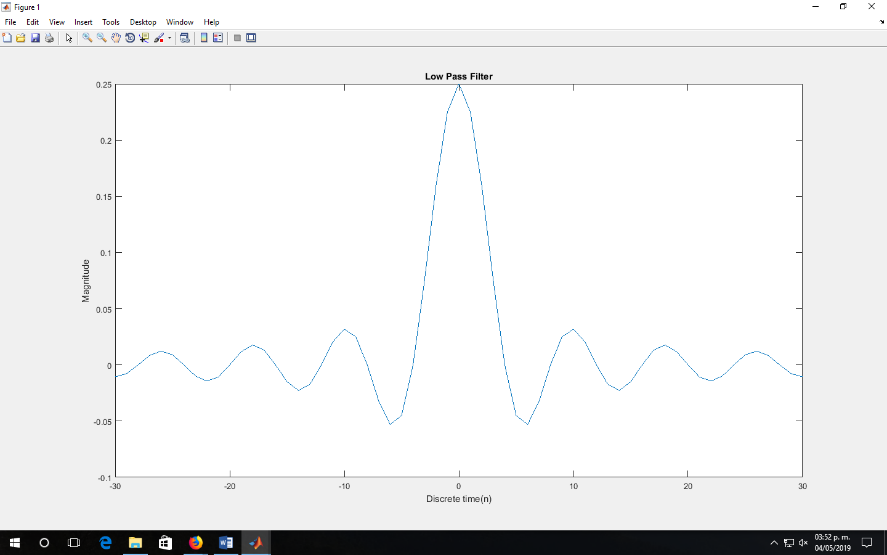
**Design of a FIR Digital Filter**

Andrea Perez Huizar  
IE698276

**Part I**

Design a low pass filter that has a cut off frequency of π/4, firstly having a length of 11 elements and later on of 17 elements.

1. Calculate the ideal h(n) for the desired cut off frequency
   1. Define n from -30 to 30
   2. Define h(n) for low pass filter
   3. Graph h(n)

%Define h(n)

n=-30:30;

wc=pi/4;

h1=(wc/pi)\*sinc(wc\*n/pi);

figure;

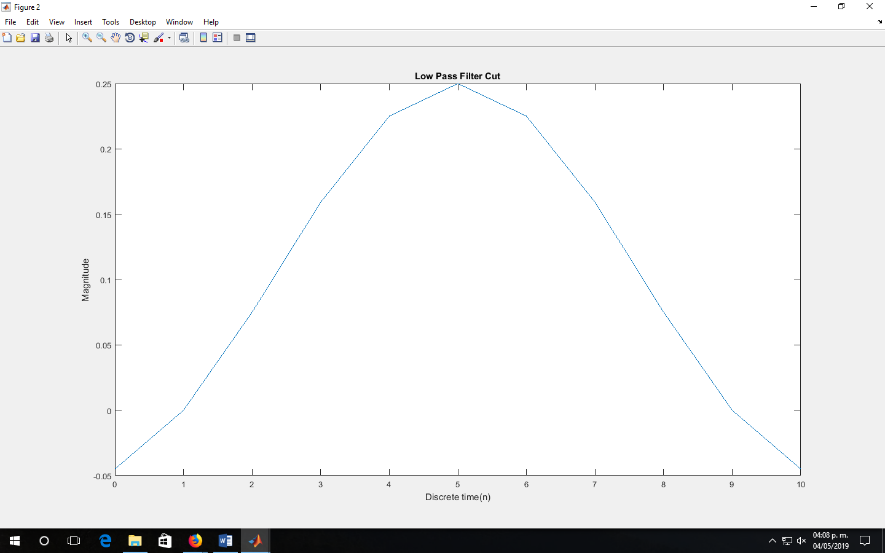
plot(n,h1);

xlabel('Discrete time(n)');

ylabel('Magnitude');

title('Low Pass Filter');

1. Truncate h(n) using a rectangular window that has a length of 11 elements
   1. Define n from 0 to 10
   2. Define new h(n) for truncate and transfer impulse with a length of 11 elements
   3. Copy values from first h(n) to new h(n), from n equals -5 to 5
   4. Graph new h(n)

%Truncate and transfer

n1=0:10;

h2=zeros(1,11);

h2=h1(26:36);

figure;

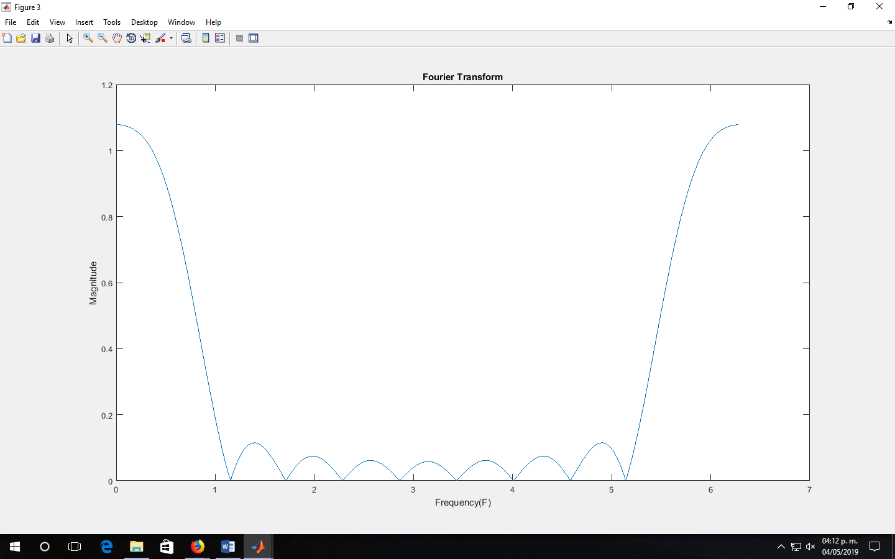
plot(n1,h2);

xlabel('Discrete time(n)')

ylabel('Magnitude')

title('Low Pass Filter Cut')

1. Check frequency response of the calculated filter
   1. Define angular frequency variable from 0 to 2π with 512 points
   2. Calculate system frequency for new h(n) for the FFT with 512 points
   3. Graph magnitude and angle
   4. Analyze the results

%FFT for h(n)

f=linspace(0,(2\*pi),512);

fft\_y=fft(h2,length(f));

mfft\_y=abs(fft\_y);

figure;

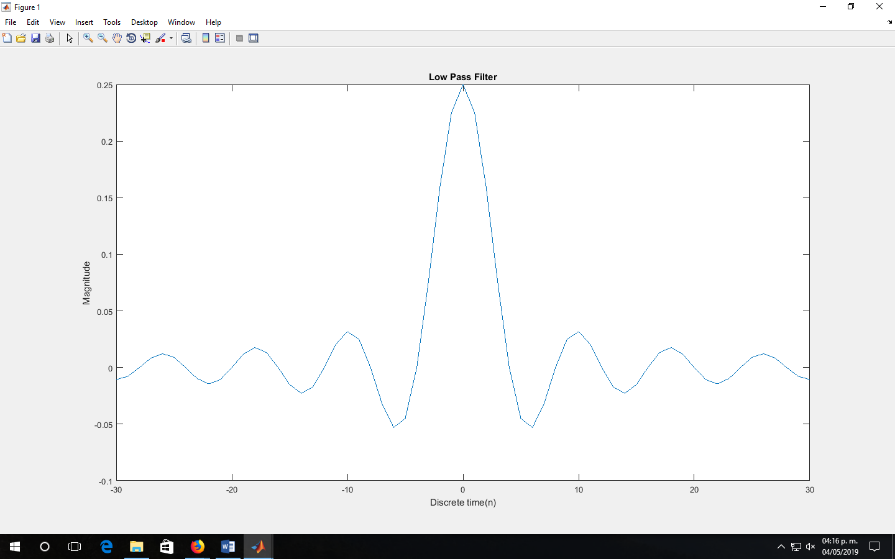
plot(f,mfft\_y);

xlabel('Frequency(F)')

ylabel('Magnitude')

title('Fourier Transform')

1. Redo steps for a filter of a 17 elements length

%Define h(n)

n=-30:30;

wc=pi/4;

h1=(wc/pi)\*sinc(wc\*n/pi);

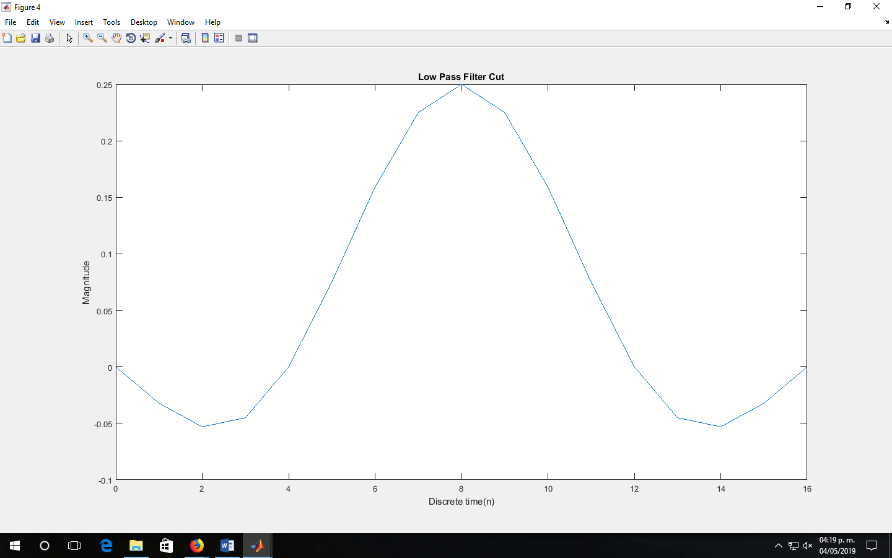
figure;

plot(n,h1);

xlabel('Discrete time(n)');

ylabel('Magnitude');

title('Low Pass Filter');

%Truncate and transfer

n1=0:16;

h2=zeros(1,17);

h2=h1(23:39);

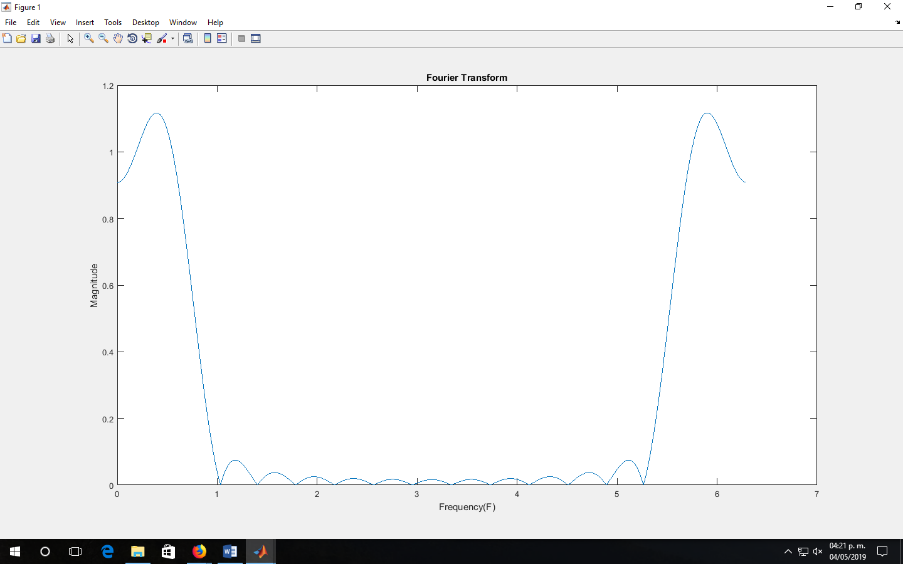
figure;

plot(n1,h2);

xlabel('Discrete time(n)')

ylabel('Magnitude')

title('Low Pass Filter Cut')

%FFT for h(n)

f=linspace(0,(2\*pi),512);

fft\_y=fft(h2,length(f));

mfft\_y=abs(fft\_y);

figure;

plot(f,mfft\_y);

xlabel('Frequency(F)')

ylabel('Magnitude')

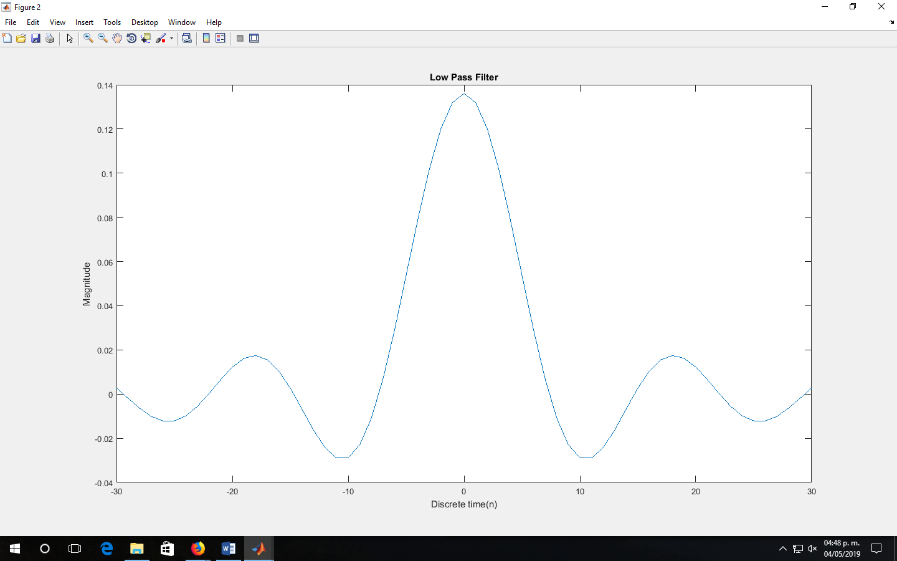
title('Fourier Transform')

Both filter even though they have different lengths are working properly since we can see in the graphs that the values that exceed π/4 are almost equal to zero, this means the higher frequencies are non-existent. We can assume the filter are correct.

**Part II**

Design and apply a low pass filter to an audio signal

1. Calculate the cut off frequency of a filter to apply it to an audio signal sampled at 44100m/s so that the cut off frequency in hertz corresponds to a 3KHz signal
   1. Operation to obtain wc frequency
   2. Design filter of a 17 element length
   3. Check frequency response in hertz
   4. Analyze frequency response

%Calculate wc

fs=44100;

fc=3000;

wc=2\*pi\*(fc/fs);

%Define h(n)

n=-30:30;

h1=(wc/pi)\*sinc(wc\*n/pi);

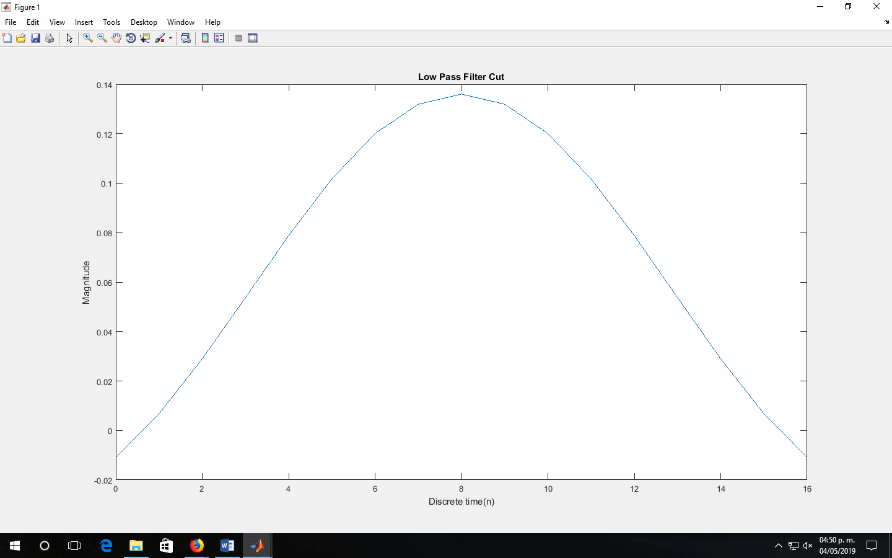
figure;

plot(n,h1);

xlabel('Discrete time(n)');

ylabel('Magnitude');

title('Low Pass Filter');

%Truncate and transfer

n1=0:16;

h2=zeros(1,17);

h2=h1(23:39);

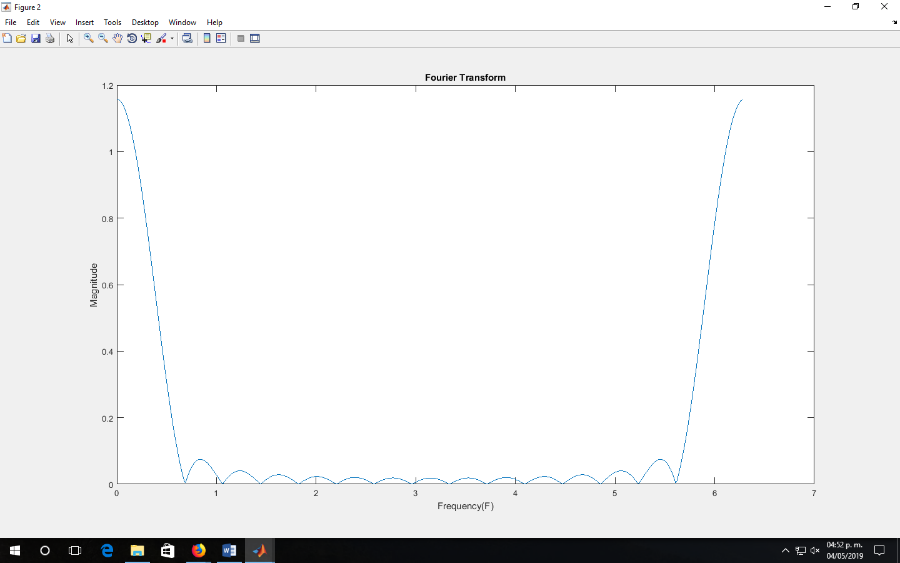
figure;

plot(n1,h2);

xlabel('Discrete time(n)')

ylabel('Magnitude')

title('Low Pass Filter Cut')

%FFT for h(n)

f=linspace(0,(2\*pi),512);

fft\_y=fft(h2,length(f));

mfft\_y=abs(fft\_y);

figure;

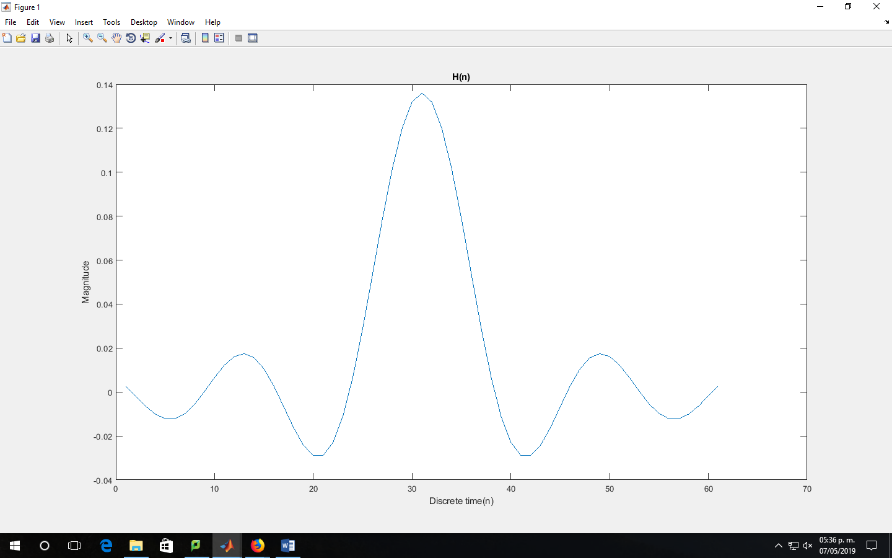
plot(f,mfft\_y);

xlabel('Frequency(F)')

ylabel('Magnitude')

title('Fourier Transform')

1. Analyze the frequency spectrum in an audio signal
   1. Assign a variable to the audio signal “spring\_Hi\_Fi”
   2. Listen to audio
   3. Generate a time variable that corresponding to the audio signal
   4. Calculate magnitude of frequency spectrum
   5. Generate frequency variable corresponding to the spectrum
   6. Graph the time signal and its frequency spectrum

%Audio signal

[x, fs] = audioread('spring\_HiFi.wav');

x\_sound = audioplayer (x,fs);

%play(x\_sound);

%Calculate wc

fs=44100;

fc=3000;

wc=2\*pi\*(fc/fs);

%Define h(n)

n=-30:30;

h1=(wc/pi)\*sinc(wc\*n/pi);

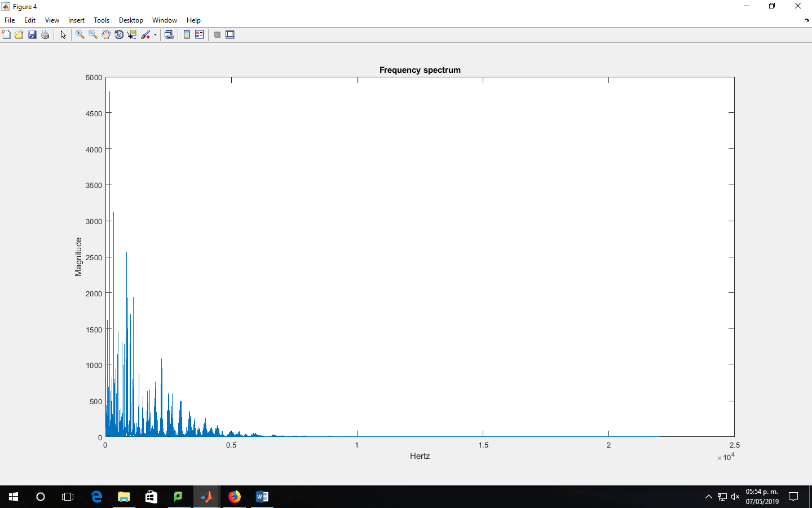
%FIR

plot(h1);

xlabel('Discrete time(n)')

ylabel('Magnitude')

title('H(n)')

%Frequency spectrum

spec=fft(x);

spec=spec(1:end/2);  
freq=linspace(0,fs/2,length(spec));

magnitud=abs(spec);

t=0:1/fs:(length(x)-1)/fs;

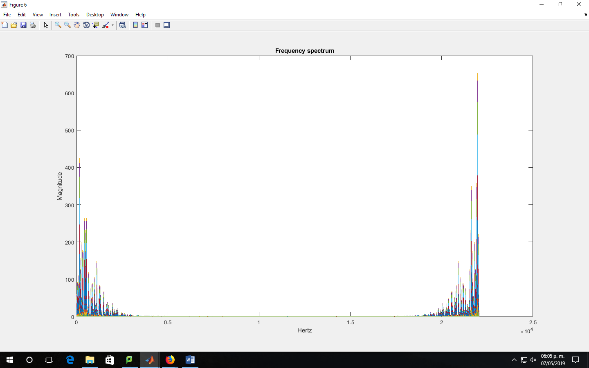
figure;

plot(freq,magnitud);

xlabel('Hertz')

ylabel('Magnitude')

title('Frequency spectrum')

1. Apply the previously design filter to the previously analyze signal
   1. Apply convolution to the filter and audio signal
   2. Calculate spectrum magnitude
   3. Graph magnitude
   4. Listen to obtain signal and compare with the original
   5. Analyze the filtered result

conv = h1.\* x;

spec\_conv=fft(conv);

freq\_conv=linspace(0,fs/2,length(spec\_conv));

magnitud\_conv=abs(spec\_conv);

t=0:1/fs:(length(x)-1)/fs;

figure;

plot(freq\_conv,magnitud\_conv);

xlabel('Hertz')

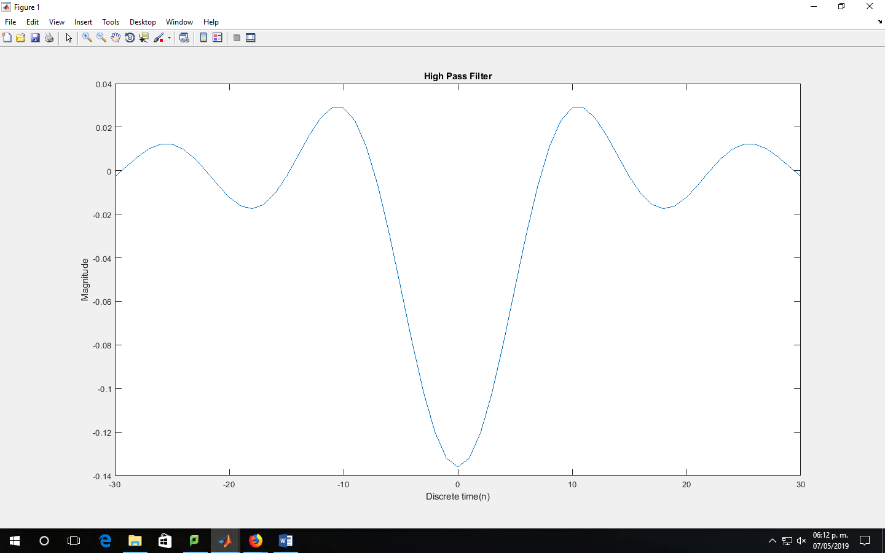
ylabel('Magnitude')

title('Frequency spectrum')

In the previous graph we can see how the filter is acting by not letting the higher frequencies go through so we can assume the filter is working properly.

**Part III**

1. Design a high pass filter with a cut off frequency of 3KHz for a sample frequency of 44.1KHz
   1. Obtain wc frequency
   2. Design 3KHz cut off ideal filter of a 17 element length
   3. Apply a hamming window to obtain filter
   4. Graph impulse responses obtain in previous steps
   5. Calculate and graph the magnitudes of the impulse responses in frequency of both filters

%High pass filter

%Calculate wc

fs=44100;

fc=3000;

wc=2\*pi\*(fc/fs);

%Define h(n)

n=-30:30;

h1=-(wc/pi)\*sinc(wc\*n/pi);

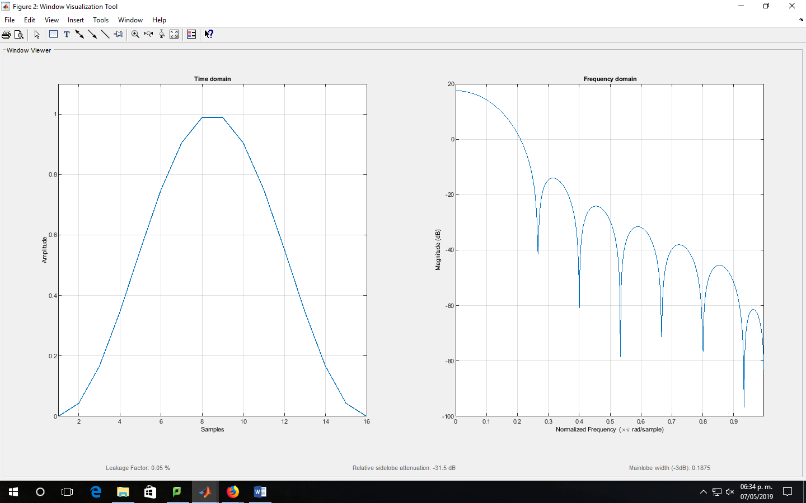
figure;

plot(n,h1);

xlabel('Discrete time(n)');

ylabel('Magnitude');

title('High Pass Filter');

 %Window

M=16;

wvtool(hann(M))

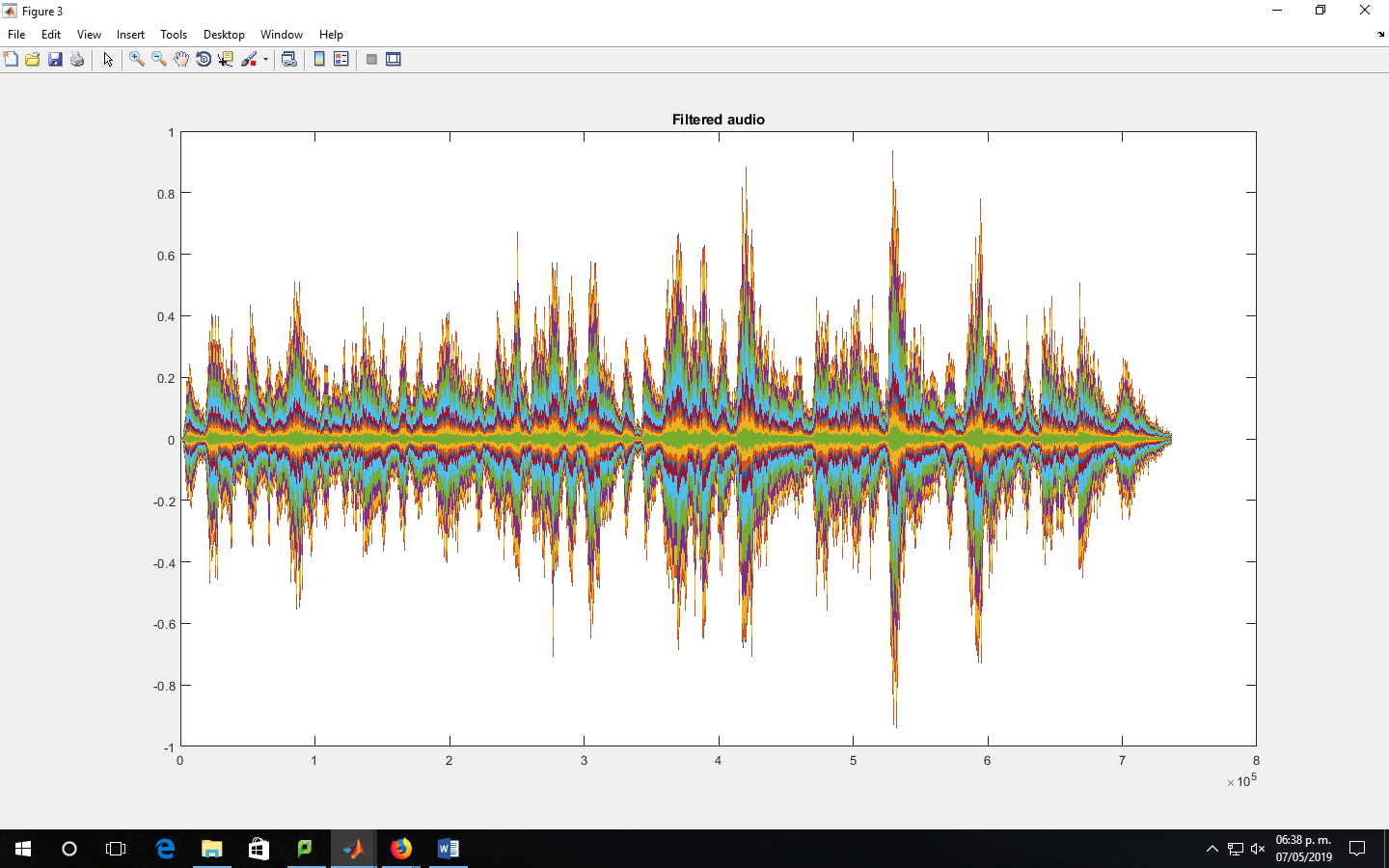
h2=h1\*wvtool;

figure;

plot(n,h2);

title('Filter windowed');

1. Apply the previously design filter to an audio signal
   1. Apply convolution to the filter with a hamming window to the audio signal “spring\_Hi\_Fi”
   2. Calculate the frequency spectrum magnitude to the obtain signal
   3. Graph the frequency spectrum magnitude
   4. Listen to the obtain signal and compare with the original
   5. Analyze the result of the applied filter

%Apply filter

tot=h2.\*x;

figure;

plot(tot);

title('Filtered audio');