Homework 3: The z-Transform and STFT

Instructions: Submit a single Jupyter notebook (.ipynb) of your work to Canvas 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.**

Short-Time Fourier Transform

In this part, you will be implementing the short-time Fourier transform (STFT) and instantaneous frequency to change the pitch of audio files. Use the same audio clips as we had in HW 2: (`winniethepooh.wav`) and (`bach.wav`), available from the class webpage.

1. 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.

2. 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, $x[n]$.

3. 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 **square-root 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 square-root Hann window and hop. What did this do to the audio? Explain why removing the phase had this effect.

4. 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 **square-root 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!

5. 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 **square-root 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 $k = -2$.)

For Grads Only (or Extra Credit for Undergrads)

6. Use your pitch scaling function from the last part to change the speed of the two audio signals, while keeping the original pitch intact. Do this by scaling the pitch and also appropriately changing the sampling rate when playing back the audio. (**Hint:** Try doubling the sample rate when playing back the audio. What happens to the pitch? Now think about how you need to scale the frequency to return to the original pitch, but keeping the playback speed at double the rate.) Try speeds of 0.5, 0.75, 1.5, and 2.0 times the original.