Spectral Envelop Estimation based on Linear Prediction Coding

MUMT 605 – Digital Sound Synthesis and Audio Processing

Harish Venkatesan
M.A. in Music Technology
Schulich School of Music
McGill University

1. Source Filter Model

Most stable resonant systems can be modelled as a combination of an excitation source and a resonant filter which shapes the input stimulus produced by the source to give periodic vibrations. This excitation-resonance model is also known as source-filter model. The shaping of sounds is achieved by superimposing a spectral envelop on the source signal. It is this spectral envelop that helps us in identification of sounds; for example, each vowel has a different spectral envelop, which we can perceive, that helps us in identifying them.

In the source filter model, by estimating the response of the filter from analysis of a given sound, we basically estimate the spectral envelop which shapes the source signal. Once the filter response is known, the source signal can be estimated. This is known as source-filter separation and can be performed in various ways. In this project, we make use of the technique of Linear Prediction Coding to estimate the spectral envelop of a sound, perform source-filter separation and implement certain audio effects.

In this report, the concept of source-filter separation is explained in the context of speech production model. The next section of the report briefly describes the speech production model, followed by detailed description of linear prediction coding and how it can be used to implement certain audio effects. All concepts presented in the paper are based on [1], [2] and [3].

2. Speech Model

Speech production can be modelled as an excitation-resonance model with the glottal vibrations acting as the excitation source and the vocal-tract acting as the linear resonant filter. The vocal tract can be considered as an acoustic resonant tube of varying cross-section. Each region of this tube can be modelled as a second-order filter. Cascading filter representations of all the regions of the vocal tract, we get a filter of order p, where p equals twice the number of sections considered. A filter of order p = 12 can reasonably approximate the human vocal tract. The speech production model is shown in fig. 1.

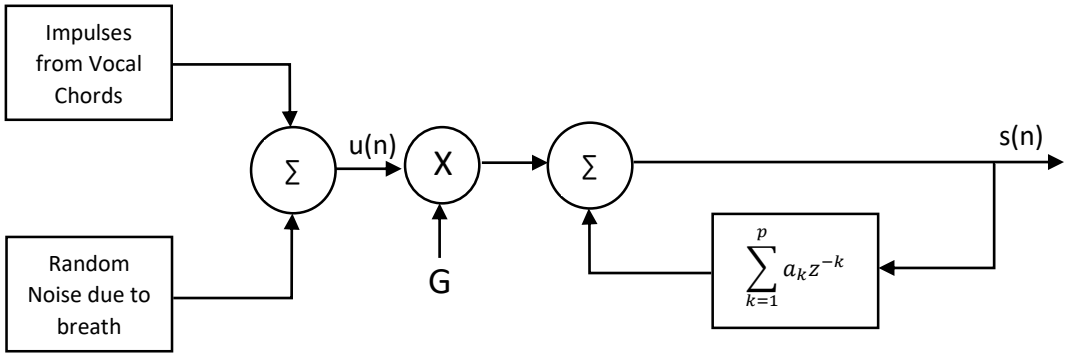

Figure 1: Speech production model

From the above figure, the output voice signal can be written as,

$$S(z) = \frac{G\ U(z)}{1 + \sum_{k=1}^{p} a_k\ z^{-k}} \tag{1}$$

$$s(n) = -\sum_{k=1}^{p} a_k s(n-k) + Gu(n) \tag{2}$$

From the above equation, we can see that the current sample of the output speech is given as the weighted sum of p past samples and the input source signal u(n).

3. Linear Prediction for Source-Filter Separation in Speech

In linear prediction coding, we try to estimate the current sample of a signal using the information we have from the past samples. We use an all-zero filter of order p to estimate the current sample as follows.

$$\hat{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \tag{3}$$

where $\hat{s}(n)$ is the estimation of the current sample.

The error in estimation of the current sample is given by,

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \tag{4}$$

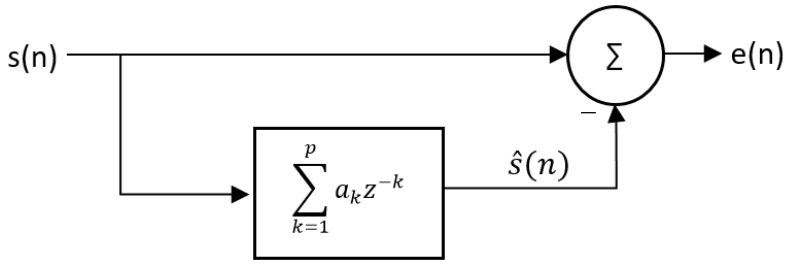

Figure 2: Linear predictor filter that predicts the current sample based p past samples

3.1 Computation of Filter Coefficients

The energy of the error signal is given by,

$$E = \sum_n e^2(n) = \sum_n \left[s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \right]^2 \tag{5}$$

For best approximation of the current input sample, we must compute the filter coefficients such that the energy E of the error signal is minimum. Speech signal consists of both deterministic (voiced) and random (unvoiced) components. Considering a deterministic signal s(n), in order to minimize the energy of the error signal,

$$\frac{\partial E}{\partial \alpha_i} = 0, \qquad 1 \le i \le p \tag{6}$$

$$\sum_n 2 \left[s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \right] [-s(n-i)] = 0 \tag{7}$$

$$-\sum_n s(n)s(n-i) + \sum_n \sum_{k=1}^{p} \alpha_k s(n-k)s(n-i) = 0 \tag{8}$$

$$\sum_{k=1}^{p} \alpha_k \sum_{n} s(n-k)s(n-i) = \sum_{n} s(n)s(n-i) \tag{9}$$

The autocorrelation R(i) of a signal s(n) is given by,

$$R(i) = \sum_{n} s(n)s(n-i) \tag{10}$$

Therefore, the eq. (9) given above can be reduced to,

$$\sum_{k=1}^{p} \alpha_k R(i-k) = R(i), \qquad 1 \le i \le p \tag{11}$$

Similarly, for a random signal s(n), the energy of the error can be given as,

$$E_r = \mathrm{E}(e_n^2) = \mathrm{E}\left[s(n) - \sum_{k=1}^{p} \alpha_k s(n-k)\right]^2 \tag{12}$$

where E_r is the energy of the error due to random signal s(n).

Minimization of this error yields results similar to the one obtained in the case of deterministic signal.

$$\sum_{k=1}^{p} \alpha_k R(i,k) = R(i), \qquad 1 \le i \le p \tag{13}$$

The eq. (11) and (13) are known as normal equations. Now, there are p simultaneous equations and p unknowns which can be solved as follows.

$$\begin{bmatrix} R(0) & R(1) & \cdots & R(p-2) & R(p-1) \\ R(1) & R(0) & \ddots & \ddots & R(p-2) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ R(p-2) & \ddots & \ddots & R(0) & R(1) \\ R(p-1) & R(p-2) & \cdots & R(1) & R(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ \vdots \\ R(p) \end{bmatrix} \tag{14}$$

The autocorrelation matrix is a symmetric Toeplitz matrix. Hence, the p-simultaneous equations can be solved using Levinson-Durbin algorithm [1].

Once the filter coefficients are computed, the energy of the error signal E_p is given by,

$$E_p = \sum_{n} s^2(n) + \sum_{k=1}^{p} \alpha_k \sum_{n} s(n)s(n-k) \tag{15}$$

$$E_p = R(0) + \sum_{k=1}^{p} \alpha_k R(k) \tag{16}$$

The linear prediction analysis filter transfer function is now given by,

$$A(z) = 1 - P(z) = 1 - \sum_{k=1}^{p} \alpha_k z^{-k} \tag{17}$$

This filter is also known as the inverse filter or the "whitening filter" as it flattens the spectrum of the input speech signal, giving only the noise and impulses as outputs.

3.2 Gain Computation

It can be seen that eq. (2) which models speech production and eq. (4) of linear prediction are similar (after rearranging the terms), but the latter is missing the gain factor G. For the two signals G.u(n) and e(n) to be equivalent, their energies must be equal. Therefore,

$$\sum_n e^2(n) = \sum_n G^2 u^2(n) \tag{18}$$

If u(n) is an impulse function,

$$\sum_n e^2(n) = G^2 \tag{19}$$

$$G^2 = E_p = R(0) + \sum_{k=1}^{p} \alpha_k R(k) \tag{20}$$

3.4 Synthesis

We have seen that the error signal obtained from the inverse filter can be given as in eq. (4), which means,

$$E(z) = S(z)A(z) \tag{21}$$

where $A(z) = 1 - \sum_{k=1}^{p} \alpha_k z^{-k}$

Let us consider another filter with transfer function H(z) as,

$$H(z) = \frac{1}{A(z)} \tag{22}$$

Now, the speech signal can be resynthesized from the error signal as follows.

$$S(z) = H(z)E(z) \tag{23}$$

$$S(z) = \frac{E(z)}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}} \tag{24}$$

$$s(n) = \sum_{k=1}^{p} \alpha_k s(n-k) + e(n) \tag{25}$$

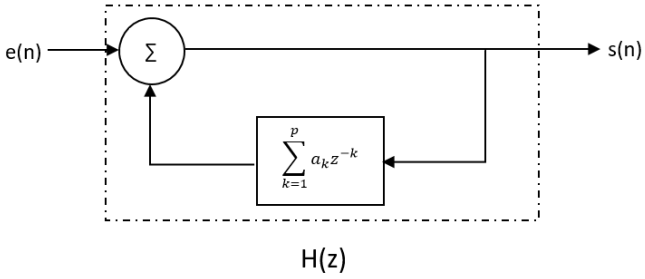

Figure 3: Linear prediction synthesis filter

We see from the eq. (2) and eq. (25) that the filter H(z) functions in the same way as vocal tract. Therefore, H(z) models the response of the vocal tract with e(n) as the excitation source. Effectively, the frequency response of the filter H(z) is the envelop of the speech signal spectrum and the number of poles determines how close the estimation of the envelop is to the actual spectrum itself, i.e. greater the number of poles more accurate is the estimation of the envelope. This can be seen from the filter responses given in fig. (4). This estimation gives the formant peaks on the given speech signal. One must also bear in mind that arbitrarily increasing the number of poles would cause the envelop to model the harmonic characteristics of the signal, which is against the purpose of estimation of the spectral envelop.

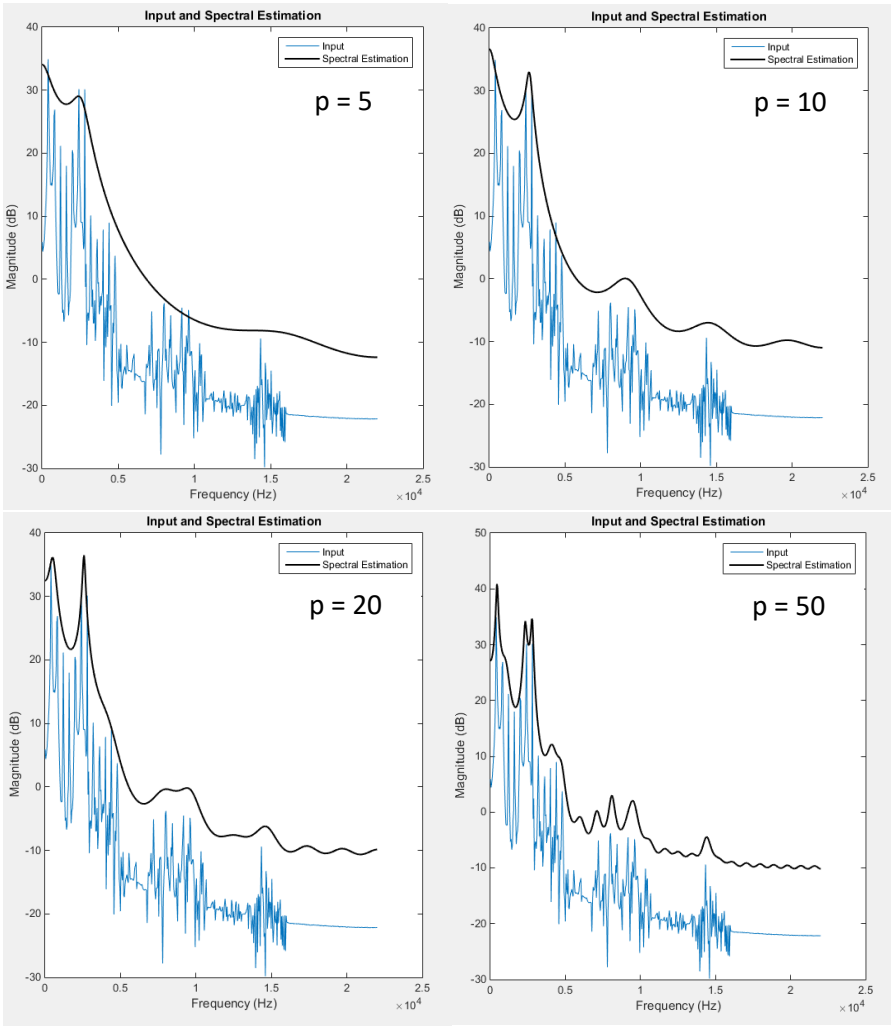

Figure 4: Spectral envelop estimation for different values of p

4. Implementation of Linear Prediction Coding in this Project

In this project, the linear prediction coding algorithm is implemented on Matlab. Due to the time varying nature of speech signals, the implementation of LPC in this project makes use of the overlap and add technique. Since speech signals are considered locally stationary within a duration of 10 to 30 milliseconds, the frame size chosen is of 1024 samples at a sampling frequency of 44100Hz, which is about 23 milliseconds in length. As the window chosen is a hann window of length 1024, the overlap factor is chosen to be greater than 4.

Following is the Matlab function for LPC analysis.

```
%   This function computes the error signal E
%   obtained from linear prediction of given
%   input signal x with the number of poles of
%   the prediction filter p.
function [E, a, g] = lpcAnalysis(x,p)
    overlap = 4;
    fftSize = 1024;
    assignin('base','fftSize',fftSize);
    h = floor(fftSize/overlap);

    e = zeros(fftSize, floor((size(x,1)/h))+1);
    x = [x; zeros(fftSize,1)]; %zero padding for analysis of the last frame
    E = zeros(size(x));        %output error signal
    a = zeros(size(e,2), p+1); %coefficients of inverse filter
    g = zeros(size(e,2),1);

    for i = 1:size(e,2)
        filtBuf = x(((i-1)*h)+1:((i-1)*h)+fftSize);
        R = xcorr(filtBuf);    %find the autocorrelation of input frame
        R = R(fftSize:end);    %truncate the first half
        a(i,:) = levinson(R,p);%find coefficients of filter using
                               %Levinson-Durbin algorithm
        g(i) = sqrt(R(1) + sum(a(i,2:end)'.*R(2:p+1)));
        e(:,i) = (filtBuf - filter([0 -a(i,2:end)], 1, filtBuf))/g(i);
        E(((i-1)*h)+1:((i-1)*h)+fftSize) ...
            = E(((i-1)*h)+1:((i-1)*h)+fftSize) + e(:,i).*hanning(1024);
    end
end
```

Following is the Matlab function for LPC synthesis.

```
%   This function computes the output of the synthesis filter
%   for a given excitation source signal and a given signal
%   and time varying response of synthesis filter.
function y = lpcSynthesis(E, a, g)
    overlap = 4;
    fftSize = 1024;
    h = floor(fftSize/overlap);

    y = zeros(size(E));
    for i = 1:size(a,1)
        filtBuf = E(((i-1)*h)+1:((i-1)*h)+fftSize); %frame to be filtered
        filtOut = filter(g(i), a(i,:), filtBuf);
        y(((i-1)*h)+1:((i-1)*h)+fftSize) ...
            = y(((i-1)*h)+1:((i-1)*h)+fftSize) + filtOut.*hanning(fftSize);
                                                    %Overlap and add
    end
    y = y / max(abs(y));    %Normalize output signal
end
```

Using the error signal and the all-pole filter estimated in the analysis step, various digital audio effects can be implemented. The following section describes one such audio effect.

4.1 Pitch Shifting with Formant Preservation based on Source Filter Separation

Many digital audio effects can be implemented by transforming the error signal and the inverse filter obtained from linear prediction coding. The general structure of such a system would be as follows.

Figure 5: General Structure of an audio processing system based on LPC

where H'(z) is the transformed version of H(z).

The audio effect implemented in this project is "pitch shifting with formant preservation". The idea is to extract the excitation source signal from an audio signal through LPC analysis, process it such that the pitch is altered by a given number of semitones without stretching or compressing it and filter this signal using the filter estimated in the analysis step of LPC to get the pitch shifted version of the input. Since the LPC synthesis filter is unaltered, the spectral envelop remains the same, hence retaining the formant characteristics of the signal.

The pitch shifting algorithm used in this implementation involves first time scaling the error signal obtained from the LPC analysis step to stretch or compress it in time without affecting the frequency characteristics of the signal using the technique described in [4]. This time scaled signal is then resampled such that the output signal length is the same as the input signal length, thereby countering the time-scaling done previously but without frequency coherence. This signal when played at the sampling rate of the input signal would then be a pitch shifted version of the input signal with the pitch shifted by an amount depending on the amount of time scaling.

The time scaling factor 'α' for a pitch shift of 'm' semitones is given as,

$$\alpha = \sqrt[12]{2^m} \tag{26}$$

A signal of length L would be scaled to a length of αL and this new signal must be resampled by a ratio 1: α to result in the pitch shifted signal which is of length L. The Matlab function for pitch shifting is given below.

```
%   This function outputs a pitch shifted version of
%   the input signal x whose pitch is shifted by a
%   specified number of semitones.
function y = pitchShift(x,semitone)
    alpha = pow2(semitone/12);      %time scaling factor
    y = timeScale(x,alpha);         %scaled signal y
    resamp = floor(alpha*10000);
    y = resample(y, 10000, resamp); %pitch shifted output
end
```

Below are some snapshots of the implementation.

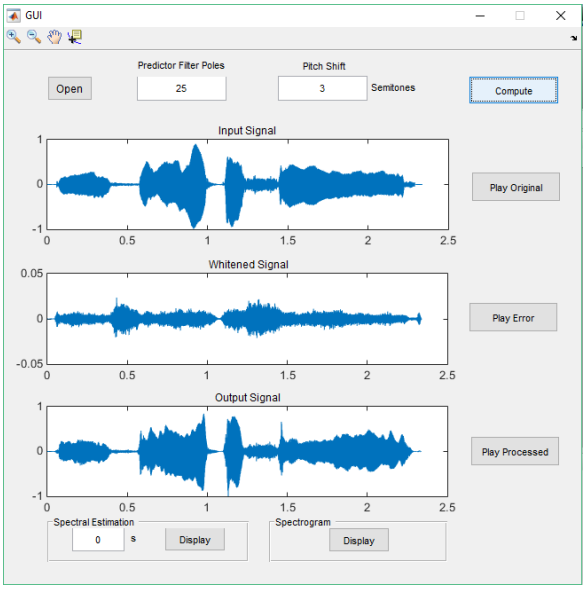

Figure 6: GUI of the pitch shift program

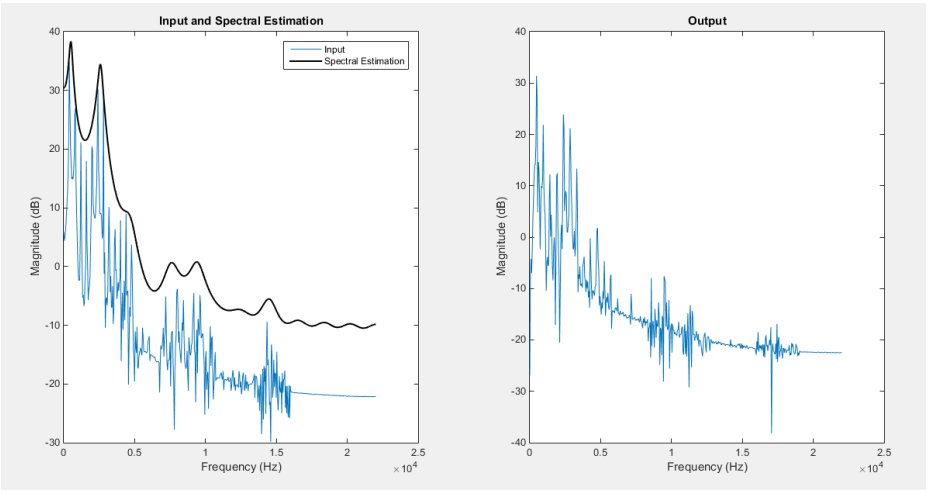

Figure 7: Fourier transform (spectral view) of the input and the pitch shifted output along with the estimated spectral envelop of the input with 25 poles

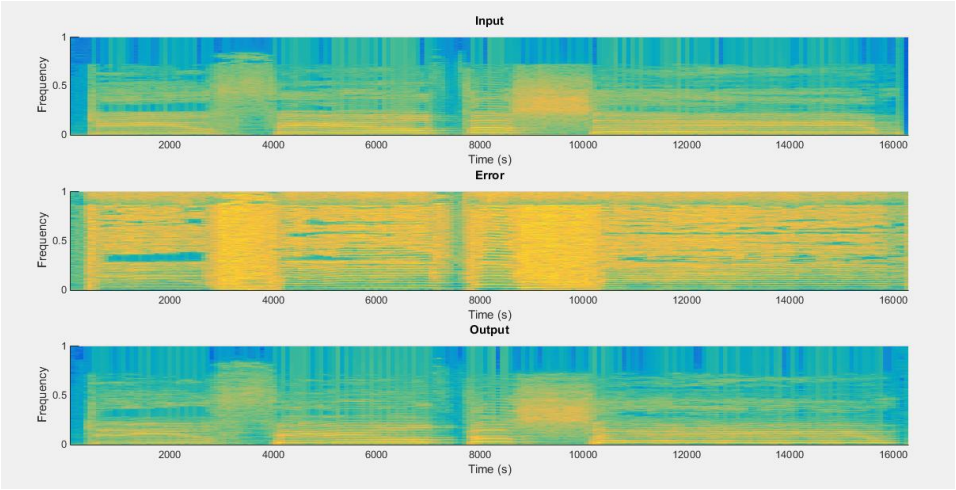

Figure 8: Spectrogram view of the input signal, computed error signal and the pitch shifted output signal

5. Comments on the Implementation of Source-Filter Separation Using Linear Prediction in this Project

The implementation of LPC in this project is only an approximation of the actual linear prediction coding, in which the inverse filter coefficients evolve at audio rate. The FIR filter implementation of the inverse filter in this project only filters the signal frame-by-frame based on the filter coefficients computed by autocorrelation and Levinson-Durbin algorithm for that frame. Moreover, the FIR filter must have access to p past samples typically, where as in this implementation, the filter does not have access to samples outside of the frame and considers them to be 0. Hence, the filter output for the first p input samples of the frame would be incorrect.

In order to obtain a reasonable approximation of the actual functioning of linear prediction, this implementation uses overlap and add technique, because of which the incorrect output samples which occur at the beginning of every frame are attenuated by the windowing function. The effect of this inaccuracy of computation is further reduced by overlapping of subsequent frames where correctly computed samples outweigh the incorrect samples.

Another way to circumvent this problem is by interpolating between filter coefficients computed frame-by-frame to obtain filter coefficients at audio rate. This interpolation, in partial sense, is performed by overlapping and adding implemented here. But, in case of direct form FIR filters (as implemented in this project), interpolating between two stable filters may not necessarily result in stable intermediate filters [3]. The movement of poles and zeros due to interpolation may cause the poles to move outside of the unit circle in the z-plane, which causes instability. However, interpolation of filter coefficients can be stably performed in a factored form filter which is made up of blocks of second order filters. But, this implementation is rather tedious. This problem can be completely done away with in case of lattice form filters, whose reflection coefficients obtained at every frame can be interpolated with stability, thereby realizing a time-varying filter whose response varies at audio rate. These reflection coefficients are in fact computed within the Levinson-Durbin algorithm.

Another problem that arises due to the frame-by-frame evolution of the filter coefficients is the perceivable “roughness” in the output of the filter. This happens due to the sudden variation of filter coefficients at the beginning of every frame. One of the suggested ways to reduce this effect given in [3] is to use an all-pass filter in order to distort the phase, thereby reducing the prominence of the glottal pulse. Another way to reduce this effect is by using a large enough frame [3], but doing so can cause other problems such as loss of time resolution which causes a buzzing effect and finer estimation of the spectral characteristics of the input, which defeats the purpose of spectral envelop estimation.

6. Conclusion

Linear prediction coding has been successfully implemented with some approximation to estimate the spectral envelop of an audio signal. This technique was also successfully used to separate the excitation source and the resonant filter to implement pitch shift with formant preservation. The loop holes of this implementation were also studied and methods to solve the issues encountered were suggested from previous literature.

7. References

[1] J. Makhoul, "Linear prediction: A tutorial review," in *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561-580, April 1975. doi: 10.1109/PROC.1975.9792

[2] Arfib, D., Keiler, F. and Zölzer, U. (2002) Source-filter Processing, in DAFX: Digital Audio Effects (ed U. Zölzer), John Wiley & Sons, Ltd, Chichester, UK. doi: 10.1002/047085863X.ch9

[3] J. Moorer, “The Use Linear Prediction of Speech in Computer Music Applications” in JAES Volume 27 Issue 3, pp. 134-140, March 1979.

[4] Puckette, Miller. (1995). Phase-locked vocoder. 222 - 225. 10.1109/ASPAA.1995.482995.