Laboratory # 3

Basic Spectral Analysis: The Fourier Series

One of the most powerful tools for analyzing signals and systems is the *spectrum*. In this lab, we'll be investigating spectral ideas via the *Fourier Series*. The basic idea of the Fourier Series is that we can represent any periodic signal simply by summing sinusoids with harmonically related frequencies and the right amplitudes and phases. NOTE: In this lab, if a sampling frequency is left unspecified, use the default value of 8192 Hz.

1. [24] The first thing you'll need to do is write a utility function that sums sinusoids. We'll call this function `sumcos.m`. You should download the template file, `sumcos.m`, from the course webpage.

   (a) The function, `sumcos`, takes four input parameters:

   - `ampl`: a vector of complex amplitudes, where the length of the vector is the number of sinusoids to create and sum, and where for each complex number in the vector, the magnitude provides the amplitude of the cosine and the phase provides the cosine's phase. You'll need to break these amplitudes into their magnitudes and phases in your code.
   - `frqs`: a vector giving the frequencies in Hz of the sinusoids to be summed. This vector *must* have the same length as `ampl`.
   - `t_max`: the length of the time axis in seconds. This parameter is optional and should default to 1 second.
   - `fs`: the sampling frequency in Hz. This parameter is optional and should default to 8192 Hz.

   (b) The function returns two output parameters:

   - `out`: the signal produced by summing the sinusoids.
   - `t`: the time axis used to produce the sinusoids. So that a signal with `t_max` = 1 and `fs` = 8192 Hz will only have 8192 samples, you should use the command `t=0:1/fs:(t_max-1/fs)` to produce the time axis.

   (c) This function is set up to simplify the process of creating and summing sinusoids. Here are some examples of how it works. Execute these examples and include the plots of the first two (using `subplot`) to verify the operation of your code.

   - `sumcos(1,1)` produces one period of a cosine wave.
   - `sumcos(-8/pi^2./(1:2:50).^2,1:2:50,3)` produces three periods of a triangle wave (see page 64 in the text).
   - `sumcos(1,300,2,8192)` produces two seconds of a cosine at an audible frequency (300 Hz). You can `soundsc` this signal to hear the sinusoid.

(d) To make sure your function is correct, execute the command `sumcos(1:4,1:4,1,8)`. Your function should return the following vector:

```
[ 10  -5.4142  2  -2.5858  2  -2.5858  2  -5.4142 ]
```

Now execute the command `sumcos([3 6 9 10],1:4,1,8)`. What vector does your function produce?

2. [10] Test your version of `sumcos` by generating an approximate square wave using the coefficients listed on page 64 of the text. (Note: there is an error in equation 3.4.5. Following the conventions in the text , X_k is only defined for positive k.) Only include the first 50 harmonics, and use a 1 Hz fundamental frequency with a time scale ranging over 2 seconds.

   (a) Use `stem` and `subplot` to plot the magnitude and phase of the coefficients you used to generate this signal in a single figure.

   (b) In another figure, plot your square wave.

3. [24] Download the function `fourier_analysis.m` and the file `lab_3.mat` from the course webpage. (You can type `help fourier_analysis` to get the function's syntax). Load `lab3.mat` and consider the three signals, `signal1`, `signal2`, and `signal3` (the other signals will be used in a later problem). For each of these three signals,

   (a) Use `fourier_analysis.m` to perform a Fourier series analysis out to the 30th harmonic.

   (b) Using `subplot`, plot the magnitude and phase of the resulting Fourier coefficients in a single figure.

   (c) Use these coefficients with `sumcos` to resynthesize the signal over one period, and plot both the original and the resynthesized signal on the same plot using `hold`.

   What is different about the original signals and your resynthesis? Why might this happen?

4. [12] Examine Figure 3.1. Match the Fourier coefficient magnitudes (on the left) with the associated signals (on the right).

5. [10] Suppose that we want to add a time-delay to a signal that is composed of sinusoids. To do this, we need to shift each sinusoid by the same amount in time. This amounts to a frequency-dependent phase shift. We can rewrite the general form for a sinusoid, $\cos(2\pi ft+\phi)$, as $\cos(2\pi f(t+\tau))$, where τ is the desired time shift. This indicates that we need to add a phase shift which is linearly proportional to frequency to achieve a constant time shift. That is, the phase for a harmonic at frequency f must be

$$\phi_f = 2\pi\tau f$$

   Using this formula and `sumcos`, synthesize a triangular wave with a time delay equal to one fourth of the signal's period. Plot both the original triangular wave and time delayed signal.

6. [20] When several people talk at the same time, most listeners have the remarkable ability to *tune in* to one speaker, while ignoring the others. This is sometimes called the *cocktail party effect*. In this part of the lab you will create a system that does something similar. Specifically, it will accept as input a signal that is the sum of two vowel sounds, and it will output just one of the vowel sounds.

   A vowel sound is an approximately periodic signal whose fundamental frequency is the vibration rate of the vocal chords. In this lab the system input will be the sum of *synthetic* vowels that are perfectly periodic. One of the vowel sounds will have

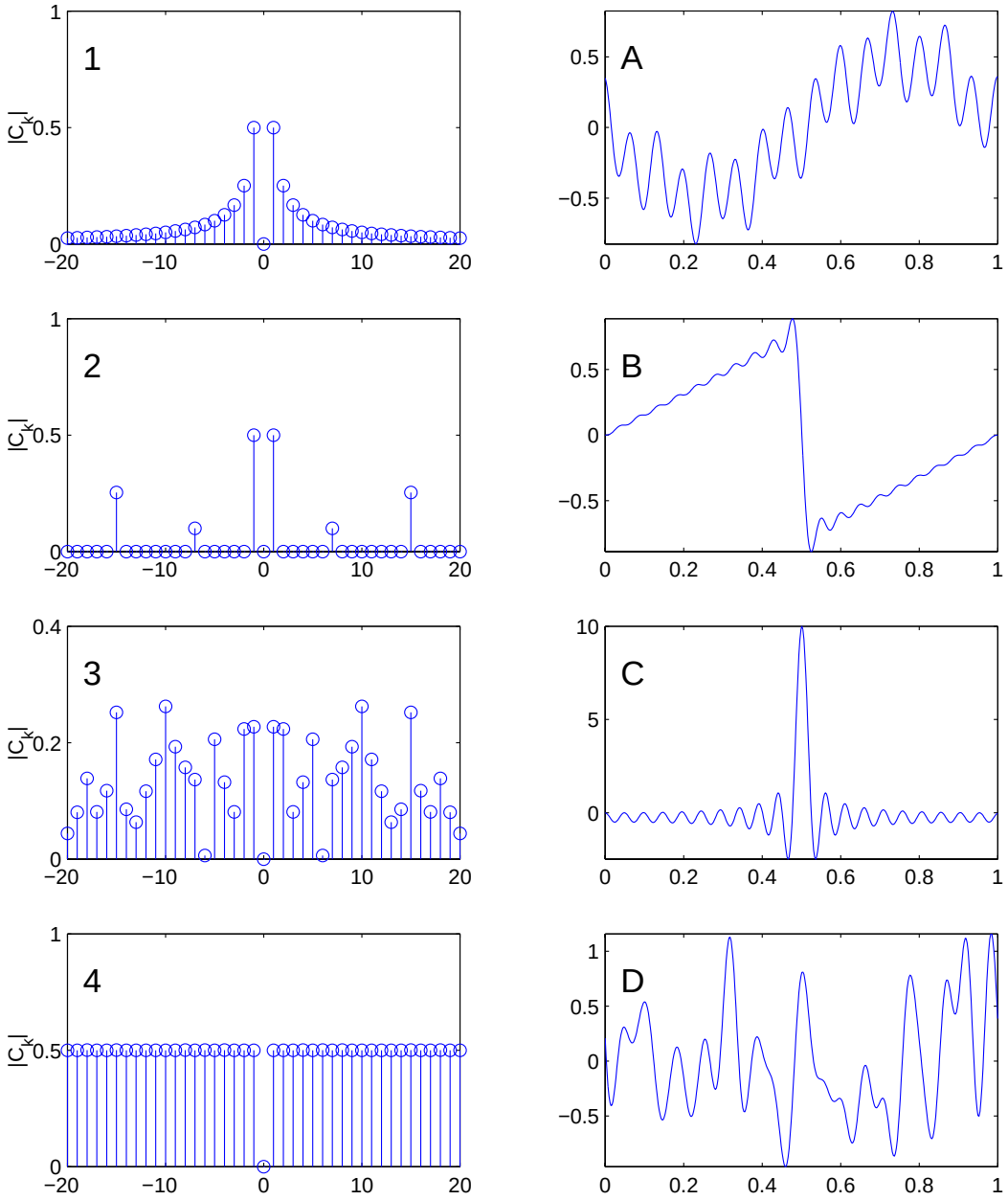

Figure 3.1: Fourier coefficient magnitudes (left) and signals (right) to be matched for problem 4.

fundamental frequency 200 Hz and the other 300 Hz. The goal of the system is to output the vowel sound with fundamental frequency 200 Hz. The system you create will *extract* or *resynthesize* the 200 Hz vowel from the sum.

Your system will perform Fourier series analysis on an input signal. This relies on the fact that the spectrum of the 200 Hz vowel contains frequencies that are multiples of 200 Hz, while the spectrum of the 300 Hz vowel contains frequencies that are multiples of 300 Hz. Your system's output will be the periodic signal reconstructed from only the Fourier coefficients corresponding to frequencies that are multiples of 200 Hz. That is, it only keeps coefficients corresponding to 200 Hz, 400 Hz, 600 Hz, 800 Hz, ... Notice that the components at 600 Hz, 1200 Hz, ... are affected by both vowels. Thus, the output of the system will not be a perfect rendition of the 200 Hz vowel – rather, it will be affected somewhat by the presence of the 300 Hz vowel.

The MATLAB workspace file `lab_3.mat` contains ten vowel signals – five vowels at both 200 Hz and 300 Hz. The vowels are "ah," "ae," "ee," "oh," and "oo" (corresponding to the vowels in the words "bat," "bait," "beet," "boat," and "boot," respectively). Each of these signals is a two-second waveform with a sampling frequency, fs, of 20000 Hz. We will be working with the signals `oh_200` and `ee_300`.

(a) Calculate a signal called `vowel_sum` by adding `oh_200` and `ee_300`. Listen to `oh_200`, `ee_300` and `vowel_sum` using the `soundsc` command. (To listen to the signal `oh_200`, for instance, use the command `soundsc(oh_200,20000)`). Describe what you hear. Can you hear the cocktail party effect? (Note: this will not work properly on a UNIX system; you should make sure you are using a Windows system or a Mac. On most CAEN systems, you will need to plug headphones into the back of the computer to hear the sound output. If you get an error, the computer you are working at is not properly set up for audio.)

(b) What is the fundamental period of

i. the 200 Hz signal?

ii. the 300 Hz signal?

iii. `vowel_sum`?

(c) Write a MATLAB function called `extract200` with one input parameter, `input`, and one output parameter, `output`. `input` is a vector presumed to contain the sum of two synthetic vowels – one with fundamental frequency 200 Hz and one with fundamental frequency 300 Hz. `output` is a reconstruction of the 200 Hz signal. In this function, you should do the following:

i. Compute the Fourier coefficients of `input` using `fourier_analysis.m`. (Recall that `fourier_analysis` requires you to pass it *one period* of a signal.)

ii. Use `sumcos` to resynthesize an approximation to the 200 Hz vowel and store it in the vector `output`. (Hint: You'll need to use an appropriate subset of the Fourier coefficients you computed from `input`.)

iii. In one figure, plot roughly one period of both the input signal (`input`) and your reconstructed signal (`output`) in separate subplots.

iv. In another figure, use `stem` to plot the magnitude of the Fourier coefficients for both `input` and `output` in separate subplots.

v. In a third figure, use `stem` to plot the magnitude of the Fourier coefficients *in decibels* for both `input` and `output`, with the two plots in separate subplots.

(d) Execute your function and include the plots that it generates.

(e) Use soundsc to listen to the resynthesized 200 Hz vowel that it outputs. Does it sound like the original?

(f) What features do you notice in your decibel plots that are not evident in the linear amplitude plots?