Stephen Schoen
May 31, 2018

An Introduction to Digital Filters

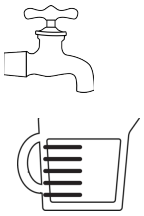
An Introduction to Integration

Much of the characteristic sound of classic synthesizers comes from the creative use of filters. In the "East Coast" or subtractive approach to synthesis pioneered by Robert Moog, a harmonically rich waveform is shaped by the use of one or more filters.

A filter is any device that accepts an input signal, modifies it in some way and produces an output signal. Mathematically we say that a filter has a *transfer* function that defines the output for a given input. Determining this function allows us to model the behavior of the filter.

The types of filters we will be discussing modify the frequency spectrum of the input signal, that is, they filter out some portion of the frequencies present in the input signal. The bass and treble controls on an amplifier are a good example of this type of filter.

At the heart of every frequency spectrum filter is a component that stores some portion of the the input signal. This component is often referred to as an "integrator". For those unfamiliar with calculus the term may be confusing. In this article I hope to explain the concept of integration and how it relates to filter design.



Picture a measuring cup under a faucet. Initially the cup is empty. If we turn on the faucet, the cup will gradually fill with water. If we know the flow rate of the faucet, integration allows us to answer the question , "how much water is in the cup?". Simple you say. Just read the marking on the cup. In this case the cup is serving as our integrator.

What if, instead, the water is flowing down the drain and all we know is the flow rate of the faucet (measured in gallons per minute) and the time. If the faucet is flowing at a constant rate,

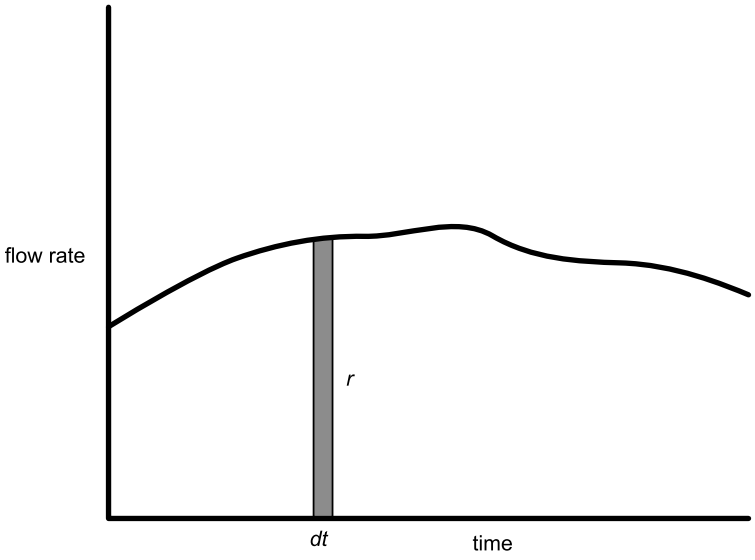
the problem is simple. If the flow rate is r and the faucet is on is for m minutes then the total amount of water is r times m. If we call the total amount of water w, then we can say $w = rm$.

What if we adjust the faucet while the water is running? Now how will we figure out how much water has flowed?. If we know the flow rate r at each instant we should still be able to figure this out. Let's call each instant of time delta t, or dt for short. The amount of water that flows in this instant is $r\,dt$. If we add all of these amounts up we should have the total amount of water. This process is referred to as integration. It is expressed using an integral symbol $\int$ which represents the summation of all the instantaneous values We can now write our equation as:

$$w = \int_{t=0}^{m} r\,\mathrm{d}t$$

Translated, this means "take the flow rate at each instant and multiply it by the length of each instant of time and add them all up from time = 0 to time = m.

Here's a graph of our water flow:

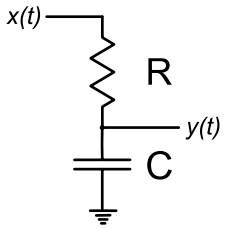


For each little slice of time we are multiplying the height of the slice by its width, or calculating it's area. The sum of all these areas is the value of the integral, which is also the total area under the curve. Because each slice is actually a small rectangle, in order to obtain the

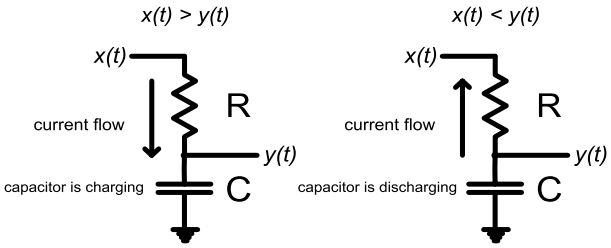
correct answer we must make the slices infinitely narrow. Otherwise the slope of the curve at the top of each rectangle will cause our calculation to be incorrect.

A Simple Analog Filter

We can apply the same process to analyzing the behavior of a simple filter. Consider a capacitor. A capacitor is a device for storing an electrical charge. This is just like our measuring cup, only it holds electrons rather than water. A capacitor stores electric charge in the same way our cup stores water and the electrons flowing through the wire into the capacitor are the same as the water flowing through our faucet. The flow rate of the faucet is equivalent to the current into the capacitor, and the height of water in the measuring cup is equivalent to the voltage across the capacitor.



Here we have a simple resistor - capacitor circuit. This circuit forms a simple lowpass filter. Let us call the input to the filter at time t, $x(t)$ and the output at time t, $y(t)$. When the input voltage $x(t)$ is greater than the output voltage $y(t)$, current flows through the resistor and the capacitor is charged, and when the input voltage is below the output voltage, the current flows the other way and the capacitor discharges.



Because the resistor and capacitor form a voltage divider, when the input voltage is changing, only a portion of the input signal appears at the output. This smooths out the fluctuations in the input signal. From the diagram you can see that the input voltage equals the sum of the voltage across the resistor and the voltage across the capacitor. If we label the voltage at time t across the capacitor as $v_c(t)$ and the voltage across the resistor as $v_r(t)$ we have:

$x(t) = v_c(t) + v_r(t)$

Since the output from the filter is connected to the top of the capacitor

$y(t) = v_c(t)$

Ohm's law states that the voltage across the resistor is equal to the current through the resistor multiplied by the resistance ($V = IR$) where I is the current and R is the resistance, so we know that the voltage across the resistor $v_r(t) = IR$

so $I = \frac{v_r(t)}{R}$

The voltage on the capacitor $v_c(t)$ multiplied by the capacitance C is equal to the total charge on the capacitor q_c.

$Cv_c(t) = q_c$

but from above $y(t) = v_c(t)$

so $Cy(t) = q_c$ or

$y(t) = \frac{q_c}{C}$

The charge q_c is also like the amount of water in our cup. In the same way that the total amount of water in the cup is the integral of the flow over time, the total charge on the capacitor is the integral of the current into the capacitor over time. So the total charge is given by:

$$q_c = \int_{t_0}^{t} I \mathrm{d}t$$

Since $y(t) = \frac{q_c}{C}$ we can say

$$y(t) = \int_{t_0}^{t} \frac{I}{C} \mathrm{d}t$$

From above we have $I = \frac{v_r(t)}{R}$ so

$$y(t) = \int_{t_0}^{t} \frac{v_r(t)}{RC} \mathrm{d}t$$

and since $x(t) = v_c(t) + v_r(t)$

$v_r(t) = x(t) - v_c(t)$ so if we substitute for $v_r(t)$

$$y(t) = \int_{t_0}^{t} \frac{x(t) - v_c(t)}{RC} \mathrm{d}t$$

$$y(t) = \int_{t_0}^{t} \frac{1}{RC}(x(t) - v_c(t))\mathrm{d}t$$

We know that $y(t) = v_c(t)$ so

$$y(t) = \int_{t_0}^{t} \frac{1}{RC}(x(t) - y(t))\mathrm{d}t$$

This assumes that the capacitor had no charge at time = 0. If the voltage on the capacitor at time = 0 is $y(t_0)$ then the voltage at any time is:

$$y = y(t_0) + \int_{t_0}^{t} \frac{1}{RC}(x(t) - y(t))\mathrm{d}t$$

RC is referred to as the time constant of the filter and represents the decay time for the circuit. For resistance in ohms and capacitance in farads the time constant will be in seconds. The value $\frac{1}{RC}$ is a measure of the frequency response of our filter. $\frac{1}{RC}$ is expressed in radians per second and is typically replaced by ω_c which is the *cutoff* of the filter (in radians per second). Omega (ω) is often used to represent a circular frequency, and the c subscript is used to represent

cutoff. Because there are 2π radians in a circle, the cutoff frequency in Hz is given by $\frac{\omega_c}{2\pi}$. So, putting it all together gives us our final equation for our filter:

$$y = y(t_0) + \int_{t_0}^{t} \omega_c(x(t) - v_c(t))\mathrm{d}t$$

This equation describes the behavior of a simple RC lowpass filter network. This particular filter is a *first-order* filter, which means there are no terms in the equation that are squared, cubed etc. You will also see these referred to as single pole filters which is related to the representation of the filter's behavior using complex numbers.

All first order filters have a slope of 6 dB per octave which means that for each octave change in the input frequency the response will change by 6 dB. For a low pass filter such as this, that means that if our cutoff frequency is 400 Hz then at 800 Hz the output will be reduced by 6 dB and at 1600 Hz it will be reduced by 12 dB, etc. By using the tools of calculus it is possible to solve this equation for a given input signal and determine how the filter behaves. You can calculate the frequency response, phase response, etc. Fortunately it's not necessary for us to solve the equation in order to build a digital model of the filter.

A Digital Model of the Filter

All audio in an application like Audulus is represented digitally. This means that an audio signal is a stream of numbers representing the shape of the waveform. Each number is a measure of the height of the wave for that sample. The sample rate for Audulus is currently 44.1 kHz. This means that for each second of sound there are 44100 samples. For the analog filter the integrator in the filter is the sum of the instantaneous values of the equation. For our integration to be correct we have to make each instant infinitely small. In the digital realm, we don't have a continuous signal, we only have a series of values, one for each time we sample the signal, so it's not possible to make the interval infinitely small. The smallest slice we can work with is one sample wide.

As a first approximation, we can simply take our sample values, multiply them by the length of time for each sample and add them all up. Of course, if we had all the samples available, this

would be a simple process, but since we want to do this continuously as new samples arrive (we don't want to wait until we've finished making our sound before we hear it) we need a way to do it step by step. For each new sample we want to take our current value and add the new sample to it. Our integrator becomes:

integrator output= previous output + integrator input * the time for each sample. If our sample time is T and the input to the integrator for sample t is $In(t)$ and the output is $y(t)$ then:

$$y(t) = y(t-1) + In(t)T$$

where $y(t-1)$ represents the integrator output from the previous sample.

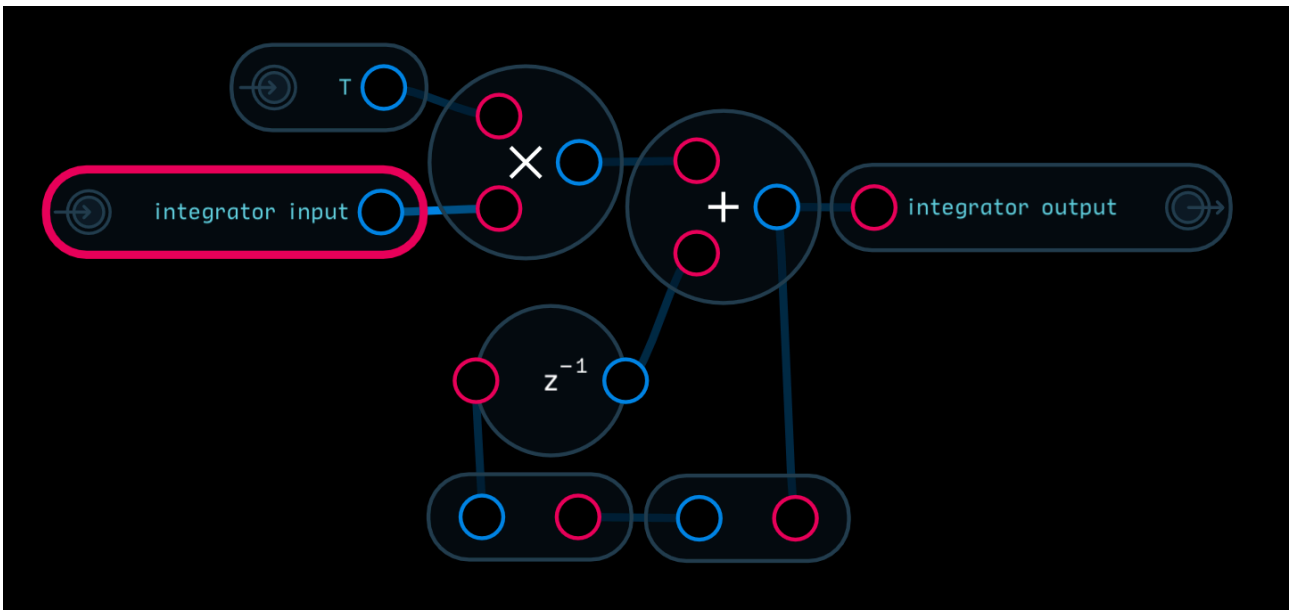
From the equation for the analog filter, we know that the input to the integrator for our low pass filter should be:

$$In(t) = (x(t) - y(t))\omega_c$$

So at any sample t (assuming $y(t_0) = 0$) the filter output will be given by

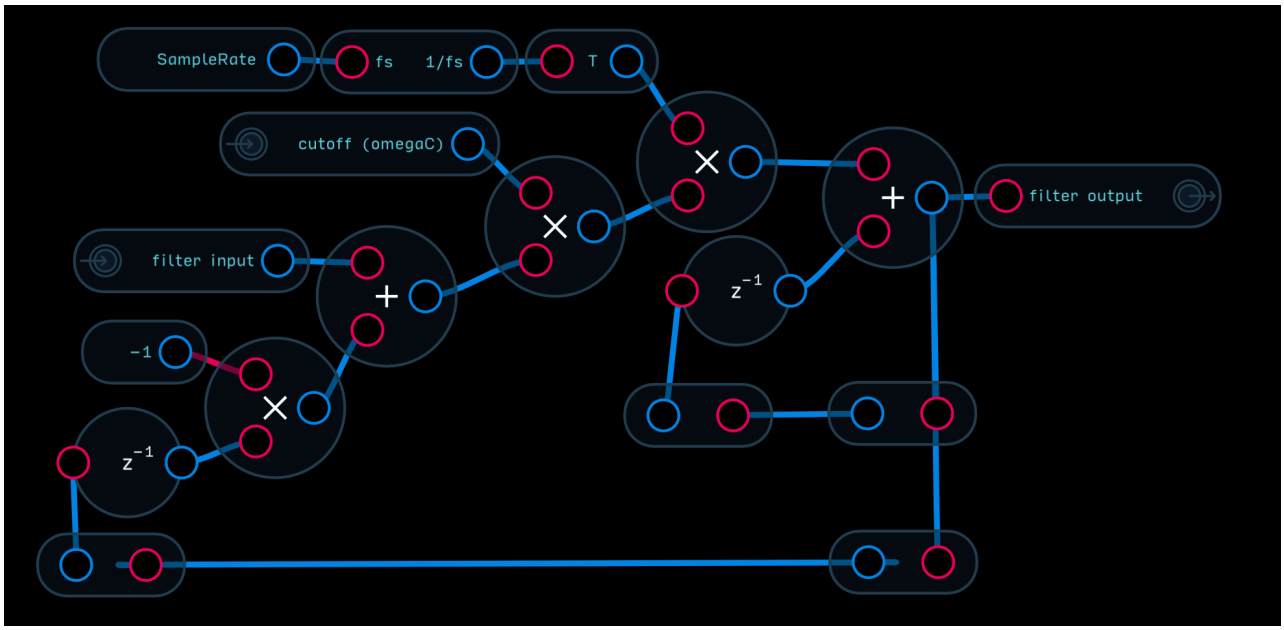
$$y(t) = y(t-1) + (x(t) - y(t-1))\omega_c T$$

From this you can see that we need a way to recall the value of our integrator for the last sample. Fortunately Audulus has just such a node. The Unit Delay node in Audulus (or the z^{-1} node) outputs the previous sample for its input, so if we add a unit delay to the output of our integrator, we will be able to access the previous value $y(t-1)$. We can construct an integrator in Audulus as follows:

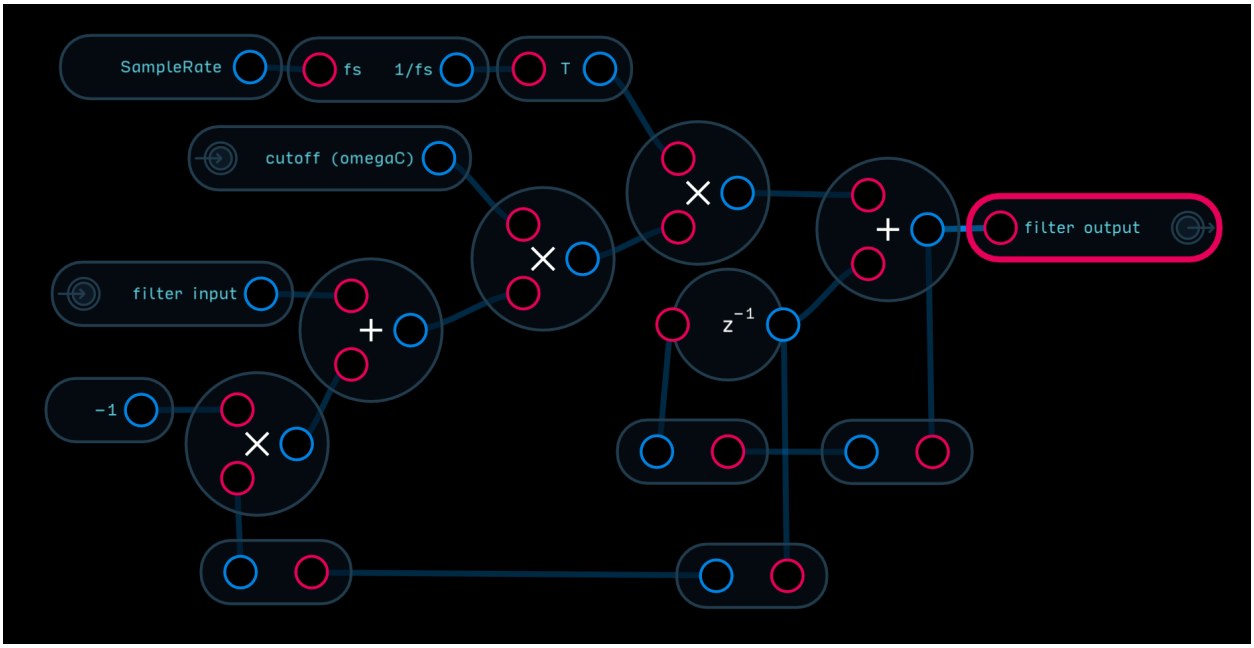


This takes an input value, adds the previous output value and sends it to output. We know that the sample rate for Audulus is 44.1 kHz so the time for each sample is 1/sample rate or in Audulus:

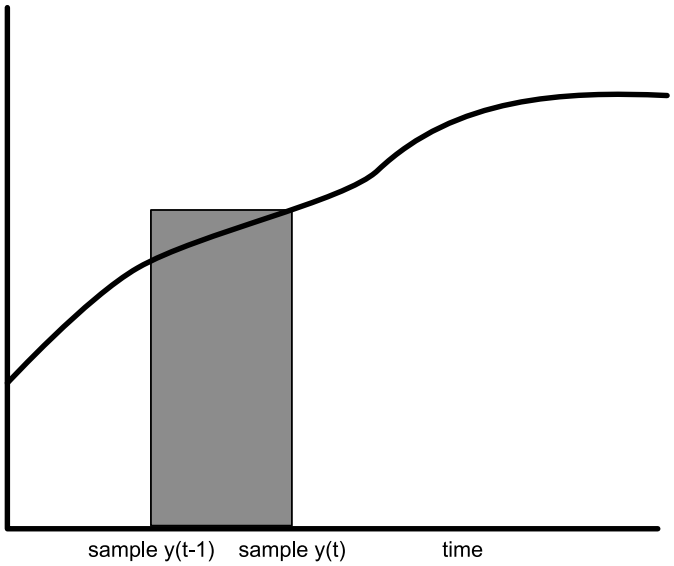
We can now build a model of our filter:



Note that the patch contains two unit delays, one is part of the integrator and the other for the previous sample that is added to the integrator. Since both z-1 nodes are attached to the integrator output, they both have the same value so we can simplify the patch by removing one of the unit delays. It would also be possible to condense the Add and Multiply nodes into a single Expression node, but this approach more clearly demonstrates the visual nature of programming in Audulus.

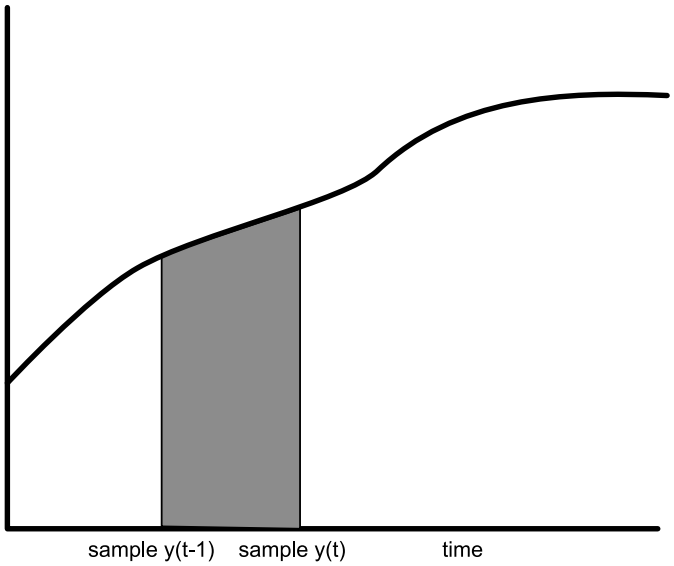


This is the simplest implementation of a first order low-pass filter, however it isn't a particularly good one. As we saw in the analog filter, when we integrate a function we are calculating the area under the curve of the function. Our current approach takes the value of sample t and multiplies it by the sample time T. This calculates a rectangular area.



This is okay as long as T is small compared to the rate of change from sample to sample, or to put it another way, so long as the maximum frequency is much less than the sample rate.

A better approximation is to calculate the area of the trapezoid from t-1 to t.



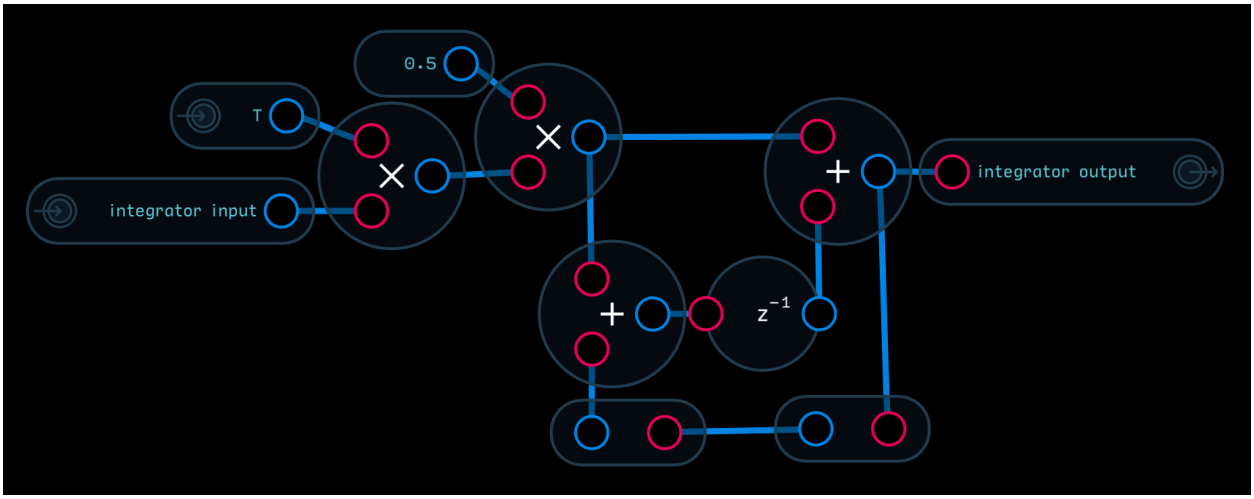
The area of the trapezoid is:

$$\left(\frac{y(t)-y(t-1)}{2}+y(t-1)\right)T$$

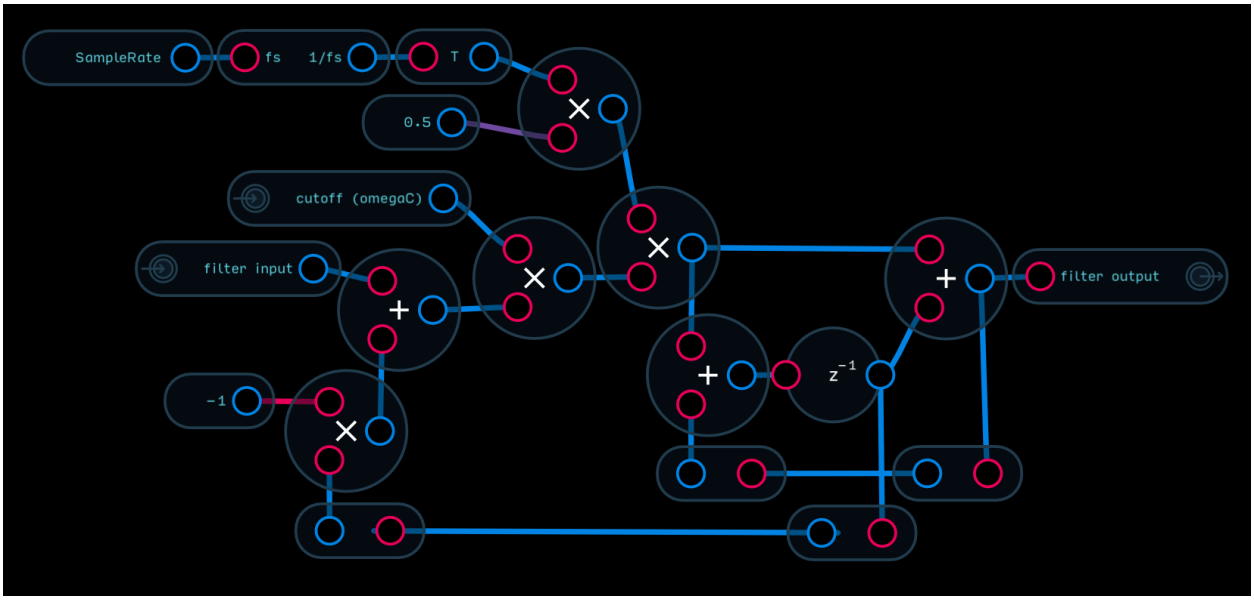
or

$$\frac{y(t)+y(t-1)T}{2}$$

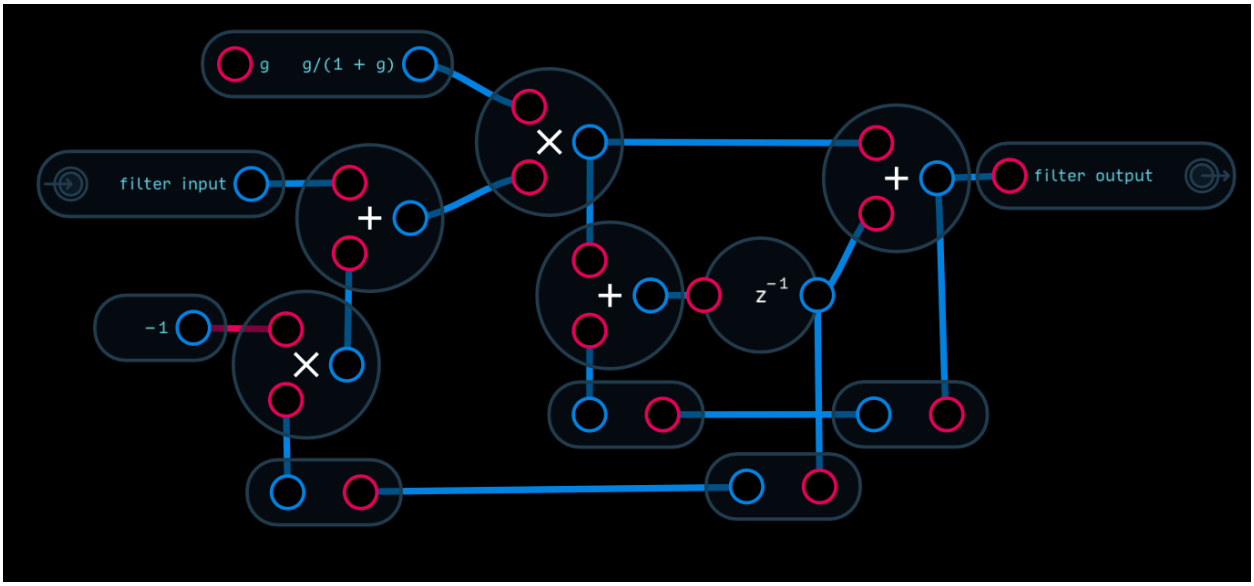
Recreating our integrator using the trapezoid instead of the rectangle gives us:



Inserting this into our filter gives us:



It turns out that using the trapezoid integrator has one disadvantage. Although it gives us a better approximation to the area under the curve, the trapezoid warps the frequency response of the filter. In order to compensate for this it is usual to *pre-warp* the coefficients. Without going into the math, if we replace our current version with:



Where g is defined as follows:

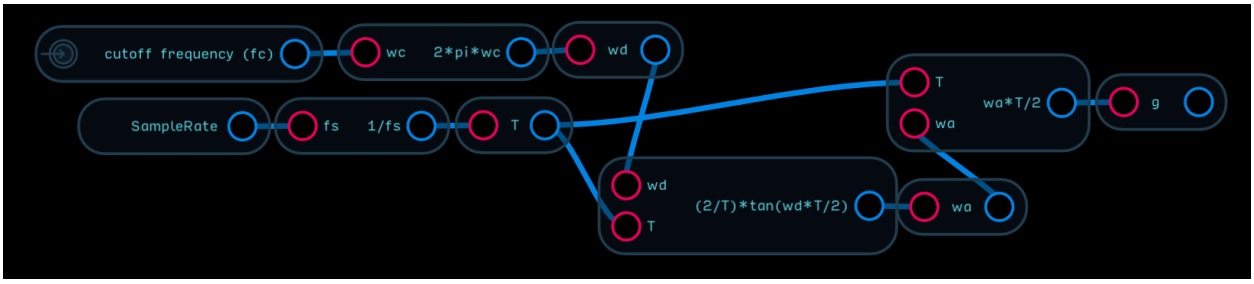
$\omega_d = 2\pi f_c$ wd = 2*pi*fc fc = cutoff frequency

$T = \frac{1}{f_s}$ T = 1/fs fc = sample frequency

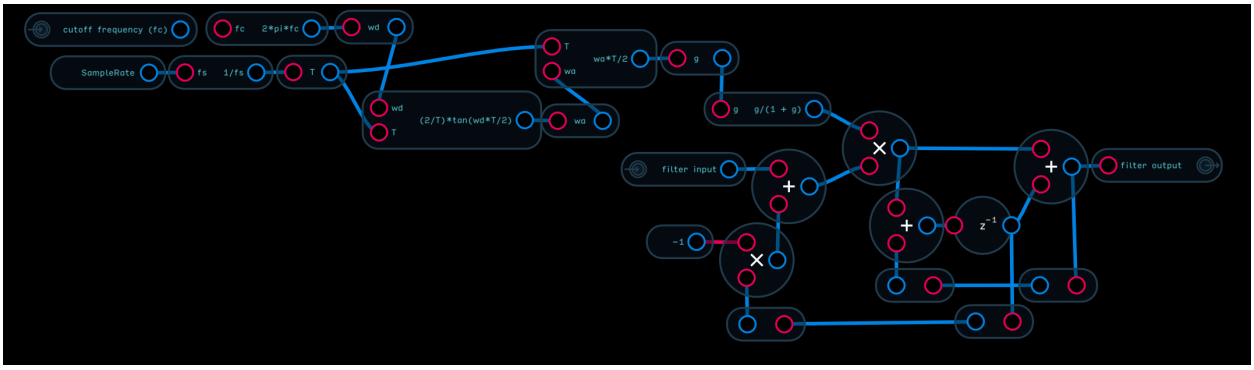
$\omega_a = \frac{2tan\left(\frac{\omega_d T}{2}\right)}{T}$ wa = (2/T)*tan(wd*T/2)

$g = \frac{\omega_a T}{2}$ g = wa*T/2

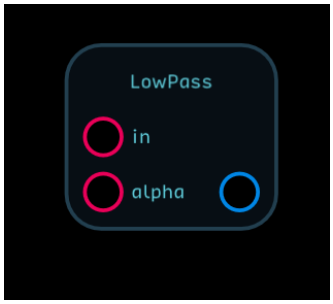
Or in Audulus:



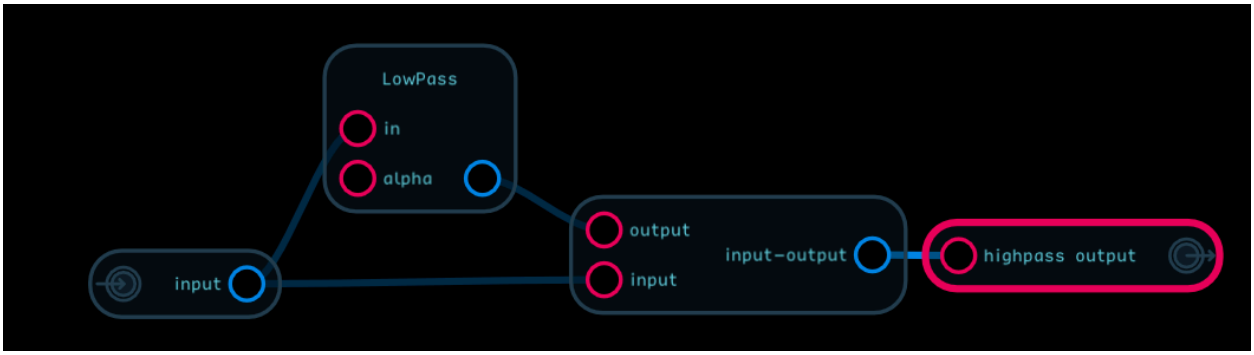
If we combine this with our filter model we get the complete module:



This is the completed digital model of the analog RC low pass filter we diagrammed at the beginning. It's also how the built-in LowPass filter node in Audulus works internally. Since the built-in node is somewhat more efficient than our module, it's generally preferable to use the built-in node.



What's alpha? Alpha is another way to represent the cutoff frequency of the filter. For use with Audulus alpha is calculated as: 1-(fc*2*pi)/(fs+fc*2*pi) where fs is the sample frequency and fc is the cutoff frequency. Note in some references you will find alpha defined as (fc*2*pi)/(fs+fc*2*pi) instead. The high pass node is similar to the low pass node. In fact, if you subtract the output of the low pass node from the input to the node, you will have a high pass filter.



Of course it's simpler to use the built-in HighPass node. As mentioned earlier, both of these nodes are first order filters and they have a cutoff slope of 6dB per octave which is not very steep. They are very efficient however and are a good choice for many filtering applications.

There are a few other filters provided with Audulus as basic nodes: The Filter node is a 2nd order low pass filter (12dB per octave) with resonance. Resonance is a peak in the filter response close to the cutoff frequency. There is also the BiQuad (bi-quadratic) node which allows you to create various types of 2nd order filters by inputting the appropriate coefficients. You will find many different filter modules built using these and other nodes in the Audulus library and online on the Audulus Forum.

There are many different types of filters used in synthesizers and they each have different characteristics and different applications. Because filters play a critical part in the "sound" of

most hardware synths, creating an accurate digital model of these filters is an important step in attempting to recreate the sound of the synth digitally. In addition to the built-in filter nodes, Audulus provides us with the tools needed to create digital models of many of these filters. This introductory tutorial covered the simplest analog filter and one digital model, but there is much more information readily available for those that are interested. These are a few references you may find useful. Many are available on the Audulus forum or can be found online by searching for the title using Google. The first two are particularly good for musical applications if you can slog through the math.

References

Zavalishin, Vadim. 2012. *The Art of VA Filter Design*, published electronically

Pirkle, Will. 2013. *App Note 4 Virtual Analog (VA) Filter Implementation and Comparisons*, published electronically

Lindquist, Claude. 1989. *Adaptive and Digital Signal Processing with Digital Signal Filtering Applications*, Chap. 1,5. Miami: Steward and Sons.

Lindquist, Claude. 1977. *Active Network Design with Signal Filtering Applications,* Chap. 2. Long Beach: Steward and Sons.

Pirkle, Will. 2012. *Designing Audio Effect Plug-Ins in C++,* Burlington: Focal Press.

Stilson, Tim., Smith, Julius O. *Analyzing the Moog VCF with Considerations for Digital Implementation*, Proceedings of the International Computer Music Conference, Computer Music Association, 1996, Hong Kong.