# EEEM030 Assignment 1

# Speech Production Modelling with Linear Prediction

Yi Zhou 6567008

## 1. Speech Production Model

Figure 1 shows the model structure of speech production. The input signal to the system is generated by an excitation generator. There are generally two kinds of input signal: voiced speech, which can be modelled as a periodical impulse train, and unvoiced speech, which can be modelled as random noise.

From the physical perspective, the vocal tract can be modelled as a time-varying dynamic system. However, since it changes its shape slowly, we can simplify it as an LTI system by adding an observing window. The window size is normally chosen on the order of 10ms. The output of the LTI system is the speech signal.

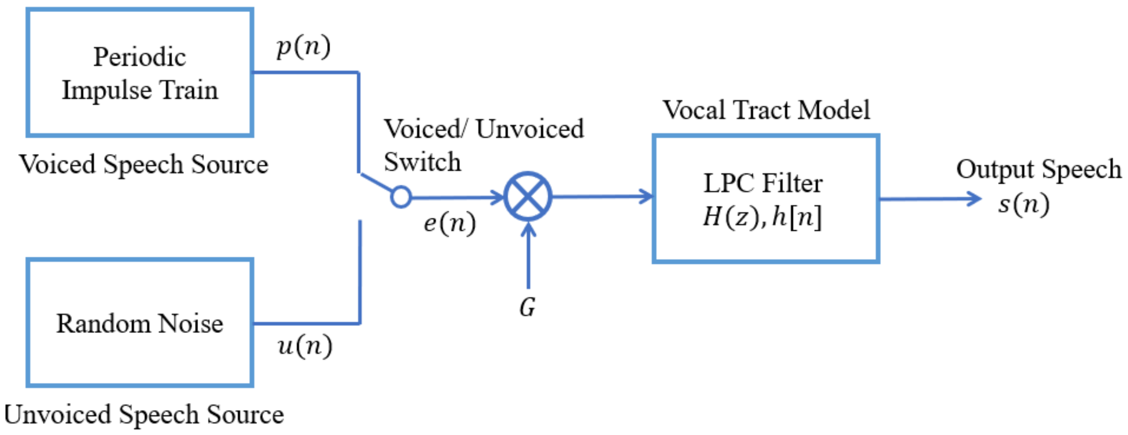


Figure 1 Source Filter Model

# 2. Linear Predictive Analysis (LPC)

## 2.1 LPC overview

The most common used speech analysis technique is called Linear Predictive Coding (LPC). It firstly convolutes the speech sequence with a shifted window, e.g. Hamming window, to get an LTI system. Then, it assumes an all-pole structure. Equation (2.1) and (2.2) show the transfer function and its corresponding difference equation,

$$H(z) = \frac{S(z)}{E(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{2.1}$$

$$s(n) = \sum_{k=1}^{p} a_k s[n-k] + Ge[n] \tag{2.2}$$

where $s(n)$ and $e(n)$ are the speech signal and excitation signal respectively. $G$ and set of $\{a_k\}$ are the regression parameters.

Since the excitation signal $e(n)$ is modelled as a slow periodic impulse, we can further include it in the error signal. Then, the Autoregressive (AR) model is given by

$$\hat{s}(n) = \sum_{k=1}^{p} a_k s[n-k] + d(n) \tag{2.3}$$

where $\hat{s}(n)$ is the predicted speech signal and $d(n) = Ge(n)$ is the predicted error.

To estimate the weight set $\{a_k\}$ and the gain $G$, we can either use auto-correlation method or the covariance method. In this report, I choose to use auto-correlation method. The general idea is to minimize the mean squared error (MSE) between the speech signal $s(n)$ and the estimated signal $\hat{s}(n)$ over a time segment, which is shown in equation 2.4.

$$MSE = \sum_{m} d^2(m) \tag{2.4}$$

After the weight set is calculated, the gain can be determined through energy matching criterion, i.e. energy in error signal equals energy in excitation.

It should be noticed that, there are two main parameters for LPC model, i.e. the order $k$ and the segment length $m$. In the following sections, I will discuss the selection of these two parameters.

## 2.2 Segment Length Selection

From equation 3, it should be noticed that, we need make some zero paddings if the index exceeds the window size, which yielding to large errors at the starting process. Therefore, it is reasonable to use a Hamming window (Figure 2) to reduce the weights at two ends. In frequency domain, Hamming window actually serves as a low-pass filter. We select the window size as the segment length. It can be proved that the frequency width is inversely proportional to the time width [1]. Thus, the longer the segment is, the more detailed frequency domain resolution we could get, and vice versa.

In Figure 3, if the window size is too small, we can get a smooth curve in the spectral. If the window is very large, in the STFT analysis, we can get the formant frequencies as well as their harmonics. Since the response of LPC is determined by its own order, the spectral envelope is almost unchanged if we take different window sizes. However, in time domain, if we take a too large window, the LTI system assumption may not hold. This will result in a large error in the discontinuous point. Herby, the maximum of window size is always limited to 10 to 30 milliseconds, within which the properties of most speech signals are relatively constant [1].

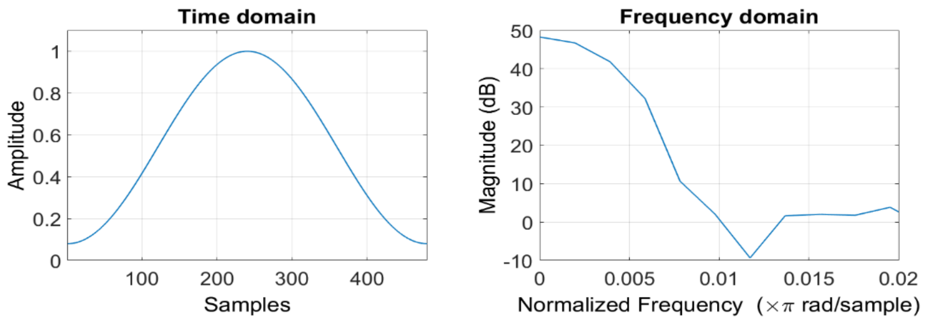


Figure 2 Hamming Window in Time Domain and Frequency Domain

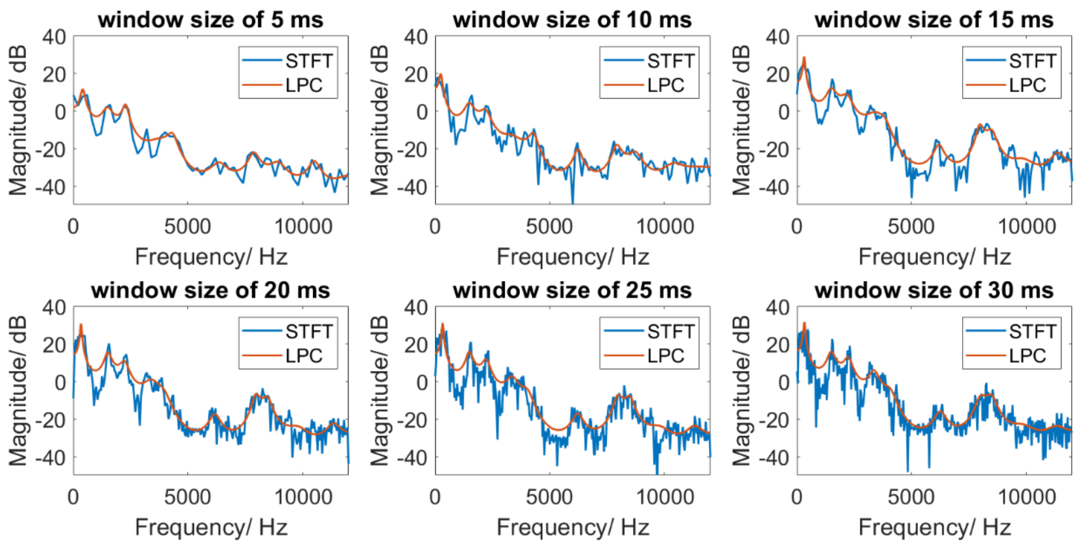


Figure 3 LPC and STFT Spectrum with Different Window Sizes

## 2.3 Order Selection

In the experiment, I choose candidate order set of [10, 30, 50]. From Figure 4, we can find the higher the order, the smaller the MSE of error signal. However, when the order is greater than 30, the model tends to overfits the input signal. From Figure 5, we can see that it is hard to identify the formant frequency from the order of 10. The order of 30 is sufficient for identifying the first several peaks. The higher order cases tend to fit the high-order unnecessary resonances. In addition, the higher order yields to more computation efforts. Thus, the selection of order could be among 20-30.

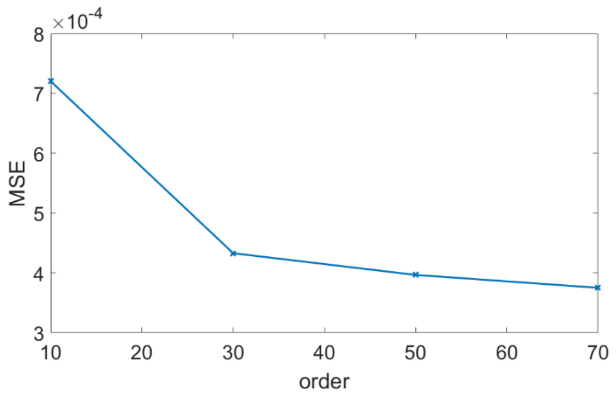


Figure 4 MSE of Residual Signal with Different Orders

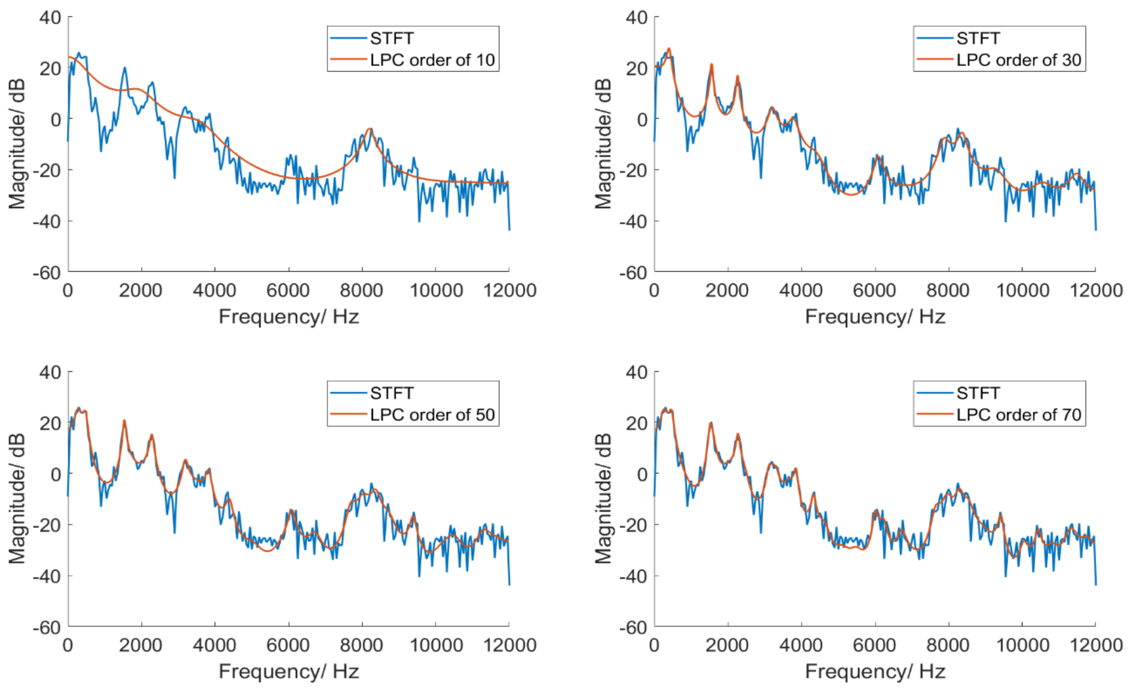


Figure 5 LPC and STFT Spectrum with Different Orders

## 2.4 LPC Coefficients and Gain

According to the requirements of this assignment, I select two vowel samples, “hood” for male voice and female voice. The characteristics of the input speech signal is summarized in Table 2.1. Then I select order of 20 and order of 30 for man and woman voice. The segment length and shift step are selected as 20 milliseconds and 10 milliseconds. The LPC coefficients are computed using the auto-correlation method. The results are shown in Table 2.2.

| | Male | Female |
|---|---|---|
| Vowel | hood | |
| Original Length | 184.5 ms | 241.3 ms |
| Sample Rate | 24000 Hz | |
| Clipped Length | 100 ms | |

Table 2.1 The Characteristics of Input Speech Signal

| | Male | Female |
|---|---|---|
| Window Size | 20 ms | 20 ms |
| Window Shift | 10 ms | 10 ms |
| Order | 20 | 25 |
| LPC Coefficients | [1.0000,    1.9484, 1.0101, 0.1425, 0.4935, -0.9980, 0.3433, 1.0609, 0.0943, -0.8405, 0.5035, -0.6385, 0.8000, -0.2443, -0.1413, 0.0426, -0.3022, 0.7235, -0.4128, -0.1298, 0.1748] | [1.0000, -1.0926, -0.6629, 0.6138, 0.8635, -0.7359, -0.4110, 0.0053, 0.9923, -0.1341, -0.7535, -0.2833, 1.0364, 0.0131, -0.6850, -0.2296, 0.7639, -0.1059, -0.1492, -0.2342, 0.0746, 0.3217, 0.1748, -0.5234, 0.0956, 0.0680 ] |
| Gain | 0.2589 | 0.4826 |

Table 2.2 LPC Parameters

# 3. LPC Results Analysis

In section 2, the LPC model is determined. In this section, some characteristics of the speech is analysed with the help of this model.

## 3.1 Formant Frequency

Formant frequency represents the resonance of vocal tract. Formant frequencies correspond to the peaks of the spectral envelope of the LPC filter. In this report, the task is to extract the first three formant frequencies. Firstly, we calculate the poles of the transfer function. Figure 6 shows the pole-zero map of the LPC transfer function. The conjugate pole pairs close to the unit circle model the formant frequency. We calculate the angles of the poles and convert them into frequencies. In Figure

7, the locations of the first three smallest frequencies are marked as the red circles. We can see that the circles are consistent with the peaks in frequency spectral envelope. The resulting formant frequencies are (336.1, 1504.9, 2240.5) and (273.4, 1594.16, 2531.7) for man and woman voice respectively. Comparing the first few peaks in Figure 7, the formant frequency of female voice is very similar to that of male voice.

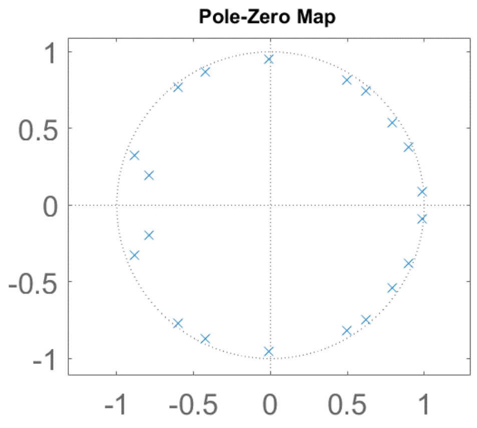


Figure 6 Pole-Zero Map of the LPC Transfer Function

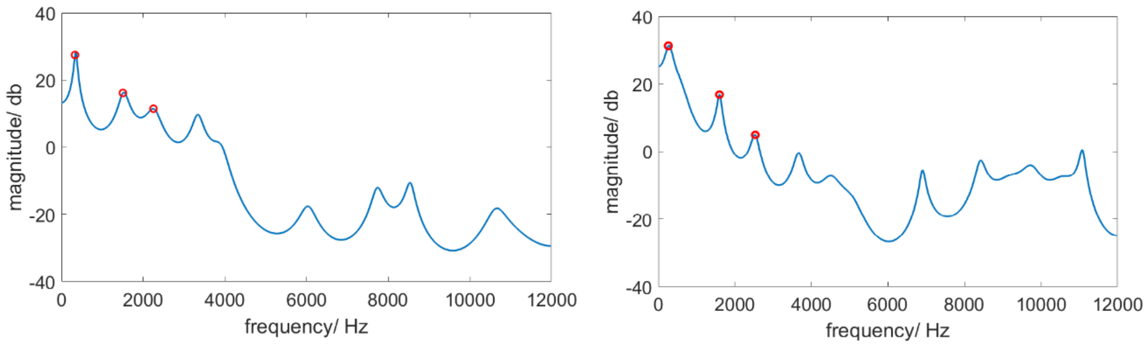


Figure 7 Formant Frequency in LPC Spectrum (Male & Female)

## 3.2 Fundamental Frequency

Fundamental frequency of the speech is determined by the frequency of the periodic excitation signal. It can be calculated from auto-correlation of the original speech signal. To reduce the computational effort, I down-sample the signal in the scale of 5 before doing auto-correlation. Figure 8 shows the correlation responses. Each peak indicates the $i^{th}$ order of harmonic frequency $f_i$, where $f_i = i * f_0$. Then we can take difference and average the intervals to get the fundamental frequency $f_0$. In this case, the values of $f_0$ are 93.85 Hz and 196.49 for male voice and female voice respectively. The result meets the common sense that the voice frequency of female is higher than that of man.

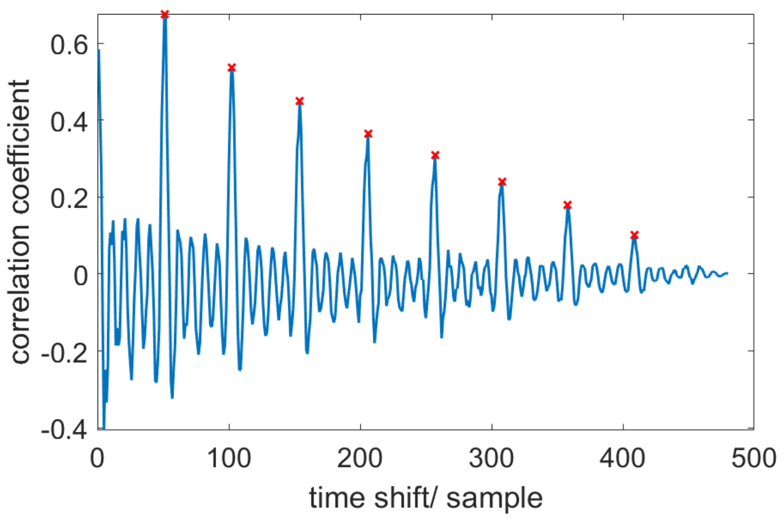


Figure 8 Auto-Correlation of The Down-Sampled Speech Signal

Furthermore, there is an alternative way to fundamental frequency using error signal, which will be introduced in the following section.

## 3.3 Error Analysis

From equation 2.3, we can get the error signal $d$ as

$$d(n) = s(n) - \sum_{k=1}^{p} a_k s[n-k] \tag{3.1}$$

Then, the prediction error filter, called inverse filter, can be defined in the form of

$$A(z) = \frac{D(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k} \tag{3.2}$$

Figure 9 shows the reconstructed speech and error signal as a result of the LPC for man voiced. We can find the reconstructed speech matches the original one very well. In an ideal case, the error signal is composed of the general background noise and the impulse excitation. From Figure 9, we can see there is periodical peak in the error signal, which indicates the excitation. Thus, we can use the error signal to predict the fundamental frequency as well. From Figure 10, we can find that there is only one peak which is above the threshold. By this way, the computing is more convenient. The fundamental frequency computed in this method is 96 Hz, which is roughly the same as the previous result.

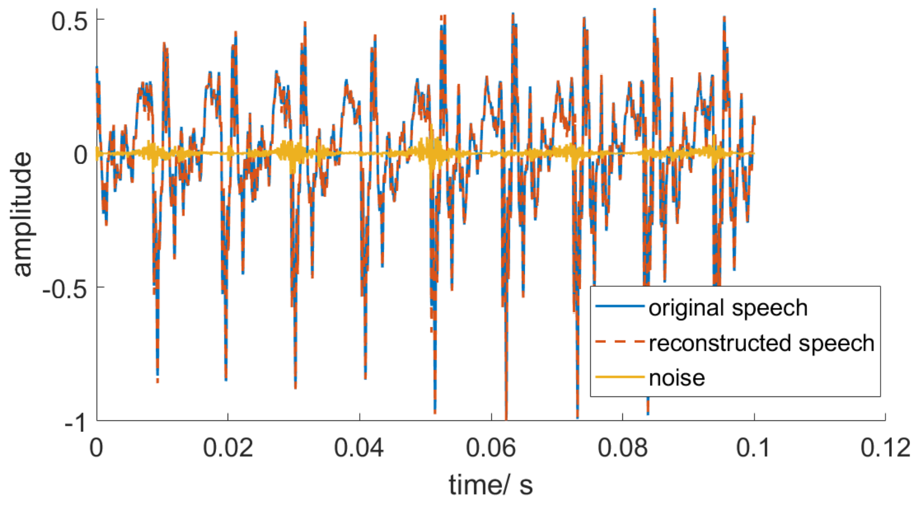


Figure 9 Reconstructed Speech Signal and Error Signal

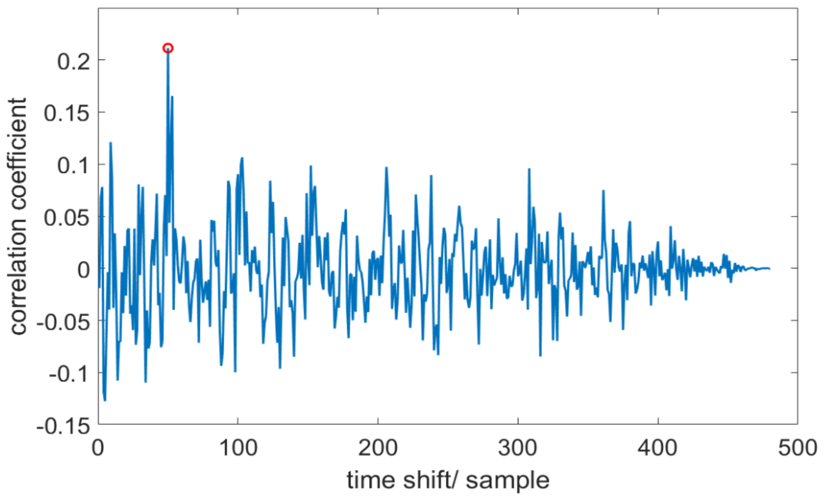


Figure 10 Auto-Correlation of The Error Signal

If listening to the error signal, we could find it is similar to the original speech rather than white noise, indicating the unmodelled dynamics in the residual signal. To further increase the accuracy, we can try a more complex model, for example, adding some zeros.

## 4. Speech Synthesis

In the previous sections, we fit an LPC model and use it to make some analysis. We can also use it to synthesize the speech signal.

## 4.1 LPC Synthesizer

Figure 11 shows the LPC synthesis model. It is the same as speech production model mentioned in Figure 1. For the voiced speech, the excitation is a designed impulse train. Then, the synthesized speech can be represented as

$$S(z) = H(z)E(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} E(z) \tag{4.1}$$

$$s(n) = \sum_{k=1}^{p} a_k s[n-k] + Ge[n] \tag{4.2}$$

where $s(n)$ and $e(n)$ are the speech signal and excitation signal respectively. $G$ and set of $\{a_k\}$ are the regression parameters.

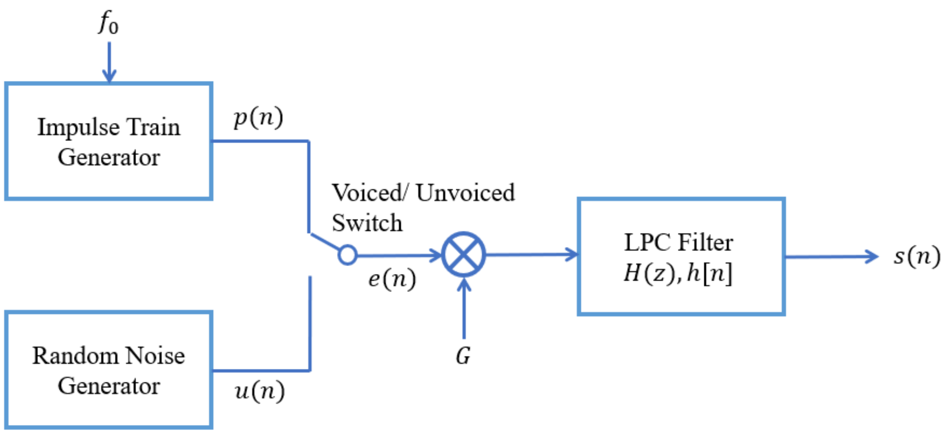


Figure 11 LPC Synthesis Model

According to the model, the first step is to generate the excitation signal. As mentioned in paragraph 3.2, the periodic frequency of excitation signal should equal the fundamental frequency of the original speech. Then, such impulse trains with length of 100 milliseconds are generated for both male and female voice respectively. The LPC filter is constructed use the LPC coefficients and gain mentioned in Table 2.2 . In each time step, within the time window, we apply the LPC filter with the excitation signal. Repeat this until reaching the end of the sample length.

## 4.2 Results Analysis

The output synthesized speech displays a similar pitch compared to the original one. However, in most of the cases, the generated speech is very buzzy and thus unrealistic.

Figure 12 and Figure 13 show the input impulse excitation and output synthesized speech. From these figures, there are several observations. Firstly, the fundamental frequency is roughly the same to the original one. Secondly, the curve is smoother than the real speech. Thirdly, the amplitude is about 2-3 times larger the original speech, indicating the gain is larger than expected. As mentioned in section 2.4 and 3.3, the gain is computed under the assumption of energy matching criterion. Since there are unmodelled dynamics in the error signal, the error energy should be larger than assumption. As a result, the gain will also be larger. Ideally, since the system is linear, the gain will only affect the speech amplitude. And this small amplitude increase is even too small to sensible.

In fact, the ideal excitation signal should be the error signal generated in section 3.3. Since the error signal do not meet the assumption, it is not suitable to use a simple impulse train as input. Perhaps, we should try more complex model or add more features in the excitation signal. In short, the LPC synthesizer is not good enough.

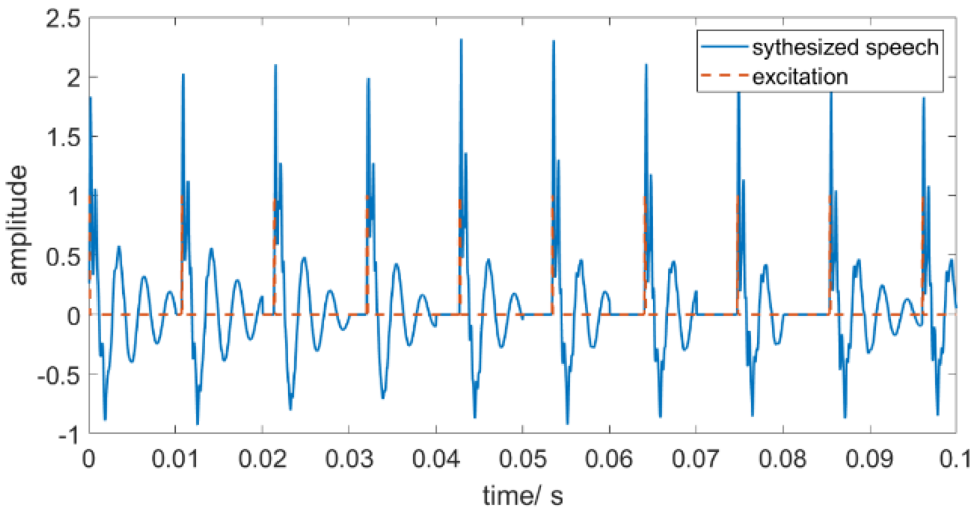


Figure 12 The Synthesized Vowel (Male)

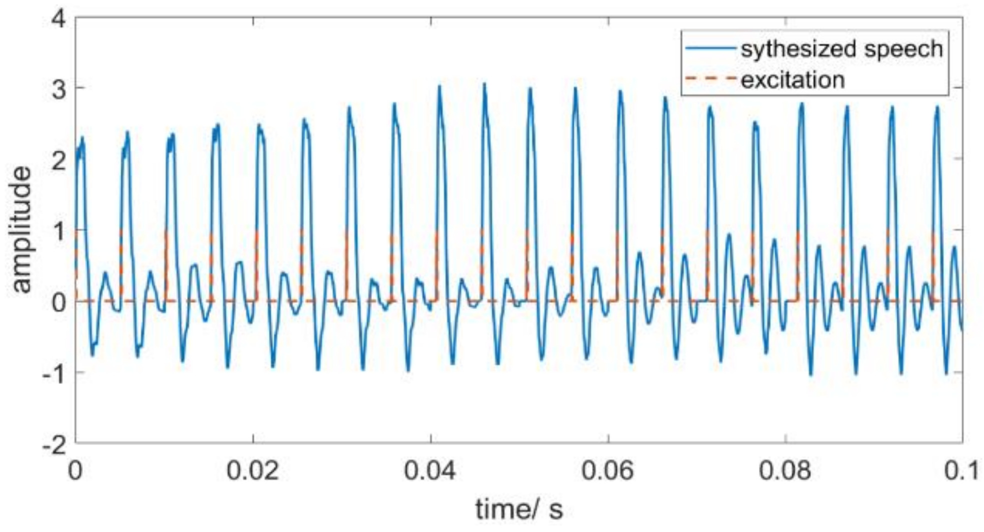


Figure 13 The Synthesized Vowel (Female)

## 5. Conclusion

In this report, we study LPC and its applications. We can see that LPC is a very effective method for speech modelling. There are two parameters to tune. i.e. the order and the window size.

The order is related to both the fitting to the speech signal in time domain and the fitting to the spectral shape in frequency domain. Too little order will underfit the input signal and make identification of formant frequency hard, while too large order will overfit the input signal and model unnecessary resonances. The normal selection is among 20-30.

When choosing the window size, we need consider the trade-off between time-resolution and spectrum resolution. The normal choice is within 10 to 30 milliseconds.

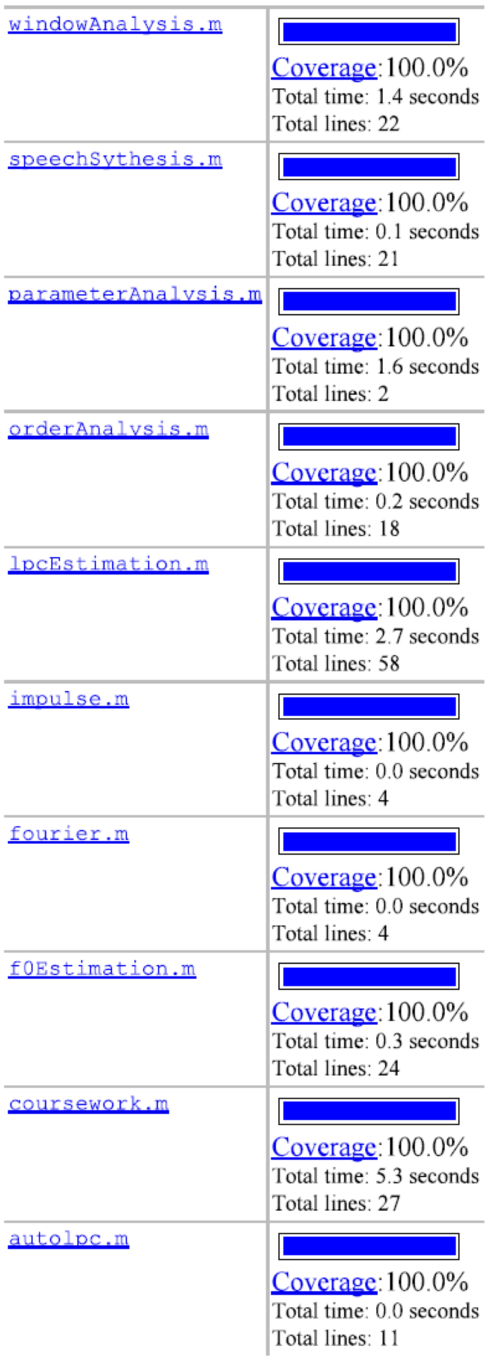
LPC is suitable for formant frequency estimation. Its error signal can be used in fundamental frequency estimation. However, the performance of LPC synthesizer is not good enough.

## Reference

[1] L. Rabiner and R. Schafer, *Theory and applications of digital speech processing*. Upper Saddle River: Pearson, 2011.

# Appendix: MATLAB Codes

In order to finish the assignments, I write a MATLAB program with one script and 9 functions. Every function is well-tested to ensure the robustness to different input speech signals. The code analysis of the program is shown in Figure 14. It should be noticed that the plotting process is very time-consuming. Thus, I add a flag called "Plot" in the main script. We can turn of the flag to directly get the synthesized speech.



| | |
|---|---|
| windowAnalysis.m | Coverage:100.0%<br>Total time: 1.4 seconds<br>Total lines: 22 |
| speechSythesis.m | Coverage:100.0%<br>Total time: 0.1 seconds<br>Total lines: 21 |
| parameterAnalysis.m | Coverage:100.0%<br>Total time: 1.6 seconds<br>Total lines: 2 |
| orderAnalysis.m | Coverage:100.0%<br>Total time: 0.2 seconds<br>Total lines: 18 |
| lpcEstimation.m | Coverage:100.0%<br>Total time: 2.7 seconds<br>Total lines: 58 |
| impulse.m | Coverage:100.0%<br>Total time: 0.0 seconds<br>Total lines: 4 |
| fourier.m | Coverage:100.0%<br>Total time: 0.0 seconds<br>Total lines: 4 |
| f0Estimation.m | Coverage:100.0%<br>Total time: 0.3 seconds<br>Total lines: 24 |
| coursework.m | Coverage:100.0%<br>Total time: 5.3 seconds<br>Total lines: 27 |
| autolpc.m | Coverage:100.0%<br>Total time: 0.0 seconds<br>Total lines: 11 |

Figure 14 Code Report by MATALB "profview" tool

## coursework.m

```
clear all;
close all;
clc;
addpath("speech");
list = dir("speech/*.wav");
global Plot; % flag to turn on/off the plot
Plot = false;
%% preprocessing
vowelIndex = 11;
fileName = list(vowelIndex).name;
[vowel, Vowel.sampleRate] = audioread(fileName);
sound(vowel, Vowel.sampleRate);
timeSpan = 0.1 * Vowel.sampleRate;
Vowel.vowel = vowel(680:680 + timeSpan);
%% lpc model
nFormants = 3;
order = 20;
windowSize = int32(0.02 * Vowel.sampleRate);
windowShift = int32(0.01* Vowel.sampleRate);
[Model, Vowel.formant, Vowel.f0] = lpcEstimation(Vowel.vowel, windowSize,
windowShift, order, 1/Vowel.sampleRate, nFormants);
%% model parameter analysis
if Plot
    orderSequence = 10:10:30;
    windowSzieSequence = int32((0.005:0.005:0.03) .* Vowel.sampleRate);
    parameterAnalysis(Vowel.vowel,windowSize,order,Vowel.sampleRate,windowS
    zieSequence,orderSequence);
end
%% sythesis
vowelSythesized = speechSythesis(Model, 1/Vowel.f0,
timeSpan/Vowel.sampleRate);
sound(vowelSythesized, Vowel.sampleRate);
fileName = [fileName(1:end-4),'_sythesized.wav'];
audiowrite(fileName,vowelSythesized,Vowel.sampleRate);
```

## lpcEstimation.m

```
function [model, formant, f0] = lpcEstimation(vowel, windowSize, shift, order, ts,
nFormants)
global Plot;
model.sampleTime = ts;
model.windowSize = windowSize;
model.windowShift = shift;
t = model.sampleTime * (1:size(vowel));
```

```
window = hamming(windowSize);
ind = 0;
model.lpcCoeffs = zeros((length(vowel)-1)/shift -1, order+1);
model.gain = zeros((length(vowel)-1)/shift -1, 1);
noise = zeros(size(vowel));
formant = zeros((length(vowel)-1)/shift -1, 3);
%% f0 frequency
f0 = f0Estimation(vowel, 1/ model.sampleTime);
%% lpc
for i = 1:shift:length(vowel)-windowSize + 1
    ind = ind + 1;
    vowelSegment = vowel(i:i+windowSize-1);
    vowelSegment = vowelSegment .* window;
    [model.lpcCoeffs(ind,:), model.gain(ind)] = autolpc(vowelSegment, order);
    noise(i:i+windowSize-1) = filter(model.lpcCoeffs(ind,:),1, vowelSegment);
    [h, w] = freqz(model.gain(ind), model.lpcCoeffs(ind,:));
    w = w/pi/2/model.sampleTime;
    magnitude = 20*log10(abs(h));
%% pole-zero map
    lpcPoles = roots(model.lpcCoeffs(ind,:));
    sys = tf(model.gain(ind),model.lpcCoeffs(ind,:), 'Ts', model.sampleTime);
%% formant
    ang = atan2(imag(lpcPoles),real(lpcPoles))/pi/2/model.sampleTime;
    ang = sort(ang(find(ang>0)));
    formant(ind,:) = ang(1:nFormants);
    wList = sort([ang(1:nFormants);w]);
    for i = 1:nFormants
        peakIndex(i) = find(wList==formant(ind,i))-i;
    end
end
sound(noise,    1/ts);
%% fundamental frequency
f0_alternative = f0Estimation(noise, model.sampleTime);

if Plot
    figure
    pzmap(sys);
    ax = gca;
    ax.FontSize = 16;
    axis equal

    figure
    plot(w, magnitude, 'LineWidth',1.5)
    hold on
    plot(w(peakIndex), magnitude(peakIndex),'ro', 'LineWidth',1.5)
    xlabel('frequency/ Hz')
    ylabel('magnitude/ db')
    ax = gca;
    ax.FontSize = 16;
    figure
    hold on
    plot(t, vowel,'LineWidth',1.5)
    plot(t, vowel-noise,'--','LineWidth',1.5)
    plot(t, noise,'LineWidth',1.5)
    legend('original speech', 'reconstructed speech', 'noise')
    xlabel('time/ s')
    ylabel('amplitude')
    ax = gca;
    ax.FontSize = 16;
end
```

## f0Estimation.m

```
function f0 = f0Estimation(vowel, sampleRate)
global Plot
scale = 5;
vowelDwonSample = downsample(vowel,scale);
[c, lags] = xcorr(vowelDwonSample, 'coeff');
c = c(lags > 0);
lags = lags(lags > 0);
ind = 1;
peakIndex(ind) = find(c == max(c));
for i = peakIndex(1):peakIndex(1):length(c)-5
    peakIndex(ind) = find(c == max(c(max(i-5,1):i+5)));
    ind = ind + 1;
end
peakIndex = peakIndex(1:8);
f0 = 1/(mean(diff(peakIndex))* 5) * sampleRate;

if Plot
    figure
    plot(lags, c,'LineWidth',1.5)
    hold on
    plot(lags(peakIndex), c(peakIndex),'rx' ,'LineWidth',1.5)
    xlabel('time shift/ sample')
    ylabel('correlation coefficient')
    ax = gca;
    ax.FontSize = 16;
end
```

autolpc.m

```
function [A,G]=autolpc(x,p)
    L=length(x);
    r=[];
    for i=0:p
        r=[r; sum(x(1:L-i).*x(1+i:L))];
    end
    R=toeplitz(r(1:p));
    a=inv(R)*r(2:p+1);
    A=[1; -a];
    G=sqrt(sum(A.*r));
    A = A';
end
```

## fourier.m

```
function [f, m] = fourier(t, sampleTime)
s = fft(t);
s = s(1:length(t)/2+1);
f = 1/sampleTime * (0:length(t)/2)/length(t);
m = 20*log10(abs(s));
```

## parameterAnalysis.m

```
function
parameterAnalysis(vowel,windowSize,order,sampleRate,windowSzieSequence,order
Sequence )
% test the effect of different order
orderAnalysis(vowel,windowSize, orderSequence, 1/sampleRate)
% test the effect of different window size
windowAnalysis(vowel,windowSzieSequence, order, 1/sampleRate)
```

## orderAnalysis.m

```
function orderAnalysis(vowel,windowSize, orderSequence, sampleTime)
%% add window
window = hamming(windowSize);
vowel = vowel(1:windowSize);
vowel = vowel .* window;
%% frequency response
[f, m] = fourier(vowel, sampleTime);
figure
plot(f, m,'LineWidth',1.5)
xlabel('Frequency/ Hz')
ylabel('Magnitude/ dB')
hold on
%% test different order of lpc
for i = 1:length(orderSequence)
```

```
        [lpcCoeffs, gain] = autolpc(vowel, orderSequence(i));
        [h, w] = freqz(gain,lpcCoeffs);
        w = w/pi/2/sampleTime;
        plot(w,20*log10(abs(h)),'LineWidth',1.5)
end
legend('STFT','LPC order of 10','LPC order of 20','LPC order of 30')
ax = gca;
ax.FontSize = 16;
```

## windowAnalysis.m

```
function windowAnalysis(vowel,windowSizeSequence, order, sampleTime)
figure
hold on
for i = 1:length(windowSizeSequence)
        window = hamming(windowSizeSequence(i));
        vowelSeg =    vowel(1:windowSizeSequence(i));
        vowelSeg = vowelSeg .* window;
        %% frequency response
        [f, m] = fourier(vowelSeg, sampleTime);
        subplot(2,length(windowSizeSequence)/2,i)
        plot(f, m,'LineWidth',1.5)
        hold on
        axis([0 12000 -50 40])
        %% lpc model
        [lpcCoeffs, gain] = autolpc(vowelSeg, order);
        [h, w] = freqz(gain,lpcCoeffs);
        w = w/pi/2/sampleTime;
        plot(w,20*log10(abs(h)),'LineWidth',1.5)
        legend('STFT','LPC')
        xlabel('Frequency/ Hz')
        ylabel('Magnitude/ dB')
        title("window size of "+ double(windowSizeSequence(i))*sampleTime*1000 + "
ms")
        ax = gca;
        ax.FontSize = 16;
end
```

## impulse.m

```
function signal = impulse(len, interval, span)
signal = zeros(len, 1);
for i = 1:interval:len
        signal(i:i+span) = 1;
end
```

## speechSythesis.m

```
function vowelSythesized = speechSythesis(model, intervalTime, timeSpan)
global Plot
timeLast = model.sampleTime;
%% generate impulse exicitation
exicitation =
impulse(int32(timeSpan/model.sampleTime),int32(intervalTime/model.sampleTime),i
nt32(timeLast/model.sampleTime));
%% speech sythesize
vowelSythesized = zeros(length(exicitation), 1);
ind = 0;
for i = 1:model.windowShift:length(exicitation)-model.windowSize + 1
        ind = ind + 1;
        vowelSythesized(i:i+model.windowSize-1) = filter(model.gain(ind),
model.lpcCoeffs(ind,:), exicitation(i:i+model.windowSize-1));
end

%%
if Plot
        figure
        plot((model.sampleTime:model.sampleTime:timeSpan),vowelSythesized,
'LineWidth', 1.5);
        %axis([0,0.1,0,1.5])
        xlabel('time/ s')
        ylabel('amplitude')
        ax = gca;
        ax.FontSize = 16;
        hold on
        plot((model.sampleTime:model.sampleTime:timeSpan),exicitation,'--',
'LineWidth', 1.5);
        legend('sythesized speech','excitation')
end
end
```

p.s. My philosophy of commenting is using the detailed names + little comments instead of abstract name+comments. Because almost all modern editors have the tab completion function. Thus, the advantage of readability outweighs the disadvantage of inconvenience. But I admit that the docstring is very useful for reusability.