Fourier Analysis with Applications — Project 1

The purpose of this lab is to begin computing the Fourier transform in MATLAB and visualize frequency components. You will be exploring connections between exponential and cosine representations of a Fourier series and how this can be used to extract the primary harmonics from a real world signal.

The instructions below contain directions for both the code you need to write as well as some exploratory questions throughout. *Answer all questions in italics as comments within the code.* If you would like additional information on how a given command works, you can get the documentation by typing `doc <command>` into the command window.

What You Need to Turn In: Submit both of the following two versions of your work:

1. The `.m` file created as described in the instructions contained in this document.
2. The PDF of your code and corresponding output generated using the "Publish" feature in the top menu of the MATLAB editor.

Due in Canvas, 11:59pm, Friday, September 26

In MATLAB, the Fourier transform is computed using an algorithm called the "fast Fourier transform" (FFT). Arguably one of the most important algorithms ever developed, this computes the *discrete Fourier transform* (DFT). For a data set for a signal $f(t)$ with N data points, $\{(t_0, f(t_0)), (t_1, f(t_1)), \ldots, (t_{N-1}, f(t_{N-1}))\}$, the *discrete Fourier series* computed at the m^{th} time sample t_m is given by

$$f(t_m) = \frac{1}{N} \sum_{k=0}^{N-1} \widehat{f}(k) e^{2\pi i k m/N}, \quad m = 0, 1, \ldots, N-1,$$

where the *discrete Fourier transform* is given by

$$\widehat{f}(k) = \sum_{m=0}^{N-1} f(t_m) e^{-2\pi i k m/N}, \quad k = 0, 1, \ldots, N-1.$$

Since this is the discrete setting, we are tracking indices and not just input/output values:

- N is the total number of data points – we need to normalize by the number of data points in order to get the frequencies as well as the appropriate magnitudes for reconstruction in the Fourier series.
- m represents the indices for the time data.
- k represents the indices for the frequencies.

While we will get into the nitty-gritty of discrete setting later in the semester, for now we can work with the idea of the DFT having computations and properties similar to the continuous Fourier transform. Thus, we can use the FFT in MATLAB to help us visualize the frequency components of a signal.

1 Analyzing Synthetic Signals

We will start by looking at the frequency components of truly periodic signals.

1. Create a cell in MATLAB titled `Setting up Signal1`. Download the data file `signal1.txt`, add it to your working directory, and load the file into MATLAB. The first column represents an input vector and the second represents the function values – split the data into two vectors accordingly called `t_1` and `y_1`.

2. Plot the data set, using appropriate labels.

3. Working in the discrete setting requires us to keep careful track of information such as time step size and anticipated frequency values. Add the following lines of code to this cell:

```
% parameters for working with DFT
dT = t_1(2)-t_1(1);  % time between samples
fs = 1/dT;  % sample rate
[N0, ~] = size(t_1);  % total number of samples
T_total = dT*N0;  % Duration of signal
f_res = 1/T_total;  % frequency domain resolution
freqs = 0:f_res:fs-f_res;  % set of all possible frequency values for this signal
```

   (a) *How many data points are we working with?*

   (b) *What is the relationship between the frequency spacing and the duration of the signal? What implications might this have for analyzing real world data?*

4. Create a new cell titled `Analyzing Frequencies in Signal 1`. Apply the FFT to `y_1` using the command `fft`. MATLAB has numerous possible settings you can use here – the only input you need for our purposes is `y_1`, **don't include any additional parameters.**

5. Since MATLAB uses the exponential representation, the coefficients produced by the FFT are complex numbers. This means they can be expressed in the form $\widehat{f}(k) = |\widehat{f}(k)|e^{i\theta_k}$, where θ_k is called the *phase* of the coefficient. Create two arrays with their magnitudes and phases.

6. Plot the magnitudes against the frequencies; use appropriate labels for the graph. *What do you notice? Do the values of the magnitudes make sense for the values in the original signal?*

7. Notice that the DFT definition contains no negative frequencies – nonetheless, with our real-valued signal we are seeing symmetry appear in the magnitudes of the frequencies, with the line of symmetry occurring at $N/2$, since N is even in our case. This phenomenon is analogous to $\widehat{f}(n) = \overline{\widehat{f}(-n)}$ that we saw in the continuous setting,

so certain frequencies are in some sense being duplicated. Moreover, the DFT requires magnitudes to be normalized by the number of data points used. Working only with the first half of the frequencies, do the following:

(a) Double the the magnitudes that appeared twice due to the symmetry.

(b) Divide all magnitudes by the size of the data set.

(c) Plot a graph of the magnitudes of the frequencies in the range [0, 20]. This time, display data using a stem plot; clearly indicate in your figure labels that this represents rescaled magnitudes.

(d) *What frequencies are in the spectrum of Signal1? What are their magnitudes?*

8. In the homework, we saw that for a function $f(t)$ with period 1, we have two ways to write its Fourier series:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{2\pi i n t} = \sum_{n=-\infty}^{\infty} |c_n| \cos(2\pi n t + \phi_n).$$

For visualizing harmonics, working with the cosine representation is really nice. We can actually take this representation a step further by applying some properties of complex conjugation and the symmetry of cosine to obtain

$$\sum_{n=-\infty}^{\infty} |c_n| \cos(2\pi n t + \phi_n) = c_0 + 2\sum_{k=1}^{\infty} |c_n| \cos(2\pi n t + \phi_n).$$

(Computations contained in the appendix). In the setting of the DFT, when N is even (as in our case) this comes out to

$$f(t_m) = \frac{c_0}{N} + \frac{2}{N} \sum_{n=1}^{N/2-1} |c_n| \cos(2\pi \omega_n t + \phi_n),$$

where ω_n and ϕ_n are the frequencies and corresponding phase shifts. Use this latter representation to:

(a) Write cosine expressions for the harmonics in the signal with nonzero magnitudes;

(b) Graph the individual harmonics present in the signal on the same plot – label the plot appropriately and include a key indicating which graph corresponds to which harmonic;

(c) Plot the graph of their sum to reconstruct the original signal and check your work. *What function generated the signal?*

2 How This Turns Out In the Real World...

We can apply the exact same process we just saw to real-world data. Let's try it out on the pseudo-periodic data we saw in the photoplethysmogram from the last lab.

1. Create a cell in MATLAB titled `Setting up PPG`. Download the data file `photoplethysmogram.txt`, add it to your working directory, and load the file into MATLAB. Split the data into two arrays for time and blood volume.

2. Once again, we have to keep careful track of the time step size and anticipated frequency values. Following the example in the first section, set up corresponding parameters for this data set.

3. Create a new cell titled `Analyzing a PPG` and repeat the steps in Items 4, 5, and 7a-c for the PPG signal, where your plot from 7c should contain only the frequencies in the range 0Hz - 8Hz.

   (a) *In comparison to the plots of the magnitudes we saw with our synthetic signal, how do these magnitudes appear?*

   (b) *What do you think causes this?*

   (c) *What challenges do you think this might produce for data collection and analysis in other studies?*

4. Due to the appearance of the spectrum, we will pick out the harmonics manually. Identify the first 5 highest peaks using the graph you generated. *What frequencies do they correspond to? With the exception of the second frequency, what relationship do you notice between them?* Use this pattern to write an array containing the first 6 dominant frequencies (don't forget the odd one out $\omega_{0.4} = 0.4Hz$).

5. Ignoring momentarily $\omega_{0.4} = 0.4Hz$, the first nonzero frequency, $\omega_1 = 1.3Hz$ (called the *fundamental frequency*) seems to control all the other frequencies present. *What physical interpretation could we give this?*

6. Look a the graph with the magnitudes of the frequencies including the range 0Hz - 70Hz. After a lot of negligible frequencies, you'll see a small spike toward the right side. *What frequency does that correspond to? What aspect of the data collection device might operate at that frequency?*

7. Let's generate some graphics summarizing what we've found. Create a figure that shows plots with each of the following (with appropriate labels):

   (a) The original PPG signal;

   (b) The magnitudes of first 70 frequencies with markers indicating the first 7 dominant frequencies $\omega_0, \omega_{0.4}, \omega_1, \ldots$, and the additional high frequency we noted in #6;

(c) The phases of the first 70 frequencies with markers plotted on the phases corresponding to the first 7 dominant frequencies;

(d) A single plot with the first 6 dominant harmonics you found with a key indicating which graph corresponds to which harmonic;

(e) A plot with a reconstruction of the PPG using only the first 6 dominant harmonics overlaid with a graph of the original signal. *How good of a representation do you think this is? What kind of information may have been removed?*

A Appendix

As promised, we will show that

$$\sum_{n=-\infty}^{\infty} |c_n| \cos(2\pi nt + \phi_n) = c_0 + 2\sum_{k=1}^{\infty} |c_n| \cos(2\pi nt + \phi_n).$$

First, recall that for a real signal $f(t)$, we have $c_n = \overline{c_{-n}}$. Therefore, $|c_n| = |c_{-n}|$. Also, since $\overline{e^{i\theta}} = e^{-i\theta} = e^{i(-\theta)}$, we have that $\phi_{-n} = -\phi_n$ (recall that one interpretation of the conjugate is reflection over the y axis). Using this, we have

$$\begin{aligned}
\sum_{n=-\infty}^{\infty} |c_n| \cos(2\pi nt + \phi_n) &= c_0 + \sum_{n=-\infty}^{-1} |c_n| \cos(2\pi nt + \phi_n) + \sum_{n=1}^{\infty} |c_n| \cos(2\pi nt + \phi_n) \\
&= c_0 + \sum_{n=1}^{\infty} |c_{-n}| \cos(2\pi(-n)t + \phi_{-n}) + \sum_{n=1}^{\infty} |c_n| \cos(2\pi nt + \phi_n) \\
&= c_0 + \sum_{n=1}^{\infty} |c_n| \cos(-2\pi nt - \phi_n) + \sum_{n=1}^{\infty} |c_n| \cos(2\pi nt + \phi_n) \\
&= c_0 + 2\sum_{n=1}^{\infty} |c_n| \cos(2\pi nt + \phi_n).
\end{aligned}$$