EE 313 Linear Signals & Systems (Fall 2018)

Solution Set for Mini-Project #2 on Octave Band Filtering for Audio Signals

Mr. Houshang Salimian and Prof. Brian L. Evans

1- Introduction (5 points)

A finite impulse response (FIR) filter has an impulse response that settles to zero after a finite amount of time. In lectures, homeworks, and tuneups, we had discussed an *L*-point averaging filter, and observed its frequency response to be lowpass. We had also discussed a first-order difference FIR filter, and observed its frequency response as a highpass filter. Applications might require lowpass, highpass or other kinds of frequency selectivity.

When detecting notes being played from the fourth octave of a piano keyboard, a lowpass filter such as an *L*-point averaging filter would not be useful. Instead, we would require a bandpass filter to pass the frequencies in the fourth octave. In this project, we will analyze and design bandpass FIR filters.

2- Overview (5 points)

As mentioned in the Introduction, an application of FIR filters is to detect the note frequencies in an audio signal, which may cover different ranges in the frequency domain. In this project, our goal is to generate a tool that receives an audio signal and detects its octave by generating a scoring vector. This tool helps us to identify the range of input signal's frequency in time domain.

A piano keyboard is made up of 88 keys, which spans 7 full octaves of 12 keys in each. The notes in one octave are at twice the frequency of the corresponding notes in next lower octave. Also, the ratio between frequencies of successive notes are constant and equal to $2^{1/12}$. By using this property, and setting $f_{key=49}$ = 440 Hz, the frequency of each note can be calculated by the following equation

$$f(n) = 2^{\frac{n-49}{12}} \times 440$$

where *n* is the key number. For example for *n* = 16, *f*(16) = 65.4064 Hz.

3- Warmup (20 points)

3.1- A bandpass FIR filter can be defined in different ways. One approach is to base the design on a rectangular window of length *L* by defining the filter coefficients as

$$h[n] = \frac{2}{L}\cos(\hat{\omega}_c n), \qquad 0 \le n < L$$

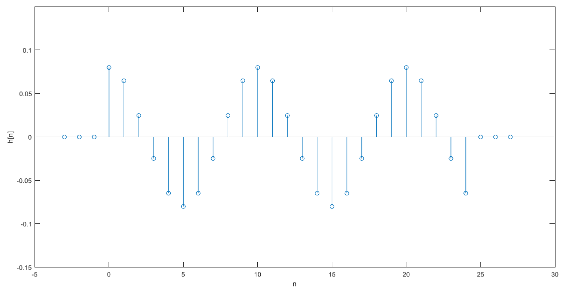
where $\hat{\omega}_c$ is the center frequency and *L* is the filter length. The rectangular pulse lasts for $0 \le n < L$ with amplitude 2/*L*. By selecting *L* = 25 and $\hat{\omega}_c = 0.2\pi$, the following code plots the impulse response and frequency response of this bandpass filter:

a)

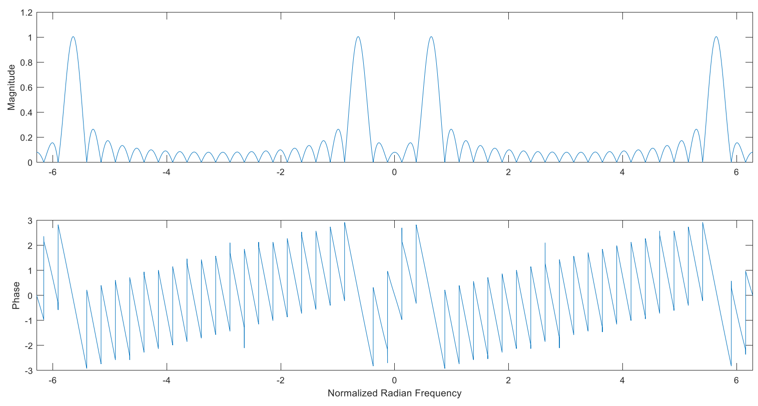
```
L = 25; % L is Length = 25
w_c = 0.2*pi; %center frequency
n = 0:(L-1);
h = 2/L*cos(w_c*n);    %filter coefficients
n2 = [-3 -2 -1 n 25 26 27];
hnew = [0 0 0 h 0 0 0];
%-------part a-------------------------
figure
stem(n2,hnew)    %plotting impulse response
ylim ([-0.15 0.15])
xlabel("n")
ylabel("h[n]")
%-------part b-------------------------
ww = -2*pi:(pi/10000):2*pi; %-- omega hat frequency axis
HH = freqz(h, 1, ww);
figure
subplot(2,1,1);
plot(ww, abs(HH))
xlim([-2*pi 2*pi])
ylabel('Magnitude')
subplot(2,1,2);
plot(ww, angle(HH))
xlim([-2*pi 2*pi])
ylabel('Phase')
xlabel("Normalized Radian Frequency")
```

T

a) The impulse response is shown below:



b) The frequency response appears below. Magnitude response is in linear units.

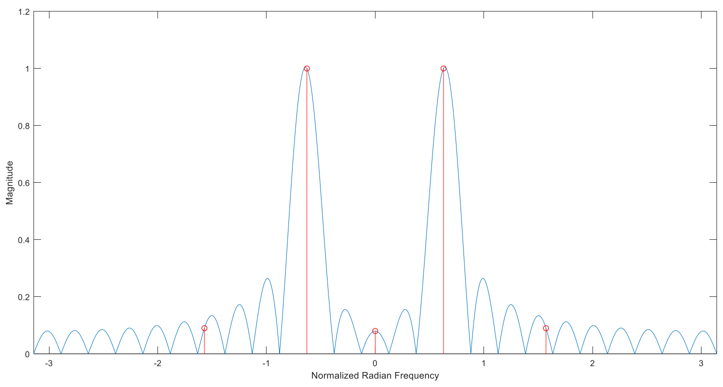


c) The bandpass cutoff points are selected as 50% of the peak value in linear units. Since this filter is normalized in magnitude, its peak is equal to one. Using the data cursor in the MATLAB plot window, the bandwidth can be estimated as

Magnitude(Passband1) = 0.5 $\rightarrow \widehat{\omega}_{P1}$ = 0.4873 rad/sample
Magnitude(Passband2) = 0.5 $\rightarrow \widehat{\omega}_{P2}$ = 0.7885 rad/sample
Bandwidth = $\widehat{\omega}_{P2} - \widehat{\omega}_{P1}$ = 0.3012 rad/sample.

3.2- Placing markers on the plot helps to find values on the plot more easily.

```
L = 25;
w_c = 0.2*pi;
n = 0:(L-1);
h = 2/L*cos(w_c*n);    %-- Filter Coefficients
ww = -pi:(pi/10000):pi; %-- omega hat frequency axis
HH = freqz(h, 1, ww);
plot(ww, abs(HH))
xlim([-pi pi])
ylabel('Magnitude')
xlabel('Normalized Radian Frequency')
hold on %hold on  prevents overwriting on the previous plot
m = pi*[-0.5 -0.2 0 0.2 0.5];
H2 = freqz(h,1,m);
stem(m, abs(H2), 'r')   %using stem for placing markers
hold off
```



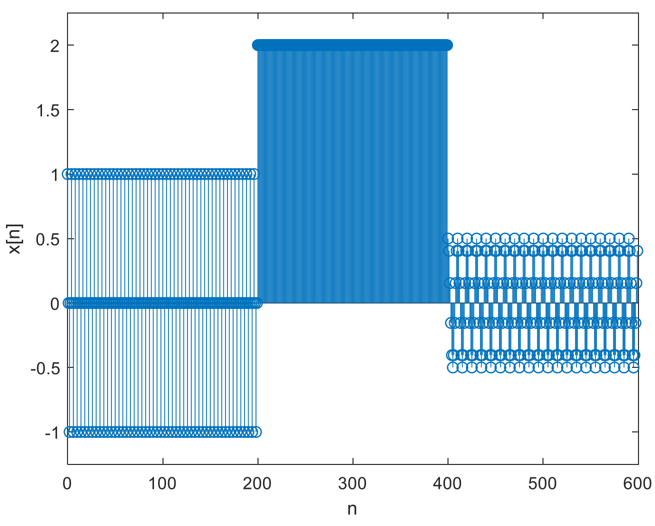
$\widehat{\omega}$ (rad)	-0.5π	-0.2π	0	0.2π	0.5π
Magnitude	0.0899	1	0.08	1	0.0899
Phase (rad)	1.676	0	0	0	-1.676

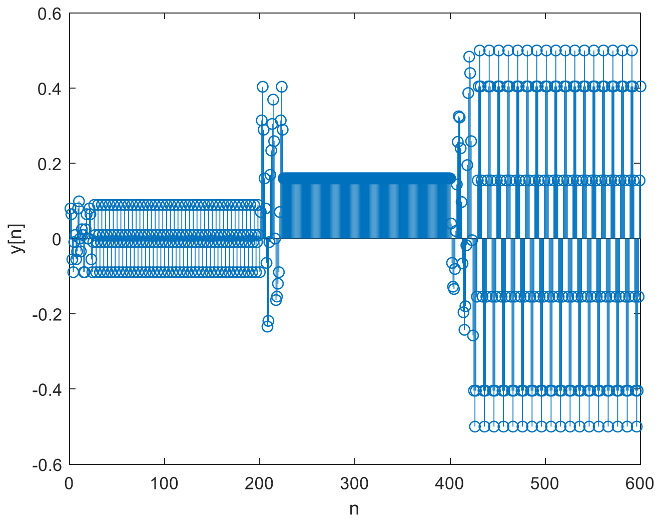
The center frequency for this filter is 0.2π. In the above table it can be seen that magnitude and phase for inputs with $\widehat{\omega}$=±0.2π remains unchanged. On the other side, for frequencies outside of bandpass range, for $\widehat{\omega}$=0, ±0.5π, the magnitude of frequency response is considerably low.

3.3- In MATLAB, one can concatenate vectors x1 and x2 into vector x via x = [x1 x2];

3.4- We input a signal consisting of one principal frequency that changes every 200 samples into the bandpass filter to see the effect of the filter. The principal frequencies in rad/sample are initially 0.5π, then 0 and finally 0.2π.

```
n1 = 0:199;
xx = cos(0.5*pi*n1);

n3 = 400:599;
x_new = 0.5*cos(0.2*pi*n3);

n = 0:599;
xx = [xx 2*ones(1,200) x_new];

figure
stem(n,xx)
xlabel('n')
ylabel('x[n]')
ylim ([-1.25 2.25])

w_c = 0.2*pi;
L = 25;
m = 0:(L-1);
h = 2/L*cos(w_c*m);
yy = filter(h,1,xx);
figure
stem(n,yy)
xlabel('n')
ylabel('y[n]')
```

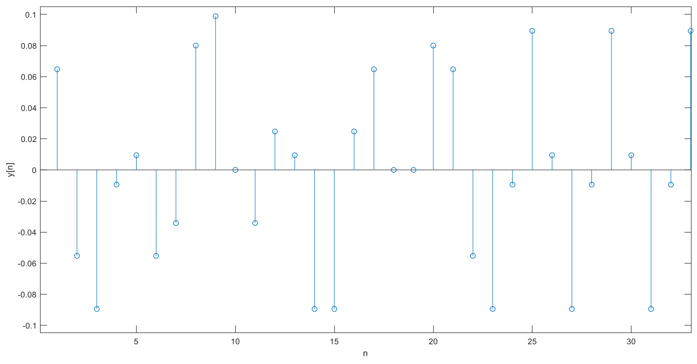


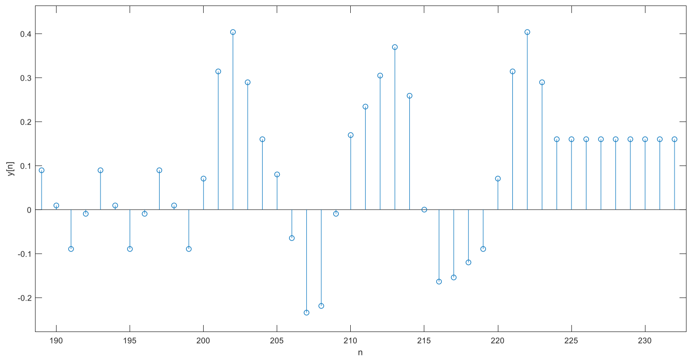


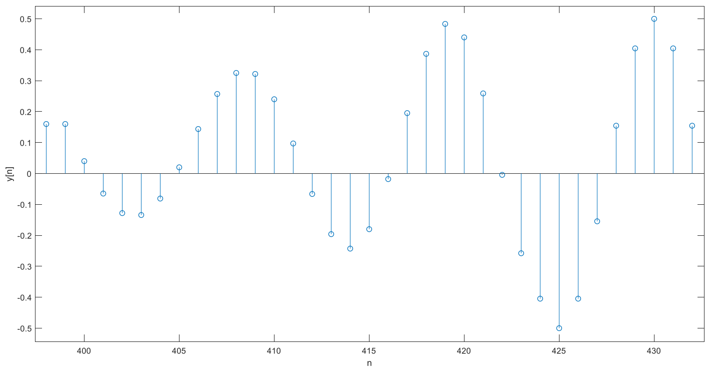
The filter's effect on the input signal can be seen in the output. Over n=0:399, the output signal has weakened, whereas the filter has kept the magnitude of the input for $\hat{\omega}$=0.2π constant.

n	0-199 (ω =0.5π)	200-399 (ω =0)	400-599 (ω =0.2π)
Input Amplitude	1	2	0.5
Output Amplitude	0.0899	0.16	0.5
Phase shift	-1.676	0	0

Also, the transients starting at *n* = 0, 200, 400 are shown in the following figure. The first transient happens from *n* = 0 to 23, its duration is equal to *L*-1 = 24, and the second and third transients are from *n* = 200 to 223 and *n* = 400 to 423, respectively. This happens, because the filter is faced with a signal with different frequency, and the output signal is *L*-1 samples longer than the input signal.

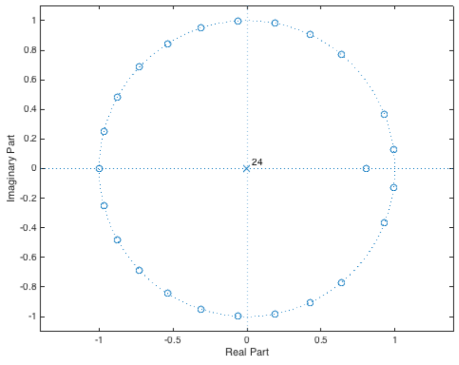






3.5 Pole-Zero Diagram for the Bandpass Filter (not required but useful)

The pole-zero diagram for the bandpass filter with L = 25 and $\hat{\omega}$=0.2π can provide insight into the filter design. The angles (frequencies) of the zeros indicate the stopband of the filter. The angles of the two gaps in the pattern of the zeros indicate the passbands of the filter.



```
L = 25;
w_c = 0.2*pi;
n = 0:(L-1);
h = 2/L*cos(w_c*n);
zplane(h);
```

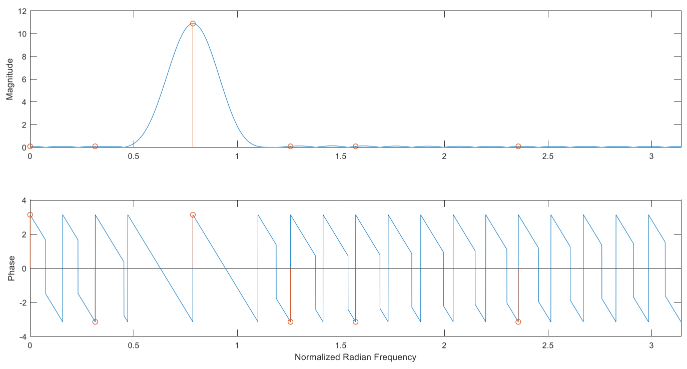
4. Bandpass filter design (20 points)

a) The previous section used a rectangular window as the basis for the bandpass filter. For frequencies far from the passband, we can detect peaks. In this Section, we will base the design on a Hamming window to obtain a much stronger attenuation in the stopband. The leftmost term in equation below is a Hamming window:

$$h[n]=\left(0.54-0.46\ \cos\left(2\pi n/\left(L-1\right)\right)\right)\ \cos\left(\hat{\omega}_c\left(n-\left(L-1\right)/2\right)\right), \qquad 0\le n<L-1$$

```
close all
clear all
clc

L =41;  %hamming window
w_c = 0.25*pi; %center frequency
n = 0:(L-1);
h = (0.54-0.46*cos(2*pi*n./(L-1))).*cos(w_c*(n-(L-1)/2));  %-- Filter
Coefficients with hamming window
ww = 0:(pi/10000):pi;
HH = freqz(h, 1, ww);

mm = pi*[0 0.1 0.25 0.4 0.5 0.75];
hm = freqz(h,1,mm);
M1= abs(hm);
Phi = angle(hm);

figure
subplot(2,1,1);
plot(ww, abs(HH))
hold on
stem(mm, M1)
xlim([0 pi])
ylabel('Magnitude')
hold off

subplot(2,1,2)
plot(ww, angle(HH))
hold on
stem(mm,Phi)
xlabel("Normalized Radian Frequency")
ylabel("Phase")
xlim([0 pi])
```

In the following plots, the range is selected as $0\le\hat{\omega}\le\pi$:



The following values are calculated by the help of red markers on the frequency response (magnitude and phase).

$\widehat{\omega}$	0	0.1π	0.25π	0.4π	0.5π	0.75π
Magnitude	0.08	0.08	10.88	1.257	0.08	0.08
Phase	π	-π	π	-π	-π	-π

b) The following code can be used for estimating bandwidths.

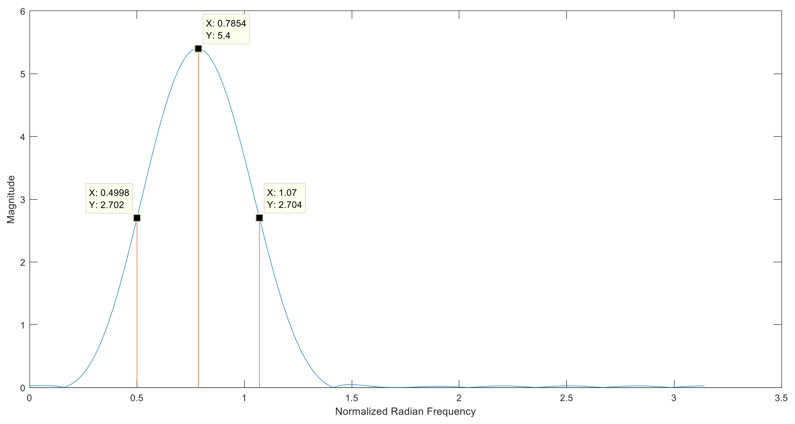
```
d = 0.5*max(abs(HH));    %set compare value = %50 of the peak value
F = find(abs(HH)>=d);    %finding values in the bandpass
flength = length(F);
BW = ww(F(flength))-ww(F(1));    %calculating bandwidth
```

L	21	41	81
Bandwidth	0.5699	0.2805	0.1414

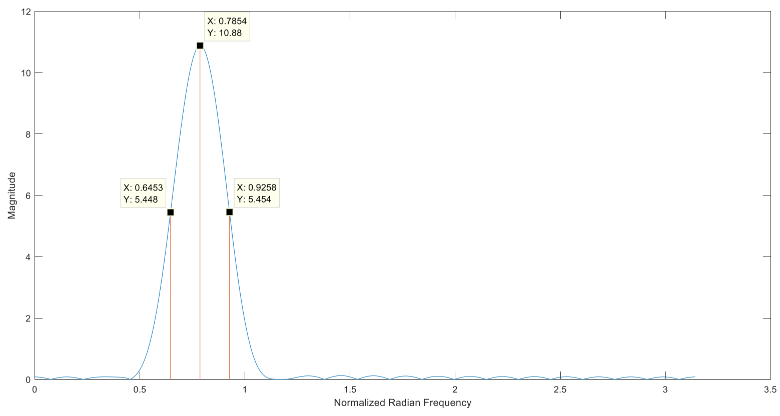
Doubling the value of *L* approximately halves the bandwidth (BW):

$$BW \propto \frac{1}{L}$$

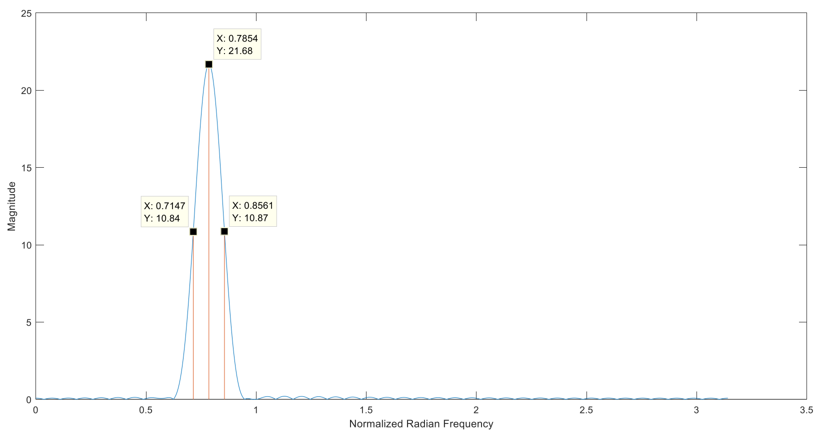
L =21



L = 41



L = 81



c)

$$x[n] = 2 + 2\cos\left(0.1\pi n + \frac{\pi}{3}\right) + \cos\left(0.25\pi n - \frac{\pi}{3}\right)$$

According to the table in Section 4.2.a for *L*=41 and $\hat{\omega}_c = 0.25\pi$

$$\left|H(e^{j0})\right| = 0.08, \qquad \angle H(e^{j0}) = \pi$$

$$\left|H(e^{j0.1\pi})\right| = 0.08, \qquad \angle H(e^{j0.1\pi}) = -\pi$$

$$\left|H(e^{j0.25\pi})\right| = 10.88, \qquad \angle H(e^{j0.25\pi}) = \pi$$

$$y[n] = (2\times 0.08) + (2\times 0.08)\cos\left(0.1\pi n + \frac{\pi}{3} - \pi\right) + 10.88\cos\left(0.25\pi n - \frac{\pi}{3} + \pi\right)$$

$$y[n] = 0.16 + 0.16\cos\left(0.1\pi n - \frac{2\pi}{3}\right) + 10.88\cos\left(0.25\pi n + \frac{2\pi}{3}\right)$$

According to the frequency response $\hat{\omega}$=0, 0.1π terms are in the stopband, their value is less than 0.01 of peak value, and due to their low amplitude after passing through the filter we cannot detect them in the output (Amplitude=0.16). $\hat{\omega}$=0.25π is the center frequency and is located in the passband, so it is the only term that shows in the output with Amplitude=10.88.

d) This filter passes frequencies around $\hat{\omega}$=0.25π with BW=0.02805 and rejects others in the stopband. For the values that are not present in passband or in stopband, their amplitude is low but not enough low to be neglected.

5- Piano Note Decoding. (40 points)

Using the formula in Section 2, we can calculate lower and upper frequencies for each octave. The center frequency (f_c) is average of lower (f_{low}) and upper (f_{up}) frequencies:

$$f_c = \frac{f_{low} + f_{up}}{2}$$

$$\hat{\omega} = 2\pi \frac{f}{f_s}, \qquad f_s = 8000Hz$$

Octave	Lower Frequency		Upper Frequency		Center Frequency		Band-width	
	Hertz	Rad/sam	Hertz	Rad/sam	Hertz	Rad/sam	Hertz	Rad/sam
2	65.40	0.05137	123.47	0.09697	94.43	0.07417	58.06	0.04560
3	130.81	0.10274	246.94	0.19394	188.87	0.14834	116.12	0.09120
4	261.62	0.20548	493.88	0.38789	377.75	0.29668	232.25	0.18241
5	523.25	0.41096	987.76	0.77579	755.50	0.59337	464.51	0.36482
6	1046.50	0.82192	1975.53	1.55158	1511.01	1.18675	929.03	0.72965

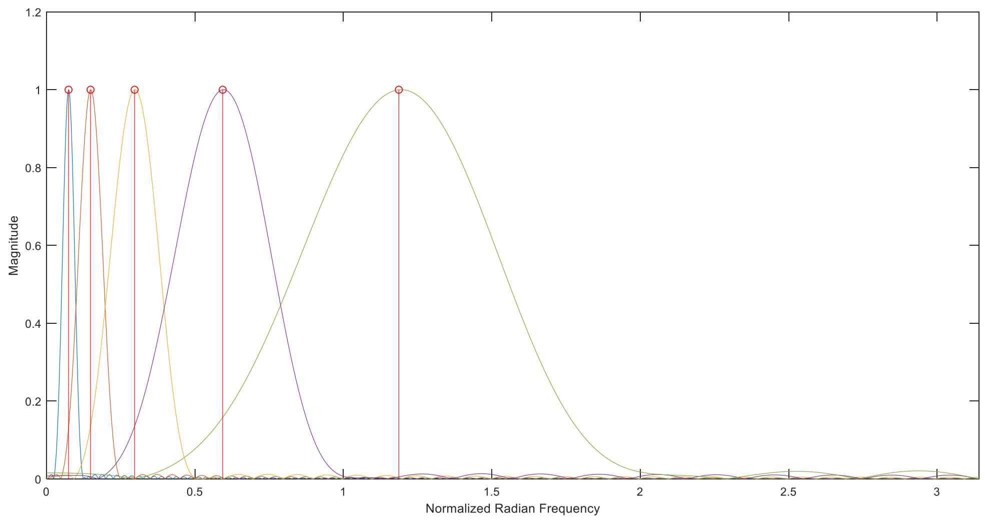
5.2- In Section 4, we saw that for a 41-point filter, the magnitude of frequency response reached to 10.88. For making magnitude of frequency response normalized, a constant value should be multiplied by coefficients.

$$h[n] = \beta\left(0.54 - 0.46\ \cos\left(2\pi n / \left(L-1\right)\right)\right)\ \cos\left(\hat{\omega}_c\left(n - \left(L-1\right)/2\right)\right), \qquad 0 \le n < L-1$$

We can use max() in MATLAB, to calculate β. In the following code normalized value is derived by this method.

```
fs = 8000;
ww = 0:(1/fs):pi; %-- omega hat frequency axis

%Octave 2
L2 = 251;   %filter size
w_c2 = 2*pi*94.4386/fs; %center frequency
n2 = 0:(L2-1);
bb2 = (0.54-0.46*cos(2*pi*n2./(L2-1))).*cos(w_c2*(n2-(L2-1)/2));  %--
Filter Coefficients
HH2 = freqz(bb2, 1, ww);
betha = 1/max(abs(HH2));
bb2 = betha*bb2;
HH2 = freqz(bb2, 1, ww);
plot(ww, abs(HH2))

hold on

%Octave 3
L3 = 126;
w_c3 = 2*pi*188.8773/fs;
n3 = 0:(L3-1);
bb3 = (0.54-0.46*cos(2*pi*n3./(L3-1))).*cos(w_c3*(n3-(L3-1)/2));
HH3 = freqz(bb3, 1, ww);
betha = 1/max(abs(HH3));
bb3 = betha*bb3;
HH3 = freqz(bb3, 1, ww);
plot(ww, abs(HH3))

%Octave 4
```

```
L4 =63;
w_c4 = 2*pi*377.7545/fs;
n4 = 0:(L4-1);
bb4 = (0.54-0.46*cos(2*pi*n4./(L4-1))).*cos(w_c4*(n4-(L4-1)/2));
HH4 = freqz(bb4, 1, ww);
betha = 1/max(abs(HH4));
bb4 = betha*bb4;
HH4 = freqz(bb4, 1, ww);
plot(ww, abs(HH4))

%Octave 5
L5 = 32;
w_c5 = 2*pi*755.5088/fs;
n5 = 0:(L5-1);
bb5 = (0.54-0.46*cos(2*pi*n5./(L5-1))).*cos(w_c5*(n5-(L5-1)/2));
HH5 = freqz(bb5, 1, ww);
betha = 1/max(abs(HH5));
bb5 = betha*bb5;
HH5 = freqz(bb5, 1, ww);
plot(ww, abs(HH5))

%Octave 6
L6 =16;
w_c6 = 2*pi*1511.017/fs;
n6 = 0:(L6-1);
bb6 = (0.54-0.46*cos(2*pi*n6./(L6-1))).*cos(w_c6*(n6-(L6-1)/2));  %--
Filter Coefficients
HH6 = freqz(bb6, 1, ww);
betha = 1/max(abs(HH6));
bb6 = betha*bb6;
HH6 = freqz(bb6, 1, ww);
plot(ww, abs(HH6))

stem ([w_c2 w_c3 w_c4 w_c5 w_c6], ones(1,5), 'r')
xlabel("Normalized Radian Frequency")
ylabel("Magnitude")
xlim([0 pi])
zoom on
```



By trial and error, the following values were selected for the bandpass filter for each octave. As mentioned in last section, doubling the bandwidth of the filter will mean that *L* will be halved. The issue is that *L* must remain an integer.

Octave	2	3	4	5	6
L	251	126	63	32	16

5.3- The following function is written for octaves 2 to 6. By calling this function we can get coefficients for the desired octave. For instance by writing hh = Octavefilter(2), we save coefficients of FIR filter for the second octave in hh.

```
function bb = Octavefilter(o)
fs = 8000;
ww = 0:(1/fs):pi; %-- omega hat frequency axis

%Octave 2
if (o == 2)
L2 = 251;
w_c2 = 2*pi*94.4386/fs;
n2 = 0:(L2-1);
bb2 = (0.54-0.46*cos(2*pi*n2./(L2-1))).*cos(w_c2*(n2-(L2-1)/2));  %--
Filter Coefficients
HH2 = freqz(bb2, 1, ww);
betha = 1/max(abs(HH2));
bb = betha*bb2;
elseif (o==3)
%Octave 3
L3 = 126;
w_c3 = 2*pi*188.8773/fs;
n3 = 0:(L3-1);
bb3 = (0.54-0.46*cos(2*pi*n3./(L3-1))).*cos(w_c3*(n3-(L3-1)/2));  %--
Filter Coefficients
HH3 = freqz(bb3, 1, ww);
betha = 1/max(abs(HH3));
bb = betha*bb3;
elseif (o == 4)
%Octave 4
L4 =63;
w_c4 = 2*pi*377.7545/fs;
n4 = 0:(L4-1);
bb4 = (0.54-0.46*cos(2*pi*n4./(L4-1))).*cos(w_c4*(n4-(L4-1)/2));  %--
Filter Coefficients
HH4 = freqz(bb4, 1, ww);
betha = 1/max(abs(HH4));
bb = betha*bb4;
elseif (o == 5)
%Octave 5
L5 = 32;
w_c5 = 2*pi*755.5088/fs;
n5 = 0:(L5-1);
bb5 = (0.54-0.46*cos(2*pi*n5./(L5-1))).*cos(w_c5*(n5-(L5-1)/2));  %--
Filter Coefficients
HH5 = freqz(bb5, 1, ww);
betha = 1/max(abs(HH5));
bb = betha*bb5;
elseif(o==6)
%Octave 6
L6 =16;
w_c6 = 2*pi*1511.017/fs;
n6 = 0:(L6-1);
bb6 = (0.54-0.46*cos(2*pi*n6./(L6-1))).*cos(w_c6*(n6-(L6-1)/2));  %--
Filter Coefficients
```

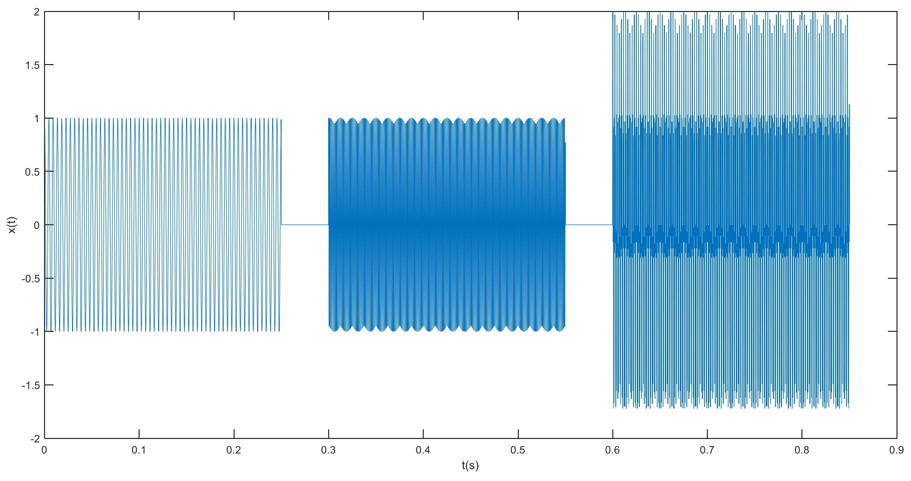
```
HH6 = freqz(bb6, 1, ww);
betha = 1/max(abs(HH6));
bb = betha*bb6;
end
end
```

a)

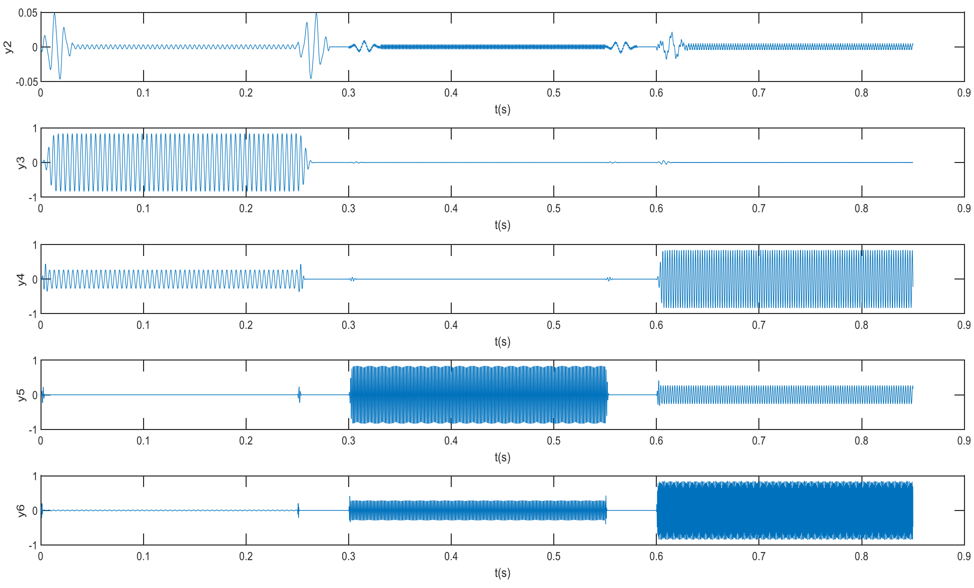
$$x(t) = \begin{cases} \cos\left(2\pi\left(220\right)t\right), & 0 \le t < 0.25 \\ \cos\left(2\pi\left(880\right)t\right), & 0.3 \le t < 0.55 \\ \cos\left(2\pi\left(440\right)t\right) + \cos\left(2\pi\left(1760\right)t\right), & 0.6 \le t < 0.85 \end{cases}$$

```
fs = 8000;
t = 0:1/fs:0.85;
%defining input
xx = cos(2*pi*220*t).*rectpuls(t-0.125,0.25)+cos(2*pi*880*t).*rectpuls(t-
0.425,0.25)+(cos(2*pi*440*t)+cos(2*pi*1760*t)).*rectpuls(t-0.725,0.25);
plot(t,xx)
ylim([-2 2])
xlabel('t(s)')
ylabel('x(t)')
%getting filter coefficients by calling Octavefilter.m
bb2 = Octavefilter(2);
bb3 = Octavefilter(3);
bb4 = Octavefilter(4);
bb5 = Octavefilter(5);
bb6 = Octavefilter(6);
%filter input(xx) via filter of each octave
y2 = filter (bb2,1,xx);
y3 = filter (bb3,1,xx);
y4 = filter (bb4,1,xx);
y5 = filter (bb5,1,xx);
y6 = filter (bb6,1,xx);

figure
subplot(5,1,1)
plot(t,y2)
xlabel('t(s)')
ylabel('y2')
subplot(5,1,2)
plot(t,y3)
xlabel('t(s)')
ylabel('y3')
subplot(5,1,3)
plot(t,y4)
xlabel('t(s)')
ylabel('y4')
subplot(5,1,4)
plot(t,y5)
xlabel('t(s)')
ylabel('y5')
subplot(5,1,5)
plot(t,y6)
xlabel('t(s)')
ylabel('y6')
```

Input signal:



Output signals after passing through the filter.



d) The frequency response for all present frequencies in the input are provided in the following table.

Freq. (Hz)	Octave 2		Octave 3		Octave 4		Octave 5		Octave 6	
	Mag.	Phase	Mag.	Phase	Mag.	Phase	Mag.	Phase	Mag.	Phase.
220	0.0029	π/8	0.8324	1.767	0.2669	0.9267	0.0002	-2.678	0.0120	1.8456
440	0.0038	π/4	0.0018	π/8	0.8340	1.8535	0.2625	0.9267	0.0099	-2.591
880	0.0032	π/2	0.0028	π/4	0.0013	0.5654	0.8306	1.8535	0.2835	1.0995
1760	0.0011	-π	0.0028	π/2	0.0029	1.1309	0.0036	-2.576	0.8390	2.1991

Based on the above table, the output plot is correct. For instance, all the terms in input are in stopband of octave 2, consequently it has blocked the input and the magnitude is almost zero. On the other hand, in octave 3, we can see it will pass the first part of input because its frequency is in its bandpass.

For octave 4, one term in part 3 of input (*f* = 440 Hz) has passed with almost the same amplitude of input signal. Part 2 of input is blocked because its frequency (*f* = 880 Hz) is in the stopband, and part 1 is present with a lower amplitude, because *f* = 220 Hz is neither in bandpass nor in stopband of octave 4.

e) The longest period of transient is for octave 2, and the shortest transient happens in filter of octave 6. This relates to the size of filter (*L*). The following formula shows relation between transient and filter's size.

$$transient = \frac{L-1}{f_s} = \frac{L-1}{8000}$$

Octave number	2	3	4	5	6
Transient (s)	0.03125	0.015625	0.00775	0.003875	0.001875

5.4- The MATLAB code for scoring function:

```
function score = octavescore(xx,hh,fs)

L = length(hh);  % returns value of L for hh filter

y = conv (hh,xx); %y is out put after passing through the filter
ystart = ceil((L-1)/2); %computing the delay, ceil converts it to an
integer value(closest upper value)
yend = length(y)-(L-ystart);
ynew = y(ystart:yend);  %y_delay is the output signal after clearing first
(L-1)/2 cells of y

ol=length(ynew); %output length

k = fs*0.050; % k shows number of samples in each 50 ms
sections_number = ceil(ol/k); % length of score vector
V = zeros(1,sections_number);
score = zeros(1,sections_number);
for m = 1:sections_number
    Q = zeros(1,k);
    r = (m-1)*k;
    if((r+k)<ol)
        Q = abs(ynew(r+1:r+k));
    else
        Q(r+1:ol) = abs(ynew(r+1:ol));
    end
    V(1,m) = max(Q);
    if (V(1,m)>=0.5)
        score(1,m) = 1;
    else
        score(1,m) = 0;
    end
end
```

In the following code, labtest.mat is read by MATLAB and has been saved in xx as the input to the system. Finally, going through filter and scoring function, the score vector (W) is calculated for each vector.

```
load labtest.mat
filename = 'labtest.flac';
audiowrite(filename,xx,fs)
clear xx fs
[xx, fs]=audioread('labtest.flac');
for i=2:6
h = Octavefilter(i);
W(i-1,:) = octavescore(xx,h,fs);
end
```

In matrix W, we can find the score value for octave 2 to 6, in rows 1-6 respectively. The vector has divided audio playing time (3.77 s) into 76 sections, i.e. each section covers 50ms of time domain.

In following table, score value for each octaves are shown in columns.

Octave 2	Octave 3	Octave 4	Octave 5	Octave 6
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	1	1
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	1	1	0
0	0	1	0	0
0	0	1	1	0
0	0	1	1	0
0	0	1	1	0
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	1	1	1
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
1	1	1	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0

1	0	0	0	0
1	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	1	0	0	1
0	1	0	0	1
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
1	1	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	0	0	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	1	1
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	1
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	1	0	1	0
0	1	0	0	0
0	1	0	0	0
0	0	0	0	0
0	0	0	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	1	1

6- Conclusion (10 points)

In this mini-project we studied FIR bandpass filter in depth, and developed a tool to identify the range of frequencies among piano notes.

Based on a rectangular window, we first designed a bandpass filter that is based on two parameters: center frequency and length. By plotting the frequency response, we saw that the filter has the highest magnitude at the center frequency— the magnitude decreases as the frequency increases (or decreases) away from the center frequency. We tested the filter by passing an input signal with different frequency components and validated that the filter reduces frequency components in the stopband and passes frequency components in the passband. We also discussed transients, which happen when a frequency in the input signal changes abruptly.

Next, we tried out a different bandpass filter design based on the Hamming window. When using a rectangular window, the filter could not sufficiently attenuate the frequencies in the stopband which could lead to false positives when detecting piano note octaves. Comparing these two windows, we observed that the Hamming window gives a bandpass filter with much stronger attenuation in the stopband (40 dB vs. 13.5 db). In this part, we also analyzed the effect of window size on the bandpass filter frequency response. For the same center frequency, doubling the window size halved the passband.

Finally, we designed an audio signal analyzer that can detect the range of frequencies that the frequency components of an input signal belong. This analyzer is composed of five parallel bandpass filters that pass octaves 2 to 6, respectively, of piano notes. Then, we generated a scoring function that uses the output of these filters and shows that in a certain time duration, the audio signal is consisted by which octave notes. This analyzer could be extended to detect other frequency ranges.