(z)$ by cascading the output of $H^-_{\mathrm{FIR}}(z)$ into the input of $H^+_{\mathrm{FIR}}(z)$. This order may be preferable since the state-variable reinitialization scheme used in Section 4 may introduce some discontinuities near or below the significance floor λ_S. Thus, postfiltering by $H^+_{\mathrm{FIR}}(z)$ attenuates any such additional noise, however insignificant.

We show in Figure 2 the results of our filter implementation. Figures 2(a)-(c) show the ideal untruncated $H^2(z)$ response from our design. Its decibel response is simply double that of $H(z)$ from Eqn. (113). Figures 2(d)-(f) show the truncated response of $H^2_{\mathrm{FIR}}(z) = H^-_{\mathrm{FIR}}(z)H^+_{\mathrm{FIR}}(z)$. Notice the presence of Gibbs overshoot due to the truncation of the filter response. Figures 2(g)-(i) show the implementation $H^-_{\mathrm{FIR}}(z)H(z)$ in which the tail of the forward response is not truncated. The response is smoother, but there may be some slight deviations from phase linearity due to the asymmetry. The error due to the tail response of $H(z)$ should be small, however, since the tail of each mode for $H^+_{\mathrm{FIR}}(z)$ is cut off at the significance floor λ_S. The phase responses for $H(z)$, $H^2_{\mathrm{FIR}}(z)$, and $H^-_{\mathrm{FIR}}(z)H(z)$ are shown in Figures 2(j)-(l).

6.2 Windows and Polynomials

Unit-magnitude mode TIIR responses are convenient for generating well-known impulse response functions for computing weighted averages of an input sequence. Often in signal processing applications, an exponential window is undesirable for averaging highly non-stationary data because of the long tail. Finite-length windows with polynomial terms may be generated using multiple poles at $z = 1$ and using Eqn. (38). The terms due to the binomial coefficients generate the polynomial coefficients. The FFIR algorithm provides a cost-effective method for implementing such weighting sequences. For example, the simple N-point rectangular window may be generated as a quasi-stable TIIR filter with filter dynamics

$$H_{\mathrm{rectangular}}(z) = \frac{1 - z^{-N}}{1 - z^{-1}}. \tag{116}$$

The $2N$-point Bartlett window is

$$H_{\mathrm{Bartlett}}(z) = \frac{1}{N}\frac{1 - z^{-N} - z^{-(N+1)} + z^{-(2N+1)}}{1 - 2z^{-1} + z^{-2}}. \tag{117}$$

The N-point Hanning window, also known as the *Hann* window, is

$$H_{\mathrm{Hanning}}(z) = \frac{1}{2}\left(\frac{1 - z^{-N}}{1 - z^{-1}} - \frac{1 - \cos\left(\frac{2\pi}{N-1}\right)(z^{-1} + z^{-N}) + z^{-(N+1)}}{1 - 2\cos\left(\frac{2\pi}{N-1}\right)z^{-1} + z^{-2}}\right). \tag{118}$$

The N-point Hamming window is

$$H_{\mathrm{Hamming}}(z) = 0.54\frac{1 - z^{-N}}{1 - z^{-1}} - 0.46\frac{1 - \cos\left(\frac{2\pi}{N-1}\right)(z^{-1} + z^{-N}) + z^{-(N+1)}}{1 - 2\cos\left(\frac{2\pi}{N-1}\right)z^{-1} + z^{-2}}. \tag{119}$$

An example of a useful polynomial impulse response is the Kay window [6] used for statistically efficient frequency estimation based on averaged phase differences with weighting coefficients

$$w_{\mathrm{Kay}}[n] = \frac{6N}{N^2 - 1}\left\{\frac{n}{N} - \left(\frac{n}{N}\right)^2\right\} \tag{120}$$

for $1 \le n \le N-1$.

$$H_{\mathrm{Kay}}(z) = \frac{6}{N(N^2-1)} \times \frac{(N-1)z^{-1} - (N+1)z^{-2} + (N+1)z^{-N-1} - (N-1)z^{-N-2}}{(1-z^{-1})^3}. \tag{121}$$

This implementation requires only six adds and two multiplies, independent of N.

7 Applications

We list here some of the many possible applications for TIIR filters.

7.1 Digital speaker crossover networks and mixers for digital audio applications

In the past, commercial digital audio systems designers have had to choose between efficient, nonlinear-phase IIR filters and linear-phase, but computationally expensive, FIR filters. Mixing the outputs from non-linear phase filters may result in unwanted nulls in the combined frequency response due to cancellation of signals with different frequency-dependent group delays. The linear-phase TIIR algorithm combines the best of these features from both types of filter.

Group delay remains a potential problem, however, for real-time mixing applications with feedback to a performing musician. Delays of greater than 10ms have been known to be disorienting to performers. Given an allowance of about 6ms for miscellaneous system delays in a digital mixer, we may have about 4ms group delay in which to perform filtering. Quite reasonable filtering can be performed within 4ms at studio sampling rates.

Consider that, with a 48kHz stereo sampling rate, a 25MHz Motorola DSP56001 can devote at most 130 operations per sample to filtering, corresponding to an FIR group delay of about 1.46ms. Thus, the DSP56001 would fall behind real-time performance with one stereo channel pair before excessive group delay became a problem. On the other hand, a nice 2-pole Chebyshev type II filter may be converted into a linear-phase TIIR filter with a cost of about 14 multiply-accumulates per sample. This example illustrates the expense of FIR filtering and the potential gains from the application of linear-phase TIIR filtering.

For off-line applications, group delay is not a large concern. In either case, considerable computational savings may be attained, depending on the filter specifications. Even in the age of rapidly expanding hardware capabilities, a factor of N increase in algorithmic efficiency is valuable because it allows N times more discrete-time filters to be simulated on the same hardware with the same time requirements.

7.2 Multirate Filtering

One disadvantage of the fast FIR filtering methods is that, although they are computationally efficient, they are limited to hop sizes of one. Multirate [7] or STFT techniques, which decimate the data stream to account for a reduction in bandwidth, can be quite efficient, thus reducing the relative advantage of TIIR filtering over FIR filters. However, polyphase interpolating filters for upsampling in multirate systems may be easily implemented as TIIR filters, providing a great savings in computation, thus gaining the best features of both filter design ideologies. The development of applications of TIIR theory to multirate filter design is beyond the scope of this paper.

7.3 High-Resolution, Optimal Frequency Estimator

Kay derived the optimal weighting sequence for estimating the frequency of a pure-frequency sinusoid in Gaussian noise from finite phase differences [8]. The window given in Eqn. (120) attains the Cramér-Rao bound for moderate SNR. Eqn. (121) shows how a running average may be computed using only 2 multiplies and 6 adds per sample, independent of the window length. Such a system may be used for computing a running estimate of instantaneous frequency.

7.4 Polynomial Impulse Responses

Sometimes a finite-length window is desired for processing non-stationary signals where long exponential tails are undesirable. Running windows such as the Bartlett, Hamming, or Hanning Window give a more localized and symmetric average than a decaying exponential. Additionally, polynomial responses with sinsuoidal modulation may be constructed easily, as seen in Section 5.3.1.

8 Summary

The power of the fast FIR algorithm stems from the ability to form complicated FIR sequences as the quotient of two polynomials with relatively few non-zero coefficients. The resulting cancellation of all the dynamic modes yields the desired FIR sequence. This may be thought of as "filter compression," in which low-entropy redundancies in certain kinds of FIR sequences are encoded using generating functions based on IIR dynamics. We utilize unstable as well as stable IIR dynamics in creating our TIIR responses by exploiting a state-variable-resetting technique which takes advantage of the fact that FIR responses have finite memory. Another technique introduced here is the use of synthetic division to generate a tail-canceling complement filter. Finally, the technique described in Section 5.2 allows the straightforward use of conventional IIR filter design methods for creating linear-phase fast FIR filters.

References

[1] A. T. Fam, "FIR filters that approach IIR filters in their computational efficiency," in *Twenty-First Asilomar Conference on Signals, Systems, and Computers, Pacific Grove*, pp. 28–30, Nov. 1987.

[2] T. Saramäki and A. T. Fam, "Properties and structures of linear-phase FIR filters based on switching and resetting of IIR filters," in *Proc. IEEE International Symposium on Circuits and Systems*, (Piscataway, NJ), pp. 3271–3274, IEEE Service Center, 1990.

[3] A. Papoulis, *Signal Analysis.* New York: McGraw-Hill, 1977.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, N.J.: Prentice Hall, 1989.

[5] A. Gray and J. Markel, "A computer program for designing elliptic filters," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 24, pp. 529–538, Dec. 1976.

[6] S. Kay, "Statistically/computationally efficient frequency estimation," in *Proc. ICASSP*, (Piscataway, NJ), pp. 2292–2295, IEEE Service Center, 1988.

[7] P. P. Vaidyanathan, *Multirate Systems and filter banks.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[8] S. Kay, "A fast and accurate single frequency estimator," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, pp. 1987–1990, Dec. 1989.

Appendix: TIIR Code Listing

```
/*
   Fast FIR code
   Avery Wang
   October 15, 1993
   (c) 1993, Avery Wang
*/


#import <math.h>
#import <stdio.h>
#import <stdlib.h>
#define real float // Choose your precision: double or float


void make_tail_canceller(
     int P,     // This is the order of the IIR dynamics
     double *a, // The coefficients of A(z)=z^P + a_1 z^(P-1) + ... + a_P.
                //   Thus A(z) is monic.  There are P+1 elements in this array.
                //   In order to remain consistent, we make the assumption that
                //   a[0]=1, a[1]=a1, ... a[P]=a_P,
                //   so that the value of a[0] is actually ignored.
     double *b, // The coefficients of B'(z)=b_0 z^P + b_1 z^(P-1)+ ... + b_P.
                //   There are P+1 elements in this array.
     double *bb,// The coefficients to B"(z) -- The result of this routine are
                //   returned here.  Must be preallocated with P+1 elements.
                //   B'(z) has degree P-1 and thus has P coefficients.
                //  The reason space is allocated for P+1 coefficients
                //  is that sometimes we have to account for the reversed h_0
                //  component.  See make_reverse_filter().
     int  N     // The number of samples to propagate the impulse response
                //   in order to form B"(z).
)
/*
     This routine performs the Euclidean algorithm
     (also known as synthetic division) on z^N B'(z) by A(z).
     The remainder is the unique polynomial B'(z) of degree P-1 such that
     z^N B'(z)-B"(z) is divisible by A(z), i.e. z^N B'(z) is congruent to
     B"(z) mod A(z).  We assume that all coefficients are real.
*/
{
     int i,j;

     real *w;  // working space.  Holds running remainder.
     real factor;

     w=(real *)malloc((P+1)*sizeof(real));
/*** load the numerator ***/
     for(i=0;i<P+1;i++){
          w[i]=b[i];
     }
/*** do synthetic division ***/
     for(i=0; i<=N;i++){
          factor=w[0];
          for(j=1;j<=P;j++){
               w[j-1]=w[j]-factor*a[j];
          }
          w[P]=0;
          /**** The remainder after the i-th step is in w[0..(P-1)] ***/
     }
/*** copy the result to the output array ***/
     for(i=0;i<P;i++){
          bb[i+1]=w[i];
```

```
    }
    bb[0]=0;
    /*** Notice that bb[0]==0.  This is used to take into account
    the fact that we may wish to add an extra term from
    make_reverse_filter().
    ***/
    free(w);
}


void make_reverse_filter(
    int P,     // This is the order of the IIR dynamics
    double *a,  // The coefficients of the forward
                //   A+(z)=z^P + a_1 z^(P-1) + ... + a_P.
                //   A+(z) is monic.  There are P elements in this array.
                //   In order to remain consistent, we make the assumption that
                //   a[0]=1, a[1]=a1, ... a[P]=a_P,
                //   so that the value of a[0] is actually ignored.
    double *b,// The coefficients of the forward
                //   B+(z)=b_0 z^P + b_1 z^(P-1)+ ... + b_P.
                //   There are P+1 elements in this array.
    int  N,    // The number of samples to propagate the impulse response
                //   in order to form B-(z).
    double *ar, // This is the output array for the coefficients of
                //   A-(z)=z^P + a_(P-1)/a_P z^(P-1) + ... + a_1/a_P z + 1/a_P.
                //   Thus A-(z) is the result of reversing the
                //   coefficients of A+(z).  We normalize by a_P in order
                //   to make A-(z) monic.  There are P+1 elements in this array.
                //   We return ar[0]=1, ar[1] = a_(P-1)/a_P,...,
                //           ar[P-2] = a_1/a_P, ar[P] = 1/a_P.

    double *br, // Output array holding the coefficients of
                //   B-(z)=b_P/a_P z^P + b_(P-1)/a_P z^(P-1)+ ... + b_0/a_P.
                //   There are P+1 elements in this array.
    double *bbr // The coefficients to B-(z) -- The result of this routine are
                //   returned here.  Must be preallocated with P+1 elements.
                //   B-(z) has degree P an thus has P+1 coefficients.
)
/*
   This function takes the parameters P, a[], b[], and N, and calculates
   ar[],br[], and bbr[].  The input arguments correspond to the system
   coefficients for a TIIR filter H+(z) with the tail-canceller specified
   implicitly.  The system given by the output arguments is a TIIR
   filter H-(z) whose impulse-response is the reverse of H+(z).  Note that
   in the general case we want the impulse response H-(z) to be the
   reverse-CONJUGATE of H+(z), however since these routines only handle
   real coefficients there is no need to conjugate the system coefficents.
   We assume furthermore that the output arguments are all pre-malloc'ed.
*/
{
   int i;
   real a_P;
   double *bb;

   /*** create ar[] ***/
   a_P=a[P];
   for(i=0;i<P;i++){
       ar[P-i]=a[i]/a_P;
   }
   ar[0]=1.0;

   /*** create br[] ***/
   bb=(double *)malloc((P+1)*sizeof(double));
   make_tail_canceller(P,a,b,bb,N );
```

```
    for(i=0;i<=P;i++){
        br[P-i]=bb[i]/a_P;
    }

    /*** create bbr[] ***/
    for(i=0;i<=P;i++){
        bbr[P-i]=b[i]/a_P;
    }

    free(bb);
}


void do_stable_tiir_filter(
     double *input,      // Input signal
     int  input_length,  // Length of input in samples
     double *output,     // Preallocated output array, same length for
                         //   input and output samples.
     int P,              // Order of IIR  filter
     double *a,          // Characteristic polynomial of IIR filter.  Assumed
                         //   to be monic and of the form
                         //   A(z)=z^P + a_1 z^(P-1) +...+ a_P, so
                         //   that a[0]=1, a[1]=a_1, ... a[P]=a_P.
                         //   Hence a[] has P+1 elements.
                         //   The value of a[0] is actually ignored.
     double *b,          // Numerator of IIR transfer function B(z)=b_0 z^P.
                         //   b[0]=b_0, ... b[P]=b_P.  P+1 elements in b[].
     double *bb,         // Numerator of tail-canceller
                         //   B'(z) = b'_0 z^P + ... +b'_P, where
                         //   bb[0]=b'_0, ... , bb[P-1]=b'_P.  P+1 elements.
     int N               // Effective order of FIR response.
                         //   Note that length=N+1 !
)
/*
     We assume that the roots of A(z) have abs value <1.
*/
{
     int i,j;
     int input_dl_length;
     int input_dl_mod;

     real *input_dl;
     real *q;                 // State variables
     real new_result;

     q= (real *)malloc((P+1)*sizeof(real));

    /***
           Need to make a delay buffer with length = 2^n >N+P+1.
           On some DSP chips we may directly implement a ring buffer
           of arbitrary length.
    ***/
    for(input_dl_length=1;
           input_dl_length<N+P+1;
           input_dl_length+=input_dl_length);
     input_dl_mod=input_dl_length-1;     // Should be 2^n -1
     input_dl=(real *)malloc(input_dl_length*sizeof(real));
     /*** zero out delayline  ***/
     for(i=0;i<input_dl_length;i++)
          input_dl[i]=0;
     /****clear out state variables***/
     for(i=1;i<=P;i++) // Note that q[0] is not used.
          q[i]=0;
```

```
/******* Do filter iterations *********/
     for(i=0;i<input_length;i++){
          new_result=0;
          /** put current input sample in delay line **/
          input_dl[i&input_dl_mod]=input[i];

          for(j=1;j<=P;j++){
               new_result-=a[j]*q[j];
          }
          for(j=0;j<=P;j++){
               new_result+=b[j]*input_dl[(i-j)&input_dl_mod];
          }
          for(j=0;j<=P;j++){
                new_result-=bb[j]*input_dl[(i-j-N)&input_dl_mod];
          }
          /*** shift states ***/
          for(j=P;j>1;j--){
               q[j]=q[j-1];
          }
          q[1]=new_result;
          output[i]=new_result;
     }

     /* free any pointers here */
    free(q);
    free(input_dl);
}


void do_unstable_tiir_filter(
     double *input,      // Input signal
     int  input_length,  // Length of input in samples
     double *output,     // Preallocated output array, same length for
                         //   input and output samples.
     int P,              // Order of IIR  filter
     double *a,          // Characteristic polynomial of IIR filter.  Assumed
                         //   to be monic and of the form
                         //   A(z)=z^P + a_1 z^(P-1) +...+ a_P, so
                         //   that a[0]=1, a[1]=a_1, ... a[P]=a_P.
                         //   Hence a[] has P+1 elements.
                         //   The value of a[0] is actually ignored.

     double *b,          // Numerator of IIR transfer function B(z)=b_0 z^P.
                         //   b[0]=b_0, ... b[P]=b_P.  P+1 elements in b[].
     double *bb,         // Numerator of tail-canceller
                         //   B'(z) = b'_0 z^(P-1) + ... +b'_(P-1), where
                         //   bb[0]=b'_0, ... , bb[P-1]=b'_(P-1).  P elements.
     int N               // Effective order of FIR response.
                         //   Note that length=N+1 !
)
/*
     We assume that the roots of A(z) have abs value >1.
     We have to use the dual-buffer trick for removing quantization
     error.  Anybody know if there is a reference for this??
*/
{
     int i,j;
     int input_dl_length;
     int input_dl_mod;

     real *input_dl1, *input_dl2;  // Input delay lines
     real *q1,*q2;          // State variables: primary and auxilliary
     real new_result;
     q1=(real *)malloc((P+1)*sizeof(real));
     q2=(real *)malloc((P+1)*sizeof(real));

    /***
            Need to make a delay buffer with length = 2^n >N+P+1.
            On some DSP chips we may directly implement a ring buffer
            of arbitrary length.
    ***/
    for(input_dl_length=1;
            input_dl_length<N+P+1;
            input_dl_length+=input_dl_length);
     input_dl_mod=input_dl_length-1;     // Should be 2^n -1
     input_dl1=(real *)malloc(input_dl_length*sizeof(real));
     input_dl2=(real *)malloc(input_dl_length*sizeof(real));
     /*** zero out delaylines  ***/
     for(i=0;i<input_dl_length;i++)
          input_dl1[i]=0;
          //input_dl2[i]= 0;
     /****clear out state variables***/
     for(i=0;i<=P;i++)
          q2[i]=0;

/******* Do filter iterations *********/
    for(i=0;i<input_length;i++){
/****************** do primary TIIR filter first *****************/
        int mod_count;
        new_result=0;
        /*** shift states ***/
        if(i%(N)==0){
            mod_count=0;

            /*** copy over auxilliary state periodically ***/
            for(j=P;j>0;j--){
                    q1[j]=q2[j-1];
                }
            for(j=1;j<=P;j++){
                input_dl1[(i-j-N)&input_dl_mod]=0;
            }
            for(j=0;j<=P;j++){
                new_result-=bb[j]
                    *input_dl1[(i-j-N)&input_dl_mod];
            }
        }
        else{
        /****** otherwise, just shift old states ******/
            for(j=P;j>0;j--){
                q1[j]=q1[j-1];
            }
            for(j=0;j<=P;j++){
                new_result-=bb[j]
                    *input_dl1[(i-j-N)&input_dl_mod];
            }
        }
        /*** put current input sample in delay line ***/
        input_dl1[i&input_dl_mod]=input[i];
        /*** Do filter dynamics ***/
        for(j=1;j<=P;j++){
            new_result-=a[j]*q1[j];
        }
        for(j=0;j<=P;j++){
            new_result+=b[j]*input_dl1[(i-j)&input_dl_mod];
        }
        q1[0]=new_result;
        output[i]=new_result;
/***************** do auxilliary TIIR filter next ***************/
       /*** shift states ***/
        input_dl2[i&input_dl_mod]=input[i];
        if(i%(N)==0){
        /***** periodically zero out states *****/
            for(j=1;j<P;j++)
                /** zero input delay line **/
                input_dl2[(i-j)&input_dl_mod]=0;
            for(j=1;j<=P;j++)
                /** zero input states **/
                q2[j]=0;
            q2[0]=b[0]*input[i];
        }
        else {
        /***** otherwise just compute auxilliary filter normally ****/
            for(j=P;j>0;j--){
                q2[j]=q2[j-1];
            }
            new_result=0;
            for(j=1;j<=P;j++){
                new_result-=a[j]*q2[j];
            }
            for(j=0;j<=P;j++){
                new_result+=b[j]*input_dl2[(i-j)&input_dl_mod];
            }

            q2[0]=new_result;
        }
     }
/*** free any pointers here ***/
    free(q1);
    free(q2);
    free(input_dl1);
    free(input_dl2);
}


void do_linear_phase_fir_filter_additive(
     double *input,       // Input signal
     int  input_length,   // Length of input in samples
     double *output,      // Preallocated output array, same length for
                          //   input and output samples.
     int P,               // Order of IIR  filter
     double *a,           // Characteristic polynomial of IIR filter.  Assumed
                          //   to be monic and of the form
                          //   A(z)=z^P + a_1 z^(P-1) +...+ a_P, so
                          //   that a[0]=1, a[1]=a_1, ... a[P]=a_P.
                          //   Hence a[] has P+1 elements.
                          //   The value of a[0] is actually ignored.
     double *b,           // Numerator of IIR transfer function B(z)=b_0 z^P.
                          //   b[0]=b_0, ... b[P]=b_P.  P+1 elements in b[].
     int N,               // Effective length of FIR response segment.
     int M,               // Offset
     int sign             // +1 or -1.  These are the real-valued choices
                          //   for phase.  If this were a complex-valued
                          //   routine, this would be exp(i phi).
)
/*
    Given the input parameters a[], b[], M,N, and P, this function
    creates the linear-phase filter as outlined in the paper by Wang
    and Smith.  The output samples are computed and returned in output[].
    It is assumed that the stable response is encoded in a[].  The
    routine generates the forward and backwards FIR responses.
```

```
*/
{
    int i;
    double *aux_output;
    double *bbx;
    double *ar,*br;

    aux_output=(double *)malloc((input_length-M)*sizeof(double));
    ar =(double *)malloc((P+1)*sizeof(double));
    br =(double *)malloc((P+1)*sizeof(double));
    bbx=(double *)malloc((P+1)*sizeof(double));

    make_tail_canceller(P,a,b,bbx,N);
    do_stable_tiir_filter(input,input_length-M,
                             aux_output,P,a,b,bbx,N);

    make_reverse_filter(P,a,b,N,ar,br,bbx);
    do_unstable_tiir_filter(input,input_length,output,P,ar,br,bbx,N);
    for(i=0;i<input_length-M;i++){
        output[i+M] += sign*aux_output[i];
    }

    free(aux_output);
    free(ar);
    free(br);
    free(bbx);
}


void do_linear_phase_fir_filter_cascade(
     double *input,      // Input signal
     int  input_length,  // Length of input in samples
     double *output,     // Preallocated output array, same length for
                         //   input and output samples.
     int P,              // Order of IIR  filter
     double *a,          // Characteristic polynomial of IIR filter.  Assumed
                         //   to be monic and of the form
                         //   A(z)=z^P + a_1 z^(P-1) +...+ a_P, so
                         //   that a[0]=1, a[1]=a_1, ... a[P]=a_P.
                         //   Hence a[] has P+1 elements.
                         //   The value of a[0] is actually ignored.
     double *b,          // Numerator of IIR transfer function B(z)=b_0 z^P.
                         //   b[0]=b_0, ... b[P]=b_P.  P+1 elements in b[].
     int N               // Effective length of FIR response segment.
)
/*
    Given the input parameters a[], b[], M,N, and P, this function
    creates the linear-phase filter as outlined in the paper by Wang
    and Smith.  The output samples are computed and returned in output[].
    It is assumed that the stable response is encoded in a[].  The
    routine generates the forward and backwards FIR responses.
*/
{
    double *aux_output;
    double *bbx;
    double *ar,*br;

    aux_output=(double *)malloc((input_length)*sizeof(double));
    ar =(double *)malloc((P+1)*sizeof(double));
    br =(double *)malloc((P+1)*sizeof(double));
    bbx=(double *)malloc((P+1)*sizeof(double));

    make_tail_canceller(P,a,b,bbx,N);
    do_stable_tiir_filter(input,input_length,
                               aux_output,P,a,b,bbx,N);
    make_reverse_filter(P,a,b,N,ar,br,bbx);
    do_unstable_tiir_filter(aux_output,input_length, output,P,ar,br,bbx,N);

    free(aux_output);
    free(ar);
    free(br);
    free(bbx);
}
```