Lab 3: Introduction to Frequency Response

Part 1

1.1 Frequency Response of the Four-Point Averager

Before beginning your work on this section, make sure that you understand the concepts of Frequency Response.

1.

Question 1: Using the Euler Formulas, *show by hand* that the frequency response for the 4-point running average operator is given by

$$H\left(e^{j\hat{\omega}}\right) = \frac{2cos\ (0.5\hat{w}) + 2cos\ (1.5\hat{w})}{4} e^{-j1.5\hat{w}}$$

2.

Implement this equation directly in code and plot the magnitude and phase response of this filter. For $\hat{\omega}$, use a vector that includes 400 samples between -π and π. Since the frequency response is a complex-valued quantity, use abs() and angle() to extract the magnitude and phase of the frequency response for plotting.

3.

The function freqz() evaluates the frequency response for all frequencies in the vector ww by using the summation formula instead of the formula from the previous part. *Use freqz() to plot the magnitude and phase of* $H\left(e^{j\hat{\omega}}\right)$ *versus* $\hat{\omega}$*.* The plot from this section should look the same as the ones from the previous section. Note that this plot might be from 0 to π only. However, for real signals the frequency response is conjugate symmetric so it is mirrored across the y-axis.

Hint: the filter coefficient vector for the 4-point averager can be defined by b = ones(1,4) / 4

1.2 MATLAB find() function

As part of signal processing functions, we often need to perform logical tests, such as with if statements. find() function allows logical tests to be done in a vectorizable form. It returns the indices of the elements that are non-zero. For example, in the following code, find() returns the indices of all the numbers that is greater than 3 in array x :

```
x = 1.4:0.33:5;
index = find(x > 3);
x(index)
```

So, the x(index) command prints the elements of the x with indices found by the find() function. Note that the argument of find() can be any logical condition. find() returns a list of indices where such logical condition is true.

Suppose that you have a frequency response:

```
b = ones(1,4) / 4;
w = -pi:pi/500:pi;
H = freqz(b,1,w);
```

Using find() , *display the list of frequencies where H is approximately zero.* Note that since there might be round-off error in calculating H , the logical test should probably be a test for those indices where the magnitude (absolute value) of H is less than some small number, e.g., 0.001.

Question 2: Does this match the frequency response that you plotted for the 4-point average?

1.3 Nulling Filter

This lab introduces two practical filters: bandpass filters and nulling filters. Bandpass filters can be used to detect and extract information from sinusoidal signals, e.g., tones in a touch-tone telephone dialer. Nulling filters can be used to remove sinusoidal interference, e.g., jamming signals in a radar. You should learn how to characterize a filter by knowing how it reacts to different frequency components in the input.

In this country, electrical power lines operate at 60Hz. Most of it can be rectified. But in many cases, we wish to null this frequency. Nulling filters completely eliminate some frequency components. The simplest possible general nulling filter can have as few as three coefficients. If is the desired nulling frequency, then the following length-3 FIR filter

$$y[n] = x[n] - 2cos\ (\hat{\omega}n)x[n-1]\ + x[n-2]$$

will have a zero in its frequency response at $\hat{w} = \hat{\omega}_n$. For example, a filter designed to completely eliminate signals of the form $Ae^{j0.5\pi}$ would have the following coefficients because we would pick the desired nulling frequency to be $\widehat{\omega}_n = 0.5\pi$. We would pick our coefficients to be:

$$b_0 = 1$$

$$b_1 = -2cos\ (0.5\pi)\ = 0$$

$$b_2 = 1$$

1.

Design a filtering system that consists of the cascade of two FIR nulling filters that will eliminate the following input frequencies: $\widehat{\omega}_n = 0.44\pi$, and $\widehat{\omega}_n = 0.7\pi$. For this part, derive the filter coefficients of both nulling filters. *Submit necessary code, and plots of the magnitude and phase responses of both filters, along with the cascaded system.*

2.

Generate an input signal that is the sum of three sinusoids:

$$x[n] = 5cos\ (0.3\pi n)\ + 22cos\ \left(0.44\pi n - \tfrac{\pi}{3}\right)\ + 22cos\ \left(0.7\pi n - \tfrac{\pi}{4}\right)$$

Make the input signal 150 samples long over the range $0 \leq\ 1 \leq 149$. *Submit your code to generate this signal.*

3.

Filter the sum-of-three-sinusoids signal x[n] through the filters designed previously. *Submit the code that you wrote to implement the cascade of two FIR filters and the command to filter the signal.*

4.

Make a plot of the first 50 points of the output signal. *Determine (by hand)* the exact mathematical formula (magnitude, phase and frequency) for the output signal for $n \geq 5$.

5.

Plot the expected output of the cascaded filter and compare it to the plot obtained in 1.3.4 to show that it matches the filter output over the range $5 \leq n\ \leq 50$.

6.

Question 3: Explain why the output signal is different for the first few points. How many "start-up" points are found? How is this number related to the lengths of the filters designed previously?

Hint: consider the length of a single FIR filter that is equivalent to the cascade of two length-3 FIRs.

2 Lab Part 2

2.1 Simple Bandpass Filter

The L-point averaging filter covered earlier is a lowpass filter centered around zero with passband width inversely proportional to L. It is also possible to create a filter whose passband is centered on some frequency other than zero. One simple way to do this is to define the impulse response of an L-point FIR as:

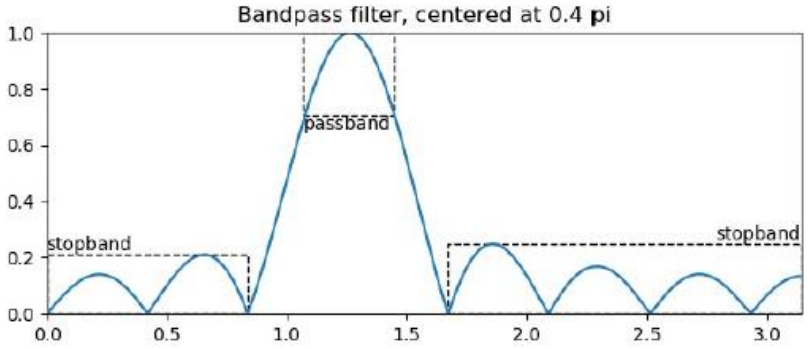
$$h[n] = \frac{2}{L} cos\ (\widehat{\omega}_c n)$$

$$0 \leq n < L$$

Where L is the filter length, and $\widehat{\omega}_c$ is the center frequency that defines the frequency location of the passband. For example, we would pick $\widehat{\omega}_c = 0.44\pi$ if we want the peak of the filter's passband to be centered at 0.44π (therefore allowing signals of frequency 0.44π to "pass through" the filter).

1.

Generate a bandpass filter that will pass a frequency component at $\widehat{\omega} = 0.44\pi$, 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: $\widehat{\omega} = 0.3\pi$, $\widehat{\omega} = 0.44\pi$ and $\widehat{\omega} = 0.7\pi$.



The passband of the BPF filter is defined by the region of the frequency response where $|H(e^{j\widehat{\omega}})|$ is close to its maximum value. If we define the maximum to be H_{max}, then the passband width can be defined as the length of the frequency region where the ratio $\frac{|H(e^{j\widehat{\omega}})|}{H_{max}}$ is greater than $\frac{1}{\sqrt{2}} = 0.707$, or 3dB (decibels) below 0dB. You can use find() function to locate those frequencies where the magnitude satisfies $|H(e^{j\widehat{\omega}})| \geq 0.707 H_{max}$

The stopband of the BPF filter is defined by the region of the frequency response where $|H(e^{j\widehat{\omega}})|$ is close to zero. In this case, we will define the stopband as the region where $|H(e^{j\widehat{\omega}})|$ is less than 25% of the maximum, $|H(e^{j\widehat{\omega}})| \leq 0.25 H_{max}$

2.

Question 1: 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).

3.

Question 2: 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 $\widehat{\omega} = 0.44\pi$, while reducing or rejecting the others at $\widehat{\omega} = 0.3\pi$ and $\widehat{\omega} = 0.7\pi$.

4.

Generate a bandpass filter that will pass the frequency component at $\widehat{\omega} = 0.44\pi$, but now make the filter length (L) long enough so that it will also greatly reduce frequency components at (or near) $\widehat{\omega} = 0.3\pi$ and $\widehat{\omega} = 0.7\pi$. *Submit code and frequency response plot.*

Question 3: Determine the smallest value of L so that the following conditions both hold: - Any frequency component satisfying $|\widehat{\omega}| \leq 0.3\pi$ will be reduced by a factor of 10 or more. - Any frequency component satisfying $0.7\pi \leq |\hat{\omega}| \leq \pi$ will be reduced by a factor of 10 or more.

5.

Question 4: Use the filter from previous part to filter the "sum of 3 sinusoids" signal from Section 1.3.3. *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.

6.

Question 5: Using the frequency response for the filter from part 2.1.4, *explain* how $H(e^{j\hat{\omega}})$ can be used to determine the relative size of each sinusoidal component in the output signal. In other words, write a sentence or two that connect a mathematical description of the output signal to the values that can be obtained from the frequency response plot.

2.2 A Better BPF

It is possible to get better performance in the frequency response by modifying the filter coefficients slightly. One easy way is to use a "Hamming Window." In this case, the impulse response for a length-L bandpass filter would be given as:

$$h[n] = \left(0.54 - 0.46\cos cos\ \left(\frac{2\pi n}{L-1}\right)\right)\cos cos\ \left(\hat{\omega}\left(n - \frac{L-1}{2}\right)\right)$$

$$n = 0, 1, 2, \ldots, L-1$$

Where $\widehat{\omega}$ is the desired center frequency for the BPF. As before, the filter length L determines the passband width, although the Hamming BPF tends to be a little wider than the BPF in the previous section. The big advantage of the Hamming BPF is that its stopband has ripples that are very small (usually less than 0.01). If you are curious, use zooming to figure out the height of the ripples in the stopband when you plot the frequency response.

1.

Generate a Hamming bandpass filter that will pass a frequency component at $\widehat{\omega} = 0.2\pi$. 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: $\hat{\omega} = 0.1\pi, 0.25\pi, 0.4\pi, 0.5\pi, 0.75\pi$. *Summarize* the values in a table.

Hint: use freqz() to calculate these values, or the find() function to extract this information from the vector that produce the plot.

2.

The passband of the BPF filter is defined by the region of the frequency response where $|H(e^{j\hat{\omega}})|$ is close to its maximum value of one. In this case, the passband width is defined as the length of the frequency region where $|H(e^{j\hat{\omega}})|$ is greater than 50% of the peak magnitude value. The stopband of the BPF filter is defined by the region of the frequency response where $|H(e^{j\hat{\omega}})|$ is close to zero.

Use the plot of the frequency response for the length-41 bandpass filter from the previous section, and *determine the passband width* using the 50% level to define the pass band. *Make two other plots* of BPFs for L = 21 and L = 81, and *measure the passband width in both*.

Question 6: Explain how the width of the passband is related to filter length L, i.e., what happens when L is (approximately) doubled or halved.

3.

If the input signal to the length-41 FIR BPF is:

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

Question 7: Determine (by hand) the formula for the output signal.

Hint: use the magnitude and phase measurement from the previous section.

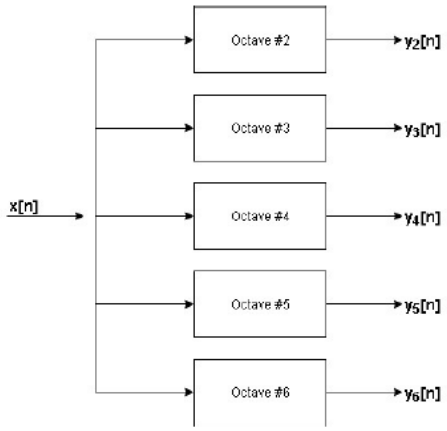
Comment on the relative amplitudes of the three signal components in the output signal, observing whether or not they were in the passband or topband of the filter.

4.

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 $\widehat{\omega} = \pm 0.25\pi$, while reducing or rejecting others.

2.3 Octave Filtering

It is possible to decode piano signals or music in general into octaves using a simple FIR filter bank. The filter bank below consists of five bandpass filters which correspond to octave 2 to 6. The input signal for all the filters is the same.



The octave filtering system needs a set of bandpass filters (BPFs) to isolate individual frequency bands (i.e., the octaves). To make the whole system work, you will have to find out the parameters for the bandpass filters.

The bandpass filter specs are determined by the frequencies of different octaves on the piano. The standard notation is to start an octave at the "C" key. For example, octave #4 starts at "middle-C" which is key #40 and extends up to key #51; the fifth octave starts at key #52 and extends to key #63, and so on.

In order to design the BPFs, we need the upper and lower frequencies for each octave in $\widehat{\omega}$. We can convert key number to frequency in hertz, and then use the sampling frequency to convert analog frequency to $\widehat{\omega}$. For this lab, assume that the sampling frequency is $f_s = 8000Hz$.

Question 8: Below, we provide you with the lower edge, higher edge, and center frequencies of all octave (in key number and Hz). Based on this, *calculate the same parameters in terms of normalized radian frequency , 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	1551

2.4 Bandpass Filter Bank Design

The FIR filters that will be used in the filter bank should be constructed according to the Hamming impulse responses described in Section 2.1 (A Better BPF). These "Hamming" BPFs require two parameters: the length L and the center frequency $\hat{\omega}$. Use the previous table of piano octave frequencies to define the passbands. Furthermore, the filters should be scaled so that their maximum magnitude at the center frequency of the passband is equal to one. This can be done by introducing a third parameter β that will scale all the filter coefficients.

$$h[n] = \beta\left(0.54 - 0.46\cos cos\ \left(\frac{2\pi n}{L-1}\right)\right)\cos cos\ \left(\hat{\omega}\left(n - \frac{L-1}{2}\right)\right)$$

The constant β gives flexibility for scaling the filter's gain to meet a constraint such as making the maximum value of the frequency response equal to one. The bandwidth of the bandpass filter is controlled by L; the larger the value of L, the narrower the bandwidth.

1.

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 code* that will do this scaling operation in general. We will use the following approach:

- Numerical: use max() function (see help max) to measure the peak value of the unscaled frequency response, and then have your code compute to scale the peak to be one.

2.

You must design five separate bandpass filters for the filter bank system. Each filter has a different length and a different center frequency. The lengths have to be different because the bandwidths of the octaves are different. In fact, the bandwidth of each octave is twice that of the next lower octave.

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.

3.

Generate the five (scaled) bandpass filters. *Plot* the magnitude of the frequency responses all together on one plot (the range $0 \le \hat{\omega} \le \pi$ 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.

As help for the previous parts, here are some comments: The passband of the BPF filter is defined by the region of $\hat{\omega}$ where $|H(e^{j\hat{\omega}})|$ is close to one. In this Lab, the passband width is defined as the length of the frequency region where $|H(e^{j\hat{\omega}})|$ is greater than 0.5 of its maximum value.

Filter Design Specifications: For each octave, choose so that the entire octave of frequencies lies within the passband of the BPF.

The stopband of the BPF filter is defined by the region of $\hat{\omega}$ where $|H(e^{j\hat{\omega}})|$ is close to zero. In this case, we can define the stopband as the region where $|H(e^{j\hat{\omega}})|$ is less than 0.01 of its maximum value, because the Hamming filters have excellent stopbands. Notice, however, that the stopbands do not eliminate all the other octaves because the stopband does not start where the passband ends. There is a small "transition region" between the pass and stop bands where the frequency response is small, but not as small as in the stopband.

4.

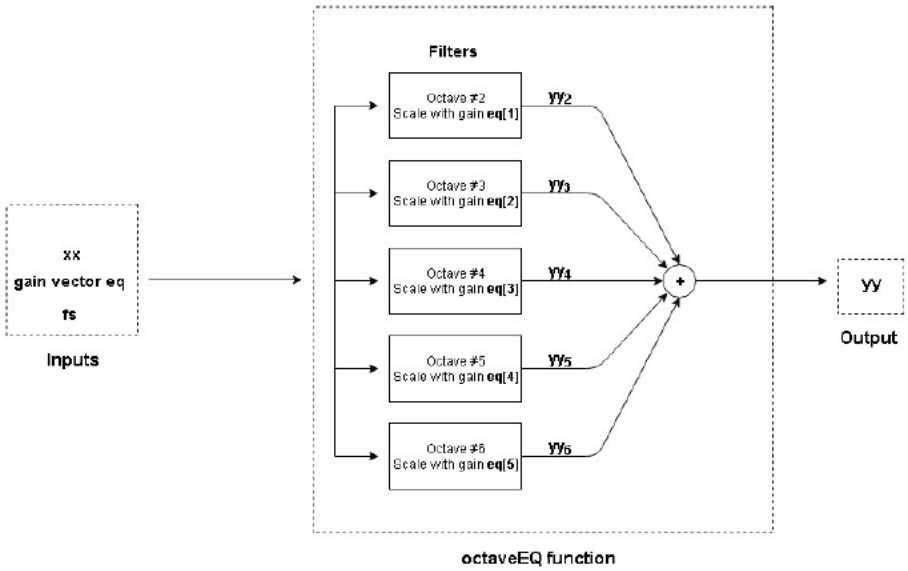
Question 9: Comment on the selectivity of the bandpass filters, i.e., use the frequency response to *explain* how the filter passes one octave while rejecting the others. *Are the filter's passbands narrow enough so that only one octave lies in the passband and the others are in the stopband?*

2.5 Equalizer

An interesting application of octave filtering is making an equalizer. In music, songs can have many notes that spans many octaves. What if we want to amplify bass notes without affecting notes from other octaves? What if we want to attenuate high notes while amplifying bass notes? In these situations, equalizer becomes extremely helpful.

The equalizer takes in the song, then uses octave filter banks to extract notes from the same octave. Once extracted, notes from the same octave will be scaled with desired gain factor. Finally, the song will be reconstructed by summing all the notes back together. Using this method, we can modify music to our preference.

In this portion of the lab, you will make an equalizer consists of 5 octave filters that have been generated in previous section (octave #2 to octave #6). The goal of this section is to create an equalizer function called octaveEQ that takes in song vector xx , gain vector eq , and sampling frequency fs then output new song vector yy with modified gain for notes in each octave. Here's the flowchart:



1.

From x-file.wav , use audioread() function (see help audioread) to obtain *the vector and sampling frequency of the input.*

2.

The center frequency of each octave (Hz) are tabled from section 2.2. *Use fs obtained in previous section 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* (**use a loop**):

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

3.

Since different filter lengths would results in different output length, 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.

4.

Turn your script into a function that takes in the input vector xx , scale vector eq , sampling frequency fs and return output vector y. 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 y .

5.

Test your function with input vector and sampling frequency obtained in the previous section. *Use scale vector eq = [20, 50, 70, 0, 0] . Plot magnitude response of your overall filter from -π to π .*

Play the output vector y to a TA for check-off. If you can't finish in time, submit a .wav file on Canvas.