ECE 4750/6750 + CS 4501/6501: Digital Signal Processing **Due Tu 3/21**

Homework 2: Discrete Fourier Transform

Instructions: Submit a single Jupyter notebook (.ipynb) of your work to Collab by 11:59pm on the due date. All code should be written in Python. **Be sure to show all the work involved in deriving your answers! If you just give a final answer without explanation, you may not receive credit for that question.**

You may discuss the concepts with your classmates, but write up the answers entirely on your own. Do not look at another student's answers, do not use answers from the internet or other sources, and do not show your answers to anyone. **Cite any sources you used outside of the class material (webpages, etc.), and list any fellow students with whom you discussed the homework concepts.**

Discrete Fourier Transform

You will be using the fast Fourier transform (FFT) and the short-time Fourier transform (STFT) to analyze two audio signals. These are a clip of a person speaking (`winniethepooh.wav`)[1] and a clip of piano music (`bach.wav`)[2]. Download these files from the class webpage.

Note: Throughout this assignment, you should use the `numpy.fft` library to do forward and inverse FFTs. You can use the `rfft` and `irfft` functions, which only compute $L/2$ length Fourier transforms because of the symmetry of the DFT of real signals ($X[L-k] = \bar{X}[k]$).

1. The frequency band for voice used in telephones is approximately 300 to 3400 Hz.[3]. To simulate this, take the FFT of the voice audio signal and apply a bandpass filter in the voice frequency range. That is, set the $Y[k] = w[k]X[k]$, where $w[k]$ is set to one for frequencies in the 300–3400 Hz range and set to zero elsewhere. **Note:** you will need to first calculate the conversion of the frequency parameter k into Hz. Take the inverse FFT and listen to the resulting audio.

2. According to the same Wikipedia page, the fundamental frequency of a male speaking voice is between 85 to 155 Hz. Apply the same bandpass filter procedure as above, but with the frequency range of 85–310 Hz. (We have to double the upper limit of the frequency range to avoid aliasing. This is due to what is called the Nyquist-Shannon theorem, which we'll cover later in class!) Again, listen to the resulting audio. Is the speech still understandable when keeping only the fundamental frequencies, or were the harmonic overtones important contributions?

3. Now compute an FFT of the piano music signal. Plot the magnitude of the FFT, $|X[k]|$, as a function of frequency in Hz. What are the 3 most prominent frequencies in Hz (a.k.a., what is the position on the frequency axis of the three highest peaks in the plot)? Refer to the following table of musical note frequencies: https://pages.mtu.edu/~suits/notefreqs.html

[1]Chapter 5 of A. A. Milne's classic, *Winnie the Pooh*, read by Phil Chenevert, audio from LibriVox: https://librivox.org/winnie-the-pooh-by-a-a-milne/

[2]J. S. Bach's *Two Part Invention Number 13 in A Minor*

[3]https://en.wikipedia.org/wiki/Voice_frequency

What musical notes do these frequencies represent? **Hint: They may not be the notes you are expecting for a piece in the key of A minor!**

Short-Time Fourier Transform

4. Write a Python function to compute the short-time Fourier transform (STFT) of a signal. Your function should take the following inputs: a signal $x[n]$, a window, $w[n]$, and the hop length, h. It should output the STFT $X[k, m]$. Note: the length of your Fourier transform will be determined by the length of the $w[n]$ array.

5. Write a Python function implementing the overlap-add (OLA) method to synthesize a signal from its STFT. Your function should take the following inputs: an STFT, $X[k, m]$, a window, $w[n]$, and the hop length, h. It should output the synthesized signal, $x[n]$.

6. Do the following for both the voice and piano audio signals:

   (a) Apply your forward STFT with a Hann window of length 2048 and a hop of 1024 time samples. Now apply the OLA with a constant window of length 2048 and a hop of length 1024. Do you recover the original audio signals?

   (b) Repeat the process in part (a), but change the hop length to be the same as the window length, 2048. What changes do you notice in the resynthesized audio, and why did this happen?

   (c) Take the STFT with a Hann window of length 1024 and a hop of 256. Next, set the resulting phase for each element of X to zero, i.e., create an array of only the magnitudes: $Y[k, m] = |X[k, m]|$. Now apply the OLA with the same Hann window and hop. What did this do to the audio? Explain why removing the phase had this effect.

7. Write a Python function to compute the exact frequency of a signal using backward differences of the phase of the STFT. Your function should compute a vector of exact frequencies (one for each frequency bin in the FFT) at all times $1 \leq m < H$, in other words, the 2D array $\omega^*[k, m]$ from the lecture. Create a sinusoid signal of length $L = 256$ and frequency $\frac{\sqrt{2}\pi}{8}$. Test your function on this signal using a Hann window of length 32 and hop of 16. Plot a spectrogram (squared magnitude $|X[k, m]|^2$)as an image. You should see a band of energy surrounding the exact frequency. Verify that your exact frequency function returns a value close to true frequency of the sinusoid!

8. Write a Python function to perform pitch scaling. Test your algorithm on both the voice and piano audio with pitch scale factors of 2.0 and 0.5. Use 2048 Hann windows for both STFT and OLA and hops of 512. (**Not required**, but feel free to experiment with different windows, hops, and scale factors. If you want to play with transposing the key in which the piano piece is played, use a scale factor of $2^{\frac{k}{12}}$, where the integer k represents the number of half-notes up (positive) or down (negative) from the original key. For example, to change the key to a higher C minor, set $k = 3$, for a lower G minor, set $k = -2$.)

9. **For Grads Only (or Extra Credit for Undergrads).** Use your pitch scaling function from the last part to change the speed of the two audio signals, while keeping the original pitch intact. Try speeds of 0.5, 0.75, 1.5, and 2.0 times the original.