Contents

QUESTION 1 COMMENTING

```
% DO NOT REMOVE THE LINE BELOW
clear; close all; clc;
```

QUESTION 2 Thinking in Three Domains 1

```
% LOAD AUDIO
% MAKE SURE 'music.wav' is in the same directory!
```

```
[x, fs] = audioread('music.wav');

% DEFINE AXES
w = -pi:pi/8000:pi-pi/8000;
t = 1/fs:1/fs:length(x)/fs;
```

Filter Application

```
f_center = (2*pi*0.14255);
mypoles = [ ...
        0.97*exp(1j*f_center*2*pi*1.1) ...
        0.97*exp(-1j*f_center*2*pi*1.1) ...
        0.99*exp(1j*f_center*2*pi*1.15) ...
        0.99*exp(-1j*f_center*2*pi*1.15) ...
        0.95*exp(1j*f_center*2*pi*1.05) ...
        0.95*exp(-1j*f_center*2*pi*1.05) ...
        0.9*exp(1j*f_center*2*pi*1.2) ...
        0.9*exp(-1j*f_center*2*pi*1.2) ...
        0.85*exp(1j*f_center*2*pi*1.3) ...
        0.85*exp(-1j*f_center*2*pi*1.3) ...
];

% DEFINE ZEROS
myzeros = [1*exp(1j*((2*pi*0.430791)/fs + 0.5)*2*pi) ...
        1*exp(-1j*((2*pi*0.430791)/fs + 0.5)*2*pi) ...
        0.96*exp(1j*((2*pi*0.585907)/fs + 0.5)*2*pi) ...
        0.96*exp(-1j*((2*pi*0.585907)/fs + 0.5)*2*pi) ...
        0.94*exp(1j*((2*pi*0.868258)/fs + 0.5)*2*pi) ...
        0.94*exp(-1j*((2*pi*0.868258)/fs + 0.5)*2*pi) ...
        0.99*exp(1j*((2*pi*1.15885)/fs + 0.75)*2*pi*2) ...
        0.99*exp(-1j*((2*pi*1.15885)/fs + 0.75)*2*pi*2) ...
        -1];

% CONVERT TO COEFFICIENTS
[b1,a1] = pz2ba(mypoles,myzeros);

% COMPUTE GAIN TO MAINTAIN SIGNAL AMPLITUDE AROUND TARGET FREQ
target_freq = f_center;
w0 = 2*pi*target_freq/fs;
G = abs(sum(a1 .* exp(-1j*w0*(0:length(a1)-1))) / sum(b1 .* exp(-1j*w0*(0:length(b1)-1))));

% FILTER SIGNAL
y = filter(G*b1, a1, x);
```

Plotting the Outputs

```
%figure(1);
%subplot(211)
%plot(w, abs(DTFT(x, w)));
%xlabel('Normalized Frequency (Radians)');
%ylabel('Magnitude');
%title('Original Signal (Frequency Domain)');

%subplot(212)
%plot(w, abs(DTFT(y, w)));
%xlabel('Normalized Frequency (Radians)');
```

```
%ylabel('Magnitude');
%title('Filtered Signal (Frequency Domain)');


%figure(2);
%pzplot(b,a);
%axis equal;

%figure(3);
%subplot(211);
%plot(t, x);
%xlabel('Time [s]');
%ylabel('Original Signal (Time Domain)')
%title('Time-Domain Input Plot');

%subplot(212);
%plot(t, y);
%xlabel('Time [s]');
%ylabel('Filtered Signal (Time Domain)');
%title('Time-Domain Plot of Output');
```

Playing the Audio

```
disp('Playing Original Music ... ');
soundsc(x, fs);
pause(length(x)/fs*1.1);

disp('Playing Filtered Music ... ');
soundsc(y, fs);
```

```
Playing Original Music ...
Playing Filtered Music ...
```

2 (a) Answer question

You would design a lowpass filter because you're trying to attenuate the higher frequency guitar and pass the lower frequency string bass. This is a really interesting application since you don't really have to consider the topology of the filter. However, you should choose convenient values for the resistor and capacitor or resistor and inductor depending on the variation that you opt for.

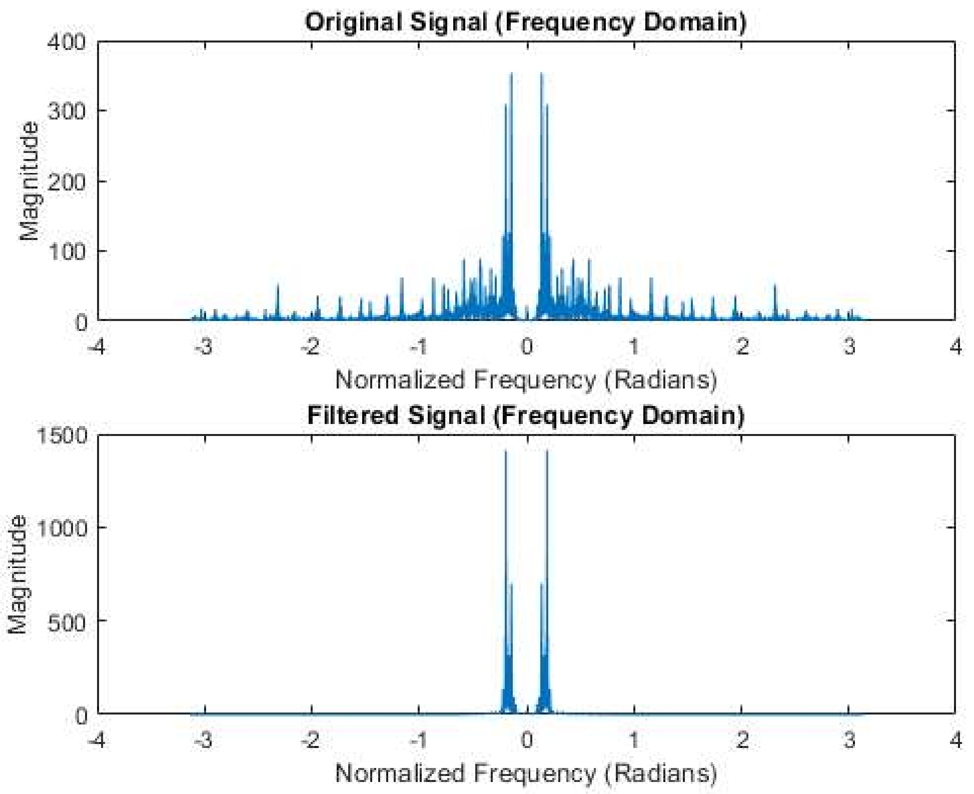
2 (b) Answer question

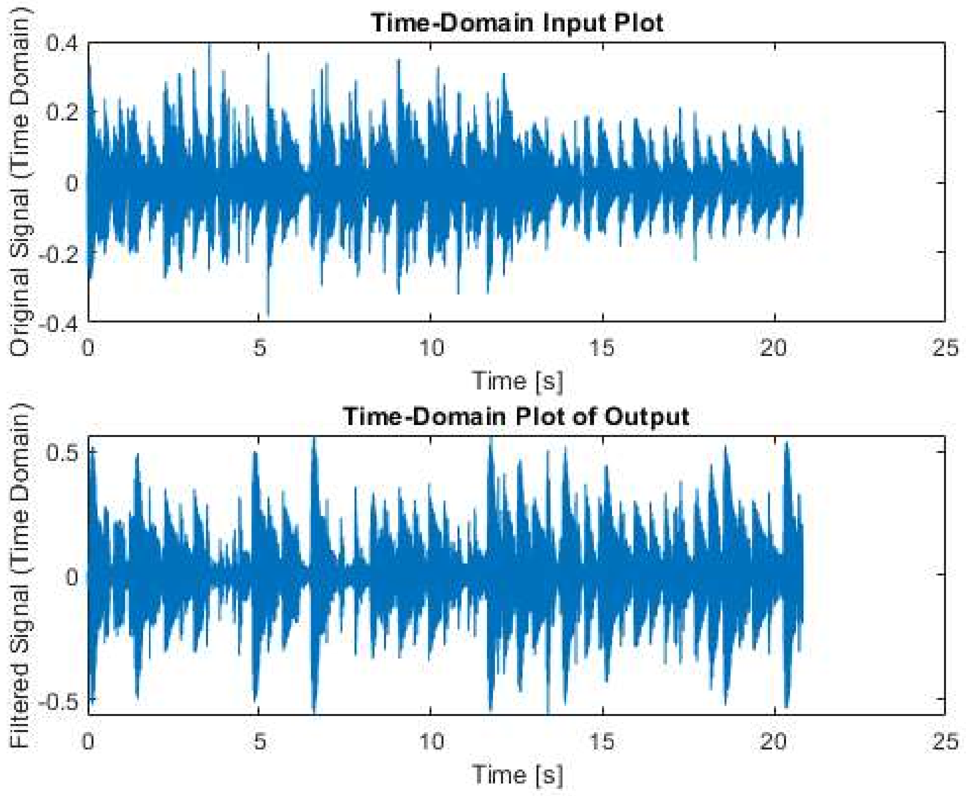
We opt for an IIR filter because we only have 10 values that we can use for poles and zeros. However, we have to use non-zero values for the poles because we want to amplify signals around certain center frequency.

2 (c) Plot inputs and outputs

```
figure(1);
subplot(211)
plot(w, abs(DTFT(x, w)));
xlabel('Normalized Frequency (Radians)');
ylabel('Magnitude');
title('Original Signal (Frequency Domain)');

subplot(212)
plot(w, abs(DTFT(y, w)));
```

```
xlabel('Normalized Frequency (Radians)');
ylabel('Magnitude');
title('Filtered Signal (Frequency Domain)');


figure(2);
subplot(211);
plot(t, x);
xlabel('Time [s]');
ylabel('Original Signal (Time Domain)')
title('Time-Domain Input Plot');

subplot(212);
plot(t, y);
xlabel('Time [s]');
ylabel('Filtered Signal (Time Domain)');
title('Time-Domain Plot of Output');
```



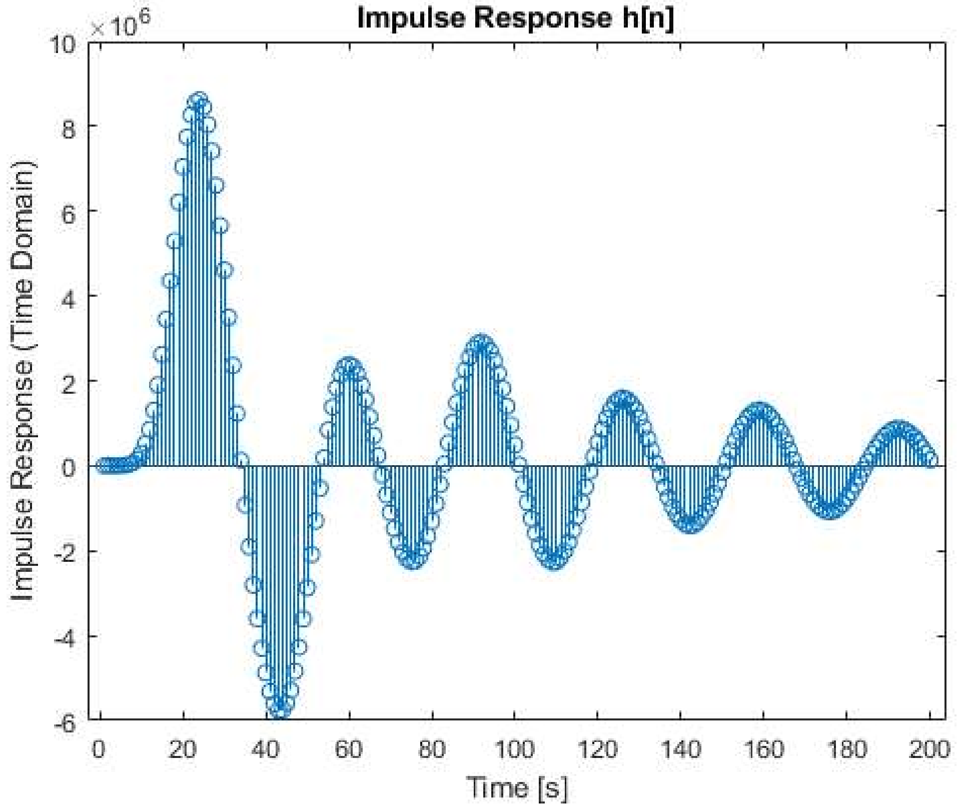


2 (d) Answer question

The plots tell me that the filter attenuates the higher frequencies whilst passing the lower frequency signals. This means that the filter is indeed a low-pass filter. The output plot does show the different areas where the guitar dominates the music as those outputs are much lower in the time domain plot of the output.

2 (e) Plot impulse response

```
h = zeros(1,200);
n = 1:100;
h(1) = 1;
y_imp = filter(b1, a1, h);

figure(3);
stem(y_imp);
xlabel('Time [s]');
ylabel('Impulse Response (Time Domain)');
title('Impulse Response h[n]');
```

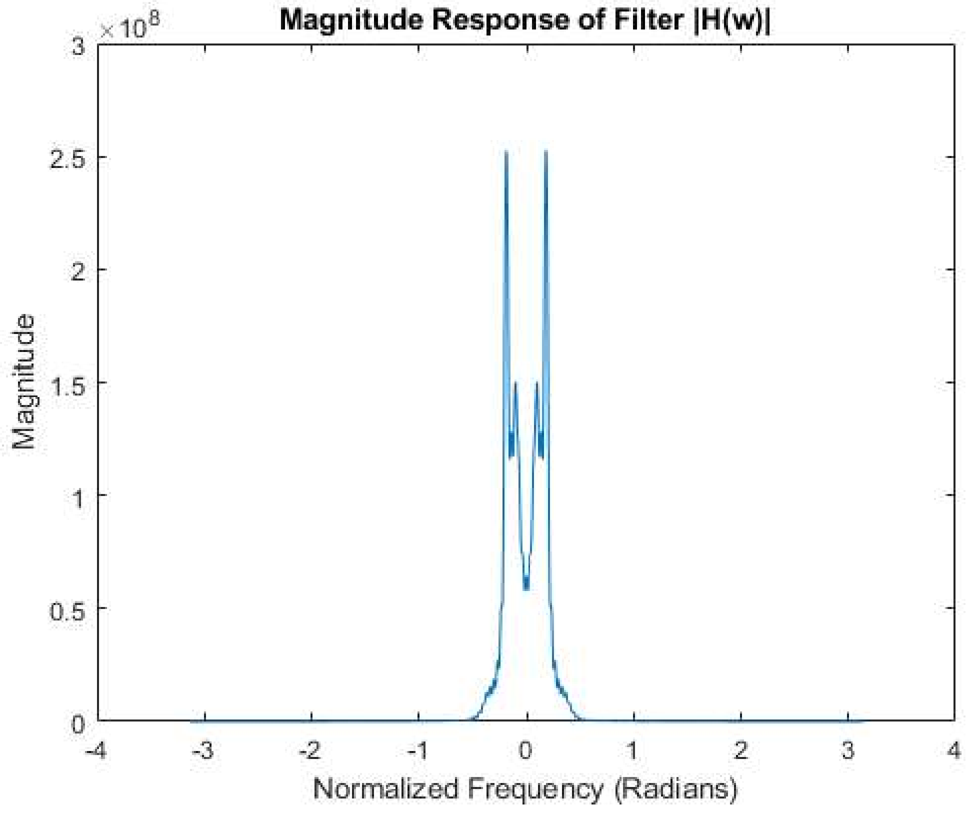


2 (f) Answer question

The system is an IIR filter with a damped response, meaning that there is are zeros with magnitudes that are less than 1. Additionally, since there a lot of poles near the unit circle but with less than a magnitude of 1, the system is stable and IIR.

2 (g) Plot frequency response

```
figure(4);
plot(w, abs(DTFT(y_imp, w)));
xlabel('Normalized Frequency (Radians)');
ylabel('Magnitude');
title('Magnitude Response of Filter |H(w)|');
```

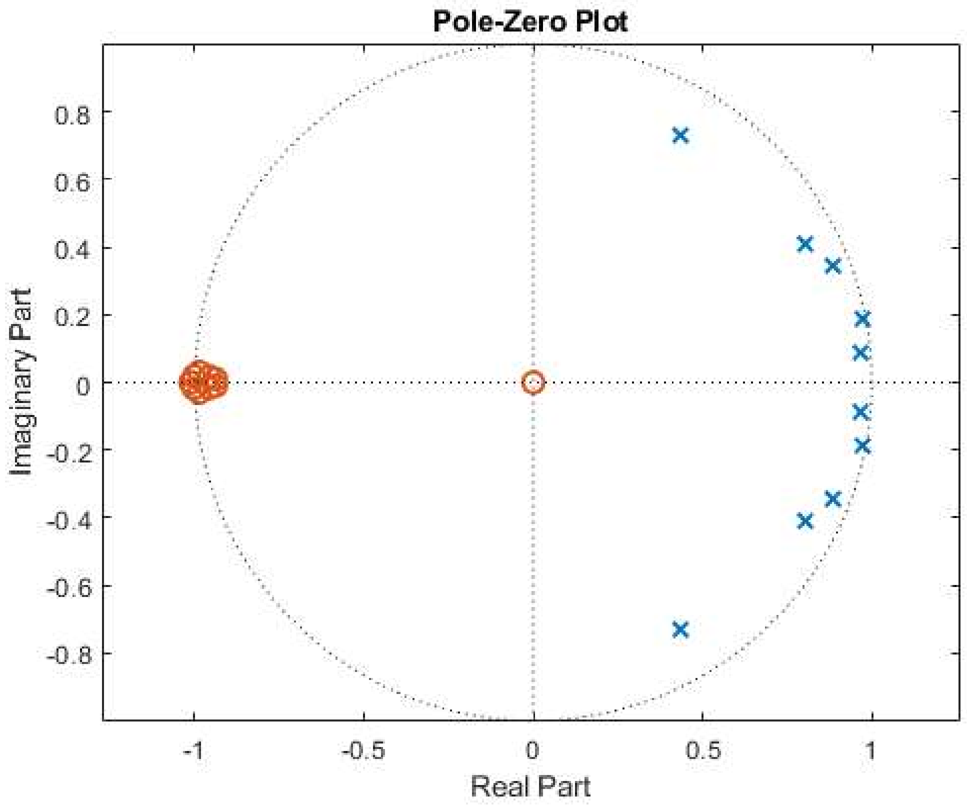


2 (h) Answer question

While the filter does have a really sharp peak, it does show the low-pass response that we are looking for. It has a really sharp transition band slope, but that could be caused by a number of things and we aren't using any specific filter topology, so it would be difficult to verify any specific filter characteristics. This filter also has a mirrored effect, so it looks a little like a bandreject filter as well, but if we only consider the positive frequencies, then it is somewhat of a low pass filter.

2 (i) Plot pole-zero plot

```
figure(5);
pzplot(b1,a1);
axis equal;
```



2 (j) Answer question

This filter is clearly a low pass filter as all the poles are centered around the lower frequencies, amplifying them, and the all zeros are centered around the higher frequencies. There is still zero at 0, but it does not seem to make a difference since there are more poles than zeros, so the system is causal (hence the right shift). Additionally, the system is stable since the poles all have a magnitude less than 1.

2 (k) Submit file on Canvas

```
audiowrite('q2.wav', y / abs(max(y)), fs);
save('q2.mat', 'b1', 'a1');
```

```
Warning: Data clipped when writing file.
```

QUESTION 3 Thinking in Three Domains 2

```
% LOAD AUDIO
% MAKE SURE 'music.wav' is in the same directory!
[x, fs] = audioread('music.wav');

% DEFINE AXES
w = -pi:pi/8000:pi-pi/8000;
t = 1/fs:1/fs:length(x)/fs;
```

Filter Design

```
f_center = (2*pi*0.14255);
myzeros = [ ...
```

```
        0.97*exp(1j*f_center*2*pi*1.1) ...
        0.97*exp(-1j*f_center*2*pi*1.1) ...
        0.99*exp(1j*f_center*2*pi*1.15) ...
        0.99*exp(-1j*f_center*2*pi*1.15) ...
        0.95*exp(1j*f_center*2*pi*1.05) ...
        0.95*exp(-1j*f_center*2*pi*1.05) ...
        0.9*exp(1j*f_center*2*pi*1.2) ...
        0.9*exp(-1j*f_center*2*pi*1.2) ...
        0.85*exp(1j*f_center*2*pi*1.3) ...
        0.85*exp(-1j*f_center*2*pi*1.3) ...
        ];

mypoles = [];

[b2,a2] = pz2ba(mypoles,myzeros);

target_freq = 2*f_center;
w0 = 2*pi*target_freq/fs;
G = abs(sum(a2 .* exp(-1j*w0*(0:length(a2)-1))) / sum(b2 .* exp(-1j*w0*(0:length(b2)-1))));

% FILTER SIGNAL
y = filter(G*b2, a2, x);
```

Play Audio

This filter is boof, I can't remove the string bass.

```
disp('Playing Original Music ... ');
soundsc(x, fs);
pause(length(x)/fs*1.1);

disp('Playing Filtered Music ... ');
soundsc(y, fs);
```

```
Playing Original Music ...
Playing Filtered Music ...
```

3 (a) Answer question

I designed a high-pass filter just because the string bass had to be attentuated, which has a lower frequency than the guitar. Since the bass is somewhat removed, the sounds super tinny. However, because of the harmonics of the string bass, some of the sound is not removed, especially at the end.

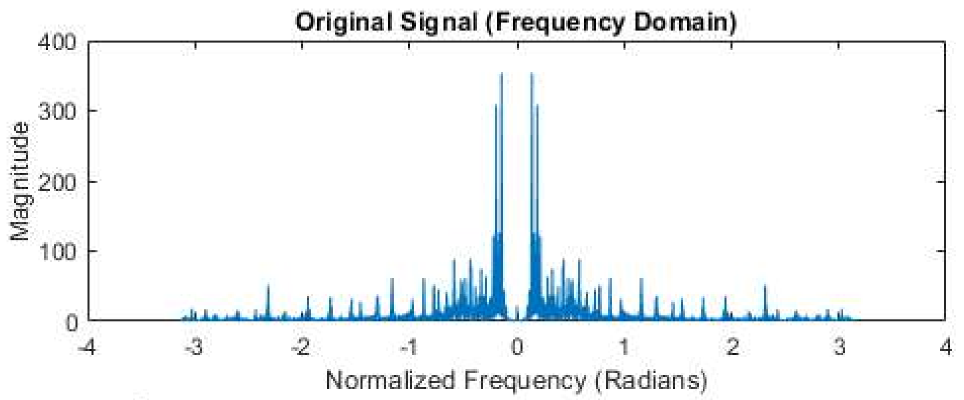
3 (b) Answer question

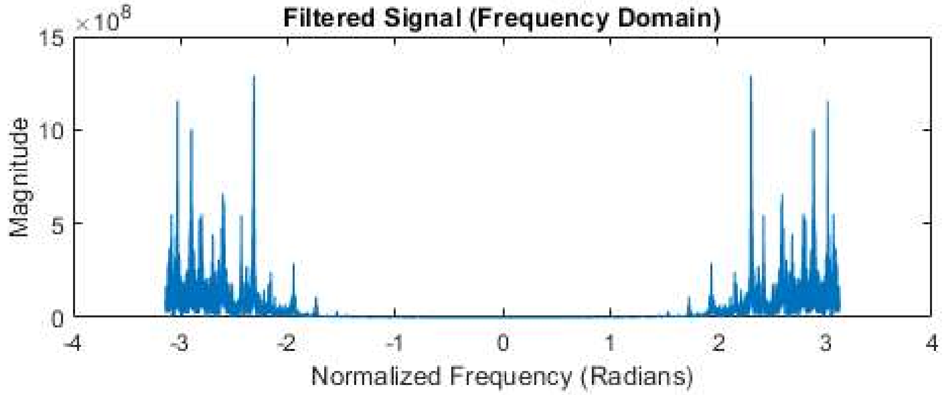
Woooo FIR FILTER!! That was one of the parameters given to us for this filter.

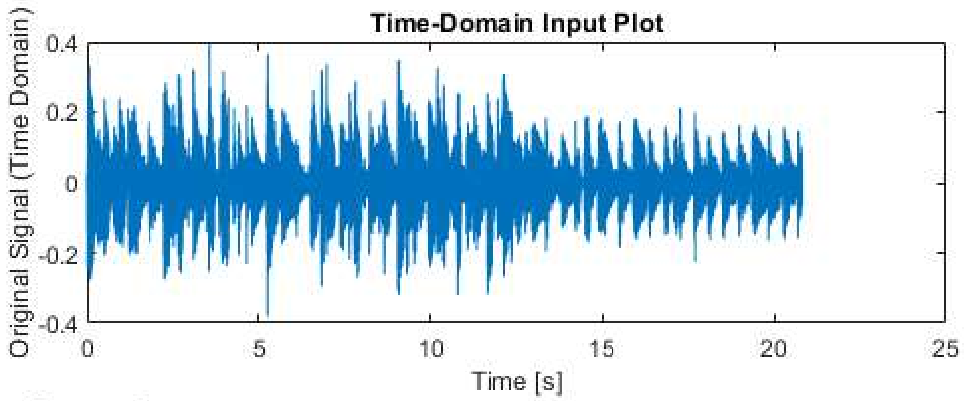
3 (c) Plot inputs and outputs

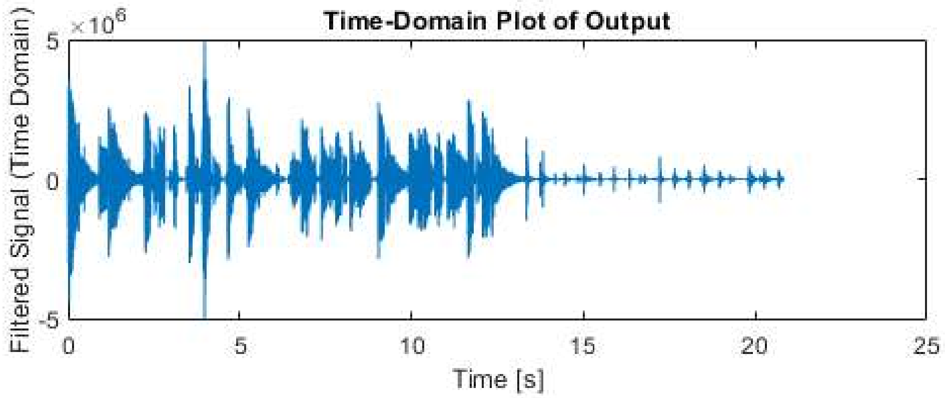
```
figure(6);
subplot(211)
plot(w, abs(DTFT(x, w)));
xlabel('Normalized Frequency (Radians)');
ylabel('Magnitude');
title('Original Signal (Frequency Domain)');
```

```
subplot(212)
plot(w, abs(DTFT(y, w)));
xlabel('Normalized Frequency (Radians)');
ylabel('Magnitude');
title('Filtered Signal (Frequency Domain)');


figure(7);
subplot(211);
plot(t, x);
xlabel('Time [s]');
ylabel('Original Signal (Time Domain)')
title('Time-Domain Input Plot');

subplot(212);
plot(t, y);
xlabel('Time [s]');
ylabel('Filtered Signal (Time Domain)');
title('Time-Domain Plot of Output');
```







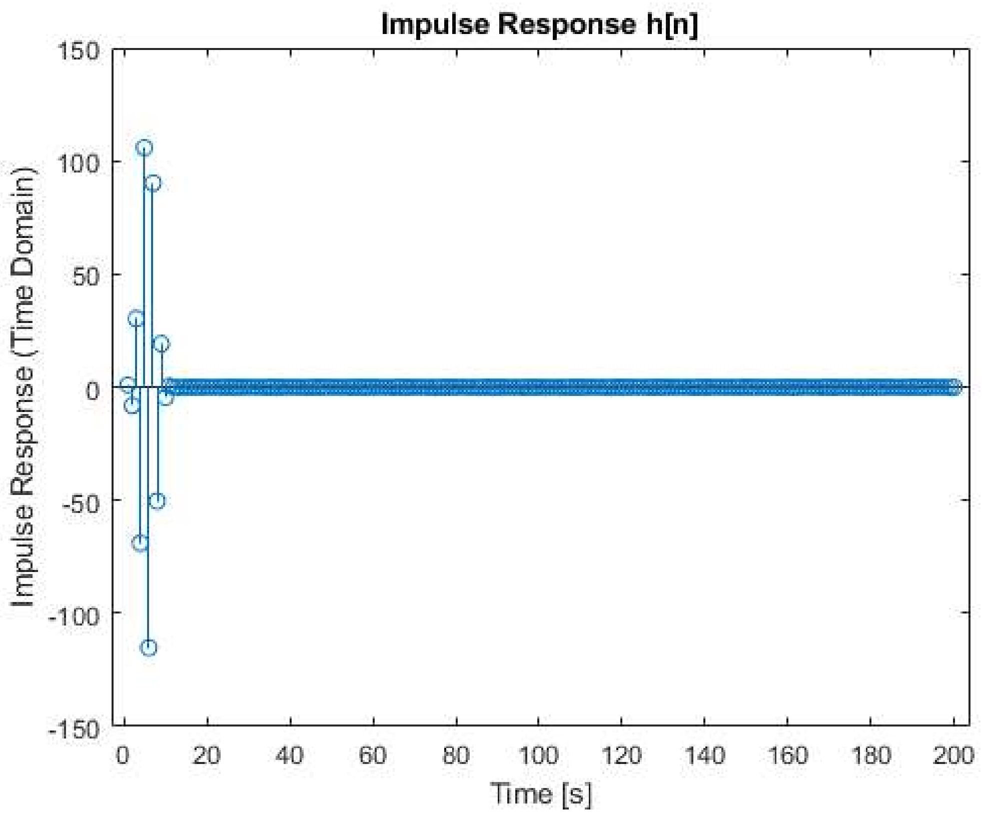


3 (d) Answer question

These plots tell me that the response is a High-Pass Filter. Also, while there is some output at the end of the file, with the string bass, it is so much smaller than the original value that it should not make much of a difference. The difference in gain between the input and output is minimal, meaning that there isn't much of a difference between the two apart from the attenuated frequencies.

3 (e) Plot impulse response

```
h = zeros(1,200);
n = 1:100;
h(1) = 1;
y_imp = filter(b2, a2, h);

figure(8);
stem(y_imp);
xlabel('Time [s]');
ylabel('Impulse Response (Time Domain)');
title('Impulse Response h[n]');
```

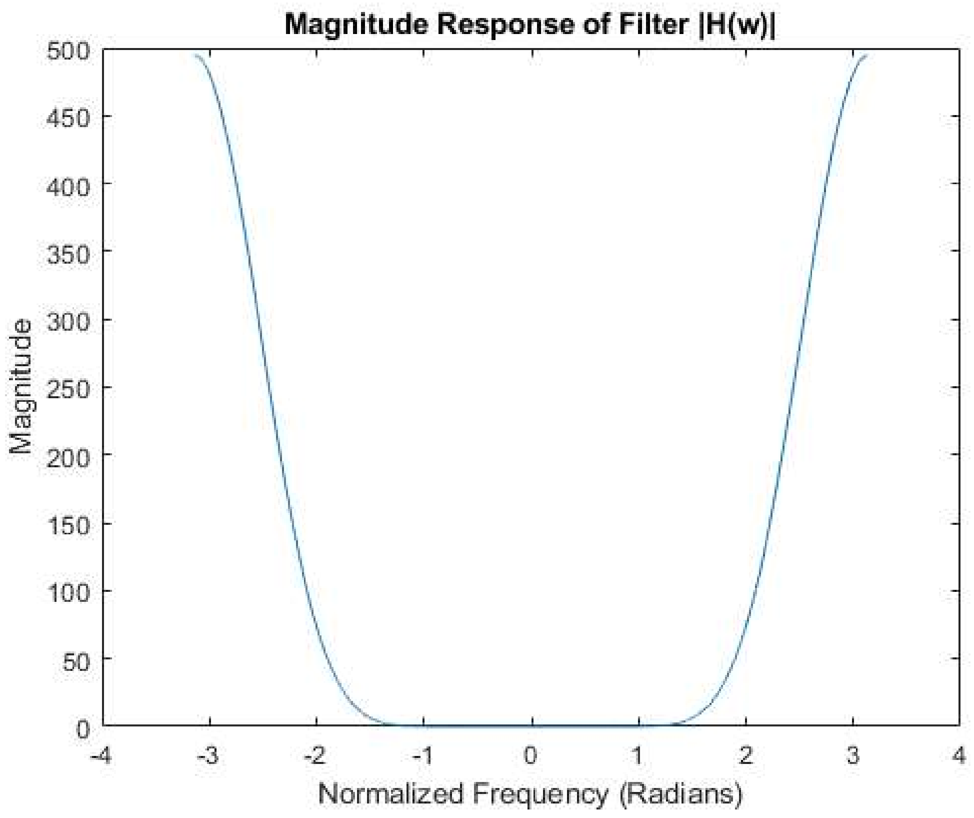


3 (f) Answer question

The impulse response tells us that this system is stable since there are no non-zero poles (so none can have a magnitude greater than 1) and the system is a finite impulse response since there are approximately only 10 values in the filtered response.

3 (g) Plot freqeuency response

```
figure(9);
plot(w, abs(DTFT(y_imp, w)));
xlabel('Normalized Frequency (Radians)');
ylabel('Magnitude');
title('Magnitude Response of Filter |H(w)|');
```

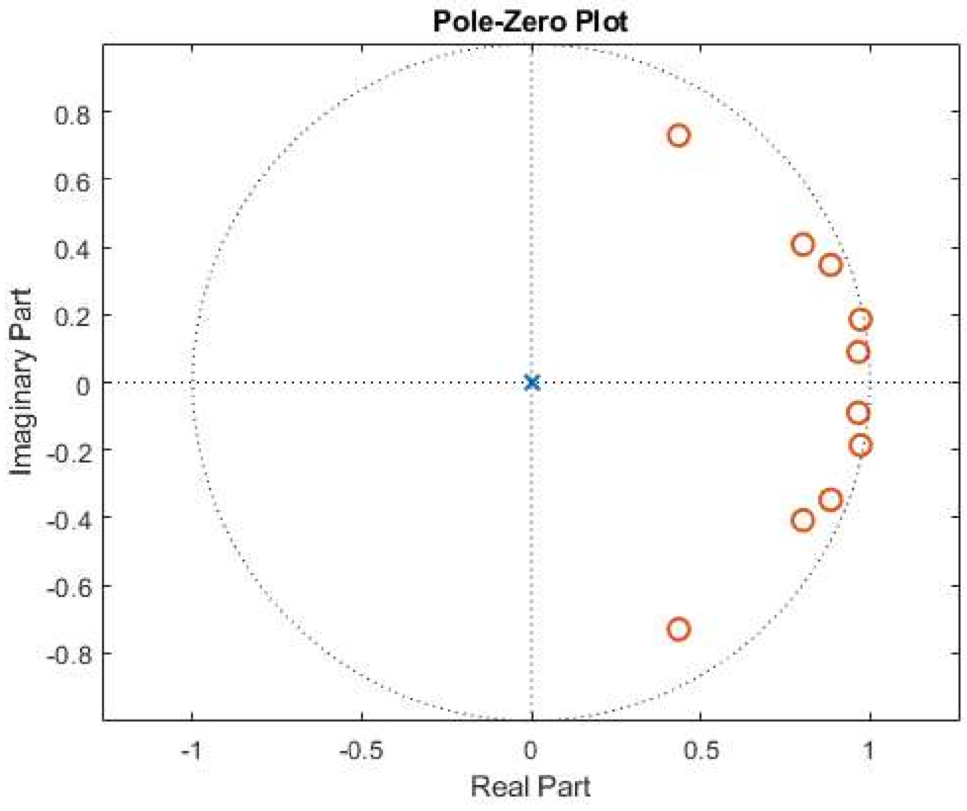


3 (h) Answer question

This shows us that the system is clearly a high-pass filter of some order. However, we do know that the order is greater than 1 given the steepness of the transition band. overall, this filter is nearly exactly what was required for this system.

3 (i) Plot pole-zero plot

```
figure(10);
pzplot(b2,a2);
axis equal;
```



3 (j) Answer question

The pole-zero plot shows us that this system is a high-pass filtering system. This is particularly easy to know because all the zeros are situated around the 0 radian line. The system is FIR because there is only a single pole at z = 0. Additionally, the system is stable because the poles are all inside the unit circle.

3 (k) Submit file on Canvas

```
audiowrite('q2.wav', y / abs(max(y)), fs);
save('q3.mat', 'b2', 'a2');
```

```
Warning: Data clipped when writing file.
```

ALL FUNCTIONS SUPPORTING THIS CODE %% function pzplot(b,a)

PZPLOT(B,A) plots the pole-zero plot for the filter described by vectors A and B. The filter is a "Direct Form II Transposed" implementation of the standard difference equation:

```
a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                      - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```

```
    % MODIFY THE POLYNOMIALS TO FIND THE ROOTS
    b1 = zeros(max(length(a),length(b)),1); % Need to add zeros to get the right roots
    a1 = zeros(max(length(a),length(b)),1); % Need to add zeros to get the right roots
    b1(1:length(b)) = b;    % New a with all values
    a1(1:length(a)) = a;    % New a with all values

    % FIND THE ROOTS OF EACH POLYNOMIAL AND PLOT THE LOCATIONS OF THE ROOTS
    h1 = plot(real(roots(a1)), imag(roots(a1)));
    hold on;
    h2 = plot(real(roots(b1)), imag(roots(b1)));
    hold off;

    % DRAW THE UNIT CIRCLE
    circle(0,0,1)

    % MAKE THE POLES AND ZEROS X's AND O's
    set(h1, 'LineStyle', 'none', 'Marker', 'x', 'MarkerFaceColor','none', 'linewidth', 1.5, 'markersize', 8);
    set(h2, 'LineStyle', 'none', 'Marker', 'o', 'MarkerFaceColor','none', 'linewidth', 1.5, 'markersize', 8);
    axis equal;

    % DRAW VERTICAL AND HORIZONTAL LINES
    xminmax = xlim();
    yminmax = ylim();
    line([xminmax(1) xminmax(2)],[0 0], 'linestyle', ':', 'linewidth', 0.5, 'color', [1 1 1]*.1)
    line([0 0],[yminmax(1) yminmax(2)], 'linestyle', ':', 'linewidth', 0.5, 'color', [1 1 1]*.1)

    % ADD LABELS AND TITLE
    xlabel('Real Part')
    ylabel('Imaginary Part')
    title('Pole-Zero Plot')

end


function circle(x,y,r)
% CIRCLE(X,Y,R)  draws a circle with horizontal center X, vertical center
% Y, and radius R.
%

    % ANGLES TO DRAW
    ang=0:0.01:2*pi;

    % DEFINE LOCATIONS OF CIRCLE
    xp=r*cos(ang);
    yp=r*sin(ang);

    % PLOT CIRCLE
    hold on;
    plot(x+xp,y+yp, ':', 'linewidth', 0.5, 'color', [1 1 1]*.1);
    hold off;

end



function H = DTFT(x,w)
% DTFT(X,W)  compute the Discrete-time Fourier Transform of signal X
% acroess frequencies defined by W.

    H = zeros(length(w),1);
    for nn = 1:length(x)
        H = H + x(nn).*exp(-1j*w.'*(nn-1));
    end

end
```

```
function xs = shift(x, s)
% SHIFT(x, s) shifts signal x by s such that the output can be defined by
% xs[n] = x[n - s]

    % INITIALIZE THE OUTPUT
    xs = zeros(length(x), 1);

    for n = 1:length(x)
        % CHECK IF THE SHIFT IS OUT OF BOUNDS FOR THIS SAMPLE
        if n-s > 0 && n-s < length(x)
            % SHIFT DATA
            xs(n) = x(n-s);
        end
    end

end



function [b,a] = pz2ba(p,z)
% PZ2BA(P,Z)  Converts poles P and zeros Z to filter coefficients
%             B and A
%
% Filter coefficients are defined by:
%    a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
%                          - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
%

    % CONVERT ROOTS (POLES AND ZEROS) INTO POLYNOMIALS
    b = poly(z);
    a = poly(p);

end


function [p,z] = ba2pz(b,a)
% BA2PZ(B,A)  Converts filter coefficients B and A into poles P and zeros Z
%
% Filter coefficients are defined by:
%    a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
%                          - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
%

    % MODIFY THE POLYNOMIALS TO FIND THE ROOTS
    b1 = zeros(max(length(a),length(b)),1); % Need to add zeros to get the right roots
    a1 = zeros(max(length(a),length(b)),1); % Need to add zeros to get the right roots
    b1(1:length(b)) = b;    % New a with all values
    a1(1:length(a)) = a;    % New a with all values

    % FIND THE ROOTS OF EACH POLYNOMIAL AND PLOT THE LOCATIONS OF THE ROOTS
    p = real(roots(a1))+1j*imag(roots(a1));
    z = real(roots(b1))+1j*imag(roots(b1));

end
```