A New Reverse IIR Filtering Algorithm

Martin Vicanek

25. October 2015, revised 5. January 2022

1 Introduction

Recursive digital filters have an infinite impulse response (IIR). Their major advantage over non-recursive filters is that they typically requrire fewer operations to meet given filter criteria. A disadvantage is that recursive filters per se have different group delay for different frequency components, which manifests itself as a nonlinear phase-frequency relation. In situations where processing may be carried out offline, Kormylo and Jain [1] pointed out that a two-pass forward backward filtering yields linear phase. Image processing is one such area of possible application. Later, Czarnach [2] devised a scheme to deal with continuous input by dividing the data stream into blocks and processing each block individually. Powell and Chau [3] improved that method by employing an overlap add sectioned convolution. All block processing methods have longer than necessary latency, however. Wang and Smith [4] introduced a conceptually different scheme based on IIR tail cancellation. Special measures need to be taken to deal with hidden unstable modes, and there is a fundamental limit to the filter length.

Recently, a member of the Flowstone forum under the alias name cbbuntz presented a surprisingly simple algorithm to implement time-revesed recursive filters with real poles. In this note we elaborate on cbbuntz' idea, extend it to complex conjugate poles and provide two specific examples. To the author's knowledge, the method has not been published before.

2 Algorithm

2.1 Real pole

Consider a simple filter with a single pole at $z = c$, with $|c| < 1$. The transfer function is given by

$$\begin{aligned} H(z) = \frac{1}{1 - cz^{-1}} &= 1 + cz^{-1} + c^2 z^{-2} + c^3 z^{-3} + \dots \\ &= (1 + cz^{-1})(1 + c^2 z^{-2})(1 + c^4 z^{-4})(1 + c^8 z^{-8}) \dots \end{aligned} \quad (1)$$

The last equality in eq. (1) says that the single-pole filter may be implemented as a cascade of delays (z^{-n}), attenuators (c^n) and bypasses, with delay times doubling at each stage, $n = 2^N$, and the amplidude decaying accordingly. Figure 1 shows the corresponding block diagram.

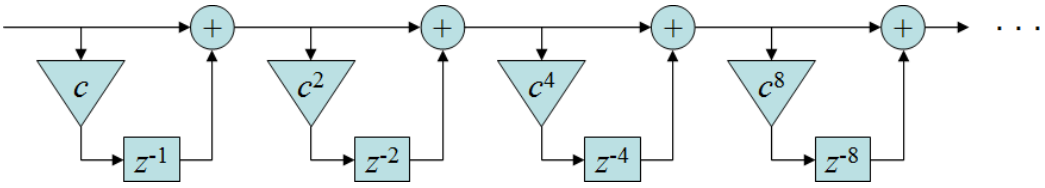

Figure 1: Block diagram of forward processing according to equation (1).

The single pole filter has infinite impulse response (IIR), hence infinitely many delay terms would have to be accounted for to render the equality exact. In practice, however, it is possible to truncate the IIR at some threshold level, thereby turning the IIR into a FIR. Since the decay is exponential, the corresponding error can be made arbitrarily small. If D is the signal-to-noise ratio in dB, then the number of stages that need to be taken into account is

$$N = \log_2 \left(-\frac{D}{20 \log_{10} |c|} \right). \quad (2)$$

Time reversal of a FIR filter is straight forward: simply reverse the order of the impulse response and add a total delay to make the time-reversed filter causal. For the (truncated) structure in figure 1 this leads to the block diagram in figure 2. Equivalently, we may write

$$H_r(z) = (c + z^{-1})(c^2 + z^{-2})(c^4 + z^{-4})(c^8 + z^{-8}) \dots (c^{2^N} + z^{-2^N}) \quad (3)$$

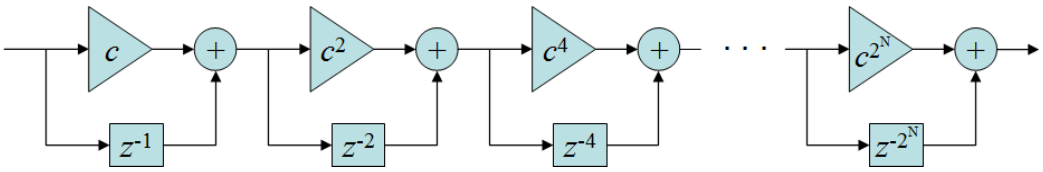

Figure 2: Block diagram of backward processing according to equation (3).

Equation (3) provides a stable algorithm for the time-reversed single-pole filter using an array of delays and attenuators.

A pseudo-code for one stage of the time-reversed filter is given below, where x_n and u_n denote streams of input and output samples, respectively.

```
u[n] = c^k * x[n] + x[n-k]
```

This may be viewed as a sparse FIR filter with only two taps. The entire filter, then, is a cascade of $N+1$ stages with $k = 1, 2, 4, 8, \ldots, 2^N$. It is advisable, of course, to pre-compute the coefficients c^k.

We note that other truncation limits than powers of 2 samples are possible, e.g. by choosing

$$H(z) = (1 + cz^{-1} + c^2z^{-2})(1 + c^3z^{-3})(1 + c^6z^{-6})(1 + c^{12}z^{-12})\ldots \quad (4)$$

and its time reversed counterpart

$$H_r(z) = (c^2 + cz^{-1} + z^{-2})(c^3 + z^{-3})(c^6 + z^{-6})\ldots(c^{3\cdot 2^N} + z^{-3\cdot 2^N}). \quad (5)$$

This may facilitate tighter latency bounds if the filter decay length falls between 2^N and 2^{N+1}.

2.2 Complex conjugate poles

Consider a filter with a pair of complex conjugate poles c and $\overline{c}$. Its transfer function is

$$H_{\text{c.c.}}(z) = \frac{1}{(1 - cz^{-1})(1 - \overline{c}z^{-1})} \quad (6)$$

Such a filter may be viewed as a cascade of two single, complex poles. Thus the obvious way to time-reverse it is to apply equation (3) twice in succession. There is, however, a better way, resulting in half as much latency and CPU cost.

The results in the previous section remain valid for a complex valued pole $c = a + ib$, as long as its modulus is smaller than 1. A filter with a single complex pole has a transfer function

$$H_c(z) = \frac{1}{1 - cz^{-1}}. \tag{7}$$

where we have used the subscript $_c$ to indicate that the pole is complex. If fed with a real input signal x_n, the filter will produce a complex output signal $u_n + iv_n$:

$$H_c x = u + iv. \tag{8}$$

On the other hand, the filter $H_{\text{c.c.}}$ in equation (6) will output a real signal. Comparing equations (6) and (7) reveals that

$$H_c(z) = (1 - az^{-1} + ibz^{-1})H_{\text{c.c.}}(z), \tag{9}$$

hence

$$u = (1 - az^{-1})H_{\text{c.c.}}x, \quad v = bz^{-1}H_{\text{c.c.}}x, \tag{10}$$

which implies

$$H_{\text{c.c.}}x = u + \tfrac{a}{b}v \tag{11}$$

In other words, we may realize a filter with c.c. poles by filtering with a single complex pole $c = a + ib$ and taking a linear combination $u_n + \frac{a}{b}v_n$ of the real and imaginary outputs, respectively.

This result is also valid for the time-reversed counterparts of H_c and $H_{\text{c.c.}}$, which may be realized with the structure in figures 3 and 4.

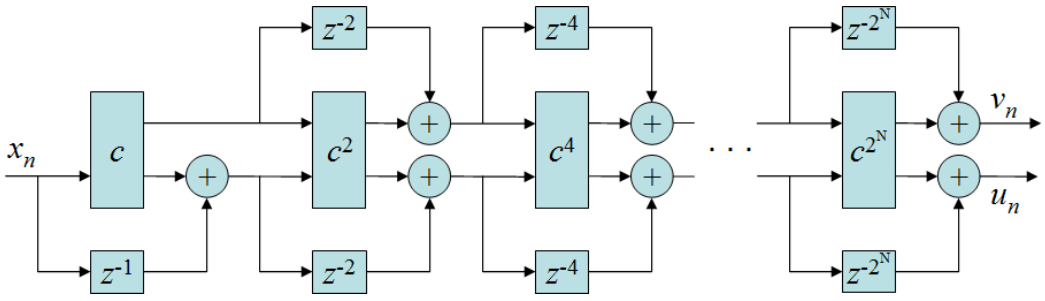

Figure 3: Block diagram of a time-reversed complex pole filter according to equation (3). Boxes labeled c, c^2, etc. represent complex multiplications.

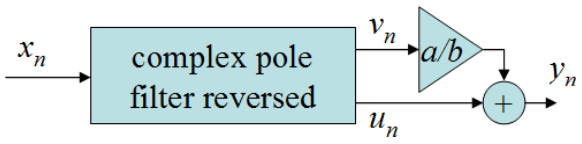

Figure 4: Block diagram of a time-reversed complex conjugate pole filter. The box labeled "complex pole filter reversed" represents the structure in figure 3.

A pseudo-code for the time-reversed filter with complex conjugate poles is given below. First pre-compute powers $(a + ib)^k$,

```
a2 = a * a - b * b
b2 = 2 * a* b
a4 = a2 * a2 - b2* b2
b4 = 2 * a2 * b2
a8 = a4 * a4 - b4* b4
b8 = 2 * a4 * b4
...
```

The first filter stage takes a real input stream x_n and outputs a complex stream u_n, v_n,

```
u[n] = a * x[n] + x[n-1]
v[n] = b * x[n]
```

Subsequent stages for $k = 2, 4, 8, \ldots, 2^N$ take a complex input stream x_n, y_n,

```
u[n] = ak * x[n] - bk * y[n] + x[n-k]
v[n] = bk * x[n] + ak * y[n] + y[n-k]
```

The filter output is obtained at the last stage (pre-compute a/b, of course):

```
out[n] = x[n] + (a/b) * y[n]
```

For higher order IIR filters there are various options to break them down into single-pole and c.c.-pole units. One obvious way is straight factorization, another is partial fraction expansion. The latter results in parralel filter configurations with lower overall latency. In that case, however, the delays of all parallel units must be equal.

3 Examples

3.1 Linkwitz-Riley Crossover

A Linkwitz-Riley (LR) crossover is a pair of complementary highpass and lowpass filters. Most common are 4th order designs, where each filter arm consists of two cascaded Butterworth biquad filters as depicted in figure 5 (a). Since the poles are identical for the lowpass and highpass filters, we may separate the biquads into products of poles and zeros, and place the poles in the common signal path, cf. figure 5 (b). To render the filter linear phase, one c.c. pole pair may be time-reversed as shown in the previous section. The zeros are symmetrical hence need not be time-reversed.

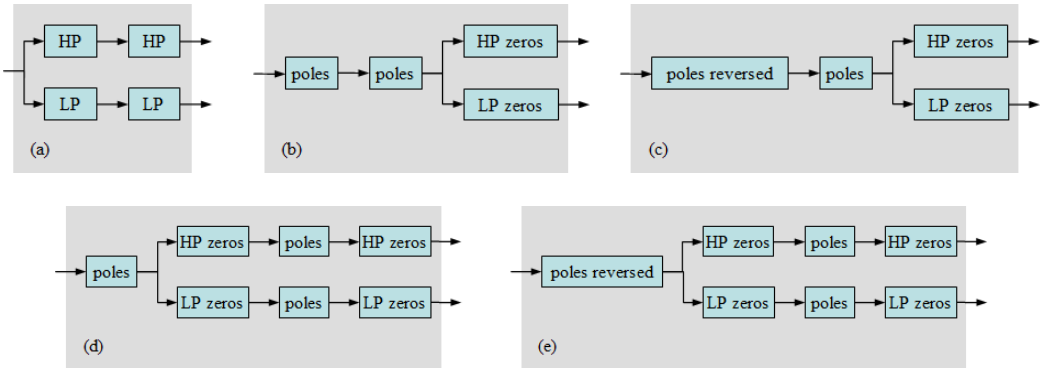

Figure 5: Block diagram of a 4th order Linkwitz-Riley crossover filter. (a): Generic structure. (b): Shared common poles. (c): Linear phase implementation. (d): Topology with improved noise characteristics. (e) Improved linear phase implementation.

It turns out that this realization has issues with the highpass branch at low crossover frequencies. The placement of all four HP zeros at the end amounts to taking a fourth(!) derivative, which greatly amplifies quantization noise from the previous processing units. Therefore it is better to place two of the zeros *before* the poles as in figure 5 (d) and (e). The cost is only two additional multiplies.[1]

For a fourth order LR with 1 kHz corner frequency and 44.1 kHz sample rate

[1] As a reader pointed out, since the design is linear phase, the high-pass filter may also be obtained by subtracting the low-passed signal from the (delayed) input signal.

it is sufficient to use 6 stages for time reversal, resulting in a latency of 64 samples. The filter operates over a dynamic range > 100 dB without any apparent truncation artifacts, see figure 6, with a total of 6 complex and 6 real multiplies per sample.

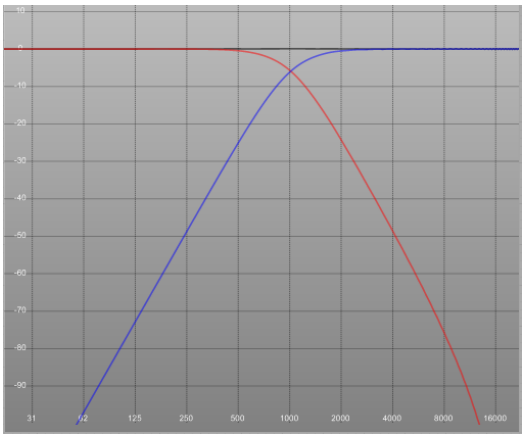

Figure 6: Transfer function of a 4th order Linkwitz-Riley crossover filter. Red: lowpass, Blue: highpass, Black: straight sum.

3.2 Hilbert filter

We start with an allpass filter network as in figure 7 (a), with outputs in quadrature over a wide portion of the Nyquist band. Each filter arm includes a cascade of eight second-order allpass units as in figure 7 (b). The coefficients are given in table 1. The idea is to time reverse one of the allpass arms and place it in cascade with the other arm as in figure 7 (c). To keep the latency as low as possible, we will first transform the allpass cascade 1-8 into a parallel structure by means of partial fractions.

The transfer function of the allpass cascade 1-8 is given by a product of eight second-order allpass transfer functions. We will transform the allpass cascade into a parallel structure by means of partial fractions with respect to z^2,

$$H(z) = \prod_{i=1}^{8} \frac{z^{-2} - a_i}{1 - a_i z^{-2}} = C + \sum_{i=1}^{8} \frac{A_i z^{-2}}{1 - a_i z^{-2}} \tag{12}$$

i	a_i	i	a_i
1	0.0406273391966415	9	0.1500685240941415
2	0.2984386654059753	10	0.4538477444783975
3	0.5938455547890998	11	0.7081016258869689
4	0.7953345677003365	12	0.8589957406397113
5	0.9040699927853059	13	0.9353623391637175
6	0.9568366727621767	14	0.9715130669899118
7	0.9815966237057977	15	0.9886689766148302
8	0.9938718801312583	16	0.9980623781456869

Table 1: Coefficients for a Hilbert allpass network.

with the constants given by

$$C = \prod_{i=1}^{8} a_i, \quad A_i = \prod_{j=1}^{8} (1 - a_i a_j) \Bigg/ \prod_{\substack{j=1 \\ j \neq i}}^{8} (a_j - a_i) \tag{13}$$

The numerical vaues are listed in table 2. The resulting topology is shown

A_1	-10.40114191	A_4	8.998241341	A_7	-0.345760229
A_2	20.43520081	A_5	-3.976562059	A_8	0.040023521
A_3	-16.29902549	A_6	1.406370891	C	0.004832826

Table 2: Coefficients for partial fractions in equation (12).

in figure 8 (a). Time reversal of each partial fraction in equation (12) is straight-forward since the corresponding poles are real. Hence we may use equation (3), however with z^2 substituted for z.

The filter has a latency of 4096 samples and a sideband suppression of 90 dB over the entire audio band (30 Hz to 22 kHz at 44.1 kHz sample rate). Figure 9 sows the impulse response.

4 Conclusion

We presented a new method for time reversal of filters with rational transfer functions. Unlike in block processing, latency can be made as small as the filter's decay time. Time reversed filter structures are stable and numerically

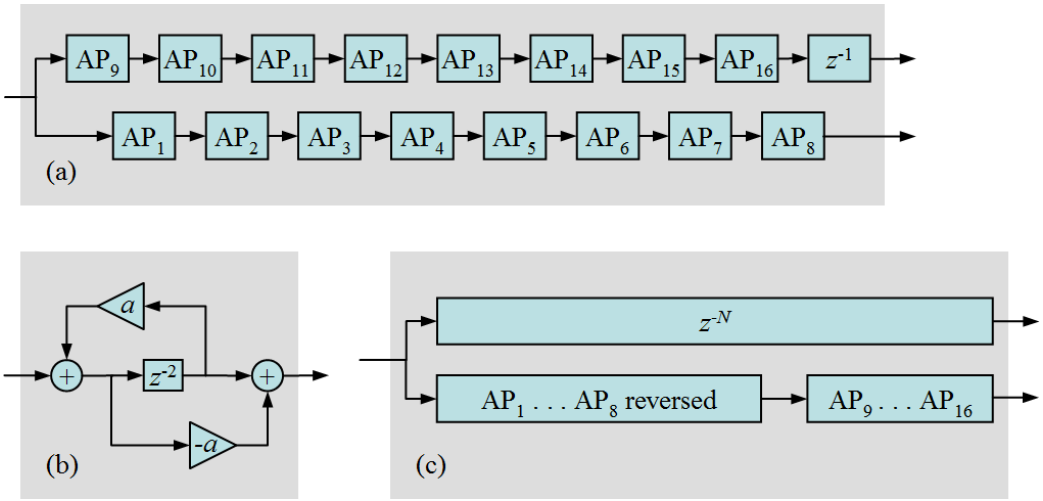

Figure 7: Hilbert filter. (a): Allpass filter network. (b): Expanded allpass unit. (c): True Hilbert transformer.

well behaved. Computational effort is a few times that of the original filter, however it is much less compared to the correspomdmg FIR filter.

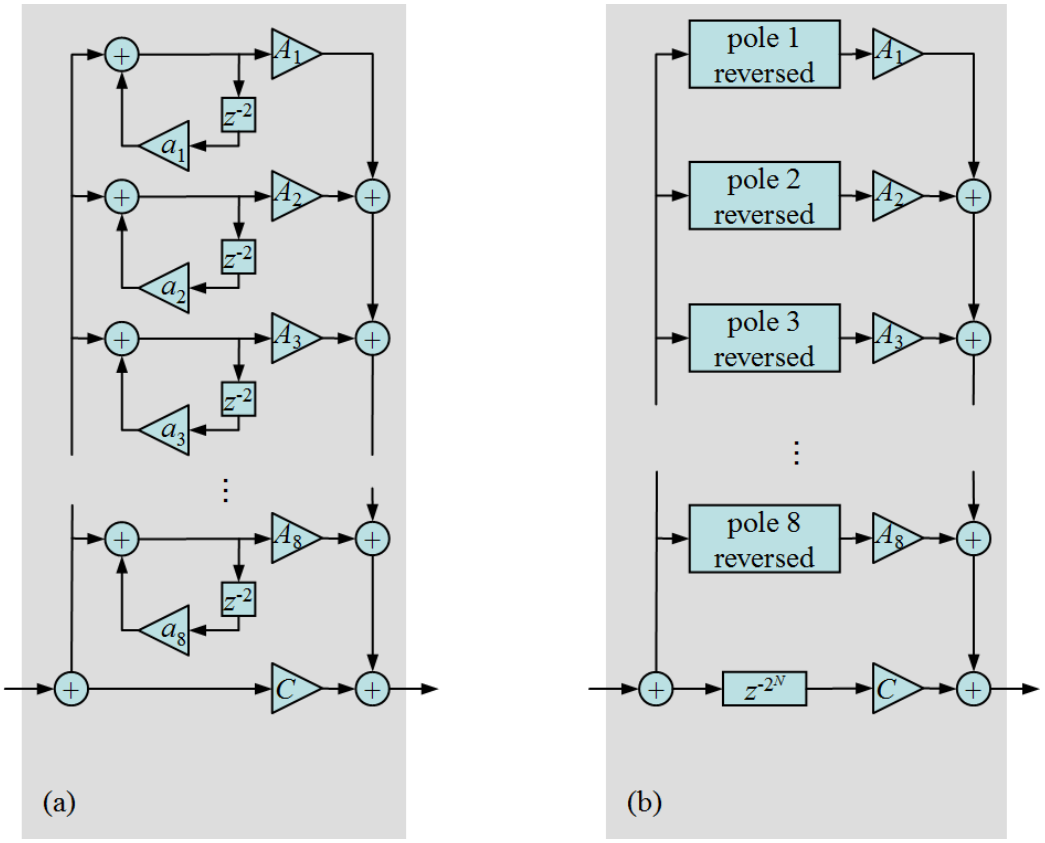

Figure 8: Time reversal of allpass filter cascade 1-8 in figure 7. (a): Parallel structure. (b) Time reversal.

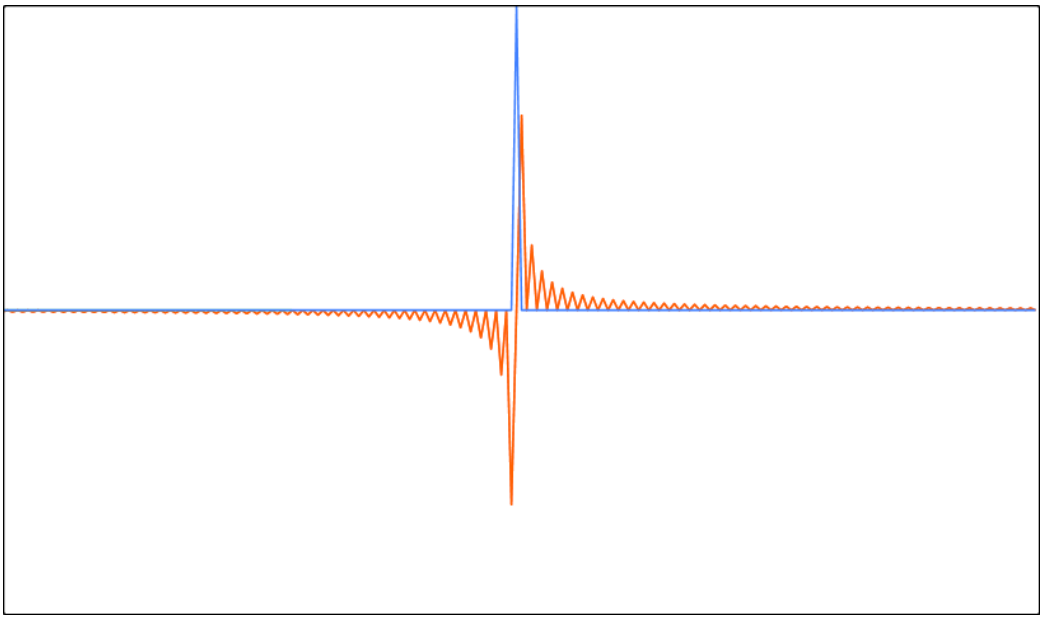

Figure 9: Section of 200 samples of the impulse response of the Hilbert filter
Orange: Filter output. Blue: Delayed input.

References

[1] J. J. Kormylo and V. K. Jain, Two-pass recursive digital filter with zero phase shift, IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-22, pp.384 -387 1974.

[2] R. Czarnach, Recursive processing by noncausal digital filters. IEEE Transactions on Acoustics, Speech and Signal Processing, (Volume:30, Issue: 3). 1982.

[3] S. R. Powell and P. M. Chau, A Technique for Realizing Linear Phase IIR Filters. IEEE Transactions on Signal Processing, vol. 39. no. 11, 1991.

[4] A. Wang, J.O. Smith III, On fast FIR filters implemented as tail-canceling IIR filters. IEEE Transactions on Signal Processing, vol. 45, no. 6, p. 1415, 1997.

[5] cbbuntz, post on the DSP Robotics User Forum, 2014.