**CSC 745 Advanced Multimedia Programming**

**Exercise: Generate and display narrowband and wideband spectrograms**

**Goals**

1. Apply the short term Fourier transform (STFT) to visualize sound over time
2. Gain an understanding of the use of windows in the FFT

**Background**

A *spectrogram* is a 3-dimensional plot of frequencies and frequency amplitudes across time. Spectrograms were developed before the advent of computers and high resolution graphics so the dimensions are not shown using x-y-z axes. Rather, time is shown on the x axis, frequency on the y axis, and amplitude is shown by darkness (if the image is grayscale) or by color. A *narrowband* spectrogram is created by using long blocks of data in order to get high frequency resolution (30-50 Hz). A *wideband* spectrogram is created by using short blocks of data in order to get good time resolution (2 – 4 ms, leading to a frequency resolution of 250-500 Hz). This exercise is based on one from an Engineering course at the University of Illinois: [Creating Spectrograms: Wideband and Narrowband](https://courses.engr.illinois.edu/ece590sip/sp2018/spectrograms1_wideband_narrowband.html), although I’ve streamlined it a bit and updated it by replacing deprecated code.

**Procedure**

1. Open *Ex\_spectrogram.py* in your IDE or text editor
2. Look at the imports and consider how they’re used
3. Write the read\_soundfile() function. Take a filename argument and return the data and the sample rate. I’ve used soundfile (imported as sf) because it allows you to specify the data type – float64, because we’ll be multiplying samples by floats. Use sf.read(fname, dtype = “float64”), returning data, sample\_rate.
4. Start the main() function
   1. Read the sound files into the variables p\_data, p\_fs (Pride and Prejudice) and w\_data, w\_fs (White Fang).
   2. Slice out a 0.6 sec segment of each file into the variables p\_wav and w\_wav; cut off 0.74 sec of silence at the beginning of Pride and Prejudice and 0.42 sec at the beginning of White Fang. In other words, start the slices at 0.74 sec and 0.42 sec, going for 0.6 sec.
   3. Use sounddevice (sd) to listen to the segments; comment this out after hearing it
5. Write the plot\_sound() function. Takes 2 sequences of samples as arguments, with 2 sample rates, and 2 titles. Use np.linspace(0, 1/fs \* len(wav), len(wav) to get a time axis. Use fig.tight\_layout().
6. Call plot\_sound() from main(); execute to see the waveforms
7. You will soon write enframe(), which will split the data into frames. Each frame will be multiplied by a window to smooth the edges of the frame. Write a function to display the window and plot various types: np.hamming, np.hanning, np.bartlett. Comment out main() and just call show\_window() with various window types.
8. Write the enframe() function, taking arguments of a sequence of samples, a number of samples to skip between frames (25-50% of a frame length is common; frames will overlap), and the number of samples in a frame.
   1. Create an empty list of frames
   2. Calculate the number of frames; 1 +\_int((len(x) – win\_len)/skip\_len), where x is the data
   3. Each frame will be np.copy(x[(index\*skip\_len) : (index \* skip\_len + win\_len)]) \* w. Calculate the frames and append each one to the list of frames
   4. Return the list of frames
9. Write test\_frames() to see if it looks like frames are correct – plot one frame from each sound; call from main() and execute, then comment out the call.
10. Write the stft() function to carry out a short-time Fourier transform on each frame. Return a list of stft frames and the frequencies covered by the stft.
11. Call test\_stft() to plot one stft frame.
12. Write the sgram function to calculate a spectrogram.
13. Write the show\_spect() function to display a spectrogram.
14. Calculate and display, for both files:
    1. Narrowband spectrograms using a window length of 35 ms and a skip length of 9 ms (0.035 and 0.009, multiplied by the corresponding sample rate)
    2. Wideband spectrograms using a window length of 4 ms and a skip length of 1 ms

**Deliverables**

Submit your .py file on Blackboard.