ECE 311

Lab4 report

Name: Yang Shi

NetID: yangshi5

Report Item 1:

Code Part:

```
%%report_item_1
N = 41; %total number of points in the impulse response,
changing between 41 and 81
n = -(N-1)/2 : (N-1)/2; %-20 to 20 or -40 to 40
wc = pi/3;
w0 = pi/2;
%get low pass filter response
h_low = wc/pi*sinc(wc/pi*n);
H_low = fft(h_low);
shift_H_low = fftshift(H_low);
%get high pass filter response
delta = dirac(n);
delta(1,(N-1)/2+1) = 1;
h_high = delta - wc/pi*sinc(wc/pi*n);
H_high = fft(h_high);
shift_H_high = fftshift(H_high);
%get band pass filter response
h_band = cos(w0*n).*sinc(wc/pi*n);
H_band = fft(h_band);
shift_H_band = fftshift(H_band);
%get radians axis information
w = fftshift((0:N-1)/N*2*pi);
w(1:N/2) = w(1:N/2)-2*pi;
%plot low pass filter
figure(1);
subplot(2, 1, 1);
stem(n, h_low);
xlabel('n');
ylabel('magnitude');
title('impulse response of low pass filter');
subplot(2, 1, 2);
plot(w, abs(shift_H_low));
xlabel('w(Radians)');
ylabel('magnitude');
title('magnitude response of low pass filter');
```

```
%get high pass filter
figure(2);
subplot(2, 1, 1);
stem(n, h_high);
xlabel('n');
ylabel('magnitude');
title('impulse response of high pass filter');
subplot(2, 1, 2);
plot(w, abs(shift_H_high));
xlabel('w(Radians)');
ylabel('magnitude');
title('magnitude response of high pass filter');
figure(3);
subplot(2, 1, 1);
stem(n, h_band);
xlabel('n');
ylabel('magnitude');
title('impulse response of band pass filter');
subplot(2, 1, 2);
plot(w, abs(shift_H_band));
xlabel('w(Radians)');
ylabel('magnitude');
title('magnitude response of band pass filter');
```

Explanation:

For n = -20 : 20

Figure 1 below is the response of the lowpass filter:

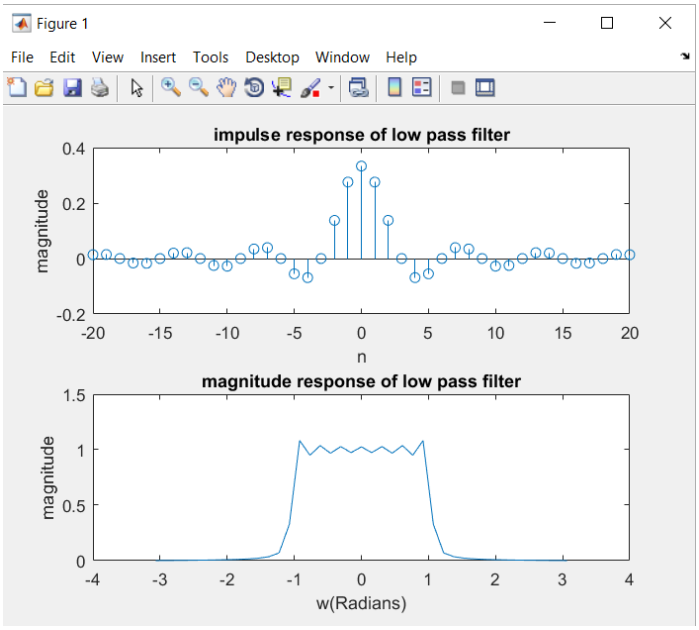


Figure 2 below is the response of the highpass filter:

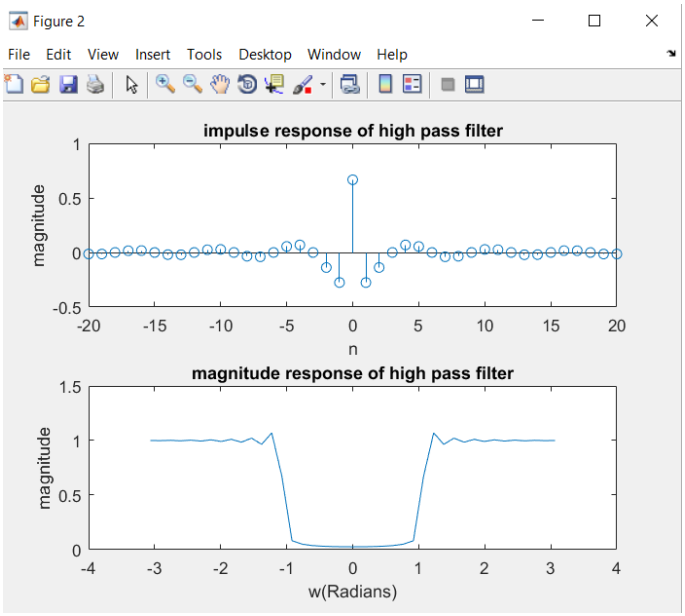
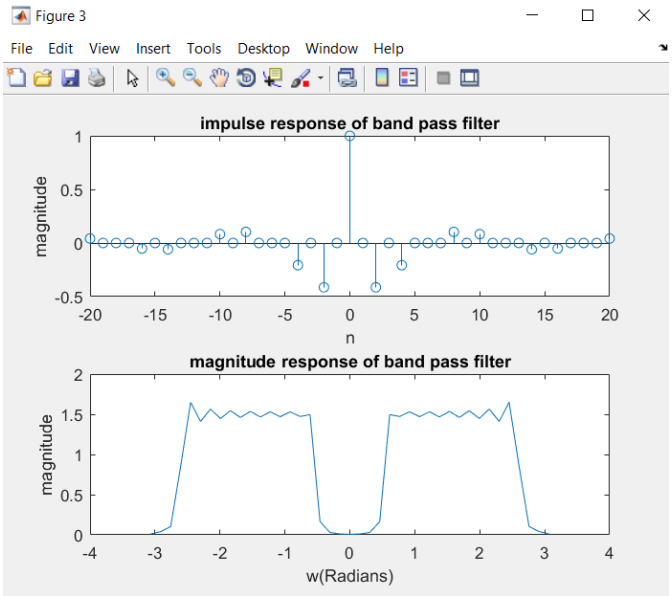


Figure 3 below is the response of the bandpass filter:



For n = -40 : 40

Figure 1 below is the response of the lowpass filter:

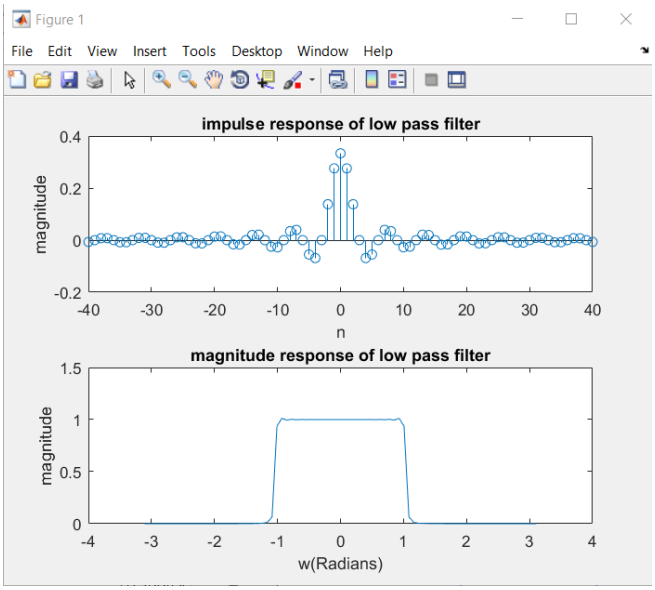


Figure 2 below is the response of the highpass filter:

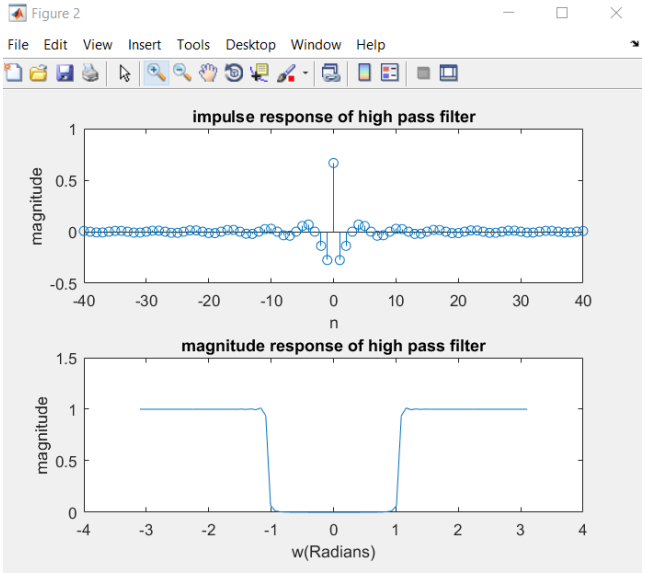
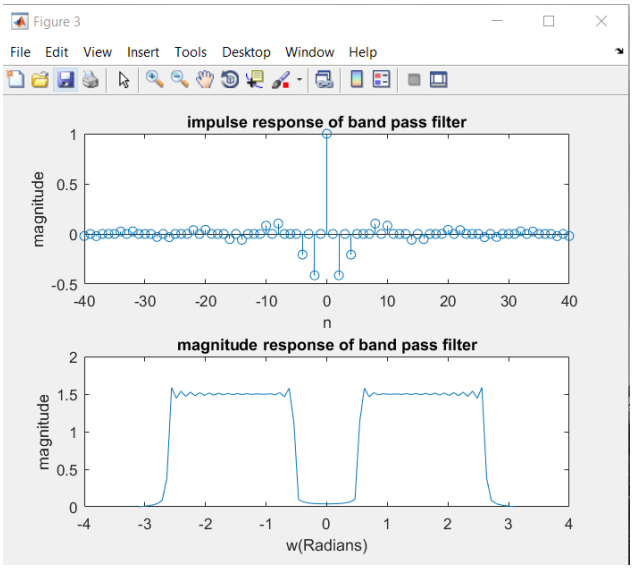


Figure 3 below is the response of the bandpass filter:



The magnitude responses have some glitches around the edges of the ideal response. The more points in the impulse response, the less the number of glitches will be. That is because we use a finite impulse response to build these filters. And they will be always continuous in the actual situation. Therefore, the ideal filters which are not continuous are not available.

Report Item 2:

Code part:

```
%report_item_2
load('impulseresponse.mat');

N = length(h);
H = fft(h);
shiftH = fftshift(H);
dbH = mag2db(abs(shiftH));%change to dB
w = fftshift((0:N-1)/N*2*pi);
w(1:N/2) = w(1:N/2)-2*pi;
%plot
figure(1);
stem(h);
xlabel('n');
ylabel('magnitude');
title('impulse response');
figure(2);
```

```
plot(w, dbH);
xlabel('w(Radians)');
ylabel('magnitude');
title('magnitude response in dB');
grid on;
figure(3);
plot(w, angle(shiftH));
xlabel('w(Radians)');
ylabel('phase');
title('phase response');
grid on;
```

Explanation part:

Figure 1 below is the impulse response of the filter:

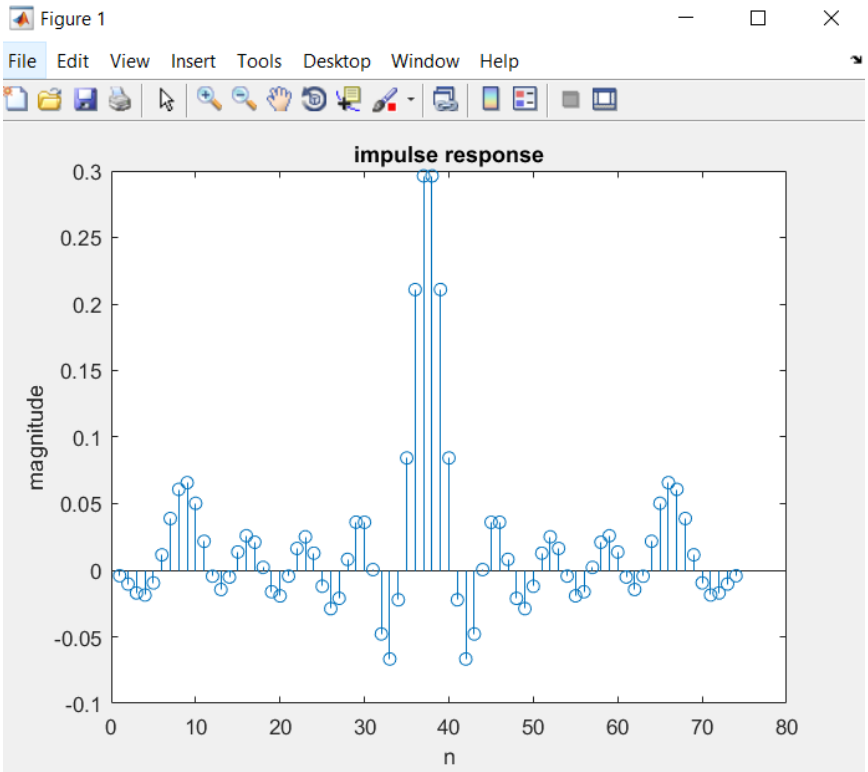


Figure 2 below is the magnitude response (in dB) of the filter:

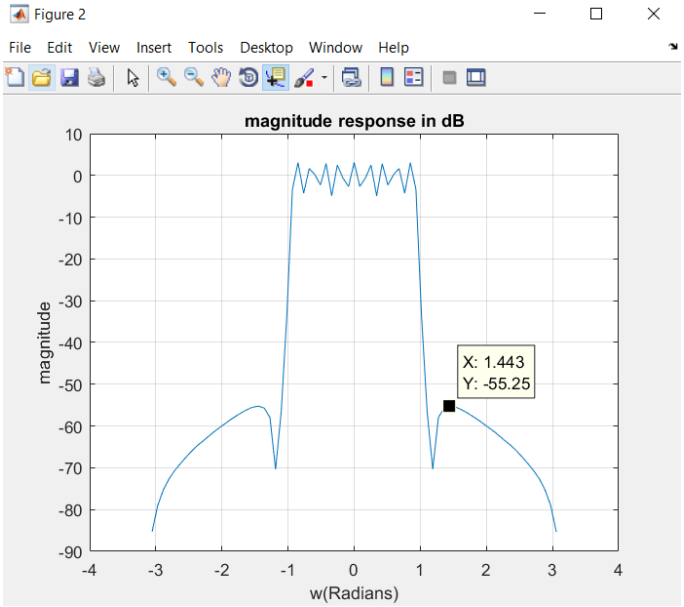
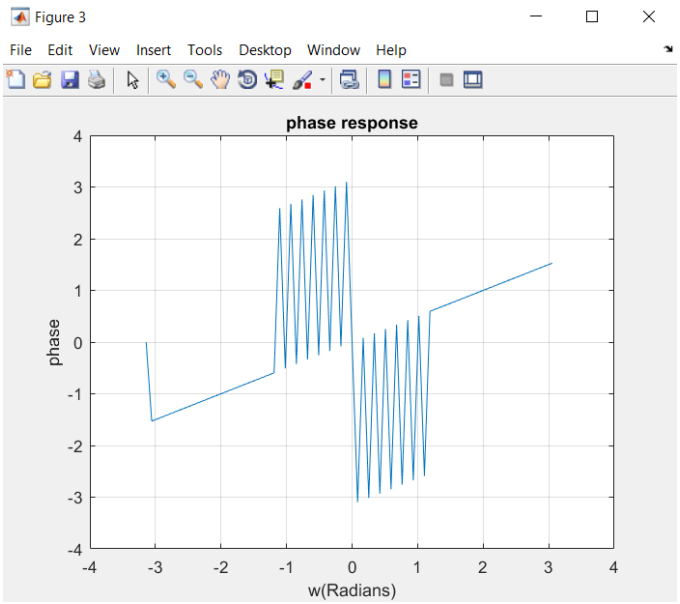


Figure 3 below is the phase response of the filter:



Using the cursor on the Matlab, we can find that the stopband ripple is (-55.25 – (-85.32) = 30.07 dB). The stopband attenuation is -55.25 dB and the passband ripple is 7.3 dB. The transition bandwidth is 1.189 - 0.934 = 0.255 radians.

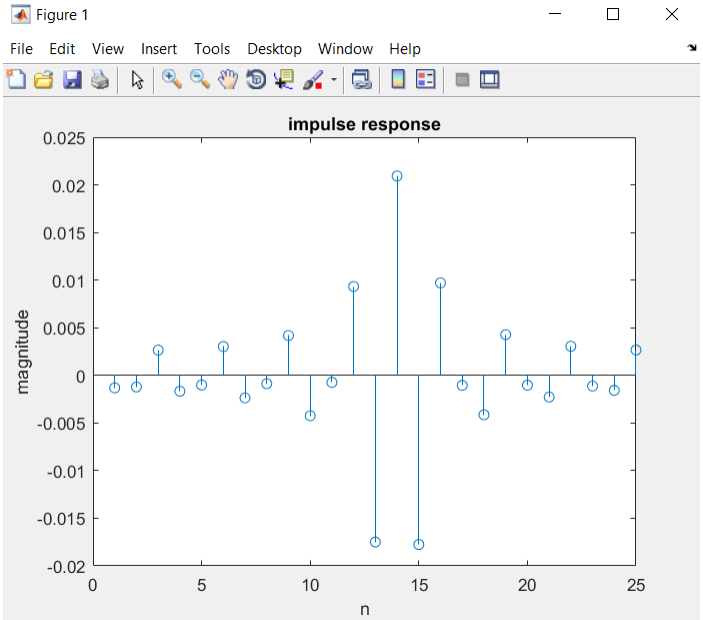
Report Item 3:

Code part:

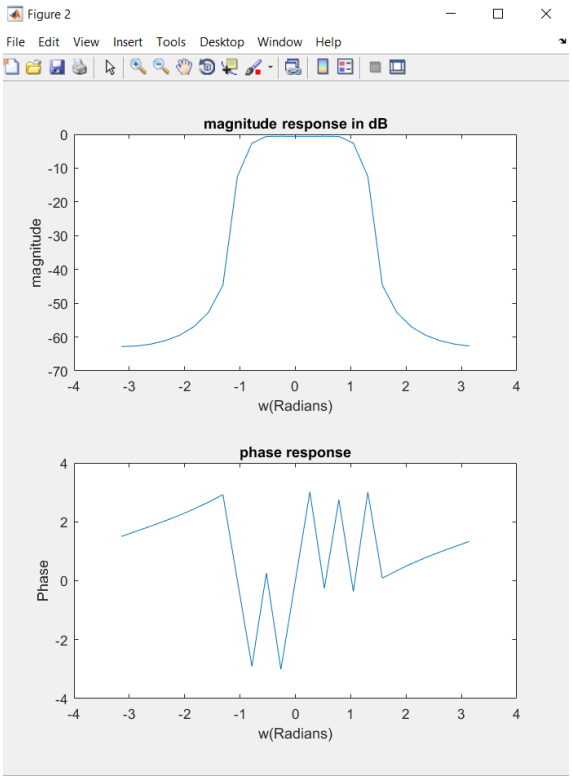
```
%Report_Item_3
N = 25;
w = linspace(-pi, pi, N);
wc = pi/3;
Gw = zeros(1, N);
M = N/2;
%find G(w)
for i = 1 : N
    if abs(w(i)) < wc
        Gw(i) = exp(-1i*M*w(i));
    else
        Gw(i) = 0;
    end
end
gn = ifft(Gw); %have gn
hn = zeros(1,N);
%add hamming window on it
for i = 1 : N
    hn(i) = gn(i)*(0.54-0.46*cos(2*pi*i/(N-1)));
end

Hw = fft(hn);

figure(1);
stem(gn);
xlabel('n');
ylabel('magnitude');
title('impulse response');
figure(2);
subplot(2,1,1);
plot(w, mag2db(abs(Hw)));
xlabel('w(Radians)');
ylabel('magnitude');
title('magnitude response in dB');
subplot(2,1,2);
plot(w, angle(Hw));
xlabel('w(Radians)');
ylabel('Phase');
title('phase response');
```

Explanation:

The figure 1 below is the impulse response:



The figure 2 below is the magnitude (in dB) and phase response of the designed FIR filter.



Using the cursor on the Matlab, we can find that the stopband attenuation is -44.64 dB and the passband ripple is 2.6 dB. The passband edge frequency is 1.047 radians, and the stopband edge frequency is 1.571 radians.

Report Item 4:

Code part:

```
%Report Item 4
fs = 1500;
f = [0.3*fs/2, 0.36*fs/2];
a = [1,0];
rp = 3;
rs = 50;
dev = [(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];
[n, fo, mo, w] = firpmord(f, a, dev, fs);
b = firpm(n, fo, mo, w);
figure(1);
freqz(b, 1);
figure(2);
impz(b);
```

Explanation:

The figure 1 below is the magnitude (in dB) and phase response of the filter:

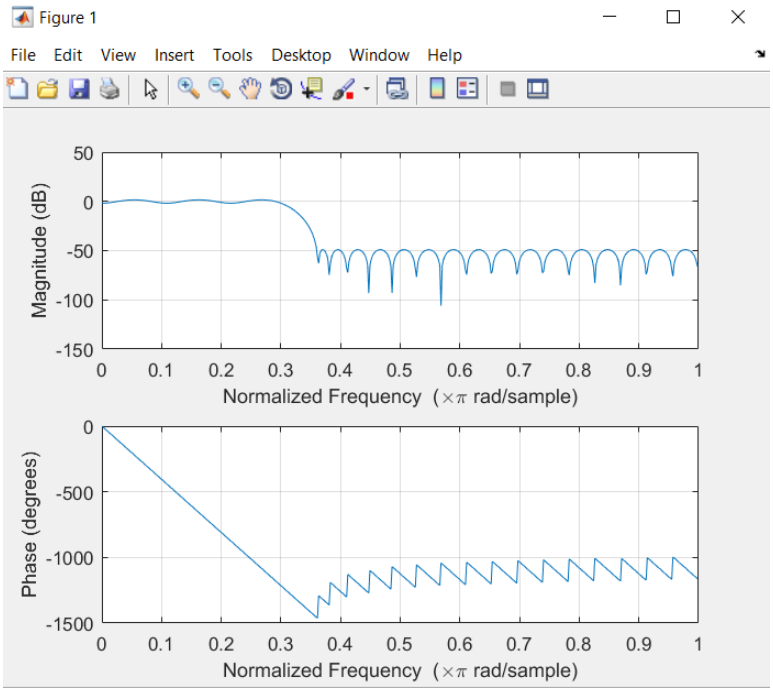
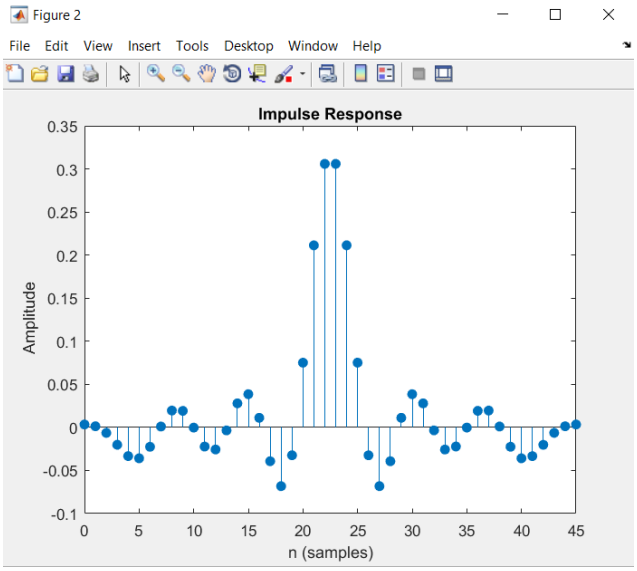


Figure 2 below is the impulse response of the filter:



Report Item 5:

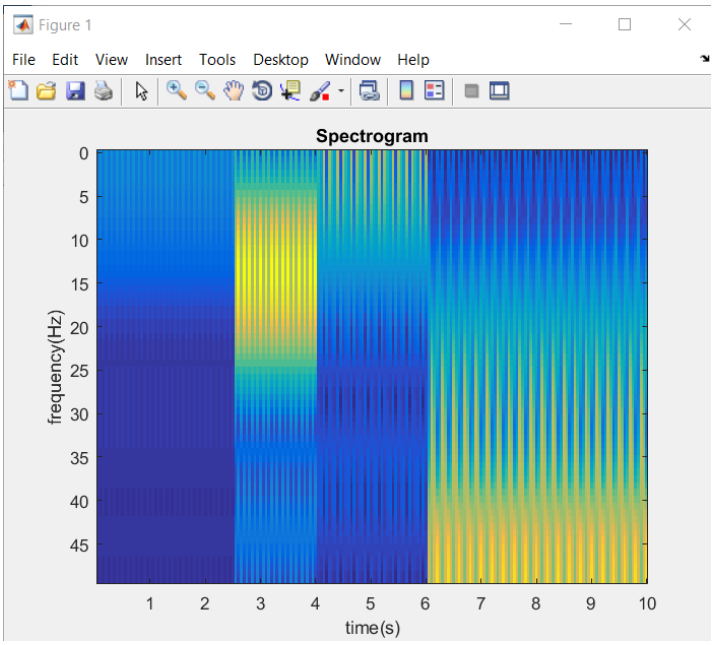
Code part:

```
function [m1, f2, t3] = mySTDFT(xname, M, D, P)
load(xname); %get x and fs
N = length(x);
row = ceil(P/2);
column = floor((N-M)/D)+1;
dt = D/fs;
f2 = zeros(1, row);
t3 = zeros(1, column);
m1 = zeros(row, column);
%get matrix m1
for m = 1 : column
    t3(m) = m*dt; %get vector t3
    temp = zeros(1, P);
    tempcolumn = zeros(1, P);
    temp(1:M) = x((m-1)*D+1 : (m-1)*D+M);
    for k = 1 : P
        for n = 1 : P
            tempcolumn(k) = tempcolumn(k) + temp(n)*exp(-
1i*2*pi*k*n/P);
        end
    end
    m1(1:row, m) = tempcolumn(1:row);
end
%get the vector f2 to store omgea
```

```
w = fftshift((0:P-1)/P*2*pi);
w(1:P/2) = w(1:P/2)-2*pi;
f2 = w(P/2+1:P)/(2*pi)*fs;
%plot
figure(1);
imagesc(t3, f2, abs(m1));
xlabel('time(s)');
ylabel('frequency(Hz)');
title('Spectrogram');
end
```

Explanation:

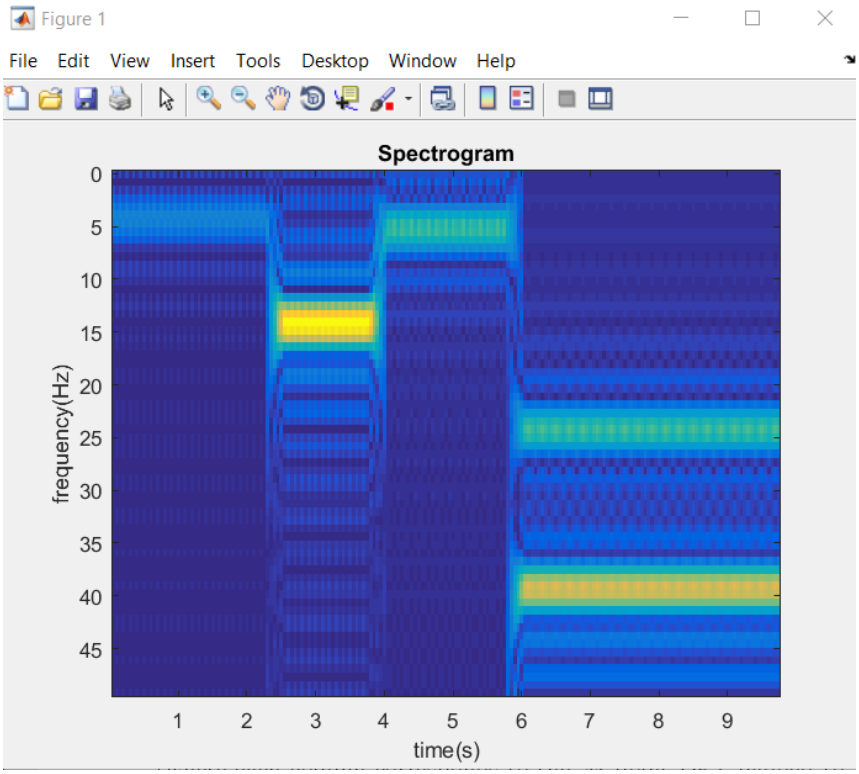
The figure 1 below is the resulting spectrogram of the spectrogram.mat for P = 128, M = 5, D = 5 using imagesc



According to the Rayleigh limit, $f_{min} > 1/(N*dt) = 1/(M/fs) = 20$ Hz

We can roughly see that there are two main frequency including one around 20 Hz (starting from 2.5 seconds, ending at 4 seconds) and the other one around 40 Hz (starting from 6 seconds and ending in the end). But the resolution is not high enough. And there is little information about the frequency smaller than f_{min}.

The figure 1 below is the resulting spectrogram of the spectrogram.mat for P = 128, M = 30, D = 5 using imagesc



According to the Rayleigh limit, $f_{min} > 1/(N*dt) = 1/(M/fs) = 3.33$ Hz

We can roughly see that there are two main frequency including one around 15 Hz (starting around 2.5 seconds, ending at 4 seconds) and the other one around 40 Hz (starting from 5.8 seconds and ending in the end). There are also some frequencies with weaker amplitude appeared in the spectrogram including 5 Hz (from 0 to 2.5 seconds and from 4 seconds to 5.8 seconds) and 25 Hz (from 5.8 seconds to the end). And there is little information about the frequency smaller than f_{min}.

The difference between two spectrogram plots is that there are more frequency bins available for the spectrogram to show if M is larger. This is because the Rayleigh limit is lower when the window length is longer.

Report Item 6:

Code part:

Change a little bit mySTDFT.m to directly read the x and fs and plot two spectrogram at the same time:

```
function [m1, f2, t3] = mySTDFT(x, fs, M, D, P, number)%remove
xname, number for the plot
if (number ~= 1) && (number ~= 2)
    number = 1;
end
%load(xname); %get x and fs
N = length(x);
row = ceil(P/2);
column = floor((N-M)/D)+1;
dt = D/fs;
f2 = zeros(1, row);
t3 = zeros(1, column);
m1 = zeros(row, column);
%get matrix m1
for m = 1 : column
    t3(m) = m*dt; %get vector t3
    temp = zeros(1, P);
    tempcolumn = zeros(1, P);
    temp(1:M) = x((m-1)*D+1 : (m-1)*D+M);
    for k = 1 : P
        for n = 1 : P
            tempcolumn(k) = tempcolumn(k) + temp(n)*exp(-
1i*2*pi*k*n/P);
        end
    end
    m1(1:row, m) = tempcolumn(1:row);
end
%get the vector f2 to store omgea
w = fftshift((0:P-1)/P*2*pi);
w(1:P/2) = w(1:P/2)-2*pi;
f2 = w(P/2+1:P)/(2*pi)*fs;
%plot
figure(number);
imagesc(t3, f2(1:row), abs(m1(1:row, :))); %row/10 for more
information
xlabel('time(s)');
ylabel('frequency(Hz)');
title('Spectrogram');
end
```

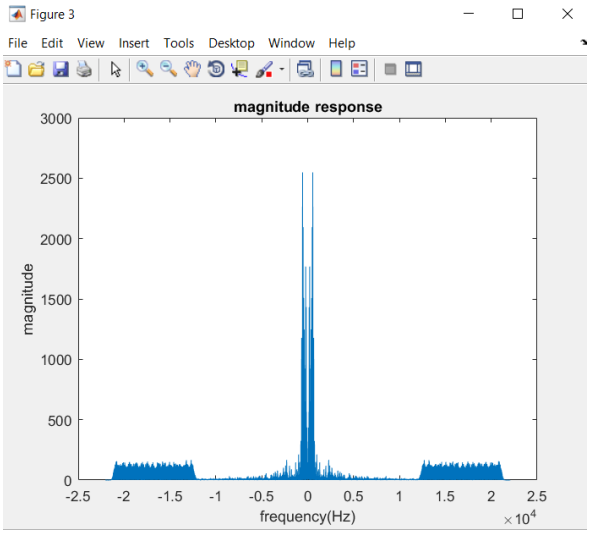
main program:

```
%Report Item 6
[y, fs] = audioread('sound1.wav');%no wavread in Matlab, use
audioread instead
%find magnitude spectrum
N = length(y);
Wy = fft(y);
shiftWy = fftshift(Wy);
omega = fftshift((0:N-1)/N*2*pi);
omega(1:N/2) = omega(1:N/2)-2*pi;
%filter
f = [0.4*fs/2, 0.6*fs/2];
a = [1, 0];
rp = 3;
rs = 50;
dev = [(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];
[n, fo, mo, w] = firpmord(f, a, dev, fs);
b = firpm(n, fo, mo, w);
%apply the filter
yfilter = filter(b,1, y);
soundsc([y; yfilter], fs); %play the original sound and the
filterd sound together

Wyfilter = fft(yfilter);
shiftWyfilter = fftshift(Wyfilter);
figure(3);
plot(omega/(2*pi)*fs, abs(shiftWy));
xlabel('frequency(Hz)');
ylabel('magnitude');
title('magnitude response');
figure(4);
plot(omega/(2*pi)*fs, abs(shiftWyfilter));
xlabel('frequency(Hz)');
ylabel('magnitude');
title('magnitude response after filtering');
mySTDFT(y, fs, 256, 512, 512, 1);
mySTDFT(yfilter, fs, 256, 512, 512, 2);
```

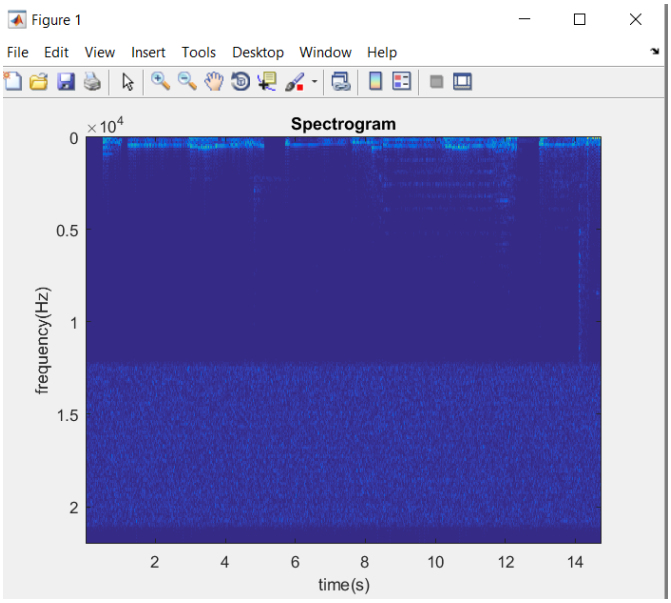
Explanation:

Fs = 44100 Hz and N = 650000. So T = 650000/44100 = 14.74 seconds. The sound file has 14.74 seconds.

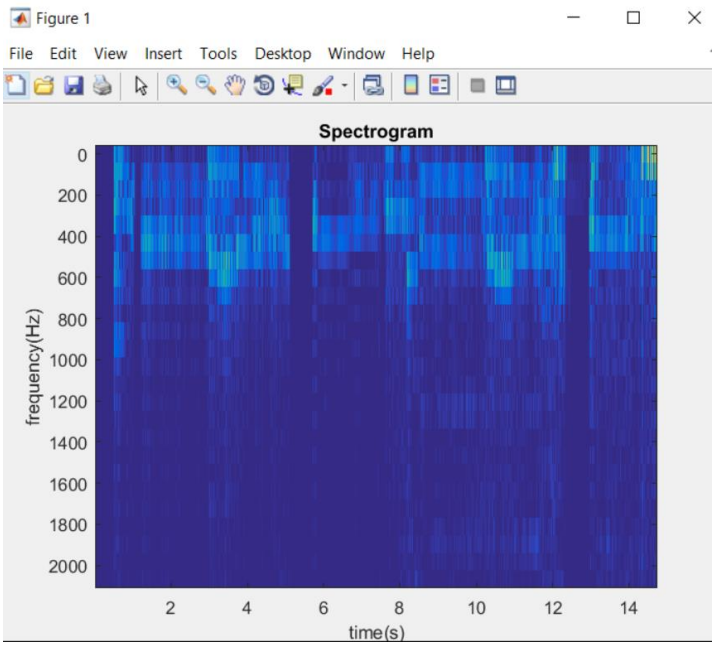
The figure 3 below is the original magnitude spectrum:



The figure 1 below is the original spectrogram of the signal with M = 256, D = 512, P = 512:

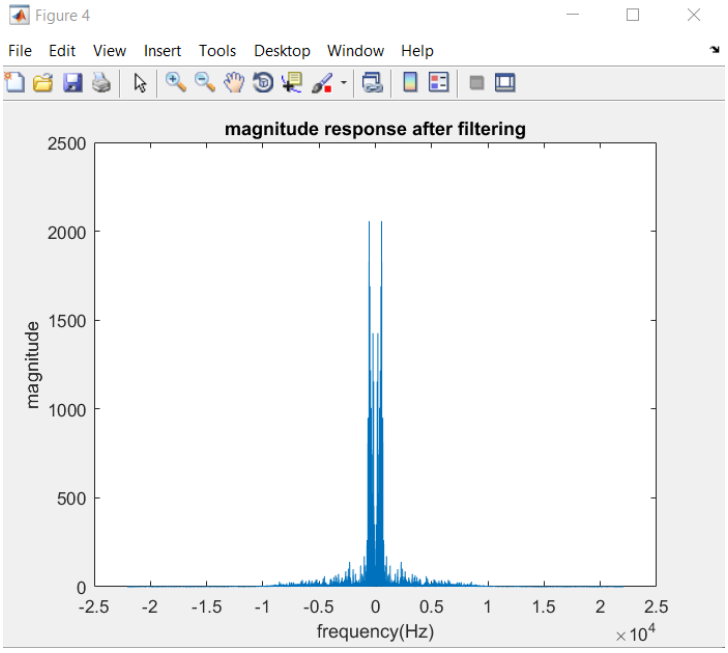


In order to see more information, I plot 1/10 of the whole frequency spectrogram in figure 1 below:

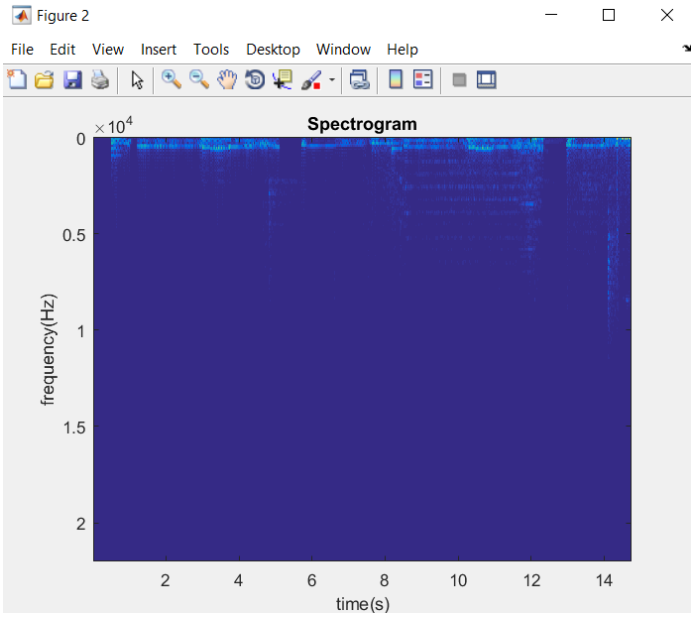


Therefore, I can find more information in the lower frequency region.

After applying a low-pass filter, the magnitude spectrum is plot in figure 4:



The figure 2 below is the spectrogram of the signal after filtering with M = 256, D = 512, P = 512:



Comparing the sound file before and after the filtering, I found that the high-frequency white noise has been largely cancelled out thanks to the low-pass filter. The filter's effectiveness is pretty good.

Report Item 7:

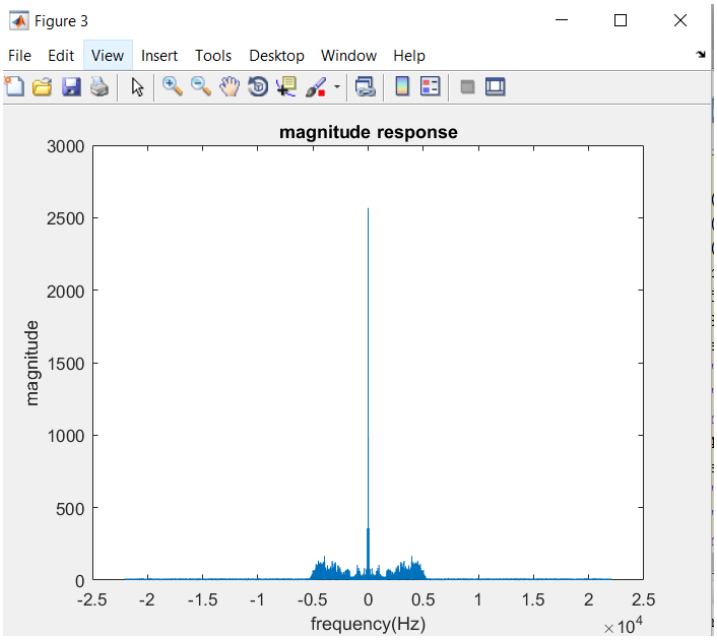
Code part:

```
%Report Item 7
[y, fs] = audioread('sound2.wav');
%find magnitude spectrum
N = length(y);
Wy = fft(y);
shiftWy = fftshift(Wy);
omega = fftshift((0:N-1)/N*2*pi);
omega(1:N/2) = omega(1:N/2)-2*pi;
%filter
f = [5000, 6000];
a = [1, 0];
rp = 3;
rs = 50;
dev = [(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];
[n, fo, mo, w] = firpmord(f, a, dev, fs);
```

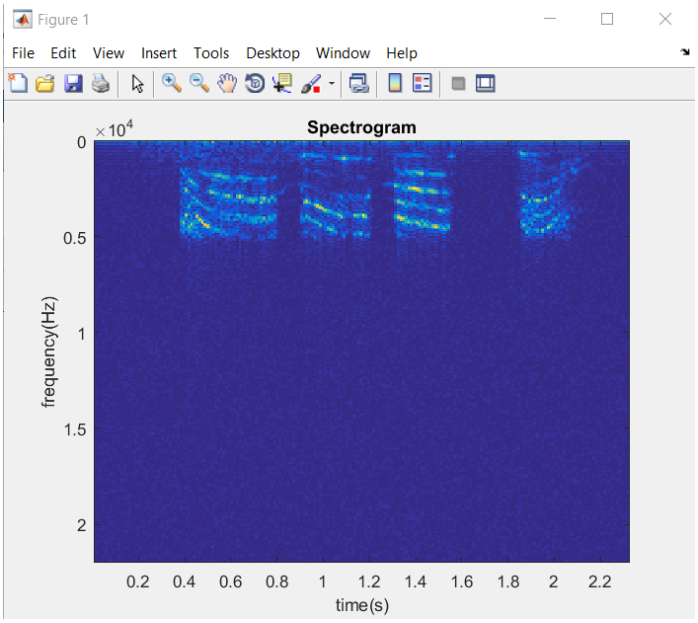
```
b = firpm(n, fo, mo, w);
%apply the filter
yfilter = filter(b,1, y);
soundsc([y; yfilter], fs); %play the original sound and the
filterd sound together
mySTDFT(y, fs, 256, 512, 512, 1);
mySTDFT(yfilter, fs, 256, 512, 512, 2);
Wyfilter = fft(yfilter);
shiftWyfilter = fftshift(Wyfilter);
figure(3);
plot(omega/(2*pi)*fs, abs(shiftWy));
xlabel('frequency(Hz)');
ylabel('magnitude');
title('magnitude response');
figure(4);
plot(omega/(2*pi)*fs, abs(shiftWyfilter));
xlabel('frequency(Hz)');
ylabel('magnitude');
title('magnitude response after filtering');
```

Explanation:

The figure 3 below is the original magnitude spectrum:

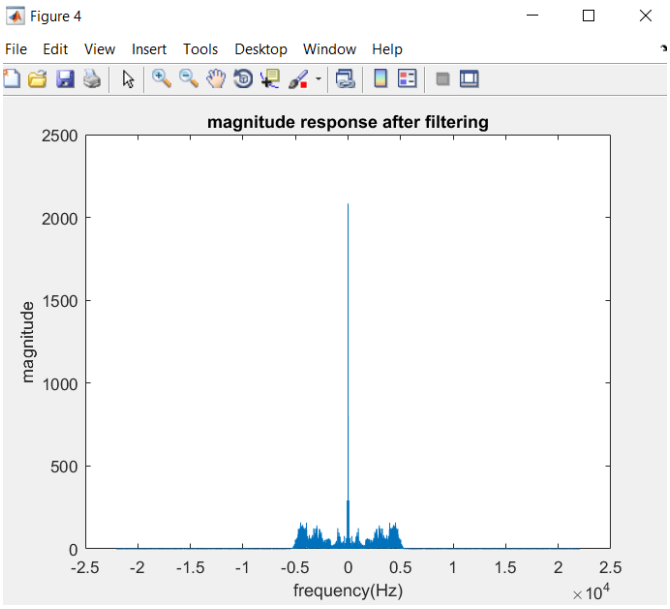


The figure 1 below is the original spectrogram of the signal with M = 256, D = 512, P = 512:

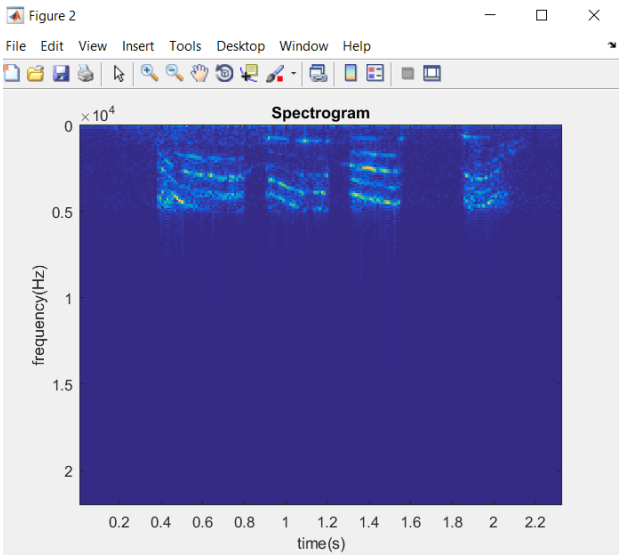


From the magnitude spectrum, I found that there is some white noise located greater than 5000Hz. After examining the spectrogram, I found that the white noise actually appears in the whole frequency region. The reason that magnitude spectrum fail to capture the noise is because that the sound is also located the same region of the spectrum.

After applying a low-pass filter, the magnitude spectrum is plot in figure 4:



The figure 2 below is the spectrogram of the signal after filtering with M = 256, D = 512, P = 512:



Comparing the sound file before and after the filtering, I found that the high-frequency white noise had been largely cancelled out but the low-frequency white noise was still there. But the overall quality had been improved.

Report Item 8:

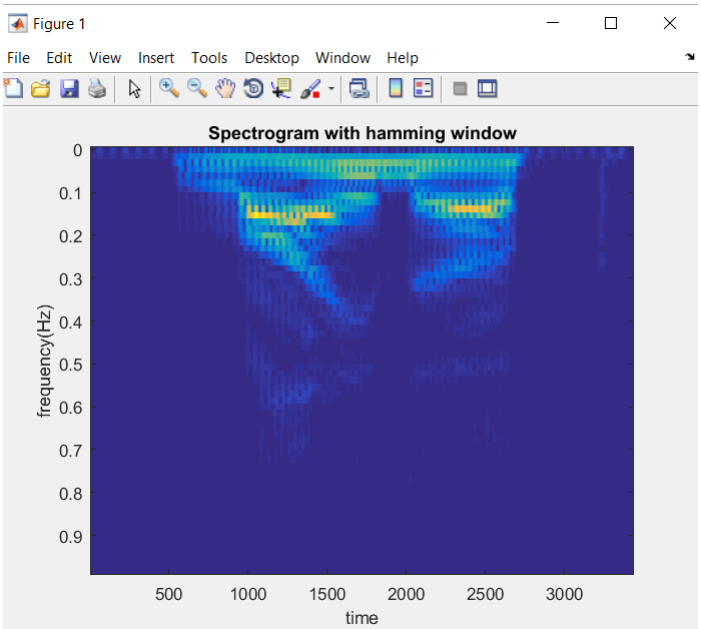
Code part:

```
%report Item 8:
load('speechsig.mat');
leng = length(x);
N = 128;%fft length
m = 2;%step size
row = N/2+1;
column = floor((leng - N) / m)+1;
f2 = zeros(1, row);
t3 = zeros(1, column);
m1 = zeros(row, column);
m2 = zeros(row, column);
hamming = zeros(1, N);
rect = zeros(1, N);
%get hamming window
for i = 1 : N
    hamming(i) = (0.54-0.46*cos(2*pi*i/(N-1)));
    rect(i) = 1;
end
%get matrix m1
```

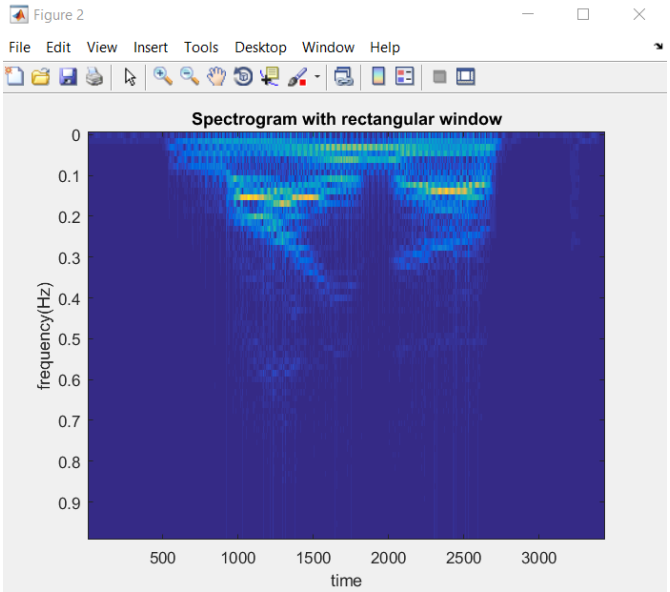
```
for i = 1 : column
    t3(i) = i; %get vector t3
    temp = zeros(1, N);
    tempcolumn = zeros(1, N);
    temp(1:N) = x((i-1)*m+1 : (i-1)*m+N);
    tempcolumn = abs(fft(hamming.*temp));
    m1(1:row, i) = tempcolumn(1:row);
    tempcolumn = abs(fft(rect.*temp));
    m2(1:row, i) = tempcolumn(1:row);
end
%get the vector f2 to store omgea
w = fftshift((0:N-1)/N*2*pi);
w(1:N/2) = w(1:N/2)-2*pi;
f2 = w(N/2+1:N)/pi;
%plot
figure(1);
imagesc(t3, f2, m1); %row/10 for more information
xlabel('time');
ylabel('frequency(Hz)');
title('Spectrogram with hamming window');
figure(2);
imagesc(t3, f2, m2); %row/10 for more information
xlabel('time');
ylabel('frequency(Hz)');
title('Spectrogram with rectangular window');
```

Explanation:

The figure 1 below is the spectrogram of the signal with hamming window:



The figure 2 below is the spectrogram of the signal with rectangular window:



From the two spectrograms, I found that there was some higher frequency still existed in the spectrogram with rectangular window. However, there was little higher frequency existed in the spectrogram with hamming window. That might be because that hamming window performs better than the rectangular window while building the low-pass filter.