ECE 3551 – Microcomputer Systems Project

**Audio Scrambler – Using Randomized Overlapping**

By: Andrew Nguyen

**Problem Statement:**

This project involves creating a signal (audio) scrambler that converts and segments pieces of the audio (frame). The small segments are denoted within these frames (called windows) in which the windows are randomly reorganized to cause a scrambled audio effect. The scrambled audio must still be identifiable as a modified version of the original audio and must be saved for future reference.

**Literature Review:**

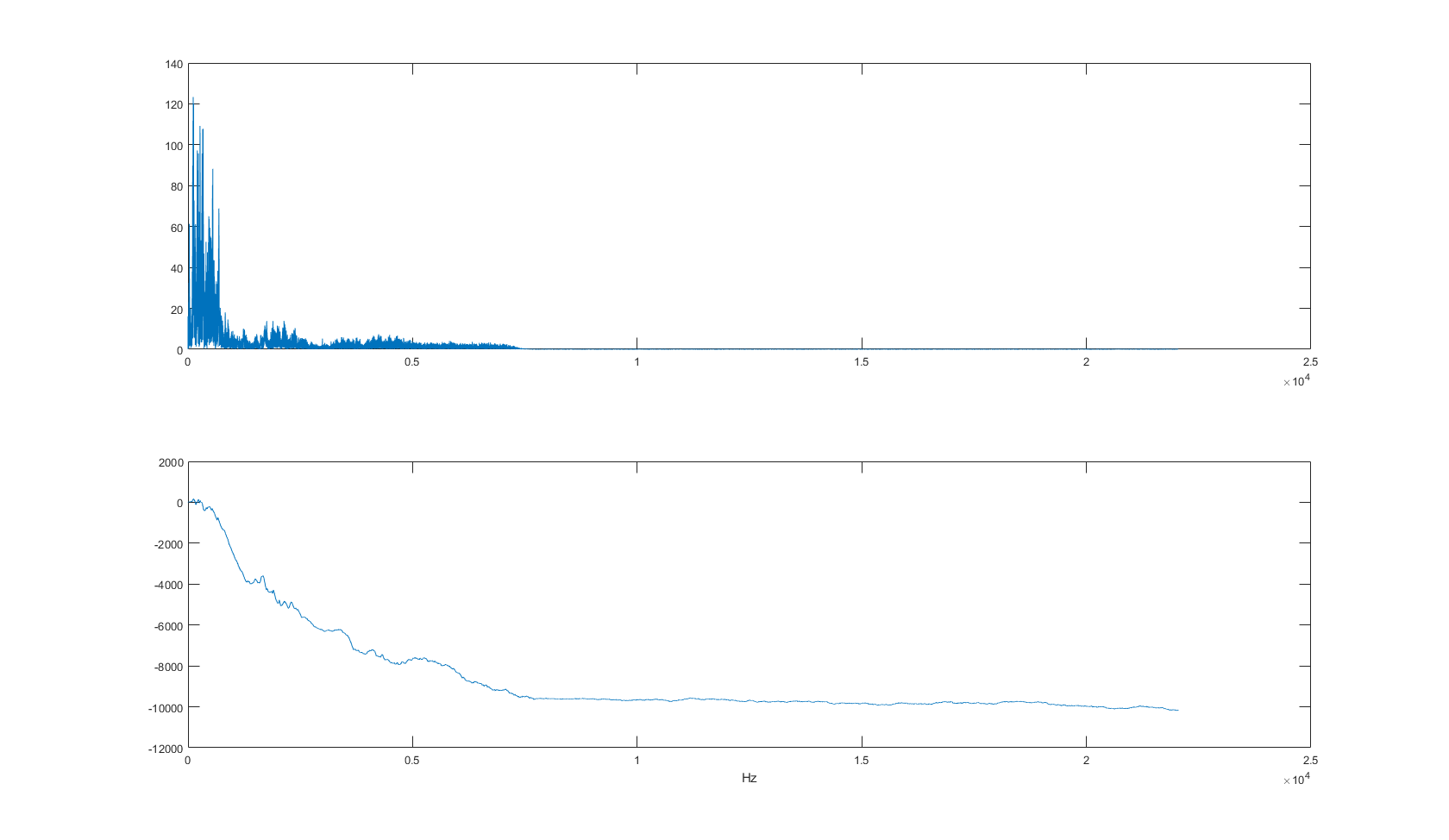
There are several concepts to the idea of audio “scrambling”. The term itself is used extremely loosely to define any method of obscuring a piece of audio. This can be done by encrypting audio using a key or running pieces of audio through a series of low pass filters and rectifying the audio form into a sine wave to distort the audio.

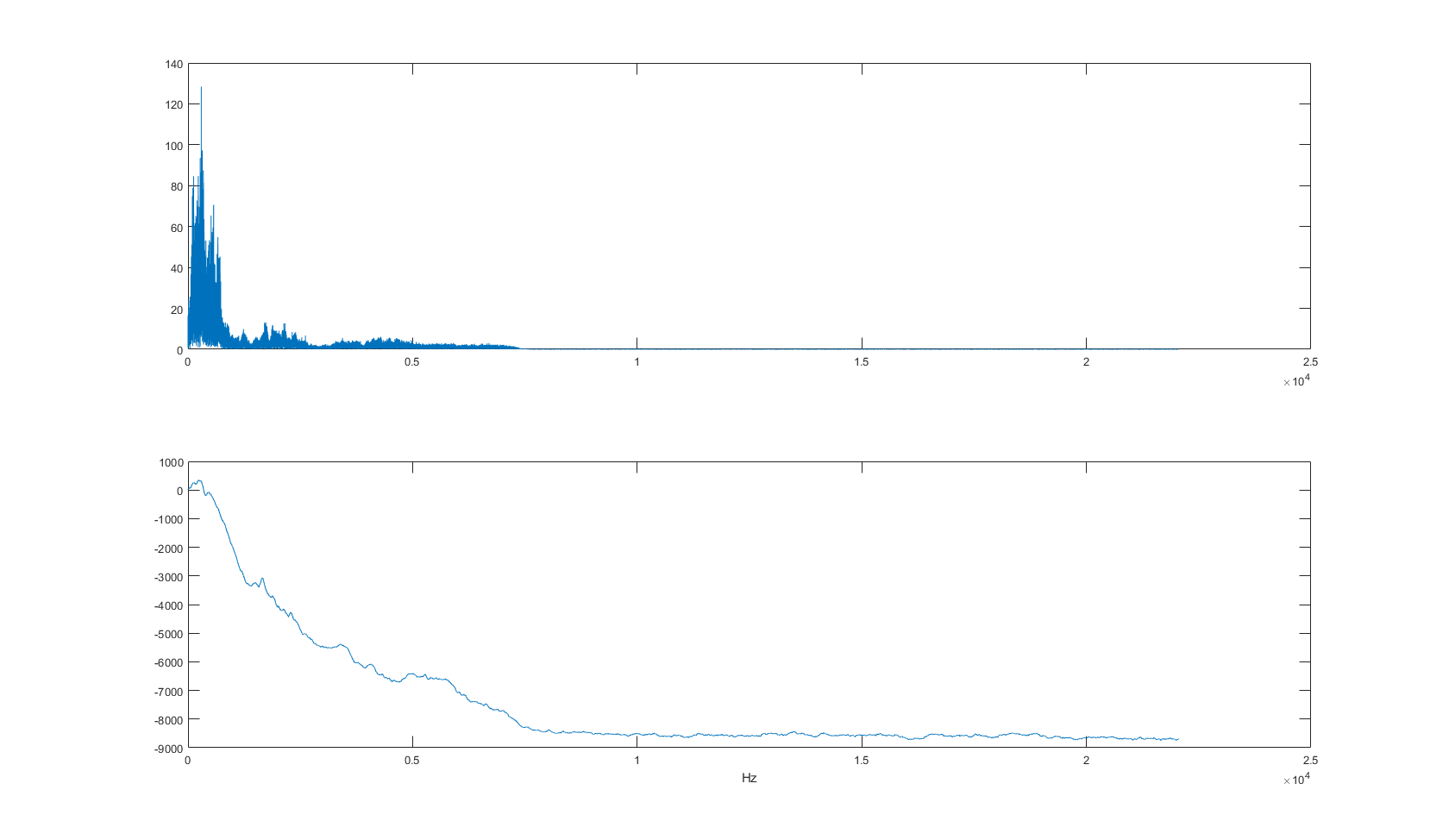
**Detailed Explanation:**

The code uses MATLAB to generate and preform the scrambling effect on a given piece of audio. The idea of code is that it takes the entire piece of audio and divides it into multiple sections (frames) that will be scrambled independent from one another. Once these frames are all fully scrambled they will be recompiled back to form the scrambled audio waveform. The easiest way to approach this was to convert the frames into a matrix, and each of the columns in the frame would represent an audio sample (window). Using a gaussian distribution to define a random scrambling index, the frames would be reorganized using the index. A gaussian distribution is used to ensure that the audio is not scrambled more than 50% to ensure that the scrambled audio can still be identified.

Through matrix manipulation of each of the frames and reconverting it back to native audio the scrambled waveform is created.

**Results:**

****

This shows the original audio waveform including the magnitude, and the overall phase of the audio clip.

This is an example of a scrambled audio waveform using a normal distribution algorithm. Overall, the phase stays relatively the same, however the peak positions have changed slightly for the magnitude. This is the result of scrambling and the audio being shuffled around.

**Concluding Remarks:**

This was probably not the simplest implementation for an audio based scrambler. Nor, is it the most ideal for any security based encryption. The reason is for any security based audio you want to be able to scramble audio and unscramble the audio when it is needed. Using a statistical distribution to randomize the position of certain samples causes the scrambled audio to be essentially irreversible. A best scenario implementation is one that only requires the audio to be looped back the same scrambling protocol and be unscrambled without much change.

Ideally, this scrambler would be much more useful in network simulation testing to test noise and interference. You can use this method to scramble packets of data so they are still recognized but must be processed over a longer period of time.