DSP First

Lab 07: Sampling, Convolution, and FIR Filtering

Pre-Lab and Warm-Up: You should read at least the Pre-Lab and Warm-up sections of this lab assignment and go over all exercises in the Pre-Lab section before going to your assigned lab session.

Verification: 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. 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, simply demonstrate the step to the TA or instructor. Turn in the completed verification sheet to your TA when you leave the lab.

Lab Report: It is only necessary to turn in a report on Section 3 with graphs and explanations. You are asked to label the axes of your plots and include a title for every plot. In order to keep track of plots, include your plot inlined within your report. If you are unsure about what is expected, ask the TA who will grade your report.

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 the filter Vis in the DSP First Toolkit to implement 1-D and 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 Interactive Demos

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

1. Continuous to discrete demo: demo for sampling and aliasing. An input sinusoid and its spectrum is tracked through A/D and D/A converters.
2. Discrete convolution demo: demo for discrete-time convolution. This is similar to the MathScript function conv and other functions used to implement FIR filters.

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:

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

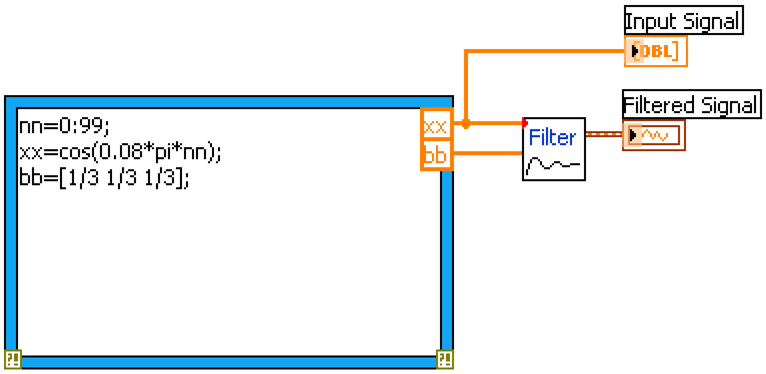
Equation (1) gives a rule for computing the nth value of the output sequence from certain values of the input sequence. The filter coefficients ¶bk◇are constants that define the filter's behavior.

As an example, consider the system for which the output values are given by

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

This equation states that the nth value of the output sequence is the average of the nth 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 $b_0 = 1/3$, $b_1 = 1/3$, and $b_2 = 1/3$.

LabVIEW has many FIR filtering VIs. However, for simplicity, we will use the filter VI in the DSP First toolkit. The following VI implements the three-point averaging system of (2):



```
nn=0:99;
xx=cos(0.08*pi*nn);
bb=[1/3 1/3 1/3];
```

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

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

In LabVIEW, all sequences have finite length because they are stored in vectors. If the input signal has, for example, Lsamples, we would normally only store the Lsamples in a vector, and would assume that $x[n] = 0$ for $n$outside the interval of Lsamples; 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 the filter VI implements (1), we will find that

```
length(filtered signal) = length(xx)+length(bb)-1
```

In the experiments of this lab, you will use the filter VI 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.

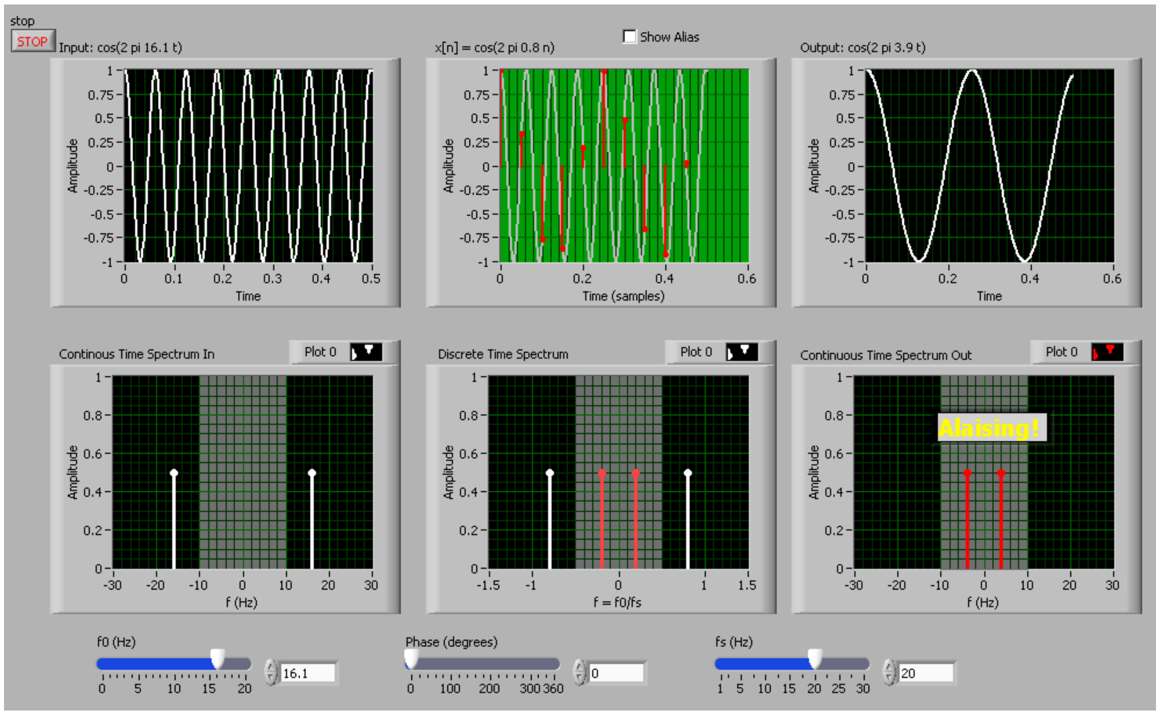
1.3 Pre-Lab: Run the GUIs

The first objective of this lab is to demonstrate usage of the two demos. These demos are on the course website and CD. You can also run them directly off the website or locally on your computer off the CD.

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

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


stop
STOP
Input: cos(2 pi 16.1 t)
x[n] = cos(2 pi 0.8 n)
Show Alias
Output: cos(2 pi 3.9 t)
Continous Time Spectrum In
Discrete Time Spectrum
Continuous Time Spectrum Out
Alaising!
f0 (Hz) 16.1
Phase (degrees) 0
fs (Hz) 20

(b) Set the sampling rate to $f\text{s} = 24$ 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 □ $f$to □ω.

(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. 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. 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:

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

(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 (which was "padded" with five extra zero samples).

(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."

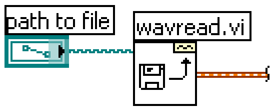
2 Warm-up

2.1 Sampling and Aliasing

2.2 Discrete-Time Convolution

2.3 Loading Data

In order to exercise basic filtering functions, we will use some "real" data. In LabVIEW, we can use prebuilt tools in the DSP First Toolkit to load data. To load a *.wav sound file, use "wavread.vi" as shown below.



The wavread VI will output a waveform signal which can be used for further manipulation and analysis.

The ZIP file lab07.zip 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 fs $= 8000$ 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.

2.4 Filtering a Signal

You will now use the signal vector derived from “x1.wav” as the input to an FIR filter.

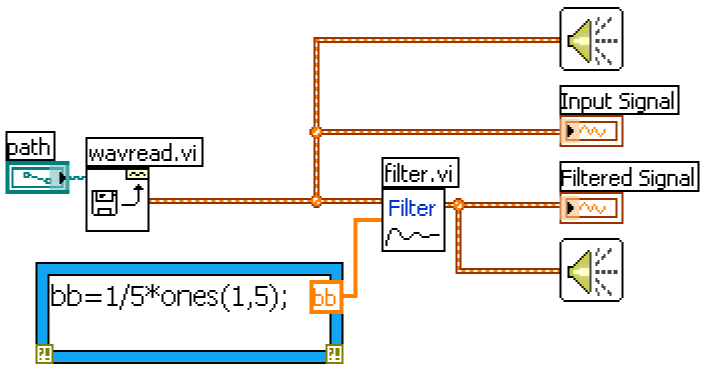
(a) For the warm-up, you should do the filtering with a 5-point averager. The filter coefficient vector for the 5-point averager is defined via:

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

Use the filter VI to process “x1.wav”. How long are the input and output signals?

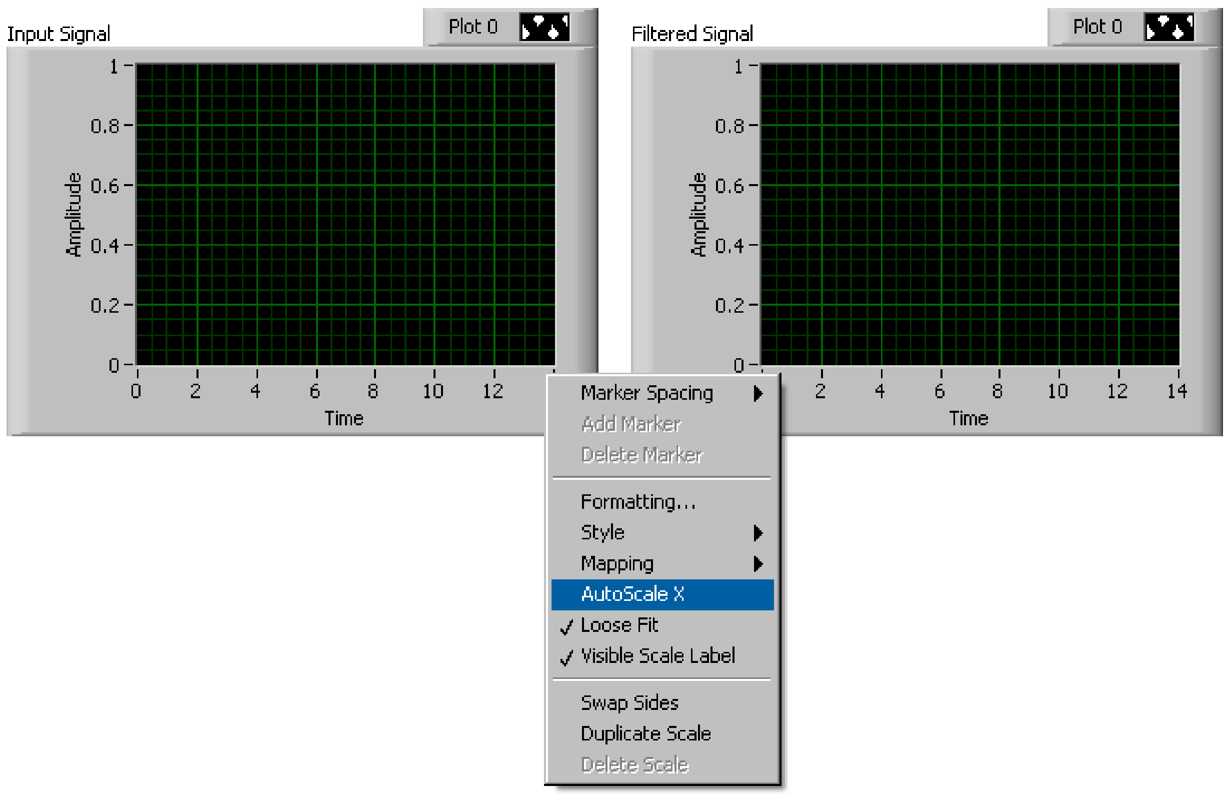
When unsure about a command, use help.

(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 *these* are discrete-time signals, a stem plot is needed.

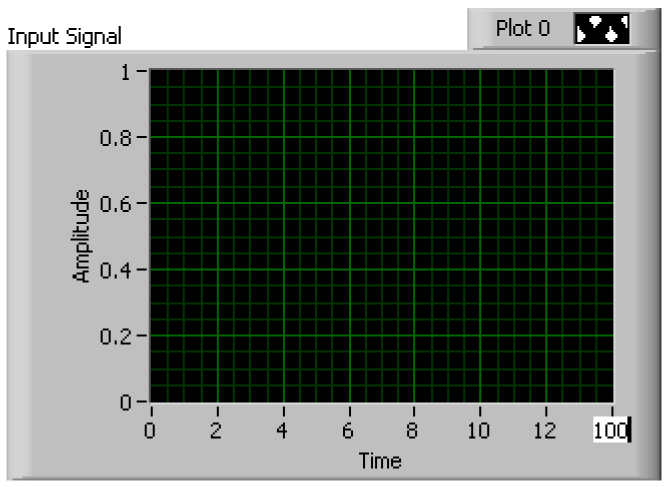


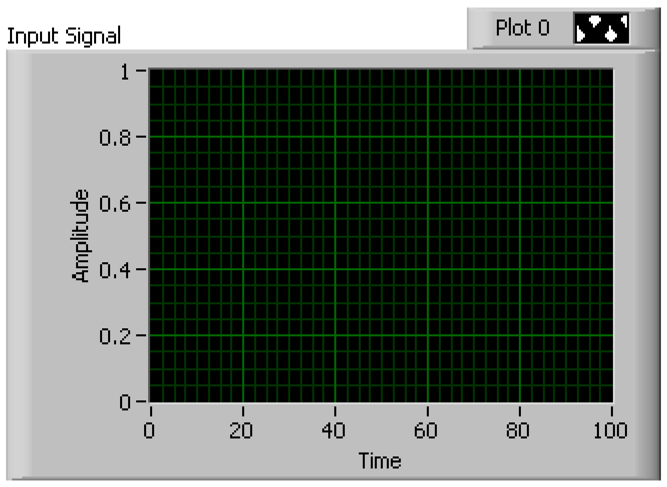
On the front panel, you can modify the scale of the output waveforms so that the plotting range for both signals is equal. Use the following procedure to change the scale.

First turn off autoscale by right-clicking on the axis that you want to modify and deselecting “AutoScale X”, where X is the axis you want to autoscale.



Then to set your own scale, simply click on the last number on the scale and change it to whatever you like. In the left image, we have changed it to 100. The scale will adjust accordingly, as shown in the right picture.





This can also be done programmatically, but these methods are beyond the scope of this course.

(c) Since the previous plot is quite crowded, it is useful to show a small part of the signals. Repeat the previous part with scale set 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)

2.5 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 $1 \le n \le N$, so we can filter this sequence with a 1-D filter using the FIR filter VI.

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 a VI 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 $y1[m,n]$ 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 $y2[m,n]$ has been filtered in both directions by a first-difference filter.

These filtering operations involve a lot of convolution calculations, so the process can be slow. Fortunately, a sub-VI has been constructed to apply an FIR filter to a pixel array in one call. The VI, “2-D Pixmap Filter.vi”, is located in the DSP First Toolkit.

(a) Load in the image echart.bmp using the included “picture to pixel array.vi” subroutine. We can filter all the rows of the image at once with the 2-D Pixmap Filter 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 MathScript statements:

```
bdiffh = [1, -1];
```

We then feed these coefficients into the 2-D Pixmap Filter function.

In other words, the filter coefficients `bdiffh` for the first-difference filter are stored in a row vector and will cause the 2-D filter to filter all rows in the horizontal direction.

Display the input image `echart` and the output image 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 another image. This is done by calling the sub-VI with a column vector of filter coefficients. Display the vertically filtered image on the screen and describe in words how the output image compares to the input.

3 Lab Exercises: 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

$$w[n] = x[n] - 0.9x[n-1]$$

on the input signal $x[n]$ defined via the MathScript statement:

```
xx = 256*(rem(0:100,50)<10);
```

You must define the vector of filter coefficients bb needed in the filter VI.

(a) Plot both the input and output waveforms $x[n]$ and $w[n]$ on the same figure. You can place two waveforms on the same plot by combining them with the Merge Signals VI. Make the discrete-time signal plots with stem plot control but restrict the horizontal axis to the range $0 \le n \le 75$. 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

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

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 $r = 0\triangleright 9$ 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 Merge Signals.

(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 (difference) between $x[n]$ and $y[n]$ over the range 0 □n

<50.

3.1.2 Worst-Case Error

(a) Evaluate the worst-case error by doing the following: use the Array Max & Min VI to find the maximum of the difference between $y[n]$ and $x[n]$ in the range 0 □n<50.

(b) What do 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?

3.1.3 An Echo Filter

The following FIR filter can be interpreted as an echo filter.

$$y_1[n] = x_1[n] + rx_1[n-P]$$

(a) You have an audio signal sampled at fs $=$ 8000 Hz and you would like to add a delayed version of the signal to simulate an echo. The time delay of the echo should be 0.2 seconds, and the strength of the echo should be 90% percent of the original. Determine the values of $r$and Pin (4); make Pan integer.

(b) Describe the filter coefficients of this FIR filter, and determine its length.

(c) Implement the echo filter in (4) with the values of $r$and Pdetermined in part (a). Use the speech signal in the file "x2.wav" found in the file lab07.zip file. Listen to the result to verify that you have produced an audible echo.

(d) (Optional) Implement an echo filter and apply it to your synthesized music from Lab #4. You will have to change the calculation of Pif you used fs $=$ 11025 Hz. Reduce the echo time (from 0.2 secs. down to zero) and try to determine the shortest echo time that can be perceived by human hearing.

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:

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

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

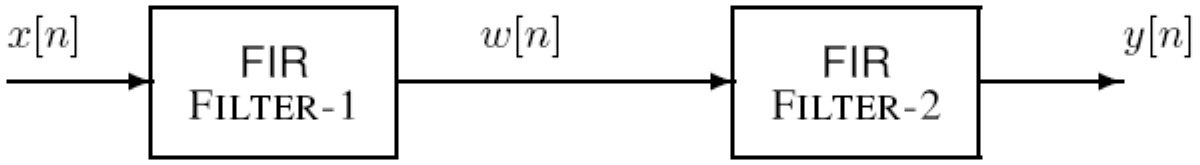


Figure 3: Cascading two FIR filters: the second filter attempts to "deconvolve" the distortion introduced by the first.

3.2.1 Overall Impulse Response

(a) Implement the system in Fig. 3 using LabVIEW 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 the filter VI. 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 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 h1[n] and h2[n], state the condition on $h_1[n] * h_2[n]$ to achieve perfect deconvolution.[1]

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

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.

(a) Load in the image echart.bmp with the picture to pixel array VI.

(b) Pick q = 0.9 in FILTER-1 and filter the image in both directions: apply FILTER-1 along the horizontal direction and then filter the resulting image along the vertical direction also with FILTER-1. In future sections, we will call this ech90.

(c) Deconvolve your result from (b) 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 discrete convolution demo 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.

3.3 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.

(a) In order to produce an echo that is audible, the delay time has to be fairly long compared to the sampling period. A delay of one sample at fs = 11025 Hz is about Ts = 1/fs = 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:

$$y[n] = \frac{1}{1+\alpha}\omega[n] + \frac{\alpha}{1+\alpha}\omega[n-P] \tag{5}$$

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 (5).

(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 (5). 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. Refer back to section 3.2 on cascading filters. Recall that two filters are said to be "in cascade" if the output of the first filter is used as the input to the second filter, and the output of the second filter is defined to be the output of the overall cascade system. This can be repeated for as many filters as are needed in the cascade system.

(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.

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 07
INSTRUCTOR VERIFICATION SHEET

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 continuous to discrete demo. Calculate the locations of the spectrum lines for the discrete-time signal. Write the values of □below.

Verified: _______________ Date/Time:_____________

Part 2.2: Demonstrate that you can run the discrete convolution demo. Explain why the output is zero for most points.

Verified: _______________ Date/Time:_____________

Part 2.5(a),(b) Process the input image "echart.bmp" 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:_____________