[Connor Dupuis]

Table of Contents

[Friday 1:55pm] - [28944] - [Naoki Sawahashi]

QUESTION 1 COMMENTING

DO NOT REMOVE THE LINE BELOW MAKE SURE 'eel3135_lab08_comment.m' IS IN THE SAME DIRECTORY AS THIS FILE

```
clc;type('eel3135_lab08_comment.m')


%% QUESTION #1 COMMENTING
clear
close all
clc

%% DEFINE FILTER AND INPUT
w = -pi:pi/8000:pi-pi/8000;
N = 100;
n = 0:(N-1);

% FILTER 1
```

```
a1 = [1 0 0 0 0 0 0 -0.9];
b1 = 1;
% <-- Answer: Is this an IIR filter or a FIR filter? Why?
% This is an IIR filter because it has poles that are non zero.
% <-- Answer: How many poles does this system have? How many zeros?
% This system has 7 poles

% FILTER 2
a2 = [1 -0.9];
b2 = 1;
% <-- Answer: Is this an IIR filter or a FIR filter? Why?
% This is an IIR filter because it has poles that are non zero.
% <-- Answer: How many poles does this system have? How many zeros?
% This system has 1 poles

% INPUT
x = zeros(N,1);
x(1) = 1;

%% DEFINE AND PLOT OUTPUT

% OUTPUT 1
y1 = filter(b1,a1,x);
y2 = filter(b2,a2,x);

% OUTPUT 2
H1 = DTFT(y1,w);
H2 = DTFT(y2,w);

% OUTPUT 3: CASCADE FILTERS
y3 = filter(b2, a2, filter(b1, a1, x));
H3 = DTFT(y3,w);
% <-- Express H3(z) as a function of H1(z) and H2(z)
% H3(z) = H1(z) * H2(z)


% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure(1)
subplot(3,1,1)
stem(n,y1)
xlabel('Time (Samples)')
ylabel('h[n]')
subplot(3,1,2)
plot(w,abs(H1))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(3,1,3)
pzplot(b1,a1)
axis equal

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure(2)
subplot(3,1,1)
```

```
stem(n,y2)
xlabel('Time (Samples)')
ylabel('h[n]')
subplot(3,1,2)
plot(w,abs(H2))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(3,1,3)
pzplot(b2,a2)
axis equal

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure(3)
subplot(2,1,1)
stem(n,y3)
xlabel('Time (Samples)')
ylabel('h[n]')
subplot(2,1,2)
plot(w,abs(H3))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')




%% ALL FUNCTIONS SUPPORTING THIS CODE %%
% ==================================================================
% NOTE: YOU DO NOT NEED TO ADD COMMENTS IN THE CODE BELOW. WE JUST
% NEEDED POLE-ZERO PLOTTING CODE AND THUS WROTE IT.
% ==================================================================
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
    b  = b(1:find(b,1,'last'));
    a  = a(1:find(a,1,'last'));
    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
% across frequencies defined by W.
```

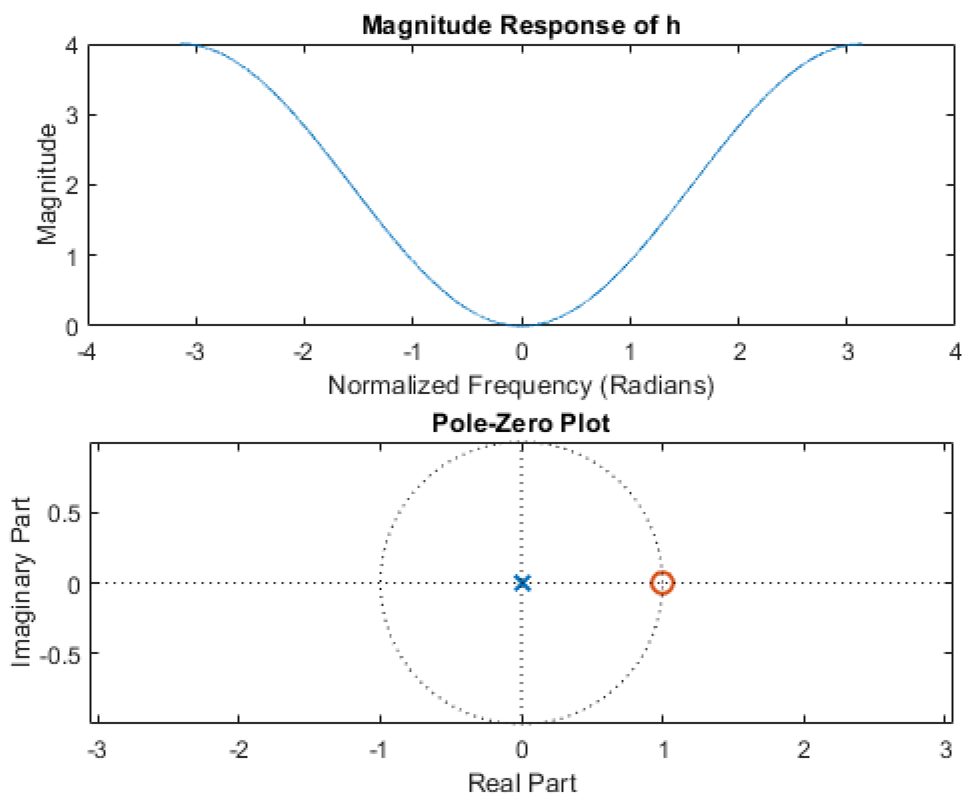
```
    H = zeros(length(w),1);
    for nn = 1:length(x)
        H = H + x(nn).*exp(-1j*w.'*(nn-1));
    end

end
```

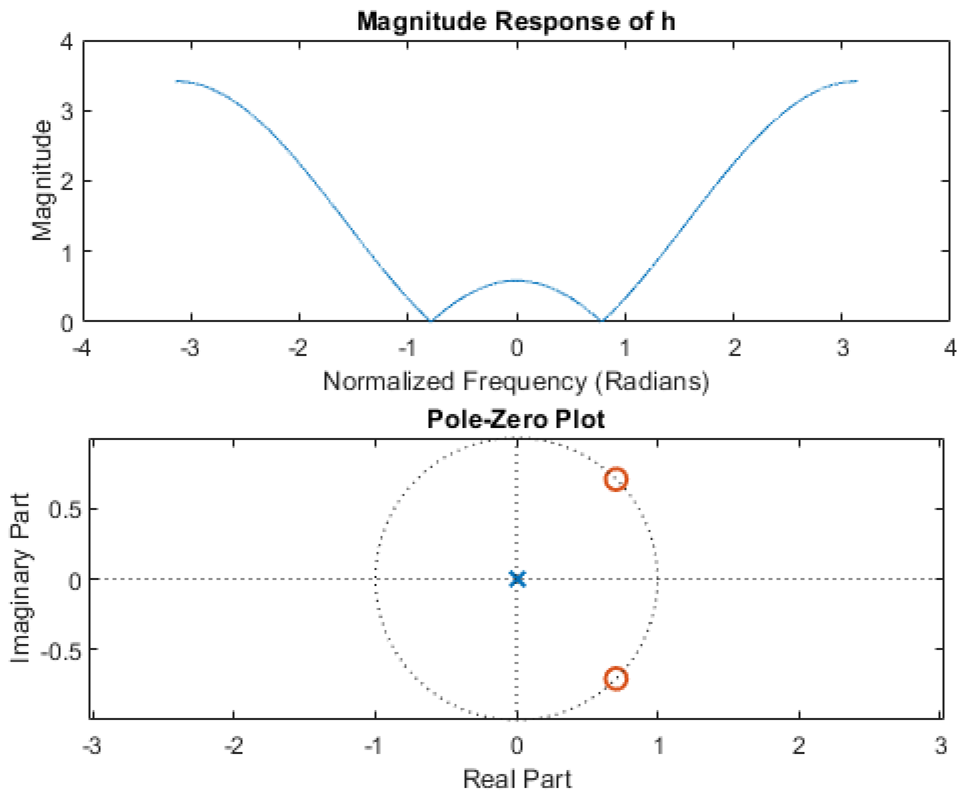
QUESTION 2: Z-TRANSFORM

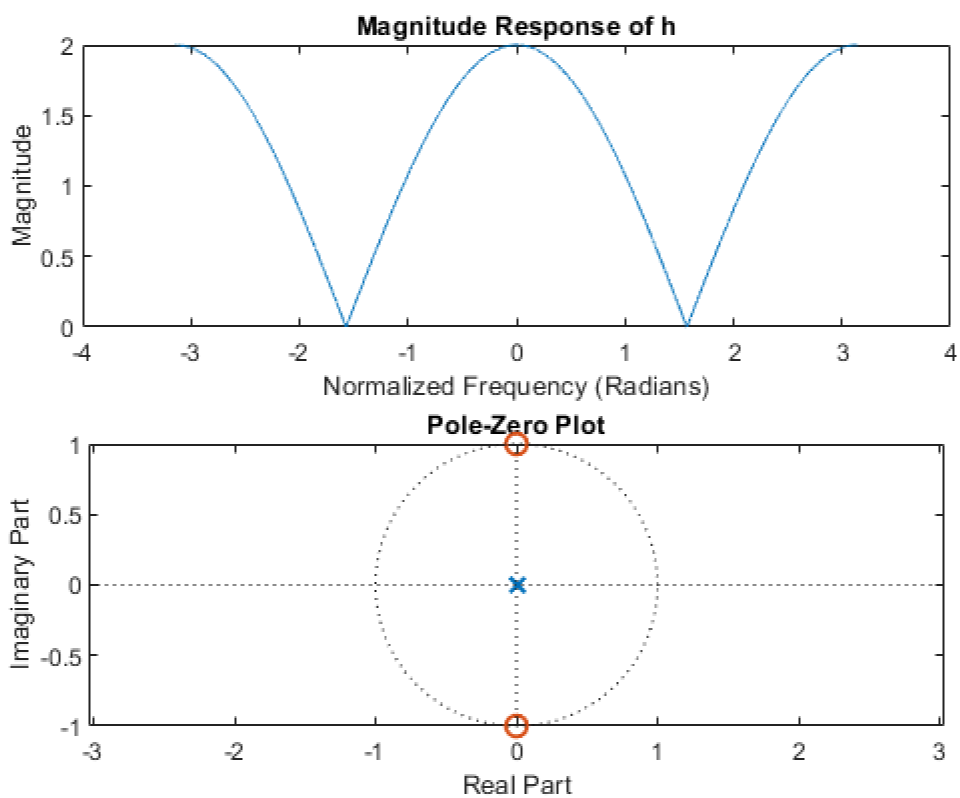
2 (a) FILTER 1

```
% DEFINE AXES
w = -pi:pi/8000:pi-pi/8000;
N = 100;

% INPUT
x = zeros(N,1);
x(1) = 1;

% FILTER 1
b1 = [1 -2 1];
a1 = 1;
y11 = filter(b1,a1,x);
H1 = DTFT(y11,w);

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b1,a1)
axis equal

% FILTER 2
b1 = [1 -sqrt(2) 1];
a1 = 1;
y12 = filter(b1,a1,x);
H1 = DTFT(y12,w);

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b1,a1)
axis equal
```

```
% FILTER 3
b1 = [1 0 1];
a1 = 1;
y13 = filter(b1,a1,x);
H1 = DTFT(y13,w);


% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b1,a1)
axis equal
```



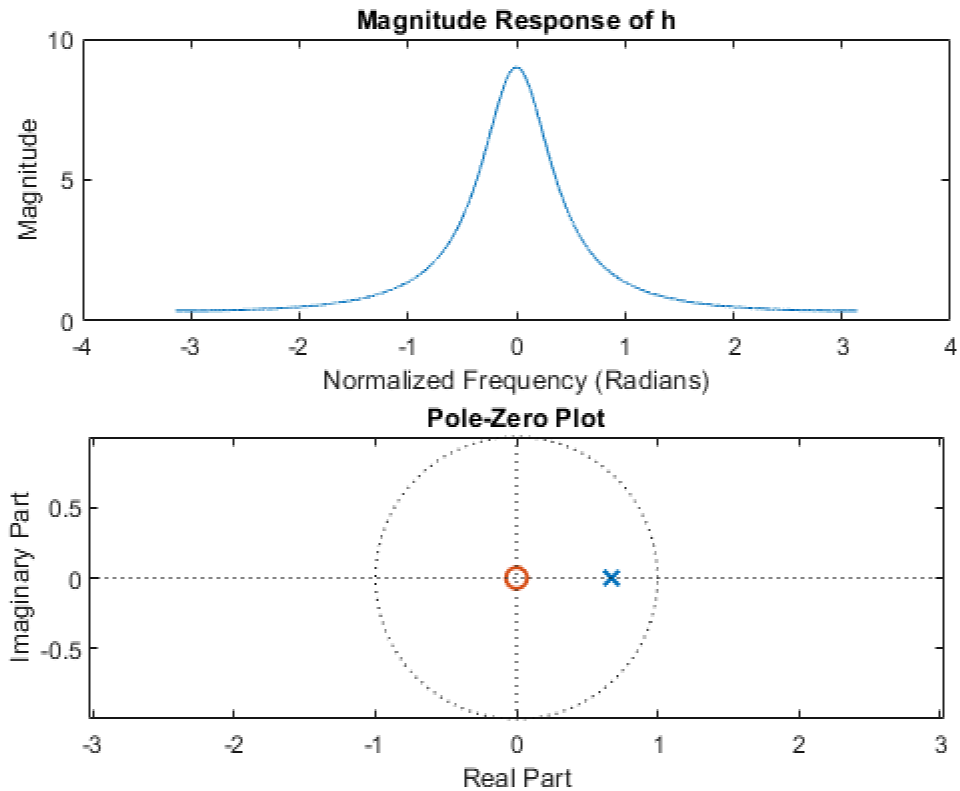

Magnitude Response of h
Magnitude
Normalized Frequency (Radians)
Pole-Zero Plot
Imaginary Part
Real Part

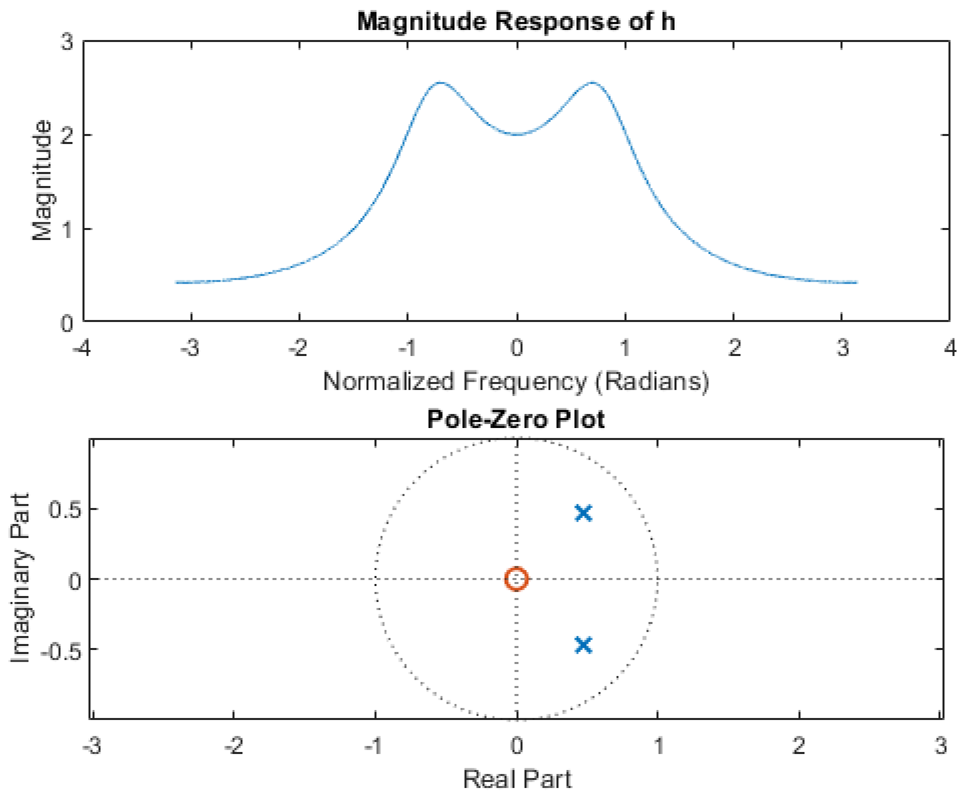

Magnitude Response of h
Magnitude
Normalized Frequency (Radians)
Pole-Zero Plot
Imaginary Part
Real Part

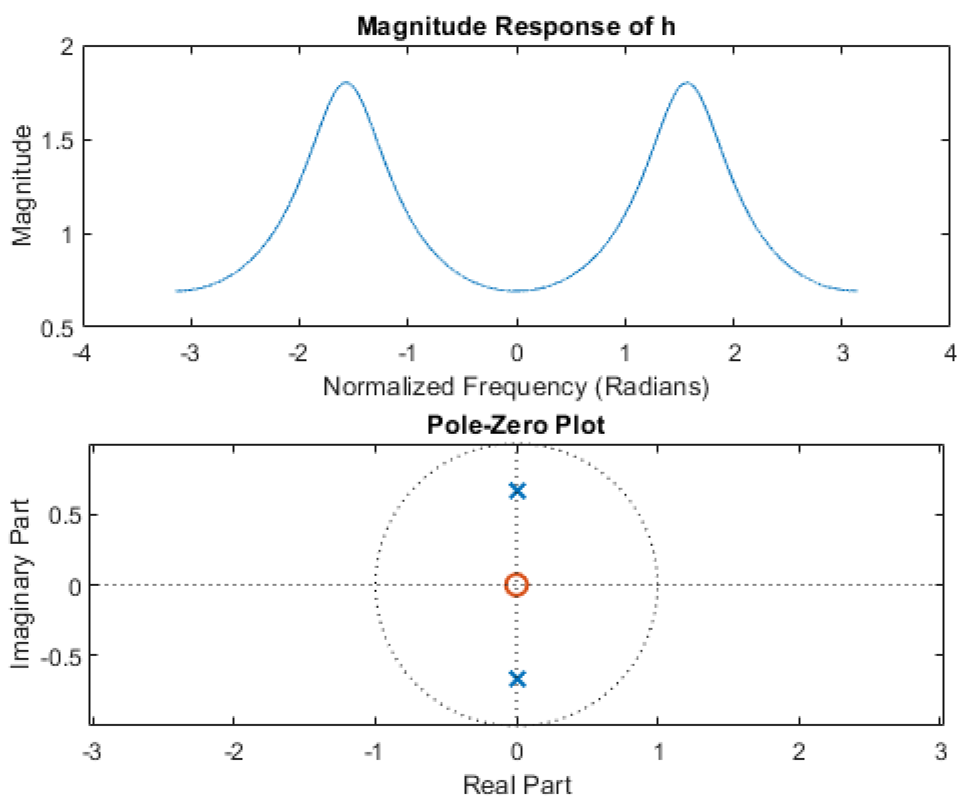
2 (b) FILTER 2

```
% DEFINE AXES
w = -pi:pi/8000:pi-pi/8000;
N = 100;

% INPUT
x = zeros(N,1);
x(1) = 1;

% FILTER 1
a1 = [1 -4/3 4/9];
b1 = 1;
y21 = filter(b1,a1,x);
H1 = DTFT(y21,w);


% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b1,a1)
axis equal

% FILTER 2

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
a1 = [1 -2*sqrt(2)/3 4/9];
b1 = 1;
y22 = filter(b1,a1,x);
H1 = DTFT(y22,w);

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b1,a1)
axis equal

% FILTER 3
a1 = [1 0 4/9];
b1 = 1;
y23 = filter(b1,a1,x);
H1 = DTFT(y23,w);
```

```
% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(H1))
title('Magnitude Response of h')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
pzplot(b1,a1)
axis equal
```




Magnitude Response of h
Magnitude
Normalized Frequency (Radians)
Pole-Zero Plot
Imaginary Part
Real Part


Magnitude Response of h
Magnitude
Normalized Frequency (Radians)
Pole-Zero Plot
Imaginary Part
Real Part

3 (a) ANSWER QUESTION

```
% Ha(Z) at w0 = 0 is a highpass filter
% Ha(Z) at w0 = pi/4 is a highpass filter
% Ha(Z) at w0 = pi/2 is a notch filter

% Hb(Z) at w0 = 0 is a lowpass filter
% Hb(Z) at w0 = pi/4 is a lowpass filter
% Hb(Z) at w0 = pi/2 is a bandpass filter
```

3 (b) ANSWER QUESTION

```
% Ha(Z) is a FIR filter
% Hb(Z) is a IIR filter
```

3 (c) ANSWER QUESTION

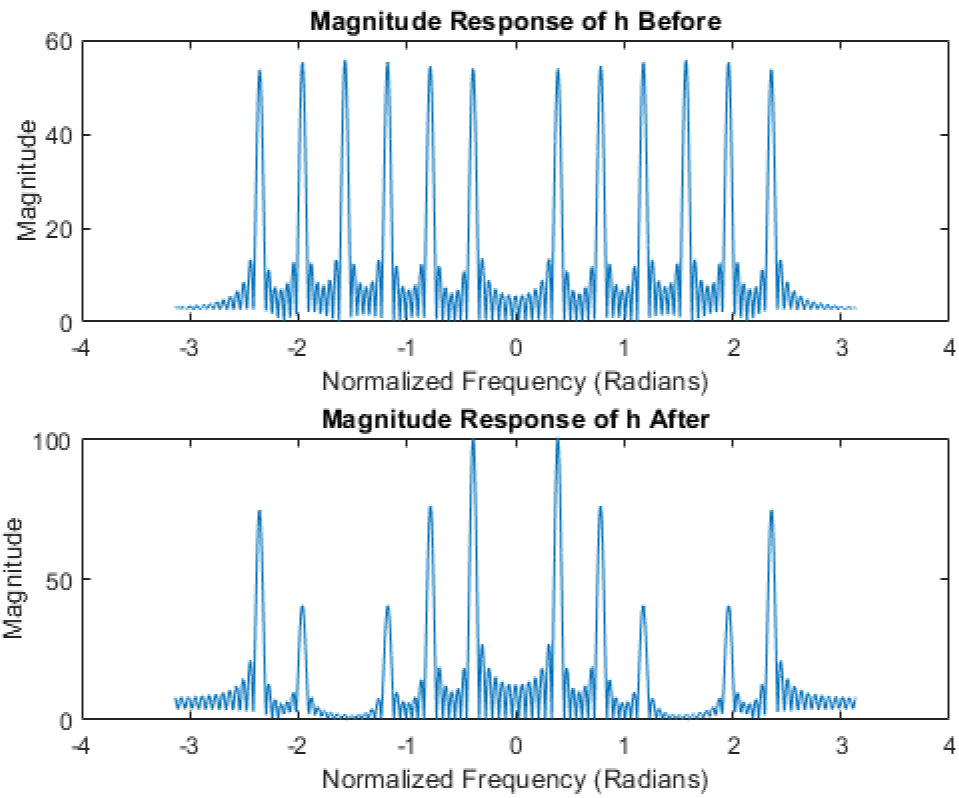
```
% Changing w0 affetcs where the poles/zeros are regarding the unit
 circle
```

QUESTION 4: IIR FILTERS IN Z-DOMAIN

```
N = 100;
n = 0:N-1;
x = cos( pi/8*n ) + cos( 3*pi/8*n ) + cos( pi/4*n ) + cos( 5*pi/8*n )
 + cos( pi/2*n ) + cos( 3*pi/4*n );
```

4 (a) APPLY NULLING FILTER

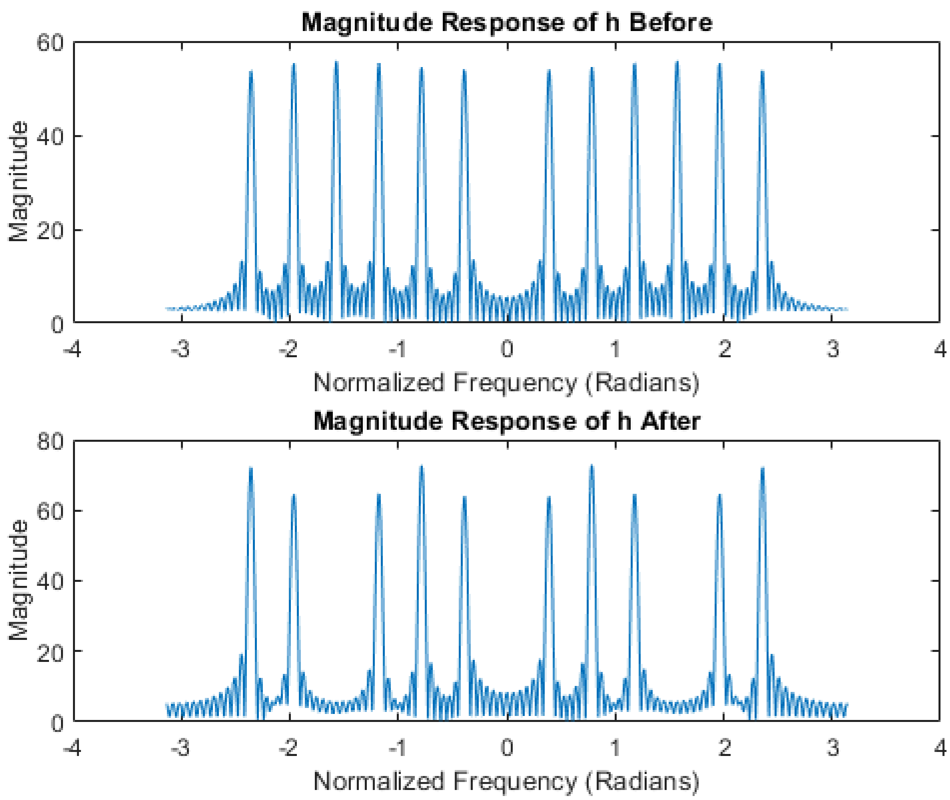
```
% Ha(Z) pi/2
b1 = [1 0 1];
a1 = 1;

y4a = filter(b1,a1,x);
Hb = DTFT(x,w);
Ha = DTFT(y4a,w);

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(Hb))
title('Magnitude Response of h Before')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
plot(w,abs(Ha))
title('Magnitude Response of h After')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
```



4 (b) NEW NULLING FILTER

```
% Ha(Z) pi/2
b1 = [1 0 1];
a1 = 1;

% Hb(Z) 3pi/8
a2 = [1 -0.510245 4/9];
b2 = 1;

% Hb(Z) 5pi/8
a3 = [1 0.510245 4/9];
b3 = 1;

% Convoluting the three filters with x
y4b = filter(b1, a1, filter(b2, a2, filter(b3, a3, x)));

Hb = DTFT(x,w);
Ha = DTFT(y4b,w);

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(Hb))
title('Magnitude Response of h Before')
```

```
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
plot(w,abs(Ha))
title('Magnitude Response of h After')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
```



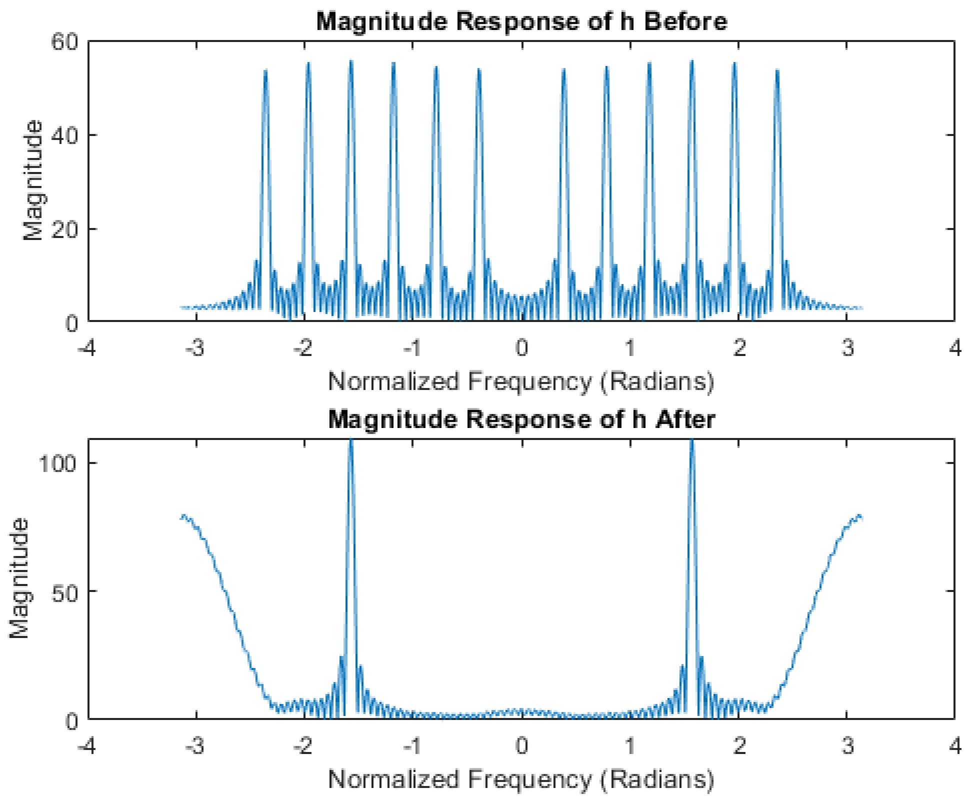
4 (c) ANSWER QUESTION

```
% It improves on the filter in (a) by bringing up the frequencies
 around
% pi/2. The first filter eliminated pi/2 but also the frequencies
 around
% it, this filter is moderately better at just eliminating pi/2.
```

4 (d) NULLING MULTIPLE FREQUENCIES

```
% Ha(Z) pi/8
b1 = [1 -1.84776 1];
a1 = 1;

% Ha(Z) 3pi/8
b2 = [1 -0.765367 1];
a2 = 1;
```

```
% Ha(Z) pi/4
b3 = [1 -sqrt(2) 1];
a3 = 1;

% Ha(Z) 5pi/8
b4 = [1 0.765367 1];
a4 = 1;

% Ha(Z) 3pi/4
b5 = [1 sqrt(2) 1];
a5 = 1;

% Ha(Z) pi (Not used)
% b6 = [1 2 1];
% a6 = 1;

% Convoluting the three filters with x
y4d = filter(b1, a1, filter(b2, a2, filter(b3, a3, filter(b4, a4,
 filter(b5, a5, x)))));

Hb = DTFT(x,w);
Ha = DTFT(y4d,w);

% PLOT THE FREQUENCY REPONSE IMPULSE RESPONSE AND DTFT
figure
subplot(2,1,1)
plot(w,abs(Hb))
title('Magnitude Response of h Before')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
plot(w,abs(Ha))
title('Magnitude Response of h After')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
```



4 (e) PLAY SOUND

```
N = 1000;
n = 0:N-1;
x = [cos( pi/8*n ) cos( 3*pi/8*n ) cos( pi/4*n ) cos( 5*pi/8*n )
 cos( pi/2*n ) cos( 3*pi/4*n )];
```

Create input audio file

```
soundsc(x,2000)
audiowrite('input_sound.wav', x, 2000)
```

Create output audio file

```
y = filter(b1, a1, filter(b2, a2, filter(b3, a3, filter(b4, a4,
 filter(b5, a5, x)))));
y_out = y/max(abs(y));

soundsc(y_out,2000)
audiowrite('output_sound.wav', y_out, 2000)
```

QUESTION 5: DESIGNING A BETTER NULLING FILTER

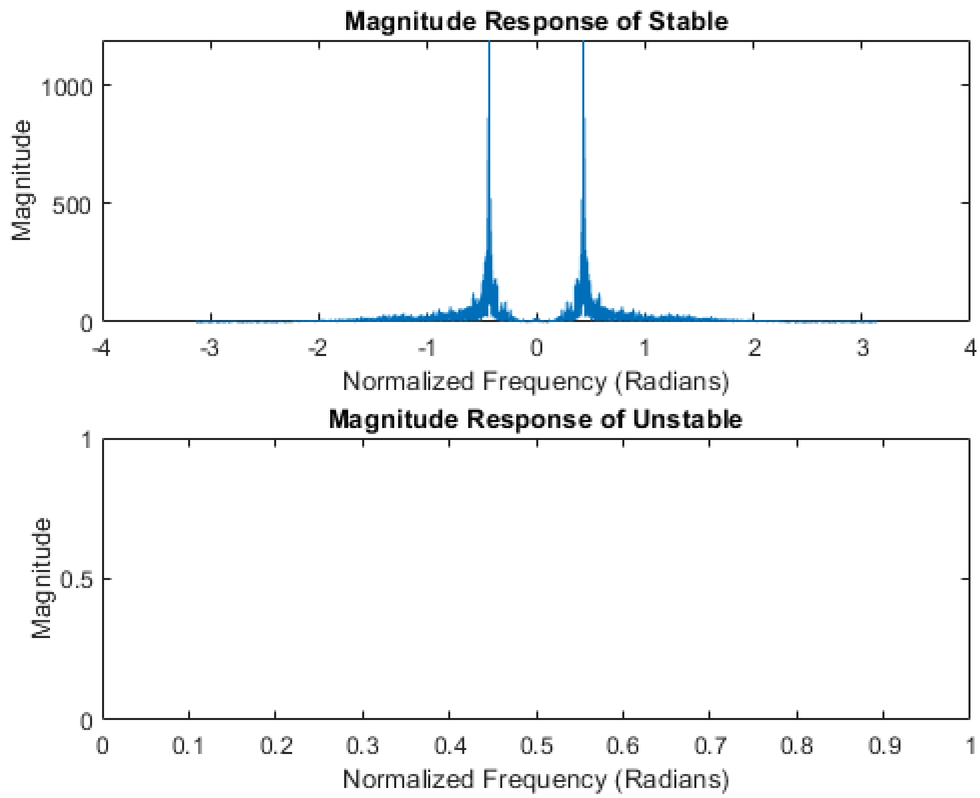
```
[x ,fs] = audioread('Noisy.wav');
w = -pi:pi/5000:pi;
X = DTFT(x, w);
```

5 (a) ANSWER QUESTION

```
% This filter would be stable with alpha < 1
```

5 (b) PLOT POLE-ZERO PLOTS

```
%w0 = 0.08922;

% H(Z) Stable alpha = 0.90
b1 = [1 -1.99205 1];
a1 = [1 -1.79284 .9801];

y1 = filter(b1,a1,x);
Hstable = DTFT(y1,w);

% H(Z) Unstable aplha = 1.1
b2 = [1 -1.99205 1];
a2 = [1 -2.19125 1.21];

y2 = filter(b2,a2,x);
Hunstable = DTFT(y2,w);

figure
subplot(2,1,1)
plot(w,abs(Hstable))
title('Magnitude Response of Stable')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')
subplot(2,1,2)
plot(w,abs(Hunstable))
title('Magnitude Response of Unstable')
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')

%Hunstable = DTFT(y,w);
```



5 (c) ANSWER QUESTION

```
% The larger the alpha value the sharper the notch
```

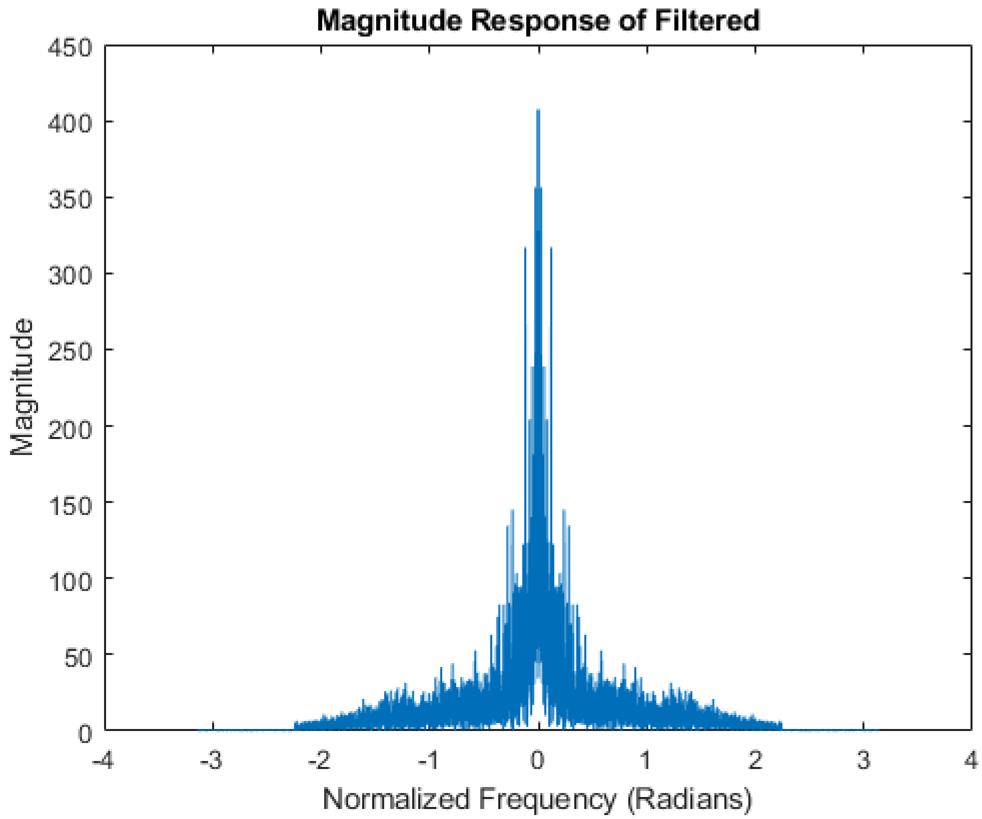
5 (d) ANSWER QUESTION

```
% The difference equation is y[n] - 1.97212*y[n-1] + 0.9801*y[n-2] =
 x[n] -
% 1.99205*x[n-1] + x[n-2]
% y[n] = 1.97212*y[n-1] - 0.9801*y[n-2] + x[n] - 1.99205*x[n-1] +
 x[n-2]
```

5 (e) - APPLYING FILTER AND PLOTTING MAGNITUDE RESPONSES

```
b1 = [1 -1.99205 1];
a1 = [1 -1.97212 .9801];
y_unscaled = filter(b1, a1, x);
Hfiltered = DTFT(y_unscaled,w);

figure
plot(w,abs(Hfiltered))
title('Magnitude Response of Filtered')
```

```
xlabel('Normalized Frequency (Radians)')
ylabel('Magnitude')

y = y_unscaled/max(abs(y_unscaled));
```



Play sound and create audio file

If most/a majority of the noise was filtered out, then you created the filter correctly

```
soundsc(y, fs);
audiowrite('Filtered_song.wav', y/max(abs(y)), fs);
```

ALL FUNCTIONS SUPPORTING THIS CODE % %

=== NOTE: YOU DO NOT NEED TO ADD COMMENTS IN THE CODE BELOW. WE JUST NEEDED POLE-ZERO PLOTTING CODE AND THUS WROTE IT. ===

```
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
% across 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
```

Published with MATLAB® R2020a