GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

**ECE 2025 Fall 2000**
**Lab #6: FIR Filtering of Images**

Date: 3–9 Oct 2000

---

This is *the official* Lab #6 description.

**Lab Quiz #1 will be given at the beginning of this lab.** You can use your old lab reports and run MATLAB during the Lab Quiz.

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**. One of the lab instructors must verify the appropriate steps by initialing on the **Instructor Verification** line. Turn in the verification sheet at the end of your lab time.

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

The report will **due during the week of 10–16 Oct. at the start of your lab**.

---

# 1 Introduction

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. In addition, we will use FIR filters to study 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 will consist of 1-D filters applied to all the rows of the image and then all the columns. As a result, you should learn how filters can create interesting effects such as blurring and echoes.

There are three activities in this lab:

1. Study properties of FIR filters.
2. Distort an image by applying an FIR filter, and then attempt to remove the distortion by applying a second FIR filter. This removal process is called *deconvolution.*
3. Produce echoes (and possibly reverberations) in a sound signal, again by FIR filtering.

# 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 $n^{\text{th}}$ 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

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

This equation states that the $n^{\text{th}}$ value of the output sequence is the average of the $n^{\text{th}}$ 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 $b_2 = \frac{1}{3}$.

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):

firfilt.m

```
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.

# 3 Warm-up

## 3.1 Loading Data

In order to exercise the basic filtering function `firfilt`, we will use some "real" data. In MATLAB you can load data from a file called `lab6dat.mat` file by using the `load` command as follows:

```
load lab6dat
```

lab6dat.mat

The data file `lab6dat.mat` contains two filters and three signals, stored as separate MATLAB variables:

- `x1:` a stair-step signal such as one might find in one sampled scan line from a TV test pattern image.
- `xtv:` an actual scan line from a digital image.

`x2:` a speech waveform ("oak is strong") sampled at $f_s = 8000$ samples/second.

`h1:` the coefficients for a FIR discrete-time filter of the form of (1).

`h2:` coefficients for a second FIR filter.

After loading the data, use the `whos` function to verify that all five vectors are in your MATLAB workspace.

### 3.2 Filtering a Signal

You will now use the signal vector `x1` as the input to an FIR filter.

(a) For the warm-up, you should do the filtering with a 5-point averager. The filter coefficient vector for the 5-point averager is defined via:

```
bb = 1/5*ones(1,5);
```

Use `firfilt` to process `x1`. How long are the input and output signals?

When unsure about a command, use `help`.

(b) To illustrate the filtering action of the 5-point averager, you must make a plot of the input signal and output signal together. Since $x_1[n]$ and $y_1[n]$ 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;
subplot(2,1,1);
stem(nn,x1(nn))
subplot(2,1,2);
stem(nn,y1(nn),'filled')   %--Make black dots
xlabel('Time Index (n)')
```

This code assumes that the output from `firfilt` is called `y1`. 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) Since the previous plot is quite crowded, it is useful to show a small part of the signals. Repeat the previous part with `first` and `last` chosen to display 30 points from the middle of the signals.

(d) Explain the filtering action of the 5-point averager by comparing the plots from parts (b) and (c). This filter might be called a "smoothing" filter. Note how the transitions from one level to another have been "smoothed." Make a sketch of what would happen with a 2-point averager.

**Instructor Verification** (separate page)

### 3.3 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 $50^{\text{th}}$ 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 $257 \times 256$). 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)

# 4 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.*

### 4.1 Deconvolution Experiment for 1-D Filters

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

$$w[n] = x[n] - 0.75x[n-1] \tag{3}$$

on the input signal $x[n]$ defined via the MATLAB statement: `xx = 256*(rem(0:100,34)<10);` In MATLAB you must define the vector `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 $0 \leq n \leq 70$. Explain why the output appears the way it does by figuring out 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 2.)

#### 4.1.1 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 Fig. 1). It performs restoration, but it only does this approximately. Use the following steps to show how well it works when $r = 0.75$ and $M = 24$.

(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 extends from $n = 0$ to $n = 70$. 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 $0 \leq n \leq 70$.

#### 4.1.2 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 $0 \leq n \leq 70$. If you divide the worst case error by the maximum amplitude of $x[n]$ what is the percentage error?

(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 in order to be insignificant?

(c) Finally, justify your result by using convolution to find a formula for the output $y[n]$ in terms of the input $x[n]$. This is easiest to do if you combine the two FIR filters into one "equivalent filter" before convolving with the input signal.
Note: you should be able to calculate the worst-case error from your formula for the impulse response of the equivalent filter.

## 4.2 Cascading Two Systems

More complicated systems are often made up from simple building blocks. In the system of Fig. 1 two FIR filters are connected "in cascade." For this section, assume that the the filters in Fig. 1 are described by the two equations:

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

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

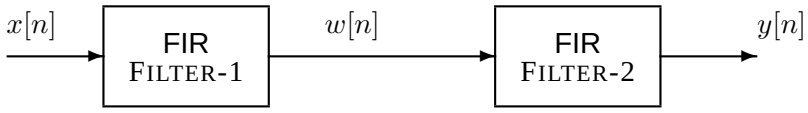



Figure 1: Cascading two FIR filters: the second filter "deconvolves" the distortion introduced by the first.

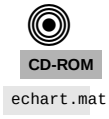
(a) Implement this system using MATLAB to get the impulse response of the overall cascaded system for the case where $q = 0.95$, $r = 0.95$ and $M = 24$. Use two calls to `firfilt()`. Plot the impulse response of the overall cascaded system.

(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]

### 4.2.1 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.




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

(b) Pick $q = 0.75$ in FILTER-1 and filter the image `echart` in both the horizontal and vertical directions with FILTER-1. Call the result `ech75`.

(c) Now try to deconvolve `ech75` with FIR FILTER-2. You have to "design" FILTER-2 by choosing an appropriate value for $r$ to get the filter coefficients for FILTER-2. You should use $M = 24$ and try several values for $r$ such as 0.5, 0.75 and 0.95. 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.

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

### 4.2.2 A Second Restoration Experiment

(a) Now do a second distortion experiment with a different FIR FILTER-1. Pick $q = 0.95$ and filter `echart` in both the horizontal and vertical directions with FILTER-1. Call the result `ech95`.

(b) Deconvolve `ech95` with FIR FILTER-2, choosing $M = 24$ and $r = 0.95$. 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. For example, what is the *worst-case error* when $r = 0.95$?

### 4.2.3 Summary of Restoration Experiments

You have performed two separate image restoration experiments: one for the case where $q = 0.75$ and the other for $q = 0.95$. Explain why the restoration works better in one case than the other. Use the worst-case error as a measure of quality. Furthermore, when you consider that a gray-scale display has 256 levels, how large is the worst-case error in terms of gray levels?

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

# Lab #6
# ECE-2025
# Fall-2000
# INSTRUCTOR VERIFICATION PAGE

Staple this page to the end of your Lab Report.

Name: _____ Date of Lab: _____

Completed Lab Quiz #1 by clicking on the FINISH button.

Verified:_____ Date/Time:_____

Part 3.2(b,c,d) Process the input signal `x1` with a 5-point averager by using `firfilt.m`. Display 30 points from the middle of the input and output signals. Make a *rough* sketch (below) of what the the output would be if the filter were a 2-point averager (use the same range of time indices).

Verified:_____ Date/Time:_____

Part 3.3(a,b) Process the input image `echart` with a 2-D filter that processes in both the horizontal and vertical directions with a first difference filter. Explain how the filter changes the "image signal."

Verified:_____ Date/Time:_____