Exam, Fall 2025

DSP Lab – ECE-GY 6183

1. Sign and submit the attached 'oath' with your exam.
2. You may use the course resources (lecture videos, demo programs, textbook, etc) and your own prior work for this course. You may consult the web for information (e.g., python documentation).
3. You may not ask for or receive help from anyone. You may not offer or give help to anyone. Do not ask the course assistants about the exam. Do not discuss the exam with anyone. Your submitted work must be 100% your own effort.
4. Do not use ChatGPT or AI/automated programming tools.
5. You may email the instructor if you have questions about the exam.
6. A violation may lead to failing the course, or receiving a lower grade.

There are two questions.

Question 1: Real-Time Frequency Scaling using the FFT

We have seen how to shift the frequencies of a signal using amplitude modulation (real and complex). This method *adds* the modulation frequency to all frequency components. If the modulation frequency is 100 Hz, then each frequency f is shifted to $f + 100$. (The frequency f is also shifted to $f - 100$ in the case of real AM.) However, this process fails to preserve the harmonic structure of the signal. Speech and audio signals often exhibit a harmonic structure, where the spectrum exhibits peaks at integer multiples of a base/fundamental frequency. (If f_1 is the fundamental frequency, then peaks in the spectrum tend to occur at $2f_1$, $3f_1$, etc.) For this reason, a more realistic approach to modify the frequencies of a signal is by *multiplying* the frequencies by a given scaling factor. This entails a *scaling* of the frequencies, in contrast to *additive* modification per amplitude modulation.

Use Python, PyAudio and TkInter to create an interactive graphical user interface (GUI) for real-time frequency scaling of audio. The method should read the signal in blocks and should process each block using the FFT. The FFT is used to compute the frequency spectrum of the block. A new frequency spectrum should be generated for the block by modifying the FFT coefficients so as to achieve frequency scaling with a scaling factor α. (As a result, it is normal that some high frequencies are lost as they will be above the Nyquist frequency (half the sampling rate) after frequency scaling.) The frequency scaling can be performed via interpolation of FFT coefficients. The inverse FFT can be used to calculate the output signal block. There will be various implementation details and choices for you to make.

The user should be able to vary the scaling factor α in real time by adjusting a slider. You can use the range $0.5 < \alpha < 2$ for the slider. The input signal can be read from a file with continuous looping until the user quits the program. You should aim to keep audible artifacts (e.g. clicking sounds, choppy sound, clipping, noise, etc.) and latency to negligible levels.

Your program for this question should *not* produce plots or animations.

Make a screen recording (as an MP4 video file) to demonstrate your program and its behavior. Your video should clearly illustrate the adjustment of the α scaling value slider and its effect on the output sound. The soundtrack of your video should be only the output audio; the input audio should not be on the video soundtrack. Write a description explaining your method and implementation choices.

To submit:

1. A screen recording (an mp4 file, not larger than 15 MB).
2. Your python program (.py file).
3. A writeup (pdf file) including your explanation of your method and comments.

Your submitted program files should run on their own (include additional audio files if needed) so that we can run your file.

Question 2: Real-Time Plotting

Add a real-time plotting of the frequency spectra (Fourier transforms) of the input and output signals. You can use the Animation library.

To submit:

1. A screen recording (an mp4 file, not larger than 15 MB).
2. Your python program (.py file).
3. A writeup (pdf file) of your comments and screenshots.