Final Project Interim Report

Ryan Bunk and Hui-Ying Siao

The overall objective of this project is to design and implement a vocoder that encodes information using the LPC algorithm. The vocoder should be able to compress the input bit stream to less than 16 kbits per second, save the information, and replay the input information as audio. Specifically, the vocoder should be implemented in Matlab, with an interactive GUI to operate the vocoder.

The tasks to be completed are as follows:

- Implement I/O in matlab from audio file formats
- Implement a spectrogram generator for audio files
- Design and implement a vocoder using the LPC algorithm
- Design and implement a voice synthesizer to reconstruct audio data

Thus far, most of the effort expended has been on understanding the nature of the LPC algorithm and how it encodes data, since that would be necessary to the design of the rest of the system. Our current understanding of the LPC algorithm is that it encodes the data as the coefficients of the prediction function, which can then be used to reconstruct an approximation of the original signal. The following link proved helpful to understanding the theory behind the LPC algorithm.

https://www.ece.ucsb.edu/Faculty/Rabiner/ece259/digital%20speech%20processing%20course/lectures_new/Lecture%2013_winter_2012_6tp.pdf

In terms of specific implementation of the system, Mathworks provides an overview and example code for implementation of an LPC audio encoding/decoding system using Matlab. We can use this for inspiration and ideas on how to approach this project. In this example, they implement the LPC algorithm without the built-in LPC function. However, they show specifically that the synthesizer is constructed using a lattice FIR filter in conjunction with an IIR all-pole filter, where the lattice FIR filter coefficients are retrieved from the output of their LPC system.

https://www.mathworks.com/help/dsp/examples/lpc-analysis-and-synthesis-of-speech.html

Finally, the most clear resource we found explained the outputs of the LPC without being bogged down in the finer details, with example input and output from LPC algorithms. This can be found here:

https://ccrma.stanford.edu/~hskim08/lpc/

As far as methodology, we intend to rely upon built-in functionality whenever possible, specifically using the built-in Spectrogram and LPC functions to handle plotting and LPC encoding. So far these have been found to work very well. However, to implement the synthesizer, we will go with the dynamic-coefficient FIR filter approach demonstrated in the Mathworks link. We are still attempting to identify any distinct advantages to a lattice structure as opposed to a direct form structure in this context.

5) Your plans, specific tasks, and date-specific milestones;

Final project due on March 16.

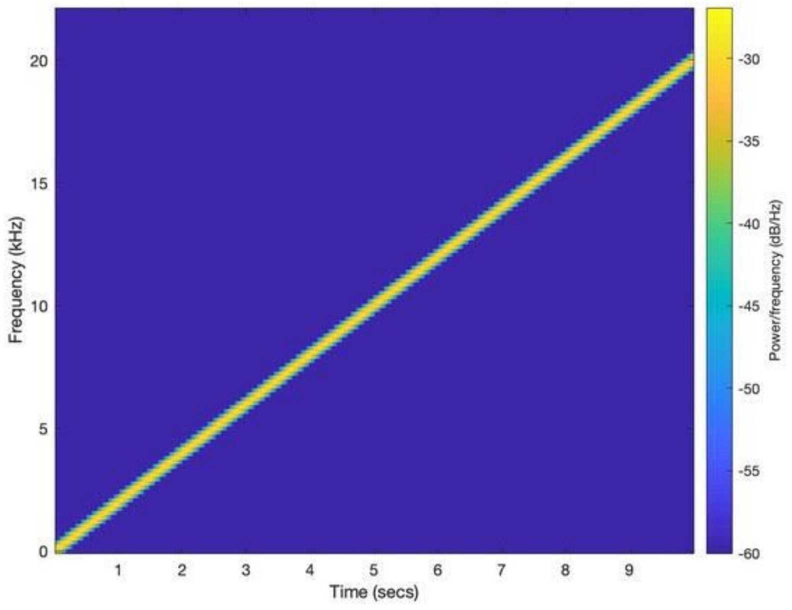
- Importing voice file to Matlab code. (Done)
- Implement spectrogram. (Done)
- Design LPC algorithm processing. (By March 8)
- Design and implement voice synthesizer. (By March 9)
- Another way of estimating the pitch of the voice signal. (By March 10)
- GUI(By March 10)
- Testing (Done by March 10)

6) Members in charge of each task.

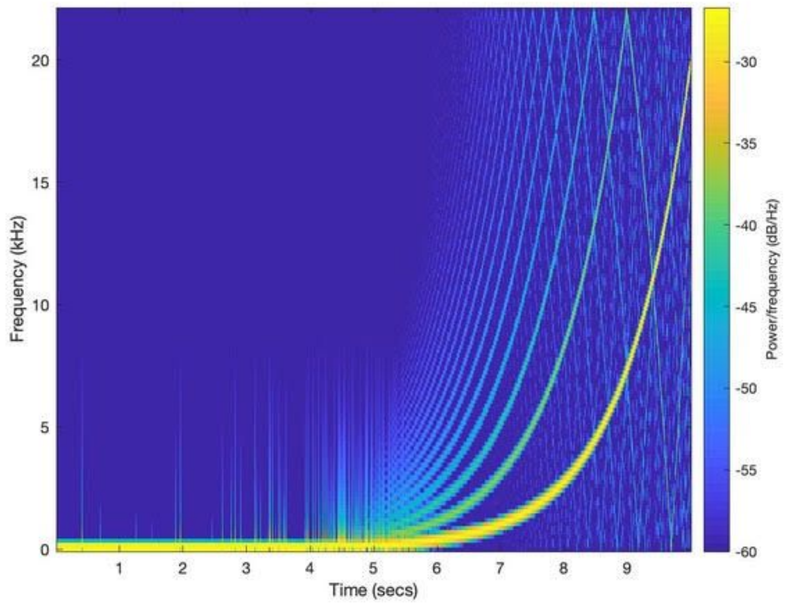
- Importing voice file to Matlab code. (Hui-Ying)
- Implement spectrogram. (Ryan)
- Design LPC algorithm processing. (Ryan)
- Design and implement voice synthesizer. (Hui-Ying)
- Another way of estimating the pitch of the voice signal. (Ryan)
- GUI(Ryan)
- Testing (Together)

7) Any preliminary findings and results.

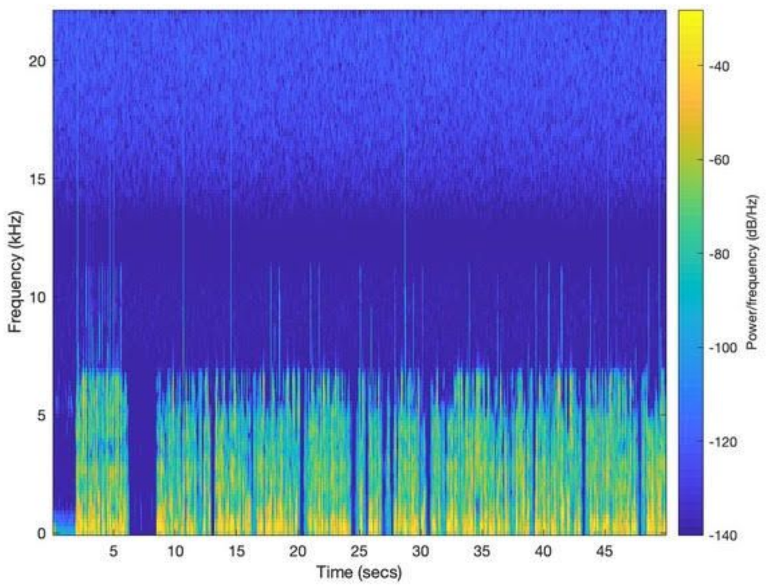
Spectrogram output using linear sinusoidal chirp signal:



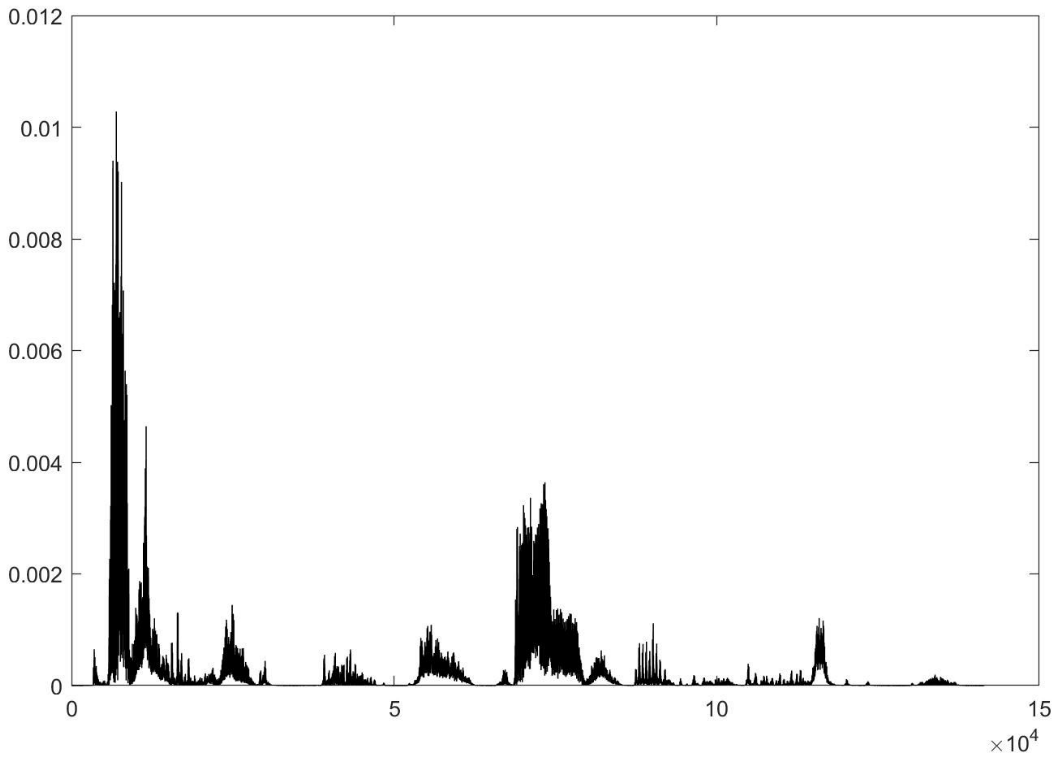
Spectrogram output using logarithmic square chirp signal:



Spectrogram output of test voice file:



Carrier output of LPC using test voice file (impulse + noise):


$\times 10^4$