Digital Signal Processing

(ELC3430)

Assignment 2

Topic: **Remove interference from a given .wav file**

Name: Maha Zakir Khan

G. No: GL3136

S. No: A3EL-02

F. No: 19ELB056

Date of performing experiment: 30|03|2022

Date of performing submission: 30|03|2022

Software Used:

MATLAB®, Release 2021a (R2021a), a programming platform designed specifically to analyze and design systems and products. The heart of MATLAB is the MATLAB language, a matrix-based language, it provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.

Procedure:

1. For the given .wav file which contains some audio signal+interference, you are required to remove the interference.
2. For your reference original file (file name is "File_Original_Signal.wav") without interference is also given (see attached file).
3. The .wav file containing signal + interference is different for each student. The file name contains the S.No. of the students. Therefore, process your own audio file.
4. Use SCILAB or MATLAB to process the given corrupted signal (containing signal + interference) so that interference can be remove to the maximum possible extend.
5. Remember that the sampling rate should be 10 kHz.
6. You will be evaluated based on assignment report along with viva-voce, The viva-voce date on assignment shall be announce latter. Upload your assignment report in pdf by 30.03.2022, 6:00 pm.
7. Assignment report must contain codes, graphs, plots, spectrum, etc along with the logic & concept you have used to reduce the interference. The analysis part should be clearly explained in the report. Each figure. graph, etc should be explained in words also.
8. Also include a paragraph explaining about interference signal in your file and the parameters of interfering parameters.

Program:

Original .wav file details:

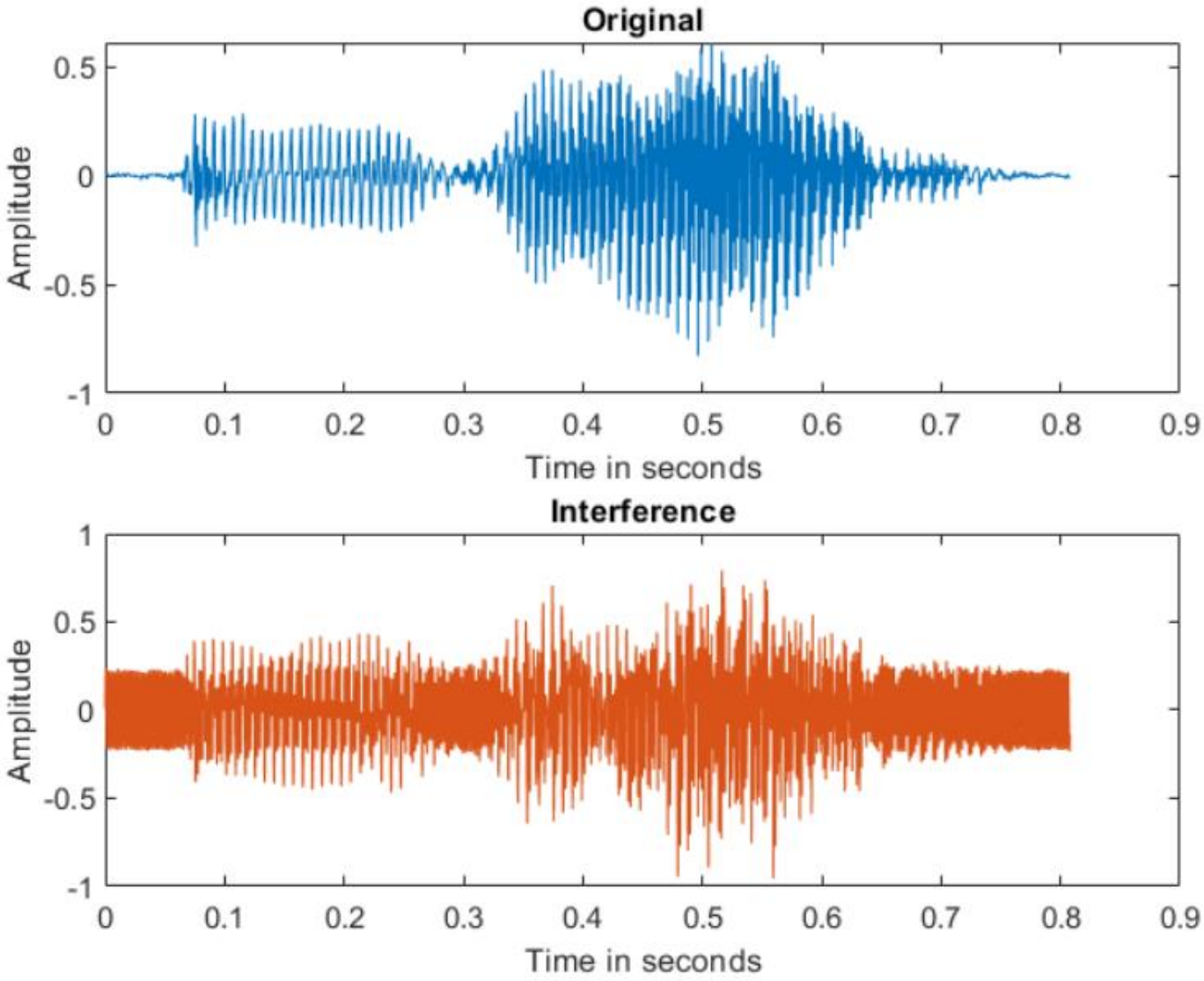
```
          Filename: 'C:\Users\Maha Khan\Downloads\File_Original_Signal.wav'
 CompressionMethod: 'Uncompressed'
       NumChannels: 1
        SampleRate: 10000
      TotalSamples: 8079
          Duration: 0.8079
             Title: []
           Comment: []
            Artist: []
     BitsPerSample: 16
```

Interference .wav file details:

```
          Filename: 'C:\Users\Maha Khan\Downloads\For_S.No._02___enjoy_10k1F4190.wav'
 CompressionMethod: 'Uncompressed'
       NumChannels: 1
        SampleRate: 10000
      TotalSamples: 8079
          Duration: 0.8079
             Title: []
           Comment: []
            Artist: []
     BitsPerSample: 16
```

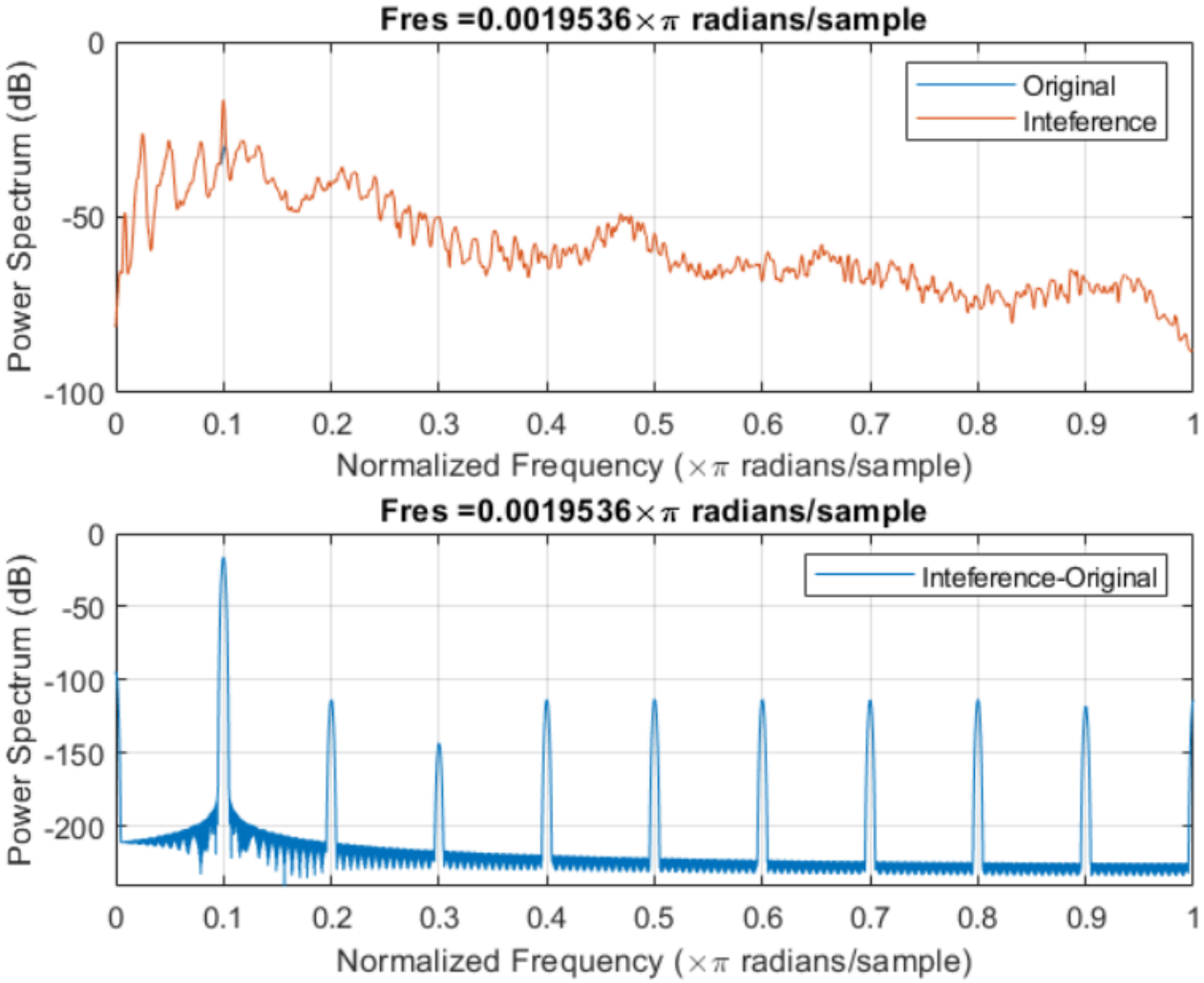
The signals under study are single channel signals with sampling rate fs =10khz. Total duration of both signals is 0.8079 seconds with bits per sample (pcm encoded with) 16 bits per sample

```
 % Program to analyse and remove interference from a given .wav file using
 % original signal as a reference

 info = audioinfo('C:\Users\Maha Khan\Downloads\File_Original_Signal.wav');
 ts = info.TotalSamples;
 n = info.Duration;
  % Original Voice signal with signal channel and fs = 10kHz  as the sampling
frequency
 [signal , fs] = audioread('C:\Users\Maha
Khan\Downloads\File_Original_Signal.wav');

 % Same Voice signal with interference and fs as the sampling frequency
 info_i = audioinfo('C:\Users\Maha
Khan\Downloads\For_S.No._02___enjoy_10k1F4190.wav');
 [signal_i , fs] = audioread('C:\Users\Maha
Khan\Downloads\For_S.No._02___enjoy_10k1F4190.wav');

 % Signal waveform of Original and Inteference signal
 figure;
 subplot(2,1,1);
 plot((0:ts-1)*n/ts,signal)
 xlabel('Time in seconds');
 ylabel('Amplitude');
 title('Original');

 subplot(2,1,2);
 plot((0:ts-1)*n/ts,signal_i,'Color','#D95319');
 xlabel('Time in seconds');
 ylabel('Amplitude');
 title('Interference');
```



We notice that the Interference signal has some high frequency noise interfering with the original waveform, to get a better understanding of the differences between the original and interference signal we will analyze the frequency spectrum of both signals

```
% Comparing Spectrum of Original and Interference signal using in-built
% matlab function pspectrum
figure;
subplot(2,1,1);
pspectrum(signal);
hold on;
pspectrum(signal_i);
legend('Original','Inteference');

subplot(2,1,2);
% Difference between spectrum of Original and Interference signal
pspectrum(signal_i-signal);
legend('Inteference-Original');
```



On observing the difference spectrum, we find that the difference peaks at approx. 0.1w or 999 Hz, therefore for this case to remove the interference we can use a notch filter with center frequency 0.1w or 999 Hz

Note: Frequency in Hertz is given as:

fn = Normalized Frequency w (in rad) * Sampling Rate / 2pi

```
      % fn = wo*fs/pi;  fn = 998.7790;

% pspectrum returns the frequencies fi corresponding frequency estimate
% dp_i.
[dp_i,fi]=pspectrum(signal_i-signal);
```

pspectrum returns the normalized frequency fi in radians and power spectrum dp_i in linear

Find the normalized frequency where the difference is maximum (po) and choose it as the center frequency of our notch filter

```
[po, i]= max(dp_i);
 po = mag2db(po)/2;

% Normalised centre frequency wo = 0.3138 rad/sample for notch filter
wo = fi(i);
```

```
po =

  -16.1381
```

(in dB)

Using a second order IIR notch-filter to design our desired filter

The notch filter can be designed as:

$$H(z) = \frac{1 - 2cos(\omega_o)z^{-1} + z^{-2}}{1 - 2rcos(\omega_o)z^{-1} + r^2z^{-2}}$$

I don't know the relation behind this, but by observation I noticed that on limiting 1- r to 0, the bandwidth bw of the filter kept on decreasing, and quality factor Q increased as:

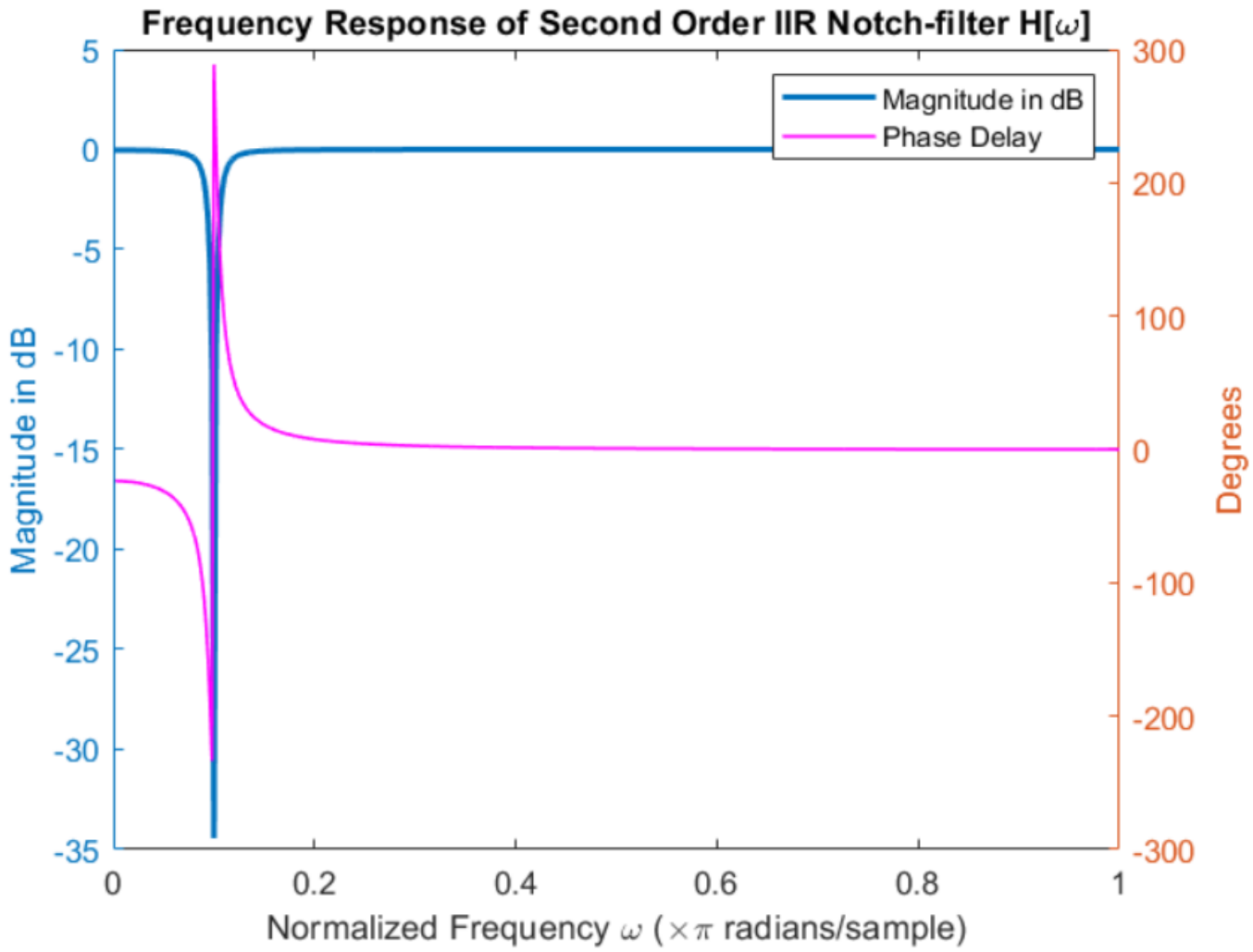
$$Q = wo/bw$$

```
r = 0.98; % Taking r as close as possible to 1 for poles to be
          % close to zeros for ideal pole-zero plot of notch filter
wo = wo;  % wo = 0.3138 radians/sample

% Filter coefficients a and b
b = [1, -2*cos(wo), 1]
a = [1, -2*r*cos(wo), r^2]
```

Filter Coefficients:

```
b = 1×3
    1.0000   -1.9023    1.0000

a = 1×3
    1.0000   -1.8643    0.9604
```

```
% Frequency Response of the filter
figure;
[db,mag,pha,grd,w]=freqz_m(b,a);

yyaxis left;
plot(w/pi,db,'linewidth',1.8);
ylabel('Magnitude in dB');
hold on
yyaxis right;
plot(w/pi,pha,'color','m','linewidth',1);
ylabel('Degrees');
xlabel('Normalized Frequency \omega (\times\pi radians/sample)');
title('Frequency Response of Second Order IIR Notch-filter H[\omega]')
legend('Magnitude in dB','Phase Delay');
```

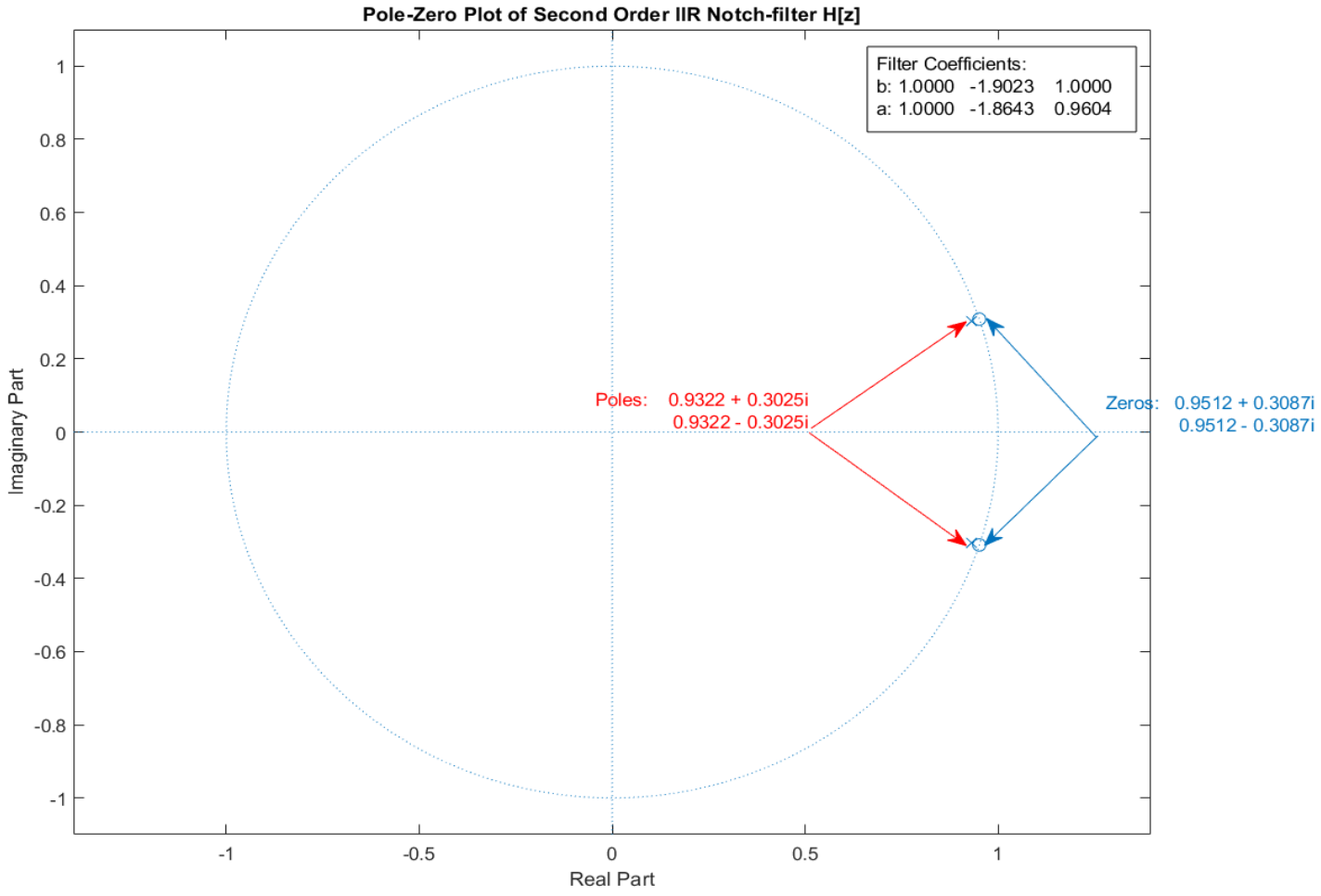


From the plot we can verify that the notch frequency is indeed around wo, with stopband attenuation of approx. 35dB.

The phase delay starts at -23 degrees reaching a limiting value of -233 degrees as w increases from 0 to wo, after that the phase delay changes abruptly to 299 degrees at wo and slowly decreases to 0 degrees

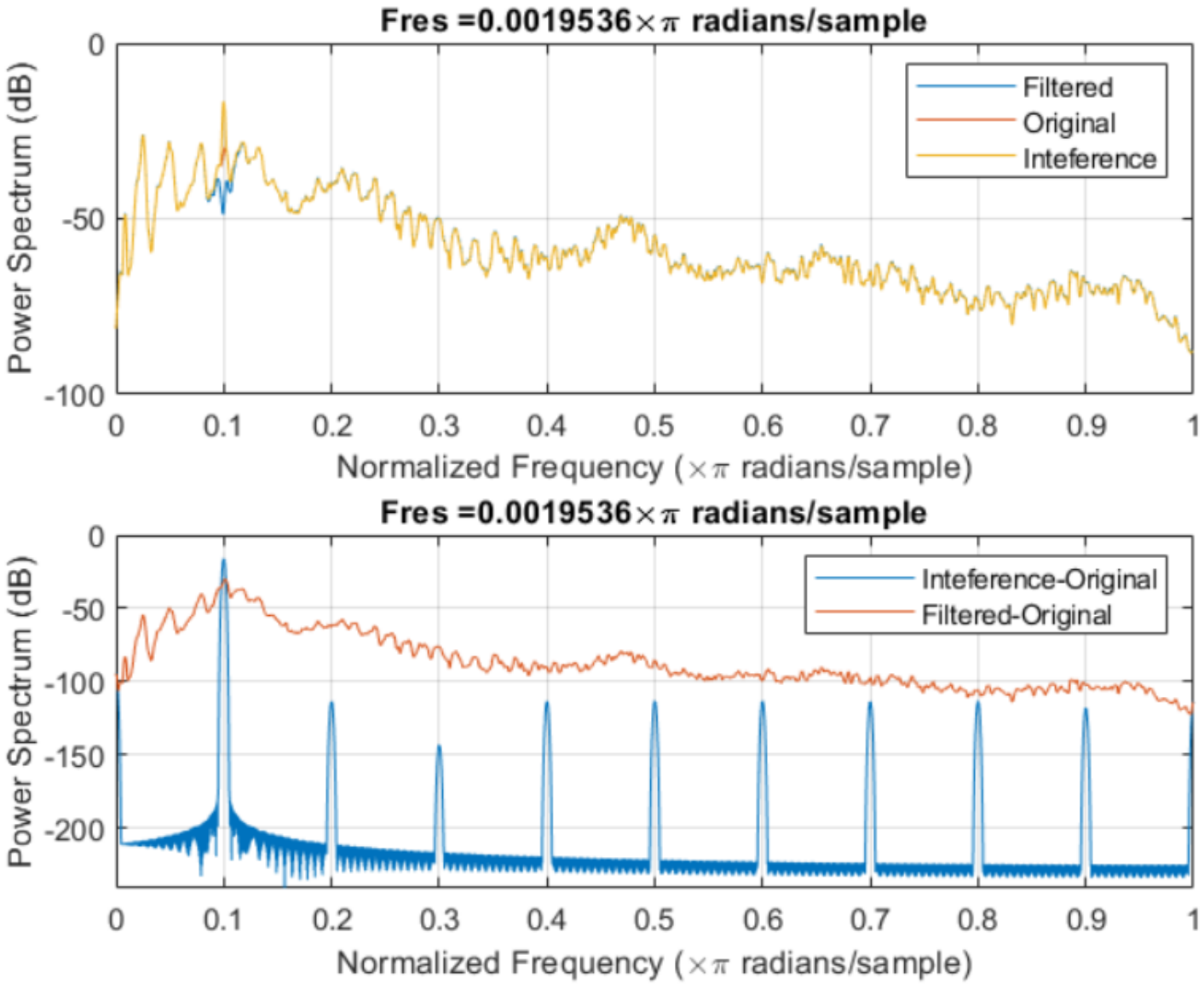
```
% Plotting poles and zeros on complex plane
figure;
zeros = roots(b)
poles = roots(a)
ax = zplane(b,a);
title('Pole-Zero Plot of Second Order IIR Notch-filter H[z]');
```

```
zeros = 2×1 complex
      0.9512 + 0.3087i
      0.9512 - 0.3087i

poles = 2×1 complex
      0.9322 + 0.3025i
      0.9322 - 0.3025i
```



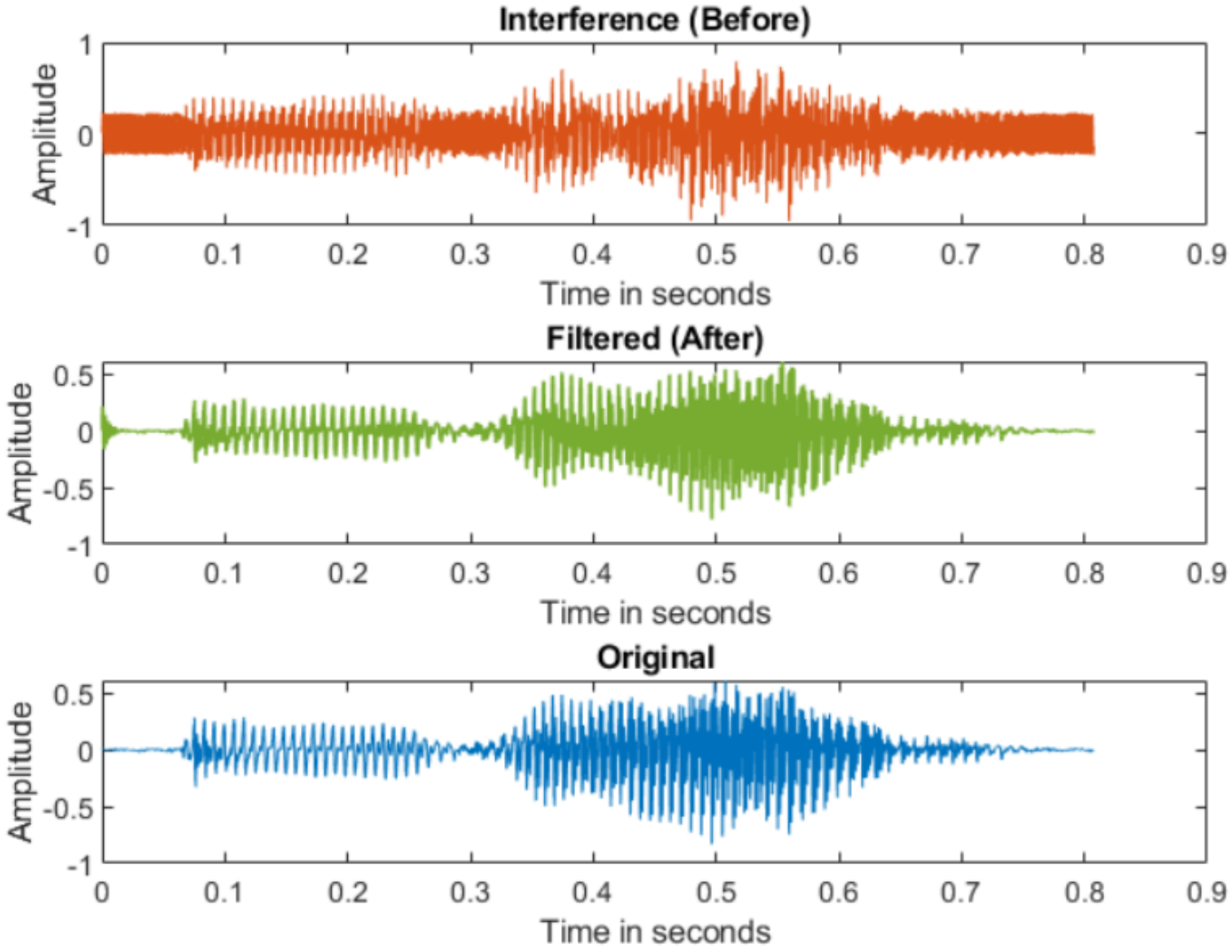
As we can see the poles are very close to zeros to keep the bandwidth as narrow as possible

```
% Frequency Response after Filtering
figure;
filtered = filter(b,a,signal_i);

subplot(211)
pspectrum(filtered);
hold on
pspectrum(signal);
hold on;
pspectrum(signal_i);
legend('Filtered','Original','Inteference');

subplot(212);
pspectrum((signal_i-signal));
hold on;
pspectrum((filtered-signal));
legend('Inteference-Original','Filtered-Original');
```



From the plot we can observe that the 999Hz frequency component is attenuated by approx. 31dB, but since the notch filter isn't ideal, nearby frequencies are also attenuated by some range.

```
% Comparing Signal in time-domain before and after filtering
figure;
subplot(3,1,1);
plot((0:ts-1)*n/ts,signal_i,'Color','#D95319');
xlabel('Time in seconds');
ylabel('Amplitude');
title('Interference (Before)')

subplot(3,1,2);
plot((0:ts-1)*n/ts,filtered,'linewidth',1,'Color','#77AC30');
xlabel('Time in seconds');
ylabel('Amplitude');
title('Filtered (After)');

subplot(3,1,3);
plot((0:ts-1)*n/ts,signal);
xlabel('Time in seconds');
ylabel('Amplitude');
title('Original');
```



In time domain analysis we observed that the filtered signal is significantly recovered and looks similar to the original waveform