Understanding IIR (Infinite Impulse Response) Filters - An Intuitive Approach

by Dan Lavry, Lavry Engineering

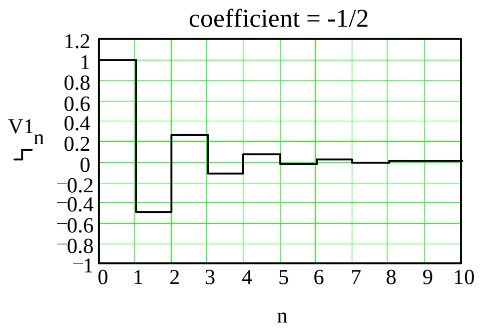
People less familiar with mathematics or network theory, often tend to assume that digital signal processing is beyond their understanding. For engineers, mathematics is a tool. Most problems are kept within some range of "common sense" understanding. This article present some of the more intuitive aspects of IIR filters.

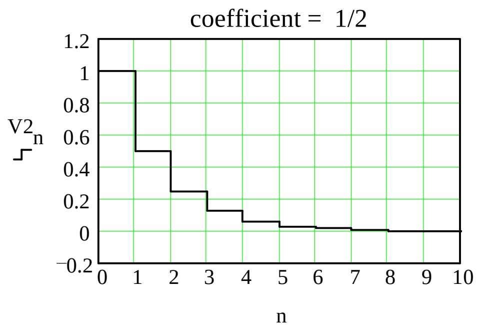
Introduction:

FIR filters offer great control over filter shaping and linear phase performance (waveform retention over the pass band). At times, the required number of taps (coefficients) exceeds available hardware compute power, or acceptable input to output time delay. Another type of digital filter, a close cousin to analog filters, is the IIR. Analog filters utilize analog components to filter continues signals and IIR's use numerical processing to filter sampled data signals, yet they are both based on "pole and zero" theory, yielding Butterworth, Chebyshef, Bessel and the other familiar filter response curves.
IIR filters offers a lot more "bang per tap" then FIR's. Computational efficiency and relatively short time delay make them desirable, especially when linear phase is of lesser importance.
FIR's are "forward" structures. Signal samples are sent forward from tap to tap. Many taps translate to long delays, and increase in the number of arithmetic operations. IIR's use feedback. The signal path is no longer a straight delay. Much of the filtering action depends on a feedback path. Portions of the output are feedback to be recomputed "over and over" by the same few coefficients. Each feedback tap contributes to shaping of many samples without the cost of additional arithmetic.

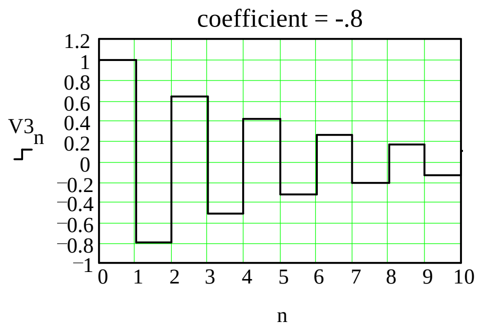
Digital feedback:

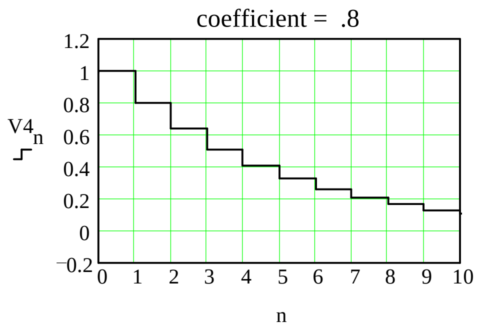
A digital oscillator may be constructed by means of simple arithmetic. let us set an initial output value to say 1. If we change the output by multiplying it by -1 at each sample clock time, we end up with a sequence of 1, -1,1,-1 and so on. Let us change the multiplying coefficient to -1/2. The output sequence becomes 1,-1/2,1/4,-1/8... yielding a wave form known as "exponentially decaying oscillations", or "damped ringing". If we choose a positive coefficient of 1/2, the sequence becomes
1,1/2,1/4,1/8... producing an exponential decay. Both ringing and decay remind us of behavior of some physical phenomena, such as encountered in tuned circuits and capacitor discharge circuits, thus providing some basis for emulating analog circuits by computational means.



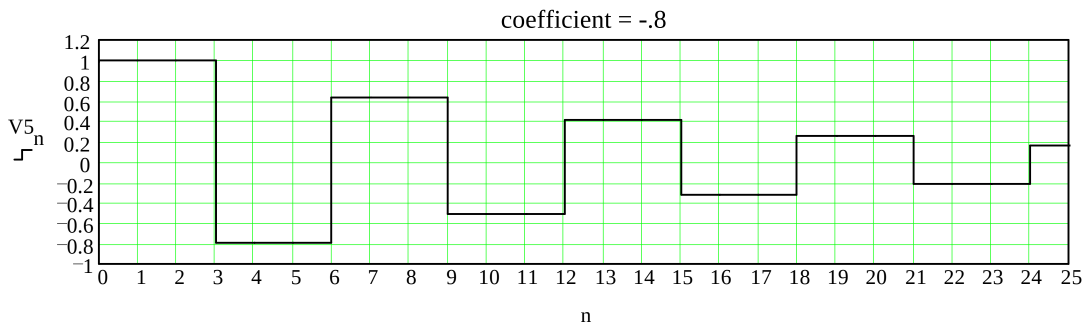


Speed of decay is is controlled by the coefficient values. Comparing the plots bellow (coefficient values -.8 and .8) to the previous plots (coefficient values -.5 and .5) shows slower decay.

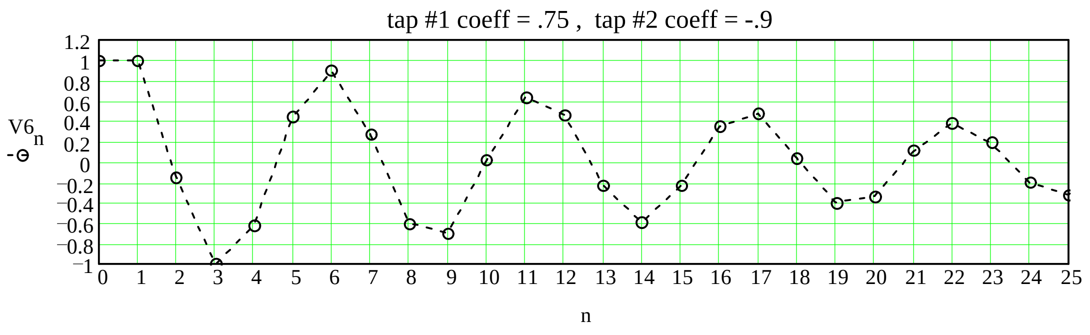




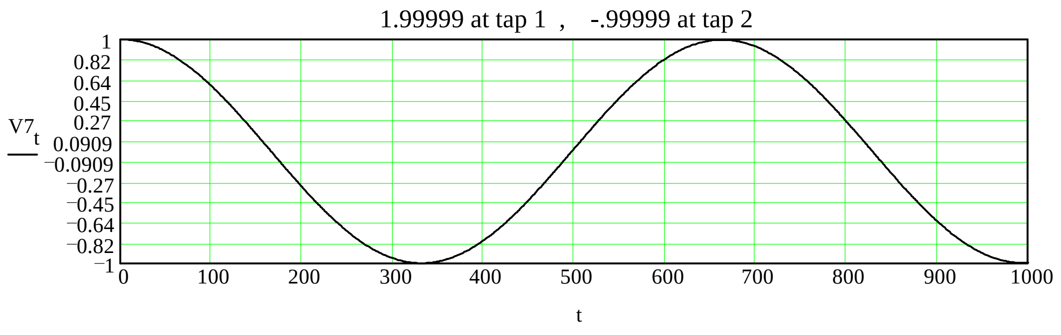
We can slow the ringing frequency down, say to a transition every third sample (instead of every sample) by inserting more time delay in the feedback. An output transition (change in value) takes three sample times before replacing the "old value" with a new value (feedback coefficient times "old value").



Greater control over the waveform is achieved by using more then one coefficient. Let us feed back two signals: a coefficient .75 with one tap delay and a coefficient value -.90 with two tap delay (tap delay is sample clock time). With two feedback coefficients we are no longer bound to "square waveforms". A quick glance at the graph below shows that we are not restricted to crossing the X axis at sample times. In fact, the tradeoffs between damping (decay) and frequency of oscillations is analogous to an analog circuit behavior, though there is a difference between sampled and continues waveforms.



The first 1000 samples using coefficients values of 1.99999 with one tap delay and -.9999 with two tap delay shows a significant slow down in both decay and frequency of oscillations. The first tap wants to almost double the output at each new sample time but the second tap serves to "counter the runaway condition" by subtracting an appropriate amount, yielding an "an amazingly clean" sine wave shape. Note the simplicity of calculations: each new sample clock replaces the "old output" by a sum of a scaled previous output and a scaled value of the "next to the last output".

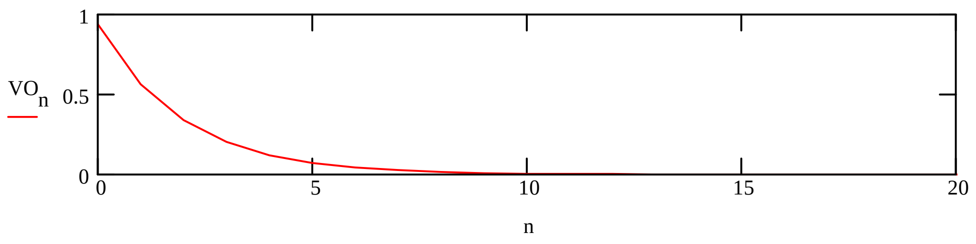


The example above is not unique. Constructing a smooth wave by means of a couple of delays and some simple arithmetic may seems somewhat mystical. The great mathematician Euler revealed very basic ties between concepts such as negative numbers, pi (circumference to diameter ratio), the values 1 and 0, the number e (base of natural logarithm), i (imaginary numbers), sine and cosine waves. Of course, most of us are too busy designing or using equipment and we tend to take such fundamentals for granted.

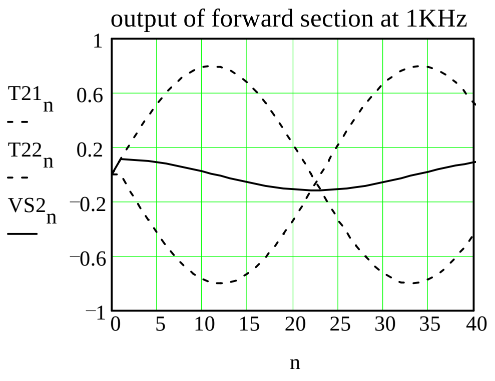
Basic IIR filters:

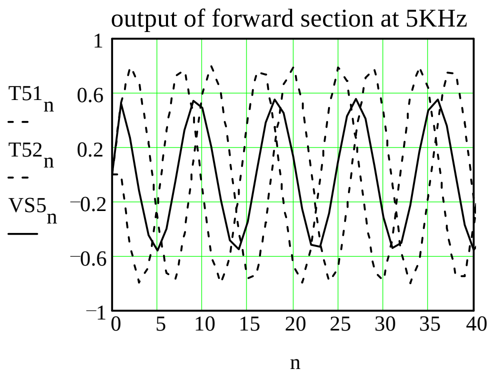
The previous examples were all based on setting an initial output value. We learned that forcing a single output value to 1 (impulse) will create a never ending cycle (converging on a point but never getting there), thus the name Infinite Impulse Response (IIR). Real IIR's consist of a forward path as well as a feedback path, both paths contributing to the output outgoing samples. A forward path is typically a short FIR like structure. IIR design relies on proper interaction and "balance" between forward and feedback sections. Each output sample is a sum of a new input and earlier input values and subtraction of previous output values, all multiplied by their respective coefficients. Computationally, feedback subtraction equals to addition with inverted coefficient values.

Let us examine a "first order" structure. We start with two forward coefficients .8 and -.8, and one feedback coefficient, .6 in value. Each new output equals: .8 X (new input) - .8 X (last input) + .6 X (last output). This filter will block DC. The contribution due to the forward coefficients is zero: .8 X DC - .8 X DC=0. The only "active coefficient" is the feedback, and it will cause any non zero output value to decay to zero (60% reduction for each sample clock time).
Often, during power on condition, an unpredictable signal values appear at the three taps. Let us assume arbitrarily that the "last input tap" was loaded to a .5 values and the feedback tap value was at .9. Feeding DC input of 1 will make a next output value of 1 X .8 - .5 X .8 + .6 x .9 = .94. During the following clock both forward taps are at 1 (the DC input value of 1 propagates to the "last input tap"). The new output is 1 X .8 - 1 X .8 + .6 X .94=.564 or simply .6 X .94 = .564. All the following incoming input samples will be canceled by the forward coefficients and the feedback tap plays the only active role. The outcome is shown bellow:

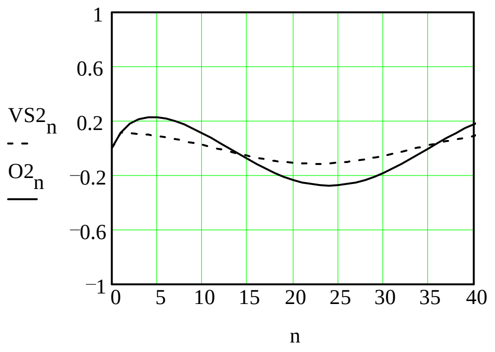


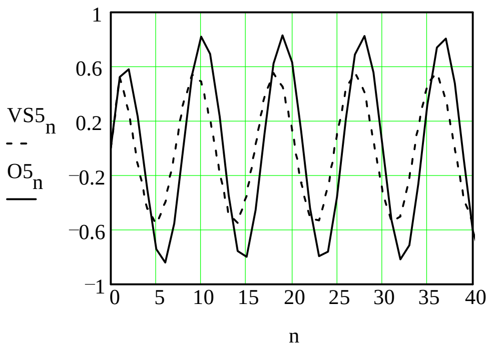
What we see is the initialization of an IIR. These filters take a number of clock samples to "discharge" the feedback to an "acceptable" level. The settling can take place at the presence of an input signal. It is not uncommon however to get "wrong" results many seconds after power is first applied, but once settled, our filter will work properly.
IIR hardware computes all taps at each sample time, keeping track of relative time slots. Viewing the filter response can be simplified by "breaking" the filter into two sections (connected in series). We can compute and sum the forward taps into an intermediary output, we then feed it to a the input of a separate feedback section. This "trick" does not emulate a "real" IIR (it adds to the overall filter delay), yet it allows us simplify wave form and frequency response analysis.
The plot bellow shows an application of 1KHz and 5KHz inputs (at 44.1KHz sampling) to the forward section. The dotted lines are the individual tap signals (.8 and -.8 coefficient) and the solid lines shows the sums of the forward section. Note the impact of the frequency on the sum of the signals.



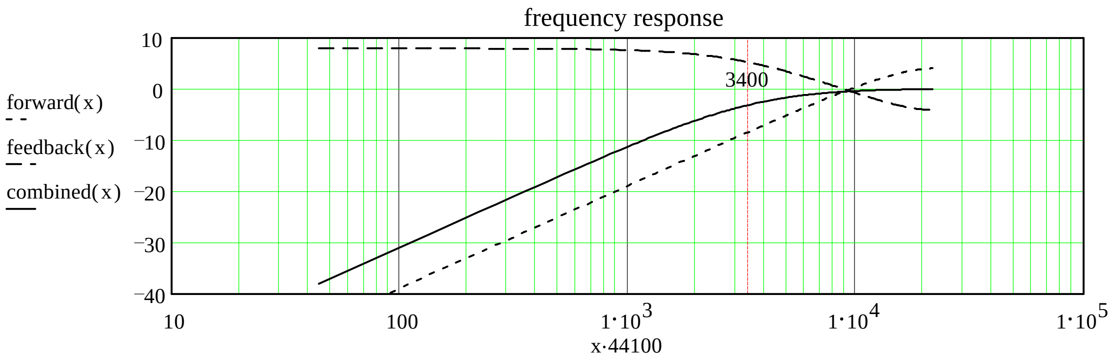


We now pass the intermediary outputs through a feedback tap (value .6). Each new output sample is a sum of the forward section output and .6 of the previous output value. Dotted lines show outcome from the forward section and solid lines show final output. The first samples show distortion due to settling.



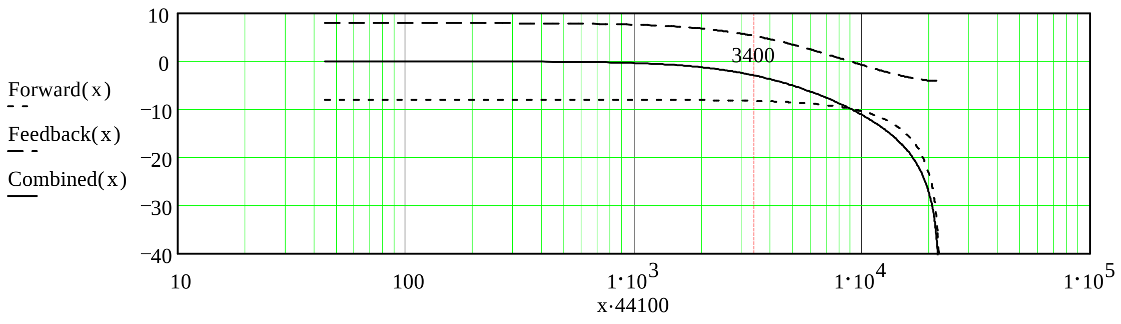


The coefficients chosen in our example are for a single pole 3.4KHz high pass filter. The 1KHz and 5Khz output plots show .25 and .8 attenuation (12dB and 2dB). Repeating the process for many frequencies will show virtually the same response we can get from an analog circuit equivalent. The frequency vs. amplitude plots below show individual forward response (dotted line), feedback (dash line) and the combined sum (solid line). The dotted and dash line addition yield the solid line (log scale):



The plots can be understood intuitively. The forward section coefficients (.8 and -.8) tend to cancel each other when the input and "last input" signals are of similar value. Such is the case at very low frequencies where change in amplitude over a sample time is small. Increasing the frequency creates larger signal difference. The reduced the cancellation increases the intermediate output (dotted line). The feedback coefficient keeps sending back some of the previous signal, building up to higher levels when the output moves slowly. At higher frequencies, the out of phase feedback subtracts from the output (dash line).

Equipped with some intuitive understanding, let us convert the high pass filter into a low pass. At high frequencies (when input and "last input" are out of phase) we want the forward section to cause signal cancellation. Equal forward coefficients (magnitude and sign) will do just that, while DC input values will double. A good start for a low pass, indeed. Also when passing DC, all taps are at a steady value. The feedback and the forward section steady state levels should add to 1, so: twice forward coefficient plus feedback coefficient equal 1.
We could try various combinations such as .1+.1+.8 or .2+.2+.6, all yielding different filter cutoff frequencies. Let us try to use the previous .6 feedback coefficient making the forward coefficient .2 (.2+.2+.6=1). The plots bellow show the response:

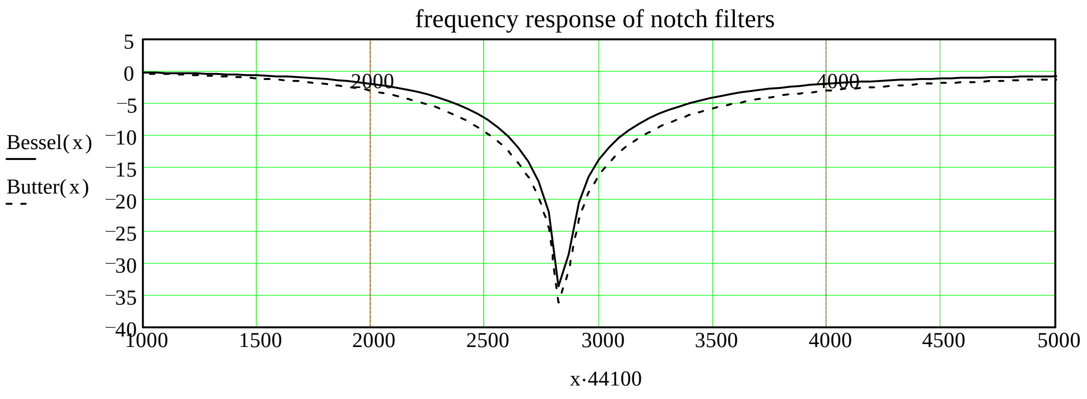


Indeed, retaining the feedback coefficient at .6 left the 3dB attenuation point at 3.4KHz. Notice that when retaining the the feedback, our new low pass forward coefficient (.2) and the old high pass coefficient (.8) add to 1. Similar uncomplicated relationships hold for higher order filters as well. The conversion from say band pass to band reject is not a a very difficult matter.

Let us now investigate and compare two notch filters (with 3 forward and 2 feedback coefficients each). Filter #1 is a Butterworth type and filter #2 is a Bessel filter, both aimed at rejecting frequencies between 2KHz and 4KHz (at 44.1KHz sampling).
The forward coefficients of filter #1 are .875, -1.60 and .875. The feedback coefficients are 1.60 and -.750
The forward coefficients of filter #2 are .900, -1.66 and .900. The feedback coefficients are 1.66 and -.800

With DC inputs:
Forward contribution for #1 is .875X2-1.6=.15, and the feedback contribution is 1.6-.75=.85 (Sum together .15+.85=1)
Forward contribution for #2 is .90X2-1.66=.14, and the feedback contribution is 1.66-.8=.86 (Sum together .14+.86=1)
At maximum input frequency (Nyquist) each tap delay causes total phase shift. the second taps get inverted signals:
Forward contribution for #1 is .875X2+1.6=3.35, and the feedback contribution is -1.6-.75=-2.35. (Together 3.35-2.35=1)
Forward contribution for #2 is .90X2+1.66=3.46, and the feedback contribution is -1.66-.8=-2.46. (Together 3.46-2.46=1)

We did not show that the filters reject frequencies in the specified band, nor is it an exercise to the casual observer. The reader is encouraged to believe that such is the case. There are a couple of points to observe, however: a "basic filter requirement" may be satisfied by use of many sets of coefficients that will seemingly satisfy a given filter requirement, a closer look shows performance differences. Our band reject implementations, show greater notch rejection for the Butterworth filter (see plot bellow). A proper selection of coefficients depends on the specific application, with the same tradeoffs encountered by analog designers (pass band flatness, attenuation, transition band, phase characteristics and more). Figuring IIR filter coefficients is commonly done by designing the analog equivalent filter, and then performing a transformation into processor oriented language of computations. The process is less intuitive and usually involves a fair amount of mathematics and network theory.



Closing remarks:

An IIR is an arithmetic engine free from familiar analog problems (such as component tolerance), yet each design needs careful evaluation. Practical considerations force theoretical input and coefficient values, to be rounded off to a certain hardware dependent accuracy. Processor and arithmetic word lengths (bits) may prove insufficient for a straightforward implementation of some theoretical models, producing problems such as instability (the output takes off) or limit cycles (unwanted signal patterns cycling around). Practice requires a proper computational structure. Much like analog, most higher order IIR filters are "broken" into a series of "first and second order" sections (typically up to three forward and two feedback taps per section). IIR's can be more sensitive to computational round off errors then FIR's, but proper combination of compute engine, filter structure and some embellishments often yield impressive results.
With some intuitive sense regarding its "inner working", it is easy to understand that high computational efficiency and short delays often make the IIR a desirable alternative when FIR's fall short (become too long), in digital feedback systems (where long filter delay cause problems) and many other applications.