# Linear Predictive Speech Synthesizer

Kongkea Ouch

**Abstract**

Linear Predictive Coding (LPC) [1] is widely considered as one of the most popular and useful methods used in speech processing and speech synthesis to provide a reasonably good quality speech at a minimal bit rate which is beneficial for low bandwidth and low size application. It takes weight summation of past samples of a speech signal in order to predict the next sample of the signal. This report aims to explore the LPC technique in the source-filter model of speech production and describe an implementation in MATLAB program based on LPC to synthesize real vowel speech sample as well as final results and discussion of the speech output.

## Introduction

The process of speech production in humans [1] can roughly be summarized as air pressure being pushed from the lungs through the vocal folds opening and closing up to the vocal tract, and output through the mouth to produce speech. Here, the lungs can be considered of as the sound's source and the vocal tract can be said as a filter that shapes and generates various types of sounds that formulate speech. There are two types of sounds, voiced and unvoiced sounds, which are generally considered in the linear speech coding and synthesizing.

Voiced sounds [1] are often vowels which are generated by the flow of air pushed from the lungs causing a quasi-periodic vibration of the vocal folds that result in a periodic train of glottal pulses with an audible pitch. The faster the vocal folds vibrate the higher pitch the sound outputted. The vibrations are then picked up and resonated by the vocal tract which form formants or peaks and these occur at the formant frequencies. Women and children's voices typically have higher pitches than men because their vibrations are faster than that of men during the speech production. Unliked voiced sounds, unvoiced sounds [1] are often consonants created by the constriction of the vocal tract forcing out a burst of noise and turbulence and are not stemmed from the vocal folds' vibrations hence the lack of pitch. Therefore, it is essential to incorporate the pitch period in the excitation signal of the impulse train to synthesize a vowel speech that closely sounds like the original speech.

The source-filter model separately treats the airflow from the lungs as a sound source and the vocal tract as a filter that shapes the sound. In the Linear Predictive Coding (LPC) model, which is based on the source-filter model, linear prediction is generally used to model vowel and low bit rate speech. From an original speech sample, LPC [1] efficiently uses the autocorrelation method of autoregressive (AR) modelling to calculate a set of filter coefficients which is an all-pole model that imitates the vocal tract's resonance system which leads to a simplified representation of the formant structure. Then, a simulated periodic impulse train made up of fundamental frequencies from the voiced sound is filtered with the all-pole filter to produce a synthesized version of the real speech signal.

## Implementation

The linear speech synthesizer program is implemented in MATLAB and it is consisted of two reusable functions for vowel segment identification (`identify_segment`) and synthesis (`lp_speech_synthesizer`) as well as a main execution workspace (`main`) to run each function as fit.

The vowel speeches chosen for synthesis are 'heed_m' for male vowel and 'heed_f' that correspond to 'i:' vowel sound and a quasi-stationary 100ms segment from each sound is extracted for analysis and synthesis. The base LPC order is 25 as normally the order between 20 and 30 is enough to model the first few formants [1]. The base configuration of mentioned LPC order and segment length and changes is used for main synthesis of the report and changed later for different evaluations.

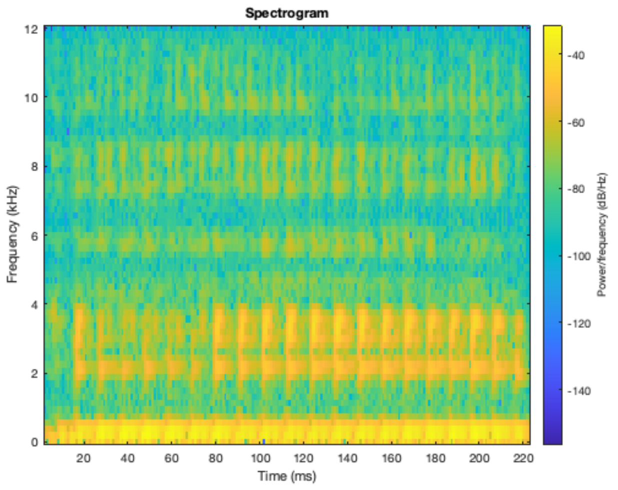

*Figure 2: Male spectogram*

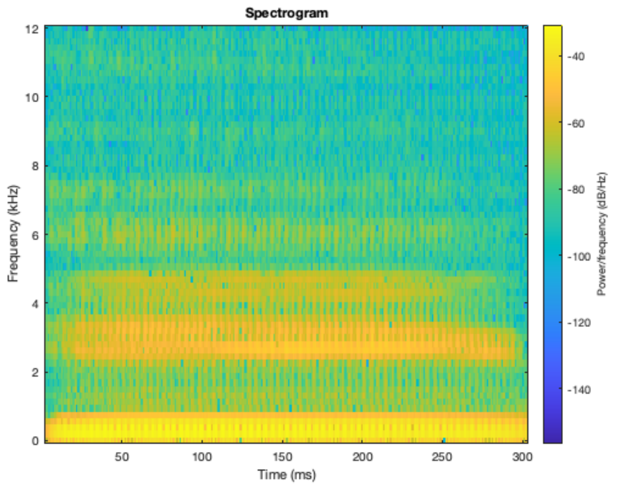

*Figure 1: Female spectogram*

To determine a voiced segment for extraction, the `identify_segment` function is employed to fetch sound information (both sounds have sample rate of 24 kHz and bit rate of 16) and observe with built-in `spectrogram` function as applied in **Error! Reference source not found.** and Figure 2 which part of the original sounds has the most consistent energy over time. Combined with headphone listening, 30ms is considered to be the best starting point for a quasi-stationary segment of both male and female sounds in this analysis.

The `lp_speech_synthesizer` function then is called to start the process of analysing and synthesizing each vowel sound of male and female by accepting different file input. Sampled data and sample rate are extracted through `audioread` method and later used for vowel segment extraction from the determined starting point up to the input duration. From waveform plotting of the segments in Figure 4 and Figure 3, it can be seen that both share a trend of periodic signals and that the female sound generally has higher signal amplitude than the male sound.

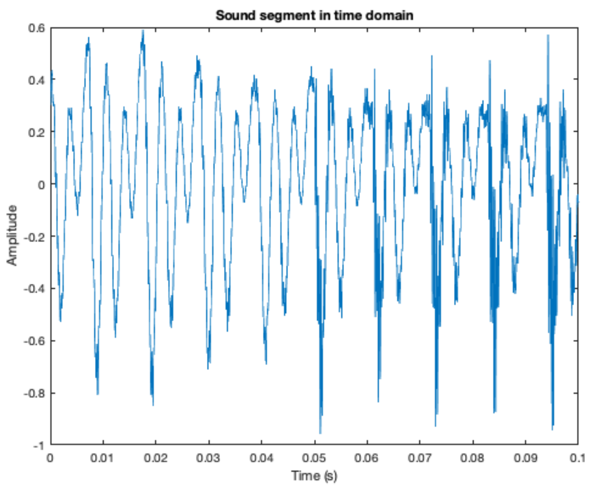

*Figure 3: Male waveform*

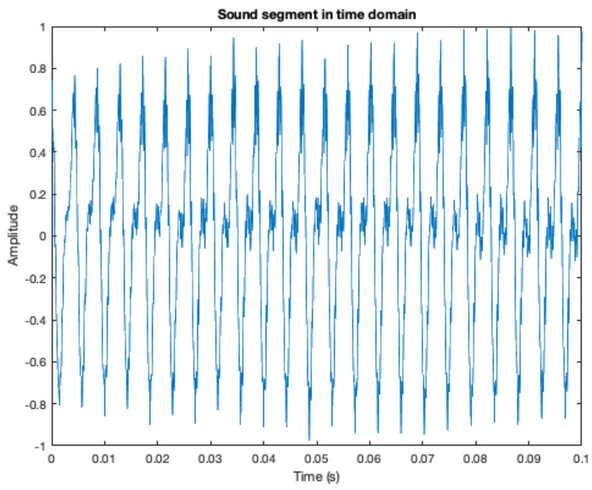

*Figure 4: Female waveform*

After obtaining the segment data, it is time to find the filter coefficients that represents the vocal tract of each source. The `lpc` method [2], which is popular in filter design and speech coding, uses the autocorrelation method of autoregressive (AR) modelling to calculate the filter coefficients by reducing the prediction error to minimal in the least squares sense, but the filter might not precisely model the original speech despite using the right order as autocorrelation windows the data implicitly.

- LPC Coefficients for male sound (1x26): [1.0000 -1.2948 0.4220 0.0422 0.4117 -0.5744 -0.1506 -0.0485 0.2946 -0.5763 0.5382 -0.2053 -0.2856 0.0928 0.3574

0.0529 -0.1284 -0.1869 0.6965 -0.2594 -0.2415 0.2919 -0.0870 -0.0184 -0.1813 0.0707]

- LPC Coefficients for female sound (1x26): [1.0000 -1.5316 0.8372 -0.0843 -0.2444 -0.0074 -0.0824 -0.1320 0.0506 0.0088 0.0475 0.0383 0.1780 0.0152 0.1023 -0.1220 -0.0133 0.1410 -0.0306 -0.2082 -0.0136 0.1011 -0.0685 0.0101 0.0508 -0.0164]

With the coefficients recovered from real speech, it is now possible to plot speech amplitude spectrum using the `fft` function in correspondence with the LPC filter response using the `freqz` function that takes coefficients and outputs frequency of the filter. The `freqz` function is also based on the `fft` function which applies Fast Fourier Transform (FFT) algorithms to compute frequency response thus allowing representation of other important speech characteristics such as pitches, harmonics and formants. Female sound overall vibrates at higher pitch than male sound from the inference of Figure 5 and Figure 6.

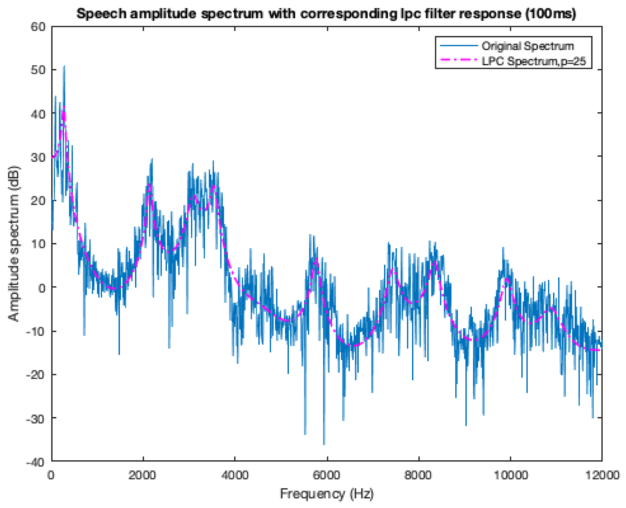

*Figure 5: Male spectrum with LPC response (p=25)*

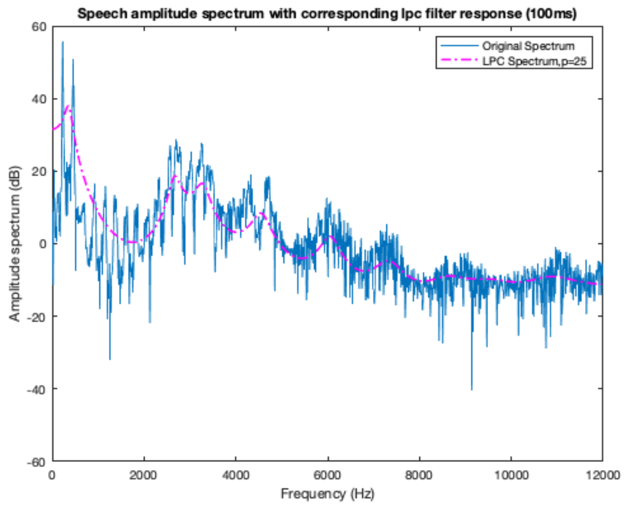

*Figure 6: Female spectrum with LPC response (p=25)*

To find the formant frequencies, we need to find resonance locations by treating the coefficients as a polynomial and calculate their roots by keeping only the ones with one sign for the imaginary part and then calculate the angles that correspond to the roots [3].

- First 3 male formant frequencies: [274.48 Hz, 2137.44 Hz, 3079.82 Hz]
- First 3 female formant frequencies: [363.39 Hz, 2661.51 Hz, 3283.85 Hz]

To find the fundamental frequency estimates over time, we use the `pitch` function that use Normalized Correlation Function (NCF) [4] to estimate the pitch between a search range for overlapped analysis windows. The estimated outputs then are averaged to find the mean fundamental frequency.

- Male mean fundamental frequency: 92.9664 Hz
- Female mean fundamental frequency: 231.2259 Hz

Synthesized vowels can finally be produced by generating a 1 sec impulse train with the obtained mean fundamental frequency and filtering the train with the LPC filter using the `filter` function. The synthesized speech is written to the disk by the `audiowrite` function with custom filename that identifies the speaker's gender, duration and AR order used.

The `lp_speech_synthesizer` function ends with iterative production of multiple filter responses and new synthesized sounds of the same length on different orders (5, 50, 100) that correspond to the original speech amplitude spectrum. It is later reused on the `main` execution workspace to try out the synthesis process and different order evaluations of a sound segment on different gender and segment lengths.

## Result

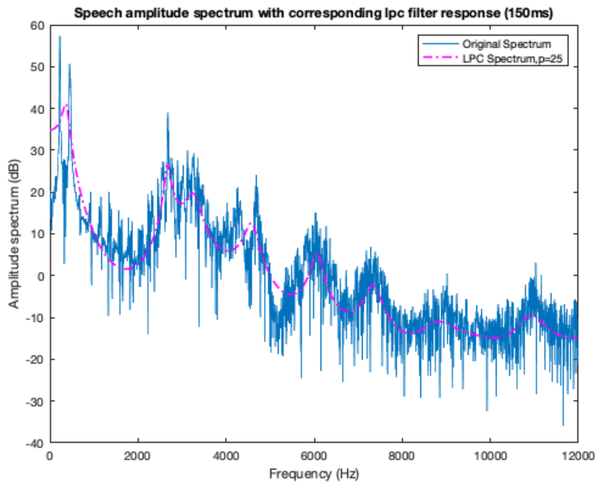

*Figure 8: Female spectrum with LPS response (150ms)*

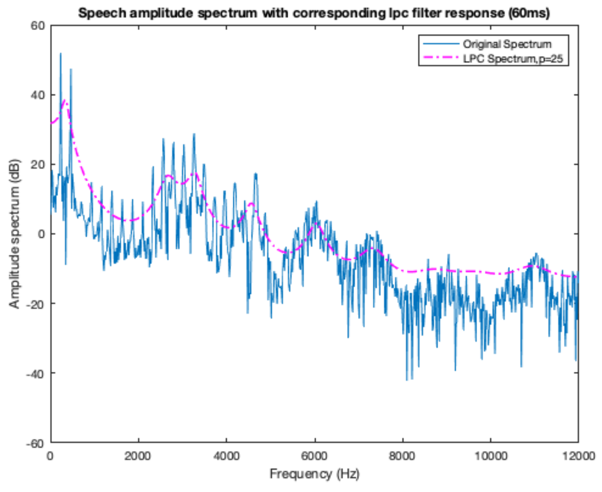

*Figure 7: Female spectrum with LPC response (80ms)*

From Figure 6, Figure 7 and Figure 8, segment increase from 80-150ms give higher spectral detail and more peaks but major formants still remain the roughly the same and the LPC filter response still consistently modelling major peaks of different segments. This makes minimal quasi-stationary length sufficient for modelling voice sounds such as vowel as it does a good job of shaping the spectral envelope and not overreaching into the valleys like the longer segment does.

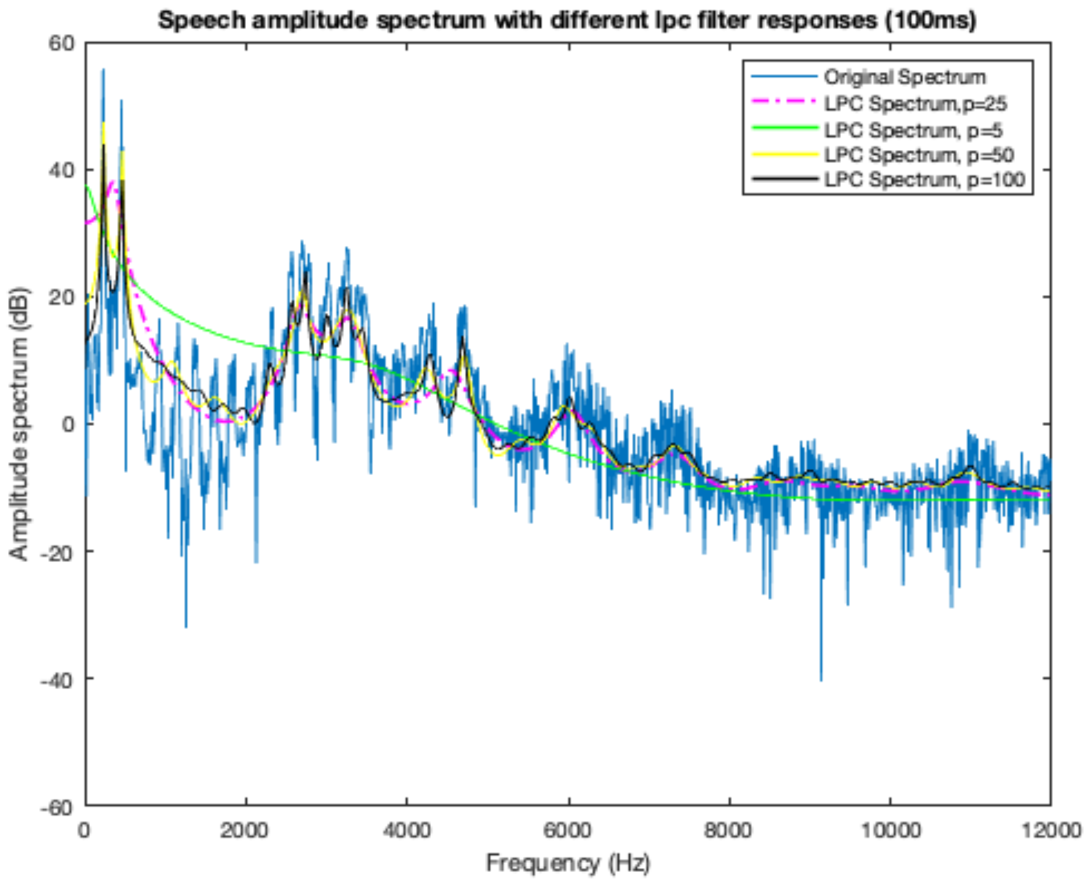

*Figure 9: Female spectrum with different LPC responses*

Judging from both male and female spectrums in Figure 9 and Figure 10, LPC order decay results in less peaks, flatter curve of filter response and poorer fit of the spectral envelope. When LPC order increases, it gives incrementally better spectral envelope until the order of 25 as from the order of 25 up it does not yield any significant peaks except the unnecessary ones that shape up from the valleys. The female sound suffers more than the male sound in the issue of these unnecessary valley peaks look randomly out of shape and do not seem to represent the original speech spectrum possibly as a result of LPC limitation [2] that cannot models the exact speech no matter how high the order.

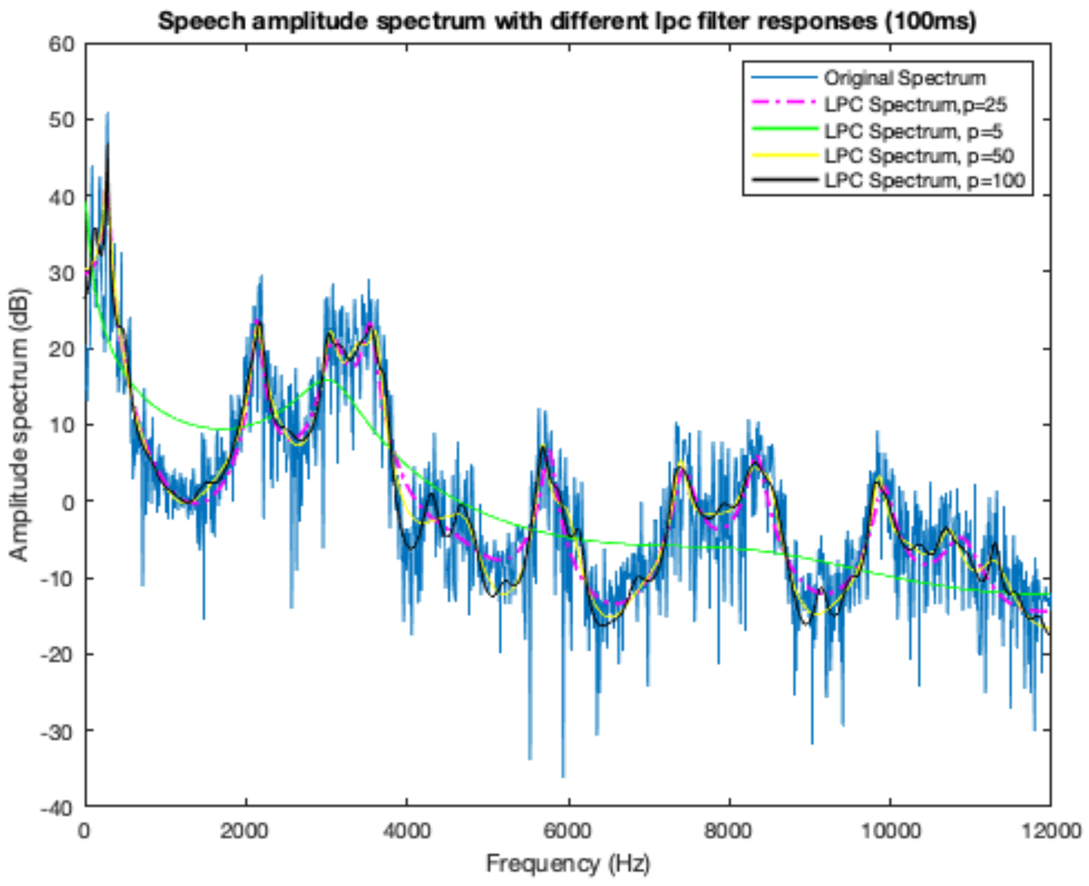

*Figure 10: Male spectrum with different LPC responses*

After listening to synthesized speeches of male and female, both sound buzzy, robotic and peaky in general as a result of enveloped-oriented LPC model. But the male sounds less buzzy and can still be distinguished as vowel at the order 25 unlike the female sound which sounds noisy and distorted at all orders. That does not excuse the male sound as when its order is tuned down the vowel sound is less audible but still it does not suck as that bad like the female vowel. Both sound qualities seem to suffer from lower order because of the lesser formants and also from the higher order probably as a result of aliasing like how spectral detail of Figure 9 and Figure 10 suffer from distorted peaks. Hence, as female sound is consistently the victim of spectral distortion, it is noticed that the LPC model worsens at high pitch frequency and order and that the model is better fit for low to medium pitch voices like male.

## References

[1] W. Wang, “Speech and Audio Processing and Recognition (Part 1),” Centre for Vision Speech and Signal Processing, 2020.

[2] L. B. Jackson, Digital Filters and Signal Processing 2nd Edition, Boston: Kluwer Academic Publishers, 1989, p. 255–257.

[3] R. C. Snell and F. Milinazzo, “Formant location from LPC analysis data,” *IEEE® Transactions on Speech and Audio Processing,* vol. 1, no. 2, pp. 129-134, 1993.

[4] A. S. Bishnu, “Automatic Speaker Recognition Based on Pitch Contours,” *The Journal of the Acoustical Society of America,* vol. 52, no. 6B, p. 1687–1697, 1972.

## Appendix 1 – Main code workspace (`main`)

```
clc
clear
close all

filenames = {'heed_m.wav','heed_f.wav'};
lens = [0.1,0.06,0.15];

% uncmt/cmt and run each fn separately to avoid mess %

% identify_segment(filenames{2}, lens(1))

% synthesize male sound on different lengths %
[lpc_coeffs, f0, formant_fs, formant_res] =
lp_speech_synthesizer(filenames{1}, lens(1))
% [lpc_coeffs, f0, formant_fs, formant_res] =
lp_speech_synthesizer(filenames{1}, lens(2))
% [lpc_coeffs, f0, formant_fs, formant_res] =
lp_speech_synthesizer(filenames{1}, lens(3))

% identify_segment(filenames{2})

% synthesize female on different lengths %
% [lpc_coeffs, f0, formant_fs, formant_res] =
lp_speech_synthesizer(filenames{2}, lens(1))
% [lpc_coeffs, f0, formant_fs, formant_res] =
lp_speech_synthesizer(filenames{2}, lens(2))
% [lpc_coeffs, f0, formant_fs, formant_res] =
lp_speech_synthesizer(filenames{2}, lens(3))
```

## Appendix 2 – Linear Predictive Speech Synthesizer (`ln_speech_synthesizer`)

```
function [lpc_coeffs, f0, formant_fs, formant_res] =
lp_speech_synthesizer(file_name, seg_len)

    % base lpc order and clip start %
    lpc_order = 25;
    t_start = 0.03;

    [sound0, fs]= audioread(file_name);

    % clip vowel segement %
    seg_samples = seg_len*fs;
    I0 = t_start*fs;
    Iend = I0+seg_samples;
    seg = sound0(I0:Iend);
%     sound(seg,fs);

    % plot segment in time domain %
    t = (0:seg_samples)/fs;
    figure(1)
    plot(t,seg);
    xlabel('Time (s)')
    ylabel('Amplitude')
    title('Sound segment in time domain')

    % estimate lpc coefficients with autocorrelation method %
    lpc_coeffs = lpc(seg,lpc_order);

    % plot frequency response of digital filter - magnitude & phase %
    figure(2)
    freqz(1,lpc_coeffs,512,fs);
    title('Frequency response of digital filter (Magnitude & Phase)')

    % plot magnitude frequency response of digital filter in dB %
    [h,f] = freqz(1,lpc_coeffs,512,fs);
    figure(3)
    plot(f, 20*log10(abs(h)));
    xlabel('Frequency (Hz)')
    ylabel('Magnitude (dB)')
    title('Magnitude frequency response of digital filter')

    % plot amplitude spectrum in correspondence with filter response %
    fft_seg = fft(seg);
    fft_seg = fft_seg(1:seg_samples/2+1);
    fft_v = 0:fs/length(seg):fs/2;

    figure(4)
    plot(fft_v, 20*log10(abs(fft_seg)), 'DisplayName', 'Original Spectrum');
    title(['Speech amplitude spectrum with corresponding lpc filter response
(' num2str(round(seg_len*1000)) 'ms)'])
    xlabel('Frequency (Hz)')
    ylabel('Amplitude spectrum (dB)')

    hold on
    plot(f, 20*log10(abs(h)), '-.m', 'LineWidth', 1.5, 'DisplayName', ['LPC
Spectrum,p=' num2str(lpc_order)]);

    legend

    % estimate first 3 formant frequencies %
    r = roots(lpc_coeffs);
    r = r(imag(r)>=0.01);
    formant_fs = sort(atan2(imag(r),real(r))*fs/(2*pi));

    formant_fs_3 = zeros(1,3);
    for i = 1:length(formant_fs)
        formant_fs_3(i) = formant_fs(i);

        if i == 3
            break
        end
    end

    formant_res = sprintf('First 3 formant frequencies: [%1.2f, %2.2f,
%3.2f]', formant_fs_3(1:3));

    % estimate mean fundamental frequency  %
    f0 = mean(pitch(seg,fs));

    % generate and plot periodic impulse train %
    t_imp = 0:1/fs:1;
    imp_train = zeros(size(t_imp));
    imp_train(1:round(fs/f0):end) = 1;

    figure(5)
    plot(t_imp,imp_train);
    title('Periodic impulse train')
    xlabel('Time (s)')
    ylabel('Amplitude')

    % produce and plot synthesized segment sound %
    syn_seg = filter(1,lpc_coeffs,imp_train);
%       sound(syn_seg,fs);
    audiowrite([file_name(1:strlength(file_name)-4) '_syn_p'
num2str(lpc_order) '_' num2str(round(seg_len*1000)) 'ms.wav'],syn_seg,fs);

    figure(6)
    plot(t_imp,syn_seg)
    title('Segment speech synthesis')
    xlabel('Time (s)')
    ylabel('Amplitude')

    % evaluate on different lpc orders %
    orders = [5,50,100];
    colors =["g", "y", "-k"];

    figure(7)
    plot(fft_v, 20*log10(abs(fft_seg)), 'DisplayName', 'Original Spectrum');
    title(['Speech amplitude spectrum with different lpc filter responses ('
num2str(round(seg_len*1000)) 'ms)'])
```

```
    xlabel('Frequency (Hz)')
    ylabel('Amplitude spectrum (dB)')

    hold on
    plot(f, 20*log10(abs(h)), '-.m', 'LineWidth', 1.5, 'DisplayName', ['LPC
Spectrum,p=' num2str(lpc_order)]);

    for i = 1:length(orders)
        e_coeffs = lpc(seg, orders(i));
        [e_h,e_f] = freqz(1,e_coeffs,512,fs);

        % produce synthesized segment sound %
        e_syn_seg = filter(1,e_coeffs,imp_train);
        audiowrite([file_name(1:strlength(file_name)-4) '_syn_p'
num2str(orders(i)) '_' num2str(round(1000*seg_len)) 'ms.wav'],e_syn_seg,fs);

        figure(7)
        hold on
        plot(e_f,20*log10(abs(e_h)),colors(i),'LineWidth', 1.5,
'DisplayName', ['LPC Spectrum, p=' num2str(orders(i))]);
        hold off
    end

    legend

end
```

## Appendix 3 – Identify Segment (`identify_segment`)

```
function [info] = identify_segment(file_name,seg_len)

    info = audioinfo(file_name)
    [sound0, fs]= audioread(file_name);

    % identify stationary vowel segement with spectogram %
    figure(8)
    window = round(seg_len*1000);
    noverlap = round(window*0.75);
    nfft = 128;
    spectrogram(sound0, window, noverlap, nfft, fs, 'yaxis')
    title('Spectrogram')

end
```