A Class of Fast FIR Filters

Martin Vicanek

January 8, 2025

1 Introduction

In digital signal processing, finite impulse response (FIR) filters have some advantages over their infinite impulse response (IIR) counterparts:

1. they can be designed with steep transition bands
2. they can be designed to have constant group delay (linear phase)
3. they recover from invalid input (IIRs will output NaNs forever)
4. they don't suffer from denormals

In some applications, the finite impulse response is a desirable feature in itself.

The main drawbacks are (i) a lack of closed design formulas for standard filter types and (ii) high CPU load. The latter may be addressed using FFT, however in its simple form it introduces additional delay of one block size.

In this article I examine a class of FIR filters with simple design rules and a low CPU load. These filters may be of interest in audio applications like envelope followers, compressors, band splitters, onset detectors etc.

2 Some Filter Basics

Consider a stream of (real valued) samples x_n with $n = 0, 1, 2, \ldots$ At each instant in time $t_n = nT$, where T is the sampling interval, we may perform a weighted sum over the last N samples to obtain a transformed (or filtered) stream y_n,

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k}. \tag{1}$$

If the incoming stream was an impulse, i.e. all samples equal to zero except $x_0 = 1$, the resulting y_n would be equal to the weights h_n for $n < N$ and zero otherwise. Hence the weights h_k in eq.(1) constitute the impulse response (IR). Since there are only N weights, the filter defined by eq.(1) has a finite impulse response (FIR).

A more general class of filters includes, in addition to the last N *incoming* samples x_{n-k}, another term with previous *transformed* samples y_{n-k}. In most (but not all!) cases such iterative filters have an infinite impulse response (IIR). It is common, though somewhat inaccurate, to call these filters IIR-filters.

Filters are characterized by their response function in the frequency domain. For digital filters, this is given by the z-transform of the impulse response,

$$H(z) = \sum h_n z^{-n}, \tag{2}$$

where $z = e^{i\omega T}$ encodes the (angular) frequency ω. In the following, we will use T as the unit of time, so $-\pi < \omega \leq \pi$ spans the Nyquist interval.

The expression on the right hand side of eq.(1) is a convolution. Straight evaluation results in $O(N)$ operations per sample. It is possible to use the fast Fourier transform (FFT) to perform that task in blocks with a reduced complexity $O(\log(N))$, however, at the price of an extra delay corresponding to the block size. On can go a step further and use an arrangement of blocks with different sizes to minimize or even remove that latency [1]. Indeed, this is the preferred if not the only viable option to implement convolution reverbs.

This article treats a much simpler case, where the IR exhibits a special form of self-similarity, thereby allowing an efficient evaluation of the sum in eq.(1).

3 Running Sum

The simplest case of a FIR filter is the so-called running sum, where all h_k are equal to 1:

$$y_n = \sum_{k=0}^{N-1} x_{n-k} \tag{3}$$

The summation may be evaluated using the previous result for y_{n-1}:

$$y_n = y_{n-1} + x_n - x_{n-N}. \tag{4}$$

The right hand side of eq.(4) has only three terms independent of the IR size N, hence may be computed much more efficiently than the original eq.(3). Even though only x_n and x_{n-N} enter in eq.(4), all N intermediate samples x_{n-k} have to be stored along the way because each will be needed in a subsequent evaluation.

There is a small caveat in floating point implementations of eq.(4) that rounding errors accumulate over time, resulting in the expression above drifting away from the actual running sum. This may be mitigated by periodically resetting the value. A simple way to do that is compute, in parallel, a reference

$$r_n = r_{n-1} + x_n \tag{5}$$

and reset

$$y_n = r_n, \quad r_n = 0 \tag{6}$$

every N samples. There is no need to store all r_n along the way, as we only require the final result to update y_n after having accumulated N samples.

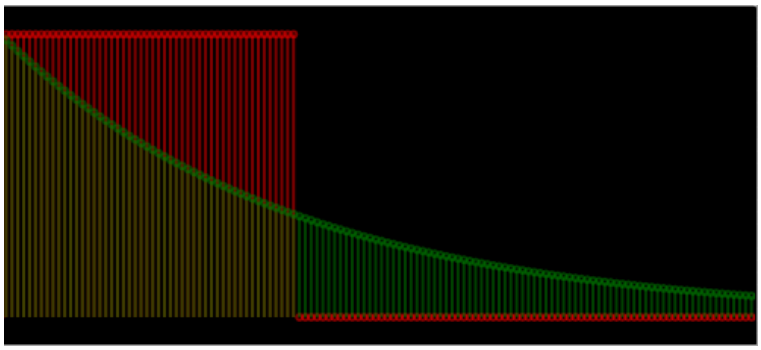

Figure 1: Impulse responses of a moving average filter (red) with $N = 50$, and a leaky integrator (green) $y_n = 0.98\, y_{n-1} + 0.02\, x_n$ for comparison.

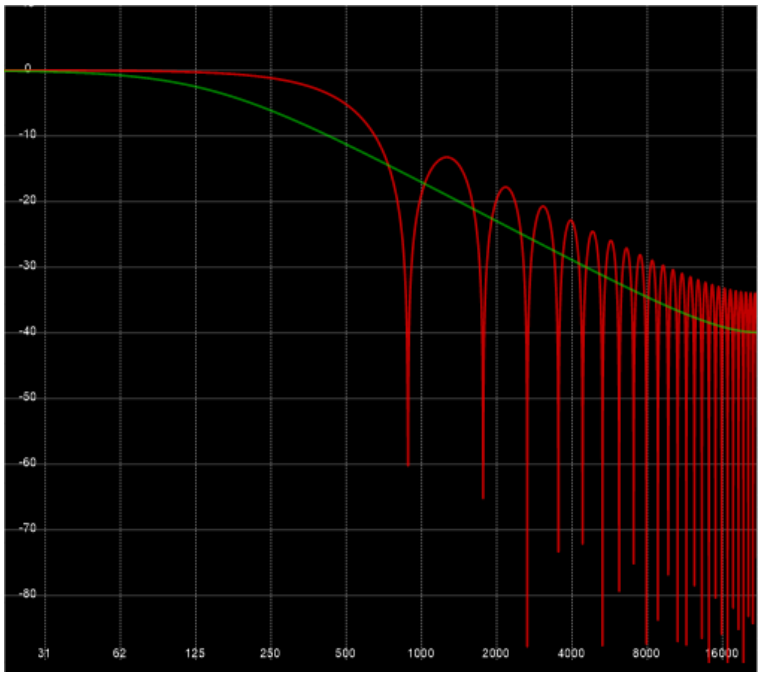

Figure 2: Magnitude responses of a moving average filter (red) and a leaky integrator (green).

The running *average*, also known as moving or sliding average, may be trivially obtained from the running *sum* by applying a factor $1/N$ to each sample of either the input or output stream. The running average may be viewed as a smoothing or lowpass filter with magnitude response

$$|H(z)| = \frac{1}{N} \frac{|\sin(\frac{N}{2}\omega)|}{\sin(\frac{1}{2}\omega)}. \tag{7}$$

At DC ($z = 1$ or $\omega = 0$), the response is equal to 1 (zeros of numerator and denominator cancel). There are equally spaced zeros (i.e. notches) at frequencies $\omega = \frac{2\pi}{N}, \frac{4\pi}{N}, \ldots$ The

magnitude in between the zeros falls off at 6 dB per octave except near Nyquist where it flattens out. Figure 1 shows the impulse response for $N = 50$, compared to that of a leaky integrator $y_n = 0.98\, y_{n-1} + 0.02\, x_n$. Figure 2 shows the corresponding magnitude responses.

The running average filter has a constant group delay (aka latency) of $\frac{N-1}{2}$ samples.

4 Sliding Goertzel

The above concept of using previous filter evaluations may be generalized to include expressions of the form

$$y_n = \sum_{k=0}^{N-1} q^k x_{n-k} \tag{8}$$

where q is some (potentially complex valued) parameter. The sum in eq.(8) may be evaluated from the previous sum y_{n-1},

$$y_n = q y_{n-1} + x_n - q^N x_{n-N}. \tag{9}$$

Eq.(9) is a generalization of eq.(4). For $|q| < 1$ this recurrence scheme is stable and may be used directly, otherwise compute in parallel

$$r_n = q r_{n-1} + x_n \tag{10}$$

and reset y_n and r_n every N samples as in eq.(6).

The particular choice $q = e^{i\omega_0}$ leads to a sliding variant of the Goertzel tone detector [2]. If we split up the response into real and imaginary parts, $y_n = u_n + i v_n$, then eq.(8) becomes

$$u_n = \sum_{k=0}^{N-1} \cos(k\omega_0)\, x_{n-k} \qquad v_n = \sum_{k=0}^{N-1} \sin(k\omega_0)\, x_{n-k} \tag{11}$$

and from eq.(9) we obtain a set of recursive equations

$$\begin{aligned} u_n &= \cos(\omega_0)\, u_{n-1} - \sin(\omega_0)\, v_{n-1} + x_n - \cos(N\omega_0)\, x_{n-N} \\ v_n &= \sin(\omega_0)\, u_{n-1} + \cos(\omega_0)\, v_{n-1} \qquad\qquad - \sin(N\omega_0)\, x_{n-N}. \end{aligned} \tag{12}$$

Usually the IR will encompass some integer or half integer number k of cycles, $N\omega_0 = 2\pi k$. The corresponding filter will have a bandpass characteristic with the passband centered around ω_0. The relative bandwidth is inversely proportional to the number of cycles k. Figure 3 and 4 show the impulse and (normalized) magnitude responses, respectively, of the real and imaginary parts u_n and v_n of a sliding Goertzel filter with IR length $N = 100$ and $k = 6$ full cycles.

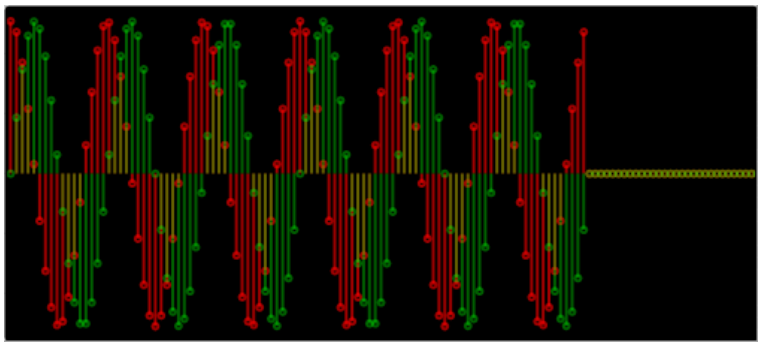

Figure 3: Real (red) and imaginary (green) impulse responses u_n and v_n of a sliding Goertzel filter, refer to eq.(11). IR length is $N = 100$, and ω_0 was chosen to fit $k = 6$ full cycles.

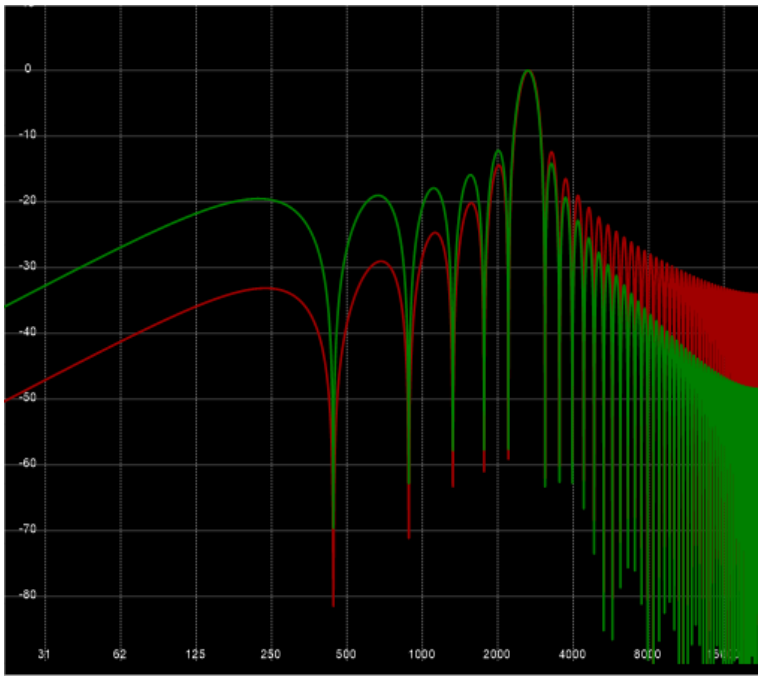

Figure 4: Magnitude responses of real (red) and imaginary (green) sliding Goertzel filter ouputs u_n and v_n, respectively. Same parameters as in figure 3.

For $k = \frac{1}{2}$, i.e. $\omega_0 = \pi/N$, the imaginary IR v_n is a half-sine, refer to the second eq.(11) and the orange curve in figure 5. If we use v_n in eq.(12) as a filter output, we get a lowpass filter with magnitude response

$$|H(z)| = \frac{1}{2}\left|\frac{\sin[\frac{N}{2}(\omega-\omega_0)]}{\sin[\frac{1}{2}(\omega-\omega_0)]} + \frac{\sin[\frac{N}{2}(\omega+\omega_0)]}{\sin[\frac{1}{2}(\omega+\omega_0)]}\right|. \tag{13}$$

The expression in eq.(13) has zeros at $\omega = \frac{3\pi}{N}, \frac{5\pi}{N}, \frac{7\pi}{N}, \ldots$, and the falloff is 12 dB per octave. This is illustrated by the orange curve in figure 6. Latency is $\frac{N-1}{2}$ samples.

5 Composite Filter Structures

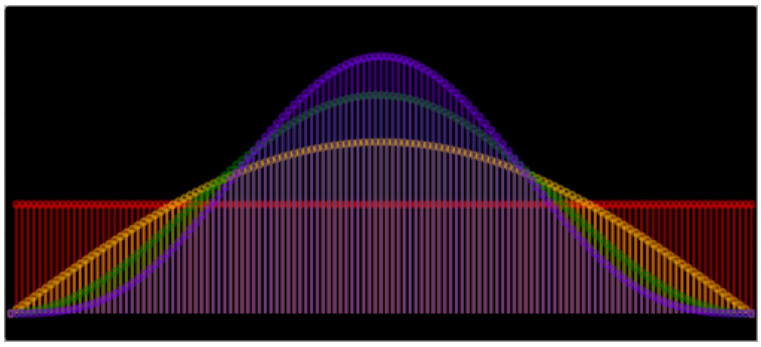

Figure 5: Impulse responses of various smoothing filters: box (red), half-sine (orange), Hann (green), and $\sin^3$ (magenta). IR length is $N = 128$ in all cases.

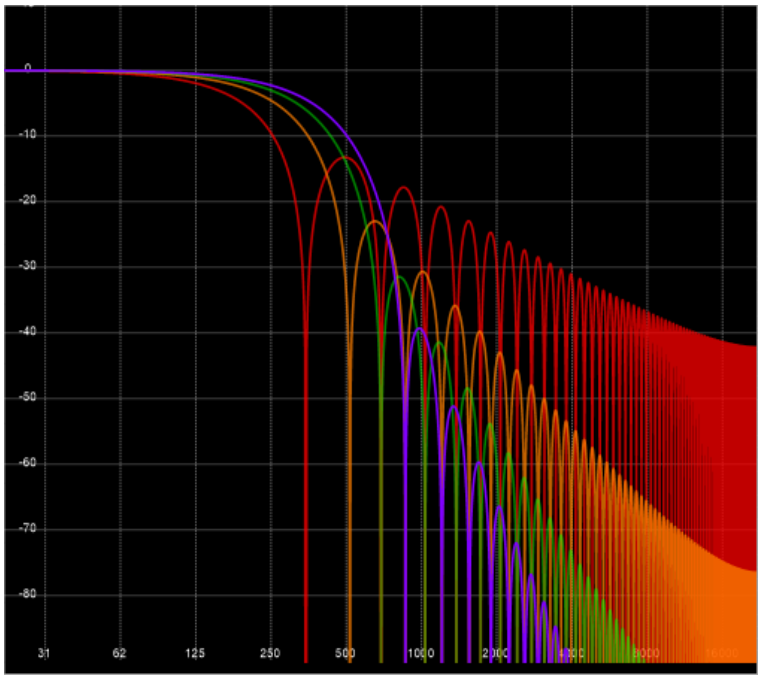

Figure 6: Normalized magnitude responses of various smoothing filters: box (red), half-sine (orange), Hann (green), and $\sin^3$ (magenta). Same parameters as in figure 5.

Because the filter evaluation according to eqs.(4) or (12) is still quite efficient and independent of filter size N, we can afford to combine several such filters to better achieve a given design goal. I will discuss smoothing filters as an example, but the concept may be easily applied to other prototypes as well.

The simplest smoothing filter is the running average filter (or box filter for short) introduced in section 3. If we cascade two box filters in series, we get a two times steeper falloff, i.e. 12

dB per octave. However, the resulting latency will be twice the latency of a single box filter. To compensate for it, we may take two box filters of length $\frac{N}{2}$ each.

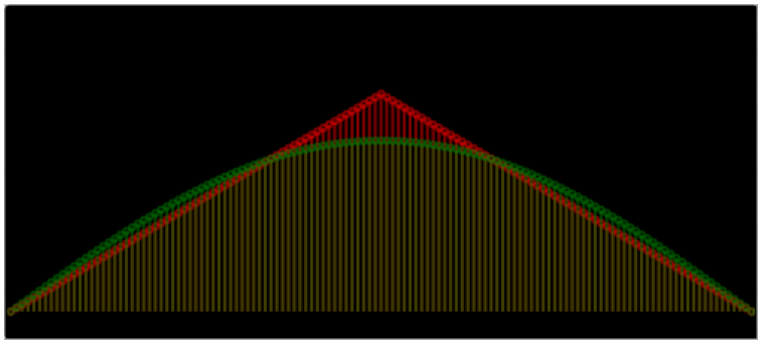

Figure 7: Impulse responses of two smoothing filters: two box filters in series (red), and a half-sine filter (green). Total IR length is $N = 128$ in both cases.

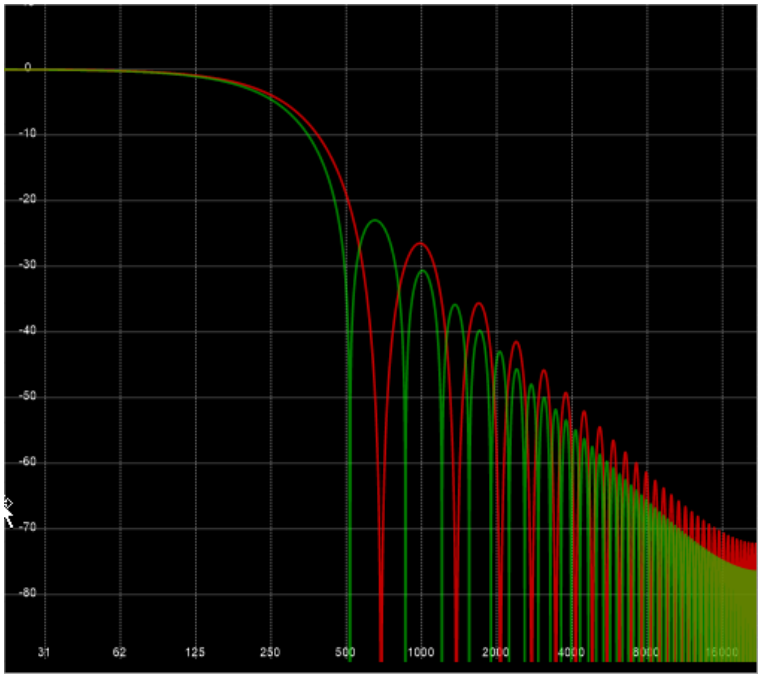

Figure 8: Normalized magnitude responses of two smoothing filters as in figure 7.

The cascaded box filter may be compared to the half-sine filter of the same length, refer to figures 7 and 8. Both have a 12 dB per octave roll-off, however the attenuation of the half-sine filter is slightly better by some 4 dB. The iteration scheme for the half-sine, eq.(12) is slightly more complex than eq.(4) for the box filter, even if the latter has to be run twice for two box filters in series. Another consideration is that the filter size has to be integer, which leaves a finer grid for the half-sine filter.

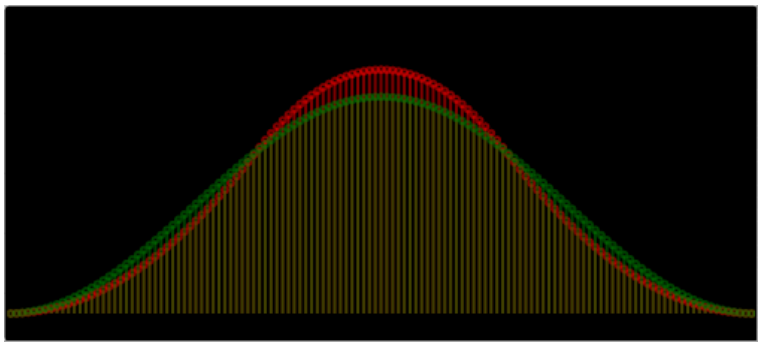

Figure 9: Impulse responses of two smoothing filters: three box filters in series (red), and Hann filter (green). Total IR length is $N = 129$ in both cases.

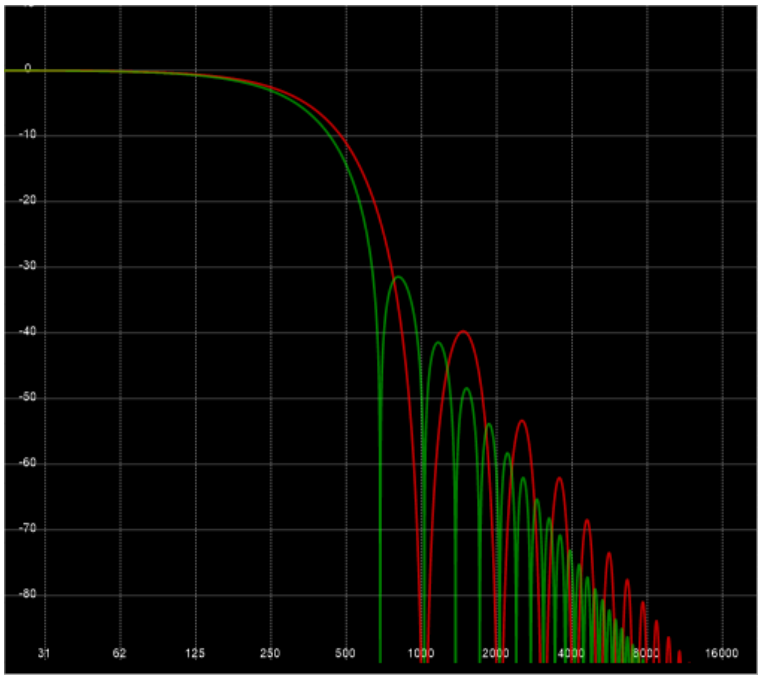

Figure 10: Normalized magnitude responses of two smoothing filters as in figure 9.

Another option to facilitate more flexible IRs is to use filters in parallel. For example, the bell-shaped IR

$$h_n = 1 - \cos(n\omega_0) \qquad \text{with} \quad \omega_0 = \frac{2\pi}{\mathrm{N}} \tag{14}$$

may be implemented as a box filter in parallel with a sliding Goertzel tapped at the real output u_n in eq.(12), subtracting the latter from the box filter output. Because eq.(14) represents a Hann window, we may call the corresponding filter a Hann filter. The roll-off is 18 dB per octave, and notches are at $\omega = \frac{4\pi}{N}, \frac{6\pi}{N}, \frac{8\pi}{N}, \ldots$ Compared to three box filters in series with same total length (refer to figures 9 and 10), it has 9 dB more attenuation.

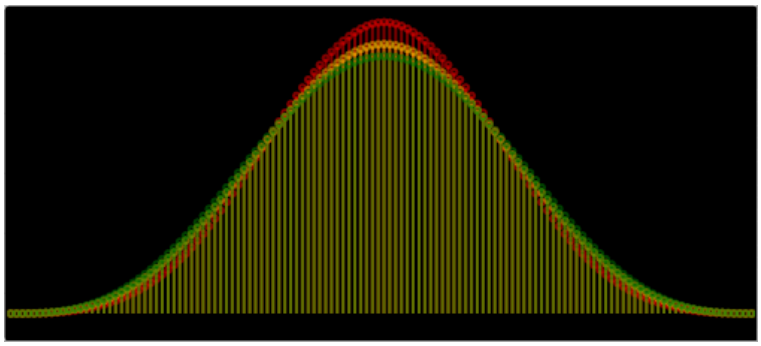

Figure 11: Impulse responses of three smoothing filters: four box filters in series (red), two half-sine filters in series (orange), and a $\sin^3$ filter (green). Total IR length is $N = 128$ in all three cases.

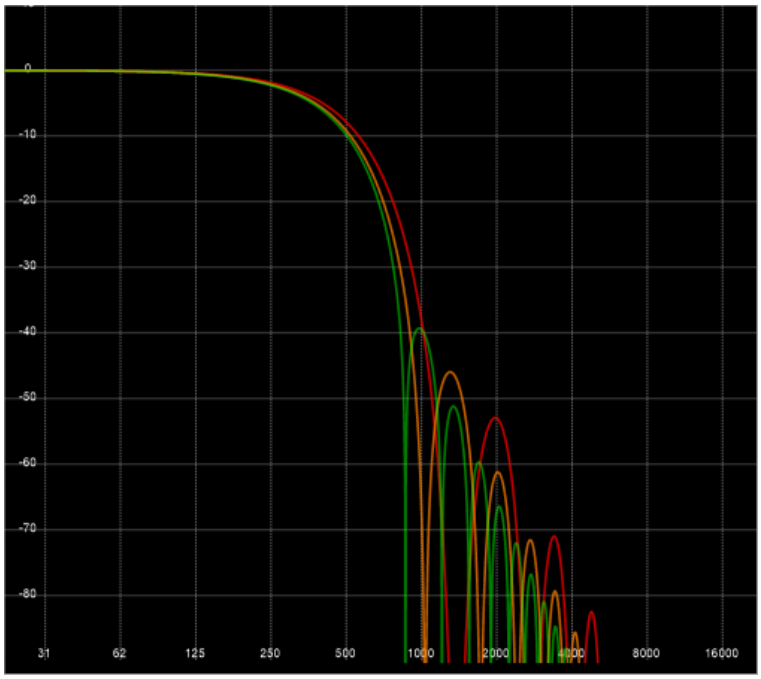

Figure 12: Magnitude responses of three smoothing filters as in figure 11.

An IR even closer to a Gaussian than the Hann window is given by

$$\begin{aligned} h_n &= \sin^3(n\omega_0) \\ &= \tfrac{3}{4}\sin(n\omega_0) - \tfrac{1}{4}\sin(3n\omega_0) \qquad \text{with} \quad \omega_0 = \frac{2\pi}{\mathrm{N}} \end{aligned} \tag{15}$$

The corresponding filter, called $\sin^3$-filter, may be implemented as a weighted sum of two parallel sliding Goertzel filters tapped at the imaginary output each. That filter has a 24 dB per octave roll-off and notches at $\omega = \frac{5\pi}{N}, \frac{7\pi}{N}, \frac{9\pi}{N}, \ldots$. The impulse and magnitude responses of the $\sin^3$ filter are shown in figures 11 and 12, respectively, together with other similar filters. Compared to four box filters in series with same total length (also 24 dB per octave),

attenuation is better by 13 dB. It may also be compared to two half-sine filters in series with equal total length (likewise 24 dB per octave). Attenuation of the $\sin^3$-filter is still better by 5 dB.

The outlined scheme allows for many possible variations to play with. For instance, one may cascade filters of different lengths to trade first side-lobe attenuation against stop-band suppression further out [3], or likewise adjust the weights in parallel structures (refer to Hann vs. Hamming window).

6 Discussion

A class of FIR filters has been examined that allow efficient recursive implementation. Corresponding filter kernels may be constant, exponentials, sines or cosines. By combining two or more such filters, more complex kernels may be created. Linear phase is easily achieved by choosing a symmetric kernel. Specific time domain requirements, such as avoiding overshoots, may easily be taken into account.

References

[1] Gardner, William G., "Efficient Convolution without Input/Output Delay", Audio Engineering Society Convention 97. Paper 3897, 1995. https://cs.ust.hk/mjg_lib/bibs/DPSu/DPSu.Files/Ga95.PDF

[2] Goertzel, G., "An Algorithm for the Evaluation of Finite Trigonometric Series", American Mathematical Monthly, 65 (1), 1958. https://courses.cs.washington.edu/courses/cse466/12au/calendar/Goertzel-original.pdf

[3] Geraint Luff, Cascaded box-filter smoothing filters, Signalsmith Audio Ltd. 2022. https://signalsmith-audio.co.uk/writing/2022/cascaded-box-filter-smoothing/