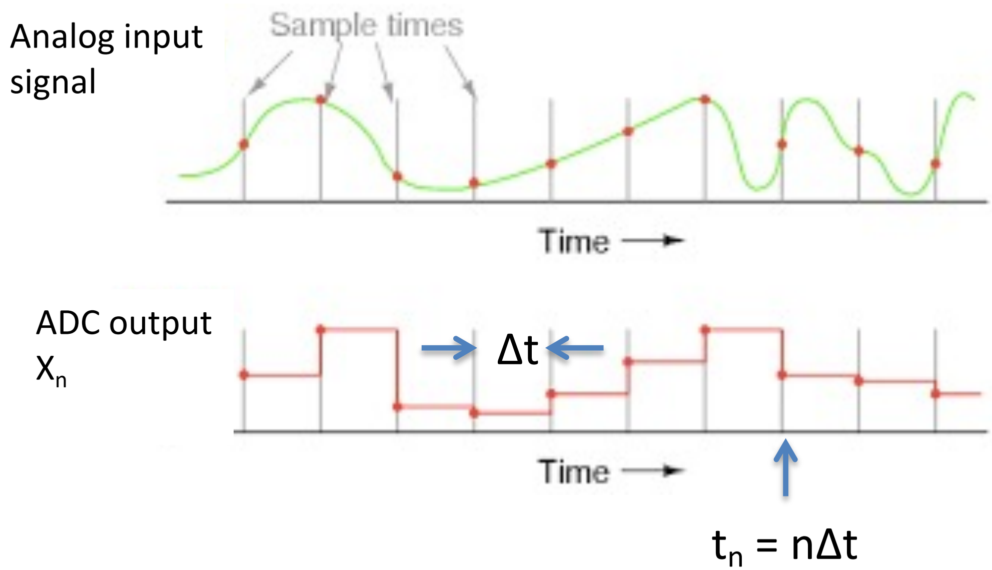
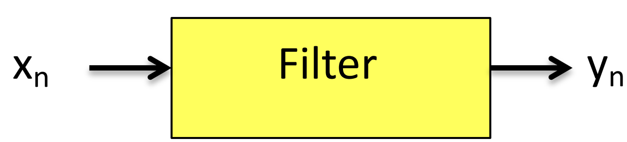
Digital Filters – A brief introduction

The top figure shows an analog waveform that has been *sampled*. The combination of a sample and hold circuit and an analog to digital converter (ADC) samples the analog waveform at the times shown by the dots and performs the conversion. The entire process of sampling, holding and converting takes some time Δt. The digital output looks like the lower figure. Let x_n be the output of the ADC corresponding to time $t_n = n\Delta t$. Now imagine storing all of the x_n in memory so we can go back and analyze it at leisure.



https://forum.allaboutcircuits.com/threads/analog-vs-digital-waveforms.157784/

As we'll see, digital filtering is a sophisticated form of averaging. Rather that V_{in} and V_{out}, it is customary to call x_n the input to the filter at time t_n and y_n the output at t_n.



Non-recursive filters

Consider a filter which looks at the data in memory and performs the following operation,

$$y_n = a x_{n-2} + b x_{n-1} + c x_n + b x_{n+1} + a x_{n+2}$$

Like any filter we need to know its frequency response. To do that, use phasors to represent the digital data at each point in time.

$$x_n = \mathrm{Re}\left(\hat{x}(\omega)e^{i\omega t_n}\right) = \mathrm{Re}\left(\hat{x}(\omega)e^{i\omega n\Delta t}\right) \qquad y_n = \mathrm{Re}\left(\hat{y}(\omega)e^{i\omega t_n}\right) = \mathrm{Re}\left(\hat{y}(\omega)e^{i\omega n\Delta t}\right)$$

Now substitute the phasors into the expression for the filter,

$$\hat{y}e^{i\omega n\Delta t} = a\hat{x}e^{i\omega n\Delta t}e^{-2i\omega\Delta t} + b\hat{x}e^{i\omega n\Delta t}e^{-i\omega\Delta t} + c\hat{x}e^{i\omega n\Delta t} + b\hat{x}e^{i\omega n\Delta t}e^{i\omega\Delta t} + a\hat{x}e^{i\omega n\Delta t}e^{2i\omega\Delta t}$$

Solving for the transfer function we obtain,

$$T(\omega) = \frac{\hat{y}}{\hat{x}} = ae^{-2i\omega\Delta t} + be^{-i\omega\Delta t} + c + be^{i\omega\Delta t} + ae^{2i\omega\Delta t}$$

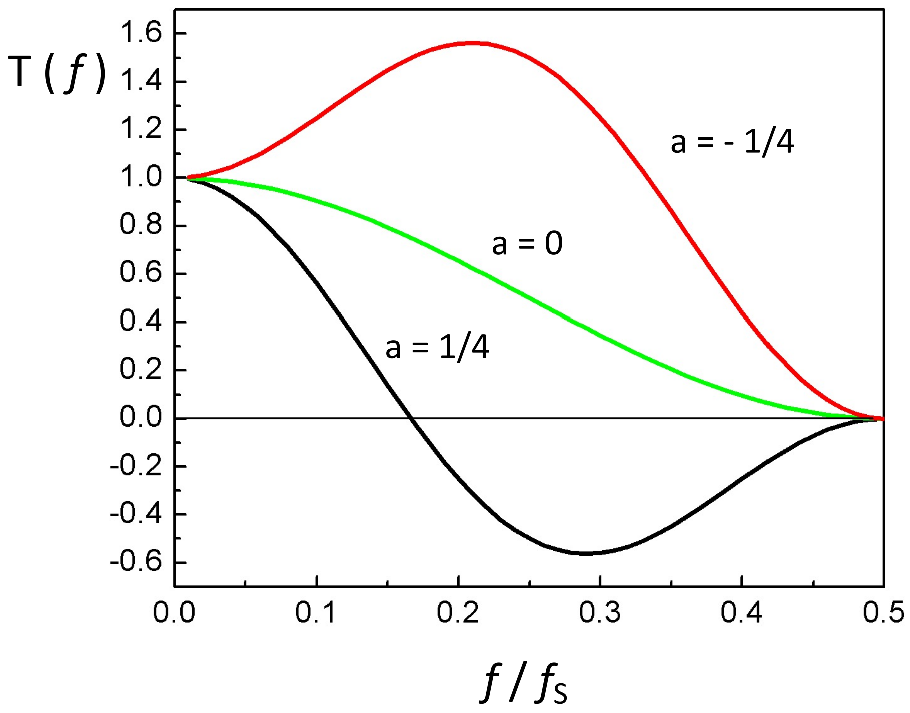
$$T(\omega) = c + 2b\cos(\omega\Delta t) + 2a\cos(2\omega\Delta t)$$

Because the filter expression is *symmetric* about t_n, the transfer function is real. Remember that Δt is a constant fixed by the properties of the ADC. There are 3 parameters, a, b and c. Suppose we want this to be a *low pass* filter. In that case, set T(0) = 1 which implies 1 = c + 2b + 2a. It would also be nice to have the filter go to zero at the Nyquist frequency, f_N = 1/2Δt. That yields a second condition: 0 = c - 2b + 2a. These two conditions imply b = ¼ and c = ½ - 2a so the free parameter *a* now determines the remaining shape of the filter.

The plot shows the transfer function T for a = ¼, 0 and -1/4. T is plotted as a function of f / f_S where $f_S = 1/\Delta t$, the sampling frequency. $f / f_S = 0.5$ corresponds to the Nyquist frequency. For a = ¼ , T crosses zero. That might be useful if you were trying to eliminate that particular frequency. The (-) sign corresponds to a 180° phase shift.

This filter is *non-causal*, meaning that the output at time t_n requires input data at times *after* t_n. That's perfectly okay if the data is stored somewhere and you come back later to analyze it, but it obviously cannot function as a filter in *real time* since we can't know the future. A causal filter might have the form,

$$y_n = c_o x_n + c_1 x_{n-1} + c_2 x_{n-2} + \ldots$$



The filter is also *non-recursive,* meaning that the output is purely a function of the input data. It does not require knowledge of the filter output at earlier times.

Recursive Filters

Now consider the following filter,

$$y_n = a\, y_{n-1} + c\, x_n$$

In this case, the output at time t_n depends on *both* the input data at t_n and the output of the filter at an earlier time. This is the simplest example of a *recursive* filter. This one is also causal and could be used to filter data in real time.

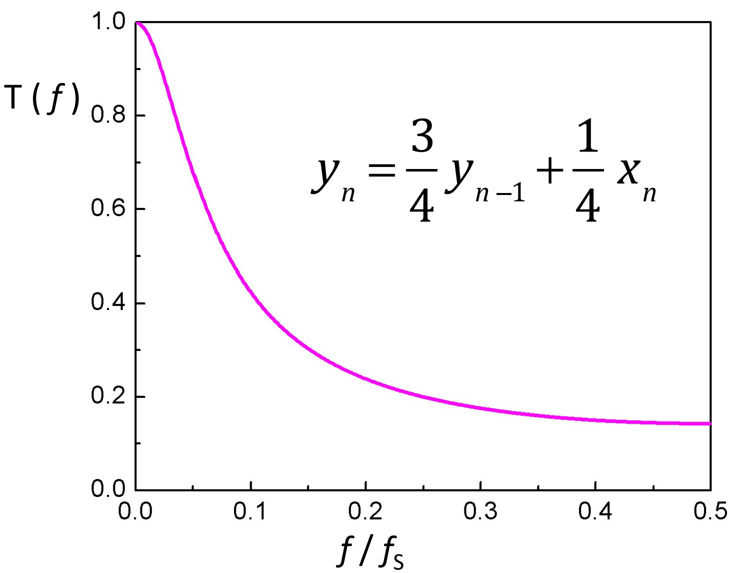
Let, $$y_{n-1} = \mathrm{Re}\left(\hat{y}(\omega) e^{i\omega t_{n-1}}\right) = \mathrm{Re}\left(\hat{y}(\omega) e^{i\omega n\Delta t} e^{-i\omega\Delta t}\right)$$

Substitute the phasors into the expression for the filter and cancel the common exponential factor to obtain the transfer function,

$$\hat{y} = a\,\hat{y}\,e^{-i\omega\Delta t} + c\,\hat{x} \;\Rightarrow\; \frac{\hat{y}}{\hat{x}} = T(\omega) = \frac{c}{1 - a e^{-i\omega\Delta t}}$$

Let's again make it a low pass filter and set the magnitude of the transfer function to 1 at ω = 0. Also use real coefficients. The transfer function is given by,

$$\left|T(0)\right| = \frac{c}{1-a} = 1 \;\Rightarrow\; \left|T(\omega)\right| = \frac{1-a}{\sqrt{1 - 2a\cos\omega\Delta t + a^2}}$$



I've plotted the transfer function for a = ¾. It's a low pass filter with a 3 dB frequency of 0.46 f_S. The horizontal axis goes up to 0.5 which corresponds to the Nyquist frequency. There's no need to plot it for higher frequencies since it is symmetrical about the Nyquist frequency. Moreover, we would try to eliminate any signals with frequencies near to or above the Nyquist frequency with an anti-aliasing filter ahead of the ADC.

It's instructive to return to the complex form of the transfer function and write it in terms of $f = \omega/2\pi$ and the Nyquist frequency,

$$T(f) = \frac{1-a}{1-ae^{-i\omega\Delta t}} = \frac{1-a}{1-ae^{-i2\pi f\Delta t}} = \frac{1-a}{1-ae^{-i2\pi f/f_S}} = \frac{1-a}{1-ae^{-i\pi f/f_N}} \qquad f_N = \frac{1}{2\Delta t}$$

Let's assume that any signals reaching the filter will be well below the Nyquist frequency. In that case we can expand the exponential in the denominator and rearrange things,

$$T(f) \approx \frac{1-a}{1-a\left(1-i\pi f/f_N+\ldots\right)} = \frac{1}{1+ai\pi f/\left(1-a\right)f_N} = \frac{1}{1+i\,f/f_0}$$

$$f_0 = f_{3\,dB} = f_N\frac{1-a}{\pi a} = \frac{1}{2\pi\Delta t}\left(\frac{1-a}{a}\right)$$

This now looks just like a low pass filter whose 3 dB frequency is set by the sampling time Δt and the filter coefficient a. The *larger* the value of a, the *lower* the 3 dB frequency. Since *a* is the coefficient of the previous output value, we might say that as the filter weights the past (y_{n-1}) more heavily than the present (x_n) the the lower its 3 dB frequency will be. Using a = 0.75, this expression gives a 3 dB frequency of 0.053 f_S. That's a little larger than the value of 0.046 f_S we obtained using the exact transfer function on the previous page. The approximate expression given above is often good enough.

Higher order filters

As we've seen with analog filters, selectivity requires more stages. This is where digital filters come into their own since adding more stages is just a matter of more calculations. As a simple example, consider the recursive filter defined by,

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + c_0 x_n$$

Substituting phasors as done earlier gives,

$$T(f) = \frac{c_0}{1 - a_1 e^{-i 2\pi f \Delta t} - a_2 e^{-i 4\pi f \Delta t}}$$

Choose $a_2 = -(a_1)^2/4$ to obtain,

$$T(f) = \frac{c_0}{\left(1 - \frac{a_1}{2} e^{-i 2\pi f \Delta t}\right)^2}$$

By expanding the exponential for frequencies much less than the Nyquist frequency we obtain,

$$T(f) \approx \frac{c_0}{\left(1 - \frac{a_1}{2}\left(1 - i 2\pi f \Delta t + \ldots\right)\right)^2} = \frac{c_0}{\left(1 - a_1/2\right)^2} \frac{1}{\left(1 + i \frac{4\pi a_1}{2 - a_1} f \Delta t\right)^2} = \frac{A}{\left(1 + i f/f_0\right)^2} \qquad f_0 = \frac{2 - a_1}{4\pi a_1} f_S$$

This is a *2-pole* filter with a cutoff frequency f_0. Going beyond these simple examples to find the filter coefficients for specific design requirements (number of poles, cutoff frequency, Butterworth, Chebyshev, etc.) is mathematically involved and requires something called the *z-transform*. It's interesting but, as they say, outside the scope of this discussion. The good news is that there are programs in Matlab, Mathematica, LabView and so on that will work out the filter coefficients for you.

Impulse response

If you work with digital filters you will encounter terms like FIR and IIR. FIR is short for *finite impulse response*. Consider an impulse $x_n = 1$ sent to the digital filter at time t_n. For all other n , $x_n = 0$. Suppose the filter is nonrecursive and for the sake of argument, has just 2 terms:

$$y_n = c_0 x_n + c_1 x_{n-1}$$

The outputs for n, n+1 and n+2 are,

$$y_n = c_0 x_n + c_1 x_{n-1} = c_0$$

$$y_{n+1} = c_0 x_{n+1} + c_1 x_n = c_1$$

$$y_{n+2} = c_0 x_{n+2} + c_1 x_{n+1} = 0$$

$$y_{n+3} = 0 ,...$$

For times corresponding to n+2 or later, the filter output is zero. The response to the impulse is *finite* so it's called an **FIR** filter. No matter how many terms on the right side, a non-recursive filter will have a finite impulse response.

Now consider the recursive filter, $y_n = a\, y_{n-1} + c\, x_n$ subject to the same impulse. The outputs are,

$$y_n = a\, y_{n-1} + c\, x_n = c$$

$$y_{n+1} = a\, y_n + c x_{n+1} = ac$$

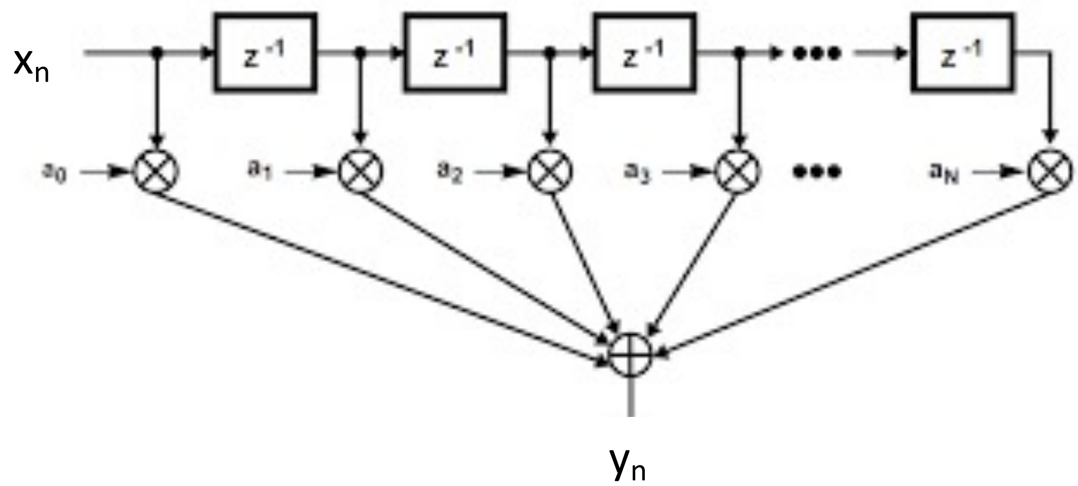
$$y_{n+2} = a\, y_{n+1} + c\, x_{n+2} = a^2 c$$

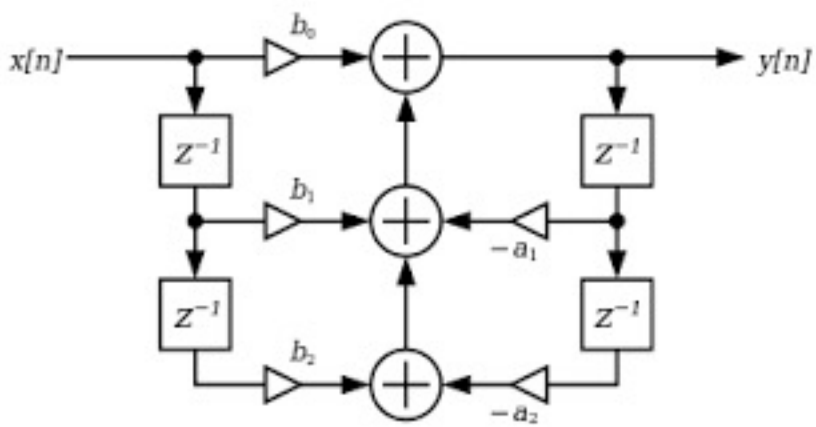
$$y_{n+3} = a^3 c, \quad etc.$$

The output never goes away. If a < 1 then it just gradually dies away as time goes to infinity. This type of filter is called **IIR** for *infinite impulse response*. IIR filters generally require fewer computations to achieve the same filtering action as FIR filters so they are preferred in many cases. However, since they involve feedback, they may be unstable.

You'll see diagrams like the one below that are essentially flow charts or schematics for digital filters. Each box labeled z^{-1} corresponds to a delay of one sampling time. The *a*'s are the filter coefficients and the circles with a cross indicate multiplication. The little down arrows are sometimes called *taps*. This diagram would correspond to the following **FIR filter**,

$$y_n = \sum_{k=0}^{N} a_k \, x_{n-k}$$





https://en.wikipedia.org/wiki/Digital_filter

This diagram, on the other hand, has *feedback* so it corresponds to an **IIR filter**:

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} - a_1 y_{n-1} - a_2 y_{n-2}$$

The theory of digital filtering is rather mathematical. A simple introduction, from which my discussion is taken, can be found in *Digital Filters*, by R.W. Hamming. Hamming was one of the pioneers of this subject and his book is characteristic of the Bell Labs style – doing as much as possible with minimal mathematics.