EE 341 Summer 2018
Lab 4 Digital Filtering and Music Equalizer

Professor: Tai-Chang Chen

August 8th, 2018

He Feng 1427841
Hanling Lei 1673094
Zikai Yang 1435493

Introduction

The purpose of this lab is to consider diverse kinds of digital filters and observe their characteristics in time domain. In the first part of the lab, we aimed to figure out the difference between using convolution and passing through a filter while we are processing a signal. We also create two different types of the filters (FIR Digital filter and IIR Digital Filter) in the following sections and apply them to some application such as Stock Market Data and Music Data. In the second part of the lab, We developed a music equalizer and we download the music.wav file to test the equalizer. We considered different set of gains and we observe the sound differences while the set of gains are changing. Generally, as the last lab of the quarter, we focused on the digital filters, and we put the digital filters we learned into a music equalizer application to prove our understanding about different filters.

Results and Discussion

Part I

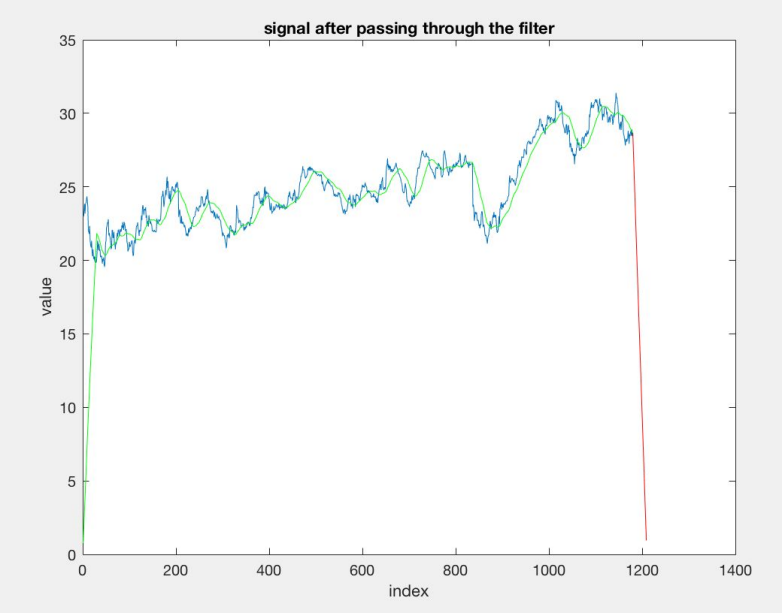
The first part of this lab contains two exercises. In the first exercise, we figured out the difference between using convolution and using digital filter to process a signal. The output result of convolution and using digital filter are different, but both of them are smoother than the original curve. We drawn each curve with diver colors, and based on the color we figured out two methods of implementing a filter can bring different output as result. The second exercise focus on the different between FIR Digital Filter and IIR Digital Filter. We created and implemented these two filters, and apply them onto different application to test their characteristics.

Ex1.m

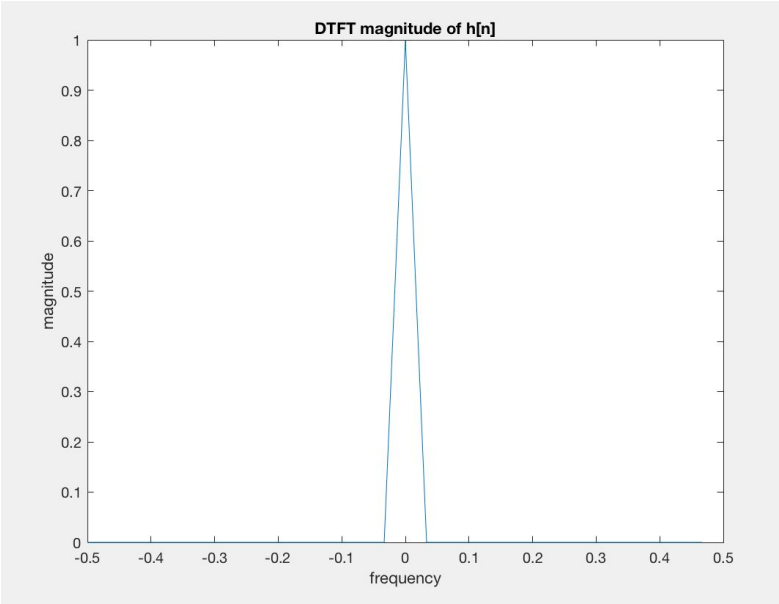
We initially load the “microsoft_stock.txt” file by using load function, and we draw it directly by using plot function. Then we build h[n] in spec, and we applied convolution on the loaded file with h[n] which we defined before. After this step, we defined variables b and a, which are the numerator and denominator while we use the filter function later. We apply the filter to the denominator and numerator defined above and the original loaded data and we create a new signal, which we called filtered_y. Then we plot two graphs within one window. The first one is the signal after convolution, and we drawed it with the red color. The second one is the signal after passing through the filter, and we drawed it with the green color. Here is the code which we implement the filter by using two methods:

```
% Implement the filter
h = 1/30*ones(1,30);
conv_y = conv(data_loaded,h);
b = ones(1,30);
a = 30;
filtered_y = filter(b,a,conv_y);
```

After passing the original signal from the filter, we plot the graphs in one window and we observed the difference among them. We can see that the original graph is more complicated, and the signal graphs after applying to the filter are smoother. Using convolution and digital filter function can bring similar output. By observing the curve in red and green, we found that they are almost the same. The only difference is the curve after the end of the index domain. These two curves can match with each other at the most time and we realized that we can get almost the same result by using these two different methods. Here are the output graph:



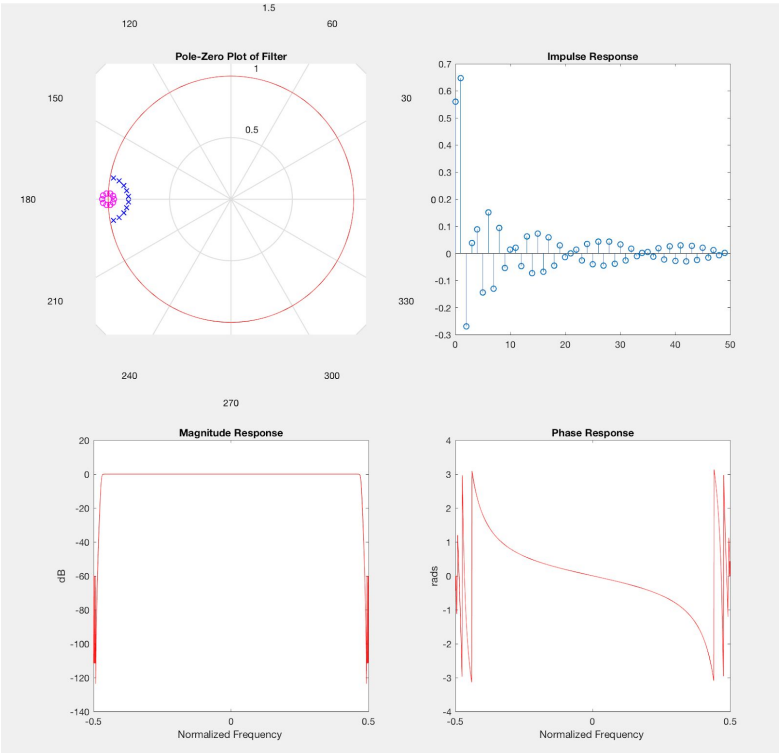
The second part of this exercise wants us to plot the DTFT magnitude of h[n] by using fft function and abs function. Because we did this kind of lab in the last section, therefore we used fftshift function to shift the function because we tried to represent the case of convolution here. The outcoming graph is shown below:



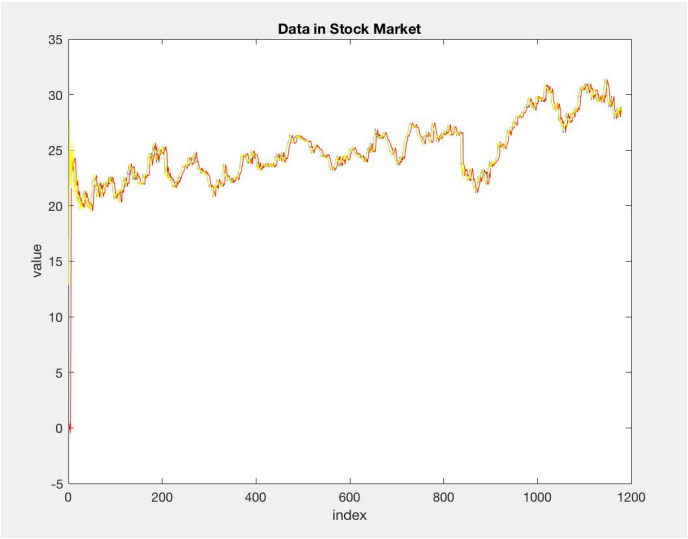
From the graph above we can see that the system is a low pass filter and the high frequency will be filtered out. To conclude, the original and filtered stock data has been shown above, and we also printed out the system frequency response and know that this system is a low pass filter. From the first graph of this exercise, we realize that they can match with each other before the end of the curve. This phenomenon occurs because filter apply the principles of convolution as well. One way to make them totally the same, which can also be called identical results, is to filter out the end of the signal after convolution such that we can only keep the identical matched sections. In this way we can make the results of two implementations identically the same.

Ex2.m

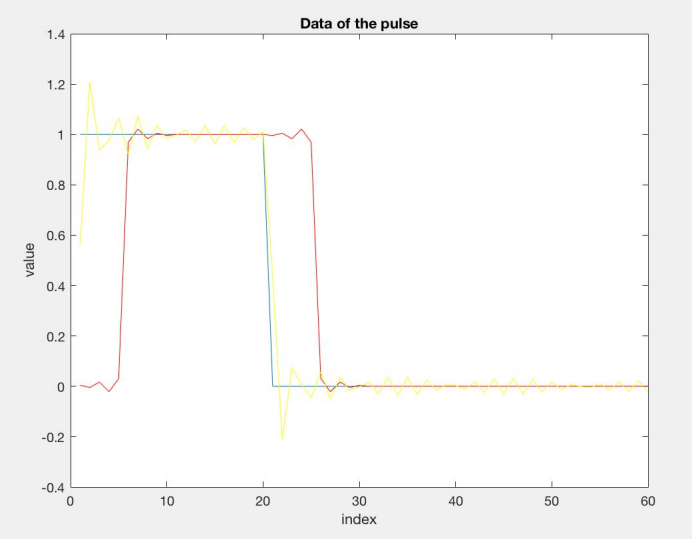
We learned the characteristics of two different filters in this exercise. The first filter is called FIR Digital Filter, which means Finite Impulse Response Digital Filter. The second one is called IIR Digital Filter, which represents Infinite Impulse Response Digital Filter. Initially we defined the filter order and cut-off frequency, then we determined the coefficients of the filters by using fir1 function and butter function. We also used frevalz01 function, which is defined in a given script on the website, to study these filters. The output graphs are shown below:



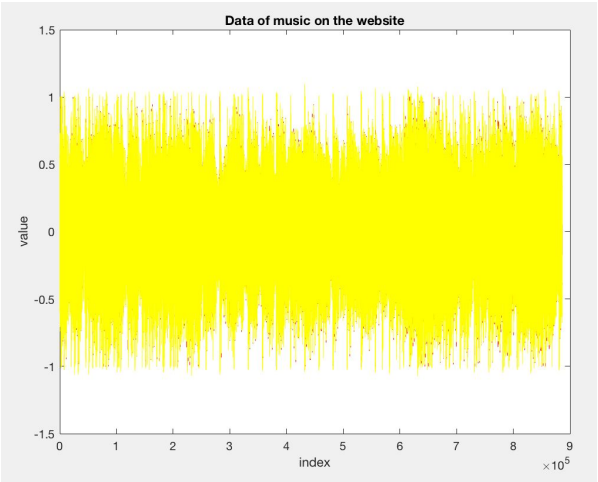
Then we applied the two filters to the stock market dataset. We plotted three graphs for this application. The first one is the original one, the second is the signal passed through the FIR filter, and the last one is the signal passed through the IIR filter. We used “hold on” command to make sure the curves are in the same window such that we can observe the differences among them. We made the first filtered signal red, and the second filtered signal yellow. We also use title function, xlabel function and ylabel function to make the window easier to understand by the users. Here is the graphs from the stock market dataset:

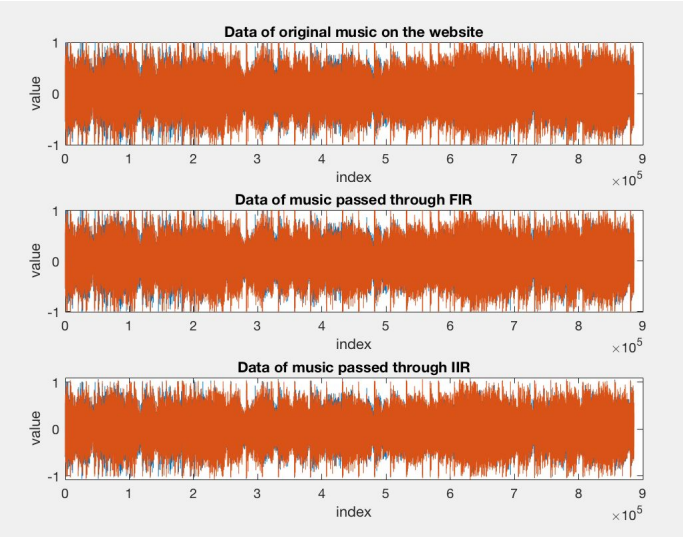


For the second mission, we initially built a matrix which only contains one and zero based on the requirement in the spec. Then we apply two filters to this matrix and do the same thing as we did to process stock market data above. The color of the signal passed through FIR filter is still red, and the color of the signal passed through IIR filter is still green. Here is the final output graph:



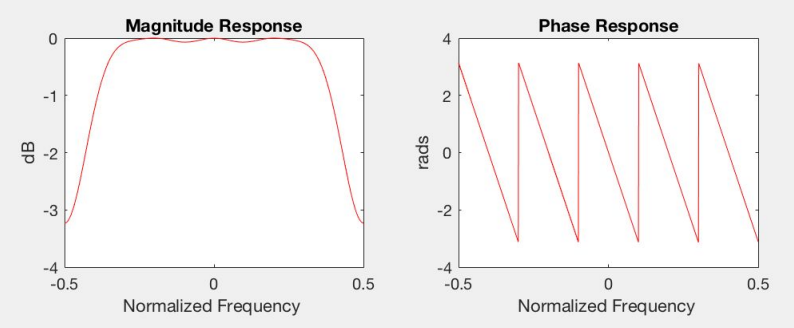
The last application is to apply the filters to the music data, and use the same method as mentioned above to process this application. We initially used audioread function to read through music.wav, then we used filter function to apply these two filters to the music data. Then we use the same method to draw the three curves in one window. However, we noticed that this curve is so complicated and if we put them in one window, the output would be hard to see the difference. Therefore we built a new figure and use subplot function such that we can put each of them in one subplot but next to each other. In this way we can observe the difference easier. Here are the output graphs from this application:



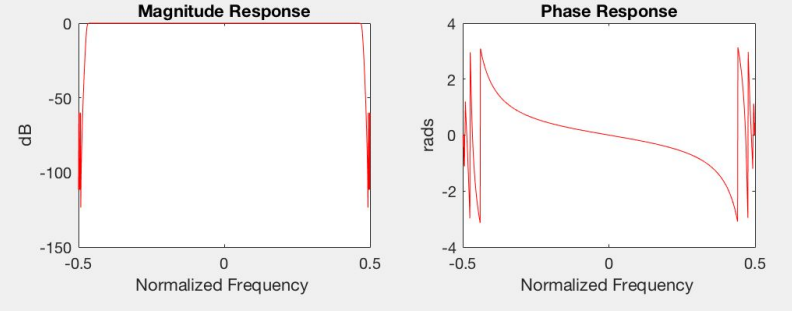


All the output graphs in this report are shown above. The frequency response for each filter is shown below:

FIR:



IIR:



Based on the curves above, we know that both of them can be considered as the low pass filter. FIR has a smoother magnitude response curve, and the middle part of the phase response curve

for IIR filter is smooth and it is decreasing unlike FIR filter. From the result we realized that IIR is hard to control and it does not has a particular phase, and FIR is easier to control and it makes the linear phase always be possible. Also according to the magnitude response of the FIR filter and IIR filter, we know that FIR filter is more stable and IIR filter is not as stable as the other type of filter. Generally both of these two filters in this section are low pass filters, and they can cause different influence to the output such as the stability and difficulty for controlling. Even though both of these filters look like low pass filter, but they still have some differences compared with the ideal low pass filter. The magnitude response of the FIR filter is not stable on the top as the frequency is zero. The curve is not totally straight. For the magnitude response of the IIR filter the increasing and decreasing section of the signal seems too vertical, and the idea low pass filter need a smooth tendency for changing.

From the output of the stock data part, the signal from two filters are smoother than the original curve. The output signal from the FIR filter in this case starts from zero and suddenly increasing to follow the changing movement of the original curve. IIR filter output signal looks closer to the original curve. The pulse data example also prove my analysis above. IIR can bring a more accurate signal graph compared to use FIR filter. FIR filter looks closer to the original curve. For the third section we put three curves into different window because each of them looks complicated which is hard to figure out the differences as we out them into one window. It is hard to figure out the difference so accurately, but the result phenomenon in this case should be similar to the results we got before.

Part II

Ex3.m

In this exercise, we aimed to develop a 3-band equalizer, and we applied this equalizer to a sample music signal to test the result. The idea of an equalizer is to break the sound file into multiple frequency bands. Then we need to weight and reconstruct the signal by summing to get the output signal. We need to use an individual script to build a function, and use this function to make the output in the exercise 3 script.

In equalizer.m, we initially defined the filter coefficients which are supported in the spec. Then we implemented the filter and apply the filter to the given music dataset. After this step we computed the magnitude of each signal by using fvtool function. We can observe the magnitude difference by using this function. Finally we use the idea in the spec to build a variable called “equalize”, and this variable will be the output of this function later.

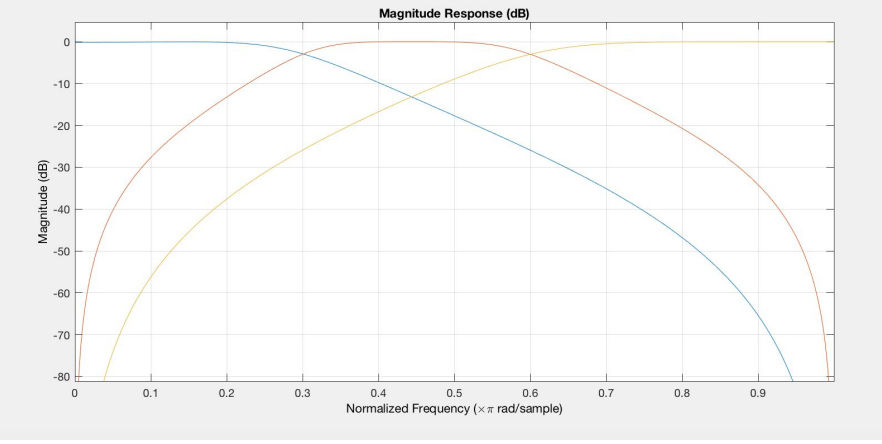
```
% Output result after processing
equalize = (first_filter.*G1) + (second_filter.*G2) + (third_filter.*G3);
```

In Ex3.m, we download the music.wav file and store the important variable by using audioread function. Then we played this music directly by using sound function. Then we applied the equalizer function and use the input y, G1=G2=G3=1 to compute a new signal and then play it out loud. Later we changed the input set to G1=G2=0, G3=1 and G1=1, G2=G3=0 to test the equalizer several more times. At the end of this exercise, we used length function to get the length of each signal to make sure we did the correct processing steps, which is not required in the spec.

```
% Filter coefficients
B1 = [0.0495 0.1486 0.1486 0.0495];
A1 = [1.0000 -1.1619 0.6959 -0.1311];
B2 = [0.1311 0 -0.2622 0 0.1311];
A2 = [1.0000 -0.4824 0.8101 -0.2269 0.2722];
B3 = [0.0985 -0.2956 0.2956 -0.0985];
A3 = [1.0000 0.5772 0.4218 0.0563];
```

B1 and A1 refer to a low pass filter;
B2 and A2 refer to a band pass filter;
B3 and A3 refer to a high pass filter.

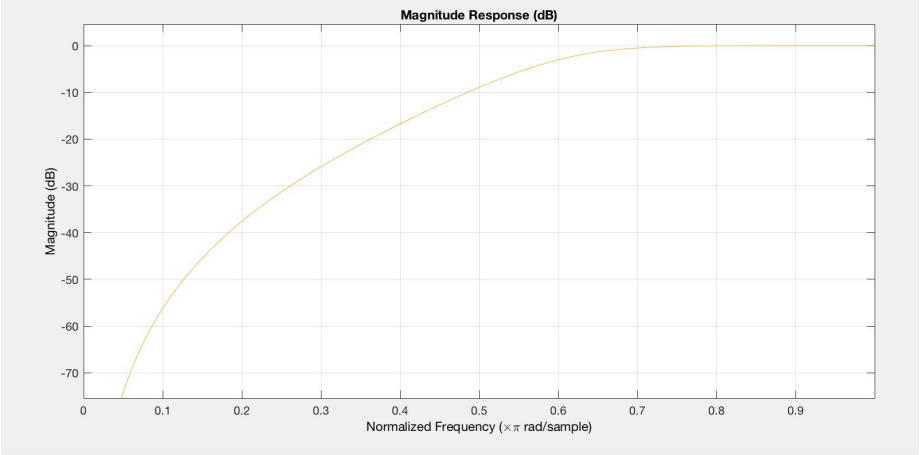
G1, G2 and G3 are the gain, and we multiply each of them with the low pass filter, band-pass filter and high pass filter separately and then sum them up to get the final output signal. Here are the test results for different cases:



G1=G2=G3=1:

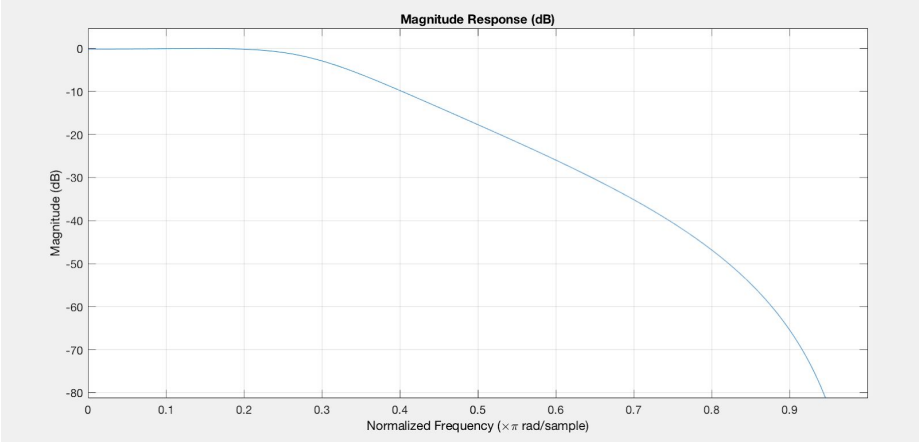
The output sound quality is as good as the original music signal. In my opinion they do not have any differences because they sound so similar.

G1=G2=0, G3=1:



The output sound quality is not as good as the original music signal, and I felt uncomfortable while we played this sound. The sound signal in this case has high frequency, and the low frequency has been filtered. We can hear that this piece of music belongs to this song, but the feelings are different because of the high frequency.

G1=1, G2=G3=0:



The sound quality in this case is also not as good as the original music signal, but unlike the sound signal above, the signal in this case have more low frequency instead of high frequency

and the overall sound quality seems okay and it almost similar to the original sound signal. This sound signal can be considered as the “opposite” signal of the signal right above due to the filtered frequency. In this case the high frequency has been filtered and we can only hear the low frequency.

To answer the questions from the spec, we know a gain of 1⁄2 corresponds to -6 dB in magnitude, a gain of 2 is an increase of 6dB in magnitude, and a gain of 1 is a change of 0 dB in magnitude. Therefore if we want the gain in dB to be from -20 dB to 20 dB, the gain should be in the range [⅛ ⅕ ¼ ⅓ ½ 1 2 3 4 5 6] because if the gain is 1/7 the magnitude would be -21 dB, and if the gain is 7 the magnitude would be 21 dB, which exceed the acceptable range.

Conclusion:

To conclude, this lab focused on different types of digital filters, and we used the digital filters to build an application , which is called the Music Equalizer. From the first part of the lab, we know that the filters made by convolution and digital filters are almost the same, and the filter we created here is a low pass filter based on the DTFT magnitude graph of h[n]. From the second section of the first part of the lab, we learned the differences between two famous filters: FIR filter and IIR filter. We implement these filters and apply them to different application to see the differences between them. From this part we know that FIR can make a linear phase but IIR cannot. IIR looks more similar to the original curve but it is also hard to implement. The last part is an application based on the digital filters, we break the signal into different frequency bands pieces by filtering, and weight them based on the given gain. At last we reconstruct the signal by summing and get the final output. G1=G2=G3=1 make the sound seems almost the same as the original sound. G1=G2=0 and G3=1 keep the high frequency and G1=1 and G2=G3=0 keep the low frequency. We learned a lot through this lab, and this lab increase my interest in signal processing. As a communication concentration student in Electrical Engineering Department, I am looking forward to the future research and study. Thank you for this amazing summer quarter.

Attached Code:

Ex1.m

```
% He Feng, Hanling Lei, Zikai Yang

% We want to experience a moving average filter in this case. We initially
% plot the data from the file on website. Then we implement a filter and
% get the new graph of the signal passed through the filter. Finally we
% plot the DTFT magnitude of h[n] to verify that the system correspond to a
% low pass filter.

clear all;
close all;

% Load the data
data_loaded = load('microsoft_stock.txt');
figure(1);
% Plot the original dataset
plot(data_loaded);
xlabel('index');
ylabel('value');
title('microsoft stock data');

% Implement the filter
h = 1/30*ones(1,30);
conv_y = conv(data_loaded,h);
b = ones(1,30);
a = 30;
filtered_y = filter(b,a,data_loaded);
hold on;
% Plot the new signal after convolution with color red.
plot(conv_y,'r');
xlabel('index');
ylabel('value');
title('signal after convolution');
hold on;
% Plot the new signal after passing through the filter with color green.
plot(filtered_y,'g');
xlabel('index');
ylabel('value');
title('signal after passing through the filter');


% Plot the DTFT magnitude of h[n] within another figure.
figure(2);
N = 30;
w = (-N/2:N/2-1)*(1/N);
H = fft(h);
H_shift = fftshift(H);
H_shift_magnitude = abs(H_shift);
plot(w,H_shift_magnitude);
xlabel('frequency');
ylabel('magnitude');
title('DTFT magnitude of h[n]');
```

Ex2.m

```
% He Feng, Zikai Yang, Hanling Lei

% We initially create two low pass filters, and we apply the filters to
% three data-sets and plot the graph and see the difference.

clear all;
close all;

% Part 1
% Filter order
N = 10;
Fc = 0.3*pi;
stock_data = load('microsoft_stock.txt');

% Section a
% Determine the coefficient of the first filter.
[b,a] = fir1(N,Fc,'low');
filter_1 = frevalz01(b,a);

% Section b
% Determine the coefficient of the second filter.
[d,c] = butter(N,Fc,'low');
filter_2 = frevalz01(d,c);

% Section c
% Apply the two filters to the stock market data, and plot the original
% graph, and the two graphs from the data passed through the filters in my
% figure.
lpf_a_stock = filter(b,a,stock_data);
lpf_b_stock = filter(d,c,stock_data);
figure(2);

plot(stock_data);
hold on;
plot(lpf_a_stock,'r');
hold on;
plot(lpf_b_stock,'y');
title('Data in Stock Market');
xlabel('index');
ylabel('value');

% Apply the two filters to the impulse data, and plot three graphs as
% mentioned above.
pulse_data = [ones(1,20),zeros(1,40)];
lpf_a_pulse = filter(b,a,pulse_data);
lpf_b_pulse = filter(d,c,pulse_data);
figure(3);
plot(pulse_data);
hold on;
plot(lpf_a_pulse,'r');
hold on;
plot(lpf_b_pulse,'y');
title('Data of the pulse');
xlabel('index');
ylabel('value');

% Apply the filters to the music data, and finish drawing by using two
% methods.
[y,Fs] = audioread('music.wav');
```

```
59 -    lpf_a_music = filter(b,a,y);
60 -    lpf_b_music = filter(d,c,y);
61 -    figure(4);
62 -    plot(y);
63 -    hold on;
64 -    plot(lpf_a_music,'r');
65 -    hold on;
66 -    plot(lpf_b_music,'y');
67 -    title('Data of music on the website');
68 -    xlabel('index');
69 -    ylabel('value');
70      % It is hard to se if we put three graphs together, so I put them
71      % separately.
72 -    figure(5);
73 -    subplot(3,1,1);
74 -    plot(y);
75 -    title('Data of original music on the website');
76 -    xlabel('index');
77 -    ylabel('value');
78 -    subplot(3,1,2);
79 -    plot(lpf_a_music);
80 -    title('Data of music passed through FIR');
81 -    xlabel('index');
82 -    ylabel('value');
83 -    subplot(3,1,3);
84 -    plot(lpf_b_music);
85 -    title('Data of music passed through IIR');
86 -    xlabel('index');
87 -    ylabel('value');
88
```

Ex3.m

```
% He Feng, Zikai Yang, Hanling Lei

% We built an music equalizer in this exercise. We initially build a
% function and we based on this function to break the sound file, weight
% and reconstruct the signals by summing.

clear all;
close all;

% Part 2
% Download the music.wav file, and comparing the output between two
% situations.
[y,Fs] = audioread('music.wav');
sound(y,Fs);
pause(20);
equalized_sound_1 = equalizer(y,1,1,1);
sound(equalized_sound_1,Fs);
audiowrite('equalized_sound_1.wav',equalized_sound_1,Fs);

% Part 3
% Change to different sets of gains to do the same procedure mentioned
% above again.
equalized_sound_2 = equalizer(y,0,0,1);
sound(equalized_sound_2,Fs);
```

```
sound(equalized_sound_2,Fs);
audiowrite('equalized_sound_2.wav',equalized_sound_2,Fs);
pause(20);
equalized_sound_3 = equalizer(y,1,0,0);
sound(equalized_sound_3,Fs);
audiowrite('equalized_sound_3.wav',equalized_sound_3,Fs);

% Testing to make sure that they have the same length.
% This is a personal testing, and it is not required in the spec.
length(y)
length(equalized_sound_1)
length(equalized_sound_2)
length(equalized_sound_3)
```

equalizer.m

```
% He Feng, Zikai Yang, Hanling Lei

% This is a function used in Ex3.m which introduces an equalizer to process
% music or other signals.

function equalize = equalizer(file,G1,G2,G3)

% Filter coefficients
B1 = [0.0495 0.1486 0.1486 0.0495];
A1 = [1.0000 -1.1619 0.6959 -0.1311];
B2 = [0.1311 0 -0.2622 0 0.1311];
A2 = [1.0000 -0.4824 0.8101 -0.2269 0.2722];
B3 = [0.0985 -0.2956 0.2956 -0.0985];
A3 = [1.0000 0.5772 0.4218 0.0563];

% Implement the filter and apply the given dataset to the filter.
first_filter = filter(B1,A1,file);
second_filter = filter(B2,A2,file);
third_filter = filter(B3,A3,file);

% Compute the magnitude of each signal before and after processing
fvtool(B1*G1,A1, B2*G2, A2, B3*G3, A3);

% Output result after processing
equalize = (first_filter.*G1) + (second_filter.*G2) + (third_filter.*G3);
```