520.445/645 Audio Signal Processing

Fall 2021

PROJECT 2

(Due 11/16/21 at 11:59pm)

Part 1:

In this project, you will design a speech communication channel using the LPC compression scheme. You are given 10 speech samples and you need to write a MATLAB code that performs:

(a) the compression of the audio signal into as few coefficients as you can so that it can be transmitted over the communication channel. Your transmitter will estimate the prediction parameters a_k and gain G for the speech signal on a frame-by-frame basis (you can use the MATLAB function *lpc* to estimate the a_k coefficients).

(b) after receiving the signal, the receiver needs to be able to re-synthesize the audio signal back. Using the transmitted prediction data (a_k and G) together with the voicing information provided, the task is to synthesize an approximate version of the sentence. Save your resulting synthesis in a file called *sampleX_received.wav* (you can use the MATLAB function *audiowrite*). For each of the 10 wavefiles, you are provided with a .mat file that contains an estimate of the fundamental frequency every 10msec. Note, you do not have to use 10msec as your frame rate.

The ultimate goal of this project is to achieve a compromise between lowest compression vs. quality of the resynthesized speech. Your final deliverable are reconstructed waveforms with the *best* possible quality, using the *least* amount of coefficients (i.e. filter coefficients, voicing information, pitch estimates, etc). Your report **MUST** include an estimate in bits/second or samples/second of how much information is required *on average* to reconstruct your signal. Your performance in this project will be assessed in comparison with how well the others perform. We will assess signal quality as well as signal compression.

Note: The files X.mat contain estimates of voiced/unvoiced decision and pitch frequencies at a frame rate of 100 frames/sec. A zero value indicates that the speech frame is unvoiced or silence, and a non-zero value is the pitch frequency estimate (in Hz) when the frame is voiced. Note that these estimates are obtained from a correlation analysis of the signal. They are only approximations, and may not be accurate for every frame. Use the MATLAB function *load* to load the pitch estimates.

Part 2:

Use MATLAB to record your own voice saying the sentence "The synthesized signal is supposed to be of high standard". Send this signal through your communication channel developed in Part 1. Now, can you optimize the choice of parameters specifically

for this sentence to improve compression and quality. Note this sentence has a large number of fricatives, specifically the phoneme /s/. What do you need to change in your parameters to improve compression/quality? Discuss your choices in your report.

In order to estimate the pitch/voicing information for this specific speech sentence, you can look at the short-time spectrogram and estimate (by eye) what your fundamental frequency is. You can also write a MATLAB script to automate pitch estimation (this will count towards your bonus sections - see below).

Few pointers:

1. Work on this project individually.
2. Pay close attention to your choice of frame rate, as well as number of coefficients. Explore varying the number of coefficients for voiced vs. unvoiced frames.
3. A number of factors may affect the quality of your synthesized speech. For instance, what goes on in a given frame is not independent of what happened in previous frames. As the pitch period changes, you will need to know where the last pitch impulse occurred in the previous frame so as to determine the location of the next impulse in the current frame. You should also examine what is the benefit of using different glottal shapes for your voiced segments.
4. You can change the vocal tract filter once per frame, or you can interpolate between frames. The voicing information is provided to you at a rate of 10msec but you should explore the frame rate that works best.
5. Explore if there is any useful information in the remaining error from the LPC coding. Does it add anything to quality if you were to compress it and send it through the communication channel as well? If you do, you will be increasing the number of bits/sec but is it worth it?
6. Listen to your synthesized speech and see if you can isolate what are the main sources of distortion.
7. If you wish, you could use an automated quality metric to evaluate the quality of your synthesized signal. Documentation and code for this metric (PESQ: Perceptual Evaluation of Speech Quality) are included with the project documents. It is your choice if you want to use this metric. Whether you evaluate your synthesized signal by ear or using an automated metric will not affect your final grade.

Report

Write a report describing how you programmed the LPC Vocoder (e.g., decisions made on frame length, excitation generation, frame overlaps, etc.), along with any graphics. Submit your project no later than 16 November 2021, 11:59pm.

Bonus points

- **Voicing (20 points):** Implement your own voicing detector, which will indicate whether a speech segment is voiced, unvoiced or silence. You can choose a set of parameters, compare their distributions (under each voicing category), and choose a threshold to detect the voicing state. You can use this detector for your LPC project. Your deliverable is a *stand-alone* MATLAB function called *voicingdetector.m*, which takes as input arguments a sound vector, sampling rate, frame length and frame overlap, and returns an array of voicing decisions for each frame with 0 indicating silence, +1 indicating voiced frame, and -1 indicating unvoiced frame.

- **Pitch (30 points):** Implement your own pitch detector based on either the signal's autocorrelation, the LPC residual signal, or any other method and use its output instead of the provided pitch files (.mat). Experiment with the pitch detection method as well as how often you estimate pitch (i.e. frame rate). Does it improve the quality of your reconstructed signal? Again, your deliverable is a *stand-alone* MATLAB function called *pitchdetector.m*, which takes as input arguments a sound vector, sampling rate, frame length and frame overlap, and returns an array of pitch values for each frame and zero for unvoiced and silence frames. Obviously, there is a voicing detection implied in this process.