DT Filter Application

These notes explore the use of DT filters to remove an interference (in this case a single tone) from an audio signal. Imagine that you are in the your home recording studio and have just recorded what you feel is a "perfect take" of a guitar solo for a song you are recording, but you discover that someone had turned on some nearby electronic device that caused electromagnetic radiation that was picked up somewhere in the audio electronics and was recorded on top of the guitar solo. Rather than try to recreate this "perfect take" you decide that maybe you can design a filter to remove it.

We will explore two different cases:

(i) a high-pitched tone that lies above the significant portion of the guitar signal's spectrum, and

(ii) a mid-pitched tone that lies in the middle of the guitar signal's spectrum.

I. Signal Access and Exploration

1. Use MATLAB's wavread command to load the guitar1.wav file:

2. Listen to the guitar signal using MATLAB's sound command.

3. Plot the first second or so of the signal in the time domain to see what the signal looks like.

4. Look at the guitar signal in the frequency domain by computing and plotting (in dB) the DFT of various 16384-pt blocks of the guitar signal. Verify that the significant portion of the guitar signal's spectrum lies below 5 kHz.

II. Adding A High Frequency Interference

1. Create a sinusoid whose frequency is 10kHz that is sampled at the same rate as the guitar signal and has the same length. The amplitude of this sinusoid should be 1.

2. Add this signal to the guitar signal to create the simulated recorded signal that has the interference (call this signal x_10 to indicate that it has an interference at 10 kHz).

3. Listen to the guitar signal with interference using MATLAB’s sound command.

4. Plot the first second or so of the signal with interference and the signal without interference.

5. Compute the DFTs of the signal that has interference. Verify that the interference is outside the significant portion of the guitar spectrum.

III. Lowpass Filter Design

MATLAB contains some easy to use routines for designing FIR filters – FIR (finite-impulse response) filters don't use any output feedback – therefore they don't really have any poles and they will always be stable. They are the most widely used type of DT filter in practice.

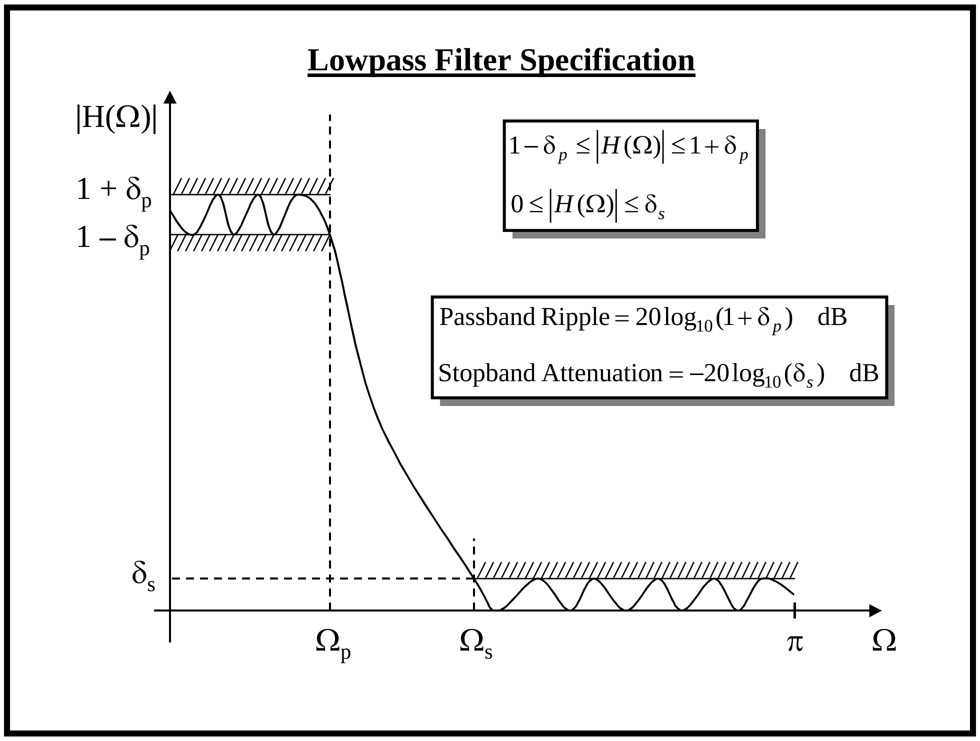
A simple FIR filter: $y[n] = \frac{1}{3}x[n] + \frac{1}{3}x[n-1] + \frac{1}{3}x[n-2]$

A more general FIR filter: $\boxed{y[n] = \sum_{i=0}^{N} b_i x[n-i]}$ N = "Order of Filter"

They are quite easy to design using software-based tools. We'll use the MATLAB FIR design routines called remezord.m and remez.m

The command remezord will give an estimate of the FIR filter order needed to achieve given specifications. The routine remez.m will then give the required design.

Here is how we state the filter specifications:



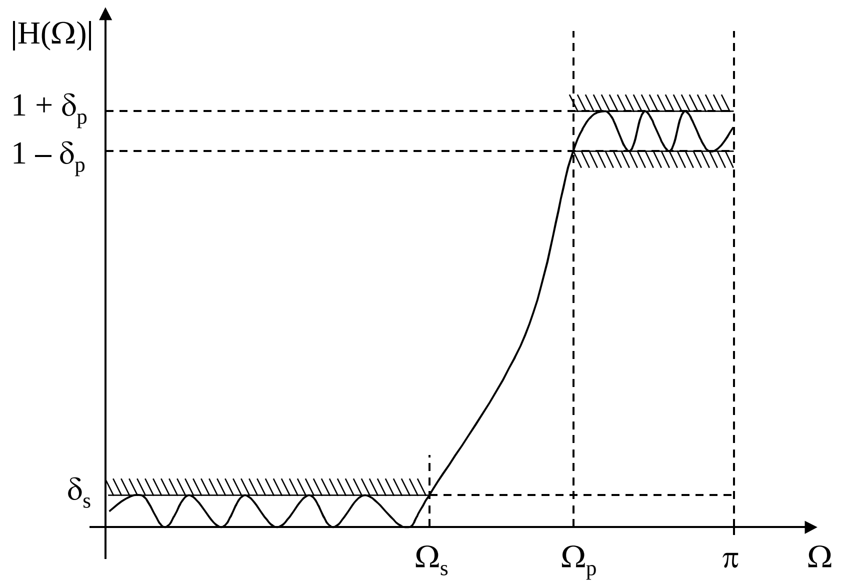
$1 - \delta_p \le |H(\Omega)| \le 1 + \delta_p$

$0 \le |H(\Omega)| \le \delta_s$

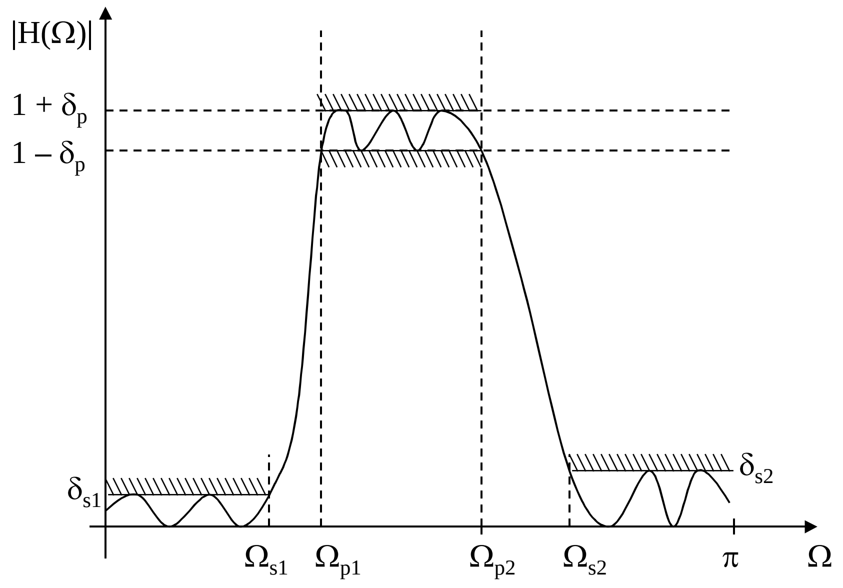
$\text{Passband Ripple} = 20\log_{10}(1+\delta_p) \quad \text{dB}$

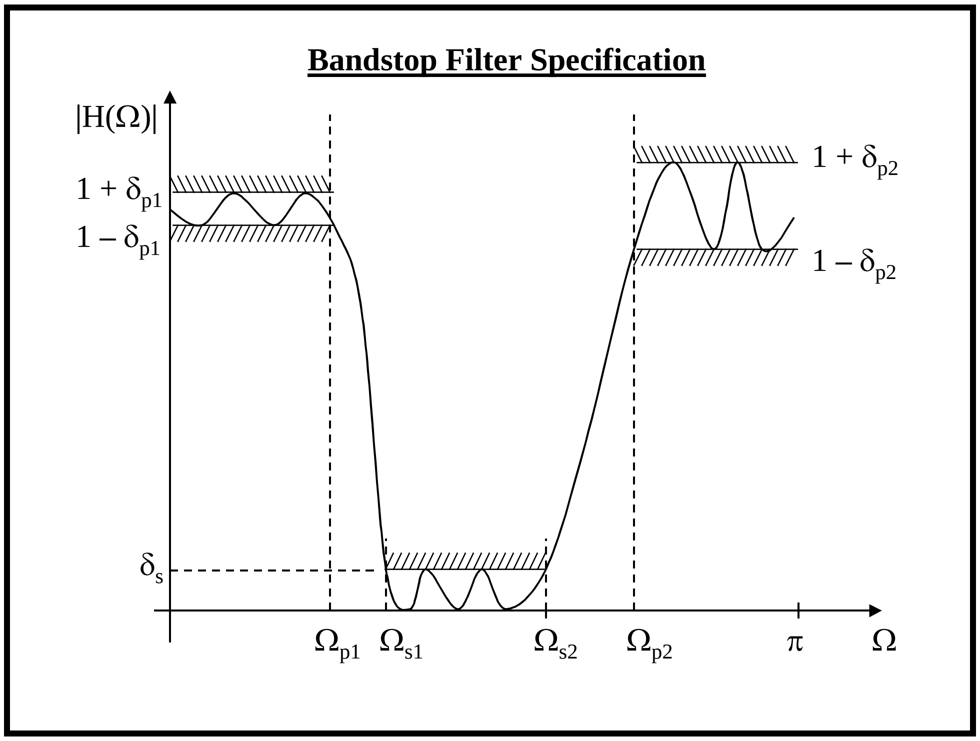
$\text{Stopband Attenuation} = -20\log_{10}(\delta_s) \quad \text{dB}$

Highpass Filter Specification



Bandpass Filter Specification





1. Use the “remezord” and “remez” commands to design a **lowpass** filter needed to achieve:
   - **<u>60 dB of attenuation</u>** in the stopband for the undesired signal
   - **<u>1 dB of passband ripple</u>**
   - **<u>passband edge at 7kHz</u>**
   - **<u>stopband edge at 9 kHz</u>**.

   Look at DFTs to see why 60 dB of attenuation is a reasonable choice.

   Use the MATLAB variable b for the vector that holds the FIR filter coefficients.

2. Plot the filter’s impulse response.
   - For an FIR filter it is easy to show that the impulse response is nothing more than the b_i coefficients in its difference equation

$$y[n] = \sum_{i=0}^{N} b_i x[n-i]$$

3. Compute and plot the filter's frequency response.

4. Make a pole-zero plot for the filter's transfer function.

$$\boxed{y[n] = \sum_{i=0}^{N} b_i x[n-i]} \Rightarrow \boxed{\begin{aligned} H(z) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N} \\ &= \frac{b_0 z^N + b_1 z^{N-1} + b_2 z^{N-2} + \cdots + b_N}{z^N} \end{aligned}}$$

<u>IV. Remove Interference with Filter</u>

1. Use the designed filter to remove the interference
   - Filter x_10 using the LPF to get x_10_out

y = filter(b,a,x) filters the data in vector x with the
filter described by vectors a and b to create the filtered
data y. The vectors a and b come from the coefficients in the difference equation:

$$\sum_{i=0}^{N_a} a_i y[n-i] = \sum_{i=0}^{N_b} b_i x[n-i]$$

a = $[a_0 \quad a_1 \quad a_2 \;\ldots\; a_{Na}\,]$

b = $[b_0 \quad b_1 \quad b_2 \;\ldots\; b_{Nb}\,]$

For an FIR filter like we have here the difference equation is:

$y[n] = \sum_{i=0}^{N} b_i x[n-i]$ so the a "vector" is a = 1

2. Assess the performance of the filter:
   - Compare x_10_out, x_10, and x in the <u>frequency domain</u>.
   - Compare x_10_out, x_10, and x in the <u>time domain</u>.
   - <u>Listen</u> to the filtered guitar signal using MATLAB’s sound command.

V. Repeat for a Midrange Interferer

Now imagine that the interfering signal is a 3000 Hz sinusoid of unit amplitude. Now you can’t simply design a <u>lowpass</u> filter because it would filter out the guitar frequencies above 3000 Hz.

1. As a test, change the lowpass filter design above to have a passband cutoff of 2500 and a stopband cutoff of 2900 and apply the filter to the **original** (interference-free) guitar signal.

2. Compare the spectrum of this filter’s output to that of the original signal

3. Listen to this filter’s output.

4. Add a unit amplitude, 3000 Hz sinusoid to the guitar signal and use the DFT to see what the spectrum looks like.

5. Design a bandstop (i.e., notch filter) using remezord and remez as follows.
   a. Look at the spectrum of the interfered-with signal to make decisions about appropriate filter spec values.

6. Look at the filter’s frequency response and pole-zero plot

7. Apply the filter to the signal and assess the result in the frequency and the time domain as well as by listening to it.