BME1473

Problem Set 1

Instructions: *For each problem, in addition to answering the questions, please include your Matlab code and a short (2-3 sentences) discussion of your results. Please use Matlab's "Publish" functionality to create your report, and submit the resulting PDF through Quercus.*

1 [Adapted from 1.6 in Semmlow]. Write a MATLAB problem to test the following quantization equation through simulation:

$$q = \frac{V_{MAX}}{2^b - 1} \text{ V} \quad (1)$$

Where q is the quantization step size, V_{MAX} is the range of the analog-to-digital converter (ADC) and b is the number of bits used for quantification. Generate a 4-Hz, 1000-point sine wave assuming a sample interval of T_s= 0.002. Use `quantization.m` to digitize it using 4-, 8-, 12-, and 16-bit ADC. Then, subtract the original signal from the quantized signal to find the error signal. The amplitude of the error signal should be equal to the quantization level, *q* in Equation 1. Use MATLAB's `max` and `min` functions to find this amplitude and compare it with the value predicted by Equation 1. Put this code in a for-loop and repeat the evaluations for the four different bit levels requested. Be sure to display the results to at least four decimal places to make an accurate comparison.

2 [Adapted from 1.8 in Semmlow]. Write a MATLAB program to generate 1 s of a 5-Hz sine wave in a 1000-point array. Plot the sine wave. Simulate the sampling of this waveform at 7 Hz by plotting a point (such as an "*") at intervals of $T_s = 1/f_s = 1/7$ s. Aliasing predicts that these sampled points should fall on a 2-Hz (7–5 Hz) sine wave. Can you find a 2-Hz sine wave that includes all seven points? [Hint: The sine wave could be phase shifted by 180 degree].

3 [Adapted from 2.8 in Semmlow]. Load the signal found in file `data_c1.mat`. First demonstrate that this signal is nonstationary. One straightforward method to demonstrate that this is the case is to segment the data and evaluate the mean and variance of the segments. If they are the same for all segments, the signal is probably stationary. If these measures change segment-to-segment, the signal is clearly nonstationary. Next, apply MATLAB's detrend operator and evaluate the detrended signal by again calculating the means and variances of signal segments. Is the modified signal now stationary?

4 [Adapted from 3.14 in Semmlow] This problem demonstrates aliasing. Generate a 512-point waveform consisting of two sinusoids at 200 and 400 Hz. Assume f_s = 1 kHz. Generate another waveform containing frequencies at 200 and 900 Hz. Take the Fourier transform of both waveforms and plot the magnitude of the spectrum up to $f_s/2$. Plot the two spectra superimposed, but plot the second spectrum as dashed and in a different color to highlight the additional peak due to aliasing at 100 Hz. [Hint: To

generate the sine waves, first construct a time vector, `t`, then generate the signal using `x = sin(2*pi*f1*t) + sin(2*pi*f2*t)` where `f1` = 200 for both signals, whereas `f2` = 400 for one waveform and `f2` = 900 for the other.]

5 [Adapted from 2.28 in Semmlow]. Use MATLAB and cross-correlation to find the *phase shift* between 10-Hz sinusoids found as variables `x` and `y` in file `sines1.mat`. Assume a sample frequency of 2 kHz. Plot the cross-correlation function and find the lag at which the maximum (or minimum) correlation occurs.

6 [Adapted from 2.32 in Semmlow]. In file `eeg_data.mat`, the variable `eeg` contains an EEG recording (f_s = 50 Hz). Use cross-correlation to find the maximum correlation of this signal and a number of sinusoids ranging in frequency between 0.25 and 25 Hz. Plot the EEG signal and the maximum correlation as a function of the sinusoidal frequency. Where are the highest correlations found? Repeat this problem correlating the signal with cosine waves.

7 [Adapted from 2.36 in Semmlow]. Construct a 512-point Gaussian noise array, then filter it by averaging segments of three consecutive samples. In other words, construct a new array in which every point is the average of the preceding three points in the noise array: $y[n] = x[n]/3 + x[n - 1]/3 + x[n - 2]/3$. You could write the code to do this, but an easier way is to convolve the original data with a function consisting of three equal coefficients having a value of 1/3, in MATLAB: `h(n) = [1/3 1/3 1/3]`. Construct, plot, and compare the autocorrelation of the original and filtered waveform. Limit the *x*-axis to ±20 lags to emphasize the difference. Note that while the original Gaussian data were uncorrelated point to point, the filtering process imposed some correlation on the signal.

8 [Adapted from 3.18 in Semmlow]. Use `sig_noise` to generate a 256-point waveform consisting of a 300-Hz sine wave with an SNR of −12 dB (`x = sig_noise(300,-12,256);`). Calculate and plot the power spectrum using two different approaches. First, use the direct approach: take the FT and square the magnitude function. In the second approach, take the FT of the autocorrelation function. Calculate the autocorrelation function using `xcorr`, then take the absolute value of the `fft` of the autocorrelation function. You should only use the second half of the autocorrelation function (those values corresponding to positive lags). Plot the PS derived from both techniques. The scales will be different because of different normalizations.

9 [Adapted from 4.3 in Semmlow]. This program illustrates one of the problems that can occur with ensemble averaging: the lack of a fixed and stable reference signal. The file `ver_problem2.mat` contains three variables: `actual_ver`, which is the noise-free VER, `ver`, which consists of 100 noise records recorded with a fixed reference signal, and `ver1`, which consists of 100 records recorded with a reference signal that varies randomly by ±150 ms. Construct ensemble averages from `ver` and `ver1` and plot separately along with the noise-free record. T_s = 0.005 s. What is the difference in the two averages?

10 [Adapted from 4.20 in Semmlow]. This problem compares causal and noncausal FIR filter implementation. Generate the filter coefficients of a 65th-order rectangular window filter with a cutoff frequency of 40 Hz. Apply a Blackman–Harris window to the filter (using the fir1 and `blackmanharris`

MATLAB commands). Then apply the filter to the noisy sawtooth wave, `x`, in file `sawth.mat`. This waveform was sampled at f_s = 1000 Hz. Implement the filter in two ways. Use the causal `filter` routine and noncausal `conv` with the `'same'` option. (Without this option, `conv` is like `filter` except that it produces extra points.) Plot the two waveforms along with the original, superimposed for comparison. Note the obvious differences. Also note that while the filter removes much of the noise, it also reduces the sharpness of the transitions. [Note: Section 4.4.1 from the Semmlow textbook (3rd edition) provides a discussion of the effects of the application of a window to the filter. Alternatively, the 1st edition of the book is available as a web resource through the U of T library, and contains a similar explanation on pages 94-97].

11 [Adapted from 4.21 in Semmlow]. Given the advantage of a noncausal filter with regard to the time shift shown in the previous problem, why not use noncausal filters routinely? This problem shows the downsides of noncausal FIR filtering. Generate the filter coefficients of a 33rd-order rectangular window filter with a cutoff frequency of 100 Hz, assuming f_s = 1 kHz. Use a Blackman–Harris window on the truncated filter coefficients. Generate an impulse function consisting of a 1 followed by 255 zeros. Now apply the filter to the impulse function in two ways: causally using the MATLAB `filter` routine, and noncausally using the conv routine with the `'same'` option. (The latter generates a noncausal filter since it performs symmetrical convolution.) Plot the two time responses separately, limiting the *x* axis to 0–0.05 s to better visualize the responses. Then take the Fourier transform of each output and plot the magnitude and phase. (For a change, you can plot the phase in radians.) Use the MATLAB `unwrap` routine on the phase data before plotting. Note the strange spectrum produced by the noncausal filter (i.e., `conv` with the `'same'` option). This is because the noncausal filter has truncated the initial portion of the impulse response. To confirm this, rerun the program using an impulse that is delayed by 10 sample intervals (i.e., `impulse = [zeros(1,10) 1 zeros(l,245)];`). Note that the magnitude spectra of the two filters are now the same, although the phase curves are different due to the different delays produced by the two filters. The phase spectrum of the noncausal filter shows reduced phase shift with frequency as would be expected. This problem demonstrates that noncausal filters can create artifact with the initial portion of an input signal because of the way it compensates for the time shift of causal filters.

12 [Adapted from 4.25 in Semmlow]. This problem is similar to the previous problem in that it illustrates problems with noncausal filtering, except that an IIR filter is used and the routine `filtfilt` is used to implement the noncausal filter. Generate the filter coefficients of an eighth-order Butterworth filter with a cutoff frequency of 100 Hz assuming f_s = 1 kHz. Generate an impulse function consisting of a 1 followed by 255 zeros. Now apply the filter to the impulse function using both the MATLAB `filter` routine and the `filtfilt` routine. The latter generates a noncausal filter. Plot the two time responses separately, limiting the *x* axis to 0–0.05 s to better visualize the responses. Then take the Fourier transform of each output and plot the magnitude and phase. Use the MATLAB unwrap routine on the phase data before plotting. Note the differences in the magnitude spectra. The noncausal filter (i.e., `filtfilt`) has ripple in the passband. Again, this is because the noncausal filter has truncated the initial portion of the impulse response. To confirm this, rerun the program using an impulse that is delayed by 20 sample intervals (i.e., `impulse = [zeros(1, 20) 1 zeros(1, 235)];`). Note that

the magnitude spectra of the two filters are now the same. The phase spectrum of the noncausal filter shows reduced phase shift with frequency, as would be expected. However, even after changing the delay, the noncausal implementation has a small amount of ripple in the passband of the magnitude spectrum.

13 [Adapted from 5.1 in Semmlow] In this problem you will apply the AR spectral method to analyze a 1000-point waveform consisting of four sinusoids buried in 12 dB of noise (i.e., SNR = −12 dB). The sinusoidal frequencies should be 100, 240, 280, and 400 Hz. Use the basic Yule–Walker equations to find the spectrum of the signal. First use a model order of 17, then explore higher-order models to better resolve the four frequencies. Since this is a fairly complicated spectrum, you may need to use a very high order for high spectral resolution. Compare the AR power spectrum with that obtained using classical Fourier transform methods. An easy way to calculate the Fourier transfer function is to use pwelch with a window that is the same length as the signal. The routine uses the “direct method” to determine the power spectrum but also provides the frequency vector and eliminates the redundant points.