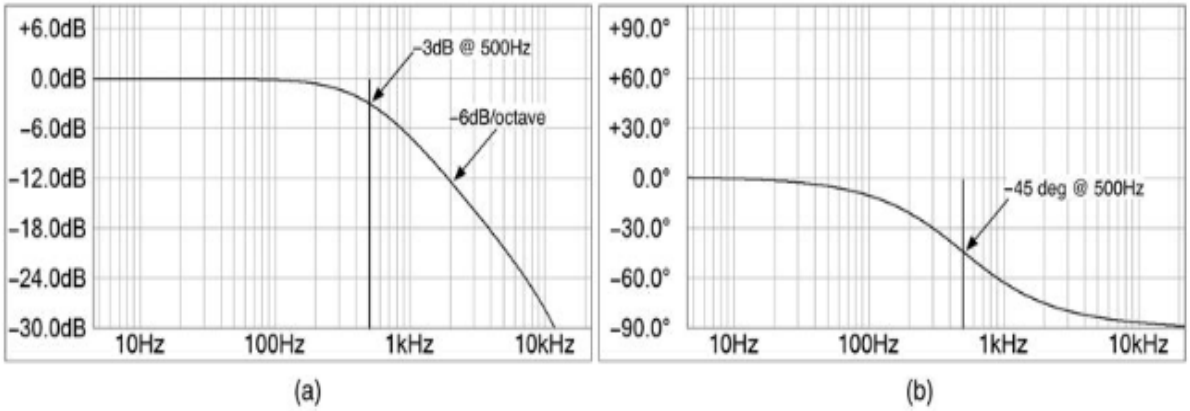
DSP in C++: 9 - How DSP Filters Work

Thursday, July 14, 2022 9:27 PM

9.1 Frequency and Phase Response Plots

The *frequency response plot* shows how a filter modifies the frequency content of a signal. The frequencies are plotted on the x-axis and the relative output amplitudes of the frequencies on the y-axis.
The *phase response plots* show the phase shift through the filter. The only difference in the two plots is the amount of phase shift that the various frequencies encounter when moving through the filter.



9.2 Frequency and Phase Adjustments from Filtering

Fourier showed that a continuous waveform could be decomposed into a set of sinusoids with different frequencies and amplitudes (which is called *Fourier decomposition*). This means that any complex (meaning made up of multiple sinusoids) signals can be expressed as a sum of pure sinusoidal waveforms which are added together. Therefore we can also recreate a complex signal using its elemental sinusoidal signals called *Fourier re-synthesis*.

"

[The figure below] shows a filter in action. The input is decomposed into four sinusoids, a fundamental, and three harmonics. The peak-amplitudes of the components are shown as dark bars to the left of the *y*-axes. For simplicity, we have chosen the input signal as a combination of four sinusoids that have the same amplitude, and that start at 0.0 at time $t = 0$, so that they have identical starting

phases of zero degrees. We can observe the following:

- The input waveform is smoothed out and is less “bumpy.”
- The lowest frequency component (called the *fundamental frequency*) comes out of the filter with
  the same amplitude and phase shift as it had when it entered the filter.
- The three harmonics have decreasing amplitudes and more phase shift as they go higher in
  frequency.

" (2019, pg. 179)

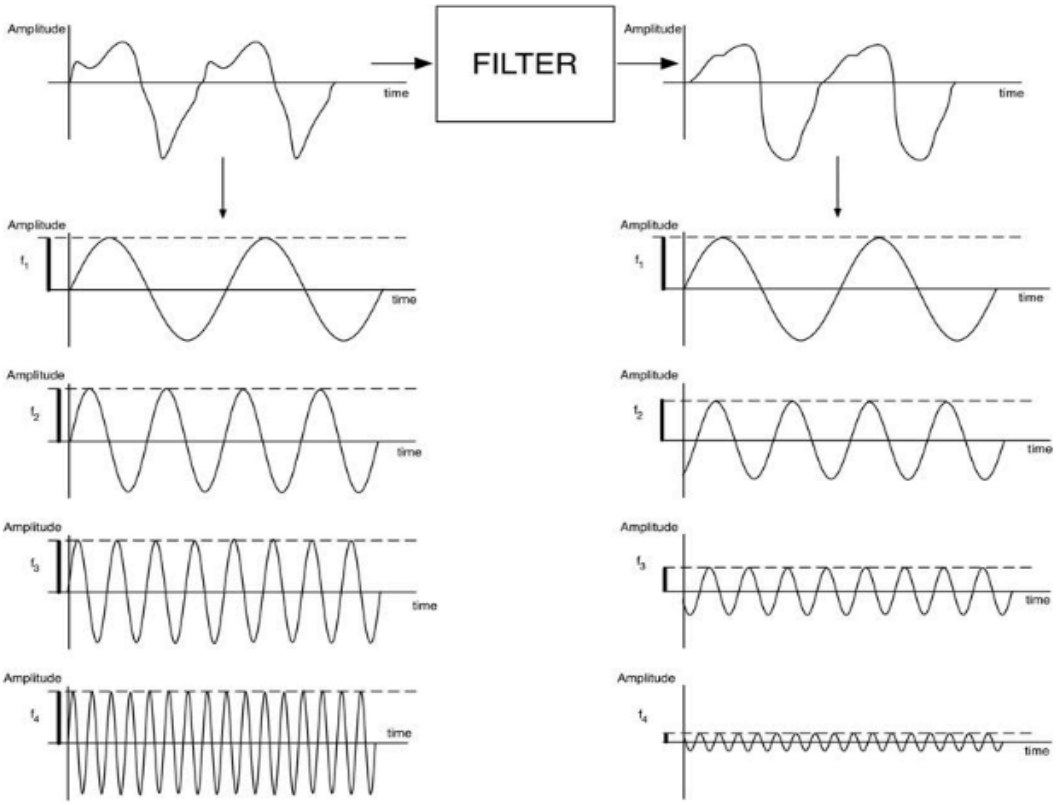
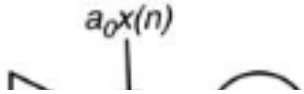
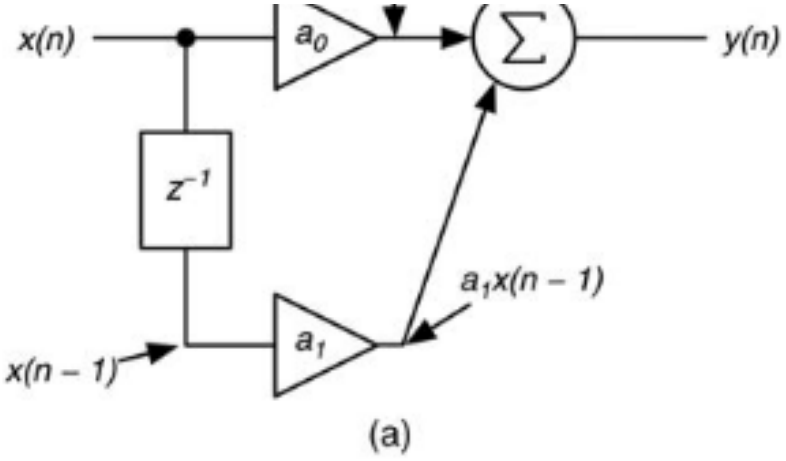


Figure 9.2: An audio waveform is filtered into a smoothed output; the input and output are decomposed into their Fourier components to show the relative amplitudes and phase shifts that each frequency encounters while being processed through the filter.

9.3 First Order Feed-Forward Filter

Let’s look at how an input signal x(n) would flow through the filter to become the output y(n). The figure below shows a filter along with the values of the samples at the various nodes of the circuit. The input sample is x(n) and the output of the storage register marked z−1 is x(n − 1), which was the previous input to the filter, one sample period prior to our observation.





(a)

The difference equation for this filter can be expressed as follows:
$y(n) = a_0 * x(n) + a_1 * x(n-1)$

You can tell why the structure is called feed-forward: the input branches feed-forward into the summer. The signal flows from input to output. There is no feedback from the output back to the input.

In order to figure out what this does, you apply the five basic digital test signals to the filter, then manually push the values through and see what comes out. For each audio sample that enters the structure, there are two phases to the operation:

- **Process phase**: the sample is read in and the output is formed using the difference equation and the previous sample in the delay register; the input is processed through to the output.
- **State update phase**: the delay element is overwritten with the input value—the older sample stored in the single z−1 register is effectively lost.

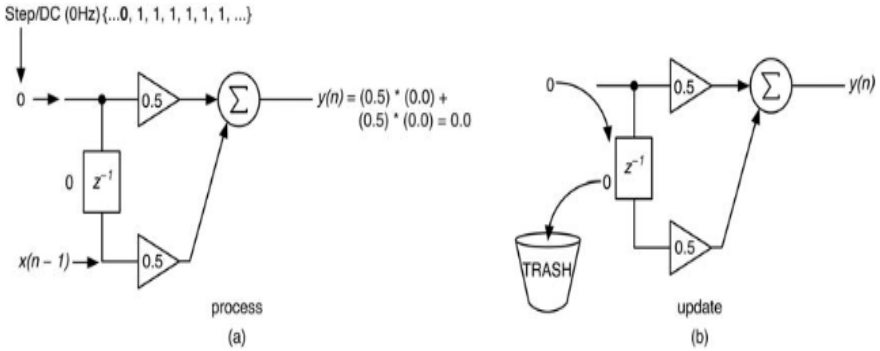
"The value in a filter's z^{-1} register is sometimes called its "*state*," and the second phase is sometimes called the *state update* phase or more simply *updating the state* of the filter. The z^{-1} register itself is sometimes called the *state register* and we refer to it this way throughout the text. It is during this second phase of operation that we prepare the z^{-1} registers for the *next* processing iteration on the next sample period. It is important that we always update the filter's state *after* the output *y(n)* has been calculated. This will become a rule for our filtering objects later on. " (2019, pg. 182)

The five waveforms we want to test are:

- DC (0 Hz)

  "Figure 9.5a shows the first iteration of the structure where we

process the input x(n) into an output y(n) and then update the filter's state by shifting the input value into the delay register, as shown in Figure 9.5b. Notice that we lose the value in the state register, symbolically sending it to the trash can, and overwriting it with the current input value. In complex filter structures with many state registers, we need to take care to ensure that the state registers are updated in the proper sequence so that only the oldest state information is ever lost or thrown away."



Continuing this process leads to the following:
Input sequence: {. . . 0.0, 1.0, 1.0, 1.0, 1.0 . . .}
Output sequence: {. . . 0.0, 0.5, 1.0, 1.0, 1.0 . . .}

The output amplitude eventually settles out to a constant 1.0 or unity gain condition, so at DC or 0 Hz, the output equals the input. However, there is a one-sample delay in the response causing the leading edge of the step input to be smeared out by one sample interval. This time smearing is a normal consequence of the filtering and gives us our first rule for feed- forward filters.

★ In a feed-forward filter, the amount of time smearing is equal to the maximum delayed path through the feed-forward branches.

- Nyquist

This time we will keep track of our tabulations in the following table.

$x(n)$	$x(n-1)$	$y(n) = 0.5x(n) + 0.5x(n-1)$
1	0	0.5
-1	1	0
1	-1	0
-1	1	0

1	-1	0
-1	1	0
1	-1	0
-1	1	0

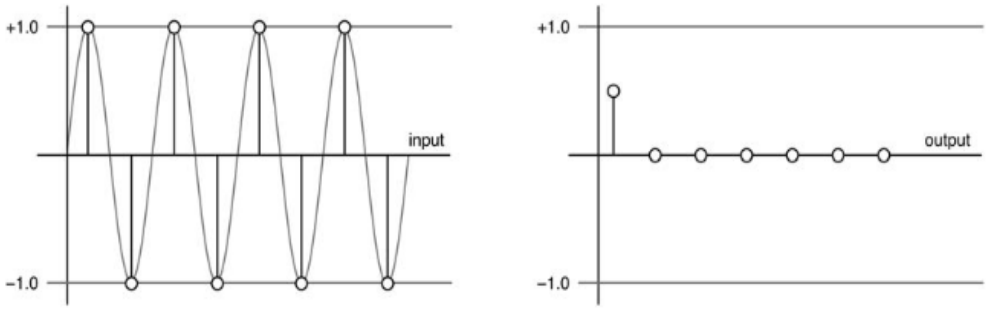


Figure 9.10: The input and output sequences of the filter at Nyquist reveals that the output completely disappears after the first sample period.

? Why did the amplitude drop all the way to zero at Nyquist?
The answer is one of the keys to understanding digital filter theory: the one-sample delay introduced exactly 180 degrees of phase shift at the Nyquist frequency and caused it to cancel out when re-combined with the input branch through a_0. This leads us to our next rule.

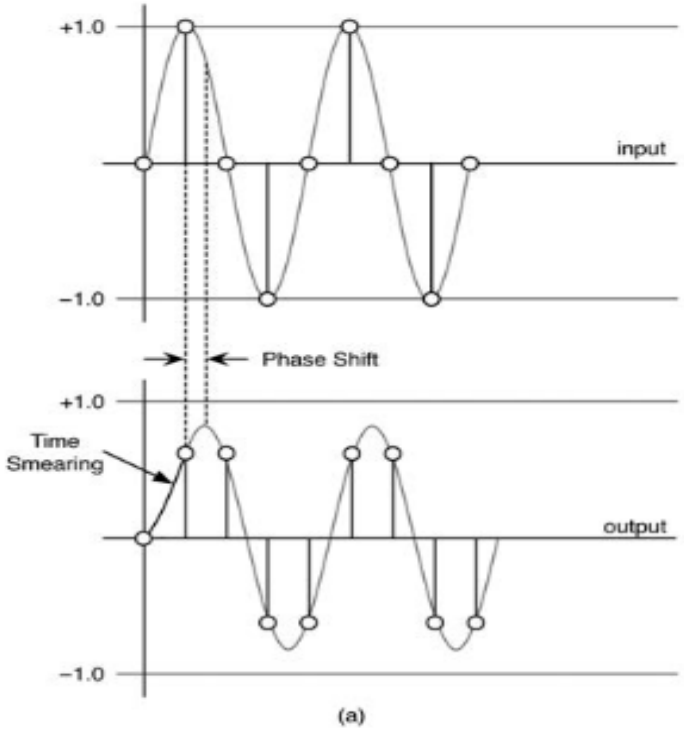
★ Delay elements create phase shifts in the signal. The amount of phase shift depends on the amount of delay as well as the frequency of the input signal.

- 1⁄2 Nyquist

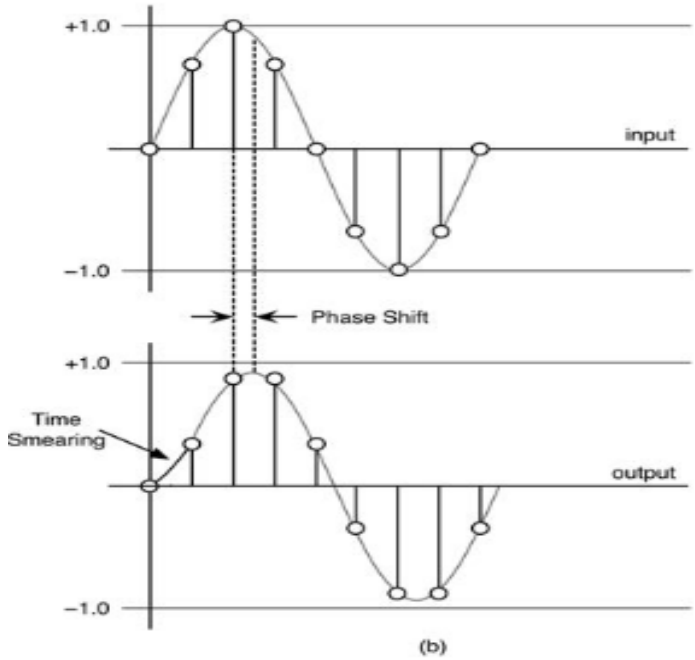
$x(n)$	$x(n-1)$	$y(n) = 0.5x(n) + 0.5x(n-1)$
0	0	0
1	0	0.5
0	1	0.5
-1	0	-0.5
0	-1	-0.5
1	0	0.5
0	1	0.5
-1	0	-0.5

0	-1	-0.5

We observe that the output sequence is periodic, but with reduced amplitude and with a phase offset.



- 1⁄4 Nyquist



Here we can estimate that the output amplitude is about 90% of the input size, and that the one sample of time smearing results in a phase shift of 1/16 of a cycle, or 22.5 degrees.

phase shift of 1/16 of a cycle, or 22.5 degrees.

Having analyzed all of the signals that impact the frequency domain of the filter, we can now combine the frequency and amplitude observations for a frequency response plot and the frequency and phase angles for a phase response plot.

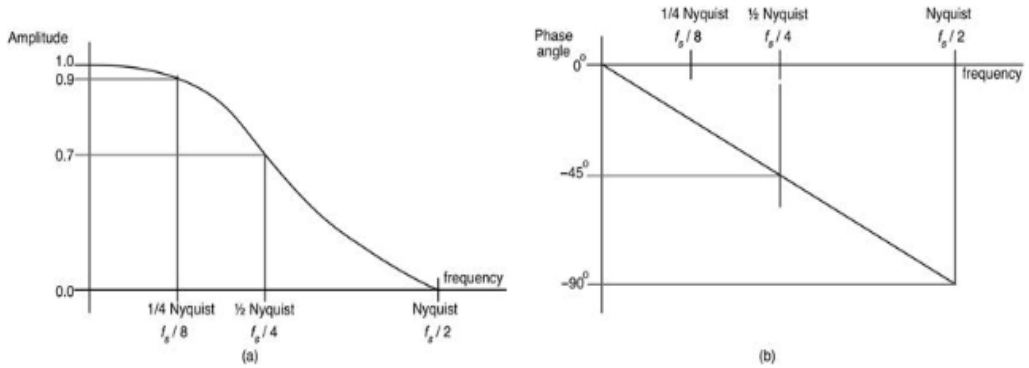


Figure 9.12: The frequency response (a) and the phase response (b) of our simple 1st order feed-forward filter.

We observe that this digital filter is a low-pass variety with a typical LPF frequency response. However the phase response is quite interesting: it is linear instead of nonlinear like the example at the beginning of the chapter. In fact, this simple filter is a *linear phase filter*. A feed-forward filter will be a linear phase filter if its coefficients are symmetrical about their center. In this case, (0.5, 0.5) is symmetrical.

These are interesting for two reasons:

1. this kind of response cannot be obtained from an analog filter; the closest we can get is an analog filter whose phase response is linear through the pass band, but then goes nonlinear in the stop band.
2. this kind of filter has useful applications where phase linearity across the spectrum is important, such as filters for loudspeaker crossovers.

- ## Impulse

  Applying the Impulse signal to the filter we will find the filters *Impulse Response*. The impulse response defines the filter in the time domain, like the frequency response defines it in the frequency domain. The basic idea is that if you know how the filter reacts to a single impulse you can predict how it will act to a series of impulses of varying amplitudes.

$x(n)$	$x(n-1)$	$y(n) = 0.5x(n) + 0.5x(n-1)$
0	0	0
1	0	0.5
0	1	0.5
0	0	0
0	0	0
0	0	0

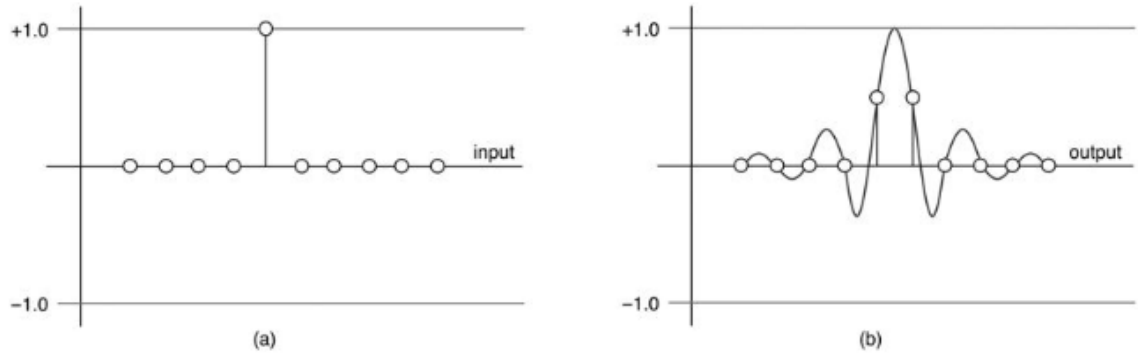


Figure 9.13: (a) Input and (b) output sequences for passing an impulse through the filter.

It is actually two points on a sin(x)/(x)-like curve. You might also notice that the two non-zero values in Table 9.3 are identical to our pair of filter coefficients: a0 and a1 = {0.5, 0.5}. This leads to another rule.

★ For a pure feed-forward filter, the impulse response is finite and consists a set of values that are identical to the filter coefficients.

? What makes this filter a low-pass filter?
It is a combination of the coefficients and the filter topology (1st order feed-forward).

There are three basic topologies:

- Feed-forward (FF)
- Feedback (FB)
- a combination of FF/FB.

★ So once the topology has been chosen, it's really the coefficients that determine what the filter will do.

We can now make some important generalizations of our filters and think about what a plugin that implements filters would need to do:

- The topology of the filter determines its difference equation.
- The coefficients of a feed-forward filter determine its filter frequency and phase responses and therefore its type (e.g. low-pass filter or LPF) and its sonic qualities.
- A filtering plugin must implement the difference equation to process input samples into output samples.
- As the user adjusts the plugin controls (filter type, cutoff frequency, etc.) we must recalculate the filter coefficients accordingly so that the filter is adjusted properly.

9.4 1st Order Feedback Filter

The 1st order feedback filter is shown in Figure 9.15 and consists of two coefficients: a_0 in the forward path and b_0 in the feedback path. There is only one state register, but this time it is in the feedback path, and it stores previous filter outputs instead of previous filter input x(n-1). You can also see that there is no b_0 coefficient in this structure—and the fact is that we will never have a b_0 coefficient. The difference equation for this filter is:

$$y(n) = a_0 x(n) - b_1 y(n-1)$$

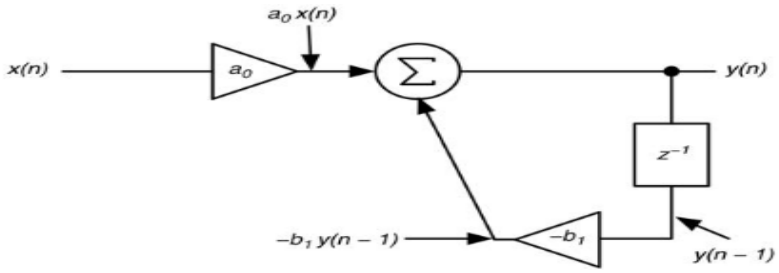


Figure 9.15: A 1st order feedback structure.

We can apply the same concept of pushing analytical sequences through this structure and examining the output to determine the frequency and phase responses. But a little thought will show that this is going to become exceedingly tedious. This in itself is good motivation for us to find a better way to analyze and design these filters, even if it requires some heavier mathematical lifting on our parts. In the next chapter we will be covering the basic DSP theoretical framework which will allow us to start analyzing these effects mathematically.

Examine the figure below which shows pairs of plots for frequency and impulse responses of this 1st order feedback structure. In each case, the a_0 coefficient is held constant at 0.5, and the b_1 coefficient is varied from 0.5 to 0.9 to 1.0. Notice what happens to the impulse response as the value for the b_1 coefficient increases: you can see that it becomes a damped oscillation, which we never observed in the feed-forward filter. This is called *ringing* and has an audible effect that is referred to as ringing, or sometimes *pinging*.

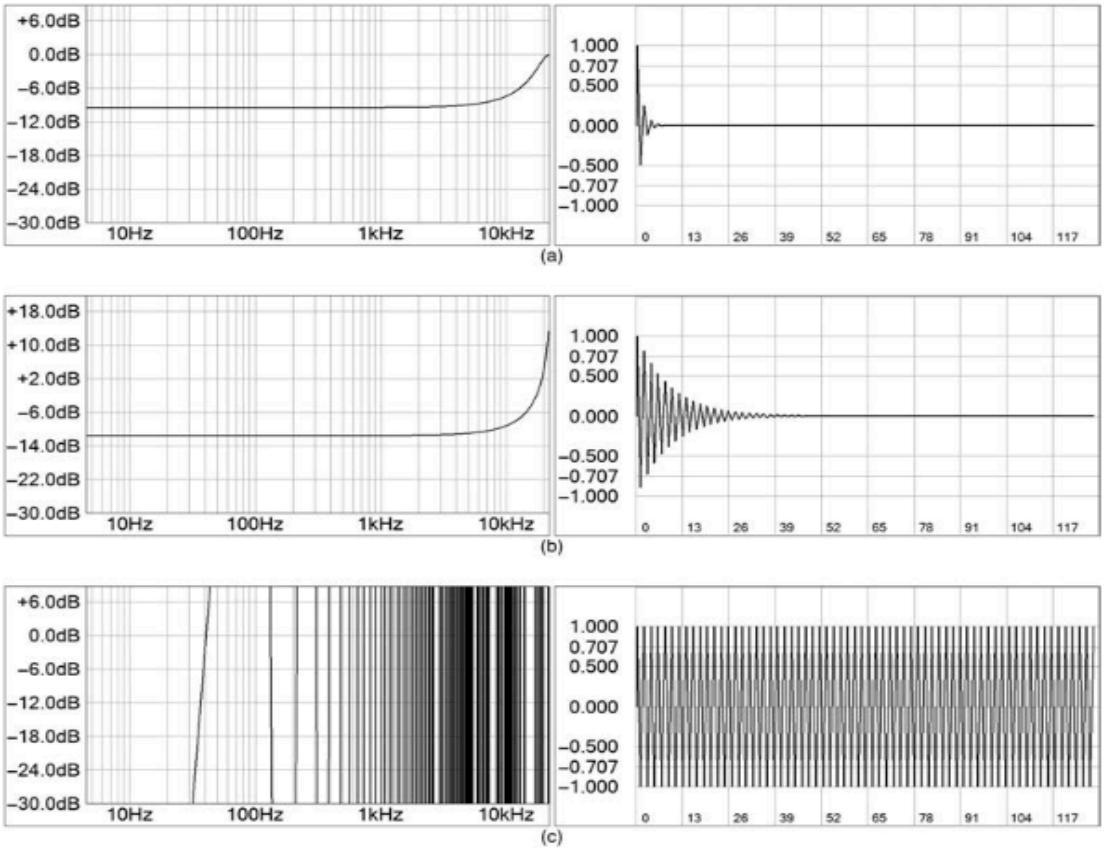


Figure 9.17: The 1st order feedback filter's frequency response (left column) and impulse response (right column) for various coefficient values of (a) a_0 = 0.5, b_1 = 0.5; (b) a_0 = 0.5, b_1 = 0.9; and (c) a_0 = 0.5, b_1 = 1.0 (notice that the impulse response here rings forever, and the frequency response is garbage).

9.5 Final Observations

Feed-Forward Filters

- operate by making some frequencies go to zero; in the case of a_0 = 1.0 and a_1 = 1.0, the Nyquist frequency went to zero; this is called a *zero of transmission* or a *zero frequency* or just a *zero*
- step (DC) and impulse responses show smearing (amount of smearing is exactly equal to the total amount of delay in the feed-forward

branches)
- don't blow up
- are called *finite impulse response* (*FIR*) filters because their impulse responses, though they may be smeared, are always finite in length

Feedback Filters

- operate by making some frequencies go to infinity; in the case of a_0 = 0.5 and b_1 = 1.0, the Nyquist frequency went to infinity but in the case of a_0 = 0.5 and b_1 = −1.0, the DC or 0 Hz went to infinity; this is called a pole of transmission or a pole frequency or just a pole
- step and impulse responses show overshoot and ringing or smearing depending on the coefficients (amount of ringing or smearing is proportional to the amount of feedback)
- can blow up (or go unstable) under some conditions
- are called infinite impulse response (IIR) filters because their impulse responses can become infinitely long