EEL3135: Lab 4

Exercise 4.1:

This exercise shows you how to carefully interprete the results obtained from your MATLAB function dtft, particularly for the cases of infinite-length signals.

Use the tables in the information section, or otherwise, to analytically determine the expressions of the DTFTs for the signals below. Use your MATLAB function dtft to calculate and plot (magnitude and angle) the DTFTs. For each signal, truncate its length to 10, if needed, when using your function dtft to calculate the DTFT. Compare your plot with the DTFT expression you obtain analytically. Do they match? Repeat the same MATLAB calculation of the DTFT and comparison with the analytic expression by truncating each signal to length 100 instead. Explain the effects of increasing the truncation length.

```
type dtft.m
```

```
function H = dtft(b, w)
H = 0;
    [r,c] = size(b);
    for i = 0:c-1
        H = H + b(i+1)*exp((-j*i).*(w));
    end
end
```

```
type plotResposes.m
```

```
function plotResposes(X, X_analytical, X_100, w, isStem)
    figure;
    subplot(3,2,1);
    plot(w,20*log10(abs(X)));
    grid on;
    title("Truncation Lenth of 10 Magnitude");
    ylabel("Amplitude (dB)");
    xlabel("Frequency ( rads/(sec*sample) )");
    subplot(3,2,2);
    plot(w,angle(X));
    grid on;
    title("Truncation Lenth of 10 Phase");
    ylabel("Pahse (rad)");
    xlabel("Frequency ( rads/(sec*sample) )");

    if isStem == 1
        subplot(3,2,3)
        plot(w,20*log10(abs(X_analytical)));
        title("Analytical DTFT Magnitude");
        ylabel("Amplitude (dB)");
        xlabel("Frequency ( rads/(sec*sample) )");
        grid on;
        subplot(3,2,4)
        plot(w, angle(X_analytical));
        grid on;
        title("Analytical DTFT Phase");
        ylabel("Pahse (rad)");
        xlabel("Frequency ( rads/(sec*sample) )");
    else
        subplot(3,2,3)
        stem(w,20*log10(abs(X_analytical)));
        title("Analytical DTFT Magnitude");
        ylabel("Amplitude (dB)");
        xlabel("Frequency ( rads/(sec*sample) )");
        grid on;
        subplot(3,2,4)
        plot(w, angle(X_analytical));
        grid on;
        title("Analytical DTFT Phase");
        ylabel("Pahse (rad)");
        xlabel("Frequency ( rads/(sec*sample) )");
    end;


    subplot(3,2,5);
    plot(w,20*log10(abs(X_100)));
    grid on;
    title("Truncation Lenth of 100 Magnitude");
    ylabel("Amplitude (dB)");
    xlabel("Frequency ( rads/(sec*sample) )");
    subplot(3,2,6);
    plot(w, angle(X_100));
    grid on;
    title("Truncation Length of 100 Phase");
    ylabel("Pahse (rad)");
    xlabel("Frequency ( rads/(sec*sample) )");
end
```

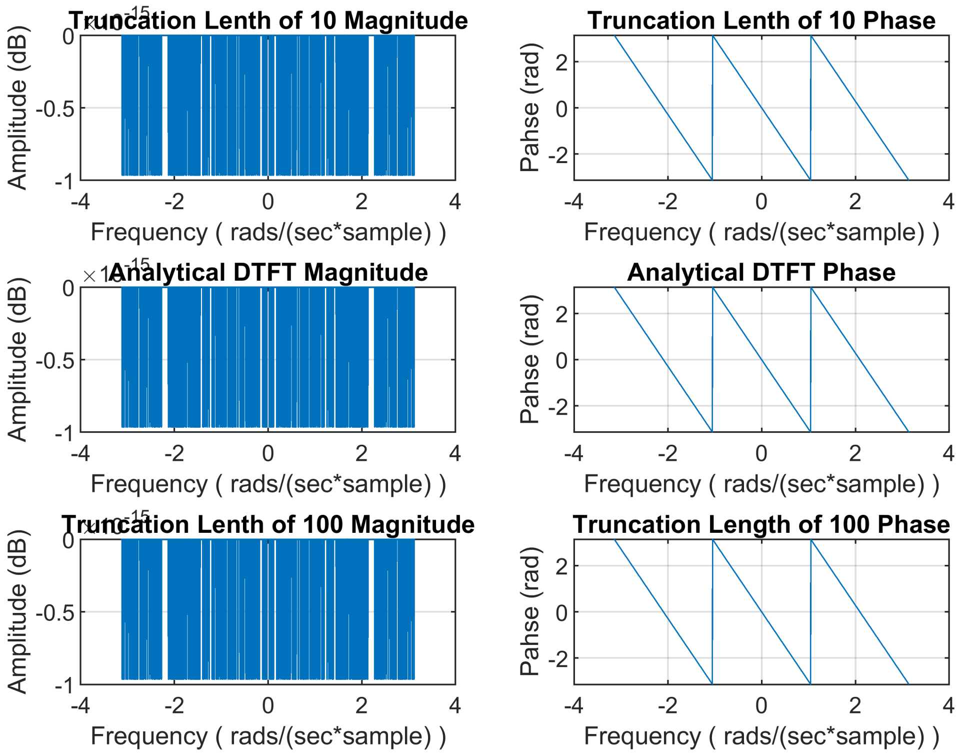
```
%frequency domain
w = -pi:pi/1000:pi;
%Time domain truncation length (100 and 10)
n = 0:10;
n_100 = 0:100;
```

The graphs shown below with differnet truncation lengths will look differnt and not match exactly to the analytical dtft even if it seem like it does; when zoomed in, there will be differences between them. However, increasing thetruncation length, n, leads to a more accurate dtft.

a) $x[n] = \delta[n-3]$

$X(\omega) = e^{-3j\omega}$

```
x = (n == 3);
X_analytical = exp(-j*w*3);
X_10 = dtft(x, w);
x_100 = (n_100 == 3);
X_100 = dtft(x_100,w);
plotResposes(X_10,X_analytical,X_100,w, 1);
```

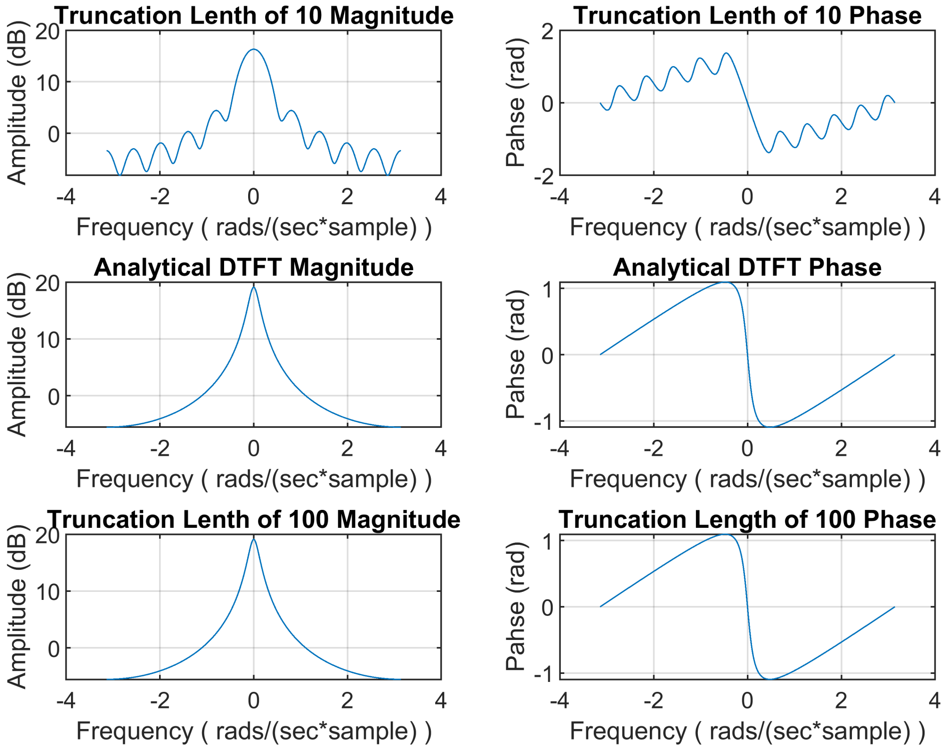


b)

$$x[n] = \left(\frac{8}{9}\right)^{n} u[n]$$

$$X(\omega) = \frac{1}{1 - \frac{8}{9}e^{-j\omega}}$$

```
x = (8/9).^n .* (n >= 0);
X_analytical = (1-(8/9)*exp(-j*w)).^(-1);
X_10 = dtft(x,w);
x_100 = (8/9).^n_100 .* (n_100 >= 0);
X_100 = dtft(x_100,w);
plotResposes(X_10,X_analytical,X_100,w,1)
```

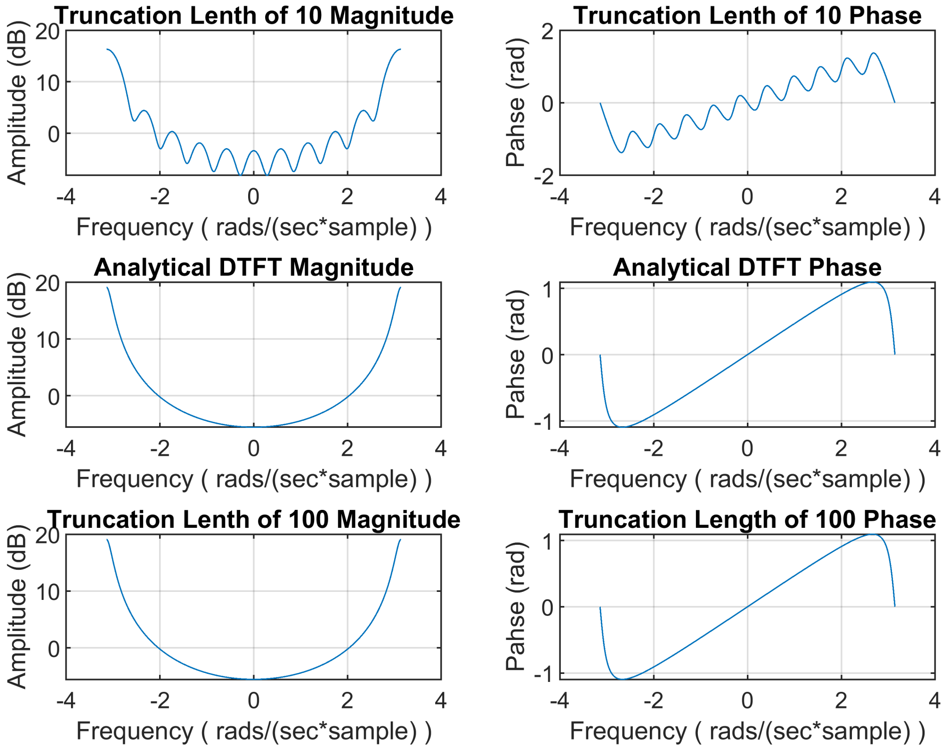


c)

$$x[n] = \left(-\frac{8}{9}\right)^n u[n]$$

$$X(\omega) = \frac{1}{1-\left(-\frac{8}{9}\right)e^{-j\omega}}$$

```
x = (-8/9).^n .* (n >= 0);
x_100 = (-8/9).^n_100 .* (n_100 >= 0);
X_analytical = 1./(1-(-8/9)*exp(-j*w));
X_10 = dtft(x,w);
X_100 = dtft(x_100,w);
plotResposes(X_10,X_analytical,X_100,w, 1)
```



d)

$$x[n] = u[n] - u[n-5]$$

$$X(\omega) = \frac{1 - e^{-5j\omega}}{1 - e^{-j\omega}}$$

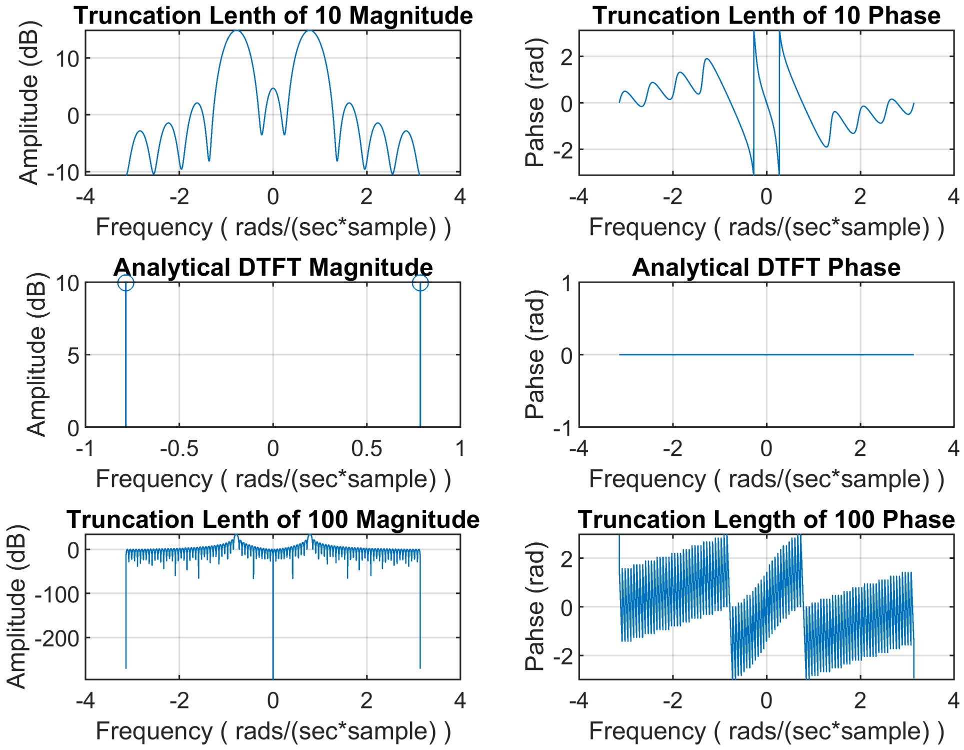
```
x = (n >= 0) - (n >= 5);
x_100 = (n_100 >= 0) - (n_100 >= 5);
X_analytical = (1 - exp(-j*w*5))./(1 - exp(-j*w));
X_10 = dtft(x,w);
X_100 = dtft(x_100,w);
```

e)

$$x[n] = \cos\left(\frac{\pi}{4}n\right)$$

$$X(\omega) = \pi\left(\delta_{2\pi}\left(\omega - \frac{\pi}{4}\right) + \delta_{2\pi}\left(\omega + \frac{\pi}{4}\right)\right)$$

```
x = cos((pi/4)*n);
x_100 = cos((pi/4)*n_100);
X_analytical = pi*((round(w,4) == round(pi/4,4))+(round(w,4)== round(-pi/4,4)));
X_10 = dtft(x,w);
X_100 = dtft(x_100, w);
plotResposes(X_10,X_analytical,X_100,w,0)
```



Increasing the truncation length, leads the DTFT to become more accurate to the actual analytical DTFT; basically, there is a smaller trancient response with larger n. Also, when increasing the truncation from 1-10 to 1-100, there will be less frequency leakage that makes it harder to find the actual frequencies with there intensities (dB) of the signal due to inacuraccy.

Exercise 4.2:

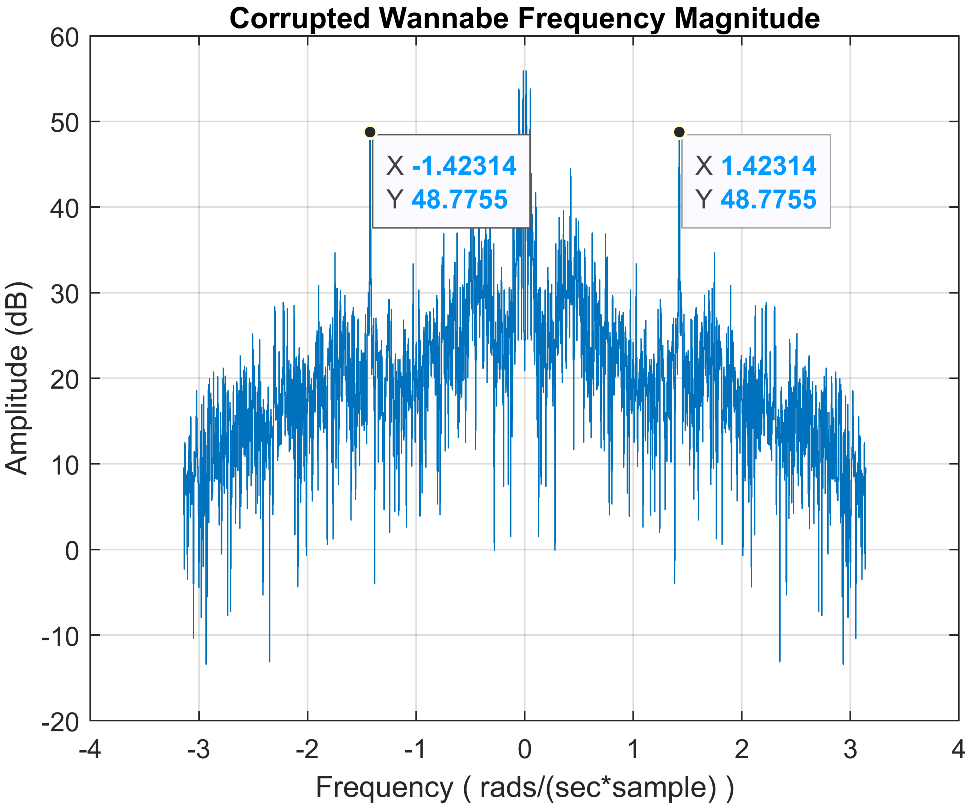
This exercise shows you how to employ DTFT to analyze an audio signal corrupted by a sinusoidal interfering signal. Specifically, you will examine the DTFT to determine the frequency of the sinusoidal interference and then use a nulling filter to remove the interference. The corrupted audio signal is provided in the WAV file corrupted wannabe.wav

a) Load corrupted wannabe.wav into MATLAB. Use your function dtft to calculate the DTFT of the audio signal. Plot the magnitude of the DTFT. Hint: You should use a frequency vector that has at least 2000 points to calculate the DTFT so as to properly identify the interference frequency in part (b) below. It may take a minute or so to compute the DTFT.

```
[x, FS] = audioread("corrupted_wannabe.wav");
x = x';
X = dtft(x, w);
figure;
plot(w, 20*log10(abs(X)));
grid on;
```

```
title("Corrupted Wannabe Frequency Magnitude");
ylabel("Amplitude (dB)");
xlabel("Frequency ( rads/(sec*sample) )");

ax3 = gca;
chart3 = ax3.Children(1);
datatip(chart3,1.423,48.78);
datatip(chart3,-1.423,48.78);
```



b) Identify the frequency of the sinusoidal interference. Give this frequency in normalized radian frequency as well as continuous-time cyclic frequency. Hint: You may use “Tools→Data Tips” to identify the value of interference frequency from the DTFT magnitude plot. You may also use the MATLAB function find to more accurate determine the frequency of interest.

$$F_s = 22050 \, \frac{\text{samples}}{\text{sec}}$$

$$\omega_0 = \pm 1.4231 \, \frac{\text{rad}}{(\text{sample})(\text{sec})} \Rightarrow 48.7755 \text{ dB}$$

$$f_0 = \frac{\omega_0}{2\pi} F_s = 4994.3 \text{ Hz}$$

Interference occurs 4994.3 Hz, or 1.4231 rads/(sample)sec

```
FS
```

```
FS = 22050
```

```
w0 = 1.42314
```

```
w0 = 1.4231
```

```
db = 48.7755
```

```
db = 48.7755
```

```
f0 = (w0/(2*pi))*FS
```

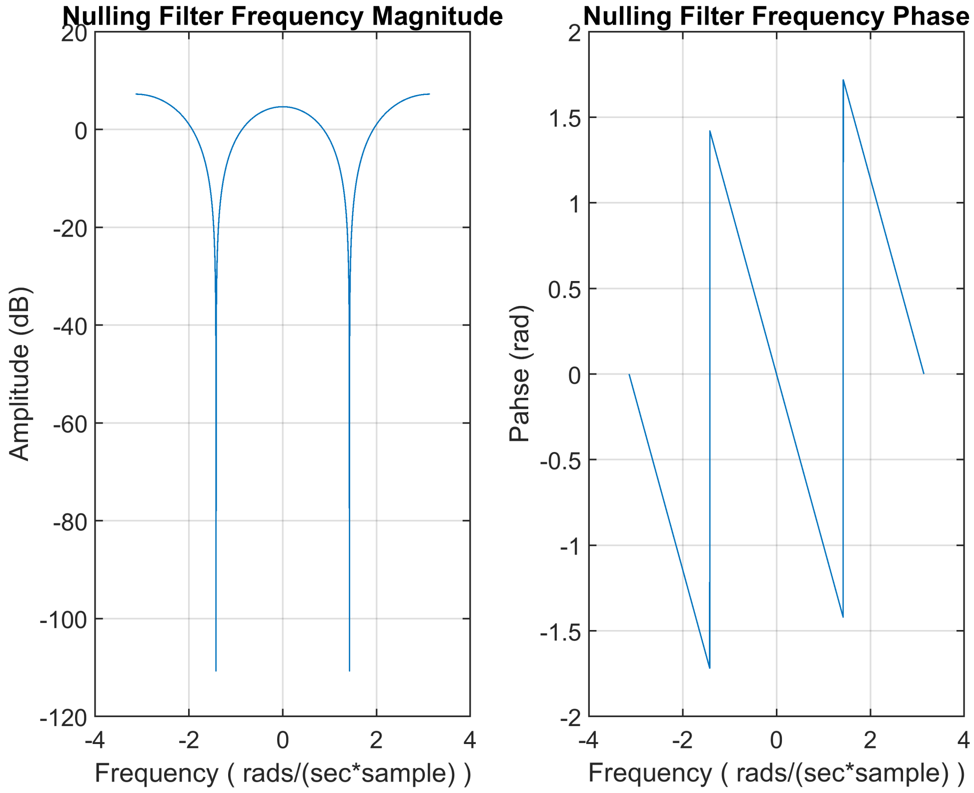
```
f0 = 4.9943e+03
```

c)This filter can remove a sinusoid at the normalized radian frequency specified by bw0. Design this filter to remove the interference that you identified in (b), i.e. specify the impulse response of the resulting filter in the vector h. Use your function dtft to calculate and plot the magnitude response and phase response of the filter.

$y[n] = x[n] - 2\cos(\omega_0)x[n-1] + x[n-2]$

$h[n] = \delta[n] - 2\mathrm{coss}(\omega_0)\delta[n-1] + \delta[n-2]$

```
h = (n_100 == 0) - 2*cos(w0)*(n_100 == 1) + (n_100 == 2);
H = dtft(h, w);
figure;
subplot(1,2,1);
plot(w,20*log10(abs(H)));
grid on;
title("Nulling Filter Frequency Magnitude");
ylabel("Amplitude (dB)");
xlabel("Frequency ( rads/(sec*sample) )");
subplot(1,2,2);
plot(w, angle(H));
title("Nulling Filter Frequency Phase");
ylabel("Pahse (rad)");
xlabel("Frequency ( rads/(sec*sample) )");
grid on;
```

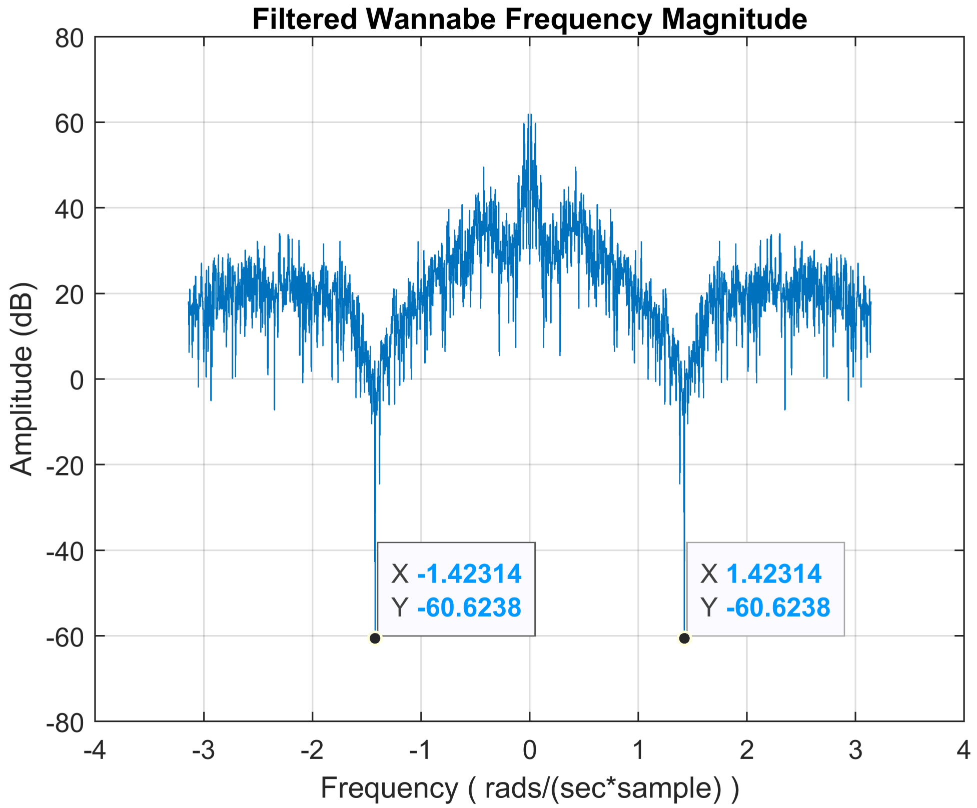


d) Apply the nulling filter to the corrupted audio signal. Listen to the filtered audio and save it to the WAV file filtered wannabe.wav. Submit the WAV file. Use your dtft function to calculate and plot the magnitude of the DTFT of the filtered audio signal. Describe the differences between the DTFTs of the original audio and the output of the nulling filter, particularly with respect to the interference signal.

```
y = conv(x, h);
%{
    After Filtering there could be peaking/Distortion that could occur, so
    it is always good to Normalize the output data, y, from the filter
%}
y = y./max(abs(y));
audiowrite("filtered_wannabe.wav",y,FS);

Y = dtft(y,w);
figure;
plot(w,20*log10(abs(Y)));
title("Filtered Wannabe Frequency Magnitude");
ylabel("Amplitude (dB)");
xlabel("Frequency ( rads/(sec*sample) )");
grid on;

ax2 = gca;
chart2 = ax2.Children(1);
datatip(chart2,1.423,-60.62);
datatip(chart2,-1.423,-60.62);
```



The diffeerence between the filltered and corrupted magnitude of wannabeis that the corrupted audio file had large spike decible (dB) intensities at plus/minus 1.42314 rads/(sec*sample) compared to the rest of the graph, which is a high pitched frequency that played over the whole crouptted file at 4994 Hz for the interference. The nulling filter attenuated the interference to very negtive dB filtering out the high pitched frquecny interfering with the file shown with a dip in the filtered magnitude plot of wannabe.