GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Spring 2006
Lab #8: Filter Design: Anti-Aliasing Filters

Date: 8–14 March 2006

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 be **due the first time your lab meets after Spring 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:

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

where M is the filter order, which should be an **even integer**. The the first term in $h[n]$ is a ***sinc function***; second term in $h[n]$ is called a Gaussian window. 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 $\hat{\omega}_c = 0.32\pi$) as a function of $\hat{\omega}$ in the range $-\pi \leq \hat{\omega} \leq \pi$:

```
M = 50;   nn = 0:M;
alpha = 2;
GW =  exp(-0.5*( ((alpha*(nn-M/2))/(M/2)).^2 ));   %-- Gaussian window
wc = 0.32*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.*GW;                  %-- 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)), zoom on, grid on
subplot(2,1,2);
plot(ww, angle(HH)), zoom on, grid on, shg
xlabel('Normalized Radian Frequency')
```

For FIR filters, the second argument of `freqz( _, 1, _ )` must always be equal to `1`. 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 the plots of $H(e^{j\hat{\omega}})$.[2]

2.2 GUI for Filter Design

There is a brand new GUI called `filterdesign` that is now available with the *SP-First* toolbox (for version 1.27 and later). The interface is shown in Fig. 1. Both FIR and IIR filters can be designed, but we will only be interested in the FIR case which would be selected from the drop-down list in the upper right. Once `FIR Filters` is selected, the window type should be selected from the drop-down list in the lower right. Lastly, it is necessary to set the order of the FIR filter and the cutoff frequency; these parameters can be entered in the edit boxes. For practice, try to carry out the same FIR filter design as in the MATLAB code in the previous section, i.e., $\alpha = 2$, $M = 50$ and $\hat{\omega}_c = 0.32\pi$.

2.3 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 Fig. 2 is a lowpass filter whose magnitude is

[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 as in the code above.

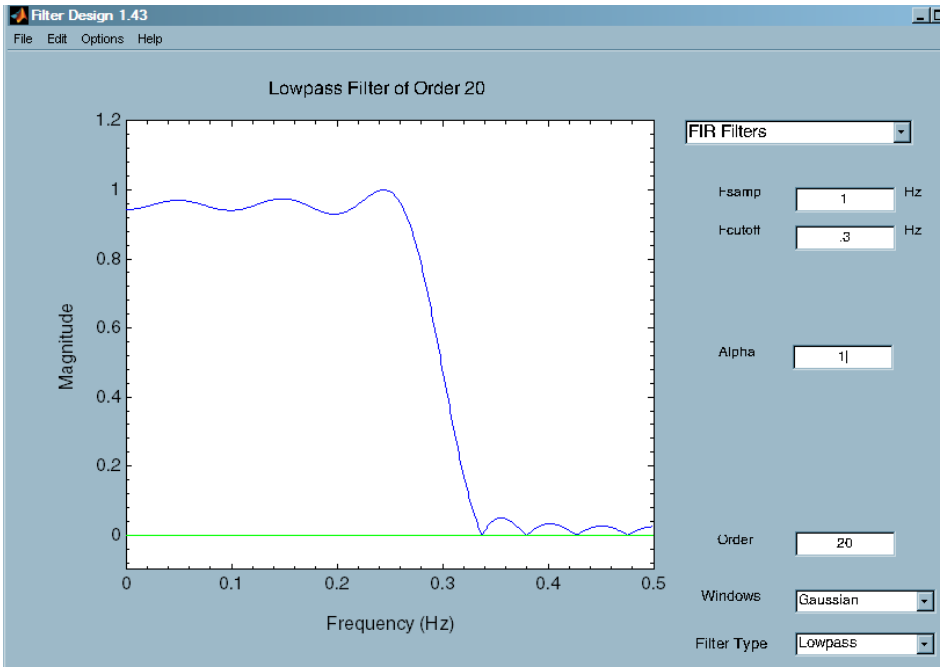

Figure 1: Filter design demo interface. The frequency is Hz, not $\hat{\omega}$. When the Filter Choice is set to `FIR Filters`, many different window types can be selected, including the Gaussian window.

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 the magnitude is very close to one, as defined by $\left||H(e^{j\hat{\omega}})| - 1\right|$ being less than 0.01. 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 $\alpha = 2$, $M = 50$ and $\hat{\omega}_c = 0.32\pi$.

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

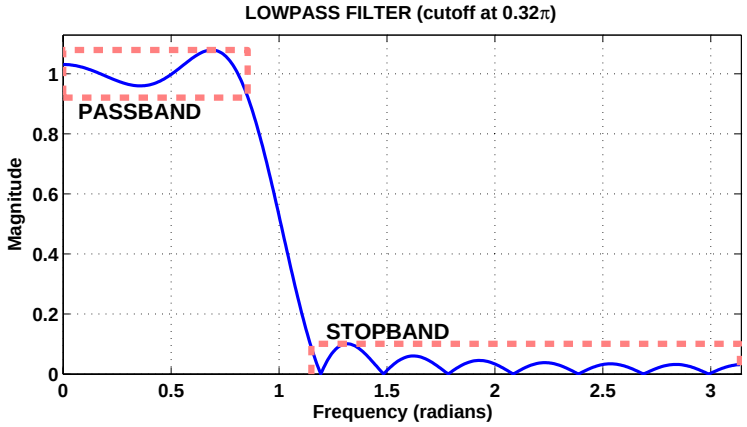

Figure 2: Passband and stopband defined for a typical lowpass filter. This particular filter is a length-21 FIR filter designed with a rectangular window and a sinc function with a cutoff frequency of $\hat{\omega}_c = 0.32\pi$. The approximate value of the passband edge is $\hat{\omega}_p = 0.27\pi = 0.85$; the stopband edge, $\hat{\omega}_s = 0.366\pi = 1.15$.

2.4 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 Fig. 2, 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 that you made in Section 2.1.1 determine the set of frequencies where the magnitude is nearly zero, as defined by $|H(e^{j\hat{\omega}})|$ being less than 0.01. 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 $\alpha = 2$, $M = 50$ and $\hat{\omega}_c = 0.32\pi$.

(b) The parameter $\hat{\omega}_s$ is called the *stopband edge.* Compare the value of $\hat{\omega}_s$ found in the previous part, and the value of $\hat{\omega}_p$ from Section 2.3 part (a), to the design parameter $\hat{\omega}_c$ in (2).

2.5 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 (piecewise) 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 $-n_s$.

3 Warm-up

The objective of this warm-up is to use the MATLAB GUIs **`filterdesign`** and **`dltidemo`**.[3] A second objective is to demonstrate the frequency responses of lowpass, highpass and bandpass filters designed by the *Gaussian-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 3 shows the interface for the **`dltidemo`** GUI. The parameters of the input sinusoid are controlled with the sliders in the lower left-hand region of the GUI. 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; likewise, right-clicking on a signal point in the left or right panels will give the numerical value of the signal at one index. The output signal is always shown, and it is also 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 GUI window.
In the Warm-up, you should perform the following steps with the **`dltidemo`** GUI:

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

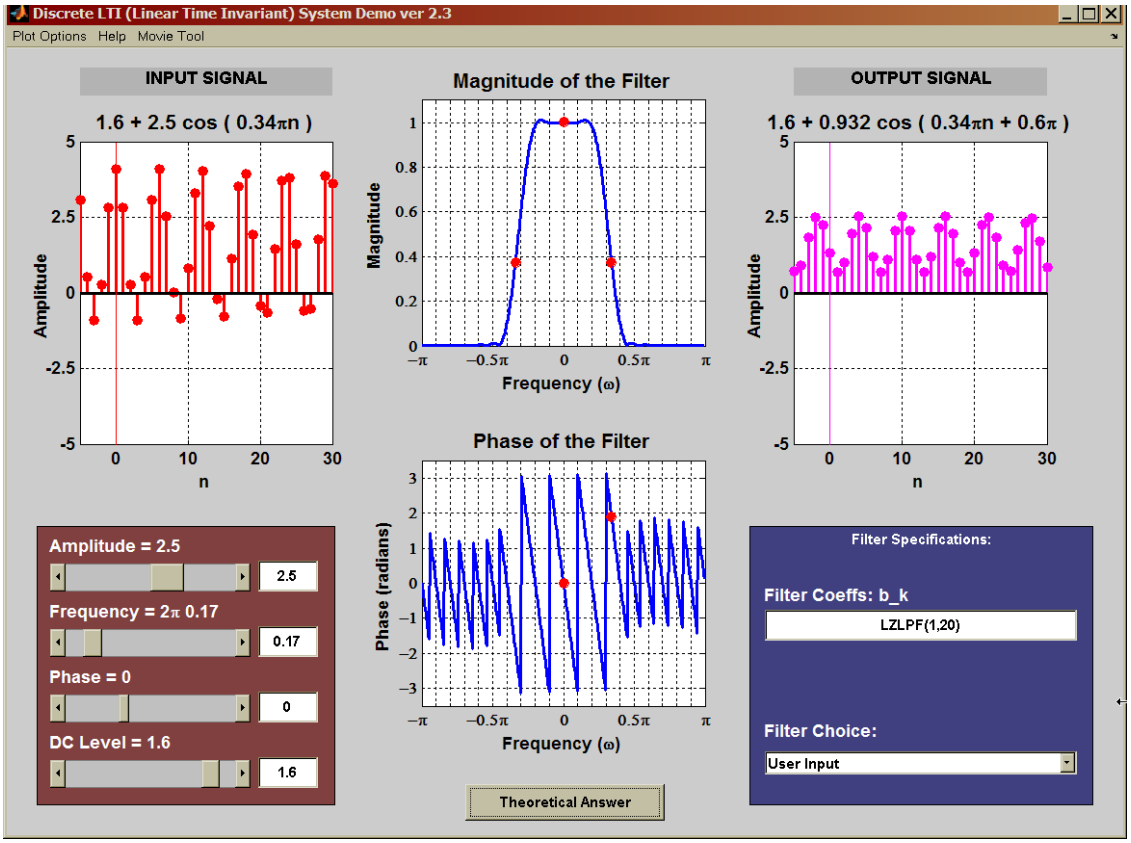

Figure 3: DLTI demo interface. The frequency label is ω because MATLAB won't display $\hat{\omega}$. When the Filter Choice is set to `User Input`, MATLAB code can be entered in the text box for the filter coefficients.

(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., make $y[n] = 0$. 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 $\hat{\omega} = 0.1\pi$. Determine the other frequencies $\hat{\omega}$ that 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 Filter Design GUI

In this section you can experiment with different values for the parameters of the Gaussian window to see the effect on the frequency response of the FIR filter.

(a) Start the **filterdesign** GUI, and select a set of nominal parameters: $\alpha = 1$, $\hat{\omega}_c = 0.3\pi$, and $M = 30$. Observe the frequency response so that comparisons can be made in the following parts.

(b) Change the filter order, M. Describe how the frequency response changes as M is increased or decreased. Double M or halve M.

(c) Change the value of α. Describe what happens when α is increased to 2, 3, and 5; and what happens when α is decreased to 0.5, 0.2 and 0.1. Concentrate of features of the frequency response.

Instructor Verification (separate page)

3.4 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 = GWLPF( wc, M, alpha )
%GWLPF   LPF design function with Gaussian window
%
%    wc = design parameter giving the approximate location
%          of the passband and stopband edges
%     M = filter order
% alpha = parameter of the Gaussian window
%    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 `GWLPF` in the text box of the user input. Make a length-37 FIR LPF (with $\alpha = 2$) whose cutoff frequency is approximately 0.37π. Recall that the length of an FIR filter is $L = M + 1$.

4 Lab Exercises

Among the many uses of lowpass filters (LPFs) is their role as *anti-aliasing* filters when downsampling signals or images. One everyday use of anti-aliasing filters is the display of fonts on a computer monitor. "Anti-aliased fonts" solve the following problem: how to display text on a low-resolution device. Since high quality printing (as seen in books) requires the equivalent of about 1200 or 2400 dots per inch (dpi), while a computer screen generally has 72 dots per inch, significant downsampling is needed to render text on a computer monitor. In reality, the problem is not that severe because laser printers at 600 dpi can produce printed pages that are indistinguishable from typeset book pages.

From the theory of sampling and aliasing, we know that high-frequency signal components will be aliased if the Sampling Theorem is not obeyed. This brings up the question of how to describe displayed fonts as having high-frequency or low-frequency components. The purpose of this lab exercise is to show that LPFs used prior to downsampling are beneficial. The LPF removes high-frequency components prior to downsampling and, as a result, the signal being downsampled is forced to obey the sampling theorem by the "anti-aliasing LPF."

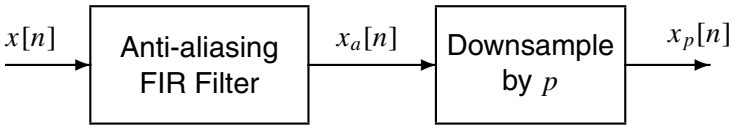

Figure 4: Cascade of anti-aliasing LPF and a downsampler.

The preceding discussion raises the following question: "if the signal $x[n]$ is downsampled by a factor of 2, how do we guarantee that there will be no aliasing?" Furthermore, how do we generalize to the case where the downsampling is by a factor of p? The answer comes from the following ad-hoc reasoning: Suppose that $x[n]$ came from the sampling of the analog signal $x(t)$ at a sampling rate of f_s. If we downsample $x[n]$ by a factor of two, we are effectively resampling the analog signal $x(t)$ at a rate of $\frac{1}{2}f_s$. At this lower sampling rate, the signal must have no frequency components above $0.25f_s$. Now, when $x[n]$ was created by sampling at the rate of f_s, the frequency $f = 0.25f_s$ mapped to $\hat{\omega} = 2\pi f/f_s = 0.5\pi$. Therefore, we conclude that downsampling $x[n]$ by a factor of $p = 2$ will not alias if the discrete-time spectrum of $x[n]$ has no frequency components in the region $0.5\pi \leq |\hat{\omega}| \leq \pi$. One way to enforce this condition is to apply a digital LPF to $x[n]$, and design the LPF to have its stopband cover the region $0.5\pi \leq |\hat{\omega}| \leq \pi$. On the other hand, its passband should cover the region $|\hat{\omega}| < 0.5\pi$ because we want to preserve all the frequency components in the original $x(t)$ below $0.25f_s$. A lowpass filter that performs this task is called an *anti-aliasing* filter.

4.1 Downsampling Music

Changing the sampling rate of an audio signal makes it easy to see how the LPF performs its anti-aliasing function. In this section, you must design an anti-aliasing filter and then downsample a music signal by 3 from 22050 Hz to 7350 Hz. The music signal is a snippet of the song "I Will Always Love You" by Whitney Houston; in particular, it has one sustained note that has high-frequency harmonics that might alias at lower sampling rates.

(a) Design a length-45 LPF that will perform the anti-aliasing. Explain how the cutoff frequency is chosen, and where stopband and passband edges actually end up.

(b) Load the signal from `WH_1.wav`, and then change its sampling rate from 22050 Hz to 7350 Hz by first filtering with the anti-aliasing filter and then downsampling.

(c) Display three spectrograms: the original signal, the original downsampled (without an anti-alias LPF), and the signal from the previous part. Explain how these spectrograms confirm the useful function of the anti-aliasing filter.

(d) In addition, listen to the three signals in order to hear the aliasing. Describe how the aliasing sounds to you.

4.2 Anti-Aliased Fonts

In this section, you must change the DPI (dots per inch) of a text image from 600 dpi to 100 dpi. The original image is too large to fit on the screen because its dimensions are 4076×2965, but when reduced by a factor of six the size is less than 700×600, which should fit on all computer monitors. In the test below, the effects of aliasing versus anti-aliasing can be seen visually.

(a) Design a length-45 LPF that will perform the anti-aliasing. Explain how the cutoff frequency is chosen, and where stopband and passband edges actually end up.

(b) Load the image from `page2.png` using `imread`, and then change its sampling rate from 600 dpi to 100 dpi by first filtering with the anti-aliasing filter and then downsampling. In the case of 2-D signals such as images, the FIR filter must be applied to all the rows and all the columns of the image.

(c) Display two images: the original downsampled (without an anti-alias LPF), and the signal from the previous part. Explain how these images confirm the benefits of the anti-aliasing filter.

Image Display: The original image is black and white with numerical values of either 0 or 255. After the LPF is applied the numerical values of the image will actually be outside of the [0,255] range. In addition, the start-up and ending regions of the FIR Filter might produce values that do not translate to reasonable gray-scale values. Since we want an image display where black is black, and white is white, the numerical values greater than 255 should be set equal to 255, and the values less than zero should be set equal to zero. This can be accomplished with `show_img` by using the "noscale" option, or with the M-file `clip` from the *SP-First* toolbox.

(d) Print three images for your lab report: the original, the original downsampled (without an anti-alias LPF), and the signal from the previous part. Assess the quality of the correctly downsampled image versus the original.

(e) Determine whether changing the length of the FIR filter (greater than 45 or less than 45) would change the quality of the processed image very much. Generally, you would want to use the shortest possible filter, so how short can you make the FIR filter and still obtain a good output.

4.3 Efficiency of 2-D Filtering (Optional)

This section is primarily for information. The original image is extremely large, so the time to run the FIR filter over all the rows and columns can be very long. Depending on how you call the `conv2` function, the time can vary quite a bit. To quantify the running time, compare the time to filter all the columns versus filtering all the rows.

```
bb = GWLPF(???);   %-- design the filter
tic, yy = conv2(bb', xx); toc    %-- xx holds the image
tic, yy = conv2(bb, xx); toc
```

Armed with this information, you can devise different strategies for doing the filtering efficiently. For example, you can filter the rows by doing three steps: transpose the image, filter the columns, and transpose it back. Which way is faster?

Lab #8
ECE-2025
Spring-2006
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 Demonstrate changes in the frequency response of a Gaussian-window filter with the **`filterdesign`** GUI.

Verified:_____ Date/Time:_____