GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 2004
Lab #8: Filter Design: Bandpass Filters

Date: 12–25 Oct 2004

You should read the Pre-Lab section of the lab and do all the exercises in the Pre-Lab section before your assigned lab time. You **MUST** complete the online Pre-Post-Lab exercise on Web-CT at the beginning of your scheduled lab session. You can use MATLAB and also consult your lab report or any notes you might have, but you cannot discuss the exercises with any other students. You will have approximately 20 minutes at the beginning of your lab session to complete the online Pre-Post-Lab exercise. The Pre-Post-Lab exercise for this lab includes some questions about concepts from the previous Lab report as well as questions on the Pre-Lab section of this lab.

The Warm-up section of each lab must be completed **during your assigned Lab time** and the steps marked *Instructor Verification* must also be signed off **during the lab time**. After completing the warm-up section, turn in the verification sheet to your TA.

Forgeries and plagiarism are a violation of the honor code and will be referred to the Dean of Students for disciplinary action. You are allowed to discuss lab exercises with other students and you are allowed to consult old lab reports, but you cannot give or receive written material or electronic files. Your submitted work should be original and it should be your own work.

It is only necessary to turn in Section 4 as this week's lab report; the lab report format is **Informal**.
The report will **due during the week 26-Oct to 1-Nov (after Fall Break).**

---

1 Introduction

The goal of this lab is to study the response of FIR filters to inputs such as complex exponentials and sinusoids. 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.

2 Pre-Lab

This lab also introduces the process of "filter design" and then uses that skill to design practical filters: ***lowpass, highpass*** and ***bandpass*** filters. Bandpass filters can be used to detect and extract information from sinusoidal signals, e.g., individual notes in a musical passage or tones in a touch-tone telephone dialer.

2.1 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 input frequencies—this is called the *frequency response*. The frequency response of a general FIR linear time-invariant system is

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

[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 find it in the *SP-First Toolbox*, or get it from the ECE-2025 WebCT page.

For the process of filter design, it is important to recognize that the choice of filter coefficients $\{b_k\}$ determines the frequency response. The most useful filter design methods provide a formula for the $\{b_k\}$ coefficients that will give a particular behavior for $H(e^{j\hat{\omega}})$, e.g., the formula in (2).

2.1.1 MATLAB Frequency Response for Lowpass Filter

MATLAB has a built-in function for computing the frequency response of a discrete-time LTI system. When the filter coefficients are given by the formula:

$$h[n] = \frac{\sin(\hat{\omega}_c(n - M/2))}{\pi(n - M/2)} \sin^2(\pi n/M) \qquad \text{for } n = 0, 1, 2, \ldots, M \tag{2}$$

where M is the filter order, which should be an even integer. The second term in $h[n]$ is called a VonHann window; the first one is a "sinc function." The design parameter $\hat{\omega}_c$ is called the ***cutoff frequency*** of the filter because it determines the passband and stopband regions of the frequency response (below). The following MATLAB statements show how to use `freqz` to compute and plot both the magnitude (absolute value) and the phase of the frequency response of the filter in (2) (when $\hat{\omega}_c = 0.3\pi$) as a function of $\hat{\omega}$ in the range $-\pi \leq \hat{\omega} \leq \pi$:

```
M = 50;   nn = 0:M;
VH = (sin(pi*nn/M)).^2;     %-- VonHann window
wc = 0.3*pi;
sincwc = sin(wc*(nn-M/2))./(pi*(nn-M/2));  %-- sinc function
sincwc(M/2 + 1) = wc/pi;        %-- fix divide by zero
bb = sincwc.*VH;                %-- Filter Coefficients
ww = -pi:(pi/400):pi;           %-- omega hat
HH = freqz(bb, 1, ww);          %<--freekz.m is an alternative
subplot(2,1,1);
plot(ww, abs(HH)), grid on
subplot(2,1,2);
plot(ww, angle(HH)), grid on
xlabel('Normalized Radian Frequency')
```

For FIR filters, the second argument of `freqz( _, 1, _ )` must always be equal to `1`.[2] The frequency vector `ww` should cover an interval of length 2π for $\hat{\omega}$, and its spacing must be fine enough to give a smooth curve for $H(e^{j\hat{\omega}})$.

2.2 Passband Defined for the Frequency Response

Certain types of digital filters have a frequency response (magnitude) that is close to one in some frequency regions, and close to zero in others. For example, the plot in Section 2.1.1 is a lowpass filter whose magnitude is close to one when the frequency $\hat{\omega}$ is near zero. This region is called the ***passband*** of the filter. It will be useful to have a precise measurement of the passband width so that we can compare different filters.

(a) From the plot of the magnitude response in Section 2.1.1 determine the set of frequencies where $\left||H(e^{j\hat{\omega}})| - 1\right|$ is less than 0.0075. This should be a region of the form $-\hat{\omega}_p \leq \hat{\omega} \leq \hat{\omega}_p$. Determine $\hat{\omega}_p$ for the case where $M = 50$.

(b) Compare the value of $\hat{\omega}_p$ found in the previous part to the design parameter $\hat{\omega}_c$ in (2).

[2] If the output of the `freqz` function is not assigned, then plots are generated automatically; however, the magnitude is given in decibels which is a logarithmic scale. For linear magnitude plots a separate call to `plot` is necessary.

2.3 Stopband Defined for the Frequency Response

When the frequency response (magnitude) of the digital filter is close to zero, we have the stopband region of the filter. In the lowpass filter example of Section 2.1.1, the magnitude is close to zero when the frequency $\hat{\omega}$ is near π (a high frequency). This region is called the ***stopband*** of the filter. We can make a precise measurement of the stopband edge as follows:

(a) From the plot of the magnitude response in Section 2.1.1 determine the set of frequencies where $|H(e^{j\hat{\omega}})|$ is less than 0.0075. This should be two regions: $\hat{\omega}_s \leq \hat{\omega} \leq \pi$ in positive frequencies, and $-\pi \leq \hat{\omega} \leq -\hat{\omega}_s$ for negative frequencies. Determine $\hat{\omega}_s$ for the case where $M = 50$.

(b) Compare the value of $\hat{\omega}_s$ found in the previous part to the design parameter $\hat{\omega}_c$ in (2).

2.4 Linear Phase in the Frequency Response

The phase of the frequency response can be related to time delay. In the lowpass filter example of Section 2.1.1, the phase plot appears to be jagged, but it is actually linear.

(a) Determine the slope of the linear segments of the frequency response for the $M = 50$ filter in Section 2.1.1. Your answer should be an integer.

(b) Plot the impulse response of the digital filter in Section 2.1.1 for the range $n = 0, 1, \ldots, M$. Then determine the symmetry point for $h[n]$, i.e., find the integer n_s such that $h[n_s + n] = h[n_s - n]$.

(c) Compare the value of the slope (from part (a)) to the "symmetry point" of $h[n]$. Verify that the phase slope is equal to $-n_s$.

2.5 Use the MATLAB FIND Function

in MATLAB the `find` function returns a list of indices where a logical condition is true. See `help` on `relop` for information. For example, if you have the frequency response from Section 2.1.1 you could use the `find` command to determine the indices where `abs(HH)` is close to one, and then use those indices to display the list of frequencies in the passband.

3 Warm-up

The objective of this warm-up is to use the MATLAB GUI **`dltidemo`**[3] A second objective is to demonstrate the frequency responses of lowpass, highpass and bandpass filters designed by the general *VonHann-window* method.

3.1 LTI Frequency Response Demo

The **`dltidemo`** GUI illustrates the "sinusoid-IN gives sinusoid-OUT" property of LTI systems. In this demo, you can change the amplitude, phase and frequency of an input sinusoid, $x[n]$, and you can change the digital filter that processes the signal. Then the GUI will show the output signal, $y[n]$, which is also a sinusoid *(at the same frequency)*. Figure 1 shows the interface for the **`dltidemo`** GUI. It is possible to see the formula for the output signal, if you click on the `Theoretical Answer` button located at the bottom-middle part of the window. The digital filter can be changed by choosing different options in the `Filter Specifications` box in the lower right-hand corner. Right-clicking on the red dots in the frequency response panels (middle) will give the numerical value of the magnitude or phase at that point;

[3]The **`dltidemo`** GUI is part of the *SP-First* toolbox, and is already installed in the ECE lab. The latest versions of all the *SP-First* GUIs can be found at: `http://users.ece.gatech.edu/mcclella/matlabGUIs/index.html`

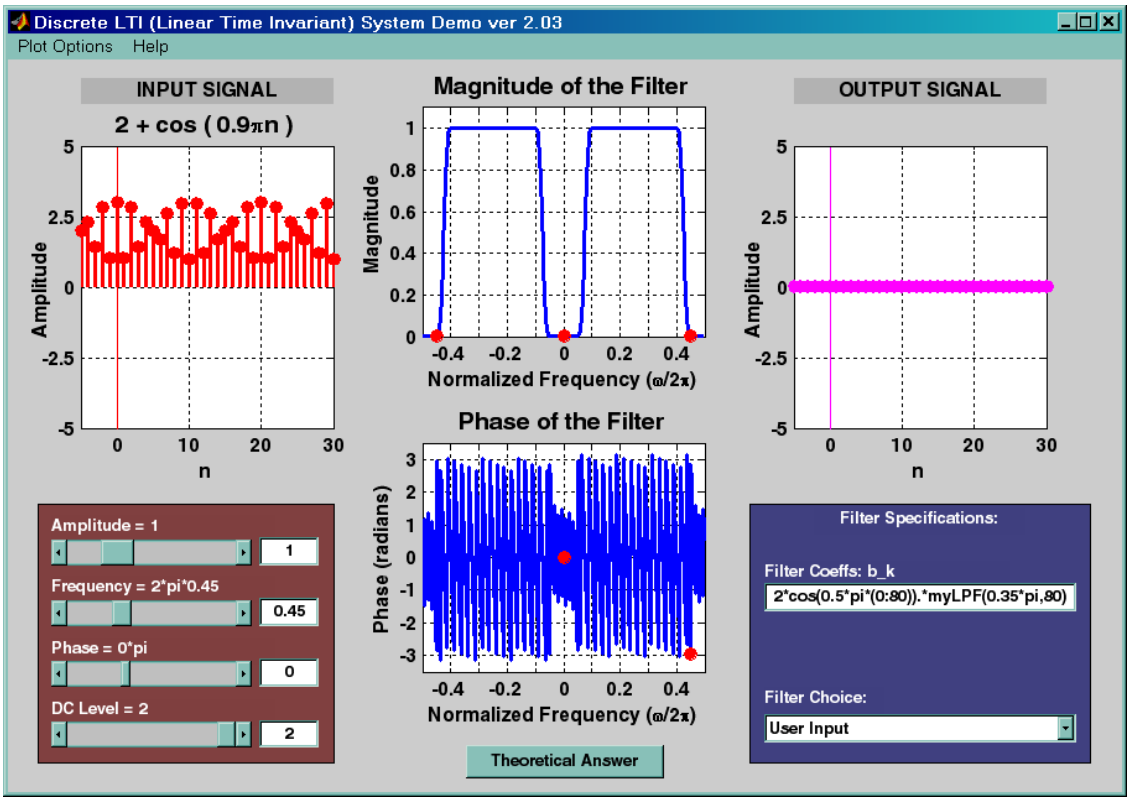

Figure 1: DLTI demo interface. The frequency label is ω because MATLAB won't display $\hat{\omega}$. When the Filter Choice is set to `User Input`, MATLAB code can be entered in the text box for the filter coefficients.

likewise, right-clicking on a signal point in the left or right panels will give the numerical value of the signal at one index.
In the Warm-up, you should perform the following steps with the **`dltidemo`** GUI:

(a) Set the input to $x[n] = 1.5\cos(0.1\pi(n-4))$

(b) Set the digital filter to be a 9-point averager.

(c) Determine the formula for the output signal and write it in the form: $y[n] = A\cos(\hat{\omega}_0(n-n_d))$.

(d) Using n_d for $y[n]$ and the fact that the input signal had a peak at $n = 4$, determine the amount of delay through the filter. In other words, how much has the peak of the cosine wave shifted?

(e) Now, change the length of the averaging filter so that the output will be zero, i.e., $y[n] = 0$. Use the GUI to show that you have the correct filter to zero the output. If the filter length is more than 15, you will have to enter the "Filter Specifications" with the `User Input` option.

(f) When the output is zero, the filter acts as a *Nulling Filter*, because it eliminates the input at $\hat{\omega} = 0.1\pi$. Which other frequencies $\hat{\omega}$ are also nulled? Find at least one.

Instructor Verification (separate page)

3.2 Lowpass, Bandpass and Highpass Filters

The **`dltidemo`** can be used to show different filter types: lowpass filters (LPF), bandpass filters (BPF) and highpass (HPF) filters. You should perform the following steps with the **`dltidemo`** GUI:

(a) Set the input to $x[n] = 2+1\cos(0.9\pi n)$, which is a signal consisting of one low frequency component and one high frequency component.

(b) Set the digital filter to be the filter **Lowpass, L=15**. Observe the output and determine the exact mathematical formula for the output signal.

(c) Set the digital filter to be the filter **Highpass, L=15**. Observe the output and determine the exact mathematical formula for the output signal.

(d) Set the digital filter to be the filter **Bandpass, L=21**. Observe the output and determine the exact mathematical formula for the output signal.

(e) Use the frequency response, $H(e^{j\hat{\omega}})$, shown in the **dltidemo** GUI to explain why the outputs are different in these three cases.

Instructor Verification (separate page)

3.3 MATLAB Function for LPF Design

In this section you must write an M-file that will design a lowpass filter according to the formula given in (2). The M-file should return the filter coefficients given the order (M) and the frequency $\hat{\omega}_c$ as inputs.

(a) Use the following comments as a template for writing the M-file.

```
function  hh = VHLPF( wc, M )
%VHLPF   LPF design function with VonHann window
%
%   wc = design parameter giving the approximate location
%        of the passband and stopband edges
%    M = filter order
%   hh = impulse response of the FIR lowpass filter
```

(b) Demonstrate that your M-file works by using it in the **dltidemo** GUI. There is a `User Input` option as one of the filter choices. You should be able to write a call to `VHLPF` in the text box of the user input. Make a length-37 FIR LPF whose cutoff frequency is approximately 0.37π. Recall that the length of an FIR filter is $L = M + 1$. If you look carefully at $h[n]$ for the "length-37" VonHann LPF, what is the true length of the impulse response?

Instructor Verification (separate page)

4 Lab Exercises

The L-point averaging filter is a lowpass filter. Its passband width is controlled by the filter length L, being inversely proportional to L. In fact, you can use the GUI **`dltidemo`** to view the frequency response for different length averagers and measure the passband widths. The L-point averager has two shortcomings when a high-performance filter is needed: its passband is not flat and its stopband is not very close to zero. In the warm-up section, an alternative method of lowpass filter design was presented in which reasonably good filters can be obtained with relatively flat passbands and stopband that are within 0.75% of their desired values of one or zero.

4.1 Bandpass and Highpass Filters

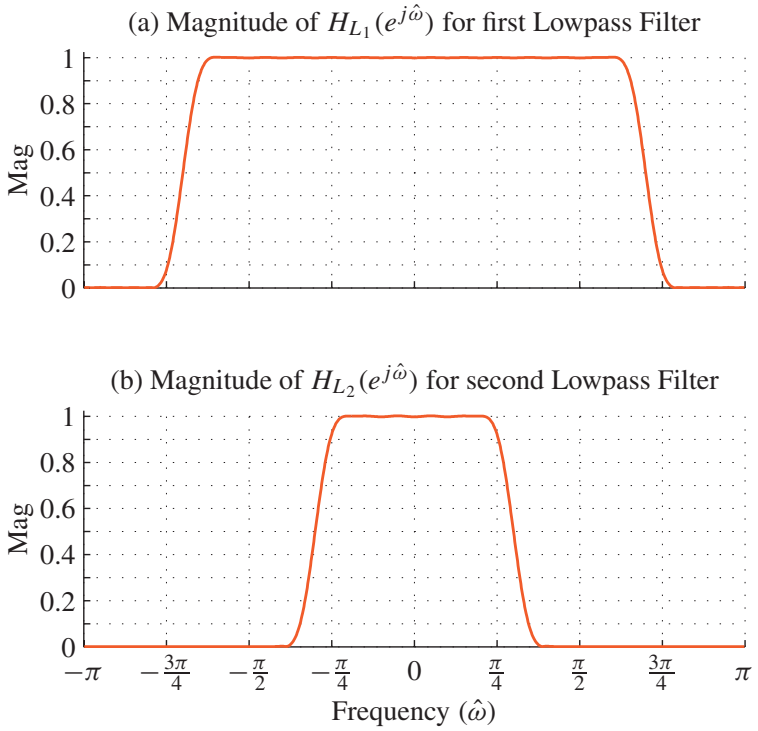

Figure 2: Frequency response magnitudes for (a) wide LPF and (b) narrower LPF. The bandpass filter can be constructed from the difference (approximately): $|H_B(e^{j\omega}) \approx |H_{L_1}(e^{j\omega})| - |H_{L_2}(e^{j\omega})|$.

In this lab exercise, a systematic approach to bandpass and highpass filter design will be implemented based on the LPF design function that was written in the warm-up. For bandpass filters, we base our approach on Fig. 2 which seems to suggest that a BPF is the "difference of two LPFs." On the other hand, a highpass filter can be created by "multiplying the impulse response of an LPF by minus one to the n."

4.1.1 Bandpass Filter Design

For the frequency response of a BPF, we can write

$$H_B(e^{j\omega}) = H_{L_1}(e^{j\omega}) - H_{L_2}(e^{j\omega}) \tag{3}$$

where $H_{L_1}(e^{j\omega})$ and $H_{L_2}(e^{j\omega})$ are LPFs that have exactly the same order.

(a) Write an M-file called `VHBPF(w1,w2,M)` that will create an M^{th} order BPF with cutoff frequencies at $\hat{\omega}_1$ and $\hat{\omega}_2$. The two arguments `w1` and `w2` should be the cutoff frequencies $\hat{\omega}_1$ (the lower cutoff)

and $\hat{\omega}_2$ (the upper cutoff). This M-file should call the previously written M-file, `VHLPF`. Use $\hat{\omega}_1$ and $\hat{\omega}_2$ to determine the cutoff frequencies for two LPFs in Eq. (3).
Note: the "cutoff frequency" is neither the passband edge nor the stopband edge; instead, it is somewhere in between. However, the "cutoff frequency" (along with the filter order M) does control the location of the passband and stopband edges.

(b) Generate a BPF whose passband covers the region from 0.15π to 0.45π; and whose stopbands are the regions $[0, 0.1\pi]$ and $[0.5\pi, \pi]$. As before, the definition of the passband and stopband is that the magnitude response be within 0.75% of the true values of one in the passband and less than 0.0075 in the stopband. This will require some trial and error to figure out the appropriate filter order to meet these specifications.
Note: The filter order M must be an even integer.

(c) Plot the frequency response of the BPF designed in the previous part. On the magnitude plot, mark the exact values for the passband edges and the stopband edges. Rather than plot the phase, determine the slope of the phase, which should be linear (after you ignore the phase jumps).

4.1.2 Highpass Filter Design

For the frequency response of a HPF, we can use the following "multiply by minus one to the n" trick. Thus we can write:

$$h_H[n] = h_L[n](-1)^n \tag{4}$$

(a) Explain why (4) will produce a HPF by giving the mathematical expression for the frequency response of the HPF, $H_H(e^{j\omega})$ in terms of the frequency response of the LPF, $H_L(e^{j\omega})$.
Hint: use the fact that $e^{j\pi n} = (-1)^n$.

(b) Write an M-file called `VHHPF(wc,M)` that will create an M^{th} order HPF with a cutoff frequency $\hat{\omega}_c$. This M-file should call the previously written M-file, `VHLPF`.

(c) Plot the frequency response of a length-51 HPF whose cutoff frequency is $\hat{\omega}_c = 0.4\pi$. Determine the passband edge where the frequency response is within 0.75% of the passband value of one; and the stopband edge where the frequency response (magnitude) is less than 0.0075.

4.2 Filtering a Music Signal

FIR filters can be used to separate out frequency components of a signal. One common method for coding signals is to filter the signal into bands and then code the bands. This approach succeeds in audio coding because the human hearing system is sensitive to frequency. In this section, a music signal will be filtered into separate bands and the energy measured in each band.

(a) For this processing we will use the music signal "Für Elise" created in a previous lab from sinusoids sampled at $f_s = 2756.25$ Hz. We will denote this music signal as $f[n]$. It is a WAV file contained in the zip file `FEout.zip`. Use `wavread` to read `FEout.wav` into MATLAB.

(b) Design three filters: a LPF with a cutoff frequency of 250 Hz; a BPF whose passband extends from a low cutoff frequency of 250 Hz to a high cutoff of 540 Hz; and a HPF whose cutoff frequency is 540 Hz. Make the order (M) of all the filters equal to 200. Use the M-files developed in Sections 4.1.2 and 4.1.1
Note: The passbands of the filters will not be exactly equal to the numbers given, but they will be close.

(c) Filter the music signal $f[n]$ through each of the three filters to produce three outputs: $y_L[n]$, $y_B[n]$, and $y_H[n]$. Listen to the outputs and describe what you hear. Which one contains the most notes?

(d) Form a new signal as the sum of the three filtered signals.

$$z[n] = y_L[n] + y_B[n] + y_H[n]$$

Listen to this signal and compare it to the original music signal $f[n]$. Comment on what you hear. Is it identical to the original?

(e) Make four spectrograms with `plotspec`: the original and each of the filtered signals. Use the spectrograms to explain what you heard in the previous two parts.

(f) Write a MATLAB function that will count the number of notes in one of the output signals. Use the following strategy based on filtering:

i. Square the k^{th} output signal, e.g., $q_k[n] = y_k^2[n]$.

ii. Filter $q_k[n]$ with a 37-point running average.

iii. Compare the averaging filter's output $z_k[n]$ to a threshold of 0.07. After thresholding you end up with a binary signal $b_k[n]$ defined as follows:

$$\begin{aligned} z_k[n] \geq 0.07 &\Longrightarrow \quad b_k[n] = 1 \\ z_k[n] < 0.07 &\Longrightarrow \quad b_k[n] = 0 \end{aligned}$$

iv. Any long run of ones in $b_k[n]$ is a note; and each note should be separated by a short run of zeros. You can use a first-difference filter to detect the beginning and end of a run of ones.

v. The output of the function should be a count, and also the starting index (or time) of each note found.

Hint: It would be informative to make plots of the signals $q_K[n]$, $z_k[n]$, and $b_k[n]$ in order to see how the threshold is used to find the notes.

(g) Demonstrate that your note counting function works by giving the results for the three filtered signals $y_L[n]$, $y_B[n]$, $y_H[n]$, and also for the original $f[n]$.

4.3 Summarize with a Concept Map

For the Summary section of your lab report, draw a concept map that contains at least five concepts that were used during this lab. Since experts use many links between concepts, try to produce a map that has a high "link to node" ratio. Use the Concept Navigation Tool (***CNT***) to produce the map.

Possible concept names might be: Bandpass Filter, Lowpass Filter, Highpass Filter, Unit Step, Unit Impulse, Sinusoid, Phasor Addition, z-Transform, FIR Filter, Frequency Response, Causality, Impulse Response, Aliasing, Nulling Filter, Convolution, Linearity, Time-Invariance, VonHann Window, Sinc Function, Running Average, First Difference, Music, Beethoven, Fur Elise, and Stem Plot. This list of terms is meant to be suggestive of concept names, but you are not required to include all these terms and you are not restricted to this list.

Please print out your concept map for your lab report, but also try to ***save it to the web*** by using that option in ***CNT.*** Include an identifier that refers to Lab #8.

Lab #8
ECE-2025
Fall-2004
INSTRUCTOR VERIFICATION PAGE

For each verification, be prepared to explain your answer and respond to other related questions that the lab TA's or professors might ask. Turn this page in at the end of your lab period.

Name: _____ Date of Lab: _____

Part 3.1(d) and (f) Use the **`dltidemo`** GUI to illustrate the operation of a 9-point averaging filter. Determine the amount of delay through the filter and the frequency components that are nulled, and write your answers in the space below.

Verified:_____ Date/Time:_____

Part 3.2(b,c,d,e) Use the **`dltidemo`** GUI to demonstrate lowpass, highpass and bandpass filters. For each filter, write the expressions for the output $y[n]$ in the space below.

$y_L[n] =$

$y_H[n] =$

$y_B[n] =$

Verified:_____ Date/Time:_____

Part 3.3(b) Demonstrate that your M-file for LPF design works correctly by using it in the **`dltidemo`** GUI to obtain a length-37 LPF.

Verified:_____ Date/Time:_____