LPC Vocoder Project B

Final Project – EEC 201 – Digital Signal Processing, Winter 2019

Anupam Mohanti
Student ID – 916681128

Mark Allen-Piccolo
Student ID – 994172581

Table of Contents

Introduction:

Speech processing is the study of speech signals and processing methods and is the intersecting point of digital signal processing and natural language processing. Speech processing technologies are used for a variety of applications like digital speech coding, text-to-speech synthesis, spoken language dialog systems etc. because speech is a more intuitive way of accessing information. Information such as speaker, gender, or language identification, or speech recognition can also be extracted from speech. For this we need to code the speech signal through an algorithm, which is application specific. There are many kinds of speech coding techniques available of which the most popular method is parametric coding. Parametric speech coding again consists of different methods, the most basic of which is the linear predictive coding (LPC). Although there are several variations of this linear predictive code exists, in this project we have implemented LPC-10 method which is the basic linear predictive coding with 10 coefficients.

Problem Statement:

- Design and implement an LPC vocoder and synthesizer in Matlab with GUI.
- The input to the system will be a sampled speech signal.
- The output of the system will be reconstructed speech signal based on LPC-10 algorithm.

Additional Design constraint:

- Generate a spectrogram of the original signal and the synthesized signal for comparison.
- Provide at least two ways of estimating the pitch of the voice signal. Compare the result of the two different pitch estimation

Design Approach:

In the first half of the system i.e. LPC-10 ENCODER, input speech signal is segmented into frames and processed to extract information like whether the frame is voiced or unvoiced, power of the predictive error and pitch period to compute gain, and LPC coefficients for the frame.

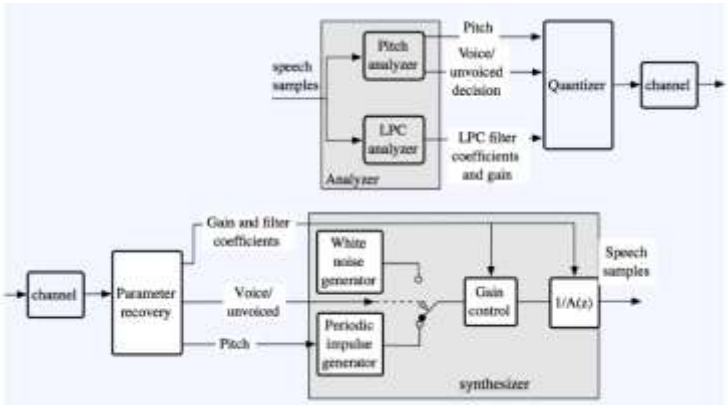
In the latter half i.e. LPC-10 DECODER, the information extracted at the encoder- linear predictive coefficients, pitch period, voiced/unvoiced classification, gain- is used to reconstruct the speech signal. If the reconstructed signal is intelligible, then the designed Vocoder is a good model.

LPC-10 Speech Coding Algorithm

A continuous speech waveform consists of various phonemes spoken in succession and hence it becomes a challenge to determine the different characteristics of the speech samples if taken altogether. However, if we break up the speech samples into smaller frames, the parameters of the speech samples are assumed to be constant for the short interval of 1 frame i.e. 30 ms for our experiment. So, instead of quantizing and figuring out the parameters of the whole signal, the parameters of speech are estimated in the form of filter coefficients. The filter shape and the voicing determine the phoneme that is spoken.

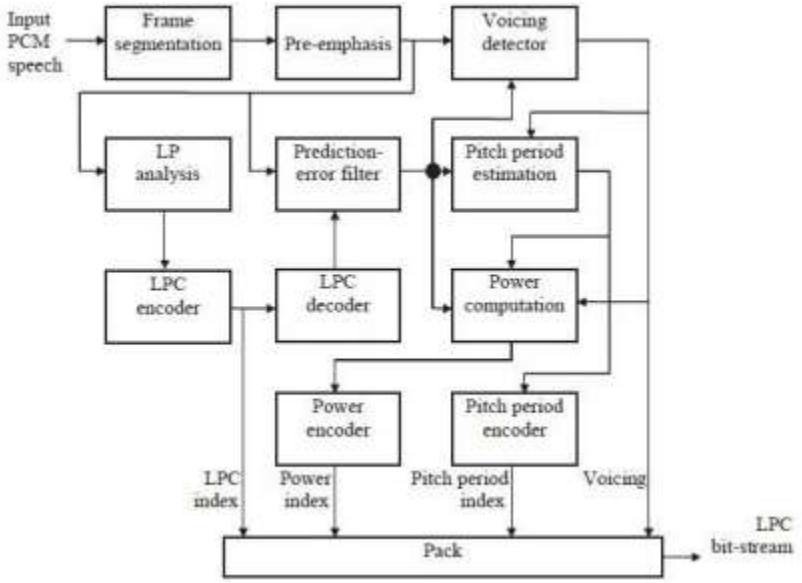
The LPC speech coding algorithm is a simple speech production model and the block diagram is shown of the whole model is shown in Figure. The speech samples are passed through an analyzer where the following parameters are computed for the input speech samples.

- **Voicing Information**- whether the frame is voiced or unvoiced
- **Gain** - representing the predictive error energy in the frame
- **Pitch period** – using Autocorrelation and Cepstrum Analysis (in voiced frames only)
- **Analyze filter parameters** - The filter parameters are estimated by solving Levinson-Durbin recursion.



At the decoder, using these filter coefficients and artificial excitation (either an impulse train, in case of voiced speech or white noise, in case of unvoiced speech), inverse filtering is carried out and the speech signal is resynthesized. LPC-10 Vocoder system mainly consists of 2 blocks an ENCODER and a DECODER. These blocks are explained in next two sections along with some design decisions.

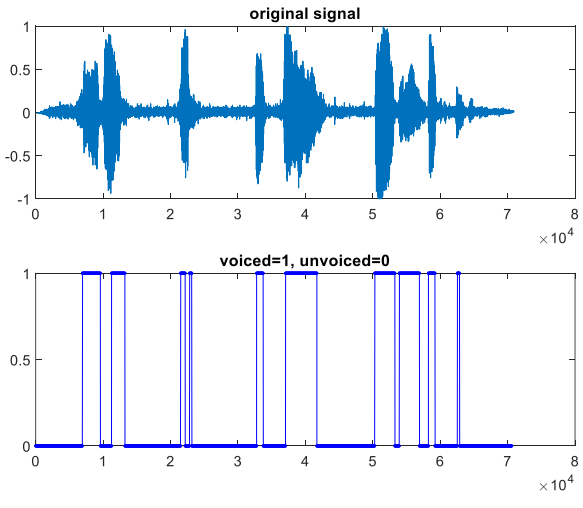
LPC-10 Encoder:



The block diagram of the Encoder portion of our LPC-10 is shown above. Input to the encoder is a sampled speech signal. Here we have used sampling rate of 8 kHz and 11.25 kHz. The output is generated. The input signal is first segmented into frames of 30 ms (with 50% overlap) using a hamming window. This step helps with the short time analysis of the input speech signal. Each frame is then passed through a pre-emphasis filter with transfer function $(1/(1-0.9375z^{-1}))$. This is done so as to amplify the high frequency components so as to increase the SNR of the whole signal. The pre-emphasized frame is then subject to voiced/unvoiced detection, the pitch period and gain of the voiced frames are calculated. In the last step we compute the LP cofficients by using Levinson-Durbin recursion. These computed values are then sent to the decoder for regeneration of the speech signal. Some of the important sections of the LPC-10 encoder is described in the following section.

Detection of Voiced/Unvoiced frame:

There exist several methods like Zero Crossing Rate, energy of error signal, and low band energy, to find out the voicing information of the signal. In this experiment we tried the Zero crossing rate to determine if a frame is the voiced/unvoiced. With this method we count the number of times the signal crosses 0 i.e. number of sign changes occurring in that frame. Generally, voiced signal has lower zero crossing rate than that of the unvoiced frame. The reason is that compared to the unvoiced signal, voiced speech has significantly higher magnitude and periodicity compared to



unvoiced speech which is highly variable. Thus, the frequency with which a voiced speech crosses zero is significantly lower than unvoiced speech.

Pitch Detection:

Like voiced/unvoiced detection, there are several methods to determine the pitch of a voiced frame of signal. Here we have employed two different methods for pitch detection: autocorrelation and cepstrum analysis. Both procedures are explained below:

Autocorrelation:

The idea of autocorrelation is to provide a measure of similarity between a signal and itself at a given lag. There are several ways to approach it, but for the purposes of pitch detection, one can think of it as a search procedure. In other words, you step through the signal sample-by-sample and perform a correlation between the reference window and the lagged window. The correlation at "lag 0" will be the global maximum because we compare the reference to a verbatim copy of itself. For a periodic signal, at some point the correlation it will begin to increase after decreasing, and then reach a local maximum again. The distance between "lag 0" and that first peak gives an estimate of the pitch of the signal. In our project we have implemented just this.

Cepstrum:

An alternate approach to pitch detection is implemented using the cepstrum method (the word is made from reversing SPEC in spectrum). In this method, the time-domain audio is analyzed by shifting a 10-milisecond frame over the audio and analyzing each frame separately from the preceding frame. Each frame contains information that allows for pitch identification.

Each 10-milisecond frame contains a finite number of samples. This number changes depending on the sample rate of the audio. For example, at 8,000 samples per second, the frame will contain 80 samples; whereas at 48,000 samples per second, the frame will contain 480 samples. For each frame, the discrete Fourier transform is taken (taking the DFT of a shifting frame is itself called a short-term-Fourier-transform (STFT)). The frequency response is squared, and its log is taken. The inverse discrete of Fourier transform is taken. Within this transformed signal, at the location of every maximum peak, the frame contains a quefrency. By dividing the sample rate by the quefrency, the pitch is determined. It is then returned to the vocoder and is used alongside the voiced/unvoiced processing and LPC coefficients in the analysis section.

This process is best described by A.M. Noll who details how the process was initially used to detect seismic activity but could be repurposed as a new function for pitch detection:

"…devise a new function in which the effects of the vocal source and vocal tract are nearly independent or easily identifiable and separable. The Fourier transform of the logarithm of the power spectrum is such a new function and, indeed, separates the effects of the vocal source and tract. The reason for this is that the logarithm of a product equals the sum of the logarithms of the multiplicands.

$$\log|F(\omega)|^2 = \log[|S(\omega)|^2 \cdot |H(\omega)|^2] = \log|S(\omega)|^2 + \log|H(\omega)|^2$$

The Fourier transform of the logarithm power spectrum preserves the additive property and is

$$\mathcal{F}[\log|F(\omega)|^2] = \mathcal{F}[\log|S(\omega)|^2] + \mathcal{F}[\log|H(\omega)|^2]$$

The source and tract effects are now additive rather than convolved as in the autocorrelation…The effect of the vocal tract is to produce a low frequency ripple in the logarithmic spectrum, while the periodicity of the vocal source manifests itself as a high frequency ripple in the logarithmic spectrum. Therefore, the spectrum of the logarithmic power spectrum has a sharp peak corresponding to the high frequency source ripples in the logarithmic spectrum and a broader peak corresponding to the lower frequency formant structure in the logarithmic spectrum. The peak corresponding to the source periodicity can be made more pronounced by squaring the second spectrum…The cepstrum consists of a peak occurring at a high quefrency equal to the pitch period in seconds…"

Calculating LPC Coefficients using Levinson-Durbin recursion:

Linear prediction is a mathematical operation where future values of a discrete-time signal are estimated as a linear function of previous samples. The solution for a random signal is given by"

$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + e[n]$$

Where {ak} are pth order linear predictor coefficients. The causal solution for the parameters {ak} is found in each case to be of the form:

$$\begin{bmatrix} \phi[0] & \phi[1] & \phi[2] & \phi[3] \\ \phi[1] & \phi[0] & \phi[1] & \phi[2] \\ \phi[2] & \phi[1] & \phi[0] & \phi[1] \\ \phi[3] & \phi[2] & \phi[1] & \phi[0] \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} \phi[1] \\ \phi[2] \\ \phi[3] \\ \phi[4] \end{bmatrix}$$

Or equivalently or equivalently, $T_p * a_p = r_p$, Where {φ[k]} represent the autocorrelation coefficients of the signal s[n]. Levinson-Durbin recursion provides for a faster solution for in the system of equations. The basic simple ideas behind the recursion is first that it is easy to solve the system for k =1, and second that it is also very simple to solve for a k +1 coefficients sized problem when we have solved a for a k coefficients sized problem. We see that this hinges on the fact that T_p is a p×p Toeplitz matrix, that is, that T_p is symmetric, and that all entries along a given diagonal are equal. The equations of the Levinson-Durbin recursion, which are used to compute the corresponding reflection coefficients and LPC parameters are:

$$E^{(0)} = \phi[0]$$

$$k_i = \frac{\left\{ \phi[i] - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} \phi[i-j] \right\}}{E^{(i-1)}}, \text{ calculated for } 1 \le i \le P$$

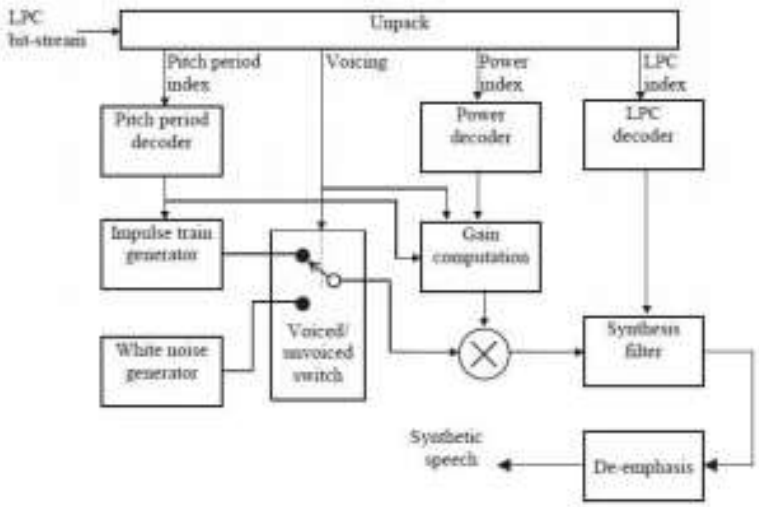
$$\alpha_i^{(i)} = k_i$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}, \text{ for } 1 \le j \le i-1$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}$$

The set of equations are solved recursively for i = 1, 2,…p and the final solution is given by $\alpha_j = \alpha_j^{(p)} = 1 \le j \le p$. We have tried to implement the whole process in Matlab. There is a Matlab command 'lpc' which directly computes the coefficients and our results match with the results from using 'lpc' command.

LPC-10 Decoder:



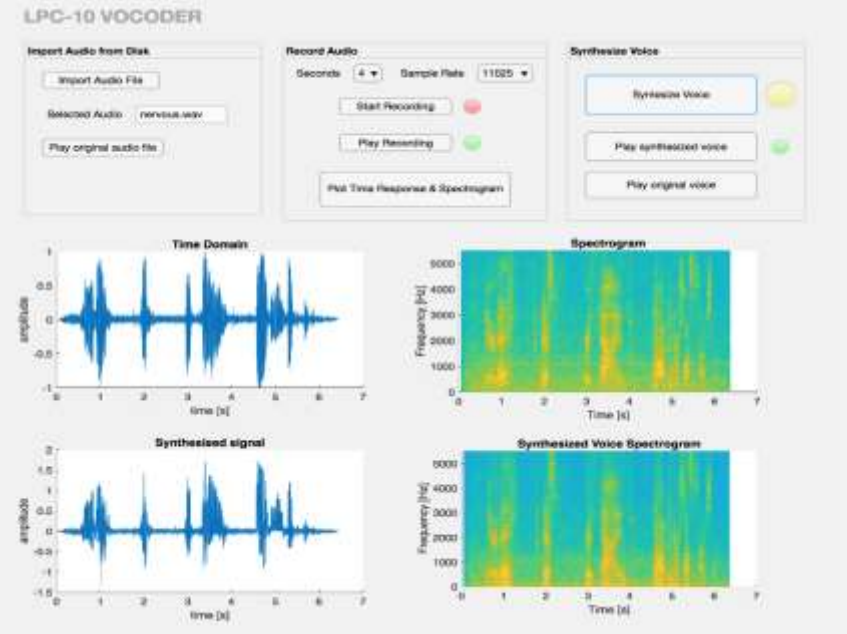
Above we have attached the block diagram of a typical LPC decoder. The decoder takes all the information transmitted by the encoder about each frame to reconstruct the signal. It takes in the voicing information and depending on that multiplies with a pulse train and pushes that as output. For an unvoiced frame it generates white noise. For a voiced frame, the pulse train is multiplied with the pitch period and excitation for that frame is generated. This excitation is then multiplied by the gain factor. This resultant signal is then passed through the synthesis filter. Interesting point to observe here is the synthesis filter performs the inverse filtering as that of analysis filter. Then this filtered signal is fed to de-emphasis filter, which an inverse function of pre-emphasis filter. Output of de-emphasis filter is the reconstructed speech.

GUI Implementation using MATLAB:

The most important criteria for our GUI implementation is that it should be intuitive to use, that it gives the user feedback, and that it functions smoothly.

The GUI has four main sections: import audio from disk, record audio from computer microphone, synthesizer, and plotting.

The **Import Audio from Disk** section allows the user to select an audio file that is stored on their hard drive. Once selected, the time response and spectrogram are plotted. The file name is displayed, and the user has the option to play the audio clip. The user has the option to select a different file which will overwrite the whatever file was chosen previously.



The **Record Audio** section allows the user to record directly from their computer microphone. The audio is recorded in mono. The recording length and sample rate are user selectable through two pull down menus. This can be changed in the code. Once the audio is recorded, the user can preview what they recorded by pressing *Play Recording*. If the user is satisfied with what they recorded, they can plot the time response and spectrogram by pressing the *Plot Time Response and Spectrogram* button. When the program is actively recording, the red lamp will enable. It continues until the recording stops. Similarly, when the audio is played back, the green lamp will illuminate until the playback finishes.

The **Synthesize Voice** section implements the LPC-10 vocoder. The audio file that is currently stored is passed through the vocoder and stored. Its time response and spectrogram are plotted below the plots of the original signal so that the user can compare their similarities and differences. The two buttons, *Play Synthesized Voice* and *Play Original Voice* start playback of the separate files.

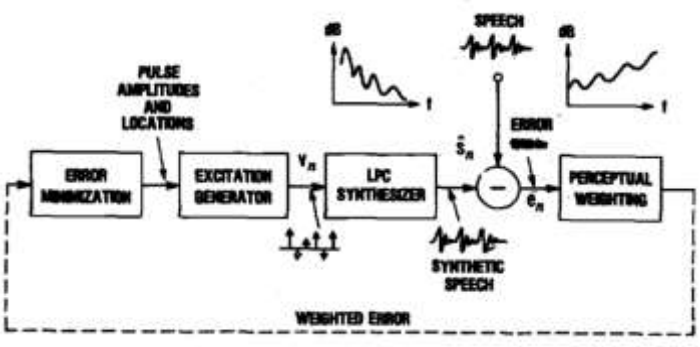
The **Plotting** section displays the time response and spectrograms of both the original audio file and the synthesized audio file.

Challenges and Future Directions:

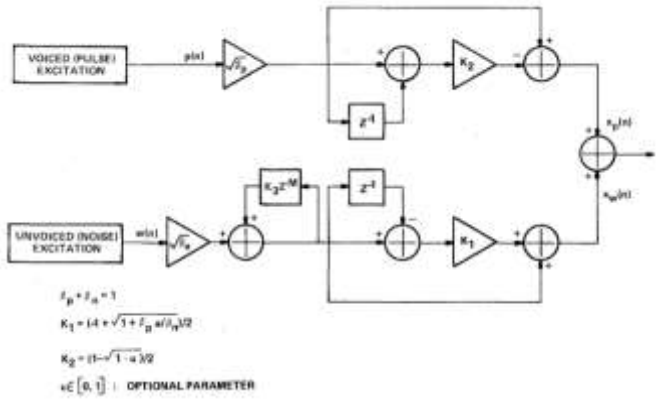
One key challenge we faced is that the fidelity of the synthesized voice is low. It has a buzzy and robotic quality, particularly for male speakers. Through our research, we discovered that this is a common problem with rudimentary LPC-10 vocoders, and there has been a lot of effort to improve it. In their paper "An enhanced LPC vocoder with no voiced/Unvoiced switch" authors Soon Kwon and A. Goldberg confirmed our problem saying “The LPC speech produced with excitation source [periodic pulse for voiced and white random noise for unvoiced] often sounds synthetic, unnatural, and buzzy, particularly for male speakers.” One technique to improve the standard LPC vocoder is developed by B. Atal and J. Remde. They hypothesize that the transition from voiced to unvoiced cannot be represented simply by a switch because in actual speech, the transition is itself not a sharp transition. Rather it contains both voiced and unvoiced information. Or, as Atal and Remde put it, it is difficult to classify the region because the modes are mixed. In their paper, "A new model of LPC excitation for producing natural-sounding speech at low bit rates," they design what they call a Multi-Pulse excitation model. They explain how it functions:

“The synthetic speech samples are compared with the corresponding speech samples of the original (natural) speech signal to produce an error signal e_n. This error is not very meaningful and must be modified to take account of how the human perception treats the error…Broadly speaking, the error in the formant regions must be dc-emphasized. This is done by a linear filter which suppresses the energy in the error signal in the formant regions. The error signal is thus weighted to produce a subjectively-meaningful measure of the difference between the signals $\hat{s}_n$ and s_n. The weighted error is squared and averaged over a short time interval 5 to 10 millisecond in duration to produce the mean-squared weighted error e. The locations and amplitudes of the pulses are chosen to minimize the error e… The procedure for finding the locations and amplitudes of various pulses in any given time interval can be summarized as follows: At the beginning, without any excitation pulse, the synthetic speech output is entirely generated from the memory of the all-pole filter from previous synthesis intervals. The contribution of this past memory is subtracted out from the speech signal and the location and amplitude of one single pulse, which minimizes the mean squared weighted error, is determined. A new error signal is now computed by subtracting out the contribution of the pulse just determined. The process of locating new pulses to reduce the mean-squared weighted error is continued until the error is reduced to acceptable limits.”

This process is visualized in the following block diagram.



Soon Kwon and A. Goldberg post their own improvements, called LPC Plus, which they claim would improve fidelity while at the same time keeping the computation low—a disadvantage of Atal and Remde's design. Their approach is to combine the voiced excitation with noise excitation with an additional parameter pitch gain β.



We were not able to implement this method, but according to Kwon and Goldberg if β is quantized with more than 1 bit, the sensitivity to voice errors is greatly reduced. The implementation requires an additional comb filter that reduces the hoarseness that is introduced by the white noise excitation source.

We were not able to implement the function where a male voice is transformed into a female voice, and vice versa. Given more time, however, we would use PSOLA method. As described in the paper by Mousa, Allam, "Voice Conversion using Pitch Shifting Algorithm by Time Stretching with PSOLA and Re-Sampling", we can implement time-stretching together with resampling. Resampling has the effect of changing the pitch, however it also speeds up or slows down the time. Time-stretching will counteract the change in time caused by resampling.

Although our system works properly with 10 LPC coefficients, it does not work for LPC orders greater than 10. With a higher number of coefficients, the synthesized voice becomes unintelligible to non-existent. We checked the LPC coefficients we generate against the MATLAB command lpc( ) and they are the same values. Even though our LPC coefficients seem to be correct, the error value increases as we increase the order. We suspect that this is our issue. Given more time, we would decrease the error which would allow a wider range of LPC order.

In the GUI, recording audio from the computer could have been a better user experience if there were a start and stop button, rather than a drop-down button that selects the recording time. We tried to implement this feature but the syntax that works in the MATLAB command window does not work in app designer. Given more time, we could have found a way to implement this. Despite this, having a dropdown menu for recording times is easy enough to use.

Conclusion:

Through the use of linear prediction, we were able to reconstruct speech which is intelligible. However, as mentioned in the previous section, the reconstructed voice sounds automated, robotic and in a different tone than the original. Through our literature study we found that it is a defect of LPC vocoders as it reconstructs using artificial excitations using white noises and impulses instead of prediction errors. However, for applications with emphasis on low bit rate this is a good approach to reconstruct speech.

This project has been really helpful to understand different concepts related to speech processing. Through self-study, discussion with fellow classmates, the professor and TA we were able to learn several different things which may not have been possible to learn in a class. It was a fun project and we enjoyed every bit of it. It gives us immense pleasure that our vocoder can reconstruct the original speech signal intelligibly.

References:

- Noll, A. Michael. “Cepstrum Pitch Determination”. The Journal of the Acoustical Society of America 41, 293 (1967); doi: 10.1121/1.1910339
    - A paper with fundamental design concepts and techniques about Cepstrum Pitch Detection.

- Tan L, Karnjanadecha M. Pitch detection algorithm: autocorrelation method and AMDF. In: Proceedings of the 3rd international symposium on communications and information technology; 2003. pp. 551–556.
    - A paper about pitch tracking techniques using autocorrelation method and Average Magnitude Detection Function to extract the pitch pattern.

- L. Rabiner, "On the use of autocorrelation analysis for pitch detection," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 25, no. 1, pp. 24-33, February 1977. doi: 10.1109/TASSP.1977.1162905
    Another paper describing types of autocorrelation to determine the pitch of a speech signal

- Ozgu Ozun, Philipp Steurer and Deniel Thell,” Wideband Speech Coding with Linear Predictive
- Coding (LPC) “ in EE214A: Digital Speech Processsing , Winter 2002
    - A project carried out at University of California, Los Angeles for the department of Electrical Engineering. Interesting in that it gives a sense of how another group approached this same design and can offer insights into our own design. They also have some examples of MATLAB script. While we will write our own, it can be helpful to see how another group approached the same problems that may come up.

- Veena and D. D. Geetha. “Voice Excited Lpc for Speech Compression by V / Uv Classification.” (2016). DOI: 10.9790/4200-0603026569

- R. Bachu, S. Kopparthi, B. Adapa, and B. Barkana, “Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal,” in Proc. Amer. Soc. Eng. Edu. Zone Conf., 2008, pp. 1–7.

- Gordon, John W, and John Strawn. An Introduction to the Phase Vocoder. Stanford, Calif: CCRMA, Dept. of Music, Stanford University, 1987. Print. –

Although not directly related to LPC-vocoders, reading about phase vocoders gives an alternate approach, providing insights into how speech can be synthesized.

- Yip, W, Barron, D. Efficient codebook search for CELP vocoders. (1991) (US Patent).
    - We won't implement this vocoder, but it is a design that improves the fidelity as compared to the LPC that we are designing.

- Paranjape A., Gadgil, M. "Linear Predictive Coding LPC-10 VoCoder" - Term Project -ECE 252A- speech Compression, Winter 2017 UCSD –
    - An interesting handbook on the components of an LPC and how they can be implemented

- D. W. Griffin and J. S. Lim, "Multiband excitation vocoder," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 8, pp. 1223-1235, Aug. 1988. doi: 10.1109/29.1651

- Soon Kwon and A. Goldberg, "An enhanced LPC vocoder with no voiced/Unvoiced switch," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 32, no. 4, pp. 851-858, August 1984. doi: 10.1109/TASSP.1984.1164377

- B. Atal and J. Remde, "A new model of LPC excitation for producing natural-sounding speech at low bit rates," ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing, Paris, France, 1982, pp. 614-617. doi: 10.1109/ICASSP.1982.1171649

- Mousa, A. "Voice Conversion using Pitch Shifting Algorithm by Time Stretching with PSOLA and Re-Sampling". Journal of Electrical Engineering. 61. 2011. doi: 10.2478/v10187-010-0008-5.