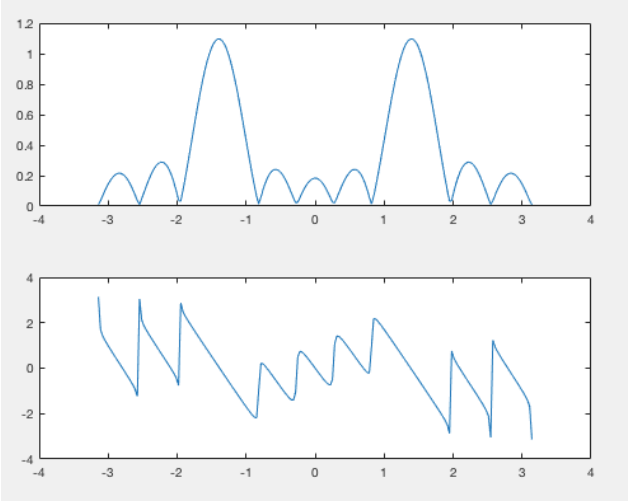
Taylor Rembos
EEL 3135
Lab 6 Part 2

2.1 Simple Bandpass Filter:

(a) Generate a bandpass filter that will pass a frequency component at $\hat{\omega} = 0.44\pi$, with $L = 10$. Submit necessary code and plots of magnitude and phase response. Measure the gain of the filter (i.e., the magnitude) at the three frequencies of interest: $\hat{\omega} = 0.3\pi$, $\hat{\omega} = 0.44\pi$ and $\hat{\omega} = 0.7\pi$.

```
>> L = 10;
>> n = 0:(L-1);
>> h = (2/L)*cos(0.44*pi*n);
>> ww = -pi:(pi/100):pi;
>> h = freqz(h,1,ww);
>> subplot(2,1,1)
>> plot(ww,abs(h));
>> subplot(2,1,2)
>> plot(ww, angle(h));
```



0.3*pi -> filter gain: 2.836
0.44*pi -> filter gain: 1.096
0.7*pi -> filter gain: 0.286

(b) For the $L = 10$ bandpass filter from part (a), determine the passband width (for passband defined above). Repeat the plot for $L = 20$ and $L = 40$, and explain how the width of the passband is related to filter length L (i.e., what happens when L is doubled or halved).

When the bandpass filter L is doubled, the passband width for the passband is halved.
Width is 0.4712. For L=20, width is 0.2356. For L = 40, width is 0.1178.

```
>> ww(find(abs(0.707 * max(h)) < abs(h)))

ans =

  Columns 1 through 6

   -1.6336   -1.6022   -1.5708   -1.5394   -1.5080   -1.4765

  Columns 7 through 12
```

```
  -1.4451   -1.4137   -1.3823   -1.3509   -1.3195   -1.2881

  Columns 13 through 18

  -1.2566   -1.2252   -1.1938   -1.1624    1.1624    1.1938

  Columns 19 through 24

   1.2252    1.2566    1.2881    1.3195    1.3509    1.3823

  Columns 25 through 30

   1.4137    1.4451    1.4765    1.5080    1.5394    1.5708

  Columns 31 through 32

   1.6022    1.6336

>> 1.6336-1.1624

ans =

   0.4712

>> L = 20

L =

   20

>> ww(find(abs(0.707 * max(hn)) < abs(hn)))

ans =

  Columns 1 through 6

  -1.6336   -1.6022   -1.5708   -1.5394   -1.5080   -1.4765

  Columns 7 through 12

  -1.4451   -1.4137   -1.3823   -1.3509   -1.3195   -1.2881

  Columns 13 through 18

  -1.2566   -1.2252   -1.1938   -1.1624    1.1624    1.1938
```

```
  Columns 19 through 24

    1.2252    1.2566    1.2881    1.3195    1.3509    1.3823

  Columns 25 through 30

    1.4137    1.4451    1.4765    1.5080    1.5394    1.5708

  Columns 31 through 32

    1.6022    1.6336
```

```
>> L = 40;
>> ww(find(abs(0.707 * max(hn)) < abs(hn)))

ans =

  Columns 1 through 6

   -1.6336   -1.6022   -1.5708   -1.5394   -1.5080   -1.4765

  Columns 7 through 12

   -1.4451   -1.4137   -1.3823   -1.3509   -1.3195   -1.2881

  Columns 13 through 18

   -1.2566   -1.2252   -1.1938   -1.1624    1.1624    1.1938

  Columns 19 through 24

    1.2252    1.2566    1.2881    1.3195    1.3509    1.3823

  Columns 25 through 30

    1.4137    1.4451    1.4765    1.5080    1.5394    1.5708

  Columns 31 through 32

    1.6022    1.6336
```

(c) Comment on the selectivity of the $L = 10$ bandpass filter. In other words, which frequencies are "passed by the filter" and which are "nulled by the filter". Use the frequency response to

explain how the filter can pass one component at $\hat{\omega} = 0.44\pi$, while reducing or rejecting the others at $\hat{\omega} = 0.3\pi$ and $\hat{\omega} = 0.7\pi$.

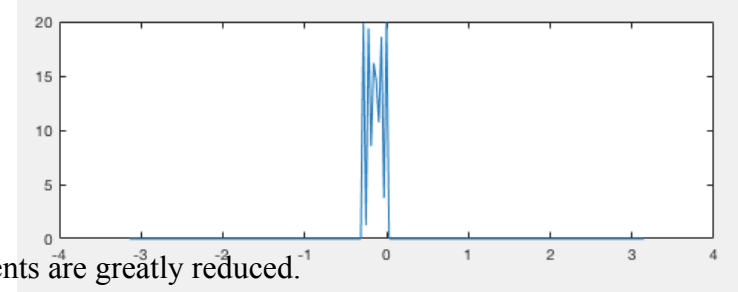
The frequencies that are "passed by the filter" are those that are frequencies from 1.1624 and 1.6336 width for L = 10. The rest would be "nulled by the filter". The component at 0.44*pi is allowed to pass through the filter because it is the maximum value. 0.3*pi and 0.7*pi are not and thus are the the "nulled" frequencies.

(d) Generate a bandpass filter that will pass the frequency component at $\hat{\omega} = 0.44\pi$, but now make the filter length (L) long enough so that it will also *greatly* reduce frequency components at (or near) $\hat{\omega} = 0.3\pi$ and $\hat{\omega} = 0.7\pi$. Submit code and frequency response plot. Determine the *smallest* value of L so that the following conditions both hold:

>> L = 30;

>> subplot(2,1,1)

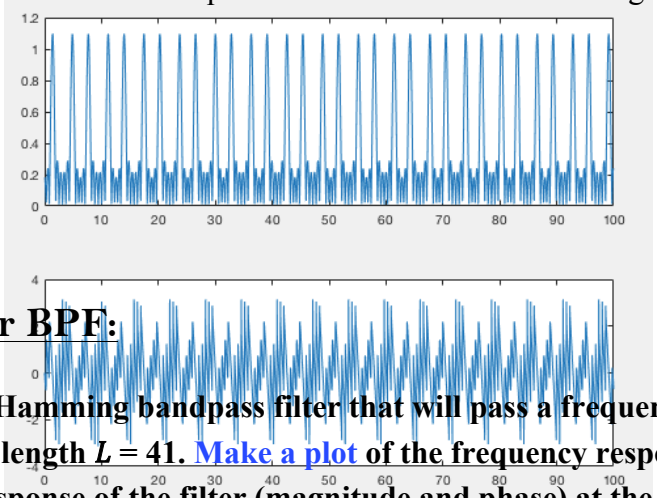
>> plot(ww,abs(hn));



At this filter length, the frequency components are greatly reduced.

(e) Use the filter from the previous part to filter the "sum of 3 sinusoids" signal from Section 2.1. Make a plot of 100 points (over time) of the input and output signals, and explain how the filter has reduced or removed two of the three sinusoidal components.

The filter has removed two of the three sinusoidal components by filterer out the frequencies that are not around the maximum value.
H(e^jw) can be used to determine the relative size of each sinusoidal component in the output signal because the amplitude of the frequencies is around 60% of the original amplitude.

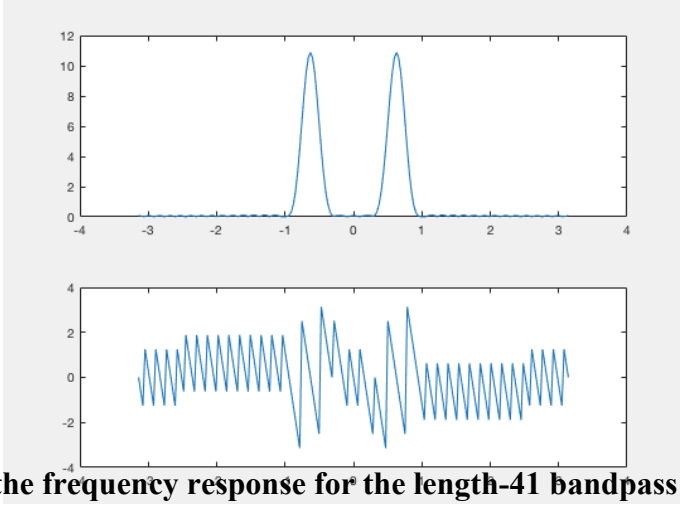


2.1 A Better BPF:

(a) Generate a Hamming bandpass filter that will pass a frequency component at $\hat{\omega} = 0.2\pi$. Make the filter length $L = 41$. Make a plot of the frequency response magnitude and phase. Measure the response of the filter (magnitude and phase) at the following frequencies of interest:

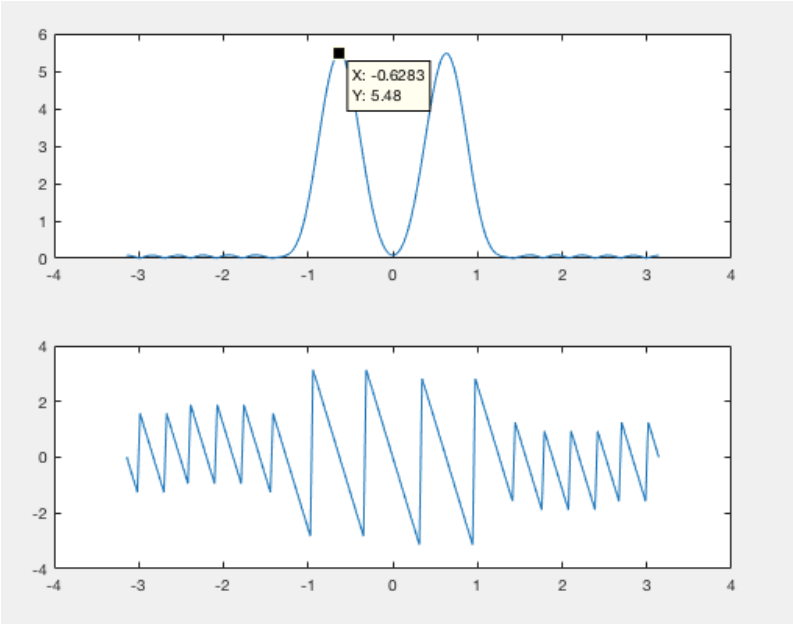
ω = {0, 0.1π, 0.25π, 0.4π, 0.5π, 0.75π}. Summarize the values in a table.

```
>> ww = -pi: (pi/100):pi;
>> wc = 0.2*pi;
>> L = 41;
>> n = 0:L-1;
>> h = (0.54-0.46*cos(2*pi*n/(L-1))).*cos(wc.*(n-((L-1)/2)));
>> hn = freqz(h,1,ww);
>> subplot(2,1,1)
>> plot(ww,abs(hn))
>> subplot(2,1,2)
>> plot(ww,angle(hn))
>> freq = [0, 0.1*pi, 0.25*pi, 0.4*pi, 0.5*pi, 0.75*pi];
>> mag = abs(freqz(h,1,freq));
>> ang = angle(freqz(h,1,freq));
>> mag

mag =

   0.0800    0.0800    4.5200    0.0800    0.0800    0.0800

>> ang

ang =

         0   -0.0000    3.1416   -0.0000    0.0000    0.0000
```



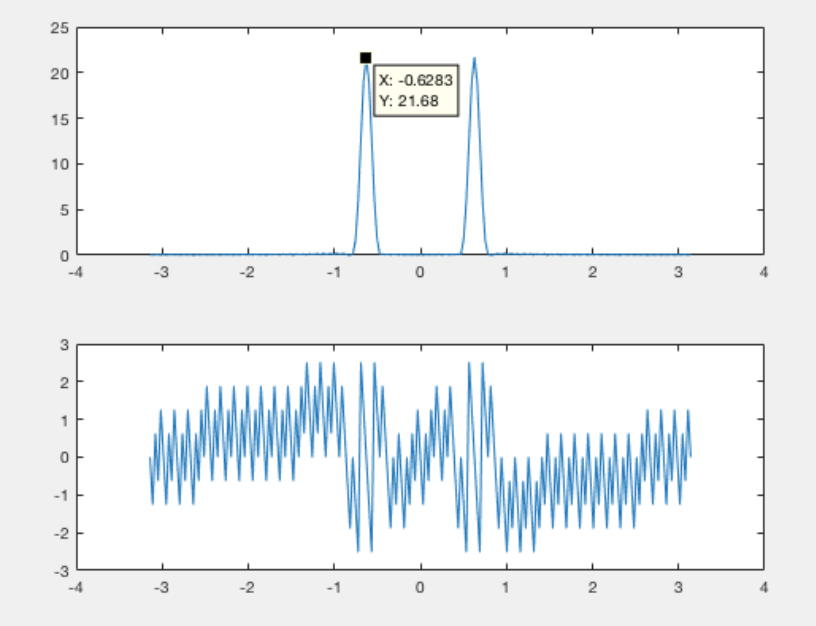
(b) Use the plot of the frequency response for the length-41 bandpass filter from part (a), and determine the passband width using the 50% level to define the pass band. Make two other plots of BPFs for *L* = 21and *L* = 81, and measure the passband width in both. Then explain how the width of the passband is related to filter length *L*, i.e., what happens when *L* is (approximately) doubled or halved.

```
>> bplength = length(find(abs(hn) >= 0.5*10.88));
>> bplength = (pi/100)*bplength/2

bplength =

    0.0628

>> L = 21;
>> n = 0:L-1;
>> h = (0.54-0.46*cos(2*pi*n/(L-1))).*cos(wc.*(n-((L-1)/2)));
>> hn = freqz(h,1,ww);
>> subplot(2,1,1)
>> plot(ww,abs(hn))
>> subplot(2,1,2)
>> plot(ww,angle(hn))
```




```
>> bplength = length(find(abs(hn) >= 0.5*5.48));
>> bplength = (pi/100)*bplength/2

bplength =

    0.5341

>> subplot(2,1,1)
>> plot(ww,abs(hn))
>> subplot(2,1,2)
>> plot(ww,angle(hn))
>> subplot(2,1,1)
>> plot(ww,abs(hn))
>> subplot(2,1,2)
>> plot(ww,angle(hn))
>> L = 81;
>> n = 0:L-1;
```

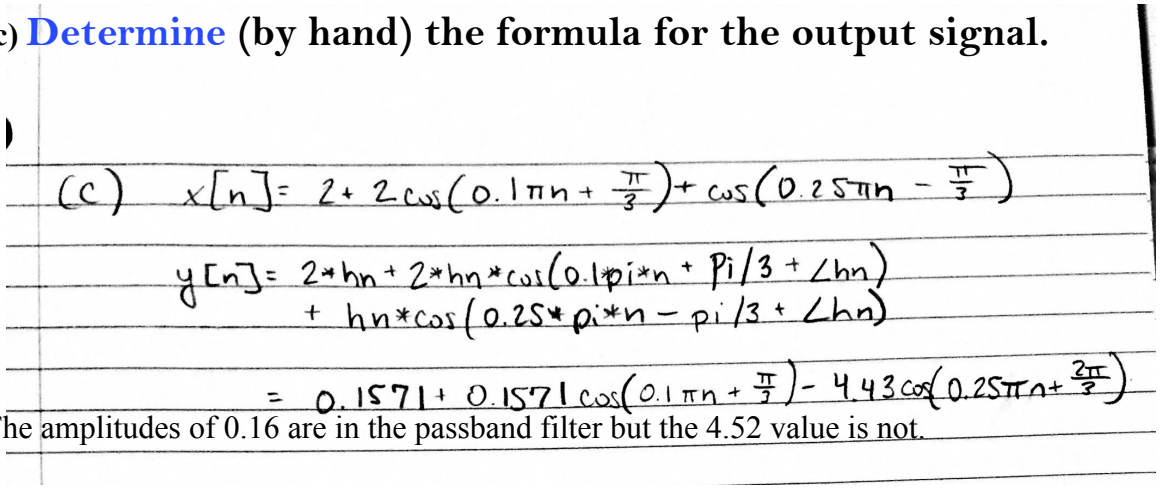
```
>> h = (0.54-0.46*cos(2*pi*n/(L-1))).*cos(wc.*(n-((L-1)/2)));
>> hn = freqz(h,1,ww);
>> subplot(2,1,1)
>> plot(ww,abs(hn))
>> subplot(2,1,2)
>> plot(ww,angle(hn))
```




```
>> bplength = length(find(abs(hn) >= 0.5*21.68));
>> bplength = (pi/100)*bplength/2

bplength =

    0.1571
```

As width of the passband increases, the bandpass filter changes by about 2.

(c) Determine (by hand) the formula for the output signal.



The amplitudes of 0.16 are in the passband filter but the 4.52 value is not.

(d) Use the frequency response (and passband width) of the length-41

bandpass filter to explain how the filter is able to pass the components at $\hat{\omega} = \pm 0.25\pi$, while reducing or rejecting others.

The filter rejects frequencies not within the amplitude centered around 0.25pi.

2.2 Octave Filtering

Below, we provide you with the lower edge, higher edge, and center frequencies of all octave (in Hz). Based on this, calculate the same parameters in terms of normalized radiant frequency $\hat{\omega}$, and submit a table consists of those values.

Octave	**2**	**3**	**4**	**5**	**6**
Lower Edge (key #)	16	28	40	52	64
Lower Edge (Hz)	65.4	130.8	261.6	523.25	1046.5
Higher Edge (key #)	27	39	51	63	75
Higher Edge (Hz)	123.5	246.9	493.9	987.8	1978.5
Center (Hz)	94.4	188.85	379.24	755.51	1511

```
>> w = freq.*2*pi;
>> out = 2.*(1/fs);
>> keynum = [16, 28, 40, 52, 64, 27, 39, 51, 63, 75];
>> freq = 440*(2.^((keynum-49)/12))

freq =
```

```
 1.0e+03 *

Columns 1 through 6

  0.0654    0.1308    0.2616    0.5233    1.0465    0.1235

Columns 7 through 10

  0.2469    0.4939
  0.9878    1.9755
```

	16	**28**	**40**	**52**	**64**	**27**	**39**	**51**	**63**	**75**
freq	65.4064	130.8128	261.6252	523.2511	1046.5	123.4708	246.9417	493.8833	987.766	1975.5
w	410.9605	821.921	1643.842	3287.684	6575.368	775.7901	1551.58	3103.16	6206.32	12412.64
out	0.0514	0.1028	0.2056	0.4112	0.8224	0.0970	0.1940	0.388	0.7758	1.5516

middle

	2	3	4	5	6
freq	94.4	188.8	377.6	755.2	1510.4
out	0.0742	0.1484	0.2968	0.5936	1.1872

2.3 Bandpass Filter Bank Design

(a) Devise a strategy for picking the constant β so that the maximum value of the frequency response will be equal to one. Write the one or two lines of MATLAB code that will do this scaling operation in general. We will use the following approach:

```
>> h = (0.54-0.46*cos(2*pi*n/(L-1))).*cos(wc.*(n-(L-1)/2));
>> B = max(abs(h));
>> h = h/B;
```

(b) For each filter, determine the length L that is required to get the correct bandwidth. Use the bandedges determined in Section 2.3. This filter design process will be trial- and-error, so each time you change L you will have to make a frequency response plot (magnitude only) to see if the filter is correct.

The length needs to be a factor of 256, 128, 64, 32, or 16.

```
>> A = octave - 2;
>> L = 256/(2^A);
```

(c) Generate the five (scaled) bandpass filters. Plot the magnitude of the frequency responses all together on one plot (the range $0 \leq \hat{\omega} \leq \pi$ is sufficient because $|H(e^{j\hat{\omega}})|$is symmetric). Indicate the locations of each of the center frequencies of the five octaves on this plot and illustrate that the passbands cover the separate octaves.

```
>> bplength = length(find(abs(hn) >= 0.5*hmax));
>> bplength = (pi/100)*bplength/2;
```

function pickB.m

```
function [ B ] = pickB( wc, octave )
%function to define passbands with constant B for scaling filter
A = octave - 2;
L = 256/(2^A);
ww = 0:pi/500:pi;
n = 0:L-1;

h = (0.54-0.46*cos(2*pi*n/(L-1))).*cos(wc.*(n-(L-1)/2));
B = max(abs(h));
h = h/B;
hn = freqz(h,1,ww);
hmax = max(abs(hn));
hold on
plot(ww,abs(hn))

bplength = length(find(abs(hn) >= 0.5*hmax));
```

```
bplength = (pi/100)*bplength/2;
end
```

>> pickB(256,3)

ans =

0.8782

>> pickB(128,2)

ans =

0.9673

>> pickB(64,2)

ans =

0.9858

>> pickB(32,1)

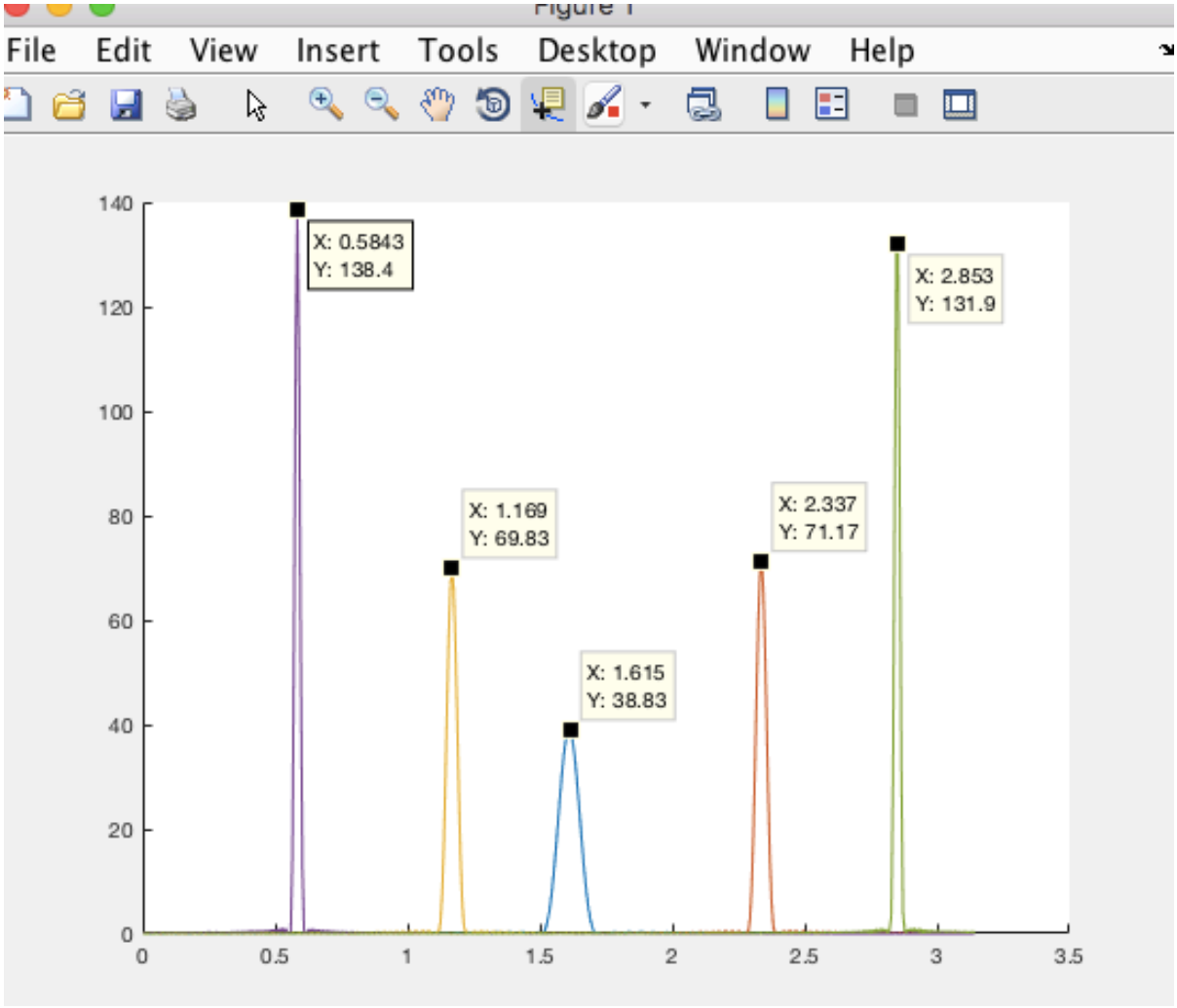
ans =

0.9964

>> pickB(16,1)

ans =

0.9983



2.5 Equalizer

(a) From **x-file.wav**, **use audioread() function to obtain the vector and sampling frequency of your input**. Note: use the [Y, FS]=audioread(FILENAME) iteration of this function.

```
>> [y,fs] = audioread('x-file.wav');
```

(b) The center frequency of each octave (Hz) are tabled from section 2.2. **Use fs obtained in part a to re-calculate the normalized center frequencies. Submit a table similar to section 2.2 with this sampling frequency.** With the same code you used in previous section to generate the five octave filters, **modify your filters to satisfies these filter lengths using a loop:**

Octave	#2	#3	#4	#5	#6
Filter length	256	128	64	32	16

(c) Since different filter lengths would results in different output length, and as we know, matrices of different sizes cannot be summed. Hence, **modify your script so that all five filters have the same length as the longest filter. Put the relevant filter coefficients exactly in the middle of the filter and pad empty indexes on both sides with zeros.**

(d) **Turn your script into a function that takes in the input vector xx, scale vector eq, sampling frequency fs and return output vector yy.** Your function must:

. Create a vector of center frequency based on the inputted sampling frequency fs.

. Create five octave filters of the same length as the longest filter.

. Scale each filter with a factor of $10^{(eq(i)/20)}$ where eq is the scale vector.

. Sum up all five filter coefficients.

. Use convolution to obtain yy. (e) Test your function with input vector and sampling frequency obtained in part a. **Use scale vector eq = [20, 50, 70, 0, 0]. Plot magnitude response of your overall filter from -pi to pi. Play the output vector yy to a TA for check-off.**

```
function [ sc ] = octave( y, h, fs )
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

yy = firfilt(h,y);
sc = yy(1:0.05:length(yy));

for i = 1:length(sc)
    if sc(i) >= 0.5*max(sc)
        sc(i) = 1;
    else
        sc(i) = 0;
    end
end
end
```

```
function [ B ] = pickB( wc, octave, y )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
A = octave - 2;
fs = 8000;
L = 256/(2^A);
ww = 0:pi/500:pi;
n = 0:L-1;

h = (0.54-0.46*cos(2*pi*n/(L-1))).*cos(wc.*(n-(L-1)/2));
B = max(abs(h));
h = h/B;
hn = freqz(h,1,ww);
hmax = max(abs(hn));
%hold on
%zoom on
%plot(ww,abs(hn))

%bplength = length(find(abs(hn) >= 0.5*hmax));
%bplength = (pi/100)*bplength/2;

yy = firfilt(h, y);
score = octave(y, h, fs);
subplot(5,1,A+1)
plot(1:length(yy),score)
end
```