GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 2000
Lab #8: Octave Band Filtering

Date: 17–30 Oct 2000

******* Lab #8 will be graded out of 150 points. *******

This is *the official* Lab #8 description.

The Warm-up section of each lab must be completed in Lab and the steps marked *Instructor Verification* must also be signed off **during the lab time**.

The final lab report for this lab will be **FORMAL:** discuss your results from sections 4 and 5. In addition, be prepared to run your code in lab for testing against an unknown input.

The formal report will be **due during the week of 31 Oct.–6 Nov. at the start of your lab**.

---

1 Introduction

This lab introduces a practical application where we attempt to extract information from sinusoidal signals—in this case, piano notes. Bandpass FIR filters can be used to extract the information encoded in the waveforms. The goal of this lab is to design and implement several bandpass FIR filters in MATLAB, and use the filtered outputs to determine automatically which note is being played. However, since there are 88 keys on the piano, we will only require the system to figure out which octave the note is in, not the exact note. In the experiments of this lab, you will use `firfilt()`, or `conv()`, to implement filters and `freqz()` to obtain the filter's frequency response.[1] As a result, you should learn how to characterize a filter by knowing how it reacts to different frequency components in the input.

1.1 Frequency Response of FIR Filters

The output or *response* of a filter for a complex sinusoid input, $e^{j\hat{\omega}n}$, depends on the frequency, $\hat{\omega}$. Often a filter is described solely by how it affects different frequencies—this is called the *frequency response*. The frequency response of a general FIR linear time-invariant system is[2]

$$H(e^{j\hat{\omega}}) = \mathcal{H}(\hat{\omega}) = \sum_{k=0}^{M} b_k e^{-j\hat{\omega}k} \tag{1}$$

MATLAB has a built-in function for computing the frequency response of a discrete-time LTI system. The following MATLAB statements show how to use `freqz` to compute and plot the magnitude (absolute value) of the frequency response of an L-point averaging system[3] as a function of $\hat{\omega}$ in the range $-\pi \leq \hat{\omega} \leq \pi$:

[1] If you are working at home and do not have the function `freqz.m`, there is a substitute available called `freekz.m`. You can get it from the ECE-2025 WebCT page.

[2] The notation $H(e^{j\hat{\omega}})$ is used in place of $\mathcal{H}(\hat{\omega})$ for the frequency response because we will eventually connect this notation with the z-transform, $H(z)$, in Chapter 7.

[3] The filter length L is equal to $M + 1$.

```
bb = ones(1,L)/L;         %-- Filter Coefficients
ww = -pi:(pi/100):pi;     %-- omega hat frequency axis
HH = freqz(bb, 1, ww);     %<--freekz.m is an alternative
subplot(2,1,1);
plot(ww, abs(HH))
subplot(2,1,2);
plot(ww, angle(HH))
xlabel('Normalized Radian Frequency')
```

We will always use capital `HH` for the frequency response. For FIR filters, the second argument of `freqz( _, 1, _ )` must always be equal to `1`. The frequency vector `ww` should cover the interval $-\pi \leq \hat{\omega} \leq \pi$ for $\hat{\omega}$, and its spacing must be fine enough to give a smooth curve for $H(e^{j\hat{\omega}})$.

2 Background

2.1 Piano Notes

A piano keyboard consists of 88 keys grouped into octaves. Each octave contains 12 notes, the notes in one octave being twice the frequency of the notes in the next lower octave.

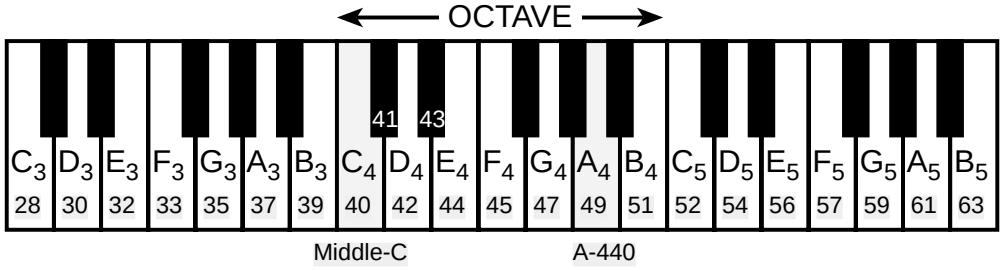

Figure 1: Layout of a piano keyboard. Key numbers are shaded. The notation C_4 means the C-key in the fourth octave.

For example, the reference note is the A above middle-C which is usually called A-440 (or A_4) because its frequency is 440 Hz. Each octave contains 12 notes (5 black keys and 7 white) and the ratio between the frequencies of the notes is constant between successive notes. As a result, this ratio must be $2^{1/12}$. Since middle C is 9 keys below A-440, its frequency is approximately 261 Hz. Consult chapter 9 for even more details.

If we want to produce a system capable of writing music directly from a recorded signal $x(t)$, we need to analyze the frequency content of the signal. One way to do this analysis is to use a set of bandpass FIR filters, each one having its passband designed for one note on the keyboard. This would require a very large number of filters. Another way to do the analysis would be to use a two stage approach. First, we would use a set of bandpass FIR filters where each passband would pass the frequencies in one octave. Then these "octave filters" would be followed by more precise bandpass filters (BPFs) that would determine which key inside the octave is being played. The work in this lab will be to produce a working set of "octave filters."

3 Warm-up: Bandpass Filtering

3.1 Create a Bandpass Filter (BPF)

There are many ways to get the filter coefficients for a bandpass filter. One easy choice is to define the impulse response of an L-point FIR filter as:

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

where L is the filter length, and $\hat{\omega}_c$ is the center frequency that defines the frequency location of the passband. For example, we would pick $\hat{\omega}_c = 0.2\pi$ if we want the peak of the filter's passband to be centered at 0.2π. The bandwidth of the bandpass filter is controlled by L; the larger the value of L, the narrower the bandwidth. This particular filter is also discussed in the section on useful filters in Chapter 7.

(a) Generate the impulse response of a length-25 bandpass filter with $\hat{\omega}_c = 0.2\pi$, and plot $h[n]$ with a `stem` plot.

(b) Compute the frequency response of the length-25 BPF from the previous part and plot its magnitude and phase versus $\hat{\omega}$ over the range $-2\pi \le \hat{\omega} \le 2\pi$.

(c) Use the magnitude response plot to describe the passband of the BPF. For example, measure the bandwidth at a convenient point such as 50% of the peak.

Instructor Verification (separate page)

3.2 Overlay Plotting

Sometimes it is convenient to overlay information onto an existing MATLAB plot. The MATLAB command `hold on` will inhibit the figure erase that is usually done just before a new plot. Demonstrate that you can do an overlay by following these instructions:

(a) Plot the magnitude response of the 25-point BPF filter defined in (2). For this plot, it is sufficient to use a horizontal frequency axis that extends from $-\pi$ to $+\pi$.

(b) Use the `stem` function to place vertical markers at several points on the frequency response $\{0, \pm 0.2\pi, \pm 0.5\pi\}$.

```
hold on, stem(pi*[-0.5,-0.2,0,0.2,0.5],0.9*ones(1,5),'r.'), hold off
```

3.3 Signal Concatenation

In a previous lab, a very long music signal was created by joining together many sinusoids. When two signals are played one after the other, the composite signal is created by the operation of *concatenation.* In MATLAB, this can be done by making each signal a row vector, and then using the matrix building notation as follows:

```
xx = [ xx, xxnew ];
```

where `xxnew` is the sub-signal being appended. The length of the new signal is equal to the sum of the lengths of the two signals `xx` and `xxnew`. A third signal could be added later on by concatenating it to `xx`.

3.3.1 Comment on Efficiency

In MATLAB the concatenation method, `xx = [ xx, xxnew ]`, would append the signal vector `xxnew` to the existing signal `xx`. However, this becomes an *inefficient* procedure if the signal length gets to be very large. The reason is that MATLAB must re-allocate the memory space for `xx` every time a new sub-signal is appended via concatenation. If the length `xx` were being extended from 400,000 to 401,000, then a clean section of memory consisting of 401,000 elements would have to be allocated followed by a copy of the existing 400,000 signal elements and finally the append would be done. This is very inefficient for long signals, but would not be noticed for short signals.

An alternative is to pre-allocate storage for the complete signal vector, but this can only be done if the final length is known ahead of time.

3.4 Filtering a Signal

Use the filter from a previous section (Sec. 3.1) to process an input signal composed of several sinusoids:

$$x[n] = \begin{cases} \cos(0.5\pi n) & 0 \le n < 200 \\ 2 & 200 \le n < 400 \\ 0.5\cos(0.2\pi n) & 400 \le n < 600 \end{cases}$$

(a) Generate $x[n]$ in a vector called `xx`.

(b) Filter `xx` to obtain `yy`.

(c) Make `stem` plots of `xx` and `yy`.

(d) Use the frequency response to validate that the output signal has the correct magnitude and phase in each of the three regions where the input has different frequencies.

(e) Comment: observe the *transient* effect at the transitions ($n = 200$ and $n = 400$). This is due to the start-up of the FIR filter as it encounters a new sinusoid. How long does the transient last (in samples)?

Instructor Verification (separate page)

4 Bandpass Filter Design

Section 4.1 should not be included in the lab report. It is provided as an extension of the Warm-up. *Section 4.2 must be included.*

4.1 Simple Bandpass Filter Design

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

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

where L is the filter length, and $\hat{\omega}_c$ is the center frequency that defines the frequency location of the passband.

(a) Generate a bandpass filter that will pass a frequency component at $\hat{\omega} = 0.4\pi$. Make the filter length (L) equal to 40. Make a plot of the frequency response magnitude and phase.

(b) The *passband* of the BPF filter is defined by the region of the frequency response where $|\mathcal{H}(\hat{\omega})|$ is close to its maximum value of one. Typically, the passband width is defined as the length of the frequency region where $|\mathcal{H}(\hat{\omega})|$ is greater than some level such as 0.5, or $0.707 = 1/\sqrt{2}$. Note: you can use MATLAB's `find` function to locate those frequencies where the magnitude satisfies $|\mathcal{H}(\hat{\omega})| \geq 0.5$ (similar to Fig. 2). Use the plot of the frequency response for the length-40 bandpass filter from part (a), and determine the passband width using the 0.5 level to define the pass band.

(c) Make two other plots of BPFs for $L = 20$ and $L = 80$ with the same $\hat{\omega}_c$; and measure the passband width for both. Then explain how the width of the passband is related to filter length L, i.e., what happens when L is doubled or halved.

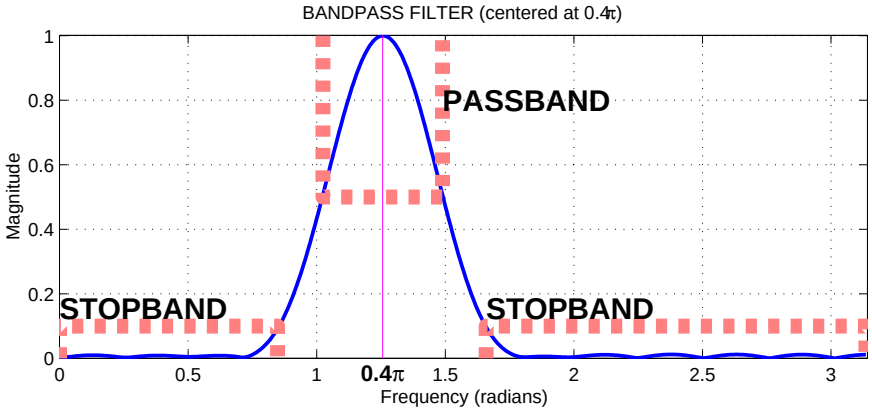

Figure 2: The frequency response of an FIR bandpass is shown with its passband and stopband regions. In this case, the passband is defined to be the region where $|H(e^{j\hat{\omega}})|$ is greater than 0.5; and the stopband is the region where the magnitude is less than 0.1.

4.2 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:

$$h[n] = (0.54 - 0.46\cos(2\pi n/(L-1)))\cos(\hat{\omega}_c(n - (L-1)/2)) \qquad \text{for } n = 0, 1, 2, \ldots L-1 \qquad (3)$$

where $\hat{\omega}_c$ is the desired center frequency for the BPF. The first term is the Hamming window. 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). Use zooming to figure out the height of the ripples in the stopband when you plot the frequency response (below).

(a) Generate a Hamming bandpass filter that will pass a frequency component at $\hat{\omega} = 0.25\pi$. Make the filter length (L) equal to 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: $\hat{\omega} = \{0,\ 0.1\pi,\ 0.25\pi,\ 0.4\pi,\ 0.5\pi,\ 0.75\pi\ \}$. Summarize the values in a table.
Hint: use MATLAB's `freqz()` function to calculate these values, or the `find()` function to extract this information from the vector that produce the plot.

(b) The *passband* of the BPF filter is defined by the region of the frequency response where $|\mathcal{H}(\hat{\omega})|$ is close to its maximum value of one. In this case, the passband width is defined as the length of the frequency region where $|\mathcal{H}(\hat{\omega})|$ 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 $|\mathcal{H}(\hat{\omega})|$ is close to zero.

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 $L = 21$ and $L = 81$, 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.

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

$$x[n] = 2 + 2\cos(0.1\pi n + \pi/3) + \cos(0.25\pi n - \pi/3)$$

Determine (by hand) the formula for the output signal. (Hint: use the magnitude and phase measurements from part (a).) Comment on the relative amplitudes of the three signal components in the output signal, observing whether or not they were in the passband or stopband of the filter.

(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.

5 Project: Piano Note Decoding

It is possible to decode piano signals into octaves using a simple FIR filter bank. The filter bank in Fig. 3 consists of seven bandpass filters which each pass only one of the seven possible octaves. The input signal for all the filters is the same.

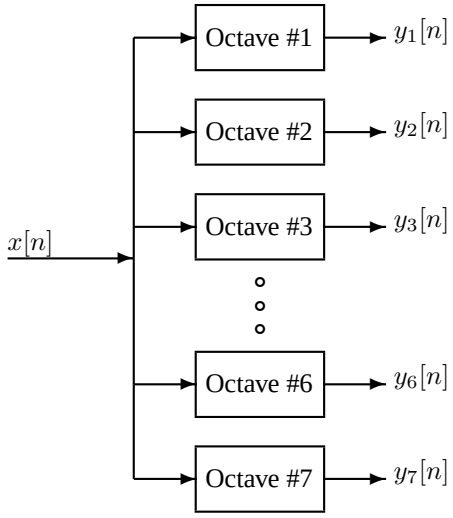

Figure 3: Filter bank consisting of bandpass filters which pass frequencies corresponding to the seven octaves on a piano. (Each octave contains 12 keys, but we will ignore the four extra keys outside the seven complete octaves.)

The octave filtering system needs two pieces: a set of bandpass filters (BPFs) to isolate individual frequency bands (i.e., the octaves), and a detector to determine whether or not a given octave is active. The detector must "score" each BPF output and determine whether or not the octave contains one or more notes.

In a practical system where noise and interference are also present, this scoring process is a crucial part of the system design, but we will only work with noise-free signals to illustrate the basic functionality of the system.

To make the whole system work, you will have to write at least three function M-files:

1. A filter design function to produce the filter coefficients for the BPFs in Fig. 3.
2. A filtering function that will filter the input signal through each channel.
3. A scoring function that will evaluate the output of each channel for "activity."

The following sections discuss how to create or complete these functions.

5.1 Piano Octaves

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. See Fig. 1 for the layout of a piano keyboard.

In this implementation, we want to design filters that will isolate five different octaves: octaves #2 through #6. Octave #2 starts with key #16, octave #3 starts at key #28, and octave #6 starts at key #64 (the last key in octave #6 is key #75).

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

Make a table defining the five BPFs that you must design.

(a) For each octave, determine the lower and upper frequencies which will become the lower and upper edges of the passband of the BPF. Give these numbers in hertz and also in $\hat{\omega}$.

(b) Then compute the center frequencies of the BPFs as the midpoint between the lower and upper band edges of the filter.

Save this information for use in the Filter Design Section (next).

5.2 Bandpass Filter Bank Design

The FIR filters that will be used in the filter bank (Fig. 3) should be constructed according to the Hamming impulse responses of (3). These "Hamming" BPFs require two parameters: the length L and the center frequency $\hat{\omega}_c$. 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.

$$h[n] = \beta\,(0.54 - 0.46\cos(2\pi n/(L-1)))\cos(\hat{\omega}_c(n - (L-1)/2)) \qquad \text{for } n = 0, 1, 2, \ldots L-1 \quad (4)$$

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.

(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. There are two approaches here:

(a) *Mathematical:* derive a formula for β from the formula for the frequency response of the BPF. Then use MATLAB to evaluate this closed-form expression for β. In this case, such analysis is nearly impossible.

(b) *Numerical:* use MATLAB's `max` function to measure the peak value of the unscaled frequency response, and then have MATLAB compute β to scale the peak to be one.

(b) 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 determine in Section 5.1. 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.

(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. Hint: use the `hold` command and markers as you did in the warm-up.

(d) 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.

Filter Design Specifications: For each octave, choose L 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 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.

Use the `zoom on` command to examine the frequency response over the frequency domain where the octaves lie. Comment on the selectivity of the bandpass filters, i.e., use the frequency response (passbands and stopbands) 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?

5.3 Piano Octave Decoding

There are several steps to decoding a piano signal to figure out which note is being played:

1. Filter the individual segments to extract the possible frequency components using carefully designed BPFs.

2. Determine a set of time segments over which to examine the output signals. In general, this could be hard to implement so we will restrict the notes to occur at regular intervals based on the timing of the song.

3. Determine which frequency components are present in each time segment by measuring the size of the output signal from all of the bandpass filters. This tells you which octaves are active.

Here is how the system should work: When the input to the filter bank is a piano signal, the outputs from a few of the bandpass filters (BPFs) should be larger than the rest. If we can detect (or measure) the large outputs, then we know which octaves contain notes. These octaves are then used to label the notes partially.

A good measure of the output levels is the *peak value* of the filter outputs, because when the BPF is working properly it should pass only the sinusoidal signals within one octave and the peak value would represent the combined amplitude of those sinusoids. If there is only one sinusoid within the octave then the peak value is exactly equal to the amplitude of the one sinusoid present.

(a) In order to test your design so far, generate the following signal composed of several sinusoids:

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

Use a sampling rate of $f_s = 8000$ Hz, and generate $x(t)$ in a vector called `xx`.

(b) Filter `xx` through the five filters of the filter bank

(c) Make plots of the five outputs all together in one figure (use `subplot(5,1,n)`).

(d) Use the frequency responses to validate that the output signals have the correct magnitude and phase in each of the three regions where the input has different frequencies. You must determine which octave the signals are in.

(e) Comment: observe the *transient* effect at the transitions. This is due to the start-up of each FIR filter as it encounters a new sinusoid. How long does the transient last (in seconds)? Is it different for each filter in the filter bank?

5.4 Scoring for Activity

The final objective is detecting which octaves contain notes. In order to *automate* this detection process, we need a *score* function that rates each octave for possible activity.

(a) Complete the `octavescore` function based on the skeleton given in Fig. 4. The input signal `xx` to the `octavescore` function must be the input signal containing all the notes. Rather than trying to detect the beginning and end of individual notes, we will have the score function calculate a score every 50 milliseconds.

(b) Use the following rule for scoring: the score for the i-th channel equals $\max_n |y_i[n]|$, where the maximum is taken over a 50 millisecond interval. The signal $y_i[n]$ is the output of the i-th BPF. Since score must be computed every 50 milliseconds, each channel has a score vector with 20 scores per second.

(c) Note: After filtering and prior to scoring, adjust the output signal for the delay through the filter. Since each filter has a different length, the delays are slightly different. For the Hamming BPFs, the delay is $(L-1)/2$. This is a *minor* issue, but it might cause some misalignment of the output scores.

(d) Finally, use all the score vectors together to implement an automatic detection system, based on the fact that all filter passbands are above 50%. If all the input signals have an amplitude of one, then comparing the scores to a threshold of 0.5 would detect the presence of a signal. This is a big assumption on the inputs, but it will be true in the test data (below).

Comparing the scores to a threshold will give an output that is either zero (score less than threshold) or one (score greater than threshold). So you will end up with a $5 \times N$ matrix consisting of ones

```
function  sc = octavescore(xx, hh, fs)
%OCTAVESCORE
%  usage:     sc = octavescore(xx, hh, fs)
%  returns a score based on the max amplitude of the filtered output
%     xx = input signal containing musical notes
%     hh = impulse response of ONE bandpass filter
%     fs = sampling rate
%
%   The signal detection is done by filtering xx with a length-L
%   BPF, hh, and then finding the maximum amplitude of the output
%   within 50 millisecond segments.
%   The score is a vector containing the maximum amplitudes
%     of all the segments.
```

Figure 4: Skeleton of the `octavescore.m` function. Complete this function with additional lines of code.

and zeros, where N is the number of 50-millisecond intervals. Use your experience with plotting the output signals in the previous section to judge how well your automatic system is working, i.e., are the ones in the correct locations and correct octaves.

5.4.1 Testing

Once you get your system working you could test it in three ways (only #1 needs to be included with the lab report):

1. Use the test signal provided with the lab in `lab8test.zip` which contains the MAT file called `lab8test.mat`. It contains a signal sampled at 8000 samples/sec whose duration is 3.77 seconds. The signal has either 0, 1 or 2 sinusoidal notes present at one instant of time. The amplitudes of all sinusoids is one. You should run your system to determine which octaves are active in each 50-millisecond time slice. The MAT file also contains a matrix called `notes` which has the correct notes in the input (as key numbers). The durations vary between 0.16 and 0.32 secs.

2. Generate a music signal that is a progression of notes starting at key #16, ending at key #87, and taking every other note, or every third note. To keep the overall signal reasonably short, make each note about 0.2 secs. long, and make the amplitude of every sinusoid equal to one. This test signal is useful because you know the correct answer.

3. Generate your own musical tones, as you did in lab #4. Make sure to use $f_s = 8000$ Hz. In this case, you will have several notes playing simultaneously, so the system will have a hard time making 100% correct decisions.

When you run your system, there might be errors some of the time, especially when a note frequency is near the boundary between two octaves. Summarize the behaviour of your test cases above, and comment on how well your system works. For example, what is the percentage of errors (i.e., incorrect octave designations)? What happens if two simultaneous notes are in the same octave?

5.4.2 Demonstration

When you turn in your lab report, you must demonstrate to the Lab TA that your system works correctly. We will have a "mystery" signal similar to the signal in `lab8test.zip`.

Lab #8
ECE-2025
Fall-2000
INSTRUCTOR VERIFICATION PAGE

Staple this page to the end of your Lab Report.

Name: _____ Date of Lab: _____

Part 3.1: Create a bandpass filter; plot the magnitude response, and then determine the width of the pass band at 50% of the peak.

Verified:_____ Date/Time:_____

Part 3.4: Filter the three-sinusoid signal through the BPF, and explain the amplitudes of the output signal in the three sections.

Verified:_____ Date/Time:_____