Digital Filters: Understanding the Mechanics

Martin Vicanek

11. January, 2023

1 Introduction

Filters are perhaps the most widespread devices in signal processing. Applications cover analysis, signal enhancement, noise reduction, and much more. The goal of this article is to provide some basic insight to the inner workings of digital filters, and in particular to show the relation between filter coefficients and filter characteristics. To this end we will need some math, however I'll try to restrict it to the absolutely necessary.

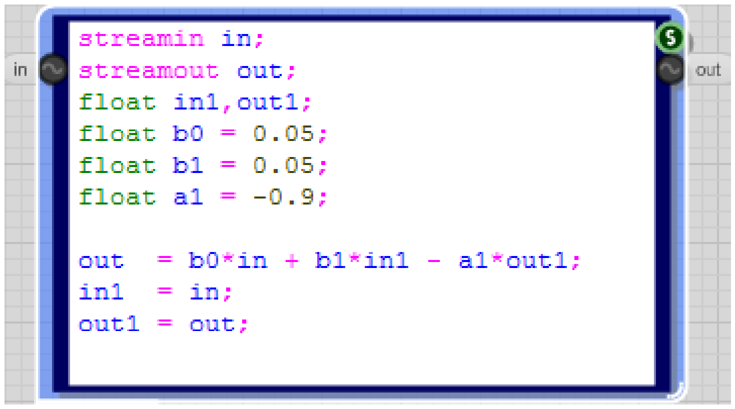
2 A Simple Filter

A digital filter acts on a digital signal, which is given as a sequence of incoming samples. The filter will modify the stream of samples and send out a resulting stream. Consider the first order recursive filter in figure 1.

It operates on the actual input sample `in` and the previous input and output samples `in1` and `out1`, respectively. These are multiplied by coefficients `b0`, `b1`, and `a1`, respectively, to yield the current output sample `out`. Then the actual samples `in` and `out` are assigned to the previous samples `in1` and `out1` for the next iteration.

Let's have a look at how our filter modifies some common input signals. In the graphs in figures 2-5, the light blue curve represents the input while the filtered signal is shown in red.

In general, we see that the amplitude is reduced by the filter, and the shape of the waveform gets distorted – except for the sine wave, where the original



```
streamin in;
streamout out;
float in1,out1;
float b0 = 0.05;
float b1 = 0.05;
float a1 = -0.9;

out  = b0*in + b1*in1 - a1*out1;
in1  = in;
out1 = out;
```

Figure 1: Simple digital filter code

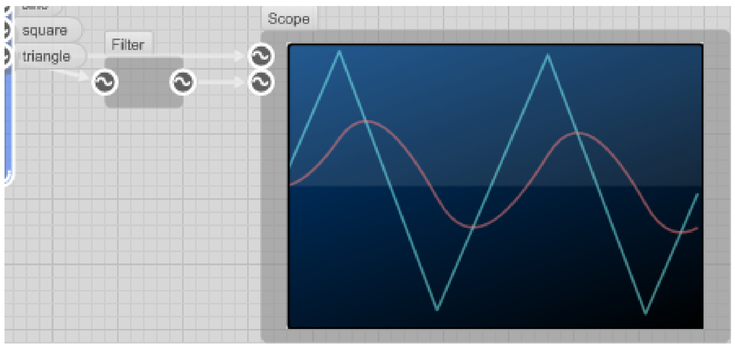


Figure 2: Triangle (light blue) and filtered triangle (red)

shape is retained (apart from attenuation and a delay or, more precisely, a phase shift in the filtered signal). For this reason, sine waves are particularly suited to test or characterize filters.

Fortunately, any incoming signal can be decomposed into a combination of sine waves with different frequencies, amplitudes, and phases. Once we know how the filter acts on each of these sine waves, we can recombine the results to get the resulting filter output for any input signal.

Obviously, the filter behavior is determined by the coefficients `b0`, `b1`, and `a1`, in the above first order filter example. Different choices will result in a lowpass, highpass, low shelf, high shelf, or allpass filter. (Other types like bandpass, notch or peaking filter would require a slightly more complex filter,

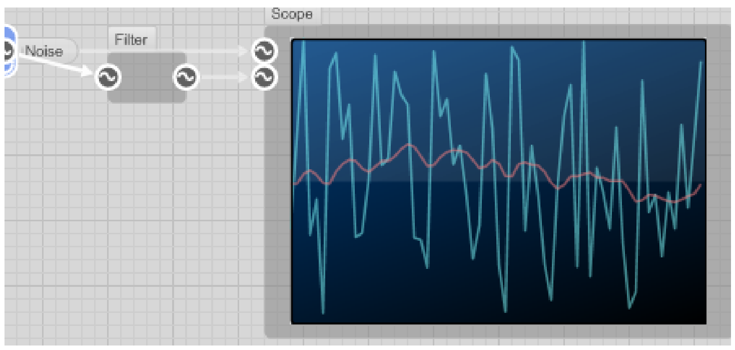


Figure 3: White noise (light blue) and filtered noise (red)

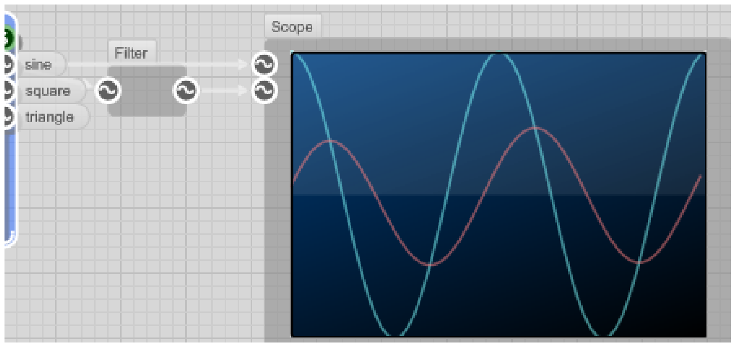


Figure 4: Sine (light blue) and filtered sine (red)

e.g. a biquad.) The relation between filter coefficients and filter characteristics is not obvious even for this simple filter, therefore we will try to gain insight in the following. Ultimately we want to be able to design a filter according to a given specification.

3 Figuring out the Filter

Let us begin by writing down the filter recursion equation. We will use letters x and y instead of words `in` and `out` for the input and output samples, respectively. At the nth iteration step, the current input sample will be denoted by x_n while the previous sample will be denoted by x_{n-1}. Likewise for the output samples. Using this notation, the first code line of our filter reads

$$y_n = b_0 x_n + b_1 x_{n-1} - a_1 y_{n-1} \tag{1}$$

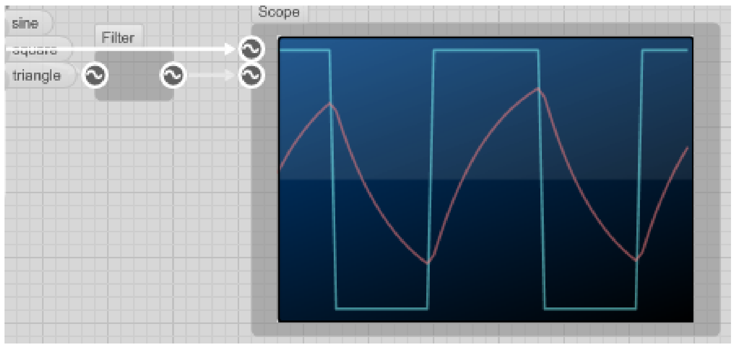


Figure 5: Square (light blue) and filtered square (red)

This equation can be solved by direct iteration: each new output value is computed from the current and previous input and output values – but that's exactly what our filter code does! There is a better way, a trick that allows us to solve for the entire output signal in one single step! Read on.

4 The Z-Transform ...

The so-called Z-transform is given by multiplying each sample with the power of some number z and taking the sum over all nonzero samples:

$$X(z) = \sum x_n z^{-n} \tag{2}$$

A similar transform may be applied to the output samples y_n. The expression on the right hand side of equation (2) yields different values for different z, hence we obtain a function $X(z)$. Applying the Z-transform to the filter equation (1) is straight forward for the x_n and y_n terms; however, the other terms involving the previous samples x_{n-1} and y_{n-1} require special consideration. It can be shown that shifting the index towards the previous sample amounts to multiplying by z^{-1} in the transformed domain. That's why z^{-1} is sometimes called the unit delay operator.

Putting it all together, we get the following transformed filter equation

$$Y(z) = b_0 X(z) + b_1 z^{-1} X(z) - a_1 z^{-1} Y(z). \tag{3}$$

Equation (3) is much simpler than the original equation (1) because it establishes a relation between the input X and the output Y for the same z.

Hence, with a modest amount of algebra, we can solve for Y to obtain the following simple expression:

$$Y(z) = H(z)X(z). \tag{4}$$

In equation (4), $H(z)$ denotes the transfer function and is given by

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}. \tag{5}$$

To summarize the result obtained so far: in the transformed domain, the action of the filter is a simple multiplication of the input signal $X(z)$ by the transfer function $H(z)$. The transfer function is a rational function of z.

In order to actually make use of this result, we need to elaborate a bit on the meaning of the z parameter in the following.

5 . . . Demystified!

There is a close relation between the Z-transform in equation (2) and the Fourier transform - so close that it is fair to call the Z-transform a disguised Fourier transform! To see this, substitute for z the expression $e^{j\omega}$, where $\omega = 2\pi f/\text{samplerate}$ is the (dimensionless) angular frequency (in radians) and $j = \sqrt{-1}$ is the imaginary unit (yes, we will use complex numbers here). Then equation (2) becomes

$$X(\omega) = \sum x_n e^{-jn\omega}, \tag{6}$$

where the right hand side is an ordinary Fourier series. Thus the left hand side of equation (6), $X(\omega)$, is simply the frequency spectrum of the input sequence x_n. The same interpretation applies to all other functions of z.

With this insight we can think of equation (4) in terms of frequency, if we substitute $e^{j\omega}$ for z. The frequency range from DC ($\omega = 0$ or, equivalently, $z = 1$) to the Nyquist limit ($\omega = \pi$ or $z = -1$) maps, on the z-plane, onto the upper half of the unit circle. Formally we also include negative frequencies to complete the unit circle. The situation is depicted in figure 6. In a nutshell, the Z-transform is a disguised Fourier transform, and the z variable is an encoded frequency variable.

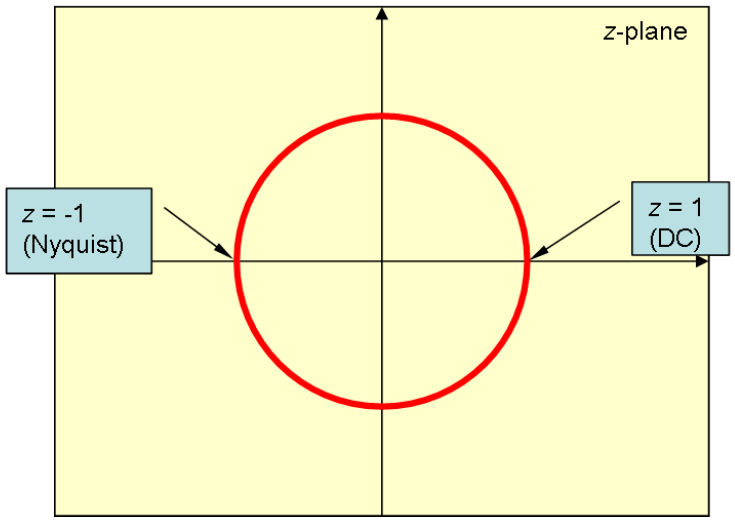


Figure 6: Magnitude response of various high-shelf filters. Solid lines: present work using equation (12). Dashed lines: analog prototype.

6 Designing Filters

We are now ready to design a filter. Our first example will be a DC blocker, i.e. a high pass filter which will be transparent for audio frequencies and will reject subsonic frequencies. Blocking DC means that the transfer function $H(z)$ is zero for $z = 1$. From equation (5) we see that this requirement implies $b_1 = -b_0$, i.e. the two coefficients have equal magnitude but opposite sign. At the other end of the spectrum, the filter is supposed to be transparent, so $H(z) = 1$ for $z = -1$. Putting this into equation (5), we get

$$b_0 = -b_1 = (1 - a_1)/2.$$

The remaining coefficient a_1 determines the cutoff frequency ω_c which separates the pass band from the stop band. You may derive the relation from equation (5) by setting $|H(e^{j\omega_c})|^2 = \frac{1}{2}$; the result is

$$a_1 = \frac{-\cos(\omega_c)}{1 + \sin(\omega_c)}. \tag{7}$$

For a 20 Hz cutoff frequency and assuming a sample rate of 44100 Hz we get $\omega_c = 2\pi \cdot 20/44100 = 0.00285$, and from there the DC blocker filter coefficients:

$$a_1 = -0.99715, \quad b_0 = 0.99858, \quad b_1 = -0.99858.$$

As a second example, we will design a smoothing or low pass filter. We will want unity gain for DC, $H(z) = 1$ for $z = 1$, and full rejection at Nyquist, $H(z) = 0$ for $z = -1$. Putting this into equation (5) yields

$$b_0 = b_1 = (1 + a_1)/2.$$

We choose a 740 Hz cutoff frequency and assume a 44100 Hz sample rate, so $\omega_c = 2\pi \cdot 740/44100 = 0.1054$. Using this value in equation (7) yields the low pass filter coefficients:

$$a_1 = -0.9, \quad b_0 = 0.05, \quad b_1 = 0.05.$$

If you scroll up, you'll notice that these are the same coefficients as those used in our code example.

So now you know how first order filters work. Wanna try biquads next? ;)