UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Department of Electrical and Computer Engineering

ECE 498MH Signal and Image Analysis

Lab 1

Fall 2014

Assigned: Thursday, September 4, 2013 Due: Friday, September 12, 2013

Reading: Jason Starck, *All About Circuits* Chapter 7: Mixed-Frequency AC Signals, `http://www.allaboutcircuits.com/vol_2/chpt_7/`

Problem 1.0.1

(a) Say "ah," or any other sustained vowel of your choice. Record your voice using the matlab command `wavrecord`. Use a sampling frequency of at least 8000 samples/second, but no more than 16000 samples/second; write down the sampling frequency you use. **Save your waveform, and send it to me along with your `runlab.m` function.**

(b) Your `runlab.m` function should start with a call to `wavread` that reads in your waveform file. Then, in figure 1, use `plot` to plot the waveform. On your own (not necessarily as part of `runlab.m`), use the `zoom` function to zoom in to an interesting part of the waveform. Notice how the mouth and throat (the vocal tract) rings like a bell each time the vocal folds close; the period between vocal fold closures is called the *pitch period*, and the frequencies at which the ringing occurs called *formant frequencies*. Choose ten consecutive pitch periods, starting with the zero-crossing just before a peak. Excise this waveform snippet using a command like `x=y(n_start:n_end)`, where `n_start` and `n_end` are the starting and ending sample numbers of your ten-pitch-period snippet.

(c) Use `t=[0:(N-1)]/Fs;` to compute the time, in seconds, of each sample in your waveform snippet, where `Fs` is the sampling frequency, and `N=length(x);`. In figure 2, use `plot(t,x);` to show the waveform snippet, with a time axis labeled in seconds.

(d) Use `X=fft(x);` to compute the Discrete Fourier Transform of your signal. The DFT is actually just a scaled version of the Fourier series; the `fft` operation finds the Fourier series coefficients. Use `MagX=abs(X);` to compute its magnitude, and `PhaX=unwrap(angle(X));` to compute its phase. In figure 3, use `subplot` to create two subplots, one above the other. Plot the magnitude in the top plot, and the phase in the bottom plot, as a function of the Fourier coefficient number `k=[0:(length(X)-1)];`. Be sure that the first sample in your plot starts at Fourier coefficient number 0, not Fourier coefficient number 1!

(e) Use `zoom` to zoom in on your magnitude plot. Notice, first of all, that the number of frequency samples is equal to the number of time-domain samples. Notice, second of all, that the upper frequencies (above $N/2$) have the same magnitude as the lower frequencies, but the opposite phase; this is because matlab is actually using the upper half of the vector `X` to store the negative-frequency Fourier series coefficients. Finally, notice that only one sample out of every ten has large magnitude; this is because you used `fft` to compute the Fourier series based on *ten* periods, insted of just one period. Fix this problem: find the true Fourier series coefficients of your vowel sound by extracting every tenth sample from the `MagX` and `PhaX` vectors, using commands something like `M=MagX(1:10:length(MagX));` and `Phi=PhaX(1:10:length(PhaX))`. Use the `stem` command to plot just the first six coefficients of `M` (`stem([0:5],M(1:6));`), and the first six coefficients of `Phi`. Create this plot in figure 4.

(f) In figure 5, create a figure with two subplots. In the top plot, plot `x` as a function of `t`. In the second plot, plot `y` as a function of `t`, where `y` is the one-cosine approximation of `x`:

$$y(t) = M_1 \cos\left(\frac{2\pi t}{T_0} + \Phi_1\right)$$

Be sure to realize that, because matlab counts vector elements starting with 1 instead of 0, the first Fourier series coefficient is given by `M1=M(2)` and `Phi1=Phi(2)`.

(g) In figure 6, repeat the previous step, but this time, use a two-cosine approximation:

$$y(t) = M_1 \cos\left(\frac{2\pi t}{T_0} + \Phi_1\right) + M_2 \cos\left(\frac{4\pi t}{T_0} + \Phi_2\right)$$

(h) In figure 7, repeat the previous step, but this time, use a ten-cosine approximation:

$$y(t) = \sum_{k=1}^{10} M_k \cos\left(\frac{2\pi k t}{T_0} + \Phi_k\right)$$

(i) In figure 8, repeat the previous step, but this time, use a twenty-cosine approximation:

$$y(t) = \sum_{k=1}^{20} M_k \cos\left(\frac{2\pi k t}{T_0} + \Phi_k\right)$$

(j) Use the `soundsc` command (with the correct sampling frequency!) to play back your waveform snippet. Play back, also, the one-cosine, two-cosine, ten-cosine, and twenty-cosine approximations.