Music 171: Introduction to Delay and Filters

Tamara Smyth, trsmyth@ucsd.edu
Department of Music,
University of California, San Diego (UCSD)

November 21, 2019

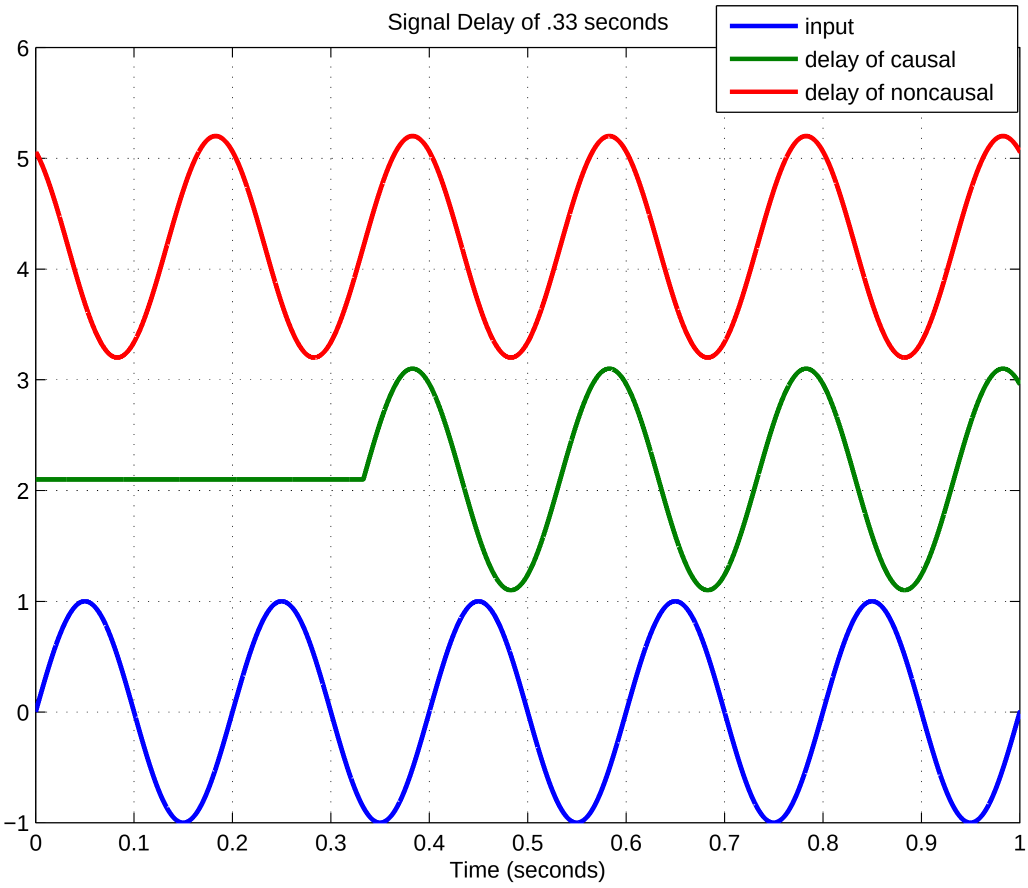
Digital Filters

- **Filter**: any medium through which a signal passes.
- Typically, a filter modifies the signal in some way:
  - audio speakers / headphones
  - rooms / acoustic spaces
  - musical instruments
- A *digital* filter is a formula for going from one digital signal (input $x(n)$) to another (output $y(n)$):

Figure 1: A black box filter.

Inside the Black Box—Pure Delay

- Digital filters typically involve signal *delay*.
- Delaying an audio signal is to
  - move it (earlier/later) in time;
  - change the **phase** of signal (the value at time=0).

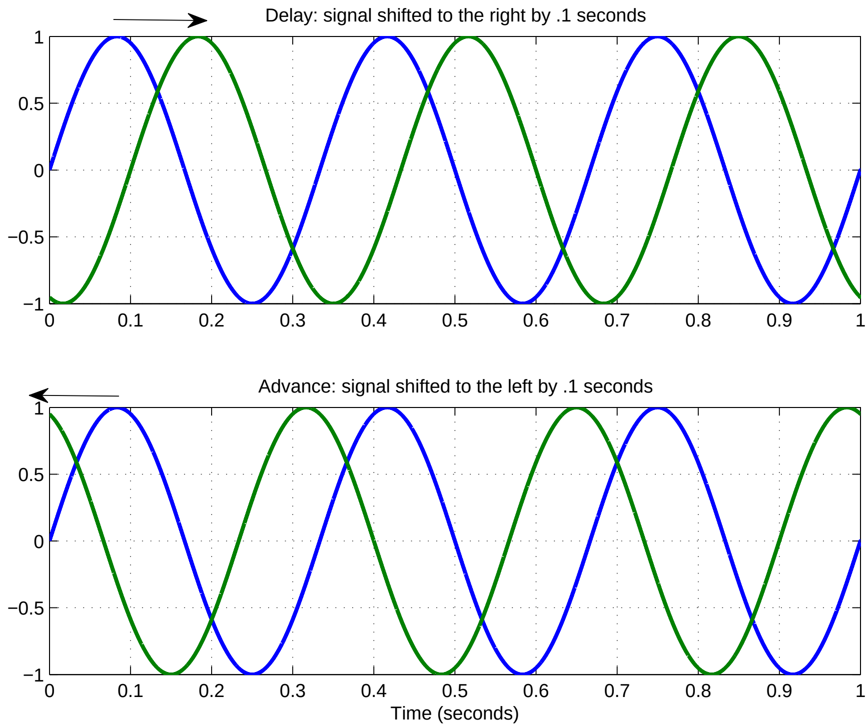


Time shifting a signal

- When a signal can be expressed in the form

$$y(n) = x(n - M),$$

$y(n)$ is a *delayed* (time-shifted) version of $x(n)$.



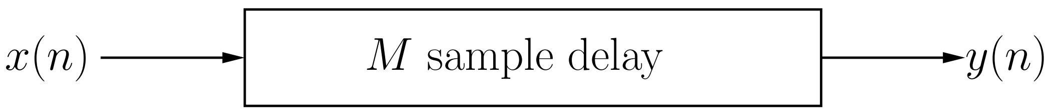
- $y(n) = x(n - M)$: $x(n)$ is **delayed** M samples:
  - shift is to the **right** on the time axis.
- $y(n) = x(n + M)$: $x(n)$ is **advanced** M samples:
  - shift is to the **left** on the time axis.

The Delay Line

- The delay line is a functional unit that models *acoustic propagation delay*.
- It is a fundamental building block of *delay effects processors*.
- The function of a delay line is introduce a time delay of M samples or

$$\tau = M/f_s \text{ seconds}$$

between its input and output.

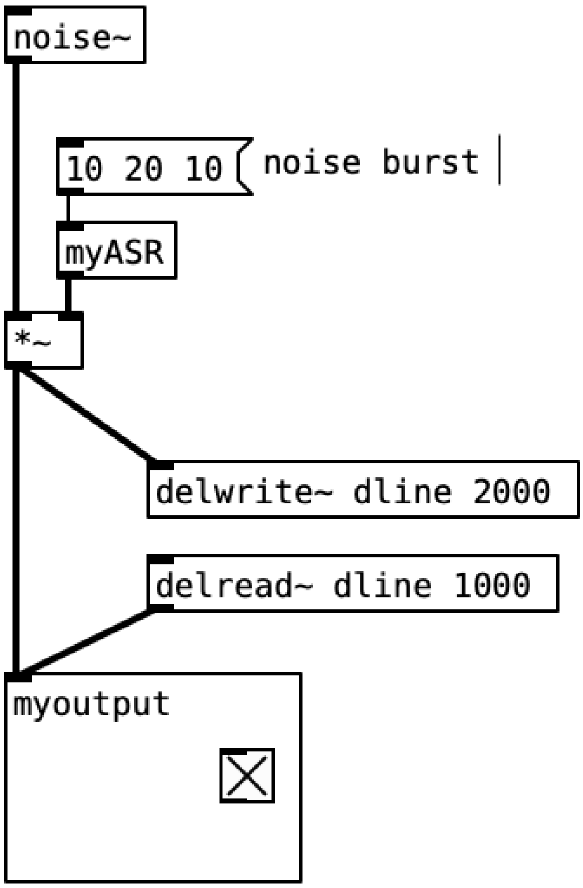


$$y(n) = x(n - M), \quad n = 0, 1, 2, \ldots$$

Time shift and addition

- Other than possible silence, there is no audible effect of a *pure* delay.

$$\boxed{x(n) \text{ and } x(n-M) \text{ sound the same}}$$



- Change arises, however, when a signal $x(n)$ is added to a delayed version of itself $x(n-M)$:

$$y(n) = x(n) + x(n-M)$$

A Running Averager

- Consider a simple case where $M = 1$.

$$y(n) = x(n) + x(n-1).$$

- (Dividing by 2), this filter **averages** adjacent samples.
  - that is, output $y(n)$ is a *running average* of input $x(n)$ **with a gain of 2**.

This filter takes the average of two adjacent samples.

Intuitive Analyis at Low Frequencies

- Consider input at 0 Hz (lowest possible frequency):

$$x_1(n) = [A, A, A, ...].$$

(at 0 Hz there is no change from sample to sample).

- The output is

$$\begin{aligned} y(n) &= x_1(n) + x_1(n-1) \\ &= \quad [A, A, A, ...] \\ &\quad + [0, A, A, A, ...] \\ &= \quad [A, 2A, 2A, 2A, ...] \\ &\approx 2x_1(n) \quad \text{(except 1st sample).} \end{aligned}$$

The filter has a gain of 2 at the lowest frequency.

Intuitive Analyis at High Frequencies

- Consider input at $\frac{f_s}{2}$ Hz (highest possible frequency):

$$x_2(n) = [A, -A, A, -A, ...].$$

(maximum change from sample to sample).

- The output of the filter is

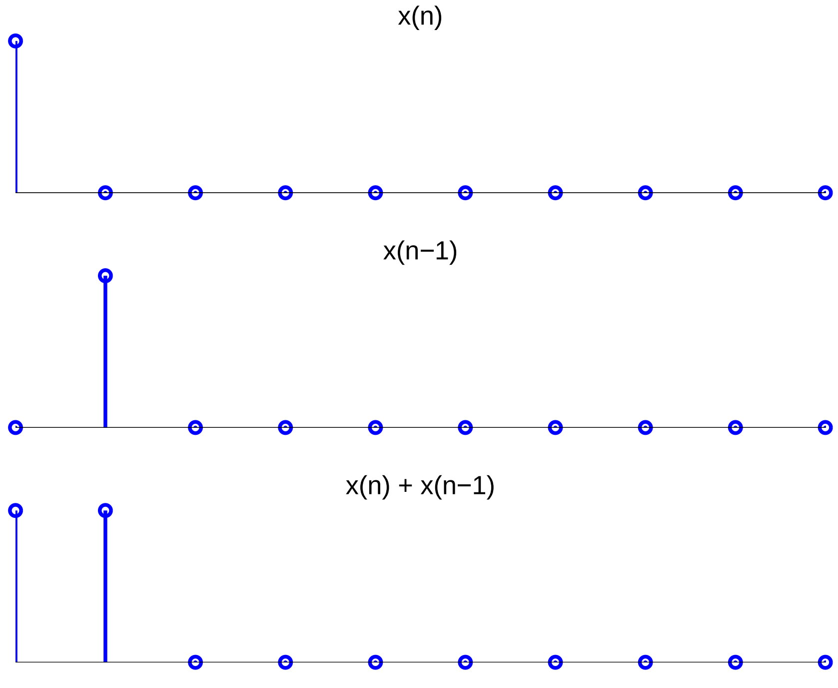
$$\begin{aligned} y(n) &= x_2(n) + x_2(n-1) \\ &= \quad [A, -A, A, -A, ...] \\ &\quad + [0, A, -A, A, ...] \\ &= \quad [A, 0, 0, 0, ...] \\ &\approx 0x_2(n) \quad \text{(except 1st sample).} \end{aligned}$$

The filter has a gain of 0 the highest frequency.

- A filter that boosts low frequencies while attenuating higher frequencies is called a **lowpass filter**.

What about frequencies in between?

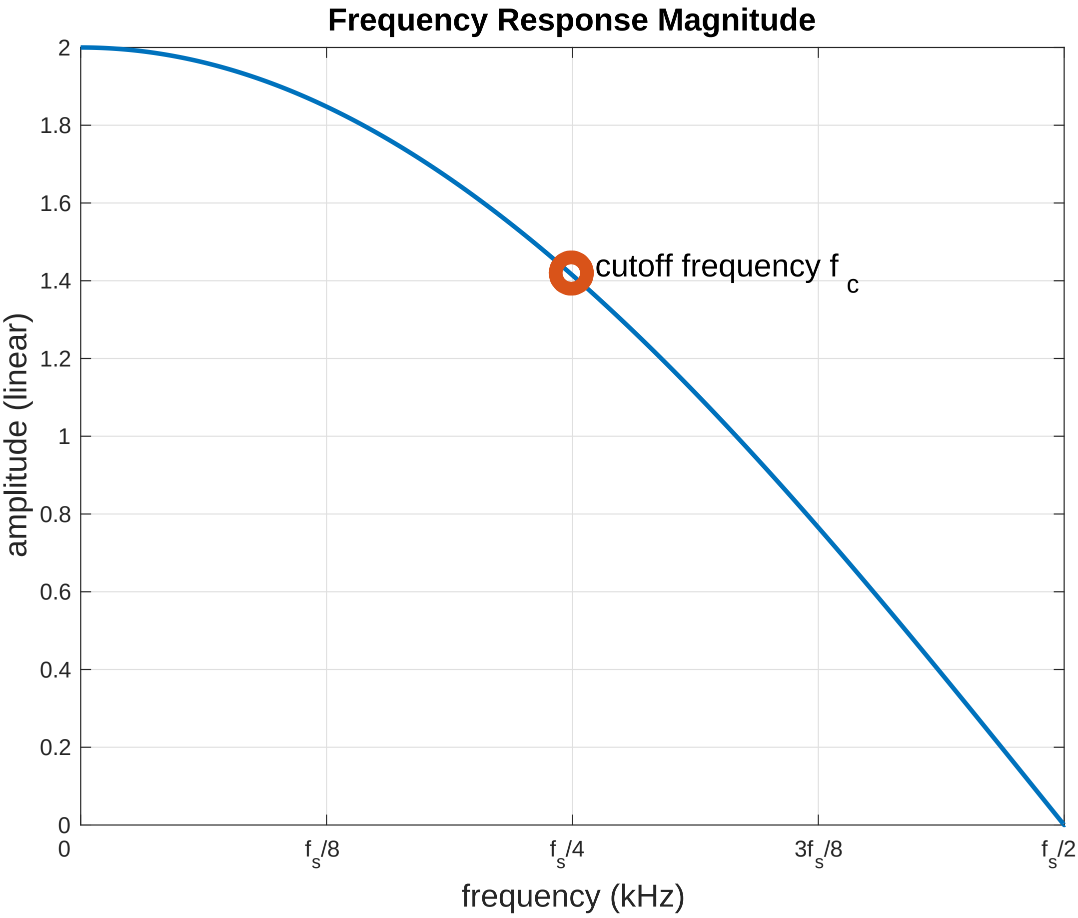
- Filter behaviour can be determined
  - using sinusoids at every possible frequency between 0 and $f_s/2$ Hz;
  - using an input signal **that contains all frequency components** and check just once!
- **Impulse**: signal with the broadest possible spectrum.



- **Impulse Response (IR)**: response to an impulse (e.g. irCave.wav).

Simple Lowpass Frequency Response

- **Frequency response**:
  - spectrum of the impulse response;
  - shows how filter modifies frequency components.
- Frequency response of $y(n) = x(n) + x(n-1)$:



Changing Filter Coefficients

- The **difference** (instead of the sum) of adjacent samples:

$$y(n) = x(n) - x(n-1).$$

is like changing the *coefficient* of $x(n-1)$ to -1.

- At 0 Hz:

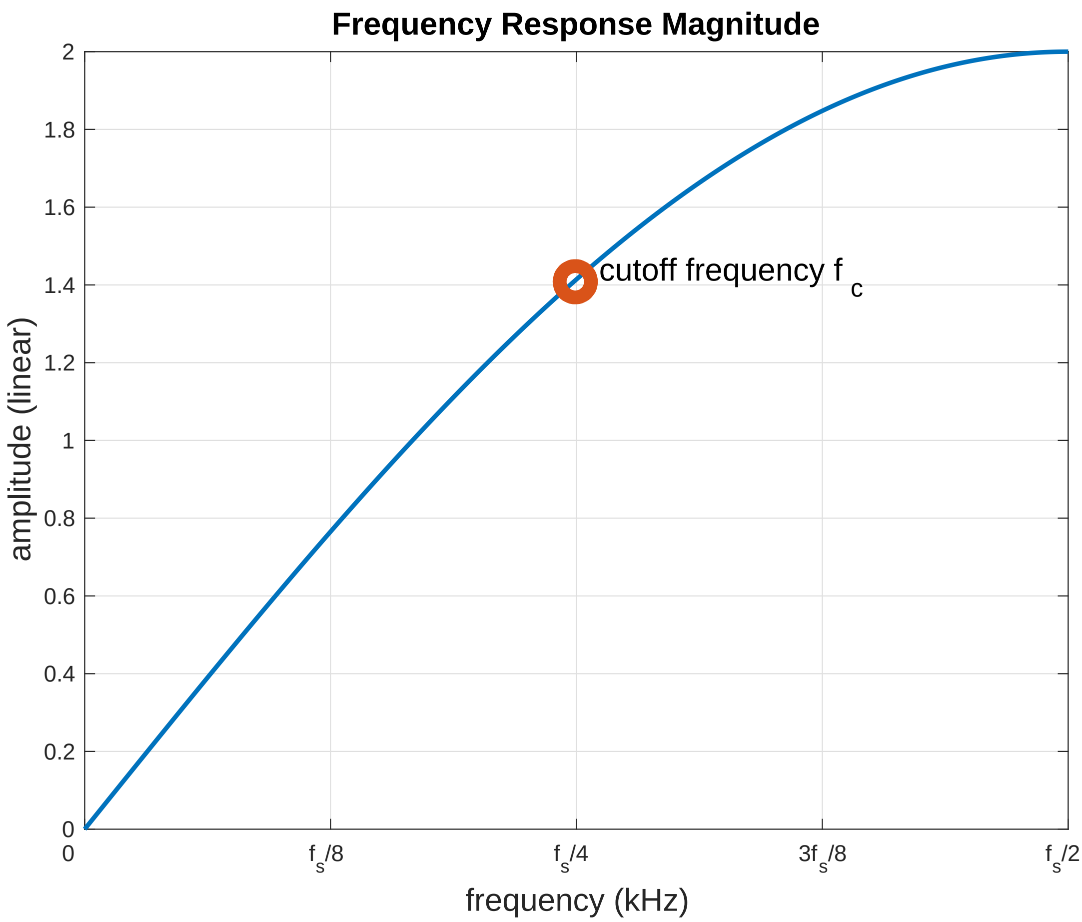
$$\begin{aligned} y(n) &= x_1(n) - x_1(n-1) \\ &= \quad [A, A, A, ...] \\ &\quad - [0, A, A, A, ...] \\ &= \quad [A, 0, 0, 0, ...] \\ &\approx 0x_1(n). \end{aligned}$$

- At $f_s/2$ Hz:

$$\begin{aligned} y(n) &= x_2(n) - x_2(n-1) \\ &= \quad [A, -A, A, -A, ...] \\ &\quad - [0, A, -A, A, ...] \\ &= \quad [A, -2A, 2A, -2A, ...] \\ &\approx 2x_2(n). \end{aligned}$$

Simple Highpass Frequency Response

- Frequency response shows a highpass filter.



- Notice the same cutoff frequency as simple lowpass.

Notch Filter

- Changing the delay of the second term (and adding):

$$y(n) = x(n) + x(n-2),$$

has output

– at 0 Hz ($x_1(n) = [A, A, A, ...]$):

$$\begin{aligned} y(n) &= [A, A, A, ...] + \ [0, 0, A, A, ...] \\ &= [A, A, 2A, 2A, ...] \approx 2x_1(n). \end{aligned}$$

– at $f_s/2$ Hz ($x_2(n) = [A, -A, A, -A, ...]$):

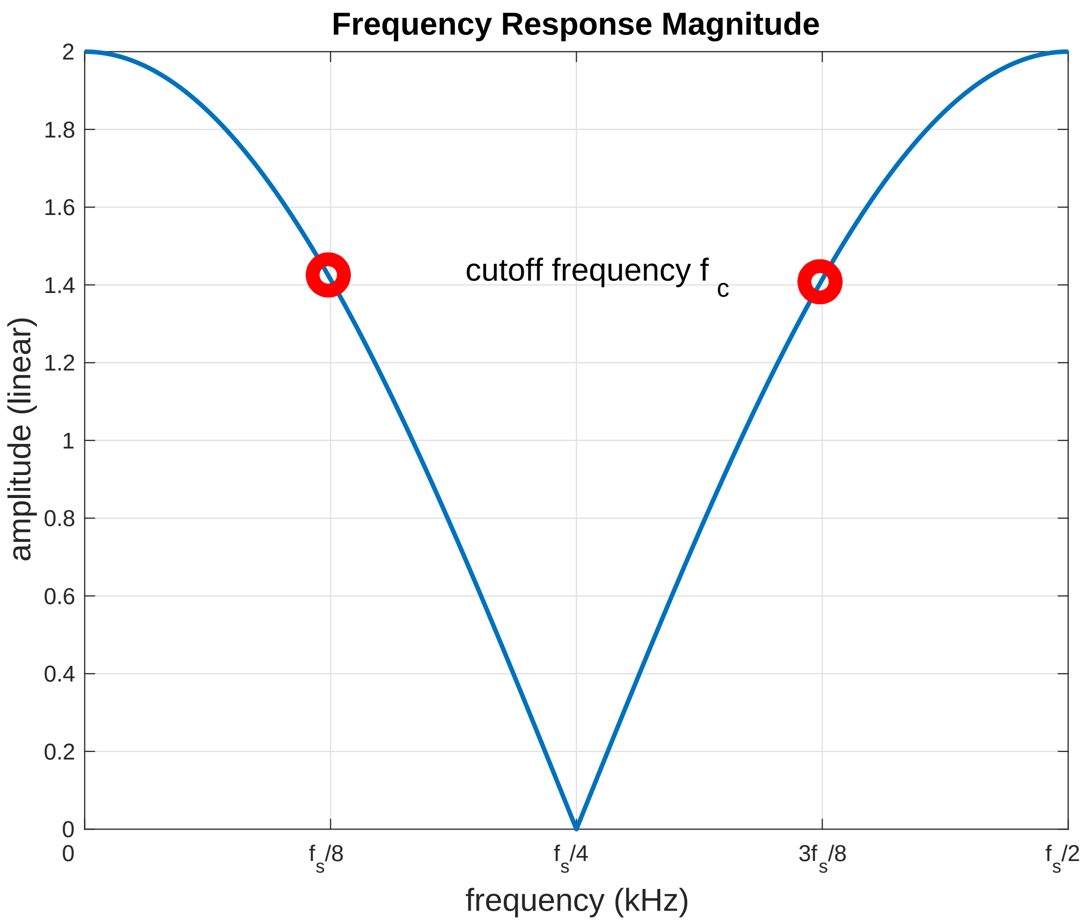
$$\begin{aligned} y(n) &= [A, -A, A, -A, ...] + [0, 0, A, -A, A...] \\ &= [A, -A, 2A, -2A, ...] \approx 2x_2(n) \end{aligned}$$

- This filter boosts both low and high frequencies!
- Output at $f_s/4$ Hz ($x_3(n) = [A, 0, -A, 0, A, 0, ...]$):

$$\begin{aligned} y(n) &= x_3(n) + x_3(n-2) \\ &= [A, 0, -A, 0, A, ...] + [0, 0, A, 0, -A, 0, ...] \\ &= [A, 0, 0, 0, ...] \approx 0x_3(n) \end{aligned}$$

Simple Notch Frequency Response

- Frequency response shows a *notch* filter.



- Notice cutoff frequency is half of that for lowpass.

Bandpass Filter

- Changing coefficient of $x(n-2)$ to -1:

$$y(n) = x(n) - x(n-2),$$

yields output

  - at 0 Hz:

$$\begin{aligned} y(n) &= [A, A, A, ...] - \ [0, 0, A, A, ...] \\ &= [A, A, 0, 0, ...] \approx 0x(n). \end{aligned}$$

  - at $f_s/2$ Hz:

$$\begin{aligned} y(n) &= [A, -A, A, -A, ...] - \ [0, 0, A, -A, A...] \\ &= [A, -A, 0, 0, ...] \approx 0x(n). \end{aligned}$$

  - at $f_s/4$ Hz:

$$\begin{aligned} y(n) &= [A, 0, -A, 0A, ...] - \ [0, 0, A, 0, -A, 0, ...] \\ &= [A, 0, -2A, 0, 2A, 0, -2A, 0, ...] \approx 2x(n). \end{aligned}$$

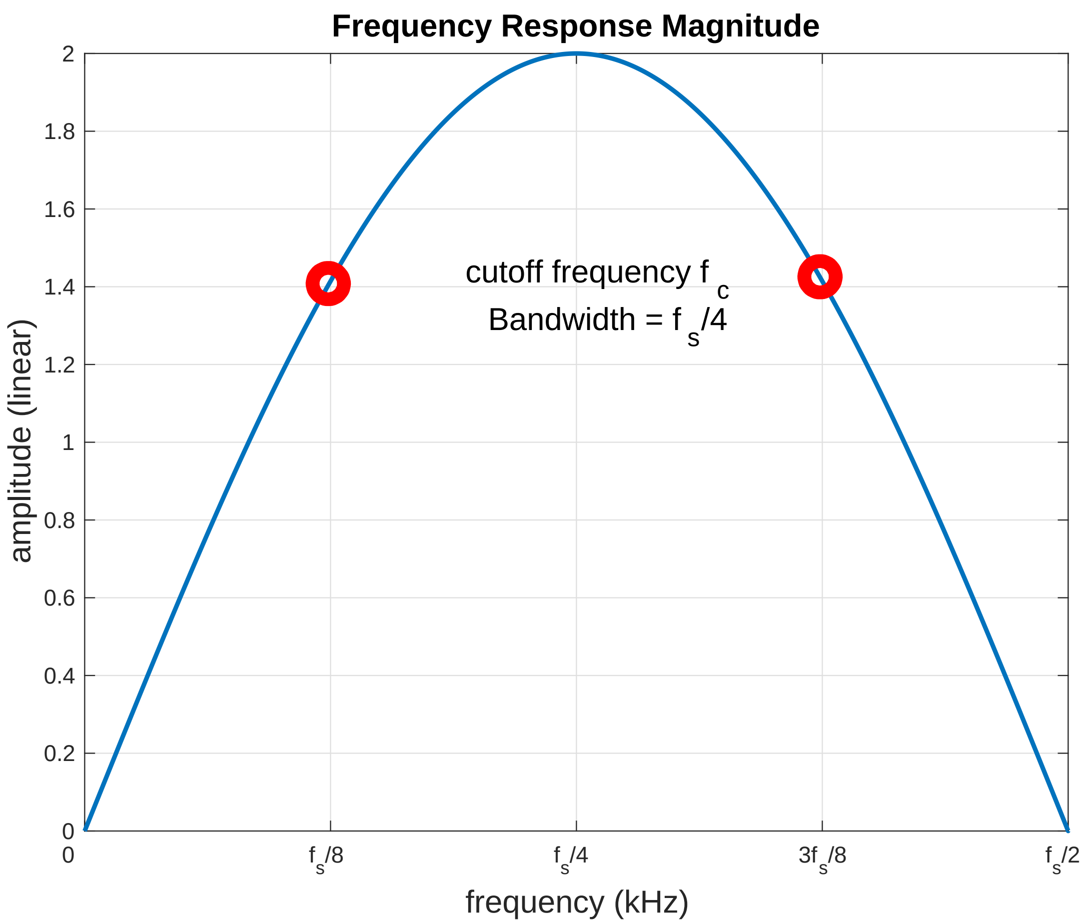
- Attenuation is at 0 and $f_s/2$ Hz, and boosts $f_s/4$

Bandpass Filter Frequency Response

- The filter frequency (amplitude) response for

$$y(n) = x(n) - x(n-2)$$

shows it is a **bandpass filter**.



- The bandwidth is determined by the frequency separation between the two cutoff points.

Plots of simple filters

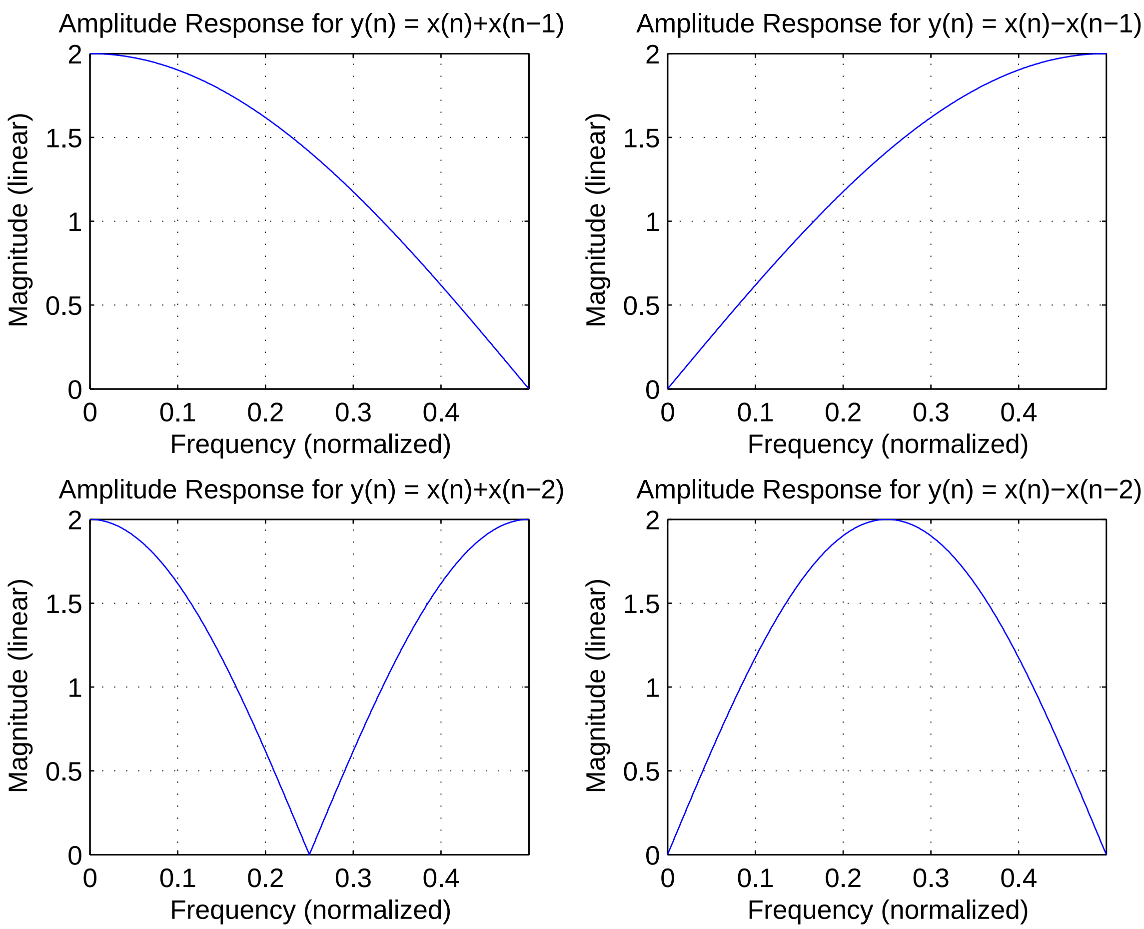


Figure 2: Amplitude Responses for simple filters

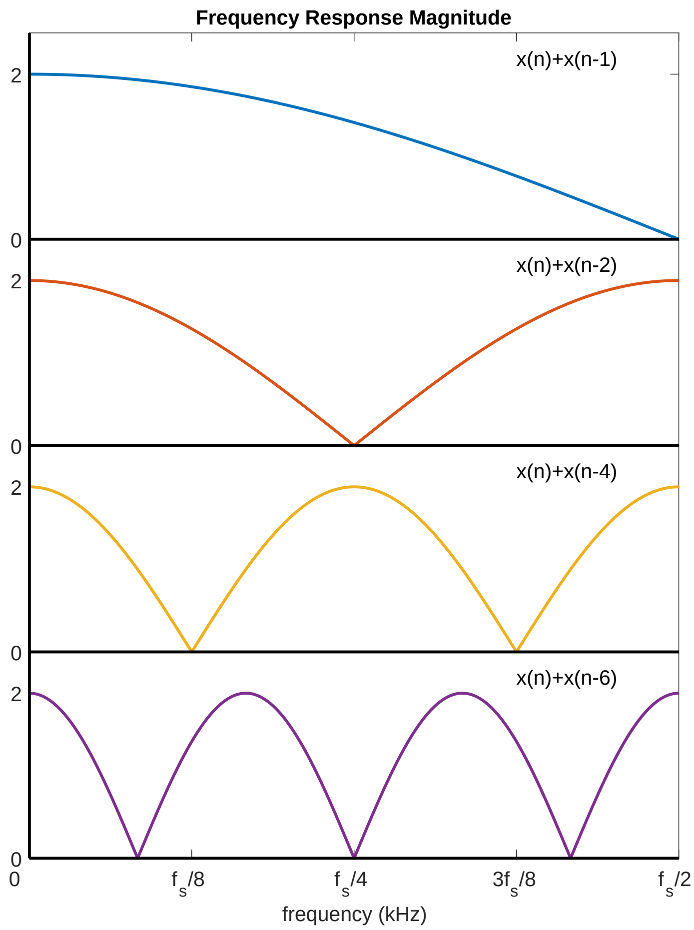
.

Increasing the phase delay

- Make the delay of the 2^{nd} term variable:

$$y(n) = x(n) + x(n - M)$$

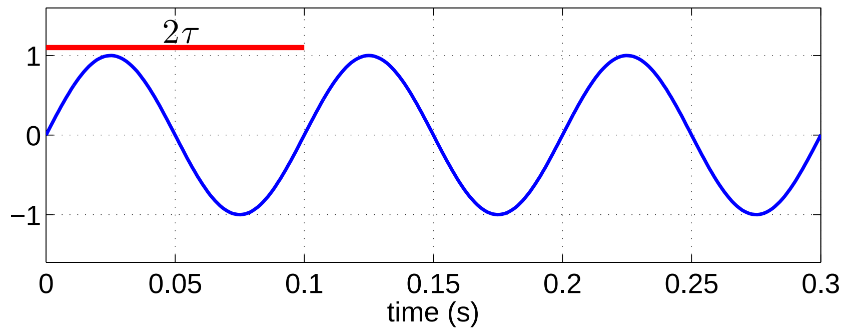
- Effects of increasing the M (the filter *order*):



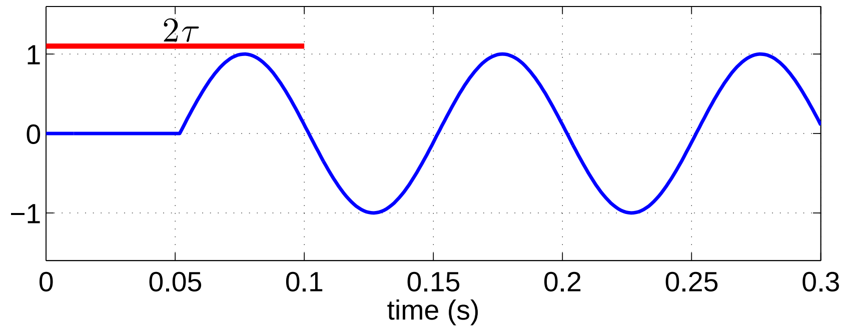
- Notice regularly spaced peaks and notches.
- Notches at odd harmonics of **what frequency**?

Cancellation at Notch Frequencies

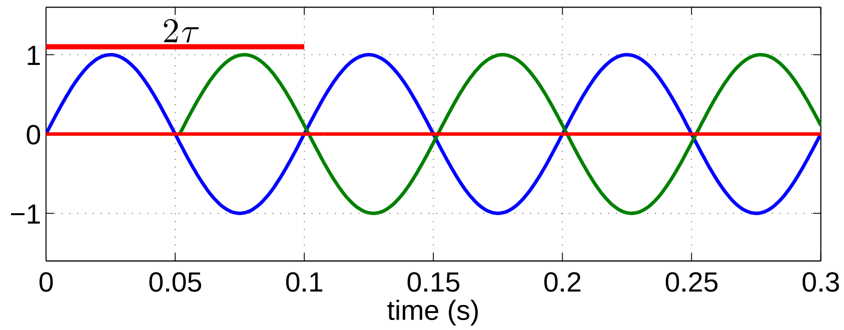
- Consider a sinusoid at $f = 1/(2\tau)$ (period of 2τ):



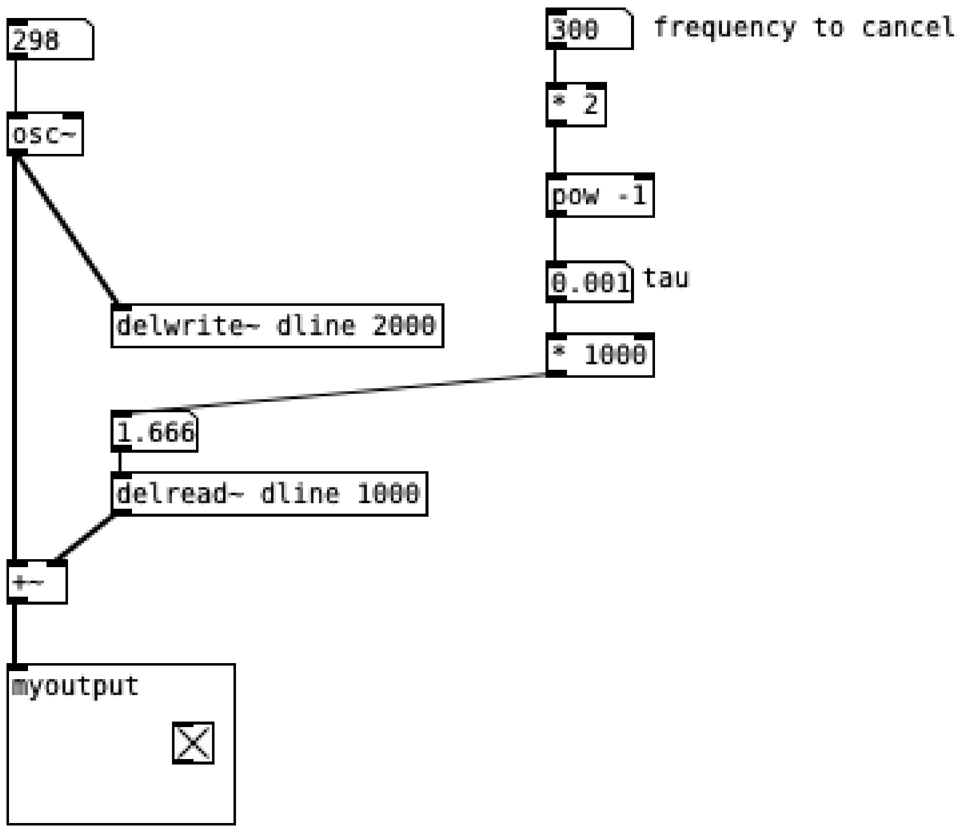
- Delaying that sinusoid by τ (1/2 a period) yields:



- Summing with original yields complete cancellation:



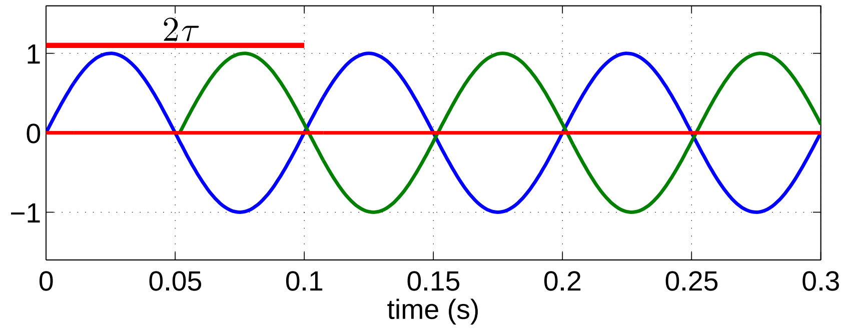
Listen to Cancellation in Pd



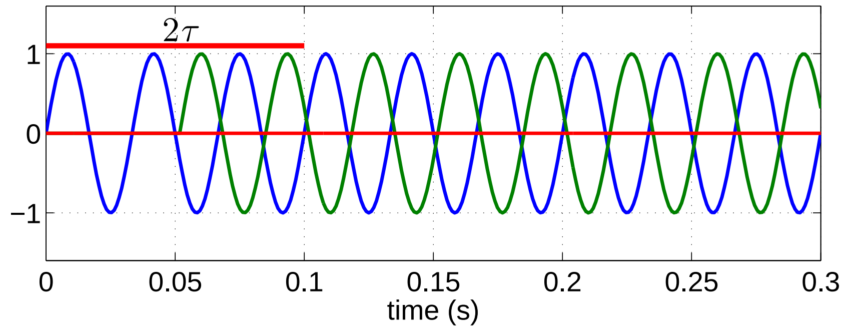
- comb11.21.19.pd

Cancellation at Odd Harmonics

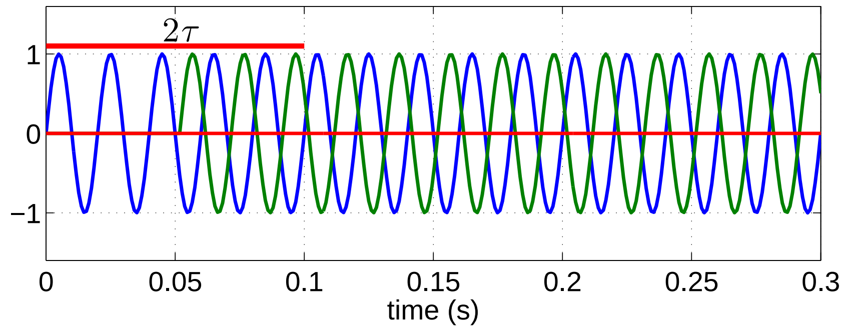
- Adding to a sinusoid at $f = 1/(2\tau)$ a version of itself delayed by τ yields cancellation at $f = 1/(2\tau)$,



- but also at $f = 3/(2\tau)$,



- and at $5/(2\tau)$



- and at all **odd** harmonics of $f = 1/(2\tau)$.

Relating τ to delay of M samples

- For $y(n) = x(n) + x(n - M)$ delay is M samples or

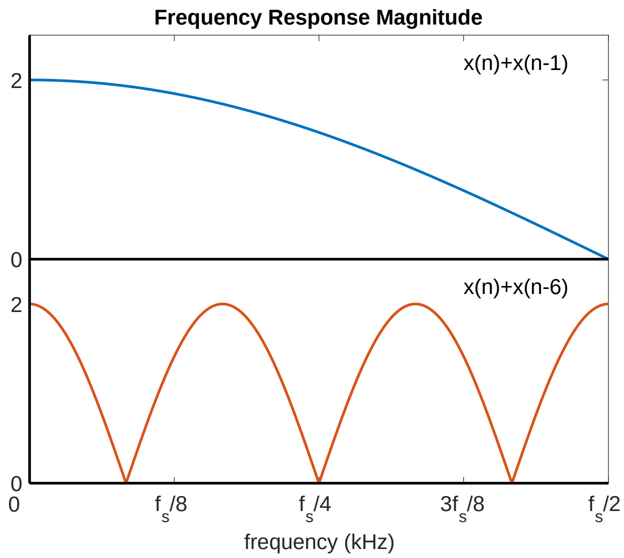
$$\tau = \frac{M}{fs} \text{ seconds.}$$

- There is complete attenuation (notch) at frequency

$$f = \frac{1}{2\tau} = \frac{1}{2M/f_s} = \frac{f_s}{2M}$$

and at odd harmonics $3f, 5f, ...$ (up to Nyquist limit).

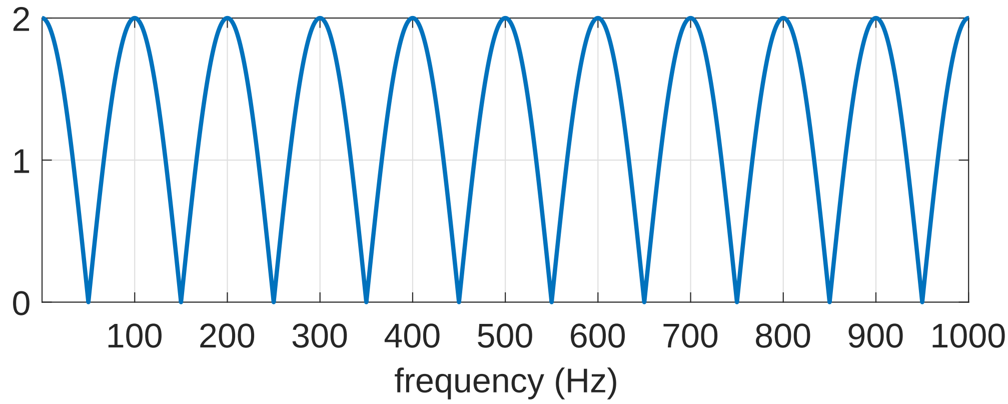
- For $M = 1$ (lowpass) there is 1 notch at $f_s/2$.



- For $M = 6$ there are notches at $f_s/12, f_s/4, 5f_s/12$.

Feedforward Comb Filter

- Regular (comb-like) spacing of peaks/notches suggests **harmonics** of a fundamental frequency f_0.



- If notches are at **odd** harmonics of

$$f_n = \frac{1}{2\tau},$$

then peaks are at harmonics of

$$f_0 = 2f_n = \frac{2}{2\tau} = \frac{1}{\tau}.$$

- For a desired fundamental (sounding) frequency f_0:

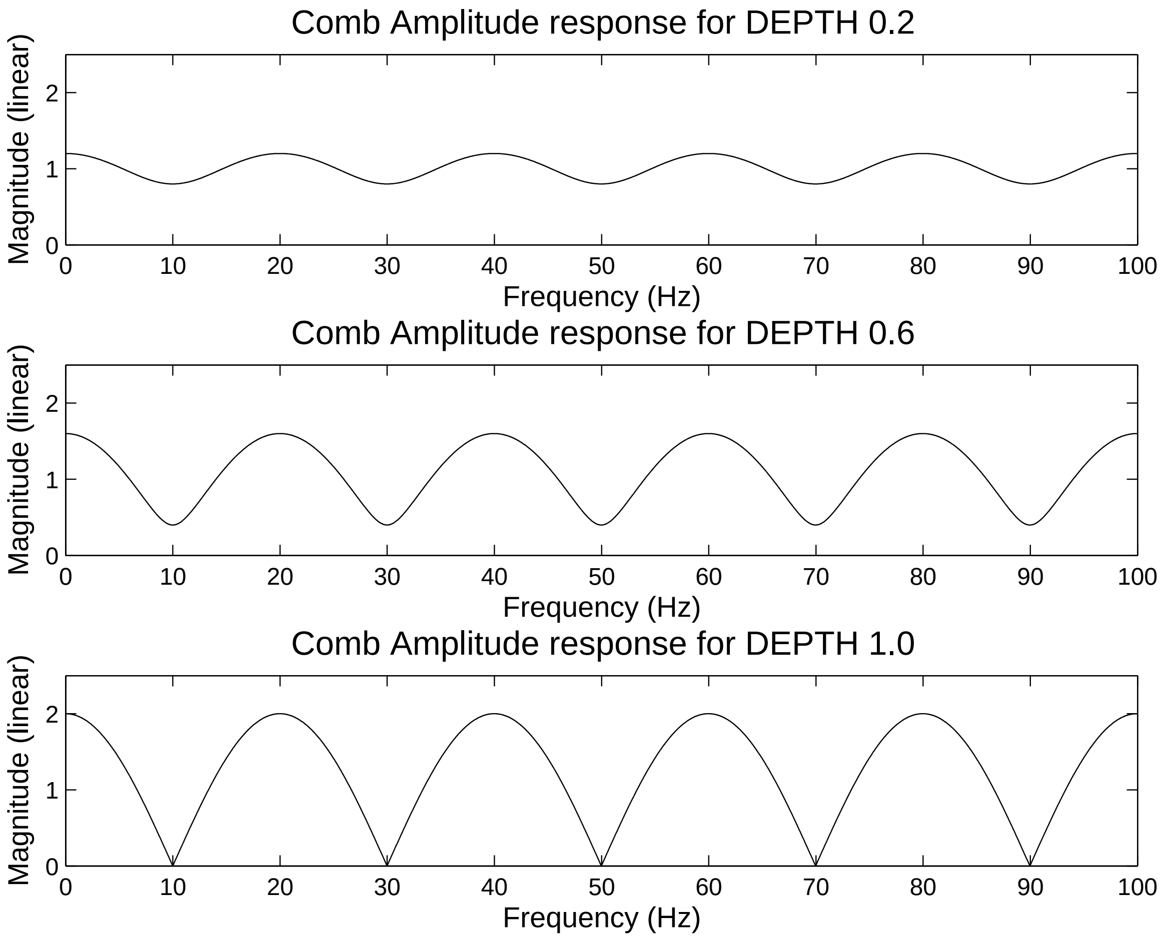
$$\tau = \frac{1}{f_0} \text{ seconds} \qquad \text{OR} \qquad M = \frac{f_s}{f_0} \text{ samples.}$$

Feedforward Comb Filter in Pd

- see ffcomb2.pd.

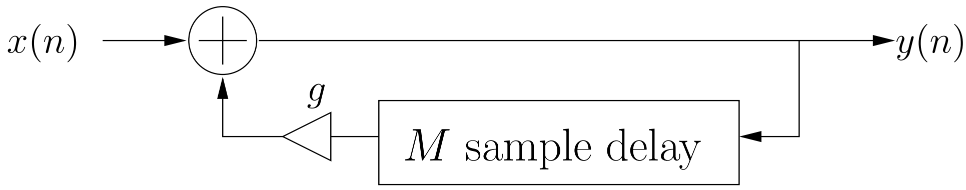
Feedforward Comb Coefficient

- Introducing a coefficient allows for control of cancellation amount and the **depth** of notches:



The Feedback Comb Filter

- What happens when the output of a delay line is multiplied by gain $g < 1$ then fed back to the input?



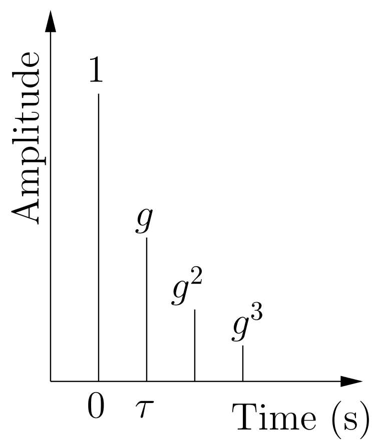
- The difference equation for this filter is

$$y(n) = x(n) + gy(n - M),$$

- If the input to the filter is an impulse

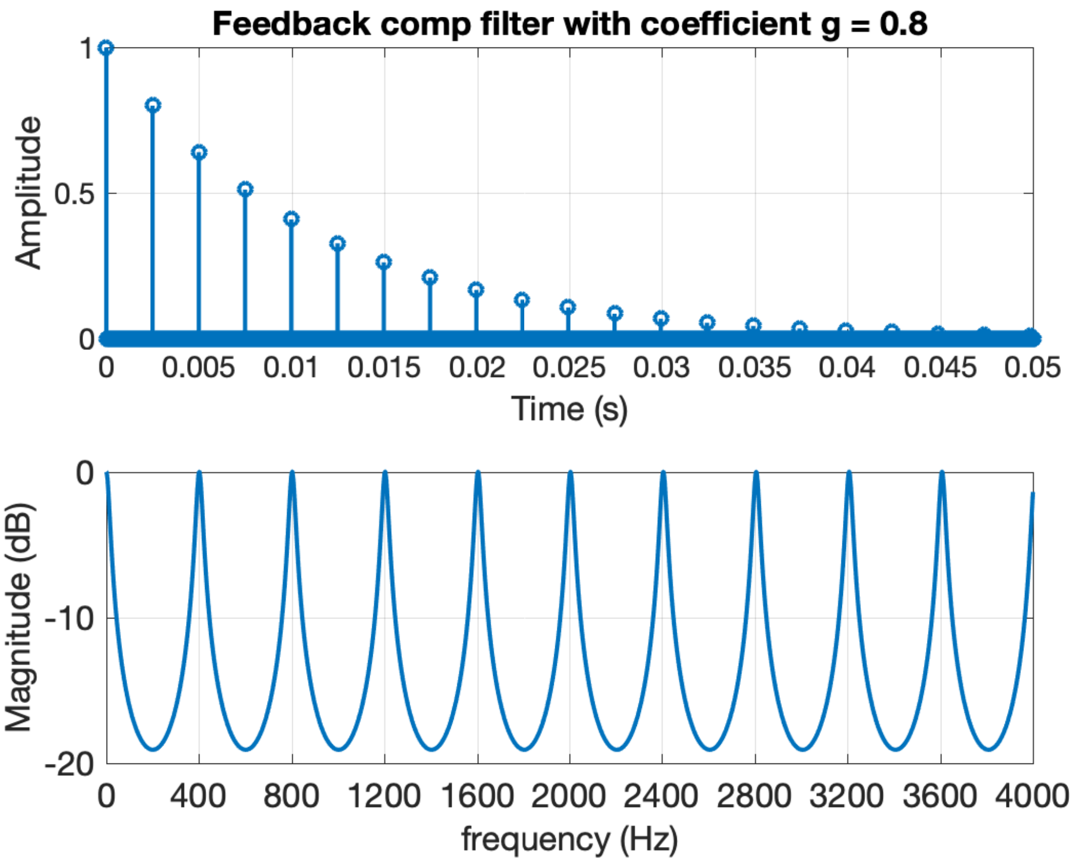
$$x(n) = \{1, 0, 0, \ldots\}$$

the output (impulse response) will be ...



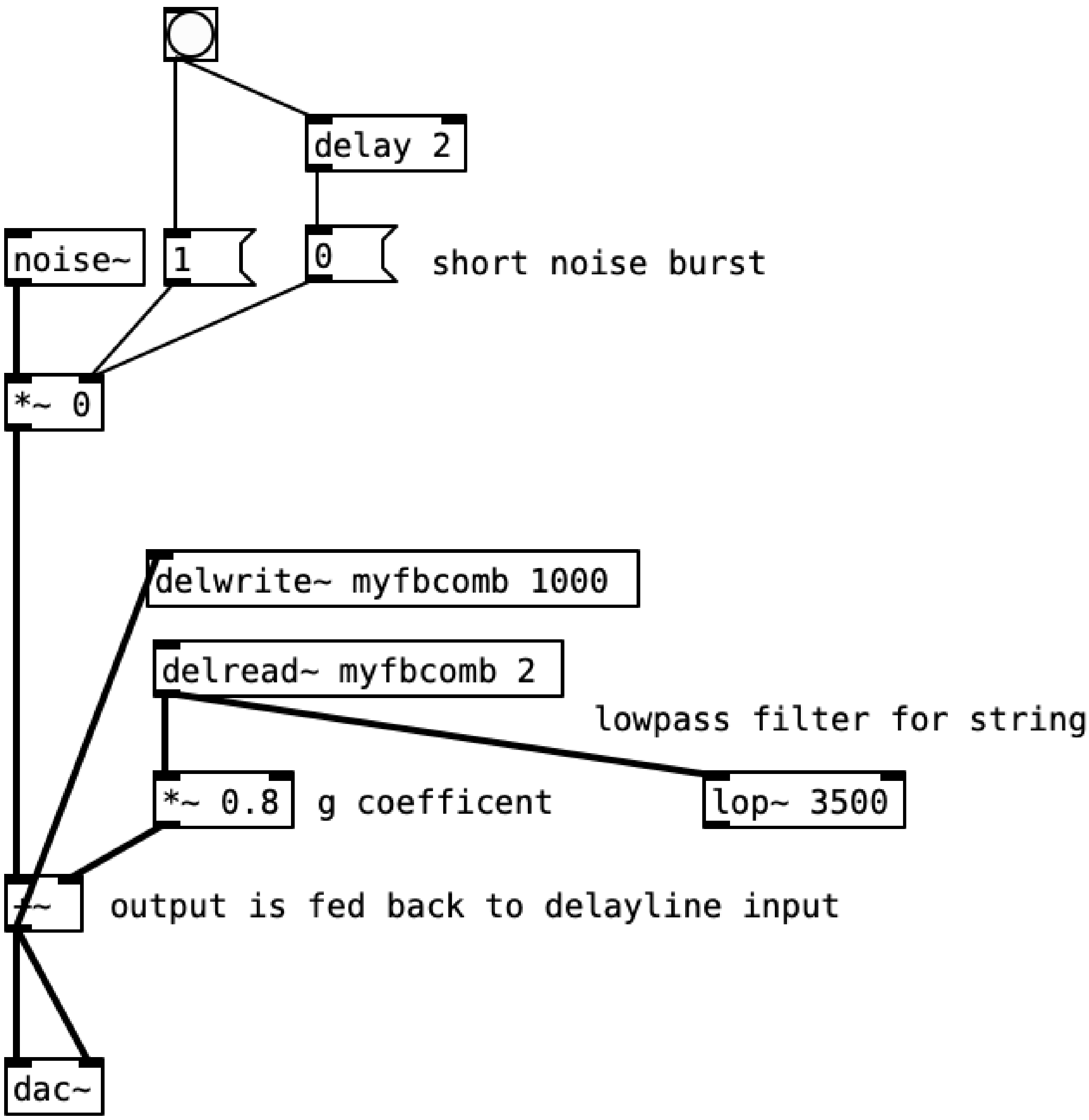
Feeback Comb Filter Frequency

- Pulses are equally spaced in time at interval $\tau = M/f_s$ seconds.
- It is periodic and will sound at frequency $f_0 = 1/\tau$.



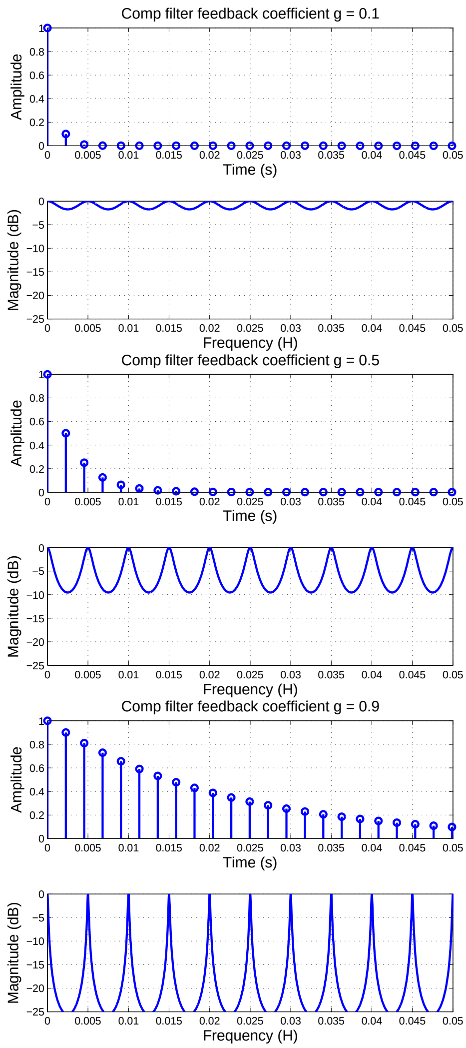
- Spacing between the maxima of the comb "teeth" is equal to the natural frequency f_0.

Feedback Comb Filter in Pd



Effect of the Feedback coefficient

- Minima depth and maxima height controlled by g.
- Values closer to 1 yield more extreme max/min.



Comb Filter Decay Rate

- The response decays exponentially as determined by the loop time and gain factor g.
- Values of g nearest 1 yield the longest decay times.
- To obtain a desired decay time, g may be approximated by

$$g = 0.001^{\tau/T60}$$

where

$$\begin{aligned} \tau &= \text{the loop time} \\ T60 &= \text{the time to decay by 60dB} \end{aligned}$$

and 0.001 is the level of the signal at 60dB down.

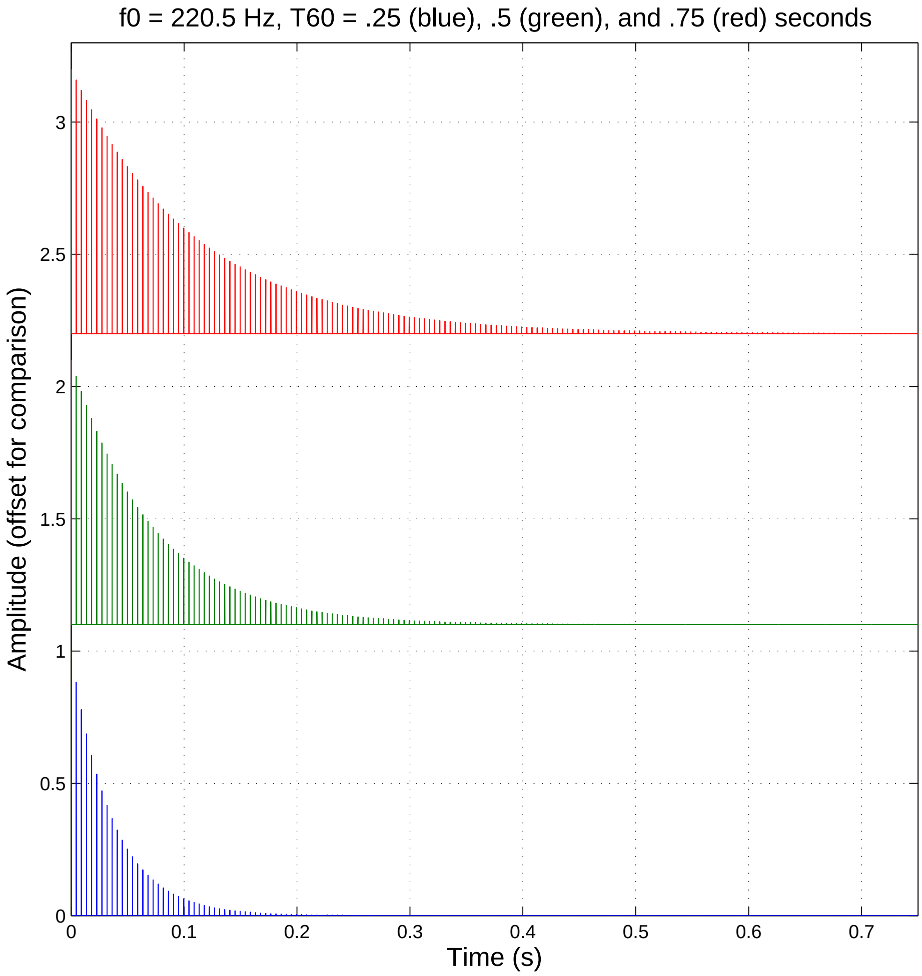


Figure 3: Comb filter impulse responses with a changing the decay rate.

A very simple string model

- A very simple string model can be implemented using a single delay line and our simple first-order low pass filter to model frequency-dependent loss.

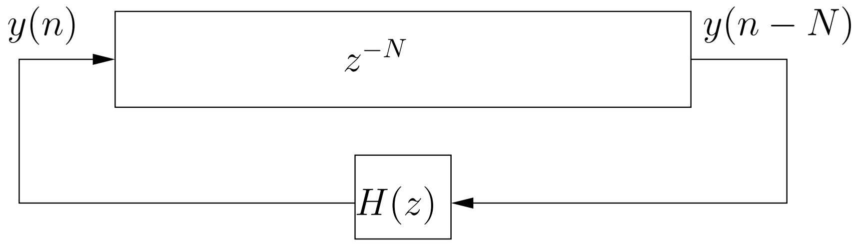


Figure 4: A very simple model of a rigidly terminated string.

- All losses have been *lumped* to a single observation point in the delay line, and approximated with our first-order simple low-pass filter

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

- Different sounds can be created by changing this filter.
- The Karplus-Strong Algorithm may be interpreted as a **feedback comb filter** (with lowpassed feedback) or a simplified **digital waveguide** model.
- How do you *pluck* the string? (noise burst.)