Lab S-4: Convolution & FIR Filters

Pre-Lab: Read the Pre-Lab and do all the exercises in the Pre-Lab section ***prior to attending lab.***

Verification: The Exercise section of each lab should be completed **during your assigned Lab time** and the steps marked *Instructor Verification* signed off **during the lab time**. One of the laboratory instructors must verify the appropriate steps by signing on the **Instructor Verification** line. When you have completed a step that requires verification, demonstrate the step to your instructor. Turn in the completed verification sheet before you leave the lab.

Lab Homework Questions: The Lab-Homework Sheet has a few lab related questions that can be answered at your own pace. The completed Lab-HW sheet is due at the beginning of the next lab.

1 Pre-Lab

Please read through the information below prior to attending your lab.

1.1 Objective

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 or speech.

1. In the experiments of this lab, you will use the MATLAB GUI called `dconvdemo` to study first-difference filters and the convolution of rectangular pulses. If you have installed the *SP-First* or (*DSP-First*) Toolbox, you will already have this demo on the `matlabpath`.

2. You will also 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.

1.2 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 formula:

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

Equation (1) gives a rule for computing the n^{th} value of the output sequence from present and past 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 all the same: $b_0 = \frac{1}{3}$, $b_1 = \frac{1}{3}$, and $b_2 = \frac{1}{3}$.

MATLAB has two built-in functions, `conv()` and `filter()`, for implementing the operation in (1), and the *SP-First* (or *DSP-First*) toolbox supplies another M-file, called `firfilt()`, for the special case of FIR filtering. The function `filter` implements a wider class of filters than just the FIR case. Technically speaking, both the `conv` and the `firfilt` function implement 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 (example)
bb = [1/3 1/3 1/3];         %<--Filter coefficients
yy = firfilt(bb, xx);       %<--Compute the output
```

In this case, the input signal `xx` is contained in a vector defined by the cosine function. In general, the vector `bb` contains the filter coefficients $\{b_k\}$ needed in (1). The `bb` vector is defined 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 L nonzero samples, we would normally store only the L nonzero samples in a vector, and would assume that $x[n] = 0$ for n outside the interval of L samples, i.e., don't 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 Unit-Step Notation

The *unit step* signal is very handy for defining finite-length signals. Recall the definition of $u[n]$

$$\text{(unit-step signal)} \qquad u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{3}$$

The unit-step signal makes a transition from zero to one at $n = 0$. A shifted unit-step signal such as $u[n-d]$ makes the transition at $n = d$. Thus, if we want to define a signal that is one for $0 \leq n < L$, then we write

$$u[n] - u[n-L] = \begin{cases} 1 & 0 \leq n \leq L-1 \\ 0 & n < 0 \;\text{ or }\; n \geq L \end{cases}$$

This notation is used everywhere in `dconvdemo` to denote finite-length signals.

1.4 Discrete-Time Convolution Demo GUI

This lab involves the use of a MATLAB GUI for convolution of discrete-time signals, **`dconvdemo`**. This GUI illustrates convolution which is the same operation done in the MATLAB functions `conv()` and `firfilt()` used to implement FIR filters. This demo is part of the *SP-First* (or *DSP-First*) Toolbox. 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 ***sliding window*** view of FIR filtering, where one of the signals must be *flipped and shifted* along the axis when convolution is computed. Figure 1 shows the interface for the **`dconvdemo`** GUI.
In the pre-lab, you should perform the following steps with the **`dconvdemo`** GUI.

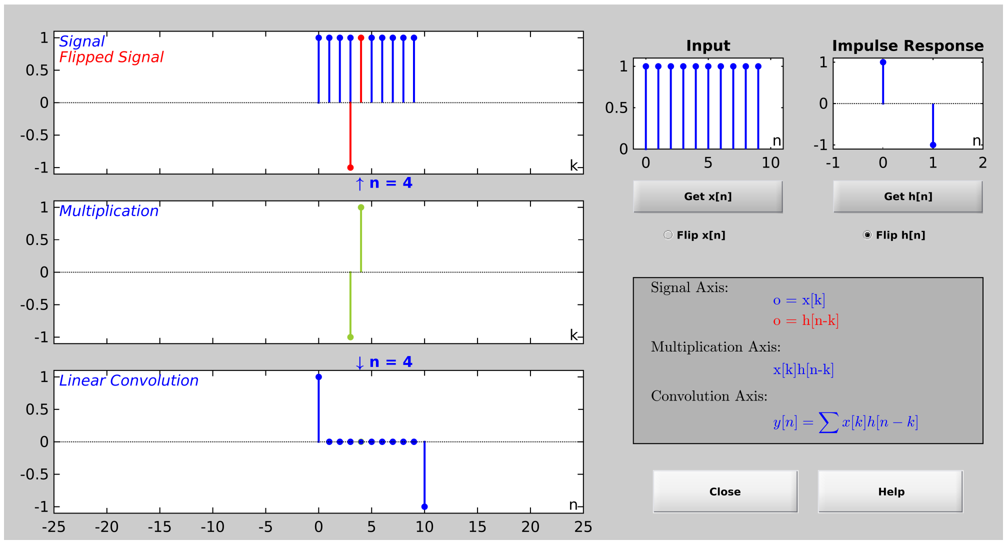

Figure 1: Interface for discrete-time convolution GUI called **dconvdemo**. This particular case is the convolution of a three-point averager with a ten-point rectangular pulse.

(a) Click on the **Get x[n]** button and set the input to a finite-length pulse: $x[n] = (u[n] - u[n-10])$. Note the length of this pulse, as well as the start and end points of the signals.

(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) Observe that the GUI produces the output signal in the bottom panel.

(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 to the left or right; or you can use the left and right arrow keys. 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."

1.5 Filtering via Convolution

You can perform exactly the same convolution as done by the **dconvdemo** GUI if you use the MATLAB function **firfilt**, or **conv**. For ECE-2026, the preferred function is **firfilt**.

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

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

Use **firfilt** to process an input signal that is 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.
The statement `xx = [ones(1,10),zeros(1,5)]` generates a length-10 pulse and put it inside of a longer vector. This produces a vector of length 15, which has 5 extra zero samples appended.

(b) To illustrate the filtering action of the three-point averager, it is informative to make a plot of the input signal and output signal 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-1,xx(nn))
subplot(2,1,2);
stem(nn-1,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. Notice that using `nn-1` in the two calls to `stem()` causes the x-axis to start at zero in the plot.

(c) Explain the filtering action of the three-point averager by comparing the plots in the previous part. This averaging filter might be called a "smoothing" filter, especially when we see how the transitions in $x[n]$ from zero to one, and from one back to zero, have become "smooth ramps."

2 Lab Exercise

2.1 Discrete-Time Convolution

In this section, you will generate filtering results for commonly used simple FIR filters. Use the discrete-time convolution GUI, **`dconvdemo`**, to do the following:

(a) The convolution of two impulses, $\delta[n-3]*\delta[n-5]$.

(b) Filter the input signal $x[n] = (-3)\{u[n-2]-u[n-8]\}$ with a *first-difference filter.*

$$y[n] = x[n] - x[n-1]$$

Make $x[n]$ by selecting the "Pulse" signal type from the drop-down menu within **`Get x[n]`**, and also use the text box "Delay."
Next, set the impulse response to match the filter coefficients of the first-difference. Enter the impulse response values by selecting "User Signal" from the drop-down menu within **`Get h[n]`**. Illustrate the output signal $y[n]$ and write a simple formula for $y[n]$ which should use only two impulses.

(c) Explain why $y[n]$ from the previous part is zero for almost all n.

Instructor Verification (separate page)

(d) Convolve two rectangular pulses: one with an amplitude of 2 and a length of 7, the other with an amplitude of 3 and a length of 4. Make a sketch of the output signal, showing its starting and ending points, as well as its maximum amplitude.

(e) State the *length* and *maximum amplitude* of the convolution result from the previous part.

Instructor Verification (separate page)

(f) The first-difference filter can be used to find the *edges* in a signal or in an image. This behavior can be exhibited with the `dconvdemo` GUI. Set the impulse response $h[n] = \delta[n] - \delta[n-1]$. In order to set the input signal $x[n]$, use the *User Input* option to define $x[n]$ via the MATLAB statement `double((sin(0.5*(0:50))+0.2)<0)`, which uses the logical operator "less than" to make a signal that has *runs* of zero and ones.
The output from the convolution $y[n]$ will have only a few nonzero values. Record the locations of the nonzero values, and explain how these are related to the *transitions* in the input signal. Also, explain why some values of $y[n]$ are positive, and others are negative.

Instructor Verification (separate page)

2.2 Lab-HW: Filtering an Image to Find Edges

One-dimensional FIR filters, such as running averagers and first-difference filters, can be used to process one-dimensional signals such as speech or music. These same filters can be applied to images if we regard each column (or row) 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 $1 \le n \le N$, so we can filter this sequence with a 1-D filter using the `conv` or `firfilt` operator, e.g., to filter the m_0-th row:

$$y_1[m_0, n] = x[m_0, n] - x[m_0, n-1]$$

(a) Load in the image `echart.mat` (from the *SP-First* or (*DSP-First*) Toolbox) with the `load` command. This will create the variable `echart` whose size is 257×256. Use MATLAB's `imshow` function (or `show_img` from the *SP-First* (or *DSP-First*) toolbox) to view the image.

(b) Then filter one row of the image with a first difference filter:

```
yy1 = firfilt(echart(m,:), firstDiffFilterCoeffs);
```

Pick a row where there the are several black-white-black transitions, e.g., choose row number 65, 147, or 221. Display the row of the input image `echart` and the filtered row `yy1` on the screen in the same figure window (with `subplot`). Use `stem` because these signals are one-dimensional signals.

(c) Compare the two stem plots and give a qualitative description. Note that the polarity (positive/negative) of the impulses will denote whether the transition is from white to black, or black to white.

(d) Then explain how to calculate the ***total width*** of the character "E" from the impulses in the stem plot of an ***appropriately chosen*** filtered row.
Note: Use the MATLAB function `find` to get the locations of the impulses in the filtered row `yy1`.

Lab: Convolution & FIR Filters
INSTRUCTOR VERIFICATION SHEET

Turn this page in to your lab grading TA before the end of your scheduled Lab time.

Name: ______________________ LoginUserName: ____________ Date: ______

Part	Observations (Write down answers for each part)
2.1(a)	Convolve impulses: $\delta[n-3] * \delta[n-5] =$ write formula
2.1(b)	Rectangular Pulse through a First-Difference filter: $y[n] =$ write formula for output signal
2.1(c)	Explain why the first-difference output $y[n]$ is zero for most values of n.
	Verified:____________________ Date/Time:__________
2.1(d)	Convolve two rectangles, <u>sketch result</u>; make sure you have the correct beginning and end!
2.1(e)	Maximum Amplitude and Length of the convolved-rectangles output.
	Verified:____________________ Date/Time:__________
2.1(f)	List the index locations (n) of the transitions in the output signal, $y[n]$
2.1(f)	Explain polarity (positive/negative) of the transitions in the output signal, $y[n]$
	Verified:____________________ Date/Time:__________

Lab: Convolution & FIR Filters
LAB HOMEWORK QUESTION

Turn this page in to your lab grading TA at the very beginning of your next scheduled Lab time.

Name: ______________________ LoginUserName: ____________ Date: ______

Part 2.2 Process one row of the input image `echart` with a 1-D first-difference filter. Explain how the output from the first-difference filter makes it easy to measure the width of black regions. Use MATLAB's `find` function to help in determining the width of the black "E" from the impulses in the first-difference output signal.

(a) Load in the image `echart.mat` (from the *SP-First* or (*DSP-First*) Toolbox) with the `load` command. This will create the variable `echart` whose size is 257×256. Use MATLAB's `imshow` function (or `show_img` (from the *SP-First* or (*DSP-First*) toolbox) to view the image.

(b) Then filter one row of the image with a first difference filter:

```
yy1 = firfilt(echart(m,:), firstDiffFilterCoeffs);
```

Pick a row where there the are several black-white-black transitions, e.g., choose row number 65, 147, or 221. Display the row of the input image `echart` and the filtered row `yy1` on the screen in the same figure window (with `subplot`). Use `stem` because these signals are one-dimensional signals.

(c) Compare the two stem plots and give a qualitative description. Explain why most of the output signal values (from the filter) are zero. Note that the polarity (positive/negative) of the impulses will denote whether the transition is from white to black, or black to white.

(d) Then explain how to calculate the ***total width*** of the character "E" from the impulses in the stem plot of an ***appropriately chosen*** filtered row.
Note: Use the MATLAB function `find` to get the locations of the impulses in the filtered row `yy1`.