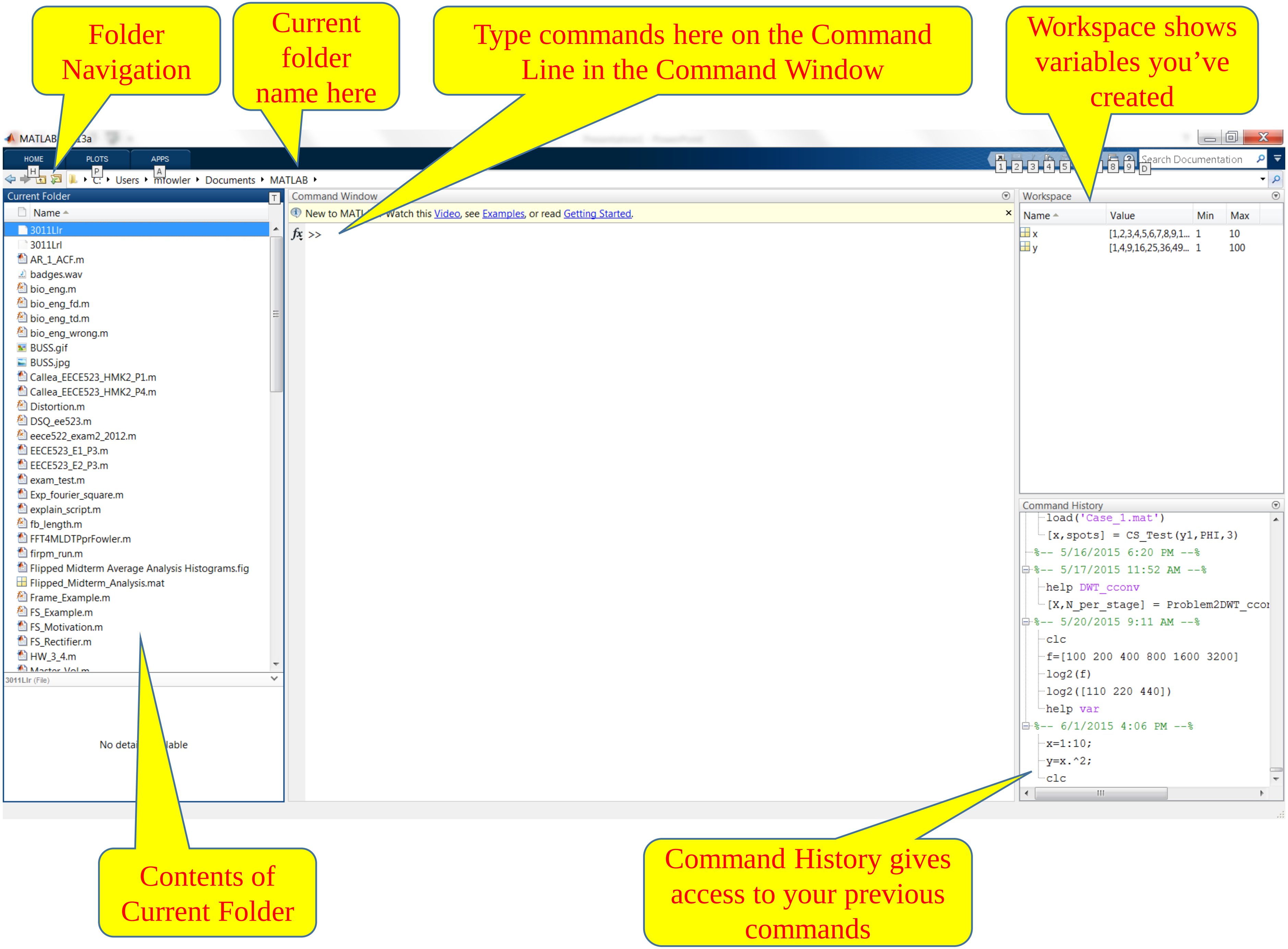
BINGHAMTON UNIVERSITY
State University of New York

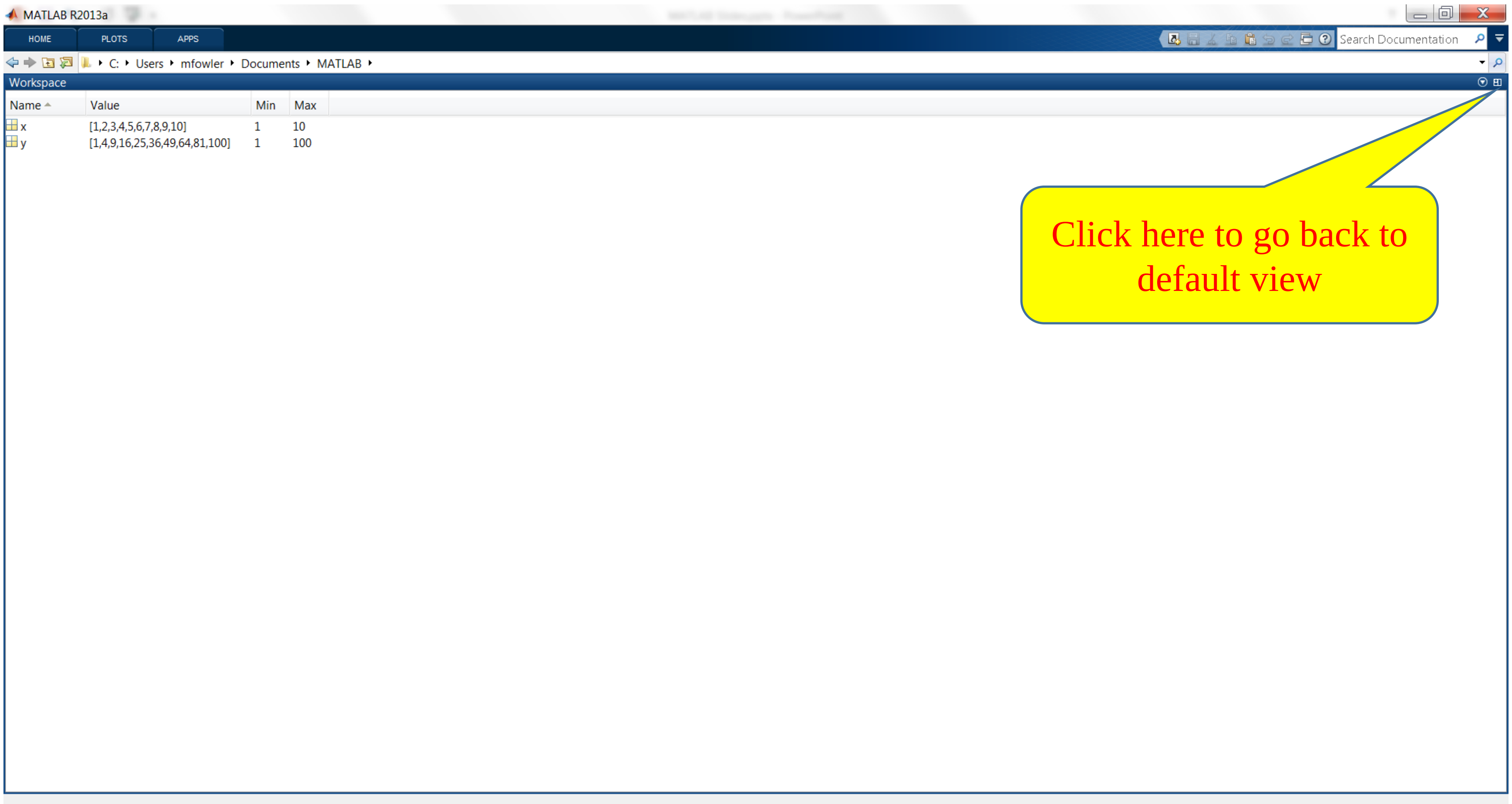
EEO 401
Digital Signal Processing

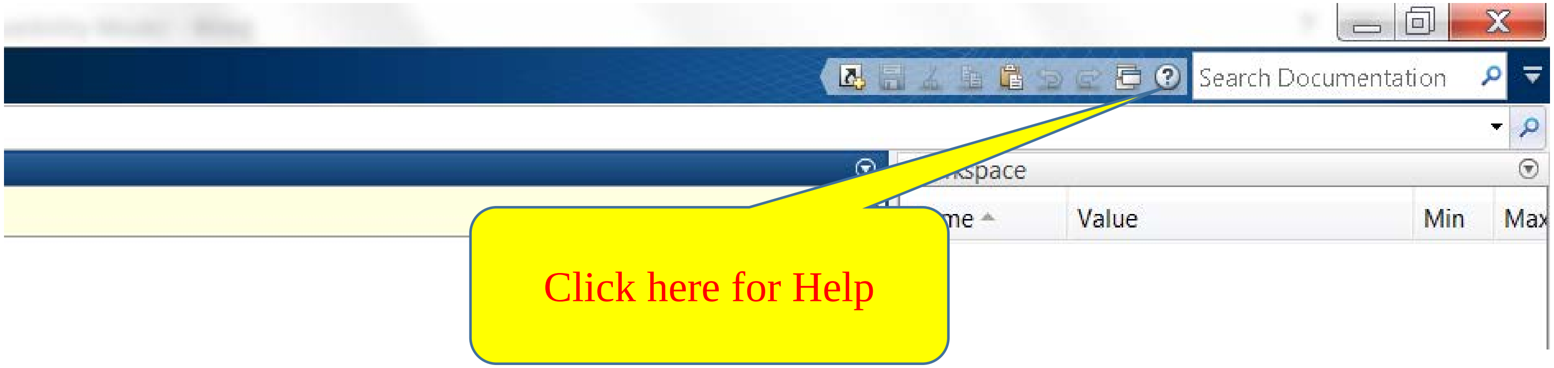
Prof. Mark Fowler

Note Set #17.5

- MATLAB Examples
- Reading Assignment: MATLAB Tutorial on Course Webpage


Folder Navigation
Current folder name here
Type commands here on the Command Line in the Command Window
Workspace shows variables you've created
Contents of Current Folder
Command History gives access to your previous commands
Current Folder
Command Window
Workspace
Command History
fx >>

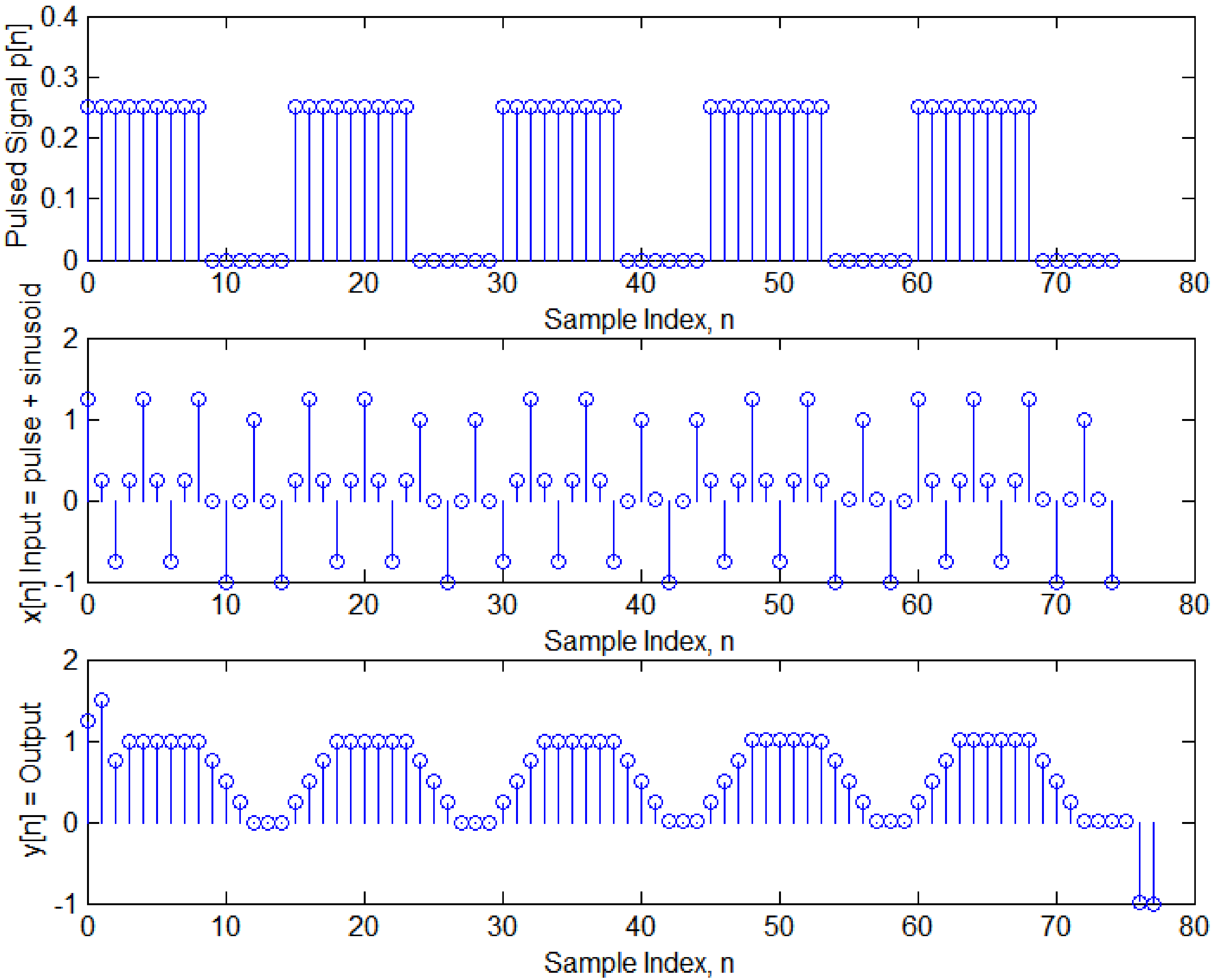

MATLAB R2013a
HOME
PLOTS
APPS
Search Documentation
C: ▸ Users ▸ mfowler ▸ Documents ▸ MATLAB ▸
Workspace
Name
Value
Min
Max
x
[1,2,3,4,5,6,7,8,9,10]
1
10
y
[1,4,9,16,25,36,49,64,81,100]
1
100
Click here to go back to default view


Search Documentation
rkspace
me
Value
Min
Max
Click here for Help

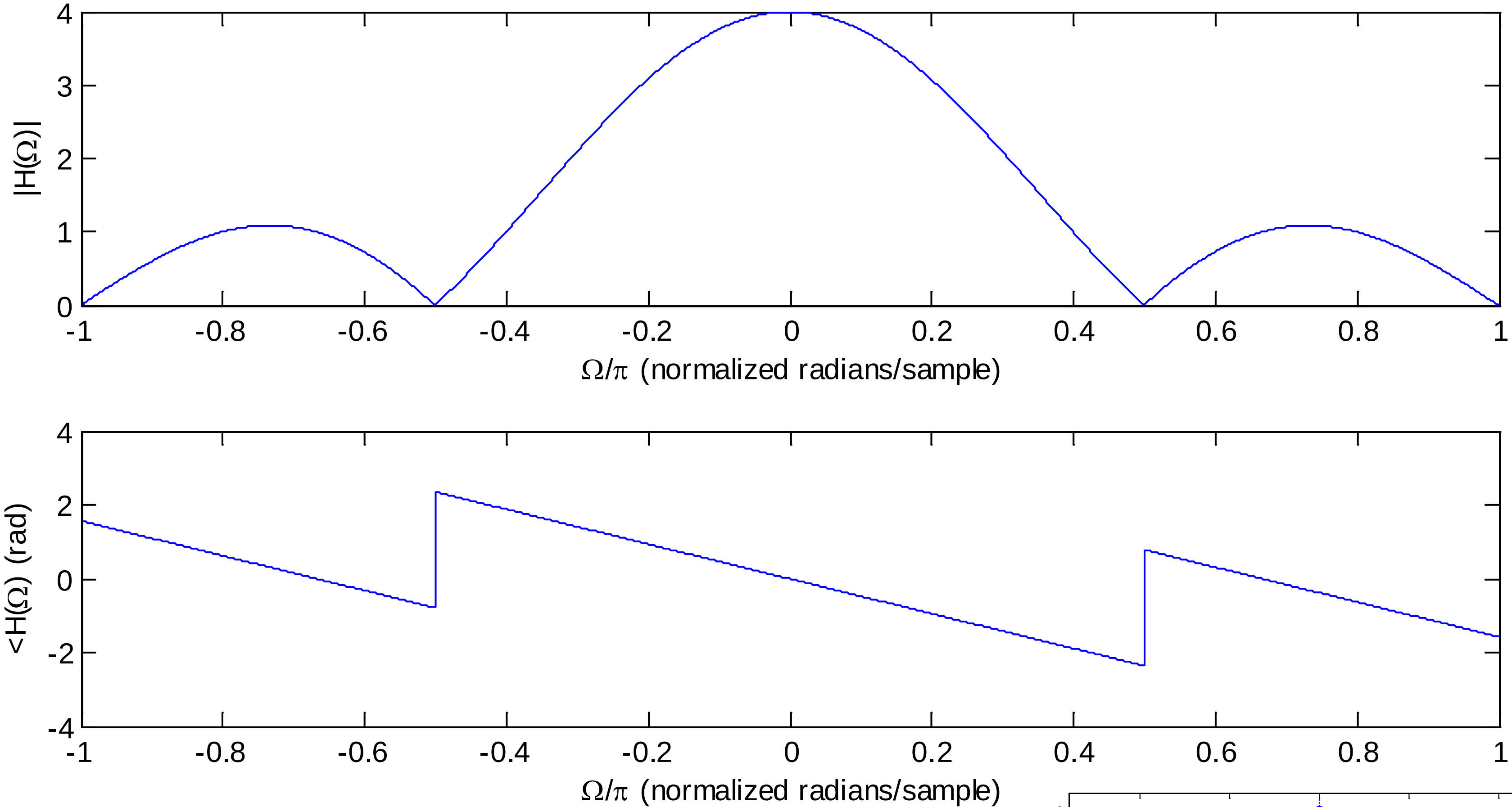
MATLAB Example – DT Convolution

DT_conv_example.m

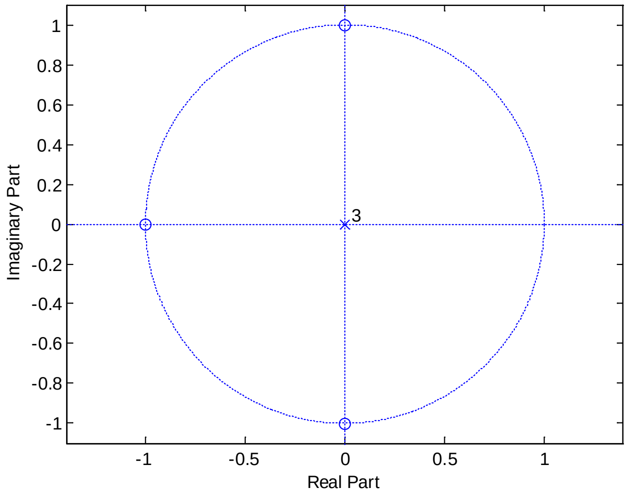
```
%%% Matlab exploration for Pulses with Interfering Sinusoid

p=[ones(1,9) zeros(1,6)];   %%% Create one pulse and zeros
p=[p p p p p];   %%% stack 5 of them together
p=0.25*p;   %%% adjust its amplitude to be 0.25
subplot(3,1,1)
stem(0:74,p)   %%% look at the sequence of pulses
xlabel('Sample Index, n')
ylabel('Pulsed Signal p[n]')
x=p+cos((pi/2)*(0:74));  % add in an interfering sinusoid
subplot(3,1,2)
stem(0:74,x)
xlabel('Sample Index, n')
ylabel('x[n] Input = pulse + sinusoid')
h = ones(1,4);   %%% define impulse response of filter
y=conv(x,h);   %% filter out sinusoid with DT Conv.
subplot(3,1,3)
stem(0:77,y)
xlabel('Sample Index, n')
ylabel('y[n] = Output')
%%%  Note that pulses are free of sinusoidal interference but have been "smoothed"
```



Pulsed Signal p[n]
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
Sample Index, n
x[n] Input = pulse + sinusoid
2
1
0
-1
0
10
20
30
40
50
60
70
80
Sample Index, n
y[n] = Output
2
1
0
-1
0
10
20
30
40
50
60
70
80
Sample Index, n




```
x=p+cos((pi/2)*(0:74));
h = ones(1,4);

w=-pi:0.001:pi;
H=freqz(h,1,w);
zplane(h,1)
```



MATLAB Trick: Create Frequency Vector for DFT Plotting

When computing the DFT (using fft) you get N numbers that tell the *values* the DFT coefficients have. But you need to know what frequencies they are at…

We'll assume that you are using fftshift, which moves the DFT coefficients around so they lie in the frequency range -π to π

To plot versus Ω in rad/sample:

- For N DFT points… the frequency spacing between them is $2\pi/N$
- With fftshift, the frequencies start at -π
- Thus the command that makes these frequency points is

```
omega = ( (-N/2):( (N/2) -1 ) )*2*pi/N
```

To plot versus *f* in Hz:

- For N DFT points… the frequency spacing between them is F_s/N
- With fftshift, the frequencies start at $-F_s/2$
- Thus the command that makes these frequency points is

```
f = ( (-N/2):( (N/2) -1 ) )*Fs/N
```

8 points
Starts at pi
Stops "just shy of pi"

Example for our N=8 case: omega = (-4:3)*2*pi/8

gives the vector [-pi -3pi/4 -pi/2 –pi/4 0 pi/4 pi/2 3pi/4]

MATLAB Demo: FIR Filter Design & Application

Imagine you are in a recording studio and recorded what you feel is a “perfect take” of a guitar solo

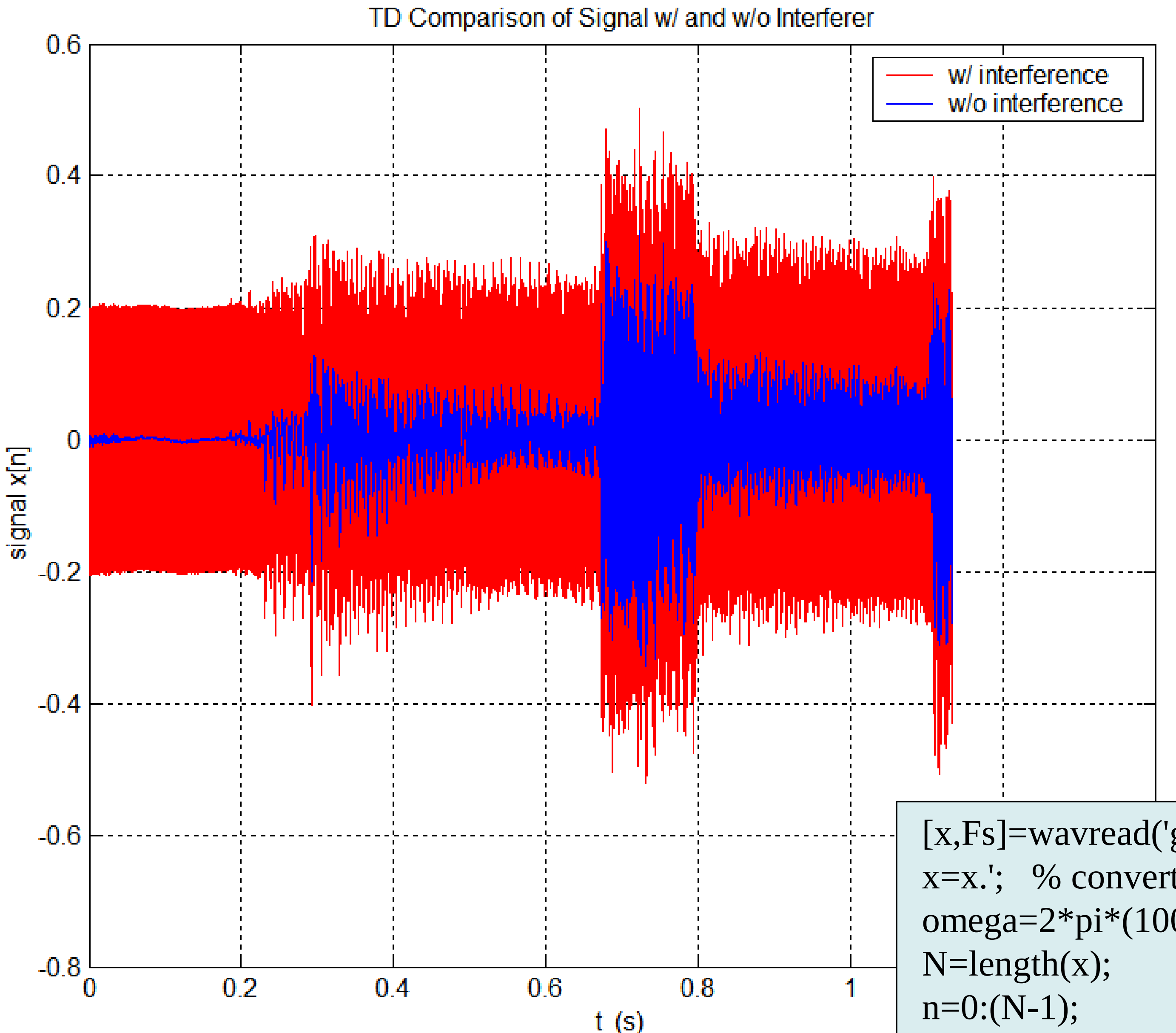
FIR_Filter_Demo.m

But some nearby electronic device caused sinusoidal EM 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.

To explore this we’ll SIMULATE it in MATLAB!!

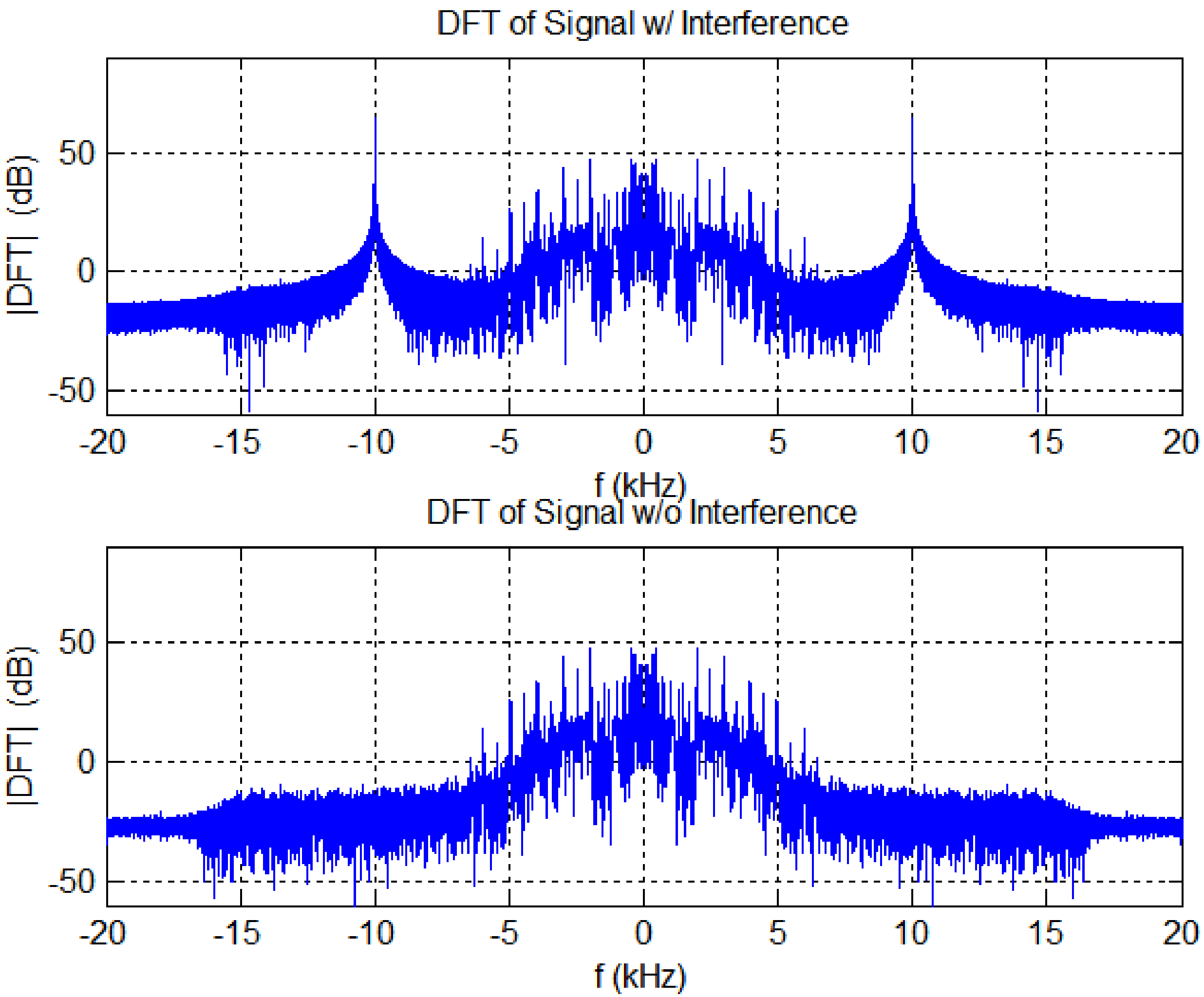
Assume the sinusoid has frequency of 10 kHz




```
[x,Fs]=wavread('guitar1.wav');
x=x.';   % convert into row vector
omega=2*pi*(10000/Fs);
N=length(x);
n=0:(N-1);
x_10=x+cos(omega*n);
t=(0:49999)*(1/Fs);
plot(t,x_10(1:50000),'r',t,x(1:50000))
```

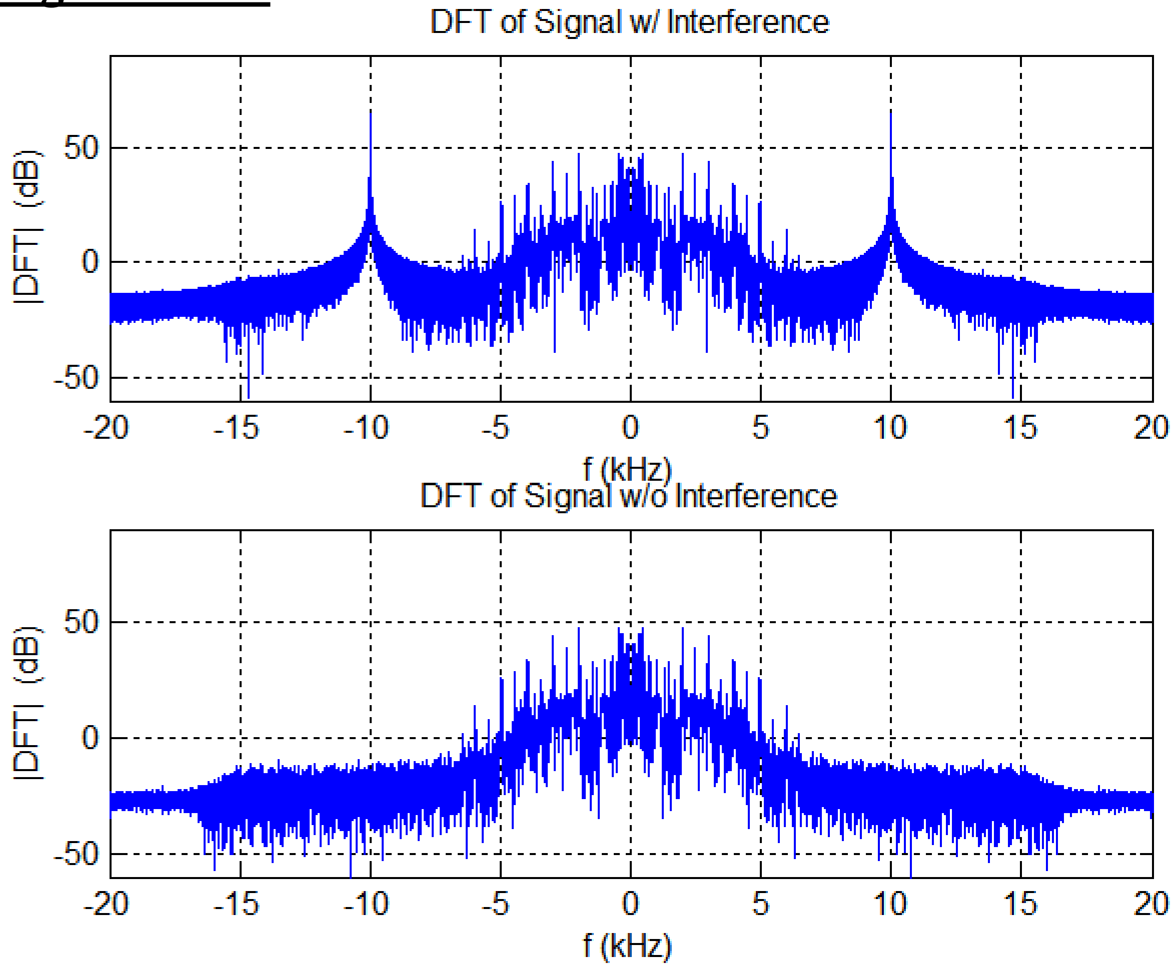
Now… look at DFT to see impact in frequency domain:

```
X=fftshift(fft(x(20000+(1:16384)),65536));
X_10=fftshift(fft(x_10(20000+(1:16384)),65536));
f=(-32768:32767)*Fs/65536;
subplot(2,1,1); plot(f/1e3,20*log10(abs(X_10)));
subplot(2,1,2); plot(f/1e3,20*log10(abs(X)));
```

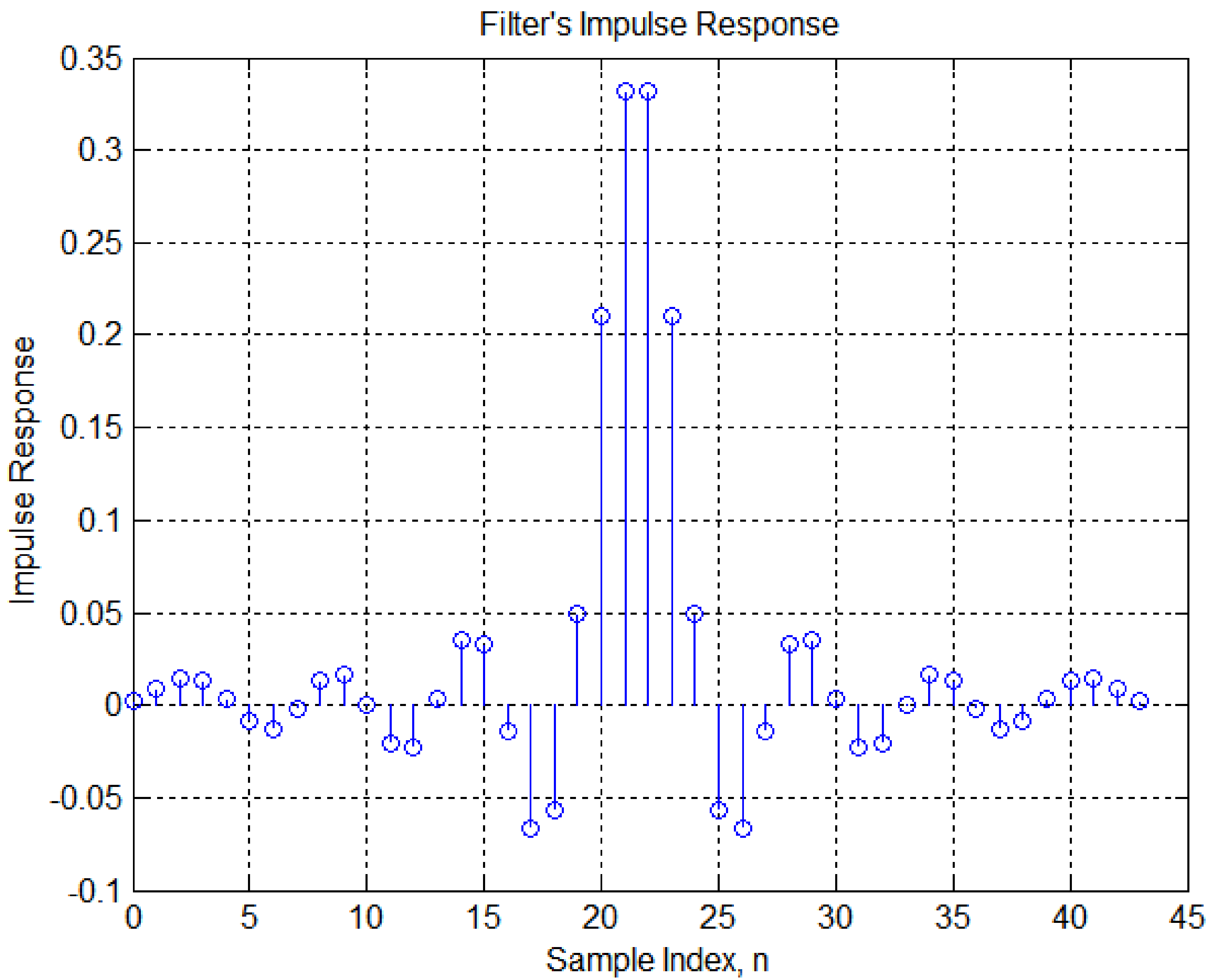


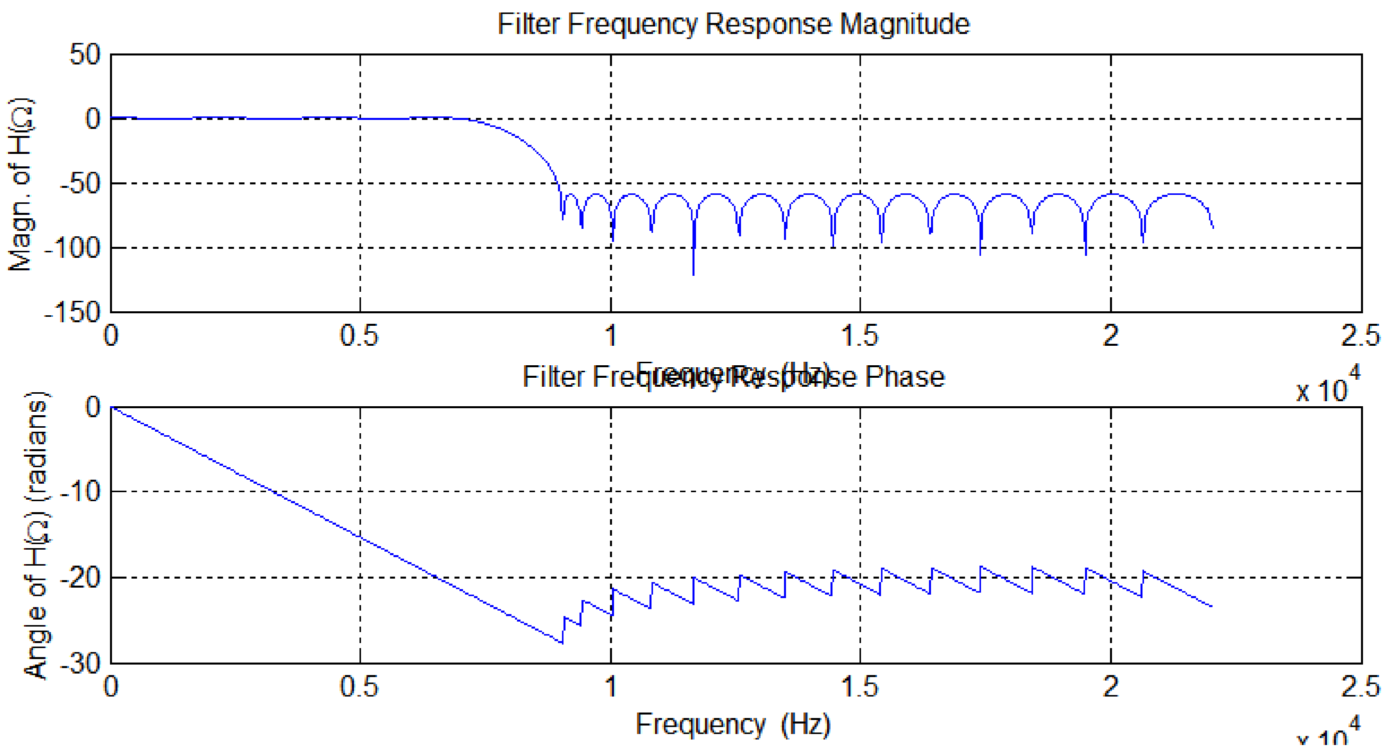
Use the “firpmord” and “firpm” commands to design **lowpass** filter to get:

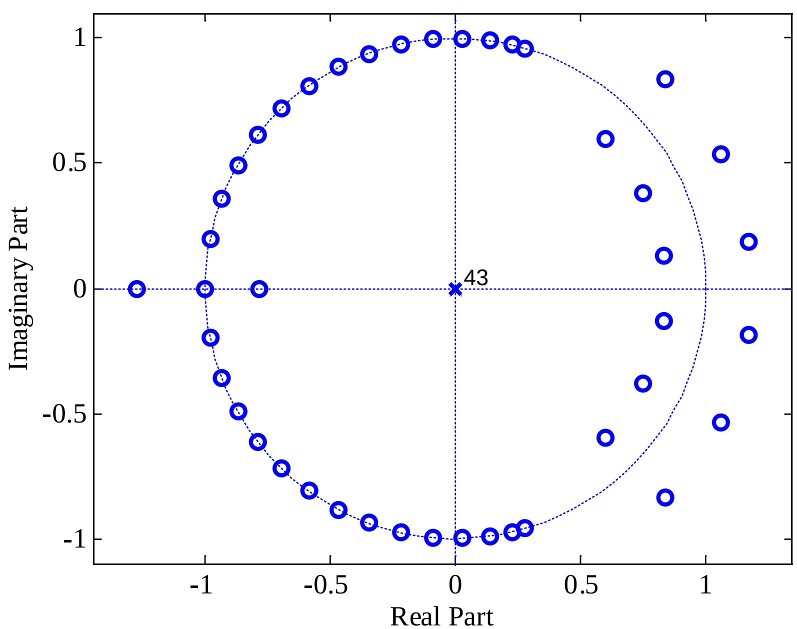
- **<u>60 dB of attenuation</u>** in the stopband for the undesired signal
- **<u>1 dB of passband ripple</u>**
- **<u>passband edge at 7kHz</u>**
- **<u>stopband edge at 9 kHz</u>**.



```
% Lowpass Filter Design
% ·         Passband & Stopband edges:  7 kHz    &    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]=firpmord(f_spec,AA,dev,Fs)    % estimates filter order and gives other parms

b=firpm(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
zplane(b,1)
```


Filter's Impulse Response
Impulse Response
Sample Index, n
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
5
10
15
20
25
30
35
40
45


Filter Frequency Response Magnitude
Magn. of H(Ω)
50
0
-50
-100
-150
0
0.5
1
1.5
2
2.5
Frequency (Hz)
x 10^4
Filter Frequency Response Phase
Angle of H(Ω) (radians)
0
-10
-20
-30
0
0.5
1
1.5
2
2.5
Frequency (Hz)
x 10^4


Imaginary Part
Real Part
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
43

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 = 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:

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

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

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

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

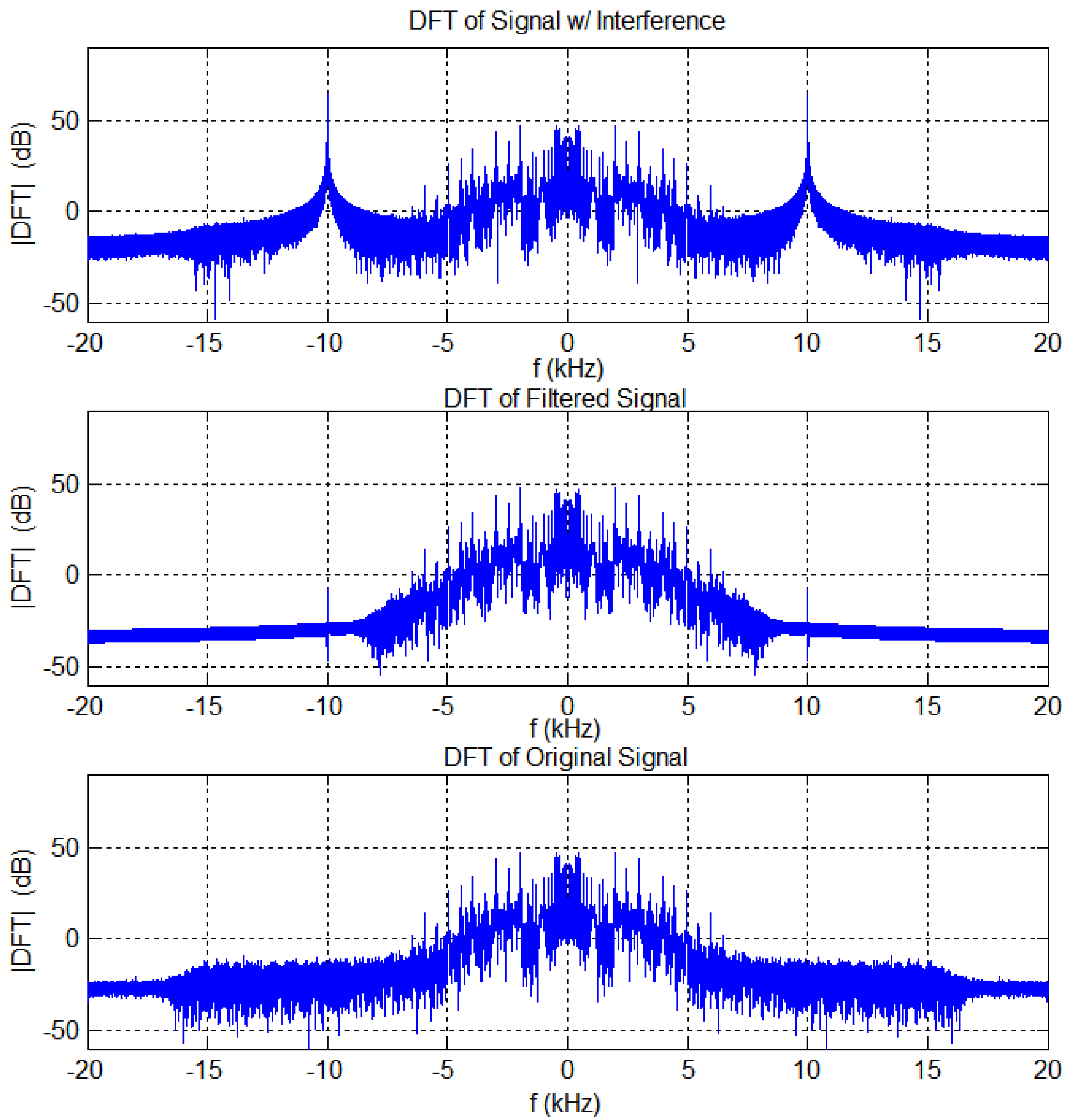
$$y[n] = \sum_{i=0}^{N} b_i x[n-i]$$

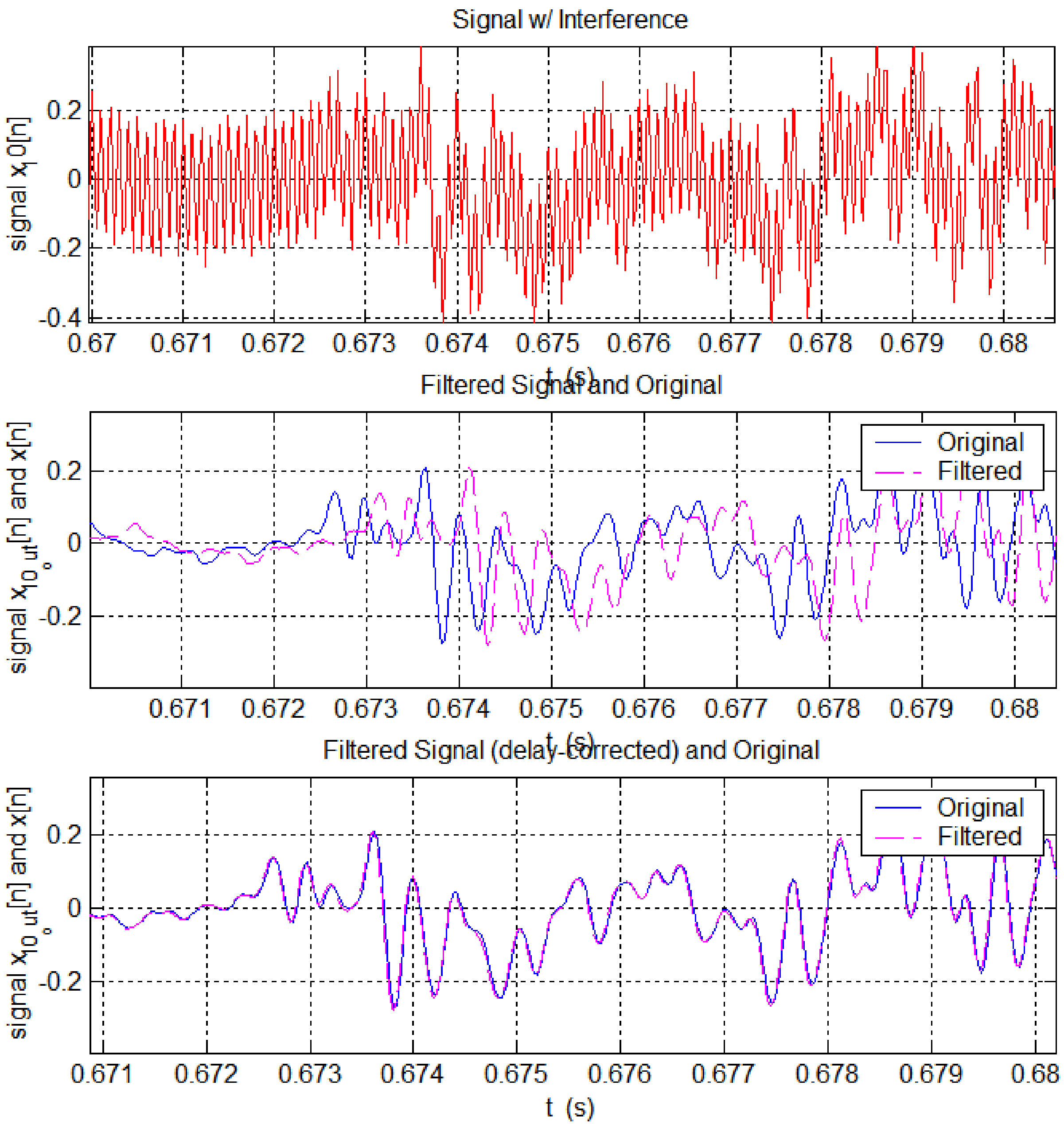
     so the “a vector” is a = [a_0] = 1

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.

Remove Interference w/ Filter

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

X_10_out=fftshift(fft(x_10_out(20000+(1:16384)),65536));

figure
subplot(3,1,1); plot(f/1e3,20*log10(abs(X_10))); title('DFT of Signal w/ Interference')

subplot(3,1,2); plot(f/1e3,20*log10(abs(X_10_out))); title('DFT of Filtered Signal')

subplot(3,1,3); plot(f/1e3,20*log10(abs(X))); 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 an 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| (dB)
50
0
-50
-20
-15
-10
-5
0
5
10
15
20
f (kHz)
DFT of Filtered Signal
|DFT| (dB)
f (kHz)
DFT of Original Signal
|DFT| (dB)
f (kHz)


Signal w/ Interference
0.2
0
-0.2
-0.4
0.67
0.671
0.672
0.673
0.674
0.675
0.676
0.677
0.678
0.679
0.68
t (s)
Filtered Signal and Original
Original
Filtered
0.2
0
-0.2
0.671
0.672
0.673
0.674
0.675
0.676
0.677
0.678
0.679
0.68
t (s)
Filtered Signal (delay-corrected) and Original
Original
Filtered
0.2
0
-0.2
0.671
0.672
0.673
0.674
0.675
0.676
0.677
0.678
0.679
0.68
t (s)

MATLAB Demo: IIR Filter "Design" & Application

Using the same guitar signal…

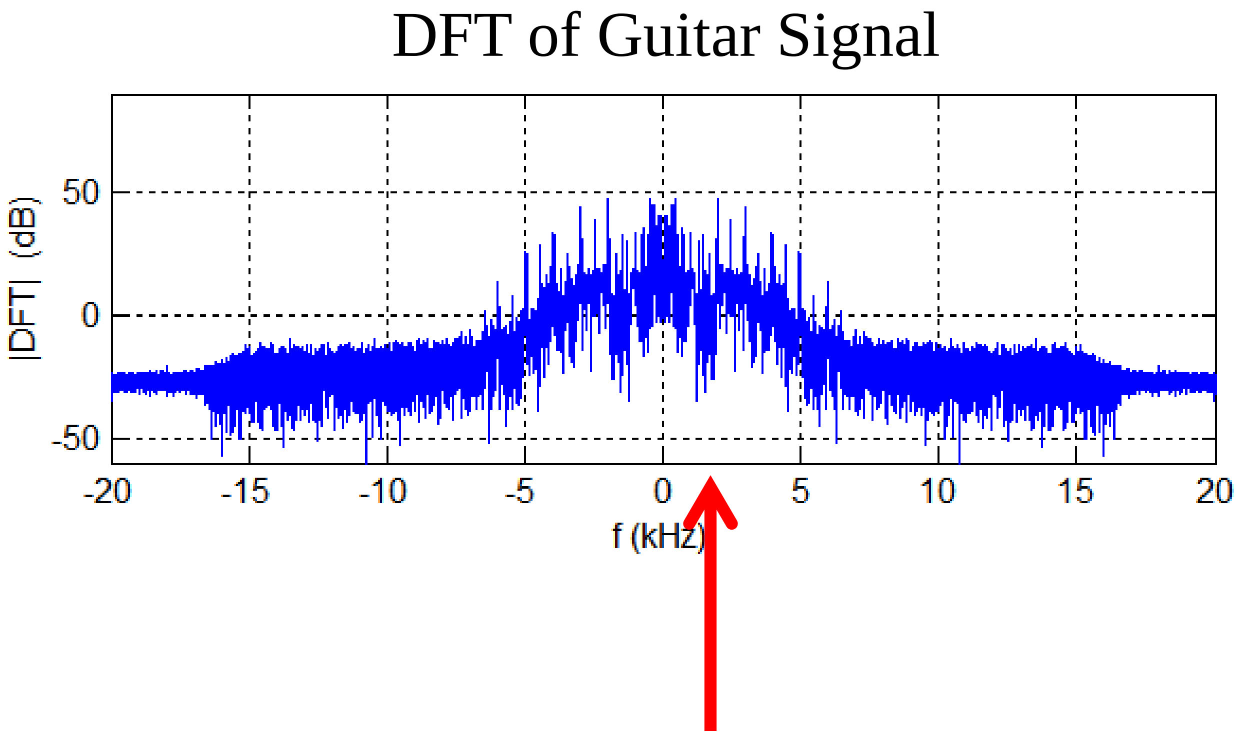
IIR_Filter_Demo.m

Suppose we want to design a DT filter that will emphasize the "middle" audio frequencies in that recording… to get a "different sounding" recording…

We'll explore this in MATLAB!!

Now… look at DFT of guitar signal to see it in frequency domain:

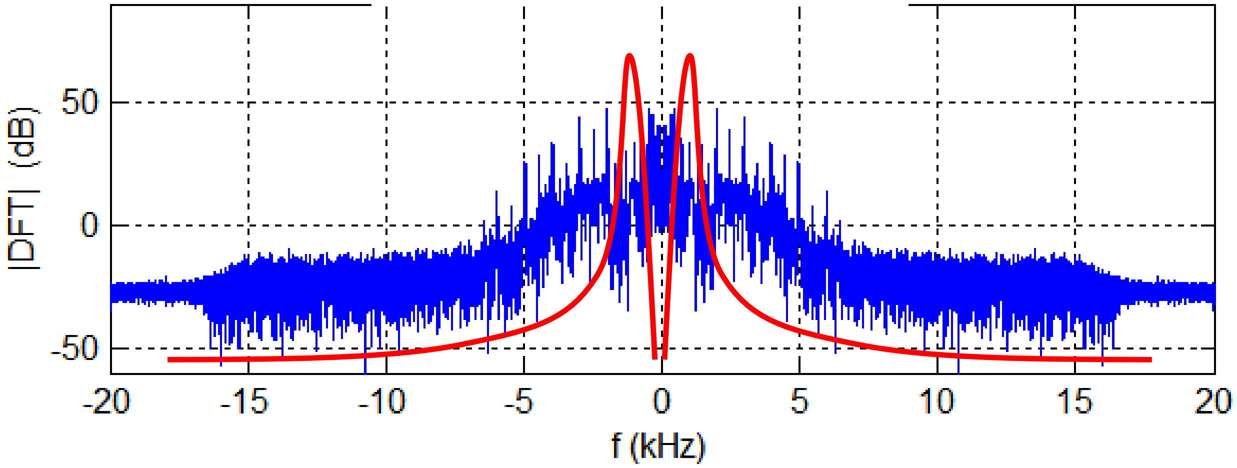
```
X=fftshift(fft(x(20000+(1:16384)),65536));
f=(-32768:32767)*Fs/65536;
plot(f/1e3,20*log10(abs(X)));
```



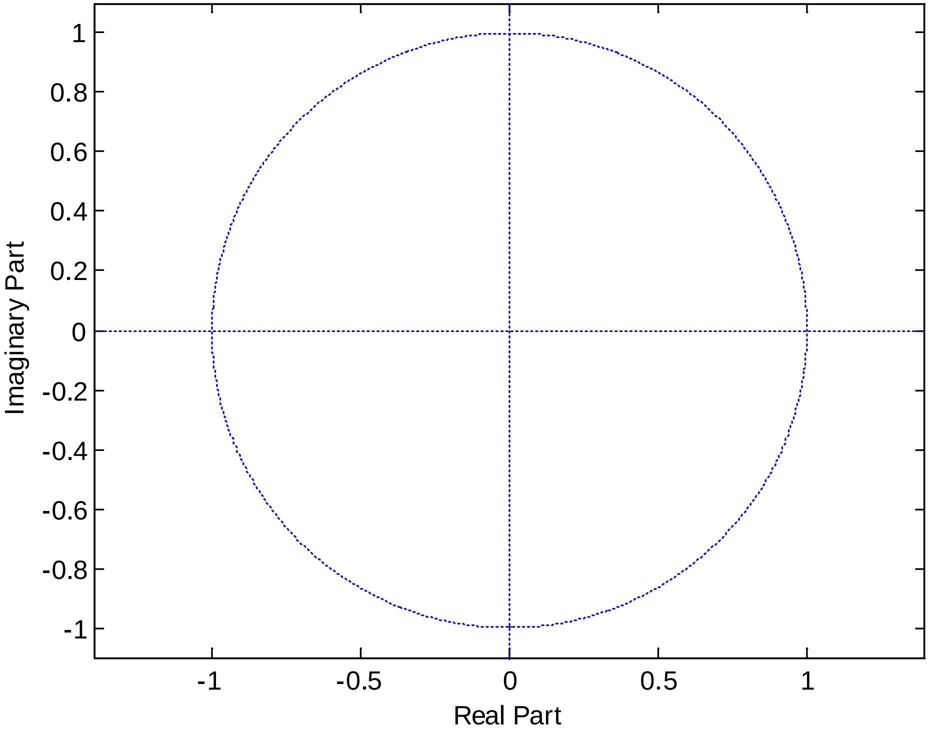
Suppose we decide that 1 kHz is the narrow region we want to emphasize

So… what might our frequency response look like?

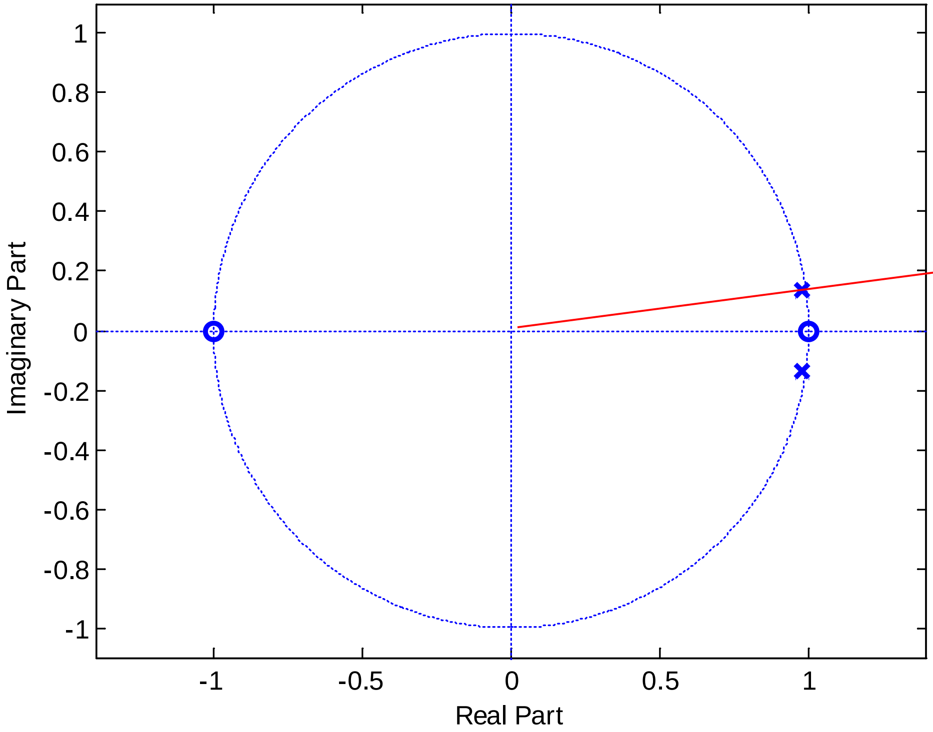
DFT of Guitar Signal



So… what might our pole-zero plot look like?



Maybe this would be good….



Ω Needs to correspond to 1000 Hz

$$\Omega = 1000\frac{2\pi}{F_s} = 1000\frac{2\pi}{44100} \approx 0.14$$

$$H(z) = \frac{(z+1)(z-1)}{(z-0.99e^{j0.14})(z-0.99e^{-j0.14})} \quad = \frac{z^2-1}{z^2-1.9606z+0.9801}$$

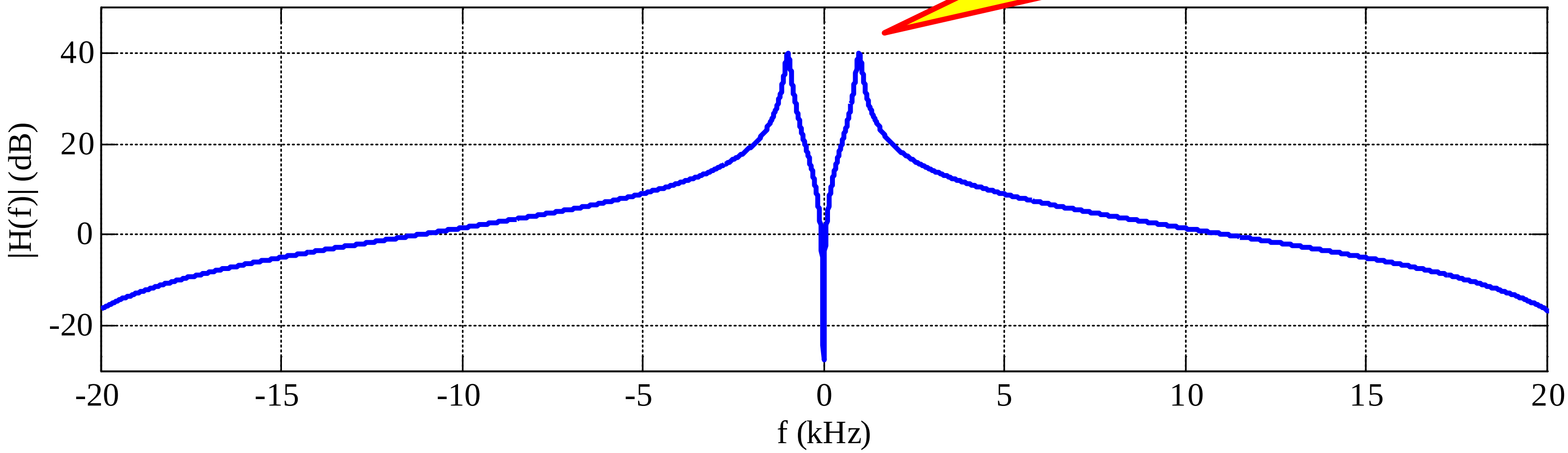
$$H(z) = \frac{1-z^{-2}}{1-1.9606z^{-1}+0.9801z^{-2}}$$

Now… how do we check the actual frequency response?

$$H(z) = \frac{1 - z^{-2}}{1 - 1.9606z^{-1} + 0.9801z^{-2}} \qquad H(\Omega) = \frac{1 - e^{-j2\Omega}}{1 - 1.9606e^{-j\Omega} + 0.9801e^{-j2\Omega}}$$

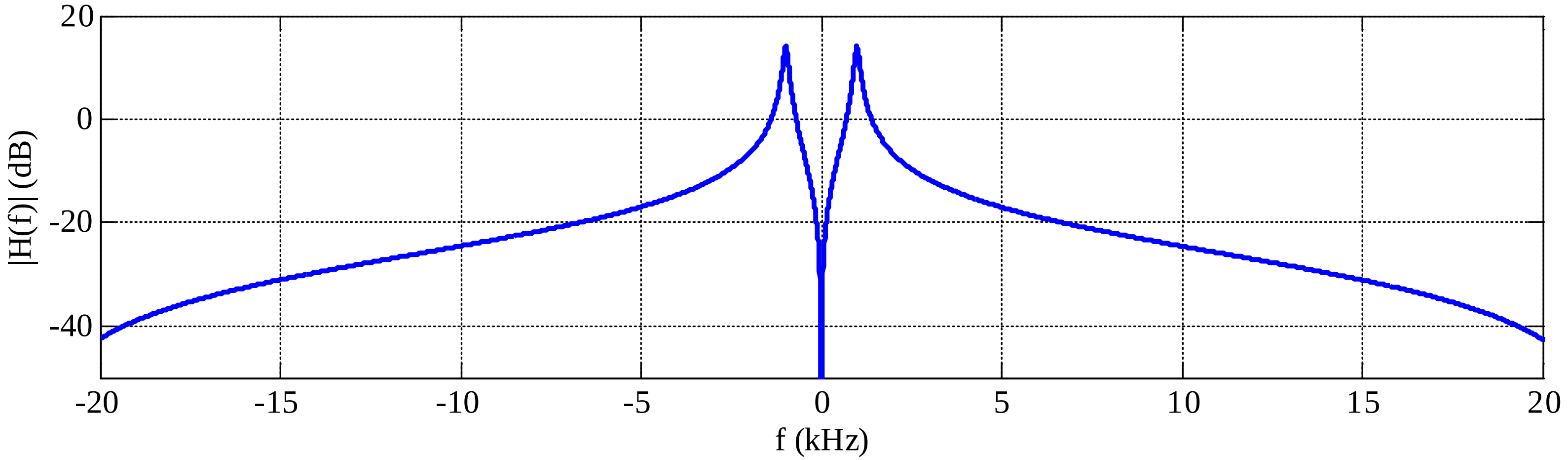
```
Omega = -pi:0.001:pi;
H=freqz([1 0 -1],[1   -1.9606   0.9801],Omega);
f = Omega*Fs/(2*pi);
plot(f/1000,20*log10(abs(H)))
```

Pretty High!!! Gain of 40 dB is 10,000 times more power!!!



So… put an overall "gain" term out front…

```
H=freqz(0.05*[1 0 -1],[1   -1.9606   0.9801],Omega);
plot(f/1000,20*log10(abs(H)))
```

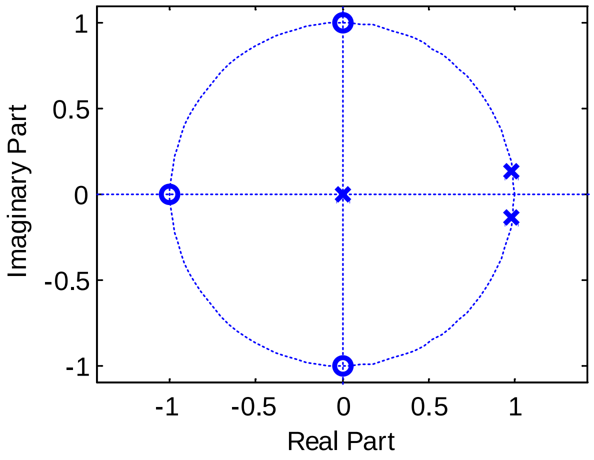


Now… apply filter and listen.

```
g_f=filter(0.05*[1 0 -1],[ 1.0000   -1.9606    0.9801],x);
sound(g_f,Fs)
```

We hear some change… can we take this a bit farther?

Back to our P-Z Plot… What do we want?



zplane([1 1 1 1], [1.0000 -1.9606 0.9801])

$$H(z) = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{1 - 1.9606z^{-1} + 0.9801z^{-2}}$$

Want a zero at z = 1

$$H(z) = \frac{(1 + z^{-1} + z^{-2} + z^{-3})(1 - z^{-1})}{1 - 1.9606z^{-1} + 0.9801z^{-2}}$$

<u>**Matlab Trick!!!**</u>

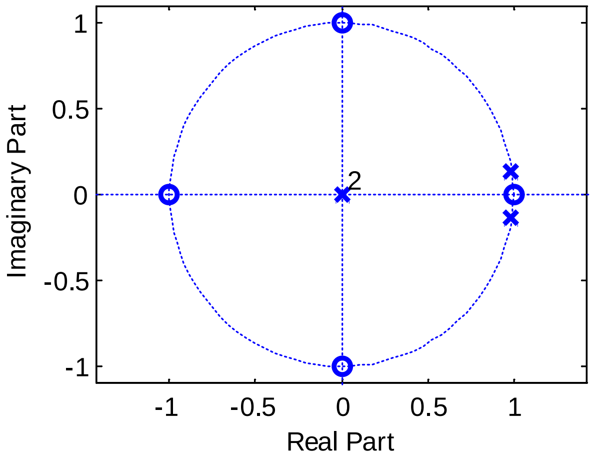
Use conv command to find coefficients of multiplied z^{-1} polynomials

conv([1 1 1 1],[1 -1]) gives 1 0 0 0 -1

$$H(z) = \frac{(1 + z^{-1} + z^{-2} + z^{-3})(1 - z^{-1})}{1 - 1.9606z^{-1} + 0.9801z^{-2}} = \frac{1 - z^{-4}}{1 - 1.9606z^{-1} + 0.9801z^{-2}}$$

$$H(z)=\frac{1-z^{-4}}{1-1.9606z^{-1}+0.9801z^{-2}}$$

zplane([1 0 0 0 -1], [1.0000 -1.9606 0.9801])

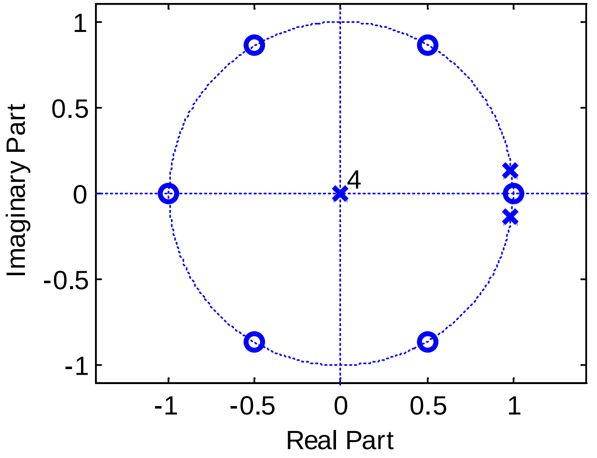


Now… an interesting thing comes from exploration of this form with EVEN integer p

$$H(z)=\frac{1-z^{-p}}{1-1.9606z^{-1}+0.9801z^{-2}}$$

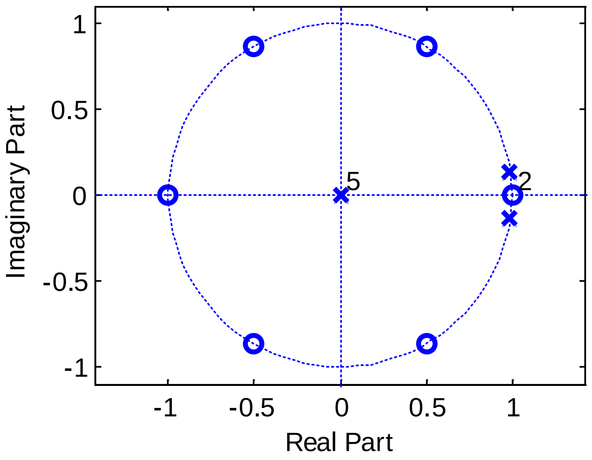
zplane([1 0 0 0 0 0 -1], [1.0000 -1.9606 0.9801])

zplane([1 zeros(1,5) -1], [1.0000 -1.9606 0.9801])

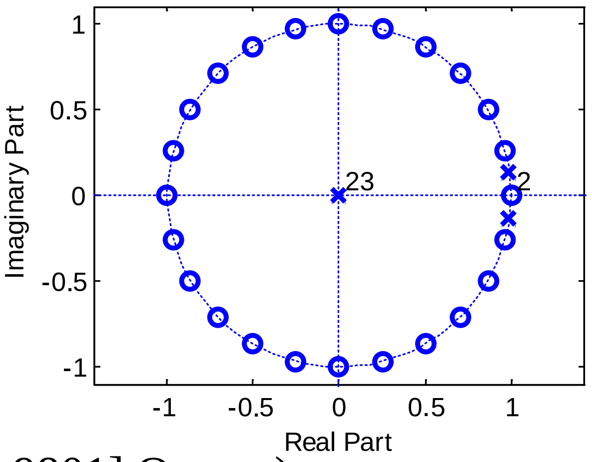


One last thing… We'd like another zero at z = 1 to push the FR down at low frequencies.

```
>> conv([1 zeros(1,5) -1],[1 -1])

ans =

     1    -1     0     0     0     0    -1     1

zplane([1 -1  zeros(1,4)  -1 1 ], [ 1.0000   -1.9606    0.9801])
```

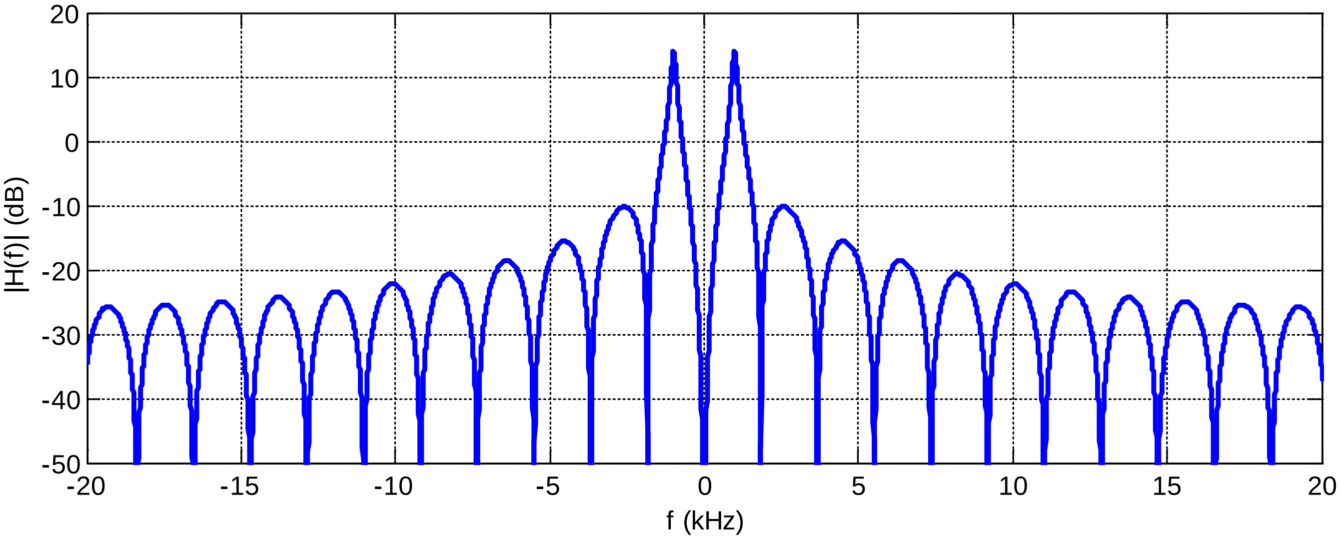


Taking this to an the extreme….

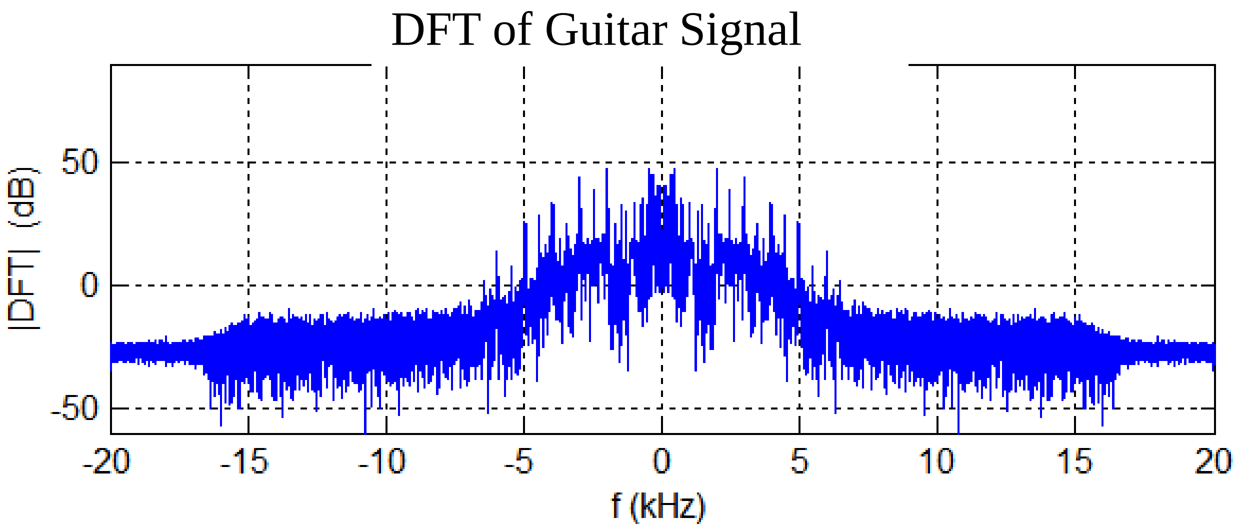


zplane([1 -1 zeros(1,22) -1 1], [1.0000   -1.9606   0.9801])

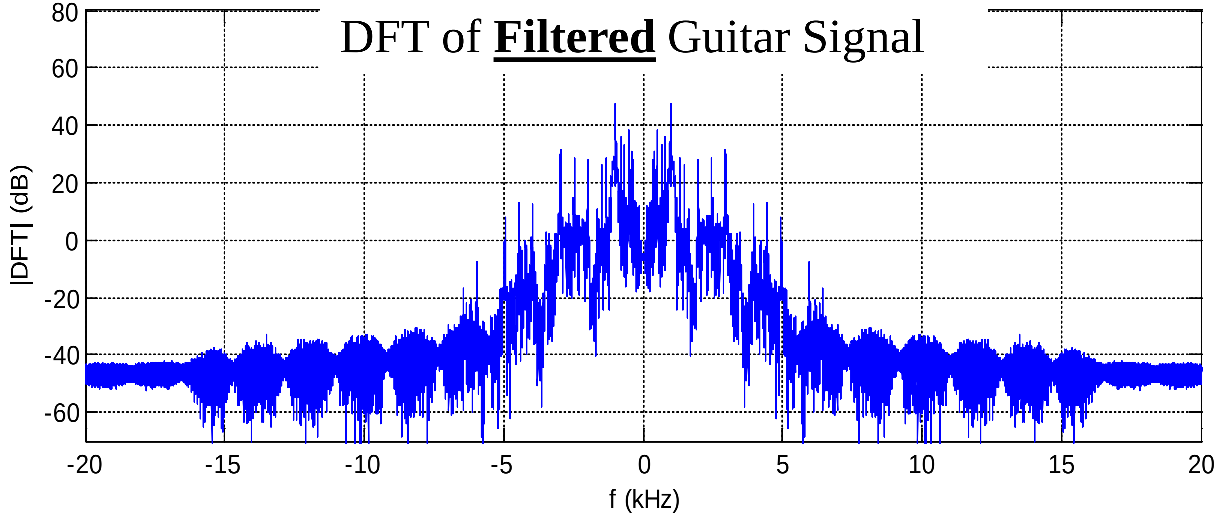
H = freqz(0.05*[1 -1 zeros(1,22) -1 1], [1.0000   -1.9606   0.9801],Omega);
plot(f/1000,20*log10(abs(H)))



g_f=filter(0.05*[1 -1 zeros(1,22) -1 1],[1.0000   -1.9606   0.9801],x);
sound(g_f,Fs)



DFT of Guitar Signal
50
0
-50
|DFT| (dB)
-20
-15
-10
-5
0
5
10
15
20
f (kHz)



DFT of **Filtered** Guitar Signal
80
60
40
20
0
-20
-40
-60
|DFT| (dB)
-20
-15
-10
-5
0
5
10
15
20
f (kHz)