University of Washington ECE Department
EE 242 Lab 3: Frequency Domain Representation of Signals

In this lab, we will learn how to build periodic signals from component sinusoids and how to transform signals from the time domain to the frequency domain. The concepts we'll focus on include: implementation of the Fourier Series synthesis equation, using a discrete implementation of the Fourier Transform (DFT) with a digitized signal, and understanding the relationship between the discrete DFT index k and frequency ω for both the original continuous signal $x(t)$.

Lab 3 Turn-in Checklist

- Four pre-lab exercises, completed individually and uploaded to canvas before the first lab (Optional)
- PDF version of Lab report (Jupyter notebook), completed and submitted as a team
- PDF version of Jupyter notebook for individual assignment to be uploaded to canvas

Pre-lab (Optional)

Read the Lab 3 Background document, then complete the following exercises:

1. A periodic signal has Fourier series coefficients $a_0 = 0.5$ and $a_k = 1/(jk\pi)$ for all other k and period $T = 50ms$. Give an equation for synthesizing this signal with sinusoids.
2. Write the Python code for synthesizing a periodic signal with inputs: length T (in seconds), Fourier coefficients $\{a_0, \ldots, a_N\}$, fundamental frequency f_0, and sampling rate F_s (in Hz).
3. In your lab, you will work with music and other natural signals. If the sampling rate is $F_s = 11025$Hz, what sample corresponds to a start time of 200ms?
4. Efficient implementations of the discrete Fourier transform use a length that is a power of 2. What is the length would you use for a signal of length 100ms and a smpline rate of $F_s = 11025$Hz? What frequency resolution would this give?

Lab Assigments

This lab has 4 assignments. Each should be given a separate code cell in your Notebook and each should be associated with a markdown cell with subtitle and discussion. As in previous labs, your notebook should start with a markdown title and overview cell, which should be followed by an import cell that has the import statements for all assignments. In this lab, you will need to use the import commands that you have used in earlier labs except for those related to working with images. As always, you should add comments to your code for clarity.

Assignment 1: Generating simple periodic signals

In the first assignment, you will develop an understanding of how some periodic signals are easier to approximate than others with a truncated Fourier Series. In this lab, we'll work with real signals and use the synthesis equation:

$$x(t) = a_0 + \sum_{k=1}^{N} 2|a_k| \cos(k\omega_0 t + \angle a_k)$$

In lecture, you saw that you get ripples at transition points in approximating a square wave (Gibbs phenomenon). This happens for any signals with sharp edges. This assignment will involve approximating two signals (a sawtooth and a triangle wave) that have the same fundamental frequency (20Hz).

1. Write a function for generating a real-valued periodic time signal given the Fourier series coefficients $[a_0\ a_1 \cdots a_N]$, the sampling frequency, and the fundamental frequency. You may choose to have complex input coefficients or have separate magnitude and phase vectors for describing a_k.

2. Define variables for the sampling frequency (8kHz) and the fundamental frequency (20Hz). Using this sampling frequency, create a time vector for a length of 200ms.

3. The sawtooth signal has coefficients

$$a_0 = 0.5, a_k = 1/(j2k\pi)$$

   Using the function from part (a), create three approximations of this signal with $N = 2, 5, 20$ and plot together in a 3×1 comparison.

4. A triangle signal has coefficients

$$a_0 = 0.5, a_k = \frac{2\sin(k\pi/2)}{j(k\pi)^2} e^{-j2k\pi/2}$$

   Create three approximations of this signal with $N = 2, 5, 20$ and plot together in a 3×1 comparison.

Report discussion: You should have noticed that the second signal converges more quickly. Discuss the two reasons for this.

Assignment 2: Synthesizing a musical note

In this assignment, you will use the same synthesis equations to try to approximate a single note from a horn, which has the frequency characteristics illustrated below. Download the file *horn11short.wav* from the

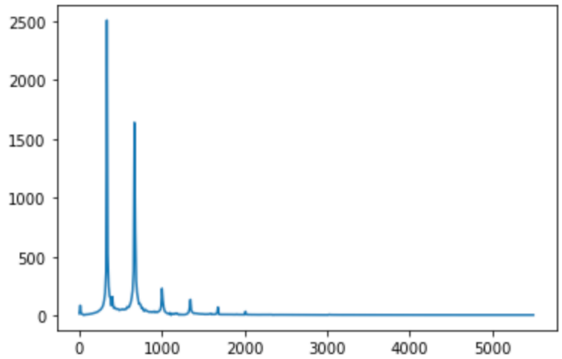

Figure 1: Frequency components of a note played by a horn.

assignment page to compare your synthesized version to the original.

1. Read in the horn signal, and use the sampling rate f_s that you read in to create a time vector of length 100ms. Define the fundamental frequency to be $f_0 = 335$Hz. Create a signal that is a sinusoid at that frequency, and save it as a wav file.

2. Create a vector (or two) to characterize a_k using:

$$\begin{aligned} |a_k| &: [2688,\ 1900,\ 316,\ 178,\ 78,\ 38] \\ \angle a_k &: [-1.73,\ -1.45,\ 2.36,\ 2.30,\ -2.30,\ 1.13] \end{aligned}$$

   assuming $a_0 = 0$ and the first element of the vectors correspond to a_1. Use the function you created in part 1 to synthesize a signal, with f_s and f_0 above, and save it as a wav file.

3. Plot the 100ms section of the original file starting at 200ms with a plot of the synthesized signal in a 2×1 plot.

4. Play the original file, the single tone, and the 6-tone approximation in series.

Report discussion: The approximation does not sound quite like the original signal and the plot should look pretty different. The difference in sound is in part due to multiple factors, including the truncated approximation, imperfect estimate of the parameters, and the fact that the original signal is not perfectly periodic. Try adjusting some parameters and determine what you think is the main source of distortion.

Assignment 3: Analyzing frequency content of a signal

For this assignment, you will use a discrete Fourier transform (specifically, the Python implementation of an FFT) to analyze the frequency content of the 100ms segment of the horn signal from assignment 2. Because this is a periodic signal, the frequency content will have spikes, but because it is a discrete-time signal, they will have finite height. You will experiment with different FFT sizes and different plotting options. The description below assumes that you import numpy as np.

1. Use the np.fft.fft function to compute the FFT for the 100 ms horn signal, with an fft size of nfft=1024, which you can call **xhf**. Recall that the result of the FFT will be a vector that spans frequencies $[0, f_s]$. If this is a real-valued signal, then the first half of the FFT matters: [0,nfft/2]. In order to get positive and negative frequencies, you need to use the np.fft.fftshift function to get **xhf2**. Create two different plots of the magnitude of result using **(np.abs(.))** in a 2x1 view: one with positive and negative frequencies and one with just positive frequencies. Be sure to scale according to time signal window length. Label the frequency axis in terms of Hz by creating a vector **freq** that scales the FFT index by $f_s/nfft$. The one-sided version should look like the picture above. The two-sided version should be an even function.

2. It is often the case that frequency content is plotted on a log scale. Again using a 2x1 view, plot the one-sided (positive frequency) magnitude using both linear and log scale.

3. Changing the size of the FFT will change the frequency resolution, but it also changes the shape of the result a bit. Just as we saw with Gibbs phenomenon where increasing the number of Fourier series coefficients gave a high frequency ringing at sharp edges, increasing the FFT window will give a "ringing" effect for sharp peaks in frequency. To see this effect, compute the FFT using nfft=2048 and plot the log magnitude compared to nfft=1024, in both cases just using positive frequencies. (The effect is easier to see when plotting magnitude on a log scale.)

Report discussion: In assignment 2, we used specific cosine frequencies to approximate the horn note, assuming the signal is periodic so the harmonics have non-zero energy. The FFT results show a different picture, and the synthesized version is easily distinguished from the original. Discuss reasons for these differences.

Assignment 4: Comparing frequency content of a signal

Many interesting time signals have changing frequency content. Music is one example, since different notes have different fundamental frequency. Speech is another example: we distinguish different vowels and consonants based on their frequency content. In this assignment, you will use the FFT to compare the frequency content of two different speech sounds in a sentence. We'll use 30ms windows, where the frequency content is relatively stable.

1. Download the signal "bluenose3.wav", and read in the file. Plot the full waveform, using the sampling frequency to correctly label the time access. Play the file.

2. Extract the samples corresponding to times [0.75,0.78]. (This corresponds to the "oo" sound in the word "grew.") Using a 2x1 plot, plot the time waveform (labeling the time axis with the specified time region) and the magnitude of the frequency response (positive frequencies only, labeling the frequency axis in Hz).

3. Repeat the exercise above using the samples corresponding to times [2.565,2.595]. (This corresponds to the "s" sound.)

Report discussion: State what size FFT you used and explain your choice. Comment on the differences between the time and frequency plots for the two segments and the auditory differences.

Team Report

When you have tested and cleaned up all your code (remember, you should only submit code for the Assignments, each in their own cell), download the .ipynb notebook file and then upload the file on Canvas (in PDF format). Remember, only one submission per team! Make sure that your file is titled Lab3-XYZ.ipynb, where XYZ are the initials of the lab partners.

Inidividual Assignment

For this assignment, choose and download one of the extra music signals provided. Extract a 50ms time window from different parts of the file that correspond to different notes (in which case they would have different frequency characteristics). Plot the log magnitude of the Fourier transforms of these two time segments. Identify the fundamental frequency for each case, and determine what musical note it most closely corresponds to. (A table of notes and their frequencies is at: https://pages.mtu.edu/ suits/notefreqs.html.)