

EECE 301
Signals & Systems
Prof. Mark Fowler

Discussion #12a

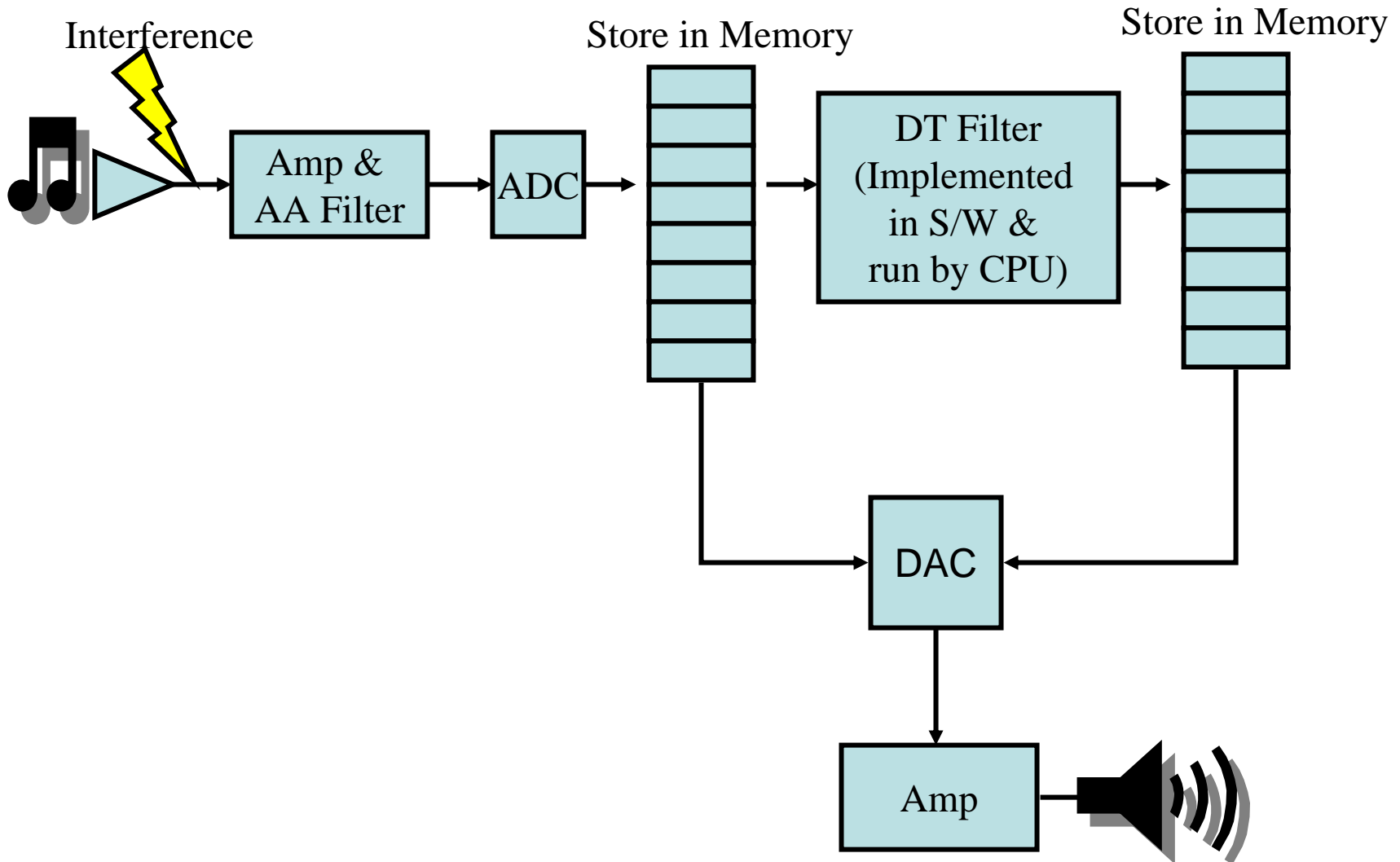
- DT Filter Application

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 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 DT 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.

Real Set-Up



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.

%%%%%%%%%% I. Signal Access and Exploration %%%%%%%%%%%

```
[x,Fs]=wavread('guitar1.wav');  
x=x.'; % convert into row vector
```

```
sound(x,Fs);
```

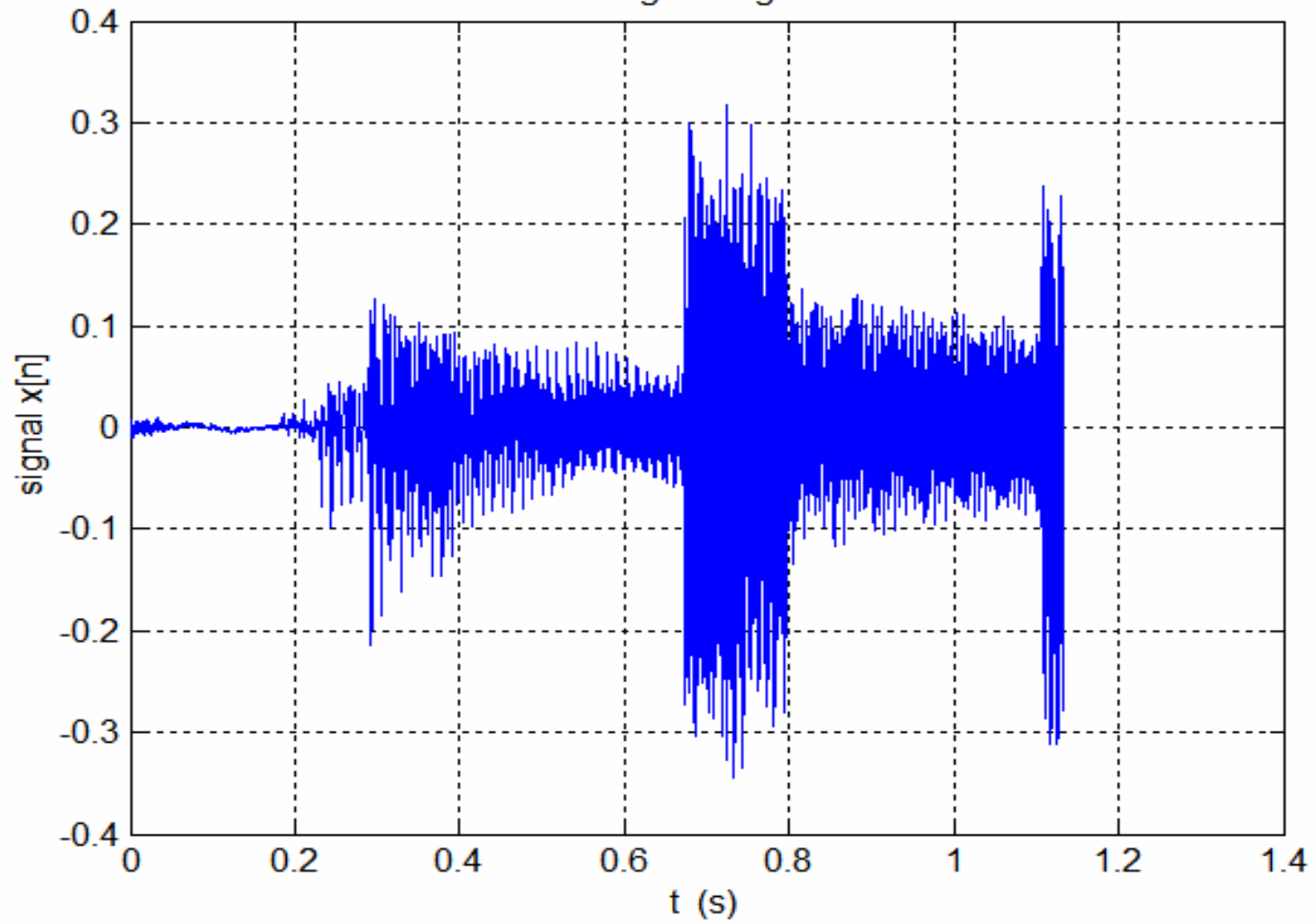
```
t=(0:49999)*(1/Fs);  
plot(t,x(1:50000))
```

```
X1=fftshift(fft(x(20000+(1:16384))),65536));  
X2=fftshift(fft(x(40000+(1:16384))),65536));  
X3=fftshift(fft(x(60000+(1:16384))),65536));  
X4=fftshift(fft(x(80000+(1:16384))),65536));
```

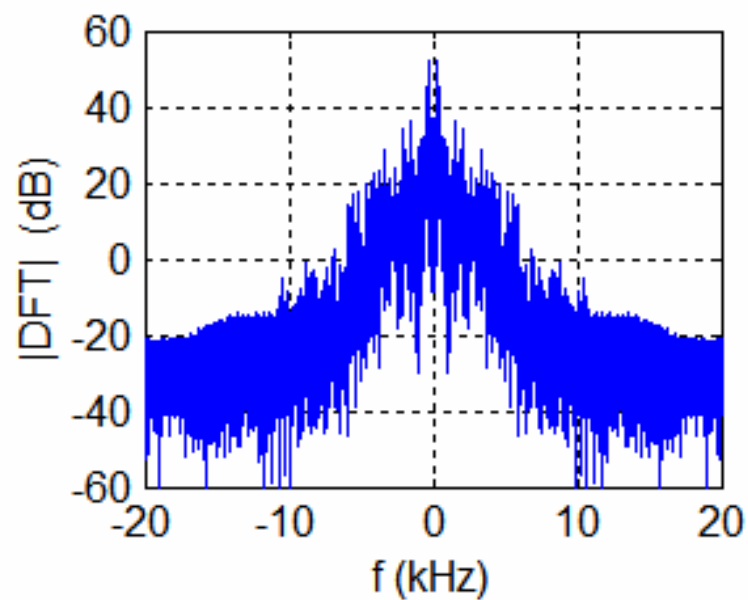
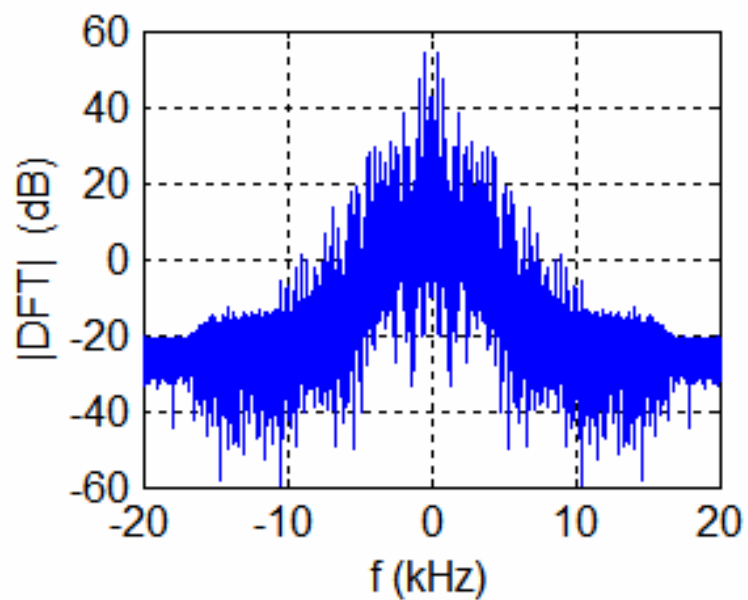
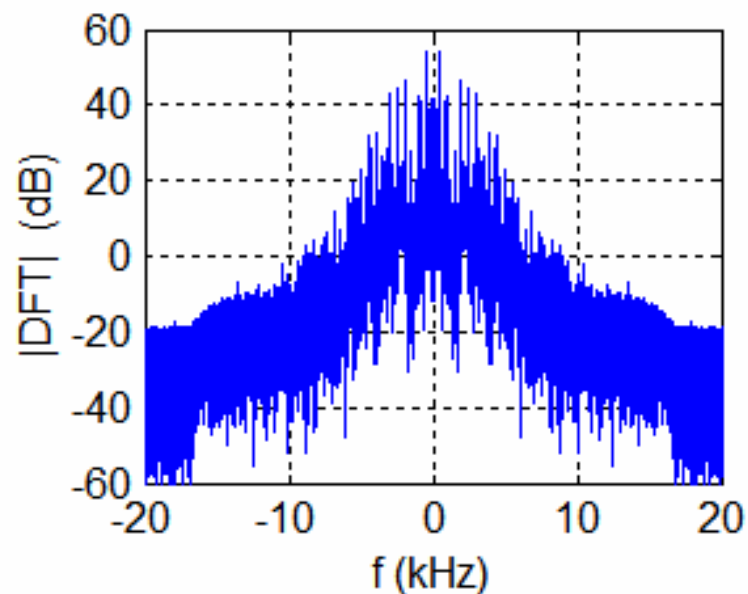
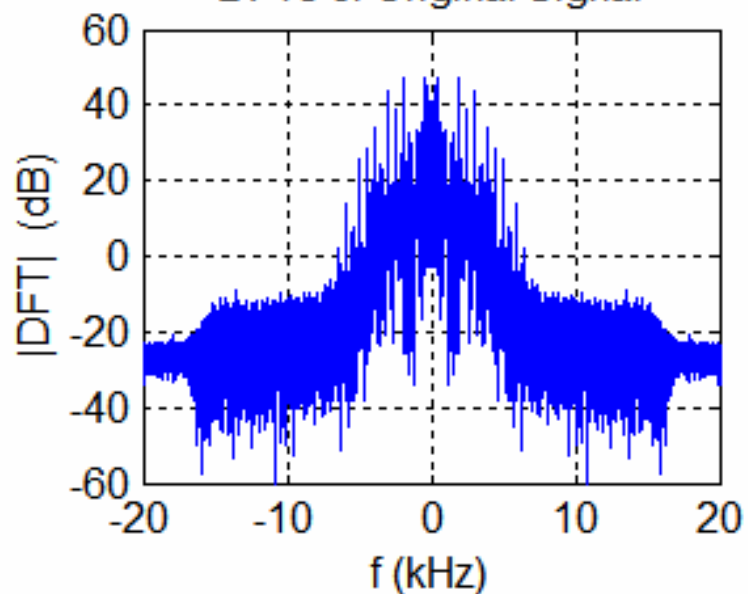
```
f=(-32768:32767)*Fs/65536;
```

```
Figure;  
subplot(2,2,1); plot(f/1e3,20*log10(abs(X1)));  
subplot(2,2,2); plot(f/1e3,20*log10(abs(X2)));  
subplot(2,2,3); plot(f/1e3,20*log10(abs(X3)));  
subplot(2,2,4); plot(f/1e3,20*log10(abs(X4)));
```

Original Signal



DFTs of Original Signal



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.

II. Adding A High Frequency Interference

```
omega=2*pi*(10000/Fs); % convert 10 kHz into DT frequency
```

```
N=length(x);
```

```
n=0:(N-1);
```

```
x_10=x+cos(omega*n);
```

```
sound(x_10,Fs);
```

```
figure
```

```
t=(0:49999)*(1/Fs);
```

```
plot(t,x_10(1:50000),'r',t,x(1:50000))
```

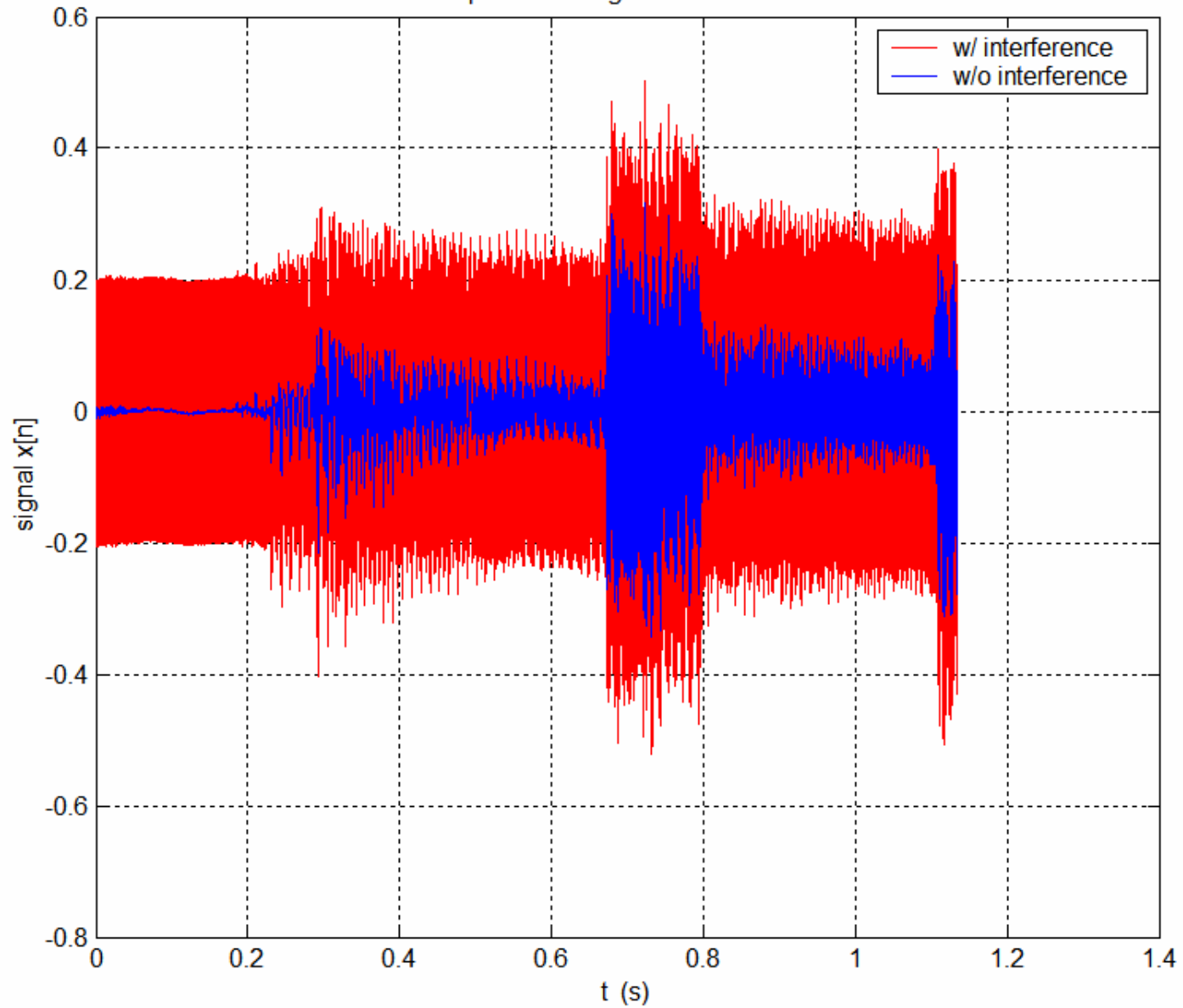
```
X_10_1=fftshift(fft(x_10(20000+(1:16384))),65536));
```

```
figure;
```

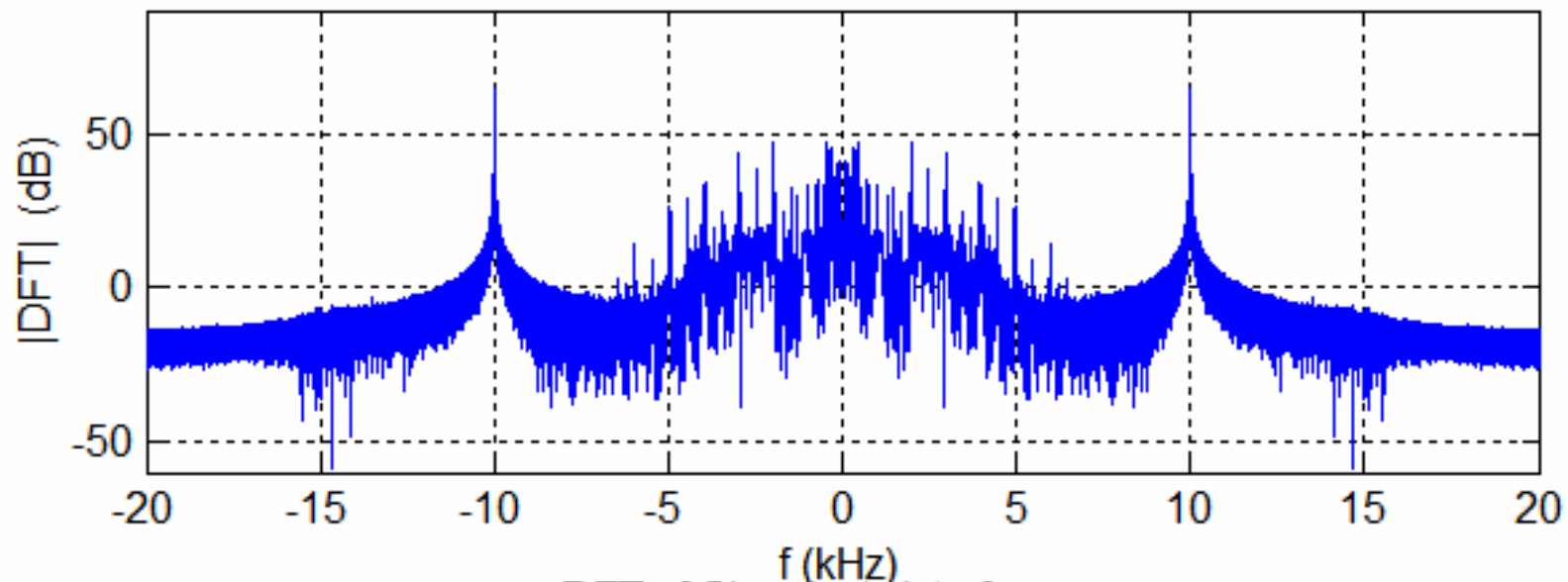
```
subplot(2,1,1); plot(f/1e3,20*log10(abs(X_10_1)));
```

```
subplot(2,1,2); plot(f/1e3,20*log10(abs(X1)));
```

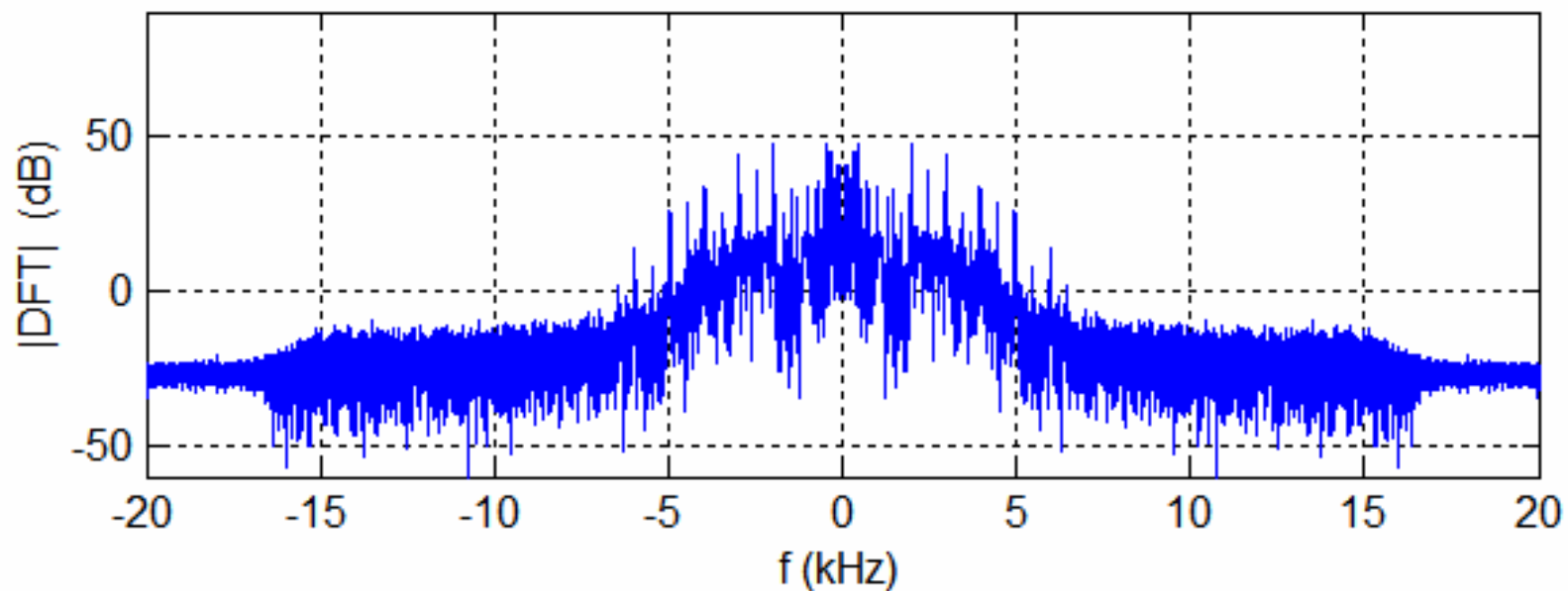
TD Comparison of Signal w/ and w/o Interferer



DFT of Signal w/ Interference



DFT of Signal w/o Interference



III. 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:

$N =$ “Order of Filter”

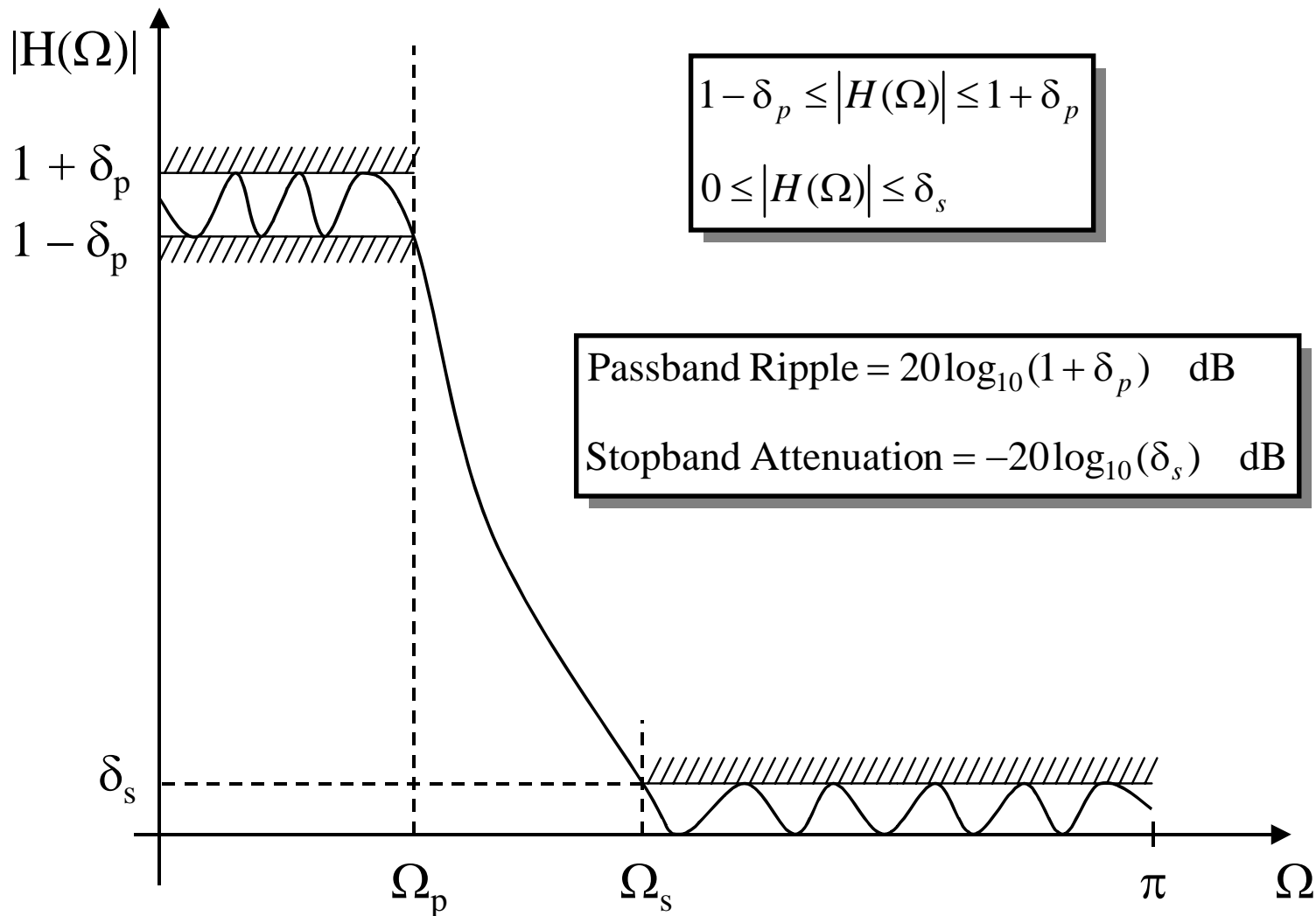
$$y[n] = \sum_{i=0}^N b_i x[n-i]$$

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

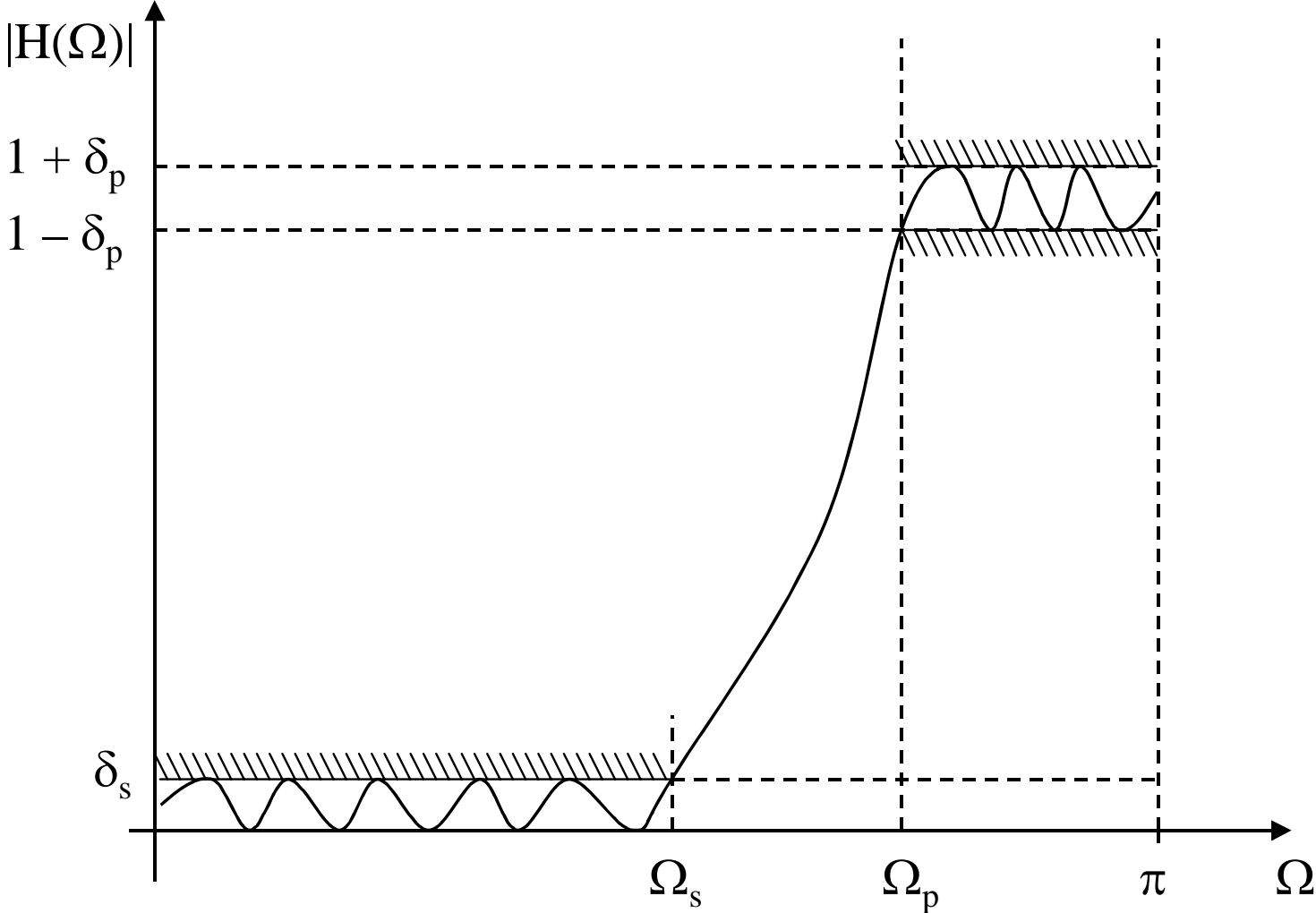
The command `remezord` will give an estimate of the FIR filter order needed to achieve given specifications. The routine `remez.m` will then give the required design.

Here is how we state the filter specifications:

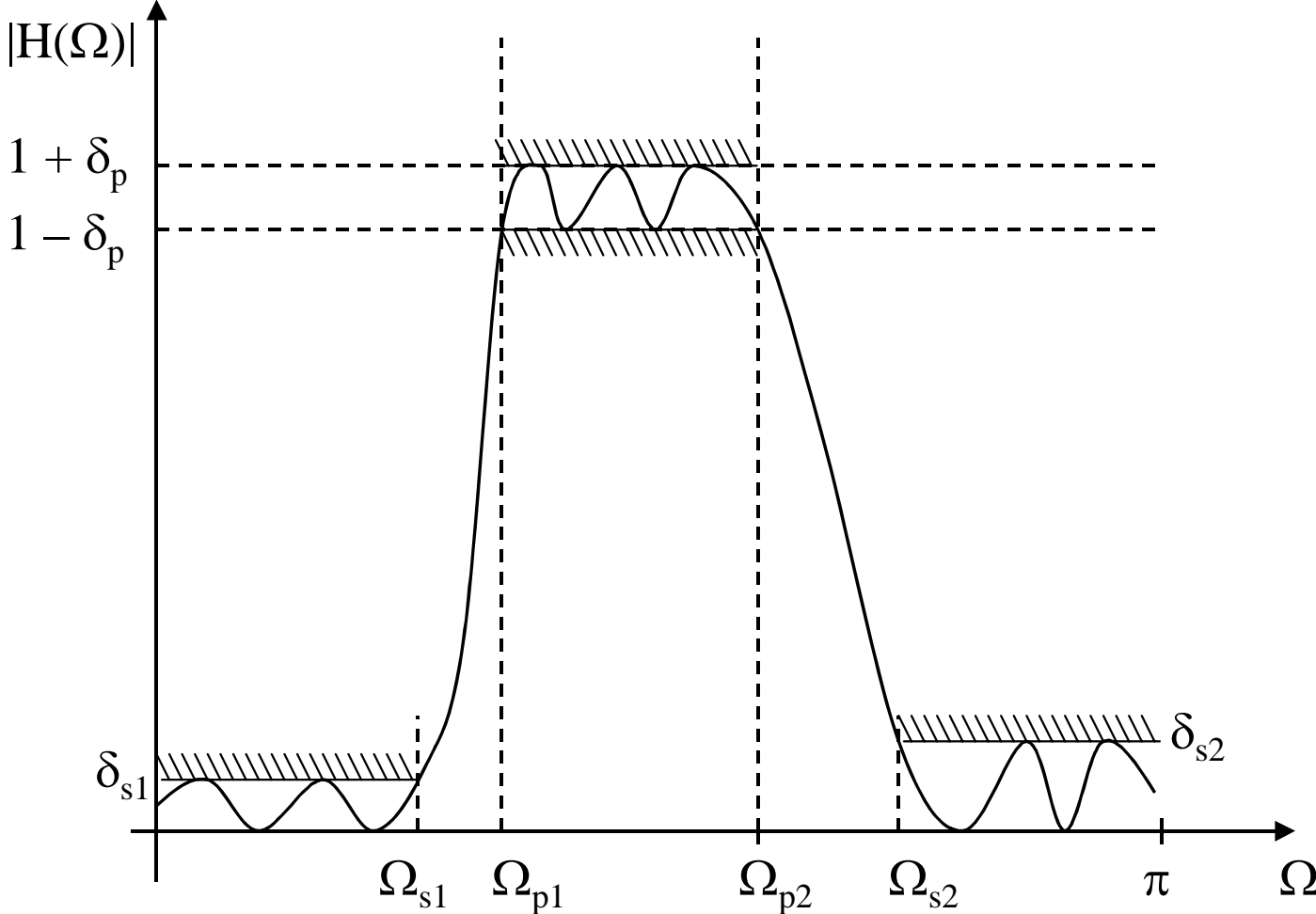
Lowpass Filter Specification



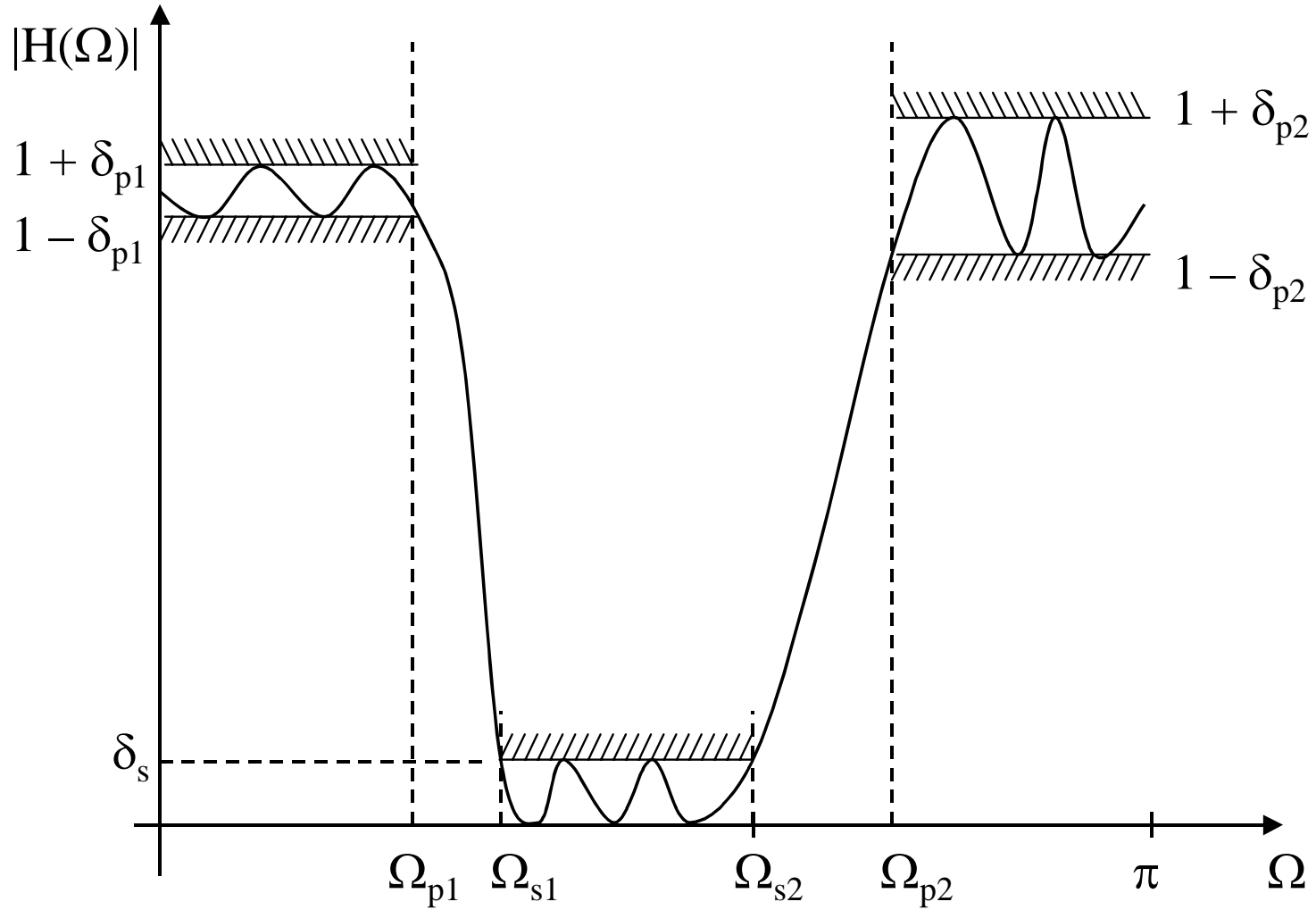
Highpass Filter Specification



Bandpass Filter Specification



Bandstop Filter Specification



III. Lowpass Filter Design Steps

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

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:

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

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


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

$$\begin{aligned} H(z) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_N z^{-N} \\ &= \frac{b_0 z^N + b_1 z^{N-1} + b_2 z^{N-2} + \dots + b_N}{z^N} \end{aligned}$$

III. Lowpass Filter Design

Lowpass Filter Design Specifications:

- Passband cutoff frequency = 7 kHz
- Stopband cutoff frequency = 9 kHz
- Sampling Frequency = 44.1 kHz (frequencies of interest 0 to 22.05 kHz)
- At least 60 dB of stopband attenuation
- No more than 1 dB passband ripple

```
rp=1; rs=60; % specify passband ripple & stopband attenuation in dB
f_spec=[7000 9000]; % specify passband and stopband edges in Hz
AA=[1 0]; %%% specfies that you want a lowpass filter
dev=[(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)]; % parm. needed by design routine
Fs=44.1e3;

[N,fo,ao,w]=remezord(f_spec,AA,dev,Fs) % estimates filter order and gives other parms

b=remez(N,fo,ao,w); % Computes the designed filter coefficients in vector b
[H,ff]=freqz(b,1,1024,Fs); % Compute the frequency response
```

```

figure; stem(0:N,b)           % Plots filter's impulse response

figure;
subplot(2,1,1); plot(ff,20*log10(abs(H)))    % Plot magnitude in dB
subplot(2,1,2); plot(ff,unwrap(angle(H)))    % Plot unwrapped angle in radians

figure
zeros=roots(b);           % compute roots of transfer function (i.e. compute zeros)

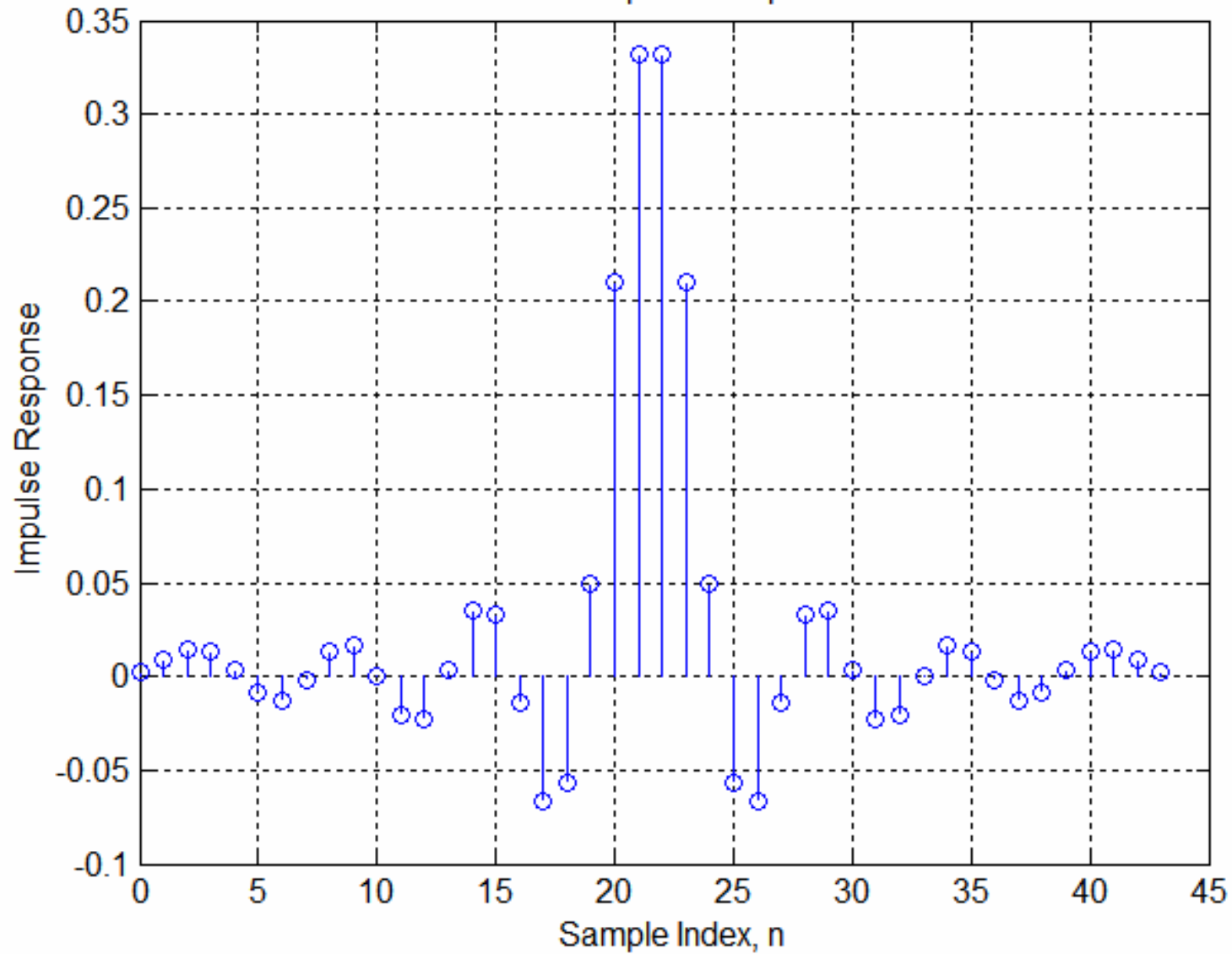
plot(real(zeros), imag(zeros),'o') % Plot zeros in complex z plane
hold on
plot(0,0,'x')

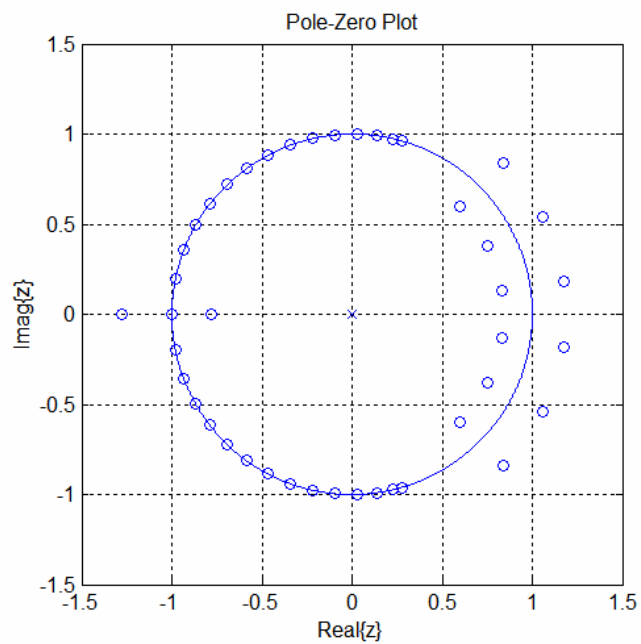
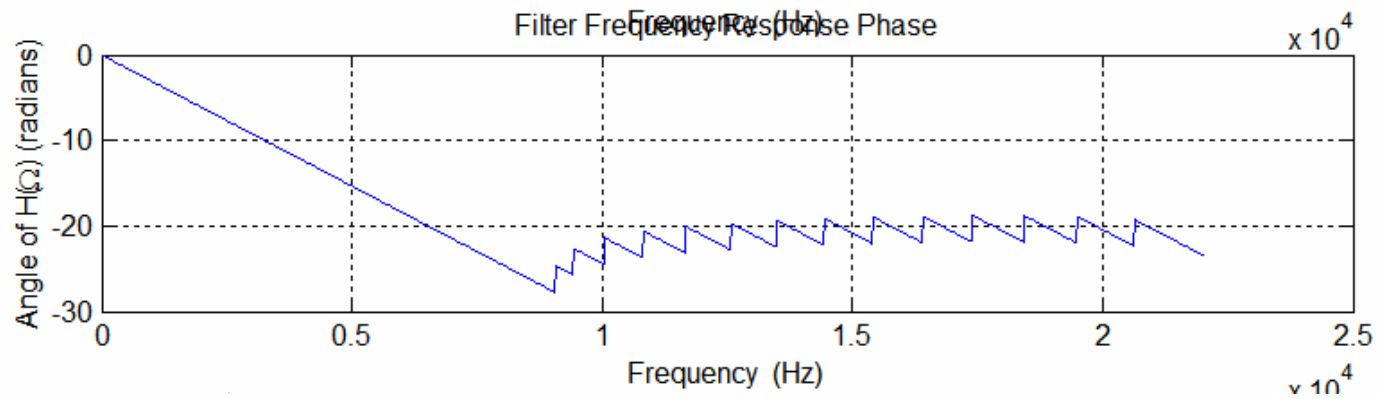
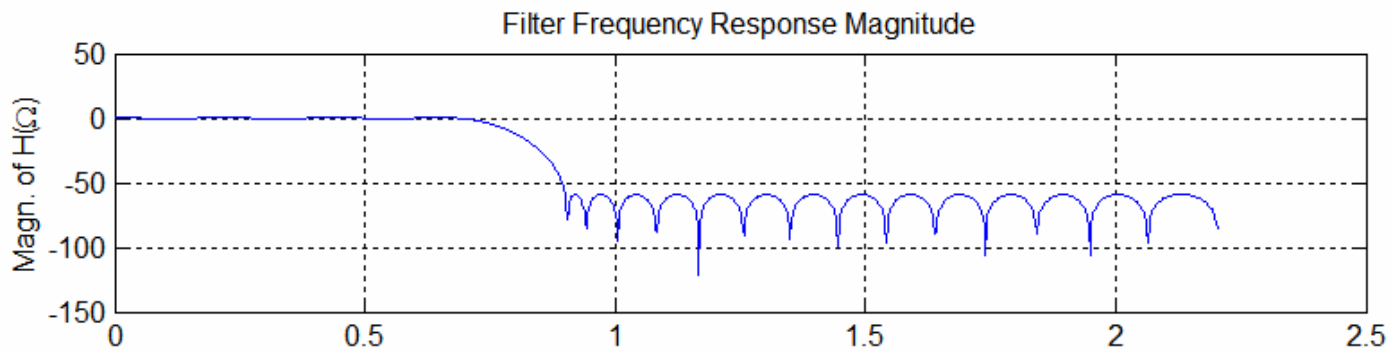
theta=linspace(-pi,pi,10000);
plot(cos(theta),sin(theta)) % Put unit circle on the plot

axis([-1.5 1.5 -1.5 1.5])
axis square

```

Filter's Impulse Response





IV. Remove Interference with Filter

1. Use the designed filter to remove the interference

- Filter x_10 using the LPF to get x_10_out

$y = \text{filter}(b,a,x)$ 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:

$$\sum_{i=0}^{N_a} a_i y[n-i] = \sum_{i=0}^{N_b} b_i x[n-i] \quad a = [a_0 \ a_1 \ a_2 \ \dots \ a_{N_a}]$$
$$b = [b_0 \ b_1 \ b_2 \ \dots \ b_{N_b}]$$

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

$$y[n] = \sum_{i=0}^N b_i x[n-i] \quad \text{so the “a vector” is } a = [a_0] = 1$$

2. Assess the performance of the filter:

- Compare x_10_out, x_10, and x in the frequency domain.
- Compare x_10_out, x_10, and x in the time domain.
- Listen to the filtered guitar signal using MATLAB’s sound command.

IV. Remove Interference w/ Filter

```
x_10_out=filter(b,1,x_10); %%% filter the signal with the designed filter
```

```
X_10_out_1=fftshift(fft(x_10_out(20000+(1:16384)),65536));
```

```
figure
```

```
subplot(3,1,1); plot(f/1e3,20*log10(abs(X_10_1))); title('DFT of Signal w/ Interference')
```

```
subplot(3,1,2); plot(f/1e3,20*log10(abs(X_10_out_1))); title('DFT of Filtered Signal')
```

```
subplot(3,1,3); plot(f/1e3,20*log10(abs(X1))); title('DFT of Original Signal')
```

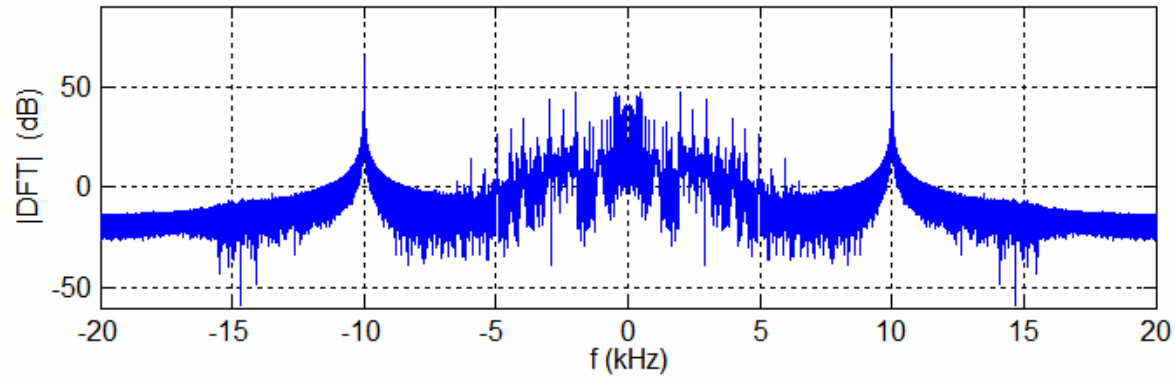
```
figure
```

```
subplot(3,1,1); plot(t,x_10(1:50000),'r'); title('Signal w/ Interference')
```

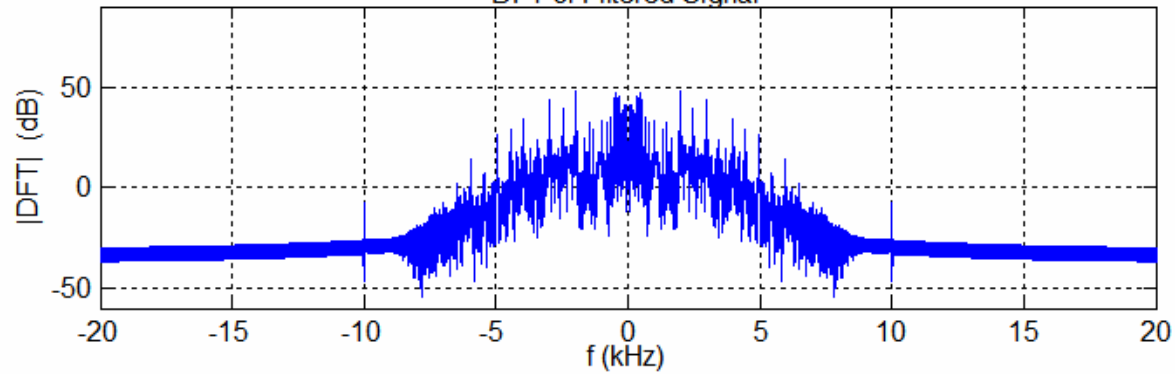
```
subplot(3,1,2); plot(t,x(1:50000),'b',t,x_10_out(1:50000),'m--');  
title('Filtered Signal and Original')
```

```
subplot(3,1,3) %%% Make a plot that accounts for the delay in the filtered signal  
%% For and odd filter order N the delay is (N-1)/2  
plot(t,x(1:50000),'b',t,x_10_out(22+(1:50000)), 'm--')
```

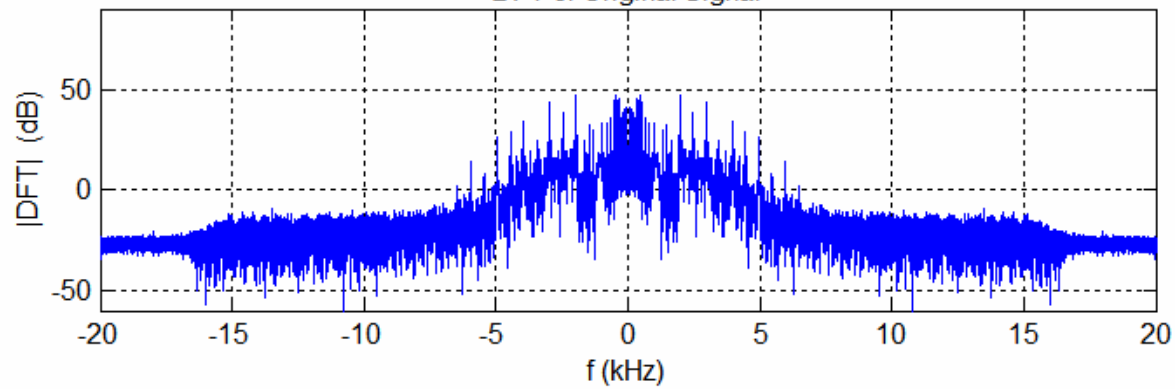
DFT of Signal w/ Interference



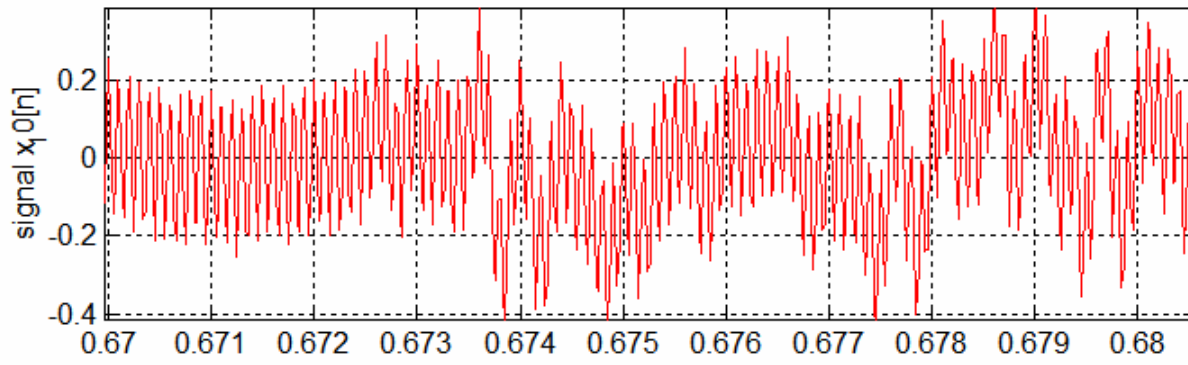
DFT of Filtered Signal



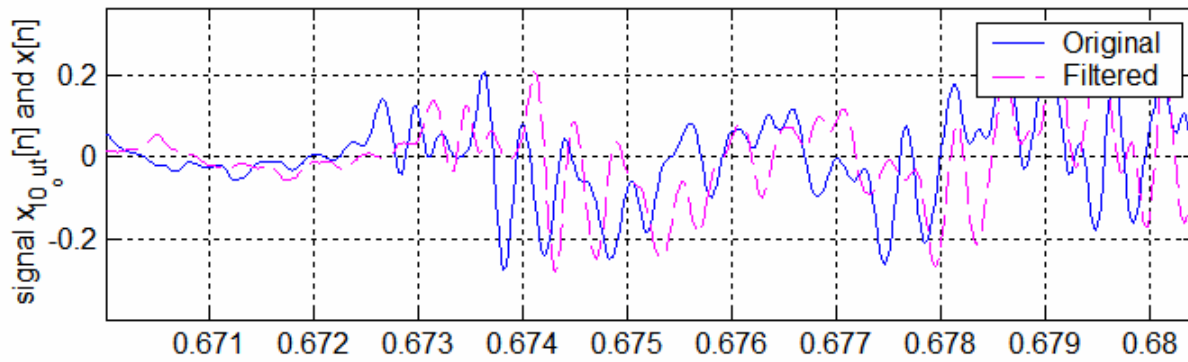
DFT of Original Signal



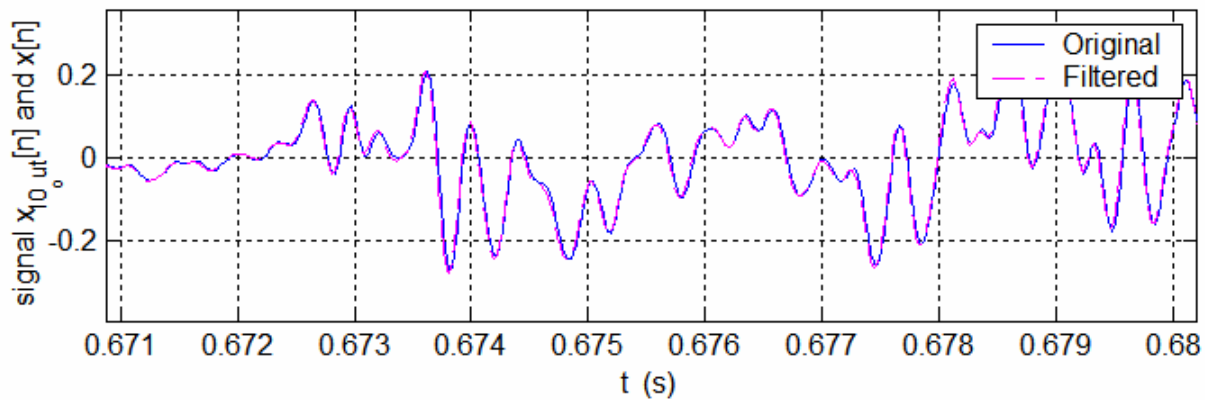
Signal w/ Interference



Filtered Signal and Original



Filtered Signal (delay corrected) and Original



V. Repeat for a Midrange Interferer

Suppose the interference is a 3000 Hz sinusoid. Now you can't simply design a lowpass filter because it would filter out the guitar frequencies above 3000 Hz.

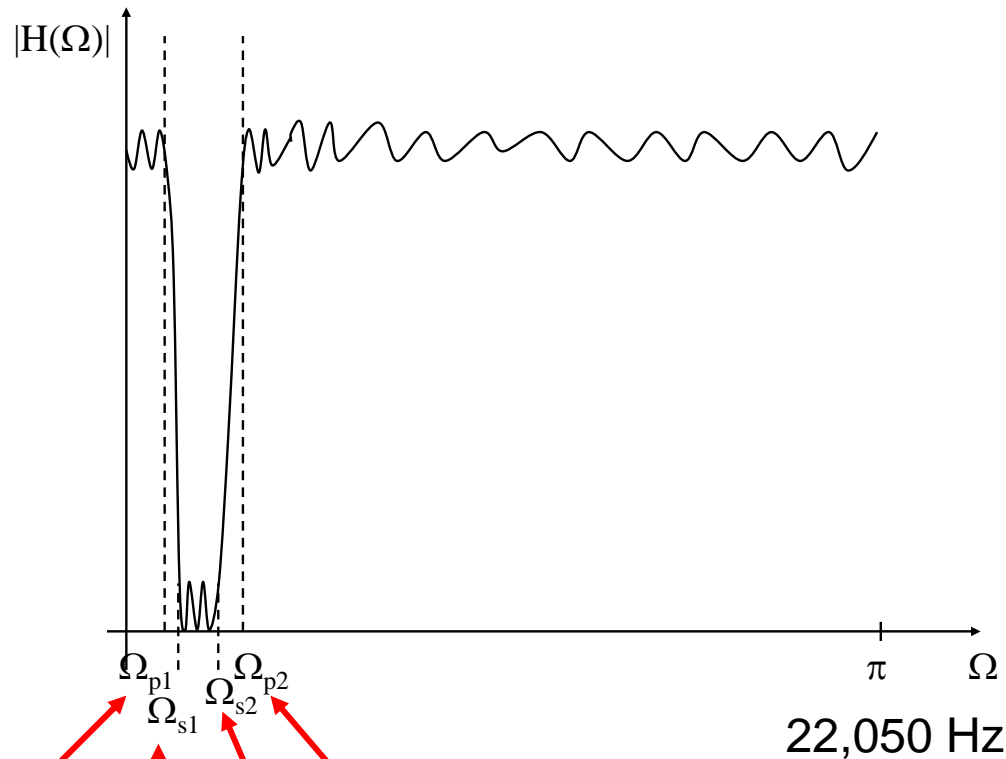
1. As a test, change the lowpass filter design above to have a passband cutoff of 2500 and a stopband cutoff of 2900 and apply the filter to the original (interference-free) guitar signal.
2. Compare the spectrum of this filter's output to that of the original signal
3. Listen to this filter's output.
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 design filter.

```
rp=1; rs=60; % ripple, stopband level
```

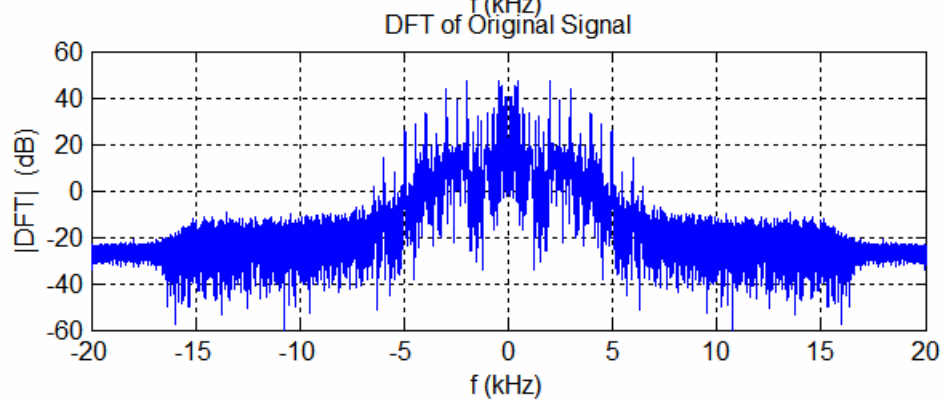
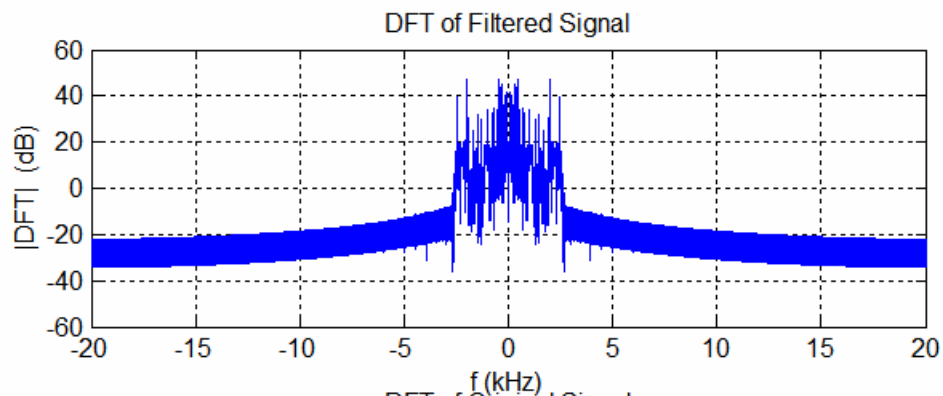
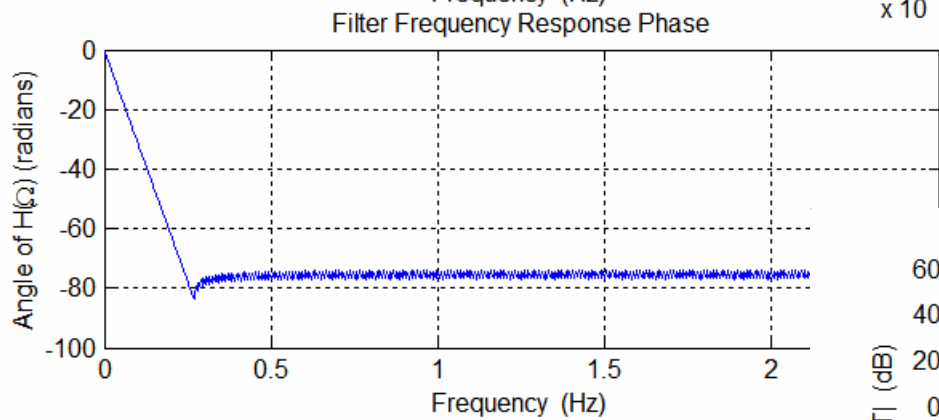
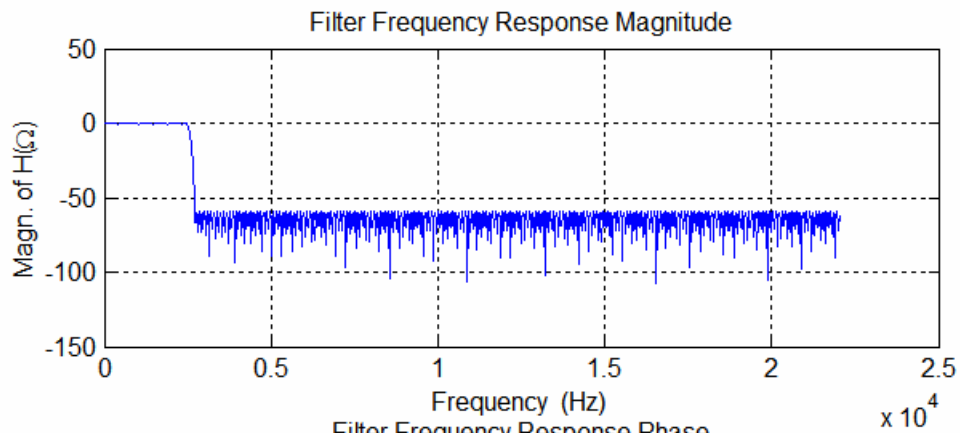
```
f_spec=[2800 2990 3010 3200]; % band edges in Hz
```

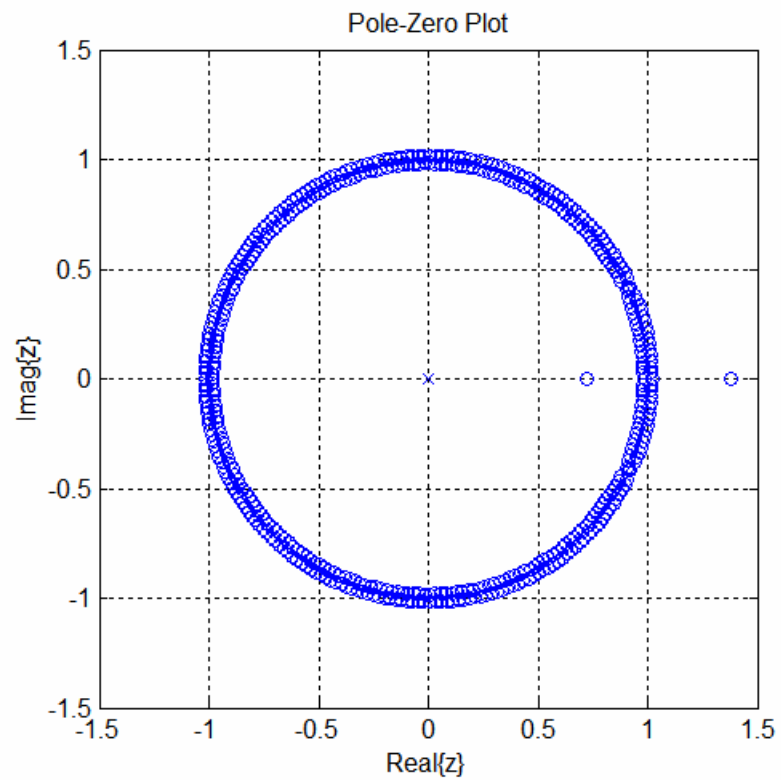
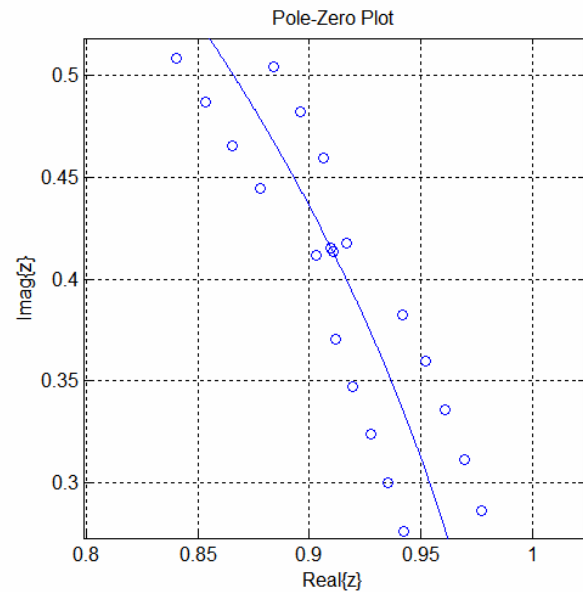
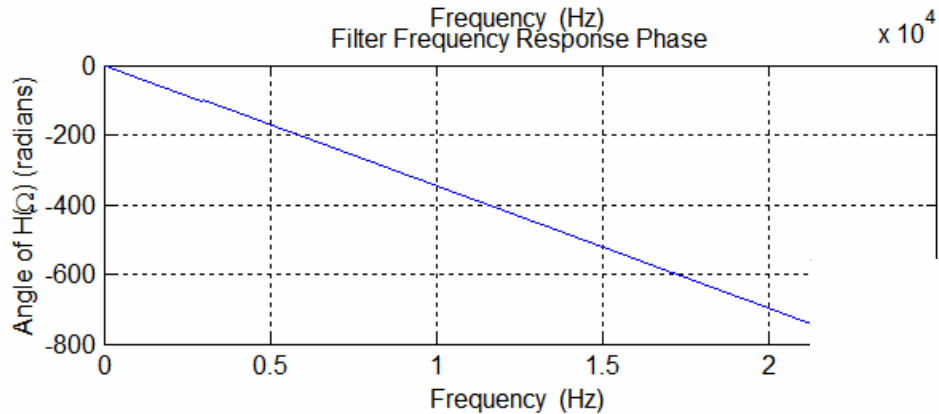
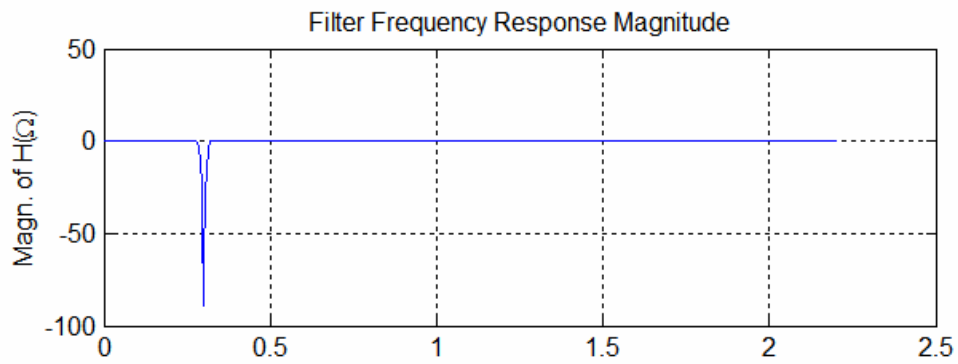
```
AA=[1 0 1]; %%% stipulates "bandstop"
```

Bandstop Filter Specification

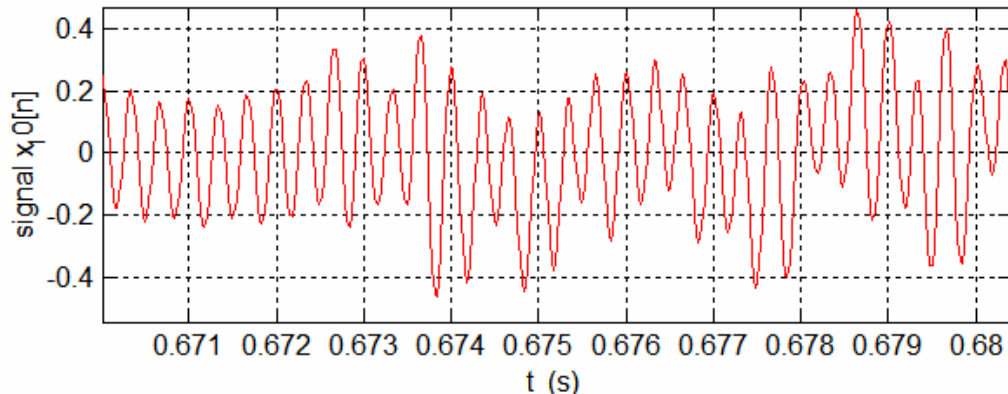


```
f_spec=[2800 2990 3010 3200]; % band edges in Hz
```

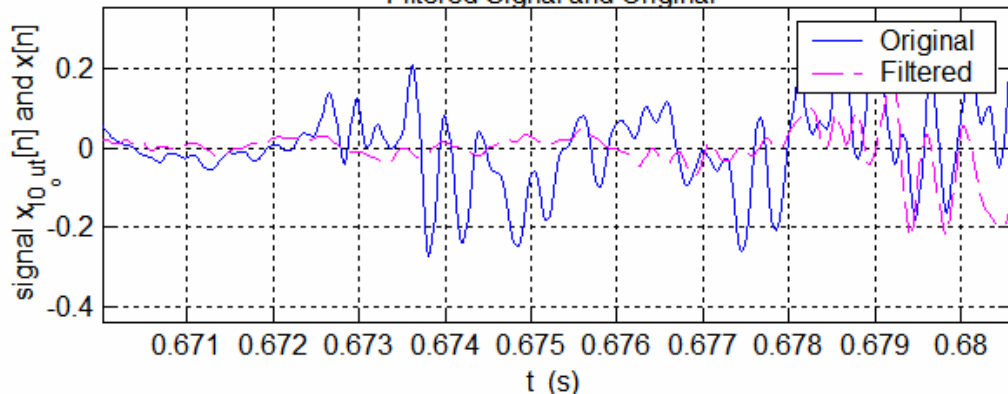




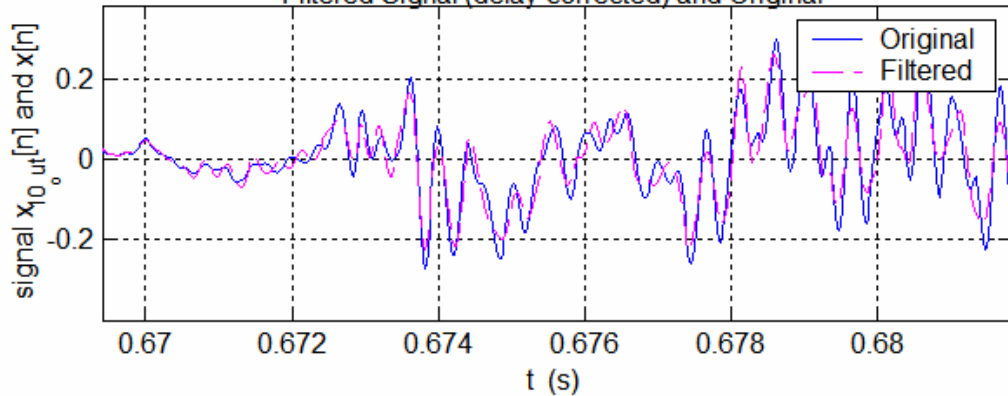
Signal w/ Interference



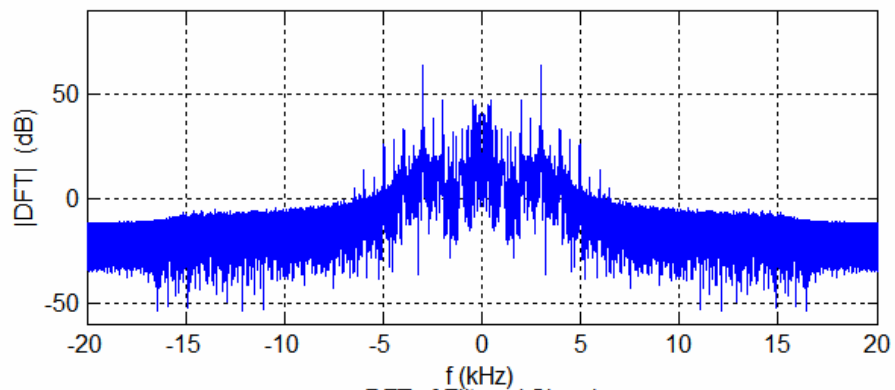
Filtered Signal and Original



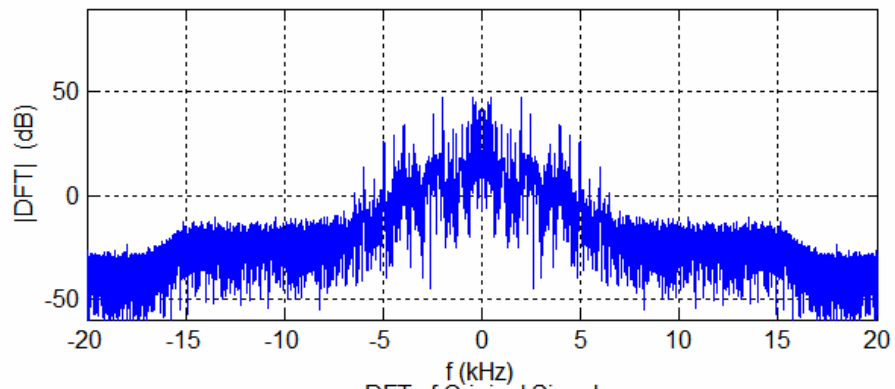
Filtered Signal (delay-corrected) and Original



DFT of Signal w/ Interference



DFT of Filtered Signal



DFT of Original Signal

