

Spectral Analysis of Musical Sound

Objectives: The purpose of this assignment is to learn how to use the discrete Fourier transformation to analyze musical sounds. In particular, you will analyze two tones from different instruments, and use your analysis to determine the frequency of the note being played in each case. You will examine the Fourier spectra generated to see what gives each instrument its distinctive sound, or timbre.

Logistics: All homework assignments consist of writing programs, one for each part of a problem, and providing the output obtained when the program is run. Organize your explanations and code in order in a **Jupyter notebook**, and then execute the code and download the entire notebook as an html file. Upload both the '.ipynb' and the '.html' files under the Assignment on Moodle, two files per homework assignment. **Please write your name and the date at the top.**

The different parts of the problem set will be graded as indicated on the assignment sheet. Every program you turn in should start with a **comment or docstring** that lists **what the problem is**. Each part of your program should also have short comments describing how the program works.

Materials: In the file folder associated with this assignment, there are two data files: *clarinet.txt* and *saxophone.txt* - these represent the text versions of waveforms of audio signals from these two instruments. The corresponding audio files, titled *clarinet.wav* and *saxophone.wav*, are also provided for you to download and listen to. Both waveforms were recorded at the industry-standard rate of 44,100 samples per second (44,100 Hz) and both instruments were playing the same note when the recordings were made.

Part 0: (5 points) Listen to both .wav files. Do they sound different to you? Describe what you hear, in particular what sounds similar, and what sounds different about them.

Part 1 (60 points) Write a program that will load the waveform, plot it, and calculate its discrete Fourier transform on each of the audio files.

- A. (20 pts) Load the data from the files *clarinet.txt* and *saxophone.txt*. Make professional-quality plots of both waveforms in time. Label your axes and title the graphs. Use the subplot command with two rows and 2 columns, leave space for the remaining graphs (part C). Have these waveforms occupy the first column of your subplots.
- B. (20 pts) Generate the discrete Fourier transform of both data sets (you can use the function `numpy.fft.rfft()`). You should note that the transformed array is only half the length of the original time sequence array. Explain why that is in a markdown cell. (Hint: it has to do with the Nyquist frequency.)
- C. (10 pts) Make professional-quality plots of the magnitudes of the Fourier coefficients. Plot these alongside the corresponding waveforms from part A in the second subplots column.
- D. (10 pts) In a separate pair of plots, plot the real and imaginary parts of the Fourier Transform of each waveform. How do these compare to the results of part C?

Part 2 (20 points) Answer these two questions in a docstring in your Jupyter notebook

- A. (10 pts) Use your spectra to determine what note is being played by the two instruments. (As an example, the musical note middle C has a frequency of 261 Hz). Explain your reasoning.
- B. (10 pts) Looking at your Fourier spectra for the two sounds from Part 1, what are the main differences between the spectrum of the clarinet vs the spectrum of the saxophone? How do you think these differences affect the way the two instruments sound?

Part 3: (15 points)

Read in the .wav files directly and plot the data. You will need to use the command:

```
rate, cwavedata = scipy.io.wavfile.read('clarinet.wav')  
rate, swavedata = scipy.io.wavfile.read('saxophone.wav')
```

Keep in mind that since the data is recorded in stereo, `cwavedata` will be a 2-column array, in which case you should plot only one of the two columns. You may have to restrict the range of the display in order to get a clear view of the waveforms.

What to hand in:

Remember to turn in your complete code for all three parts, with output and plots generated in each, as an .ipynb file and a html file.