GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 1999
Lab #6: FIR Filtering: Images & Audio

Date: 5–11 Oct 1999

This is *the official* Lab #6 description.

Lab Quiz #2 will be given at the beginning of this lab.

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 laboratory instructors must verify the appropriate steps by initialing on the **Instructor Verification** line. When you have completed a step that requires verification, simply raise your hand and demonstrate the step to the instructor.

The lab report for this week will an **Informal Lab Report.** Staple the **Instructor Verification** sheet to the end of your lab report as evidence that the appropriate steps were witnessed by the instructor.

The report will **due during the week of 12–18 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. Blur an image by applying an FIR filter, and then attempt to remove the blur 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^{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^{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 $b_2 = \frac{1}{3}$.

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):

CD-ROM
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 non-zero samples, and would assume that $x[n] = 0$ for n outside the interval of L samples; i.e., we do not have to store the 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
```

CD-ROM
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 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 `x1` as the input to an FIR filter.

(a) For the warm-up, 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 `x1`. How long are the input and output signals?

When unsure about a command, use `help`.

(b) To illustrate the filtering action of the 3-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 just 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 3-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 7-point averager.

Instructor Verification (separate page)

(e) Make a plot similar to part (b) by using the `inout` function. Call `inout` so that it breaks the signals into sections of length 50. In this case, a `stem` format is not used; instead, the "envelope" or "outline" of the discrete-time signals is shown.

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^{th} row of an image is the N-point sequence `xx[50,n]` for $1 \le n \le N$, 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:

$$y_1[m,n] = \frac{1}{11}\sum_{\ell=0}^{10} x[m, n-\ell] \qquad \text{for } 1 \le m \le M$$

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:

$$y_2[m,n] = \frac{1}{11}\sum_{k=0}^{10} y_1[m-k, n] \qquad \text{for } 1 \le n \le N$$

In this case, the image $y_2[m,n]$ has been filtered in both directions by a 11-point averager.

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 11-point averager, we form a *row* vector of filter coefficients and use the following MATLAB statements:

```
bh = ones(1,11)/11;
yy1 = conv2(echart, bh);
```

In other words, the filter coefficients `bh` for the 11-point averager stored in a *row* vector 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 11-point running averager to produce the image `yy2`. This is done by calling `yy2 = conv2(echart,bh')` 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. that FIR filters can produce echoes and reverberations because the filtering formula (1) contains delay terms

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

4.1 First Difference Filter

Use the function `firfilt( )` to implement the following FIR filter on the signal `x1` (from the `lab6dat` file).

$$y_1[n] = x_1[n] - x_1[n-1] \tag{3}$$

This is called a *first-difference* filter. In MATLAB you must define the vector `bb` needed in `firfilt`.

(a) Note that $y_1[n]$ and $x_1[n]$ are not the same length. Determine the length of the filtered signal $y_1[n]$, and explain how its length is related to the length of $x_1[n]$ and the length of the FIR filter. (If you need a hint refer to Section 2.)

(b) Plot the first 40 samples of both waveforms $x_1[n]$ and $y_1[n]$ on the same figure, using `subplot`. Use the `stem` function to make a discrete-time signal plot, but label the horizontal axis to run over the range $0 \leq n \leq 39$. By looking at the choice of filter coefficients, explain why the output appears the way it does.

4.2 Time-Invariance of the Filter

Time-invariance means that we can place a delay system before or after any FIR filter and get the same result.

(a) Define the filter coefficients $\{b_k\}$ of an FIR filter that will perform a "delay-by-3" operation.

(b) Process the signal `x1` through the "delay-by-3" filter, and then process it through the first difference filter. Call the output signal `z1`.

(c) Process the signal $y_1[n]$ from Section 4.1 through the "delay-by-3" filter. Call this output signal `w1`. Verify that `w1` is, in fact, a delayed version of `y1` by comparing plots of the two signals versus n.

(d) Is `w1` equal to `z1` for every index n? Explain why or why not.

4.3 Linearity (Superposition) of the Filter

(a) Now multiply the vector `x1` from Section 4.1 by two to get `xa=2*x1`. Generate the signal `ya` by filtering `xa` with the first difference filter given by (3).

(b) Now generate a new input vector `xb` of length 10 corresponding to the discrete-time signal

$$x_b[n] = \delta[n]$$

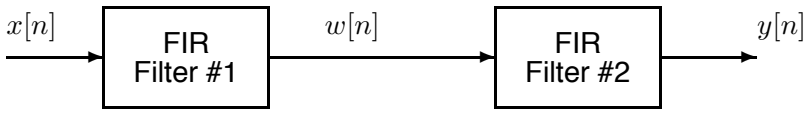
and then filter it through the first difference operator to get `yb`. Describe this output signal in terms of the impulse response of the first-difference filter.

(c) Now form another input signal `xc` that is the sum of `xa` and `xb`.[1] Run `xc` through the filter to get `yc` and then plot the first 15 points of `yc`. Compare `yc` to a plot of `ya + yb`. Are they equal ? Explain any differences that you observe.

[1] You will have to *zero-pad* the shorter signal in order to be able to add them. This amounts to appending enough zeros to the shorter signal to make the lengths equal. For example, `xb = [xb,zeros(1,length(xa)-length(xb))];`

4.4 Cascading Two Systems

More complicated systems are often made up from simple building blocks. In the system below two FIR filters are connected "in cascade."



(a) First, assume that the above system is described by the two equations

$$w[n] = \sum_{\ell=0}^{M} \alpha^{\ell} x[n-\ell] \qquad \text{(FIR FILTER \#1)}$$

$$y[n] = w[n] - \alpha\, w[n-1] \qquad \text{(FIR FILTER \#2)}$$

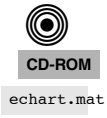
Implement this system using MATLAB to get the impulse response for the case where $\alpha = 0.8$ and $M = 9$. Plot the impulse response of the overall cascaded system.

(b) Work out the result for part (a) by hand (i.e., using convolution) as in a previous Homework.

(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.[2]

4.5 Blurring and Deconvolving Images

If we pick α to be a little less than 1.0, then the first system (FIR Filter #1) will cause blurring when applied to the rows and columns of an image. The objective in this section is to show that we can use the the second system (FIR Filter #2) to undo the blurring. 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 $\alpha = 0.9$ and $M = 10$ and filter `echart` in both the horizontal and vertical directions with Filter #1. Call the result `echart9`.

(c) Now try to deconvolve `echart9` with FIR filter #2. You have to "design" Filter #2 by choosing an appropriate value for α to get the filter coefficients for Filter #2. You should try several values for α such as 0.8, 0.9 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: relate the impulse response of the cascaded system to the visual appearance of the output image.

(d) Now do a second blurring experiment with a different FIR FIlter #1. Pick $\alpha = 0.7$ and $M = 10$ and filter `echart` in both the horizontal and vertical directions with Filter #1. Call the result `echart7`. Do you get more, or less, blurring than in part (a) ?

[2] The cascade of FIR Filters #1 and #2 does not perform perfect deconvolution.

(e) Deconvolve `echart7` with FIR filter #2, choosing the appropriate value for α. You should try several values for α such as 0.5, 0.7 and 0.9. 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.

(f) Include all images and plots for this part to support your discussions in the lab report. You have performed two separate de-blurring experiments: Explain why the de-blurring works better in one case than the other.

4.6 Filtering a Music Waveform (Extra Credit: 15 points)

Echoes and reverberation can be done by adding a delayed version of the original signal to itself. For this part, use the first 5 seconds of your synthesized song from Lab #4. In this experiment you will have to design FIR filters to process the music signal. Refer back to the section on cascading where the following two FIR filters were defined:

$$w[n] = \sum_{\ell=0}^{M} \alpha^{\ell} x[n-\ell] \qquad \text{(FIR FILTER \#1)}$$

$$y[n] = w[n] - \alpha\, w[n-1] \qquad \text{(FIR FILTER \#2)}$$

(a) In order to produce an echo that is audible, the delay time has to be fairly long with respect to the sampling rate. A delay of one sample at $f_s = 11025$ Hz is about 90.7 μsec. Instead, you need a delay of about 0.15 sec. for perception by the human hearing system . Determine the delay P needed in the following filter:

$$y[n] = \frac{1}{1+\alpha} w[n] + \frac{\alpha}{1+\alpha}\, w[n-P] \tag{4}$$

to produce an echo at 0.15 sec. The quantity α will control the strength of the echo—set it equal to 0.95 for this implementation. Then define the filter coefficients to implement the FIR filter in (4).

(b) Filter the music signal with filter defined in part (a). Describe the sound that you hear and use the impulse response to explain why it sounds that way.

(c) Reverberation requires multiple echoes. This can be accomplished by cascading several systems of the form (4). Use the parameters determined in part (a), and derive (by hand) the impulse response of a reverb system produced by cascading four "single echo" systems.

(d) Filter the music signal with filter defined in part (c). Describe the sound that you hear and use the impulse response to explain why it sounds that way.

(e) It will be difficult to make plots to show the echo and reverberation, but you should be able to do it with the M-file `inout( )` which can plot two very long signals together on the same plot. It formats the plot so that the input signal occupies the first, third, and fifth lines, etc. while the output signal is on the second, fourth, and sixth lines etc. Type `help inout` to find out more.

`inout.m`

You should plot about 0.5 sec of the original and each processed music signal to show the delay effects that are producing the echo(es). Pick a segment containing only a few notes so that you can see the delayed signals. Label the plots to point out the differences between the original and the echoed/reverb signals. Note: it will be tricky to illustrate the effect that you want to explain, but you have to find a way to see the delayed versions of the original.

Lab #6
ECE-2025
Fall-1999
INSTRUCTOR VERIFICATION PAGE

Staple this page to the end of your Lab Report.

Name: _____ Date of Lab: _____

Part 3.2(b,c,d) Process the input signal `x1` with a 3-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 7-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 averaging filter that processes in both the horizontal and vertical directions. Explain how the filter changes the "image signal."

Verified:_____ Date/Time:_____