Elec4621 S1 2018
Lab1

Elias Aboutanios

This lab is essentially a practice Matlab exercise as well as illustrating the concepts of signal representation, sampling and convolution.

When you are done, you are required to upload your code and results (observations) as a zipped file to the submission box on Moodle. You are to name the file using your student number.

1. Consider the signal $x(t) = cos(2\pi ft)$, where $f = 2000$Hz.

   (a) What is the minimum sampling frequency, f_{min} that we should use for this signal?

   (b) Using Matlab, plot 5 periods of the continuous signal $x(t)$. We simulate a continuous signal by taking a very fine sampling time. That is set $T_s = 10^{-6}$.

   (c) Let the actual sampling frequency be $f_s = 1.5f_{min}$. Plot the sampled signal (the samples) on top of the "continuous" signal. How many samples per period of the signal do we get? What is the minimum frequency cosine (or sine) that can fit those points? **Hint:** Use the function stem to plot the sampled signal.

   (d) Now let the sampling frequency be $f_s = 0.75f_{min}$. Plot the sampled signal (the samples) on top of the "continuous" signal. How many samples per period of the signal do we get? What is the minimum frequency cosine (or sine) that can fit those points?

   (e) Calculate and plot the spectra of the original and two sampled signals for $-12,000 \leq f \leq 12,000$. Use a step of 1Hz for the plot. **Hint:** You should calculate the DFT of the signal. The FFT function does not give you all the frequencies you want.

2. A discrete FIR filter has impulse response $h[n]$ given by

$$h[n] = \begin{cases} -0.0625 & n = 0 \\ 0.25 & n = 1 \\ 0.625 & n = 2 \\ 0.25 & n = 3 \\ -0.0625 & n = 4 \end{cases}$$

(a) Write Matlab code to apply this filter to the sequence

$$x[n] = \{1, 2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1\}, \quad n = 0, 1, \ldots, 12$$

Your code should directly implement the convolution equation

$$\mathbf{y} = \sum_n x[n]\,\mathbf{h}_n \tag{1}$$

That is, you should start by creating an output vector $\mathbf{y}$, large enough to accommodate the filtered output, setting all entries to 0. Your code should then add $x[n]\,\mathbf{h}_n$ into $\mathbf{y}$, for each n in turn. Do not use any special Matlab functions to do this; the idea is to learn what the equation means by implementing it directly.

(b) Equation (1) and the code you wrote in part (a) to implement it, represent an *input-based perspective* for filtering. Specifically, each input sample serves as a separate excitation source for the filter, and the output is the sum of the filter outputs from each separate excitation source. In this second part, you should implement exactly the same filter, adopting an *output-based perspective* instead. In this case, each individual output sample is written as

$$y[n] = \left\langle \mathbf{x}, \tilde{\mathbf{h}}_n \right\rangle$$

where $\tilde{\mathbf{h}} \equiv h[-n]$. As for part (a), your code should directly implement this equation, finding the output at each location (or at any given location), by taking the dot product between the translated, time-reversed impulse response, and the input signal.

(c) Write Matlab code to evaluate $\hat{h}(\omega)$ over a range of frequencies from $-\pi$ to π. Again, you should do this directly using primitive functions. Inspect the magnitude and phase of $\hat{h}(\omega)$. Do either of these possess any special properties? Is there a sense in which the filter can be said to shift (or delay) the input sequence? If so, by how much? Is this property evident from the filtered output, $y[n]$?

(d) Repeat parts (a) and (c) for the impulse response shown below

$$h[n] = \begin{cases} -0.1 & n = 0 \\ 0.6 & n = 1 \\ 0.6 & n = 2 \\ -0.1 & n = 3 \end{cases}$$

(e) Repeat parts (a) and (c) for the impulse response shown below, paying particular attention to the phase response

$$h[n] = \begin{cases} 0.2 & n = 0 \\ 0.5 & n = 1 \\ 0.2 & n = 2 \\ 0.1 & n = 3 \end{cases}$$