ECE 158 Digital Signal Processing Lab

Digital Filtering

Student : Corey Gravelle
Perm Number: 8094211
E-mail: cgravelle@umail.ucsb.edu

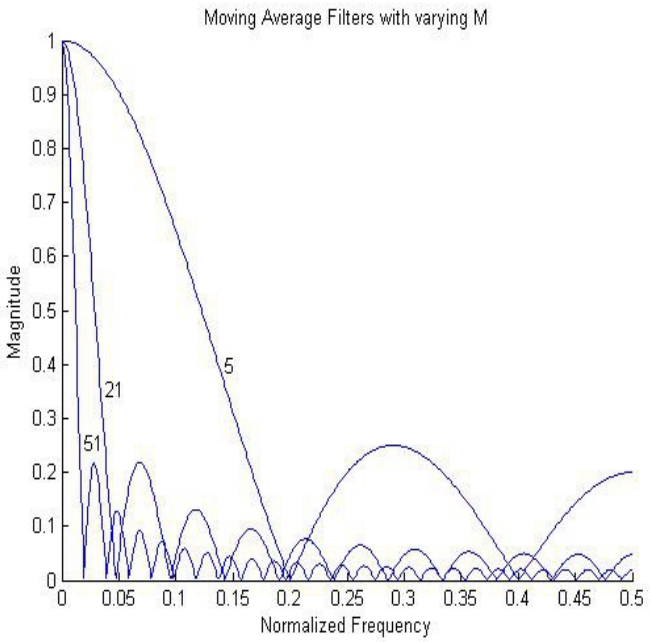
11/7/2009

Department of Electrical and Computer Engineering, UCSB. ECE158

Introduction:
This lab is an introduction to the principles of simulating various digital filters with MATLAB, such as the Moving Average Filter, Lowpass Filter, Highpass Filter, and Allpass filter. One may explore their properties such as group delay, frequency amplitude and phase response, to find the ideal use for each.

1.1.Moving Average Filter
Code:Graph:

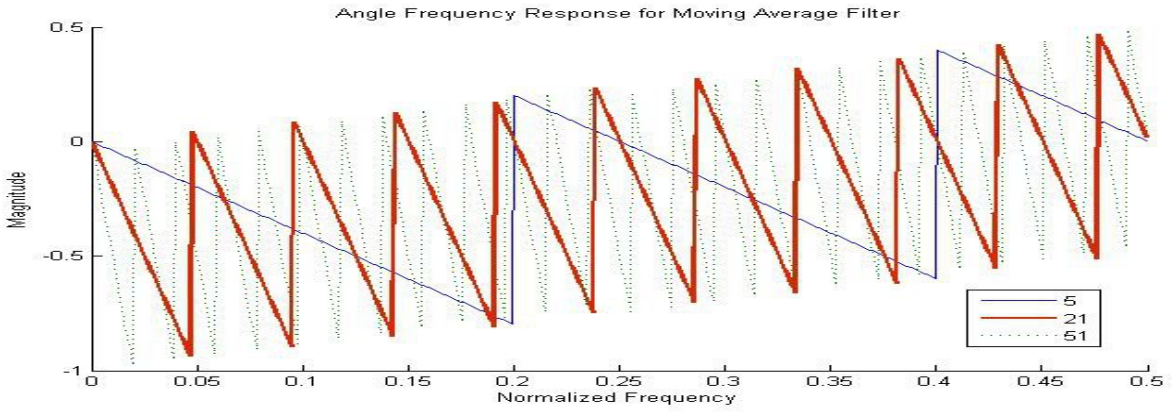
```
 function [] = movavgfxn ()

%Constants
M=[5, 21, 51];     %M = filter width
N = 1024;          %N = number of points

%Plotting fft of impulse responses:
figure
hold on
for M = [5, 21, 51];
    Imp = ones(1,M)./M%impulse responses
    impfft = abs(fft(Imp,1024));
    freq = linspace(0,1,N) ;
    plot(freq(1:N/2), impfft(1:N/2));
end
ylabel('Magnitude');
xlabel('Normalized Frequency');
gtext('5')
gtext('21')
gtext('51')
```



1.2*What happens to the passband of the moving average filter—those frequencies it lets pass with only slight attenuation—as the order M increases? What single frequency passes completely unaltered in all three filters?*

As the order of M increases, the passband shrinks; as M grows from 5 to 51, the passband shrinks from 0.15 to 0.03 times the normalized frequency. The frequency of 0 Hz passes unattenuated with all three filters, translating to a steady-state error of zero.

1.3

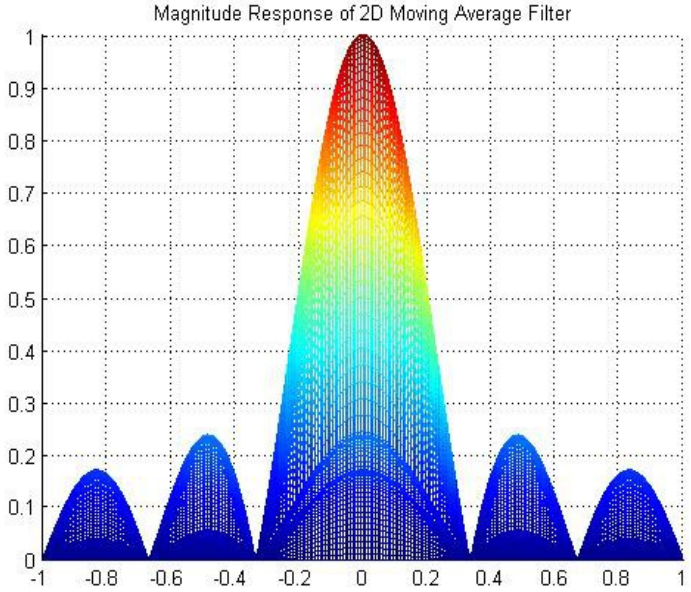


1.4*Can you explain why the phase response has a jagged appearance?*

At all frequencies which the magnitude response reached zero, the phase response jumped up pi radians; because this is calculated by taking the inverse tangent of the response, whenever it reaches +/- pi radians, this number assumes it is positive. To be more accurate, the phase angle should continue to decrease with a constant slope, but for slope visualization purposes, this graph is sufficient.

1.5

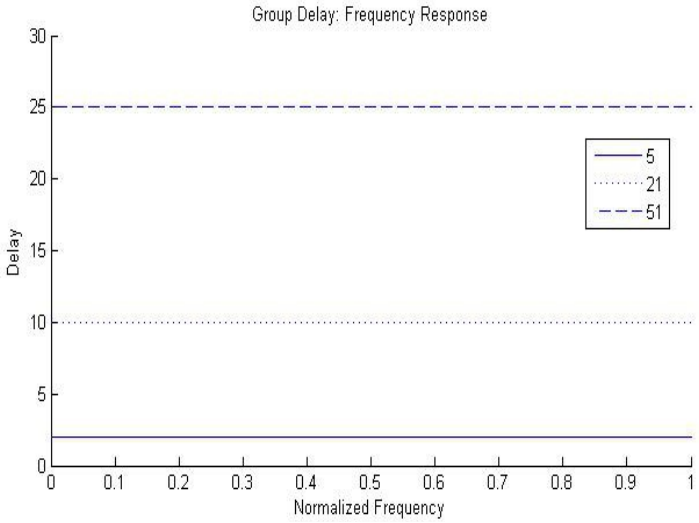
```
function [] = movavg2d ()

%Constants
N = 1024;                   %N = num-
ber of points
M = 6;

h = ones(M,M)./M^2;
[H,Fx,Fy] = freqz2(h,[256 256]);
mesh(Fx,Fy,abs(H));
AZ = 0;
EL = 0 ;
view(AZ, EL);
title('Magnitude Response of 2D
Moving Average Filter')
```



Both this graph and that from the 1-dimensional filter with M=5 in problem 1.1 are low-pass filters with fairly wide passbands.

1.3 (contd) Group Delay code and graph:

```
function [] = groupie ()

%Constants
M=[5, 21, 51];
N = 1024;

%Plotting fft of impulse responses:
figure
hold on
for M = [5, 21, 51];
    Imp = ones(1,M)./M;
    impfft = angle(fft(Imp,1024))/pi;
    g = grpdelay(ones(1,M)./M, 1,
1024, 'whole',1);
    freq = linspace(0,1,N) ;
    plot(freq, g(:,1));
end
ylabel('Delay');
xlabel('Normalized Frequency');
title('Group Delay: Frequency Response');
legend('5','21','51');
axis([0 1 0 30]);
```



1.6 Plotting the group delay, one may notice that it is not frequency-dependent. This is evident in the plot of the phase delay from 1.3; as the slopes are not changing, neither should the group delay. With this case, the delay is (M-1)/2 samples.

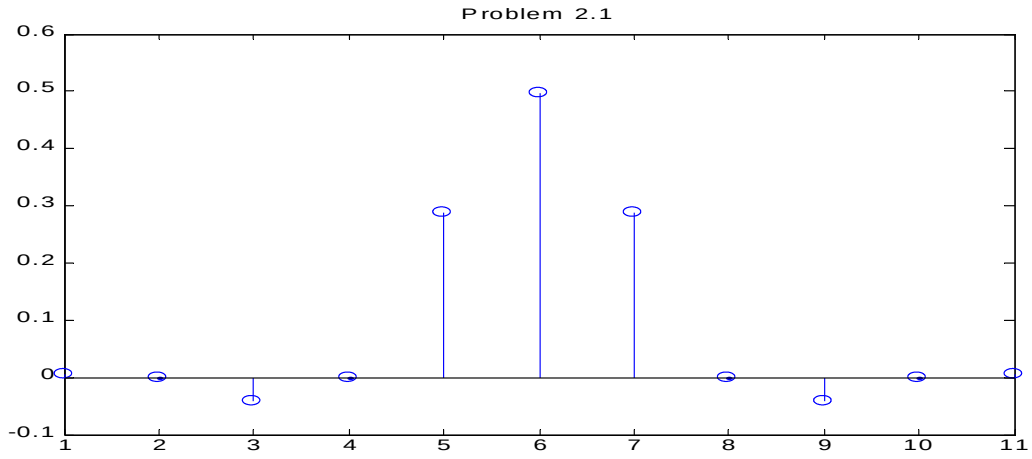
1.7 Code for combining two audio signals:

```
%%  Corey Gravelle      ECE 158         Lab 4: Digital Filtering
%   A function for combining two sound waves into a mono output waveform,
%   as with an impulse response of a room with an audio file

function [out] = zpad(sound1,sound2);

s1 = wavread(sound1);
s2 = wavread(sound2);
s1L = length(s1);
s2L = length(s2);

%test for stereo or mono
s1wid = numel(s1)/s1L
s2wid = numel(s2)/s2L
%convert to mono
if s1wid > 1
    s1 = (s1(:,1)+s1(:,2))./2;
end
if s2wid > 1
    s2 = (s2(:,1)+s2(:,2))./2;
end

z1 = zeros(length(s2)-1,1);
z2 = zeros(length(s1)-1,1);
sz1 = [s1;z1];
sz2 = [s2;z2];
s1fft = fft(sz1);
s2fft = fft(sz2);

comb = s1fft.*s2fft;
out = ifft(comb);
soundsc(out,44100);
```

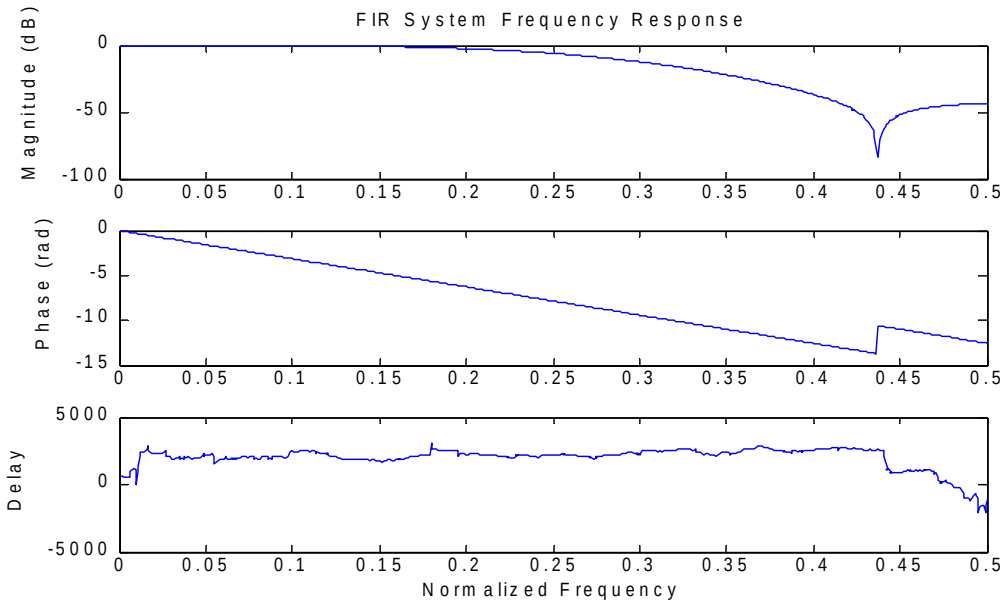
1.8*Listen to the results and explain what just happened and pen some awe inspired words of your experience of music in Killaloe Cathedral.*

The effect of combining the echo of a room with an audio signal is essentially the sound heard by playing that signal live inside the room. Or, to play the sound through the instrument that is the room. It doesn't matter which order one puts the sound files in or if they are stereo or mono signals. This technique may be used to combine any two audio signals and is not only limited to an echo characteristic and a sound file.

2.1



2.2 Graph and code for FIR Impulse Response:




```
function [] = part2();
%Given:
N=11;
graph = 0;
h = [0.00506, 0, -0.04194, 0, 0.28848, 0.49679,...
    0.28848, 0, -0.04194, 0, 0.00506];
hn = [0.00506, 0, -0.04194, 0, 0.28848, -0.49679,...
```

```
    0.28848, 0, -0.04194, 0, 0.00506];
ns = [0:N-1];

%Graph inpulse response:
if (graph == 1)
    stem(ns,h);
    xlabel('n')
    ylabel('h[n]')
    title('FIR System Impulse Response');
end

%Finding FFT:
hfft = fft(h,1024);
hdb = 20*log10(abs(hfft)./max(abs(hfft)));
hang = unwrap(angle(hfft));
ghfft = hfft(1:length(hfft)/2);
g = grpdelay(hfft, 1, 1024, 'whole',1);
mfreq = linspace(0,0.5,length(hdb)/2) ;
pfreq = linspace(0,0.5,length(hang)/2);
gfreq = linspace(0,0.5,length(g));

%Plotting mag, phase, group delay
subplot(3,1,1); plot(mfreq, hdb(1:length(hdb)/2));
title('FIR System Frequency Response');
ylabel('Magnitude (dB)');
subplot(3,1,2); plot(pfreq, hang(1:length(hang)/2));
ylabel('Phase (rad)');
subplot(3,1,3); plot(gfreq, g(:,1));
ylabel('Delay');
xlabel('Normalized Frequency');
```

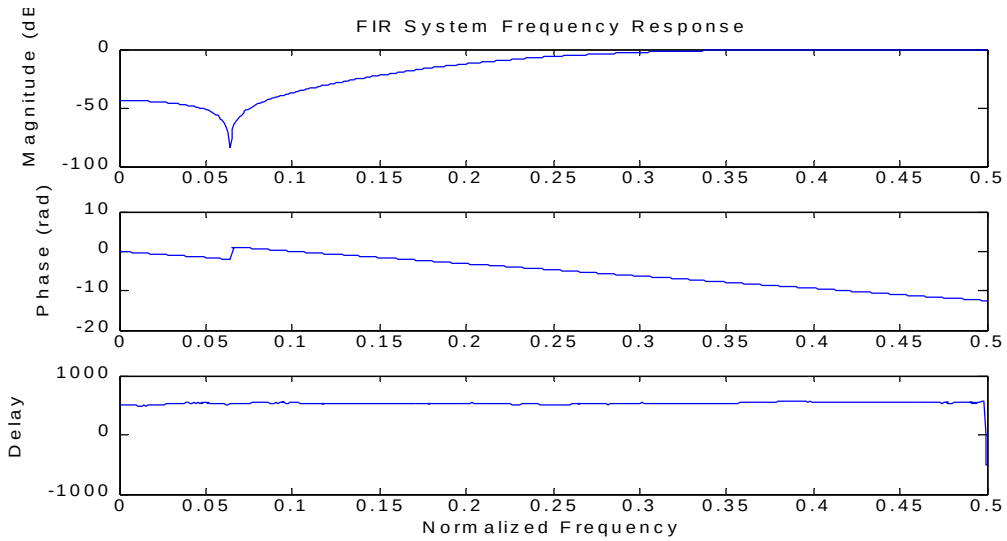
2.3 *What kind of filter is this? Find the "3 dB" point, or the normalized frequency above which the attenuation is greater than 3 dB.*

This is a low-pass filter, as it has minimal attenuation over low frequencies and maximum over high frequencies. By finding in the data set where the absolute value of the fourier transform (in decibels) is as close to -3 as possible and relating this frequency location to the corresponding normalized frequency, I found the -3dB normalized cutoff frequency is approximately 0.2094. By observing the magnitude graph in part 2.2, one can see this is the point where the drop-off (or, attenuation) becomes significant.

2.4 What is the group delay, and would this destroy information contained in the envelope of a signal?

The group delay is 512, uniform over all frequency and time. This creates a phase shift which is inaudible to the human ear and therefore doesn't do anything to an audio signal.

2.5



2.6 *What kind of filter is this? What is the cut-off frequency now?*

With a negative center number, the filter becomes a high-pass filter with a cutoff frequency of about 0.2916. The group delay seems much larger, at least at around 0.5. As one can see, this is a high-pass filter because it attenuates values of low frequencies and passes those with high frequencies.

2.7 *Why is this happening?*

Multiplying the middle term by a negative number emphasizes the peripherals of the filter rather than the middle; essentially by convolving this over a signal one emphasizes the differences between two relative points in a signal and passes those with significant changes. Rather than amplifying gradual changes relative to sudden, this filter is a high-pass filter and amplifies (or doesn't attenuate) high-frequency signals, or signals with quick, sudden changes.

2.8

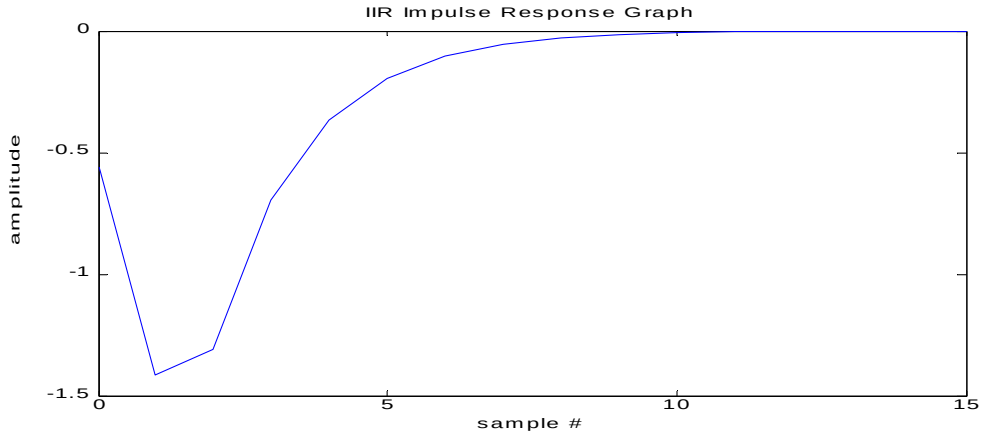
Code for applying filters to loops:

```
h1 = [0.00506, 0, -0.04194, 0, 0.28848,  0.49679,
      0.28848, 0, -0.04194, 0, 0.00506];
h2 = [0.00506, 0, -0.04194, 0, 0.28848, -0.49679,
      0.28848, 0, -0.04194, 0, 0.00506];
drum = wavread('drumloop2.wav');
drumono = (drum(:,1)+drum(:,2))./2;
drum1 = conv(h1,drumono);
drum2 = conv(h2,drumono);
soundsc(drum1,44100)
soundsc(drum2,44100)
```

After listening to the effects of the high-pass and low-pass filters, one can hear almost all high-frequency elements with the high-passed signal and almost all low frequencies with the low-passed signal. The low-pass signal is easier to hear and it seems to attenuate less of the signal than the high-pass signal.

2.9 The higher frequencies of the high-hat are attenuated using the low-pass filter from 2.2, and the bass drum and most things other than the high-hat ride are attenuated with the filter from 2.5.

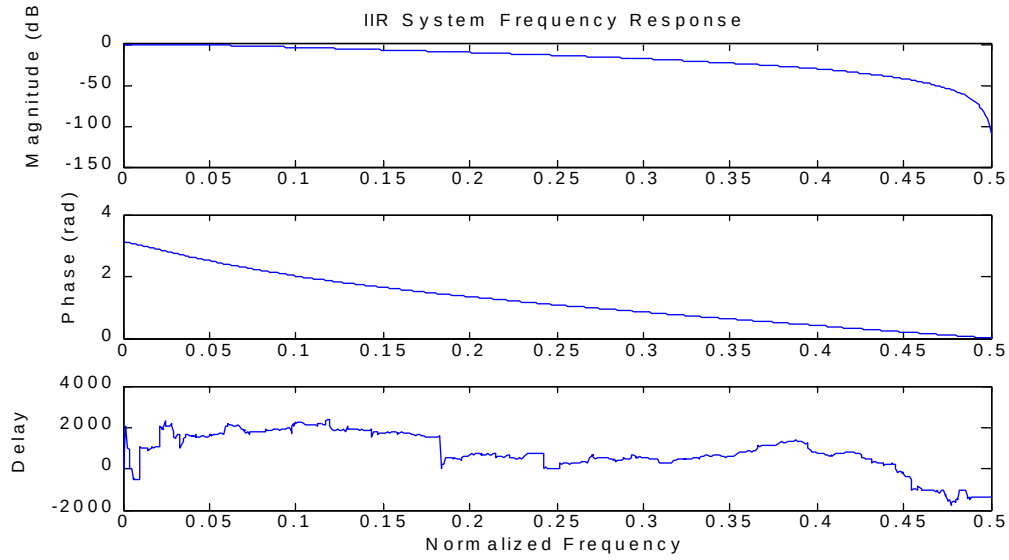
3.1



Code to graph Part 3 impulse response:

```
function [hdb,mfreq] = part3()

graphimp = 0;
graphfreq = 1;
imp = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
filta = filter([0.206572, 0.413144, 0.206572], [-0.369527,0.195816], imp);
time = [0:15];

% graph impulse response
if(graphimp == 1)
    plot(time,filta(1:16));
    ylabel ('amplitude');
    xlabel ('sample #')
    title('IIR Impulse Response Graph')
end

%frequency response:
freak = freqz(filta,1,1024,'whole',1);
hdb = 20*log10(abs(freak)./max(abs(freak)));
hang = unwrap(angle(freak));
ghfft = freak(1:length(freak)/2);
g = unwrap(grpdelay(freak, 1, 1024, 'whole',1));
mfreq = linspace(0,0.5,length(hdb)/2) ;
pfreq = linspace(0,0.5,length(hang)/2);
gfreq = linspace(0,0.5,length(g));

%Plotting mag, phase, group delay
if(graphfreq == 1)
    subplot(3,1,1); plot(mfreq, hdb(1:length(hdb)/2));
    title('IIR System Frequency Response');
    ylabel('Magnitude (dB)');
    subplot(3,1,2); plot(pfreq, hang(1:length(hang)/2));
    ylabel('Phase (rad)');
    subplot(3,1,3); plot(gfreq, g(:,1));
    ylabel('Delay');
    xlabel('Normalized Frequency');
end
```

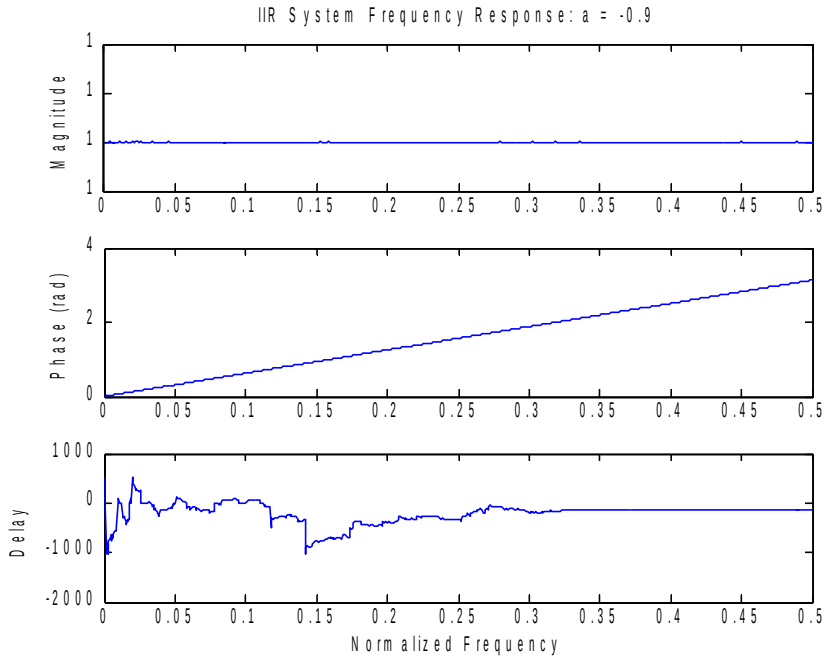
3.2



3.3 *Compare these results with those found for the FIR system described in 2.2. What are the differences? For instance, how do the cut-off frequencies compare? What is different about the group delay? How does the computational complexity of this filter compare to that of the FIR? What could be a problem if you use this filter on a carrier that has been modulated in amplitude?*

The normalized cutoff frequency of this IIR frequency response is about 0.08. This is once again a version of a low-pass filter; however, it seems to have a more continuous and gradual slope than the FIR response which explains the smaller bandwidth. The group delay seems to look similar to the overall shape of the original low-pass filter, but this version is much more unsteady. This filter was slightly more computationally complex, based on the freqz step in finding the frequency response.

3.4 *Using MATLAB, confirm that the transfer function given by equation (5) is an allpass filter for α = -0.9.*



This output shows that for a= 0.9 in:

$$A_0(z) = \frac{\alpha + z^{-1}}{1 + \alpha z^{-1}}$$

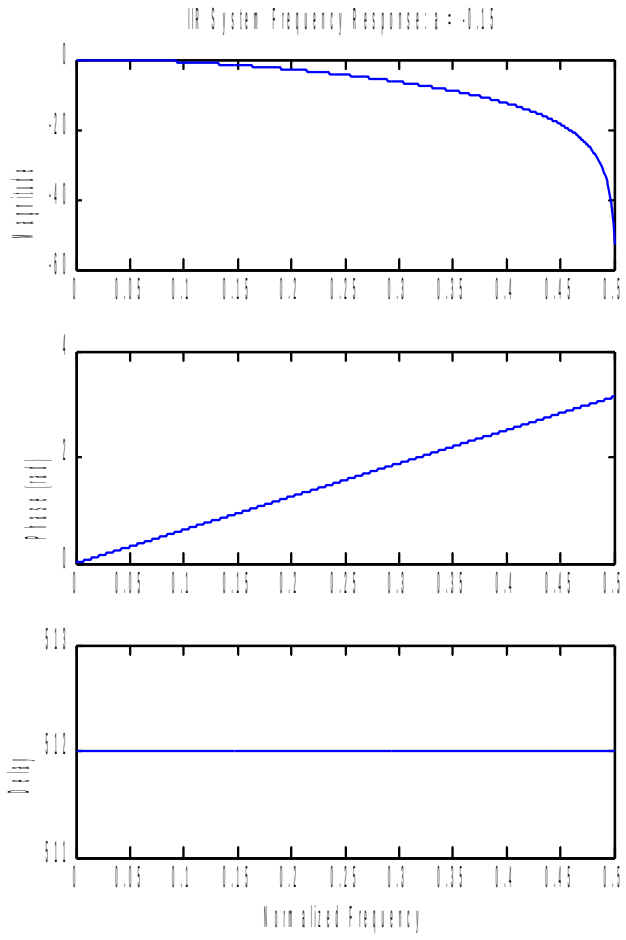
the magnitude response of the filter is 0, while the phase response differs depending on the normalized frequency.

(see next page for code)

```
function [hdb,mfreq] =
part3_4()
```

```
graphfreq = 1;

%frequency response:
a = -0.9;
[mag, ang] = freqz([a 1],[1 a],1024,'whole');
hang = unwrap(ang);
g = unwrap(grpdelay(mag, 1, 1024, 'whole',1));
mfreq = linspace(0,0.5,length(mag)/2) ;
pfreq = linspace(0,0.5,length(hang)/2);
gfreq = linspace(0,0.5,length(g));

%Plotting mag, phase, group delay
if(graphfreq == 1)
    subplot(3,1,1); plot(mfreq, abs(mag(1:length(mag)/2)));
    title(['IIR System Frequency Response: a = ',num2str(a)]);
    ylabel('Magnitude');
    subplot(3,1,2); plot(pfreq, hang(1:length(hang)/2));
    ylabel('Phase (rad)');
    subplot(3,1,3); plot(gfreq, g(:,1));
    ylabel('Delay');
    xlabel('Normalized Frequency');
end
```

3.5



```
function [sysmag,mfreq] = part3_5()

graphfreq = 1;
%frequency response:
a = -0.15;
[mag, ang] = freqz([(a+1)/2 (a+1)/2],[1
a],1024,'whole');
mag1 = 20*log10(abs(mag));
ang1 = unwrap(ang);
sysmag = mag1;
sysang = ang1;
gsys = unwrap(grpdelay(sysmag, 1, 1024,
'whole',1));
mfreq = linspace(0,0.5,length(sysmag)/2) ;
pfreq = linspace(0,0.5,length(sysang)/2);
gfreq = linspace(0,0.5,length(gsys));

%Plotting mag, phase, group delay
if(graphfreq == 1)
    subplot(3,1,1); plot(mfreq,
sysmag(1:length(sysmag)/2));
    title(['IIR System Frequency Response:
a = ',num2str(a)]);
    ylabel('Magnitude');
    subplot(3,1,2); plot(pfreq,
sysang(1:length(sysang)/2));
    ylabel('Phase (rad)');
    subplot(3,1,3); plot(gfreq,
gsys(:,1));
    ylabel('Delay');
    xlabel('Normalized Frequency');
end
```

3.6 *How does this lowpass filter compare with those created in 2.2 and 3.2?*

For the entire system, there are two z-transformed filters in parallel; to combine these, one must simply add them together. This resulted in a low-pass filter with a normalized cutoff frequency of about 0.2025. The group delay of this system seems more consistent than previous low-pass filters we have explored.

3.7 *In your own words, interpret the verse from the Book of Mitra to explain why two allpass filters in parallel may not an allpass filter make.*

Because you simply add the components of the z-transform of parallel filters, this therefore will change the numerators in order to find a common denominator. When combining the filters, we found the form was no longer in the { az + 1 / a + z } form of typical first-order allpass filters.

Conclusion

After reviewing basic digital filters like the allpass, FIR, and IIR systems, I found the parallel allpass system creates the best low-pass filter with the most consistent group delay. This comes from the properties of the z-transform. Also, one may flip the sign of the middle term of a low-pass filter to amplify the differences rather than similarities of a signal, thus creating a high-pass filter rather than a low-pass.