DSP First

Laboratory Exercise #5

FIR Filtering of Sinusoidal Waveforms

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.

1 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 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` function implements a wider class of filters than just the FIR case. Technically speaking, the `firfilt` function implements an operation called *convolution*, although we will not be concerned with the meaning of that terminology right now. 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 `x` is a vector containing a cosine function. In general, the vector `b` contains the filter coefficients $\{b_k\}$ needed in (1). These are loaded into the `b` 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(y) = length(x)+length(b)-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.

1.1 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 $H(e^{j\hat{\omega}})$.[1]

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

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 result will be a complex number whose phase describes the phase shift of the complex sinusoid and whose magnitude describes the gain applied to the complex sinusoid.

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

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

MATLAB has a built-in function for computing the frequency response of a discrete-time LTI system. 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 $-\pi \le \hat{\omega} \le \pi$:

```
bb = [1, -1];          %-- Filter Coefficients
ww = -pi:(pi/100):pi;
H = freqz(bb, 1, ww);
plot(omega, abs(H))
```

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`.

[1] 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.

2 Warm-up

The instructor verification sheet is included at the end of this lab.

2.1 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) Show that the frequency response for the three-point running average operator is given by:

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

(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) The following MATLAB statements will compute $H(e^{j\hat{\omega}})$ numerically and plot its magnitude and phase versus $\hat{\omega}$.

```
bb = 1/3*ones(1,3);
ww = -pi:(pi/200):pi;
H = freqz( bb, 1, ww );
subplot(2,1,1)
plot( ww, abs(H) ) %<-- Magnitude
subplot(2,1,2)
plot( ww, angle(H) ) %<-- Phase
xlabel('NORMALIZED FREQUENCY')
```

The function `freqz` 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)

3 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.

3.1 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 $0 \leq \hat{\omega} \leq \pi$. 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*.

3.2 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 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, and *why* is it that length? (If you need a hint refer to Section 1.)

(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 $\hat{\omega} = 0.125\pi$.

3.3 Linearity of the Filter

(a) Now multiply the vector `xx` from Section 3.2 by two to get `xa=2*xx`. Generate the signal `ya` by filtering `xa` with the first difference filter given by (11). Repeat the relative amplitude and phase measurements described in the previous section.

(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.

3.4 Time-Invariance of the Filter

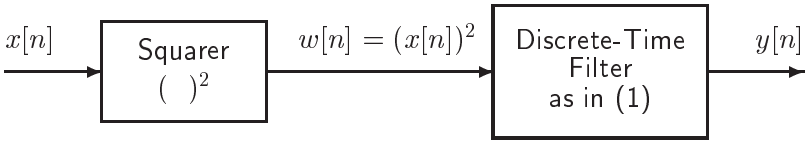
Now time-shift the input vector `xx` by 3 time units to get

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

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`.

3.5 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

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

Implement this system using MATLAB. Use as input the same vector `xx` as in Section 3.2. In MATLAB the elements of a vector `xx` can be squared by either the statement `xx.*xx`, or `xx.^2`.

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

(c) Make a *sketch*[2] of the spectrum of the three signals $\{x[n],\ w[n],\ y[n]\}$. Recall that the "squarer" is nonlinear and it is therefore possible for the frequency spectrum of $w[n]$ to contain frequency components not present in $x[n]$.

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

[2]This should, as the term implies, be done by hand. Do *not* use `specgram`.

(e) Use the linearity results to explain what happens as the signal $w[n]$ then passes through the first-difference filter.
Hint: track each frequency component through separately.

(f) 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 $y_2[n]$. 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 $y_2[n]$.

Lab 5
Instructor Verification Sheet

Staple this page to the end of your Lab Report.

Name: _____ Date: _____

Part 2.1 Find the frequency response of a 3-point averager:

Verified: _____