GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 2001
Lab #6: FIR Filtering of Images

Date: 3–9 Oct 2000

This is *the official* Lab #6 description.

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.

It is only necessary to turn in Section 3 as this week's lab report.

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 the submitted work should be original and it should be your own work.

The lab report for this week will be an **Informal Lab Report.**

The report will **due the next time your lab meets: either 10, 11-Oct, or 22, 23-Oct at the start of your lab**.
Note: if you have one week to do this report, you will have two weeks to do the Lab #7 report, and vice versa. There will be no labs the week of Fall break, 15-18 October, but Recitations will be held.

---

1 Pre-Lab

The goal of this lab is to learn how to implement FIR filters in MATLAB, and then study the response of FIR filters to various signals, including images and speech. As a result, you should learn how filters can create interesting effects such as blurring and echoes. In addition, we will use FIR filters to study the convolution operation and properties such as linearity and time-invariance.

In the experiments of this lab, you will use `firfilt( )`, or `conv()`, to implement 1-D filters and `conv2()` to implement two-dimensional (2-D) filters. The 2-D filtering operation actually consists of 1-D filters applied to all the rows of the image and then all the columns.

1.1 Two GUIs

This lab involves on the use of two MATLAB GUIs: one for sampling and aliasing and one for convolution.

1. **con2dis:** GUI for sampling and aliasing. An input sinusoid and its spectrum is tracked through A/D and D/A converters.

2. **dconvdemo:** GUI for discrete-time convolution. This is exactly the same as the MATLAB functions `conv()` and `firfilt()` used to implement FIR filters.

Both of these demos can be downloaded from the "Lab page" in WebCT; the `dconvdemo` is available from the web page: `http://users.ece.gatech.edu/mcclella/matlabGUIs/index.html`

1.2 Overview of Filtering

For this lab, we will define an FIR *filter* as a discrete-time system that converts an input signal $x[n]$ into an output signal $y[n]$ by means of the weighted summation:

$$y[n] = \sum_{k=0}^{M} b_k \, x[n-k] \tag{1}$$

Equation (1) gives a rule for computing the n^{th} value of the output sequence from certain values of the input sequence. The filter coefficients $\{b_k\}$ are constants that define the filter's behavior. As an example, consider the system for which the output values are given by

$$\begin{aligned} y[n] &= \tfrac{1}{3}x[n] + \tfrac{1}{3}x[n-1] + \tfrac{1}{3}x[n-2] \\ &= \tfrac{1}{3}\left\{x[n] + x[n-1] + x[n-2]\right\} \end{aligned} \tag{2}$$

This equation states that the n^{th} value of the output sequence is the average of the n^{th} value of the input sequence $x[n]$ and the two preceding values, $x[n-1]$ and $x[n-2]$. For this example the b_k's are $b_0 = \frac{1}{3}$, $b_1 = \frac{1}{3}$, and $b_2 = \frac{1}{3}$.

CD-ROM
`firfilt.m`

MATLAB has a built-in function, `filter( )`, for implementing the operation in (1), but we have also supplied another M-file `firfilt( )` for the special case of FIR filtering. The function `filter` implements a wider class of filters than just the FIR case. Technically speaking, the `firfilt` function implements the operation called *convolution*. The following MATLAB statements implement the three-point averaging system of (2):

```
nn = 0:99;                    %<--Time indices
xx = cos( 0.08*pi*nn );       %<--Input signal
bb = [1/3 1/3 1/3];           %<--Filter coefficients
yy = firfilt(bb, xx);         %<--Compute the output
```

In this case, the input signal `xx` is a vector containing a cosine function. In general, the vector `bb` contains the filter coefficients $\{b_k\}$ needed in (1). These are loaded into the `bb` vector in the following way:

```
bb = [b0, b1, b2, ...  , bM].
```

In MATLAB, all sequences have finite length because they are stored in vectors. If the input signal has, for example, L samples, we would normally only store the L samples in a vector, and would assume that $x[n] = 0$ for n outside the interval of L samples; i.e., we do not have to store any zero samples unless it suits our purposes. If we process a finite-length signal through (1), then the output sequence $y[n]$ will be longer than $x[n]$ by M samples. Whenever `firfilt( )` implements (1), we will find that

```
length(yy) = length(xx)+length(bb)-1
```

In the experiments of this lab, you will use `firfilt( )` to implement FIR filters and begin to understand how the filter coefficients define a digital filtering algorithm. In addition, this lab will introduce examples to show how a filter reacts to different frequency components in the input.

1.3 Pre-Lab: Run the GUIs

The first objective of this lab is to demonstrate usage of the two GUIs. First of all, you must download the ZIP files for each and install them. Each one installs as a directory containing a number of files. You can put the GUIs on the `matlabpath`, or you can run the GUIs from their home directories.

1.4 Sampling and Aliasing Demo

In this demo, you can change the frequency of an input signal that is a sinusoid, and you can change the sampling frequency. The GUI will show the sampled signal, $x[n]$, it spectrum and also the reconstructed output signal, $y(t)$ with its spectrum. Figure 1 shows the interface for the `con2dis` GUI. In order to see the entire GUI, you must select `Show All Plots` under the `Plot Options` menu.

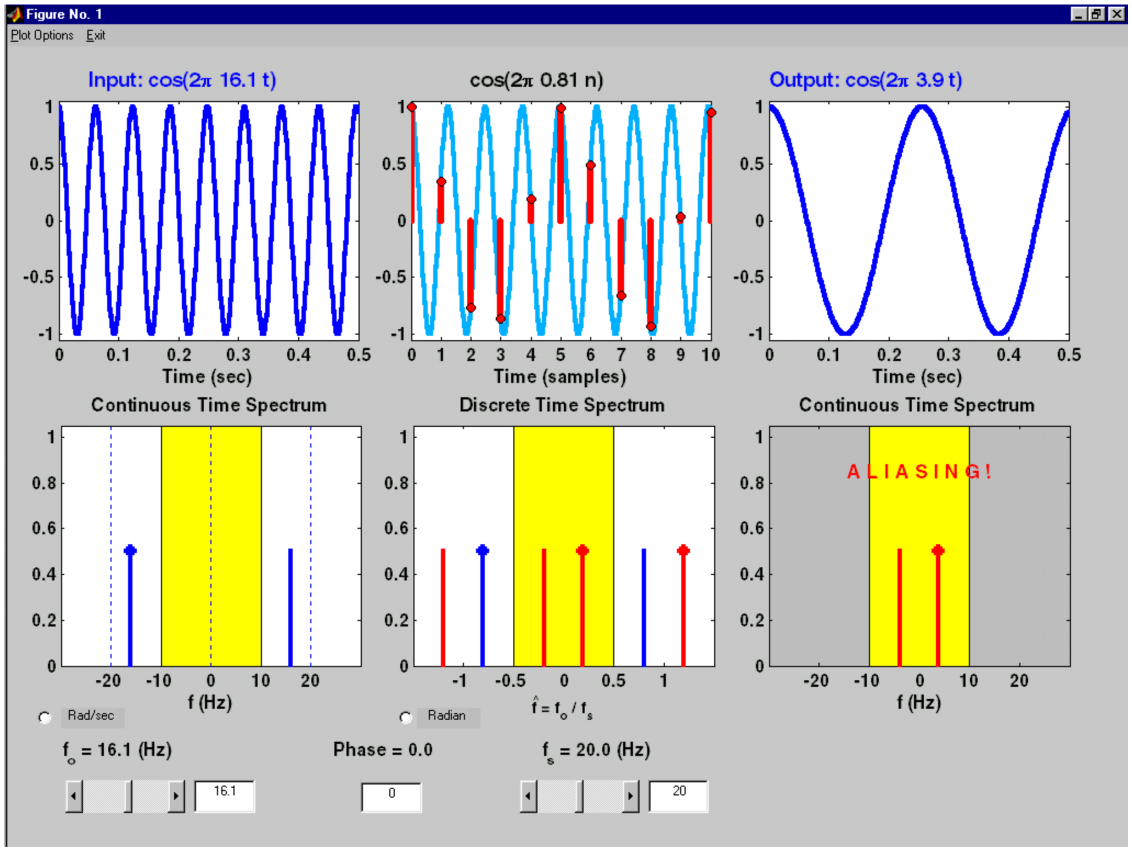

Figure 1: LTI demo interface.

In the pre-Lab, you should perform the following steps with the `con2dis` GUI:

(a) Set the input to $x(t) = \cos(40\pi t)$

(b) Set the sampling rate to $f_s = 24$ samples/sec.

(c) Determine the locations of the spectrum lines for the discrete-time signal, $x[n]$, found in the middle panels. Click the `Radian` button to change the axis to from $\hat{f}$ to $\hat{\omega}$.

(d) Determine the formula for the output signal, $y(t)$ shown in the rightmost panels. What is the output frequency in Hz?

1.5 Discrete-Time Convolution Demo

In this demo, you can select an input signal $x[n]$, as well as the impulse response of the filter $h[n]$. Then the demo shows the "flipping and shifting" used when a convolution is computed. This corresponds to the sliding window of the FIR filter. Figure 2 shows the interface for the `dconvdemo` GUI.

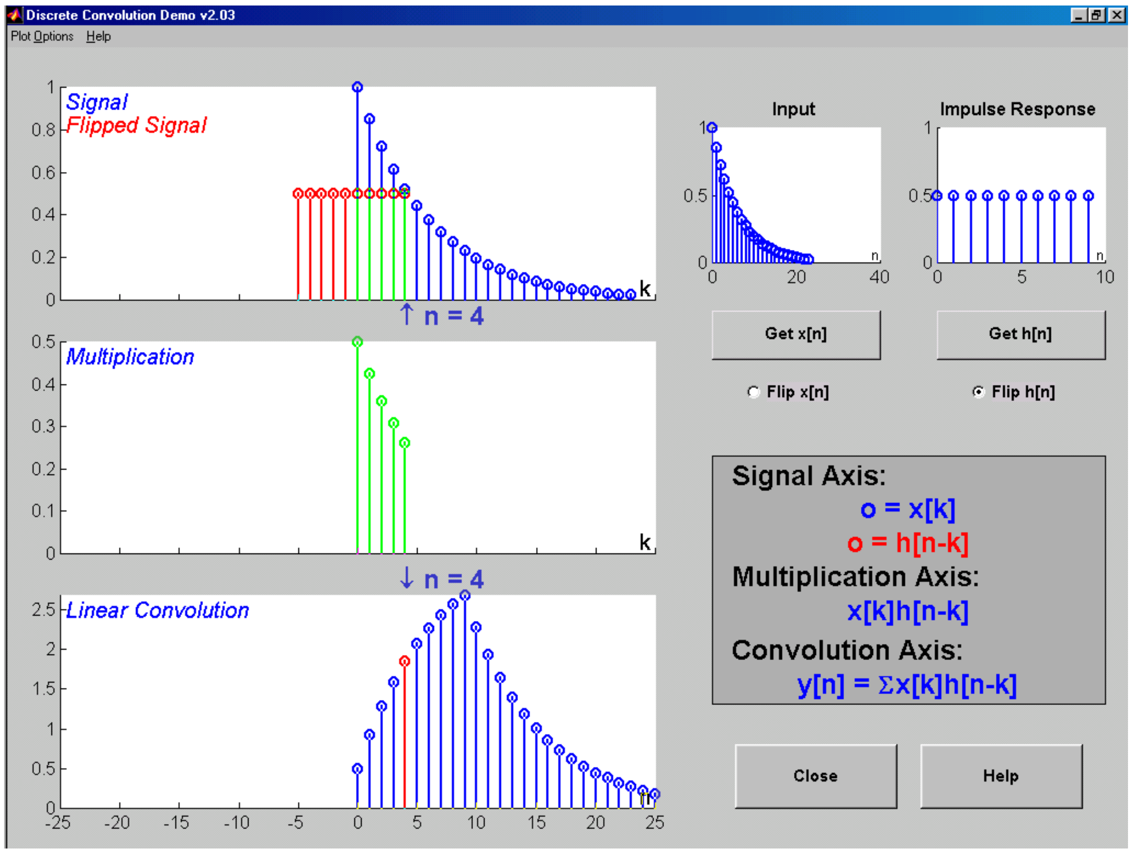

Figure 2: Interface for discrete-time convolution GUI.

In the pre-lab, you should perform the following steps with the `dconvdemo` GUI.

(a) Click on the `Get x[n]` button and set the input to a finite-length pulse: $x[n] = (u[n] - u[n-10])$.

(b) Set the filter to a three-point averager by using the `Get h[n]` button to create the correct impulse response for the three-point averager. Remember that the impulse response is identical to the b_k's for an FIR filter. Also, the GUI allows you to modify the length and values of the pulse.

(c) Use the GUI to produce the output signal.

(d) When you move the mouse pointer over the index "n" below the signal plot and do a click-hold, you will get a *hand tool* that allows you to move the "n"-pointer. By moving the pointer horizontally you can observe the sliding window action of convolution. You can even move the index beyond the limits of the window and the plot will scroll over to align with "n."

1.6 Filtering via Convolution

You can perform the same convolution as done by the `dconvdemo` GUI by using the MATLAB function `firfilt`, or `conv`. For ECE-2025, the preferred function is `firfilt`.

(a) For the Pre-Lab, you should do the filtering with a 3-point averager. The filter coefficient vector for the 3-point averager is defined via:

```
bb = 1/3*ones(1,3);
```

Use `firfilt` to process an input signal that it a length-10 pulse:

$$x[n] = \begin{cases} 1 & \text{for } n = 0,1,2,3,4,5,6,7,8,9 \\ 0 & \text{elsewhere} \end{cases}$$

NOTE: in MATLAB indexing can be confusing. Our mathematical signal definitions start at $n = 0$, but MATLAB starts its indexing at "1". Nevertheless, we can ignore the difference and pretend that MATLAB is indexing from zero, as long as we don't try to write `x[0]` in MATLAB. Thus we can generate the length-10 pulse and put it inside of a longer vector via `xx = [ones(1,10),zeros(1,5)].`

(b) To illustrate the filtering action of the 3-point averager, it is informative to make a plot of the input signal and output signals together. Since $x[n]$ and $y[n]$ are discrete-time signals, a `stem` plot is needed. One way to put the plots together is to use `subplot(2,1,*)` to make a two-panel display:

```
nn = first:last;     %--- use first=1 and last=length(xx)
subplot(2,1,1);
stem(nn,xx(nn))
subplot(2,1,2);
stem(nn,yy(nn),'filled')   %--Make black dots
xlabel('Time Index (n)')
```

This code assumes that the output from `firfilt` is called `yy`. Try the plot with `first` equal to the beginning index of the input signal, and `last` chosen to be the last index of the input. In other words, the plotting range for both signals will be equal to the length of the input signal, even though the output signal is longer.

(c) Explain the filtering action of the 3-point averager by comparing the plots in the previous part. This filter might be called a "smoothing" filter. Note how the transitions in $x[n]$ from zero to one, and from one back to zero, have been "smoothed."

2 Warm-up

2.1 Sampling and Aliasing

Use the `con2dis` GUI to do the following problem:

(a) Input frequency is 12 Hz.

(b) Sampling frequency is 15 Hz.

(c) Determine the frequency of the reconstructed output signal

(d) Determine the locations in $\hat{\omega}$ of the lines in the spectrum of the discrete-time signal. Give numerical values.

Instructor Verification (separate page)

(e) Change the sampling frequency to 12 Hz, and explain the appearance of the output signal.

2.2 Discrete-Time Convolution

In this section, you will generate filtering results needed in a later section. Use the discrete-time convolution GUI, `dconvdemo`, to do the following:

(a) Set the input signal to be $x[n] = (0.9)^n \, (u[n] - u[n-10])$. Use the "Exponential" signal type within `Get x[n]`.

(b) Set the impulse response to be $h[n] = \delta[n] - 0.9\delta[n-1]$. Once again, use the "Exponential" signal type within `Get h[n]`.

(c) Illustrate the output signal $y[n]$ and explain why it is zero for almost all points. Compute the numerical value of the last point in $y[n]$, i.e., the one that is negative and non-zero.

Instructor Verification (separate page)

2.3 Filtering Images: 2-D Convolution

One-dimensional FIR filters, such as running averagers and first-difference filters, can be applied to one-dimensional signals such as speech or music. These same filters can be applied to images if we regard each row (or column) of the image as a one-dimensional signal. For example, the 50th row of an image is the N-point sequence `xx[50,n]` for $1 \leq n \leq N$, so we can filter this sequence with a 1-D filter using the `conv` or `firfilt` operator.

One objective of this lab is to show how simple 2-D filtering can be accomplished with 1-D row and column filters. It might be tempting to use a `for` loop to write an M-file that would filter all the rows. This would create a new image made up of the filtered rows:

$$y_1[m,n] = x[m,n] - x[m,n-1]$$

However, this image $y_1[m,n]$ would only be filtered in the horizontal direction. Filtering the columns would require another `for` loop, and finally you would have the completely filtered image:

$$y_2[m,n] = y_1[m,n] - y_1[m-1,n]$$

In this case, the image $y_2[m,n]$ has been filtered in both directions by a first-difference filter

These filtering operations involve a lot of `conv` calculations, so the process can be slow. Fortunately, MATLAB has a built-in function `conv2( )` that will do this with a single call. It performs a more general filtering operation than row/column filtering, but since it can do these simple 1-D operations it will be very helpful in this lab.

(a) Load in the image `echart.mat` with the `load` command (it will create the variable `echart` whose size is 257×256). We can filter all the rows of the image at once with the `conv2( )` function. To filter the image in the horizontal direction using a first-difference filter, we form a *row* vector of filter coefficients and use the following MATLAB statements:

```
bdiffh = [1, -1];
yy1 = conv2(echart, bdiffh);
```

In other words, the filter coefficients `bdiffh` for the first-difference filter are stored in a *row* vector and will cause `conv2( )` to filter all rows in the *horizontal* direction. Display the input image `echart` and the output image `yy1` on the screen at the same time. Compare the two images and give a qualitative description of what you see.

(b) Now filter the "eye-chart" image `echart` in the *vertical* direction with a first-difference filter to produce the image `yy2`. This is done by calling `yy2 = conv2(echart,bdiffh')` with a column vector of filter coefficients. Display the image `yy2` on the screen and describe in words how the output image compares to the input.

Instructor Verification (separate page)

3 Lab: FIR Filters

In the following sections we will study how a filter can produce the following special effects:

1. *Echo:* FIR filters can produce echoes and reverberations because the filtering formula (1) contains delay terms. In an image, such phenomena would be called "ghosts."

2. *Deconvolution:* one FIR filter can (approximately) undo the effects of another—we will investigate a cascade of two FIR filters that distort and then restore an image. This process is called *deconvolution.*

3.1 Deconvolution Experiment for 1-D Filters

Use the function `firfilt( )` to implement the following FIR filter

$$w[n] = x[n] - 0.9x[n-1] \tag{3}$$

on the input signal $x[n]$ defined via the MATLAB statement: `xx = 256*(rem(0:100,50)<10);` In MATLAB you must define the vector of filter coefficients `bb` needed in `firfilt`.

(a) Plot both the input and output waveforms $x[n]$ and $w[n]$ on the same figure, using `subplot`. Make the discrete-time signal plots with MATLAB's `stem` function, but restrict the horizontal axis to the range $0 \leq n \leq 75$. Explain why the output appears the way it does by figuring out (mathematically) the effect of the filter coefficients in (3).

(b) Note that $w[n]$ and $x[n]$ are not the same length. Determine the length of the filtered signal $w[n]$, and explain how its length is related to the length of $x[n]$ and the length of the FIR filter. (If you need a hint refer to Section 1.2.)

3.1.1 Restoration Filter

The following FIR filter

$$y[n] = \sum_{\ell=0}^{M} r^{\ell} w[n-\ell] \qquad \text{(FIR FILTER-2)}$$

can be use to undo the effects of the FIR filter in the previous section (see the block diagram in Fig. 3). It performs restoration, but it only does this approximately. Use the following steps to show how well it works when $r = 0.9$ and $M = 22$.

(a) Process the signal $w[n]$ from (3) with FILTER-2 to obtain the output signal $y[n]$.

(b) Make stem plots of $w[n]$ and $y[n]$ using a time-index axis n that is the same for both signals. Put the stem plots in the same window for comparison—using a two-panel subplot.

(c) Since the objective of the restoration filter is to produce a $y[n]$ that is almost identical to $x[n]$, make a plot of the error between $x[n]$ and $y[n]$ over the range $0 \leq n < 50$.

3.1.2 Worst-Case Error

(a) Evaluate the *worst-case error* by doing the following: find the maximum of the difference between $y[n]$ and $x[n]$ in the range $0 \leq n < 50$.

(b) What does the error plot and worst case error tell you about the quality of the restoration of $x[n]$? How small do you think the worst case error has to be so that it cannot be seen on a plot?

3.2 Cascading Two Systems

More complicated systems are often made up from simple building blocks. In the system of Fig. 3 two FIR filters are connected "in cascade." For this section, assume that the the filters in Fig. 3 are described by the two equations:

$$w[n] = x[n] - q\,x[n-1] \qquad \text{(FIR FILTER-1)}$$

$$y[n] = \sum_{\ell=0}^{M} r^{\ell} w[n-\ell] \qquad \text{(FIR FILTER-2)}$$

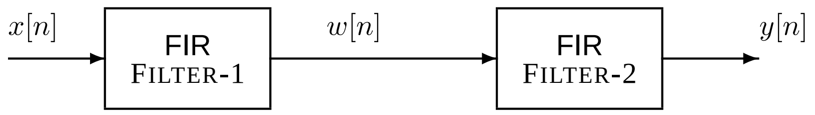

Figure 3: Cascading two FIR filters: the second filter attempts to "deconvolve" the distortion introduced by the first.

3.2.1 Overall Impulse Response

(a) Implement the system in Fig. 3 using MATLAB to get the impulse response of the overall cascaded system for the case where $q = 0.9$, $r = 0.9$ and $M = 22$. Use two calls to `firfilt()`. Plot the impulse response of the overall cascaded system.

(b) Work out the impulse response $h[n]$ of the cascaded system by hand to verify that your MATLAB result in part (a) is correct. (Hint: consult old Homework problems.)

(c) In a *deconvolution* application, the second system (FIR FILTER-2) tries to undo the convolutional effect of the first. Perfect deconvolution would require that the cascade combination of the two systems be equivalent to the identity system: $y[n] = x[n]$. If the impulse responses of the two systems are $h_1[n]$ and $h_2[n]$, state the condition on $h_1[n] * h_2[n]$ to achieve perfect deconvolution.[1]

3.2.2 Distorting and Restoring Images

If we pick q to be a little less than 1.0, then the first system (FIR FILTER-1) will cause distortion when applied to the rows and columns of an image. The objective in this section is to show that we can use the second system (FIR FILTER-2) to undo this distortion (more or less). Since FIR FILTER-2 will try to undo the convolutional effect of the first, it acts as a *deconvolution* operator.

CD-ROM
echart.mat

(a) Load in the image `echart.mat` with the `load` command. It creates a matrix called `echart`.

(b) Pick $q = 0.9$ in FILTER-1 and filter the image `echart` in both directions: apply FILTER-1 along the horizontal direction and then filter the resulting image along the vertical direction also with FILTER-1. Call the result `ech90`.

(c) Deconvolve `ech90` with FIR FILTER-2, choosing $M = 22$ and $r = 0.9$. Describe the visual appearance of the output, and explain its features by invoking your mathematical understanding of the cascade filtering process. Explain why you see "ghosts" in the output image, and use some previous calculations to determine how big the ghosts (or echoes) are, and where they are located. Evaluate the *worst-case error* in order to say how big the ghosts are relative to "black-white" transitions which are 0 to 255.

3.2.3 A Second Restoration Experiment

(a) Now try to deconvolve `ech90` with several different FIR filters for FILTER-2. You should set $r = 0.9$ and try several values for M such as 11, 22 and 33. Pick the best result and explain why it is the best. Describe the visual appearance of the output, and explain its features by invoking your mathematical understanding of the cascade filtering process. HINT: determine the impulse response of the cascaded system and relate it to the visual appearance of the output image.
Hint: you can use `dconvdemo` to generate the impulse responses of the cascaded systems, like you did in the Warm-up.

(b) Furthermore, when you consider that a gray-scale display has 256 levels, how large is the worst-case error (from the previous part) in terms of number of gray levels? Do this calculation for each of the three filters in part (a). Think about the following question: "Can your eyes perceive a gray scale change of one level, i.e., one part in 256?"

Include all images and plots for the previous two parts to support your discussions in the lab report.

[1]Note: the cascade of FIR FILTER-1 and FILTER-2 does not perform *perfect* deconvolution.

Lab #6
ECE-2025
Fall-2001
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 2.1: Demonstrate that you can run the `con2dis` GUI. Calculate the locations of the spectrum lines for the discrete-time signal. Write the values of $\hat{\omega}$ below.

Verified:____________________ Date/Time:___________

Part 2.2: Demonstrate that you can run the `dconvdemo` GUI. Explain why the output is zero for most points.

Verified:____________________ Date/Time:___________

Part 2.3(a),(b) Process the input image `echart` with a 2-D filter that filters in both the horizontal and vertical directions with a first difference filter. Explain how the filter changes the "image signal."

Verified:____________________ Date/Time:___________