GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

EE 2200 Fall 1998
Lab #6: FIR Filtering of Sinusoids and Sampled Waveforms

Date: week of 2 Nov 1998

Lab Quiz: the next one will be the week of 17–19 Nov.

This is *the official* Lab #6 description; it is based on Lab C.5 in Appendix C of the text, but parts of Lab C.6 have been added.

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**.

The lab report for this lab will be informal: discuss your results from section 4. Staple the **Instructor Verification** sheet to the end of your lab report.

The report will due during the week of **10-Nov** 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 inputs such as complex exponentials. 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( )` to implement filters and `freqz( )` to obtain the filter's frequency response.[1] As a result, you should learn how to characterize a filter by knowing how it reacts to different frequency components in the input.

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 $b_2 = \frac{1}{3}$.

[1]If you are working at home and do not have the function `freqz.m,` there is a substitute available called `freekz.m`. You can get it from the EE-2200 WebCT page.

CD-ROM
firfilt.m

MATLAB has a built-in function for implementing the operation in (1); namely, the function `filter( )`, 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 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.

2.1 Frequency Response of FIR Filters

The output or *response* of a filter for a complex sinusoid input, $e^{j\hat{\omega}n}$, depends on the frequency, $\hat{\omega}$. Often a filter is described solely by how it affects different frequencies—this is called the *frequency response*.

For example, the frequency response of the two-point averaging filter

$$y[n] = \tfrac{1}{2}x[n] + \tfrac{1}{2}x[n-1]$$

can be found by using a general complex exponential as an input and observing the output or response.

$$\begin{aligned} x[n] &= Ae^{j\hat{\omega}n + \phi} && (3)\\ y[n] &= \tfrac{1}{2}Ae^{(j\hat{\omega}n + \phi)} + \tfrac{1}{2}Ae^{(j\hat{\omega}(n-1) + \phi)} && (4)\\ &= Ae^{(j\hat{\omega}n + \phi)}\tfrac{1}{2}\left\{1 + e^{(-j\hat{\omega})}\right\} && (5) \end{aligned}$$

In (5) there are two terms, the original input, and a term which is a function of $\hat{\omega}$. This second term is the frequency response and it is commonly denoted by $H(e^{j\hat{\omega}})$.[2]

$$H(e^{j\hat{\omega}}) = \mathcal{H}(\hat{\omega}) = \tfrac{1}{2}\left\{1 + e^{(-j\hat{\omega})}\right\} \tag{6}$$

[2]The notation $H(e^{j\hat{\omega}})$ is used in place of $\mathcal{H}(\hat{\omega})$ for the frequency response because we will eventually connect this notation with the z-transform, $H(z)$, in Chapter 7.

Once the frequency response, $H(e^{j\hat{\omega}})$, has been determined, the effect of the filter on any complex exponential may be determined by evaluating $H(e^{j\hat{\omega}})$ at the corresponding frequency. The output signal $y[n]$, will be a complex exponential whose complex amplitude has a constant magnitude and phase. The phase describes the phase shift of the complex sinusoid and the magnitude describes the gain applied to the complex sinusoid.

The frequency response of a general FIR linear time-invariant system is

$$H(e^{j\hat{\omega}}) = \mathcal{H}(\hat{\omega}) = \sum_{k=0}^{M} b_k e^{-j\hat{\omega}k} \tag{7}$$

MATLAB has a built-in function for computing the frequency response of a discrete-time LTI system.[3] It is called `freqz( )`. The following MATLAB statements show how to use `freqz` to compute and plot the magnitude (absolute value) of the frequency response of a two-point averaging system as a function of $\hat{\omega}$ in the range $-\pi \leq \hat{\omega} \leq \pi$:

```
bb = [1, -1];          %-- Filter Coefficients
ww = -pi:(pi/100):pi;    %-- omega hat
H = freqz(bb, 1, ww);    %<--freekz.m is an alternative
subplot(2,1,1);
plot(ww, abs(H))
subplot(2,1,2);
plot(ww, angle(H))
xlabel('Normalized Radian Frequency')
```

We will always use capital `H` for the frequency response. For FIR filters of the form of (1), the second argument of `freqz( _, 1, _ )` must always be equal to `1`. The frequency vector `ww` must cover an interval of length 2π for $\hat{\omega}$, and its spacing must be fine enough to give a smooth curve for $H(e^{j\hat{\omega}})$.

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 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]If you are working at home and do not have the function `freq.m`, there is a substitute available called `freekz.m`. You can get it from the EE-2200 WebCT page.

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 $x[n]$ 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;
subplot(2,1,1);
stem(nn,x1(nn))
subplot(2,1,2);
stem(nn,y1(nn))
xlabel('Time Index (n)')
```

This code assumes that the output from `firfilt` is called `y1`. Try the plot with `first` and `last` chosen to include a range equal to the length of the input signal.

(c) 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 40. In this case, a `stem` format is not used; instead, the "envelope" or "outline" of the discrete-time signals is shown.

Instructor Verification (separate page)

3.3 Properties of Discrete–Time Filters

The frequency responses of discrete-time filters are *always* periodic with period equal to 2π. Explain why this is the case by stating a definition of the frequency response and then considering two input sinusoids whose frequencies are $\hat{\omega}$ and $\hat{\omega} + 2\pi$.

$$x_1[n] = e^{j\hat{\omega}n} \qquad \text{versus} \qquad x_2[n] = e^{j(\hat{\omega}+2\pi)n}$$

Prove mathematically that the outputs from (1) will be identical.

3.4 Frequency Response of the Three-Point Averager

In Chapter 6 we examined filters that average input samples over a certain interval. These filters are called "running average" filters or "averagers" and they have the following form:

$$y[n] = \frac{1}{M+1}\sum_{k=0}^{M} x[n-k] \tag{8}$$

(a) Use Euler's formula and complex number manipulations to show that the frequency response for the 3-point running average operator is given by:

$$H(e^{j\hat{\omega}}) = \mathcal{H}(\hat{\omega}) = \frac{2\cos\hat{\omega}+1}{3}\,e^{-j\hat{\omega}} \tag{9}$$

(b) Implement (9) directly in MATLAB. Use a vector that includes 400 samples between $-\pi$ and π for $\hat{\omega}$. Since the frequency response is a complex-valued quantity, use `abs()` and `angle()` to extract the magnitude and phase of the frequency response for plotting. Plotting the real and imaginary parts of $H(e^{j\hat{\omega}})$ is not very informative.

(c) In this part, use `freqz.m` in MATLAB to compute $H(e^{j\hat{\omega}})$ numerically and plot its magnitude and phase versus $\hat{\omega}$. Write the appropriate MATLAB code to plot both the magnitude and phase of the frequency response. The filter coefficient vector for the 3-point averager is defined via:

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

Note: the function `freqz(bb,1,ww)` evaluates the frequency response for all frequencies in the vector `ww`. It uses the summation in (7), not the formula in (9). The filter coefficients are defined in the assignment to vector `bb`. How do your results compare with part (b)?

Instructor Verification (separate page)

4 Lab: FIR Filters

In the following sections we will study how a filter affects sinusoidal inputs, and begin to understand the performance of the filter as a function of the input frequency. You will see the following:

1. that filters of the form of (1) can modify the amplitude and phase of a cosine wave, but they do not modify the frequency
2. that for a sum of cosine waves, the system modifies each component independently;
3. that filters can completely remove one or more components of a sum of cosine waves.

4.1 Filtering Cosine Waves

We will be interested in filtering discrete-time sinusoids of the form

$$x[n] = A\cos(\hat{\omega}n+\phi) \qquad \text{for } n = 0,1,2,\ldots,L-1 \tag{10}$$

The *discrete-time frequency* for a discrete-time cosine wave, $\hat{\omega}$, always satisfies $0 \le \hat{\omega} \le \pi$. If the discrete-time sinusoid is produced by sampling a continuous-time cosine, the discrete-time frequency is $\hat{\omega} = \omega T_s = 2\pi f/f_s$, as discussed in Chapter 4 on *Sampling*.

4.2 First Difference Filter

Generate $L = 50$ samples of a discrete-time cosine wave with $A = 7$, $\phi = \pi/3$ and $\hat{\omega} = 0.125\pi$. Store this signal in the vector `xx`, so it can also be used in succeeding parts. Now use `firfilt( )` to implement the following FIR filter on the signal `xx`.

$$y[n] = 5x[n] - 5x[n-1] \tag{11}$$

This is called a *first-difference* filter, but with a gain of five. In MATLAB you must define the vector `bb` needed in `firfilt`.

(a) Note that $y[n]$ and $x[n]$ are not the same length. What is the length of the filtered signal $y[n]$, and *why* is it that length? (If you need a hint refer to Section 2.)

(b) Plot the first 50 samples of both waveforms $x[n]$ and $y[n]$ on the same figure, using `subplot`. Use the `stem` function to make a discrete-time signal plot, but label the x-axis to run over the range $0 \leq n \leq 49$.

(c) Verify the amplitude and phase of $x[n]$ directly from its plot in the time domain.

(d) From the plot, observe that with the exception of the first sample $y[0]$, the sequence $y[n]$ seems to be a scaled and shifted cosine wave of the *same* frequency as the input. Explain why the first sample is different from the others.

(e) Determine the frequency, amplitude and phase of $y[n]$ directly from the plot. Ignore the first output point, $y[0]$.

(f) Characterize the filter performance at the input frequency by computing the relative amplitude and phase, i.e., the ratio of output to input amplitudes and the difference of output and input phases.

(g) In order to compare your measured results to the theory developed in Chapter 6 for this system, derive the mathematical expression for the output when the input signal is a complex exponential $x[n] = e^{j\hat{\omega}n}$. From this formula determine how much the amplitude and phase should change for $x[n]$ which has a frequency of $\hat{\omega} = 0.125\pi$.

4.3 Linearity (Superposition) of the Filter

(a) Now multiply the vector `xx` from Section 4.2 by two to get `xa=2*xx`. Generate the signal `ya` by filtering `xa` with the first difference filter given by (11).

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

$$x_b[n] = 8\cos(0.25\pi n)$$

and then filter it through the first difference operator to get $y_b[n]$. Then repeat the relative amplitude and phase measurements as before. In this case the measurement of phase might be a bit tricky because there are only a few samples per period. Record how the amplitude, phase, and frequency of the output `yb` change compared to the input.

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

4.4 Time-Invariance of the Filter

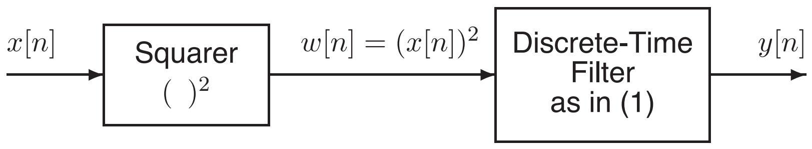
Now time-shift the input vector `xx` by 3 time units to make a new vector `xs`:

$$x_s[n] = 7\cos(0.125\pi(n-3)+\pi/3) \qquad \text{for } n = 0, 1, 2, 3, \ldots, L-1$$

and then filter $x_s[n]$ through the first difference operator to get $y_s[n]$. Compare `ys` to `yy`, the output when the input is `xx`. Find a shift of `yy` (in number of samples) so that it lines up perfectly with `ys`.

4.5 Cascading Two Systems

More complicated systems are often made up from simple building blocks. In the system below a *non-linear* system (squaring) is cascaded with an FIR filter:



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

$$\begin{aligned} w[n] &= (x[n])^2 && \text{(SQUARER)} \\ y[n] &= w[n] - w[n-1] && \text{(FIRST DIFFERENCE)} \end{aligned}$$

Implement this system using MATLAB. Use as input the same vector `xx` as in Section 4.2. In MATLAB the elements of a vector `xx` can be squared by either the statement `xx.*xx`, or `xx.^2`.

(b) Explain why the squaring system does not satisy the definition of linearity.
Note that the "squarer" is nonlinear and it is therefore possible for the frequency spectrum of $w[n]$ to contain frequency components not present in $x[n]$.

(c) Plot all three waveforms $x[n]$, $w[n]$, and $y[n]$ on the same figure with `subplot`.

(d) Make a *hand-drawn sketch* of the spectrum of the three signals $\{x[n],\ w[n],\ y[n]\}$. Do *not* use `specgram`.

(e) Observe the time-domain output, $w[n]$, of the "squarer." Can you "see" the additional frequencies introduced by the squaring operation?

(f) Use the linearity results to explain what happens as the signal $w[n]$ then passes through the first-difference filter.
This requires that you track each frequency component through separately.

(g) Now replace the first-difference filter in the above figure with the second-order FIR filter:

$$y_2[n] = w[n] - 2\cos(0.25\pi)w[n-1] + w[n-2] \tag{12}$$

Implement the squaring and filtering to produce a new output $y_2[n]$. Define the filter coefficients and calculate/plot the frequency response of the 2nd-order FIR. Determine which frequencies are present in the output signal. Explain how this new filter is able to remove a frequency component by calculating $y_2[n]$ when $w[n] = e^{j0.25\pi n}$ in (12). In addition, sketch the spectrum of $y_2[n]$.

4.6 Filtering the Speech Waveform

CD-ROM
lab6dat.mat

A sampled speech waveform is stored in the variable `x2` in the file `lab6dat.mat`. Two sets of filter coefficients are stored in `h1` and `h2` (i.e., these are the "b_ks" for two different filters). Use `length` to find out how many filter coefficients are contained in `h1` and `h2`. In this experiment we will test these filters on the speech signal.

(a) Filter the speech signal with filter `h1` using the statements

```
        y1 = firfilt(h1, x2);
        inout(x2, y1, 3000, 1000, 3)
```

CD-ROM
inout.m

The M-file `inout( )` will 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.

Compare the input and output signals. Is the output "rougher" or "smoother" than the input signal?

(b) Use `freqz( )` to plot the frequency response of the system defined by the coefficients `h1` as a function of frequency for $-\pi \leq \hat{\omega} \leq \pi$. Why is `h1` called a "lowpass filter"?

(c) Since the vector of filter coefficients is rather long, a `stem` plot of `h1` can be informative to show the nature of the filter coefficients. Use the `stem` plot to find a point of symmetry in the coefficients. How is this stem plot related to the impluse response $h[n]$ of the FIR filter?

(d) Filter the speech signal with filter `h2` and plot the input and output using the statements

```
        y2 = firfilt(h2,x2);
        inout(x2, y2, 3000, 1000, 3)
```

Compare the input and output and state whether the output is "rougher" or "smoother" than the input signal.

(e) Use `freqz( )` to plot the frequency response of the system defined by the coefficients `h2` as a function of frequency for $-\pi \leq \hat{\omega} \leq \pi$. Why is `h2` called a "highpass filter"?

(f) Make a `stem` plot of `h2` and look for a symmetry in the coefficients.

(g) Make an "A-B-C" listening comparison by executing the following statements:

```
                soundsc([x2; y1; y2], 8000)
```

Comment on your perception of the filtered outputs versus the original speech signal.

(h) What do you expect to hear when you execute the following statement? Why?

```
                soundsc([x2; (y1+y2)], 8000)
```

Was your expectation confirmed? If you add the two frequency responses together

$$H_1(e^{j\hat{\omega}}) + H_2(e^{j\hat{\omega}})$$

what would you expect the answer to be? Use linearity to explain your answer.

Lab #6
EE-2200
Fall-1998
INSTRUCTOR VERIFICATION PAGE

Staple this page to the end of your Lab Report.

Name: ______________________________ Date of Lab: ________

Part 3.2(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 the output over the same range of time indices if the filter were a 7-point averager.

Verified:________________ Date/Time:________

Part 3.2(e) Use `inout.m` to display the input signal `x1` along with the output signal from a 3-point averager:

Verified:________________ Date/Time:________

Part 3.4 Find the frequency response of a 3-point averager and make magnitude and phase plots with `freqz.m`:

Verified:________________ Date/Time:________