LINEAR PREDICTIVE SPEECH SYNTHESIZER

EEEM030: Speech and Audio Processing & Recognition

UNIVERSITY OF SURREY

Orogun, Tega O (PG/T - Comp Sci & Elec Eng)

University of Surrey | Stag Hill, University Campus, Guildford GU2 7XH

ABSTRACT

The project aims to show one of the digital signal-processing techniques that can first analyse the signal and then model and synthesise the signal. In this project, a source-filter model is created to achieve all the requirements to create a replica of a human speech signal with vowel formants using Linear Predictive Coding (LPC) coefficients for both analysis and synthesis.

Table of Contents

IMPLEMENTATION AND TECHNIQUES USED

As stated in the abstract, the main purpose of this project report is to show the results after implementing a digital processing technique to analyse and afterwards, model and synthesise speech signals. There are two categories in this report: Modelling of the speech signal and Synthesis of the speech signal.

Both stages of this project were implemented in MATLAB. This choice was made due to MATLAB having functions that assisted in calculating some of the analysis and synthesising methods used in the project. The first stage conducted was the Modelling stage.

ANALYSIS AND MODELLING OF THE SIGNAL

In speech, the initial excitation signal is created by the vocal cord. This initial signal is where the fundamental frequency (or the first harmonic frequency) of the speech signal is calculated. This initial signal then passes through the vocal tract, which acts as a filter or resonance for the signal.

For this project, we were provided with multiple sound files containing speech samples containing vowels as voiced phonemes. It is important to note that the experiments done on the speech signals were conducted on the speech samples: *hood_m.wav* and *head_f.wav*. After loading the sound sample, a segment of about 100ms was clipped from the original sound sample for analysis.

To calculate an estimate for the fundamental frequency of the signal segment, the autocorrelation() function in MATLAB (rather than the **pitch()** built-in function, as that was also an option). Autocorrelation is when a signal is compared to itself, after a delay, to show similarities or differences. Using this function, the signal will have different peaks at different time lags. These peaks correspond to the pitch period of the signal. This method was picked because it is good with relatively noisy signals and quasi-periodic signals. After

applying the autocorrelation function, the **findpeaks()** function was utilised to find the valid peaks in the signal (to find valid peaks in both the file samples, a minimum and maximum fundamental was set. 80-150Hz for male speech and 90-250Hz for female speech) (1,2). The mean of the computed peaks was then used as the calculated fundamental frequency.

To estimate the formant frequencies and the filter state of the vocal tract on the excitation signal from the vocal cords we used Linear Predictive Coding(LPC). Formant frequencies are resonant frequencies of the vocal tract. With these frequencies, we can determine the vowel sounds in a speech signal. With MATLAB's built-in **lpc(signal, order)** function, the LPC coefficients were calculated with varying orders to show any differences (3). The signal was plotted in the time domain first, but when comparing the frequency response of the LPC filter to the original signal, the signal would have to be calculated in the frequency domain. To achieve this, the Fourier Transform was applied to the signal using the **fft()** function to compute the FFT of the signal. As for the LPC filter frequency representation, this was calculated using the **freqz()** function. This produced the spectral envelope of the signal which was then co-plotted against the original signal. With the clear peaks in the spectral envelope, we could estimate the vowel formant frequencies of the signal.

SYNTHESIS OF THE SIGNAL

To synthesise the signal we will be using LPC coefficients as a filter convolved with an impulse train at a period which will be calculated from the computed mean fundamental frequency from the analysis phase (4). This will be achieved by using MATLAB's built-in **filter()** function(5). The quality of the resulting signal will depend on the order value given to the **lpc()** function. This number will be varied to acquire the best signal. The resulting best signal will then be written to a separate sound file (one for the male sound sample and one for the female sound sample).

RESULTS AND EXPERIMENTATION

LPC COEFFICIENT ESTIMATION

As stated in the previous section, the LPC Filter coefficients were estimated using a built-in MATLAB function. The order for the lpc() function was varied to observe how the filter response graph would compare to the original signal and to also discover which order was best to help calculate the formant frequencies for the sound samples. Following this testing, it was discovered that generally, the order which generated adequate formant frequencies was around 26. Below in Figures 1 and 2, the frequency response graphs for both hood_m.wav and heed_f.wav are plotted respectively. The maxima of the filter response were plotted using green crosses (to help with visibility).

As expected, the peaks from the graphs show that the formant frequencies calculated from the female speech segment are generally higher than those of the male speech segment (although, the first formant frequency calculated from the male speech segment was higher than that of the first formant frequency on the female speech segment as shown in Table 1).

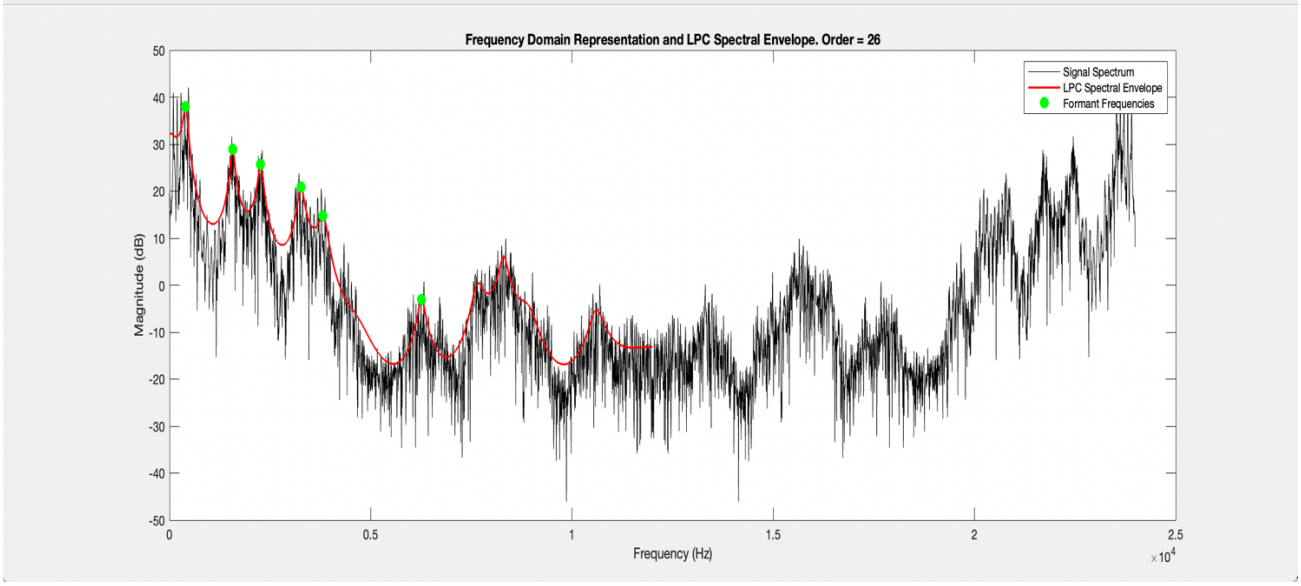


Figure 1: Frequency Response graph for hood_m. Segment length: 100ms. LPC Order: 26

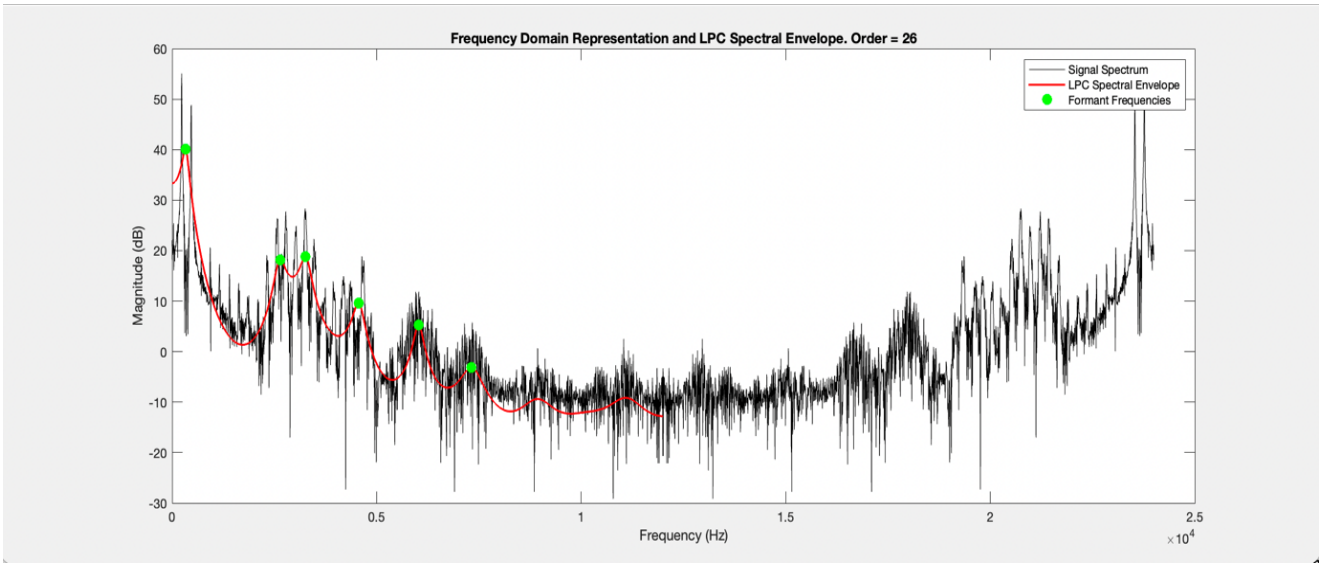


Figure 2: Frequency Response graph for heed_f. Segment length: 100ms. LPC Order: 26

FORMANT FREQUENCY ESTIMATION

To estimate the formant frequencies for both speech sample segments, as previously explained, the peaks of the filter response were used. The **findpeaks()** function was used on the absolute value of the returned values from the **freqz()** function which returns the frequency response. The formants were then chosen in order after specifying a range in which the formants should be (between 90 to 8000 Hz for human speech). Below in Table 1, the first three calculated formant frequencies are shown.

FORMANT FREQUENCIES		
Hz	*hood_m*	*heed_f*
Formant 1	395.51	325.20
Formant 2	1567.38	2645.51
Formant 3	2264.65	3254.88

Table 1: Formant Frequencies Estimated

FUNDAMENTAL FREQUENCY ESTIMATION

To estimate the fundamental frequency for the speech segments, as previously explained, the autocorrelation built-in function was used. The equation using the correlation is:

$$F0 = \frac{Fs}{T}$$

where F0 is the fundamental frequency

Fs is the Sampling Rate

T is the period of the signal

After using **xcorr()** on the segment, the **findpeaks()** function was used to the peaks in the autocorrelation function that would correspond to the expected range of pitch of female or male speech(2,6). The average of these values is then computed to mitigate the effects of errors in the peak detection. Below in Table 2, the computed results are shown, as well as the computed samples that were used for the impulse train generation which we will later discuss.

FUNDAMENTAL FREQUENCIES		
	Hood_m	Heed_f
Mean Fundamental Frequency (F0) (Hz)	112.41	164.16
Pitch Period (samples)	214	146

Table 2: Fundamental Frequency Estimations

PERIOD IMPULSE TRAIN GENERATION AND SPEECH SYNTHESIS

To synthesize the signal, Linear Predictive Coding coefficients and an impulse train were used. The period of the impulse train was first calculated based on the calculated mean fundamental frequency (this value was rounded to the nearest integer to ensure that the number of samples would return a whole number). An impulse train was then generated using the **zeros()** MATLAB function at a length equal to how long we would like the synthesized signal to play for. The critical part of this process was applying the **filter()** function to the impulse train(5). With this function, the LPC filter was applied to the impulse train. Based on the order of coefficients that was chosen, the filter shapes the impulse train to a shape that would resemble the original speech sample segment that was read. It is important to note, that during this project the best LPC order discovered was an order of 26. Increasing the order any more than this made no exceptional improvement to the sound quality of the synthesized signal. At the end of this report, I have included spectrograms of both segment samples and the accompanying synthesised signal.

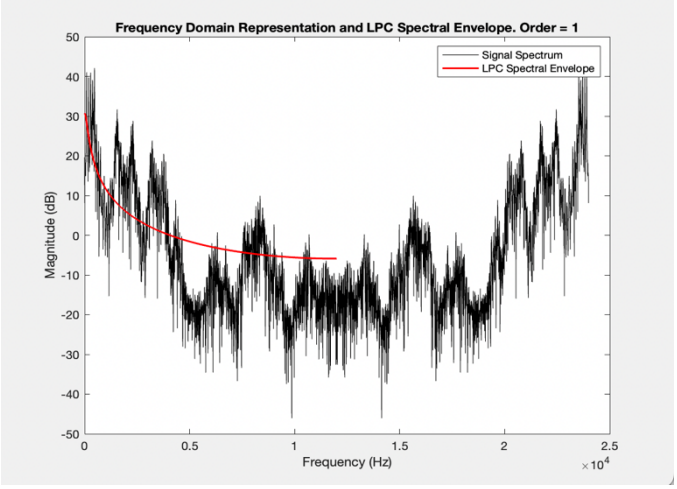
EVALUATION OF DIFFERENT LPC ORDERS AND SEGMENT LENGTH VARIATION

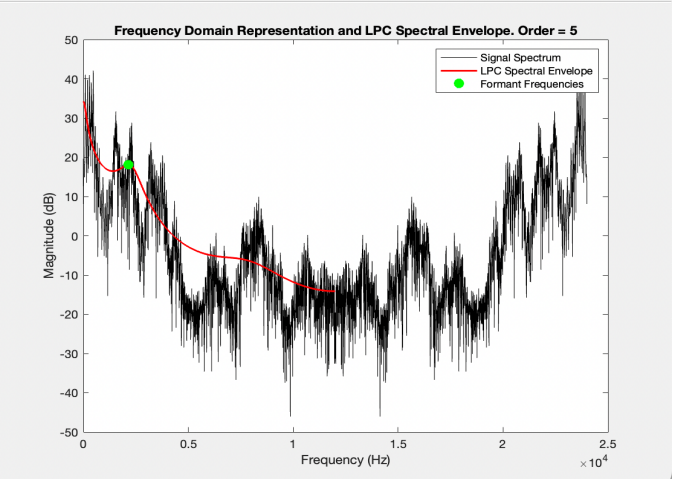
Below in Figures 3 and 4, are the observed differences when the order was changed and when the read-in segment was varied respectively when calculating the coefficients.

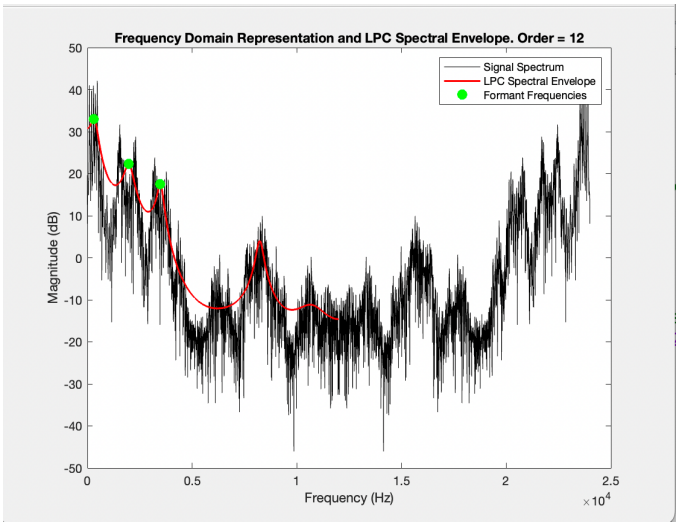
As shown in the images, when the order is increased, the response better fits the original signal spectrum forming more peaks (and matching the original signal's vowel formants). This can be seen below at the higher orders (at orders of 37 and 70). At order 70, the formant peaks almost appear to be grouped because of overfitting the envelope to the original signal spectrum. At the other end, if the order is too low (as shown in the images where the order is 1 and 5), no discernible or inaccurate peaks are calculated for the peaks.

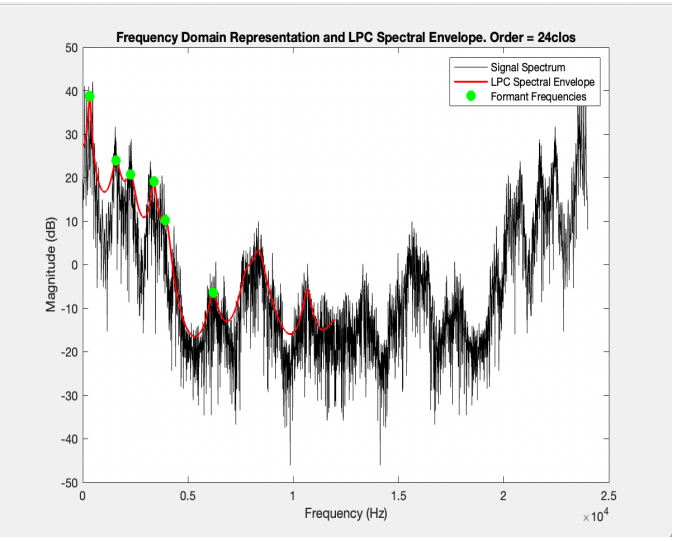
As previously stated the order variation impacts the overall quality of the speech of the synthesised signal to a specific point (about 25-35). At this point, the sound quality does not make any notable improvements.

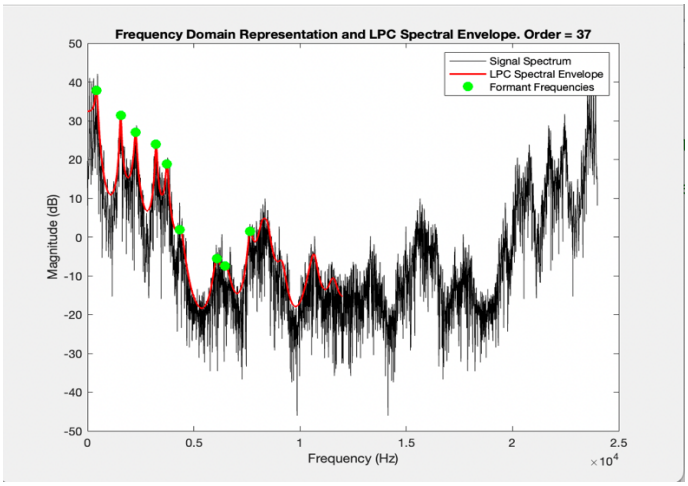
As for the variation in sample segment lengths, the peaks in the response seem to fluctuate but not dramatically. The difference in the first formant appears to be the peak with the greatest change, as at a segment length of 150ms, the first formant peak appeared at 404.30Hz (as shown in Table 3). As for the other formants, they remain relatively similar across all segment lengths.











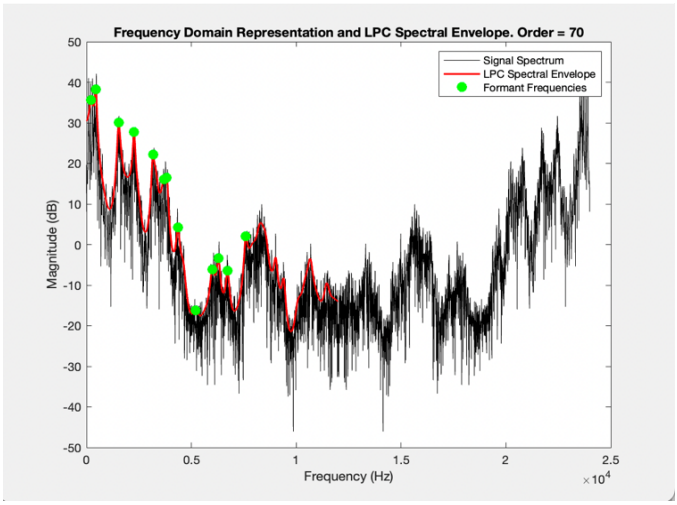
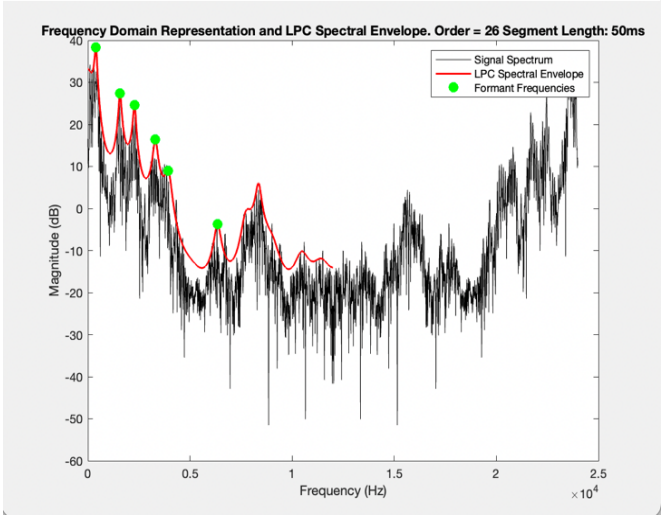
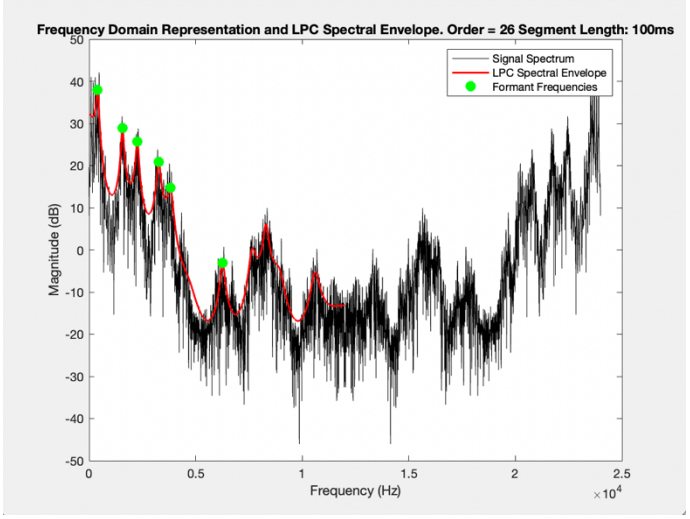
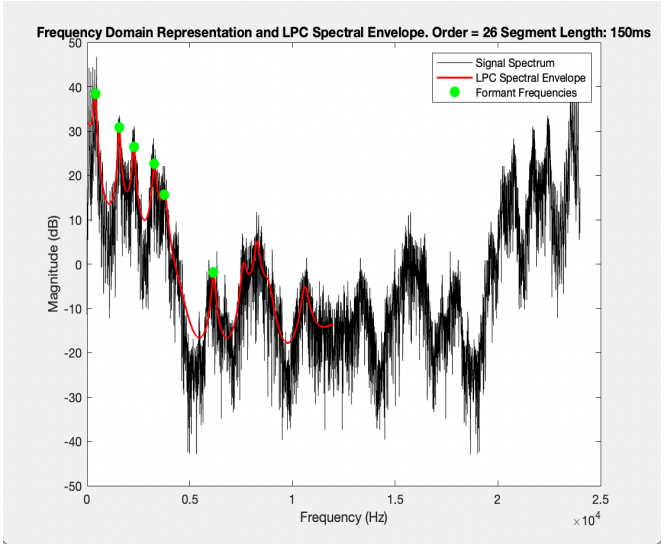


Figure 3: Frequency Response with varying Orders







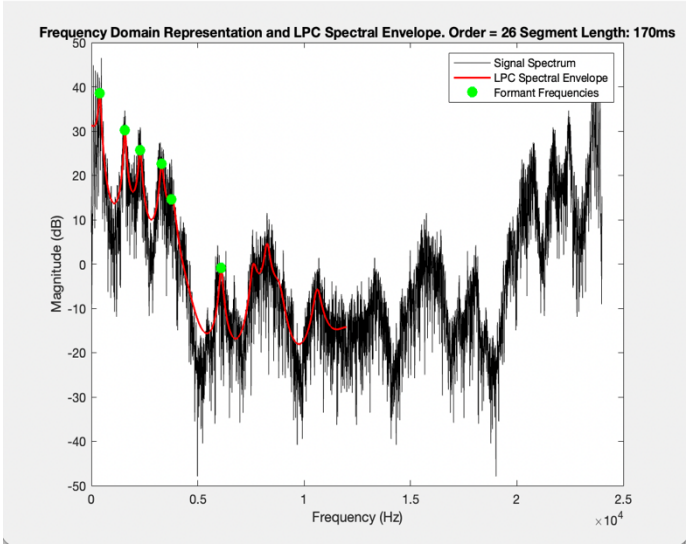


Figure 4: Frequency Response with Varying Segment Lengths

FORMANT FREQUENCIES AT DIFFERENT SEGMENT LENGTHS				
	50ms	100ms	150ms	170ms
Formant 1 (Hz)	392.58	395.51	404.30	395.51
Formant 2 (Hz)	1576.17	1567.38	1573.24	1576.17
Formant 3 (Hz)	2285.16	2264.65	2279.30	2293.95

Table 3: Formant Frequencies at Varied Segment Lengths

ASSESSMENT AND DISCUSSION

As stated previously, the LPC order is the most critical variable in this project. Variations of this variable affected different parts of the project. A low order would result in no discernible formant being formed in the response, which then affected the resulting synthesised signal. The synthesised signal would appear as a *buzzing* sound rather than hearing any vowel sounds. This is as expected because the synthesised signal is mimicking a speech signal. When the vocal cords send out the excitation signal, without the resonance of the vocal cords, the expectation would be a *buzzing* sound. When the formants are not determined, the impulse train is the overwhelming sound that is heard after playing the sound. Once the order is increased, the sounds of the vowel formants begin to be replicated and become more distinguishable.

Once the order was set to around 35, the synthesized signal did not seem to have any more significant improvements in quality. The vowel formants were now clearly audible. It is important to note that this project is only a crude method of synthesizing a speech signal. As it is now, the synthesised speech does contain the vowel sounds but it still appears as quite a robotic sound. Some other approaches would be to switch out the simple impulse train for a Glottal Pulse Model(7). This involves using a more complex pulse model that will more accurately replicate the human glottal wave(7). Other approaches would be to use some post-processing techniques after getting the resulting signal from the LPC coefficients. Some of these techniques include Equalisation, Dynamic Range Compression and Reverberation. (8)

CONCLUSION

This project has shown the creation of a source-filter model to analyse, model and synthesize a speech signal. In this project, we have used the LPC analysis method to both analyse and synthesize the signal showing how varying the LPC order results in different results for both the analysis phase and the synthesis phase. After finding a suitable order, a speech signal mimicking a human speech signal was created.

REFERENCES

1. Fouquet M, Pisanski K, Mathevon N, Reby D. Seven and up: Individual differences in male voice fundamental frequency emerge before puberty and remain stable throughout adulthood. R Soc Open Sci. 2016 Oct 1;3(10).
2. Mathworks. findpeaks() [Internet]. [cited 2023 Nov 8]. Available from: https://uk.mathworks.com/help/signal/ref/findpeaks.html?s_tid=doc_ta
3. Mathworks. lpc() [Internet]. [cited 2023 Nov 8]. Available from: https://uk.mathworks.com/help/signal/ref/lpc.html
4. Linear Predictive Coding is All-Pole Resonance Modeling [Internet]. [cited 2023 Nov 8]. Available from: https://ccrma.stanford.edu/~hskim08/lpc/
5. Mathworks. filter() [Internet]. [cited 2023 Nov 8]. Available from: https://uk.mathworks.com/help/matlab/ref/filter.html?s_tid=doc_ta
6. Mathworks. xcorr() [Internet]. [cited 2023 Nov 8]. Available from: https://uk.mathworks.com/help/matlab/ref/xcorr.html?s_tid=doc_ta
7. Pickett J. M. Perception of Vowels Heard in Noises of Various Spectra | The Journal of the Acoustical Society of America | AIP Publishing [Internet]. 1957 [cited 2023 Nov 8]. Available from: https://pubs.aip.org/asa/jasa/article-abstract/29/5/613/668185/Perception-of-Vowels-Heard-in-Noises-of-Various?redirectedFrom=fulltext
8. Stochino F, Gayarre FL. Reinforced concrete slab optimisation with simulated annealing. Applied Sciences (Switzerland). 2019 Aug 1;9(15).

APPENDIX

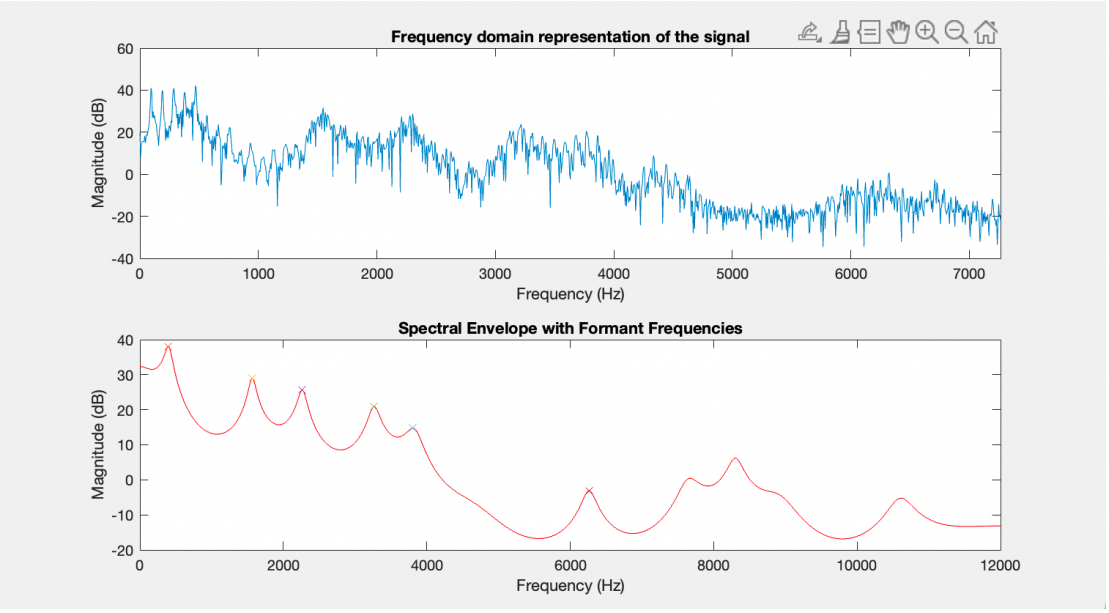


Figure 5: The Spectrum and Spectral Envelope for hood_m

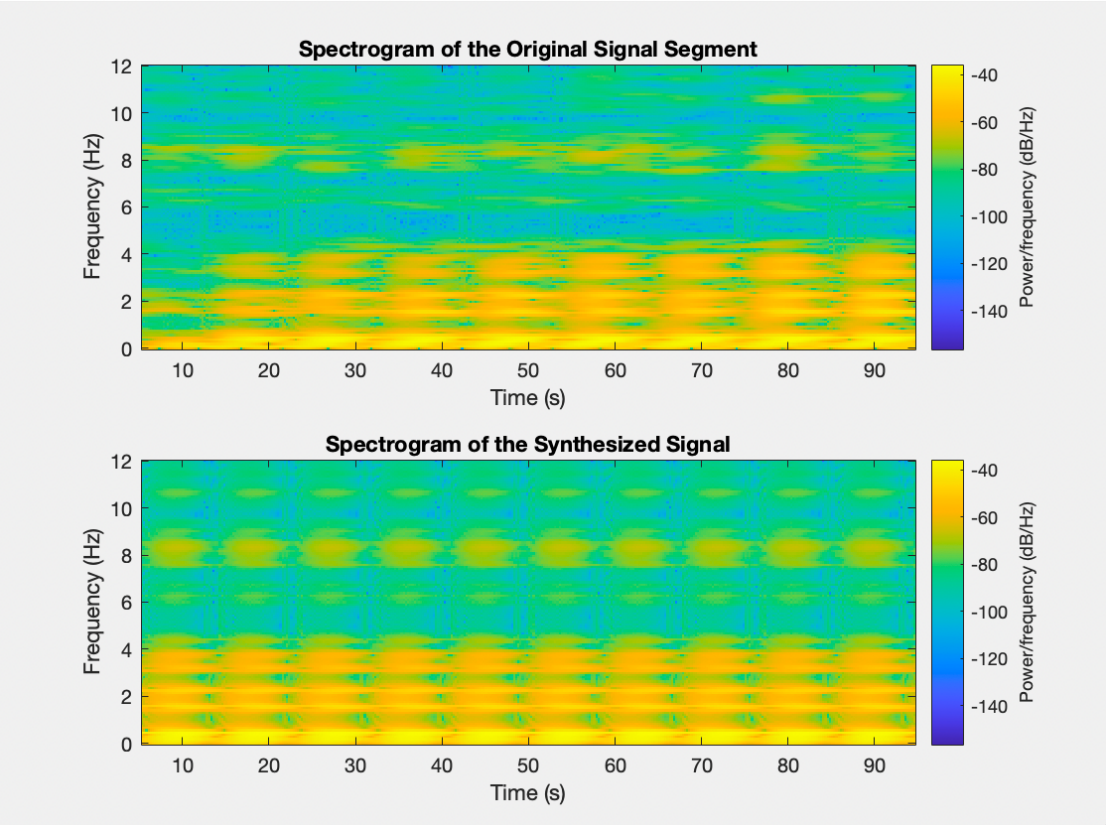


Figure 6: Original Signal Segment Spectrogram compared against the synthesised signal spectrogram for hood_m

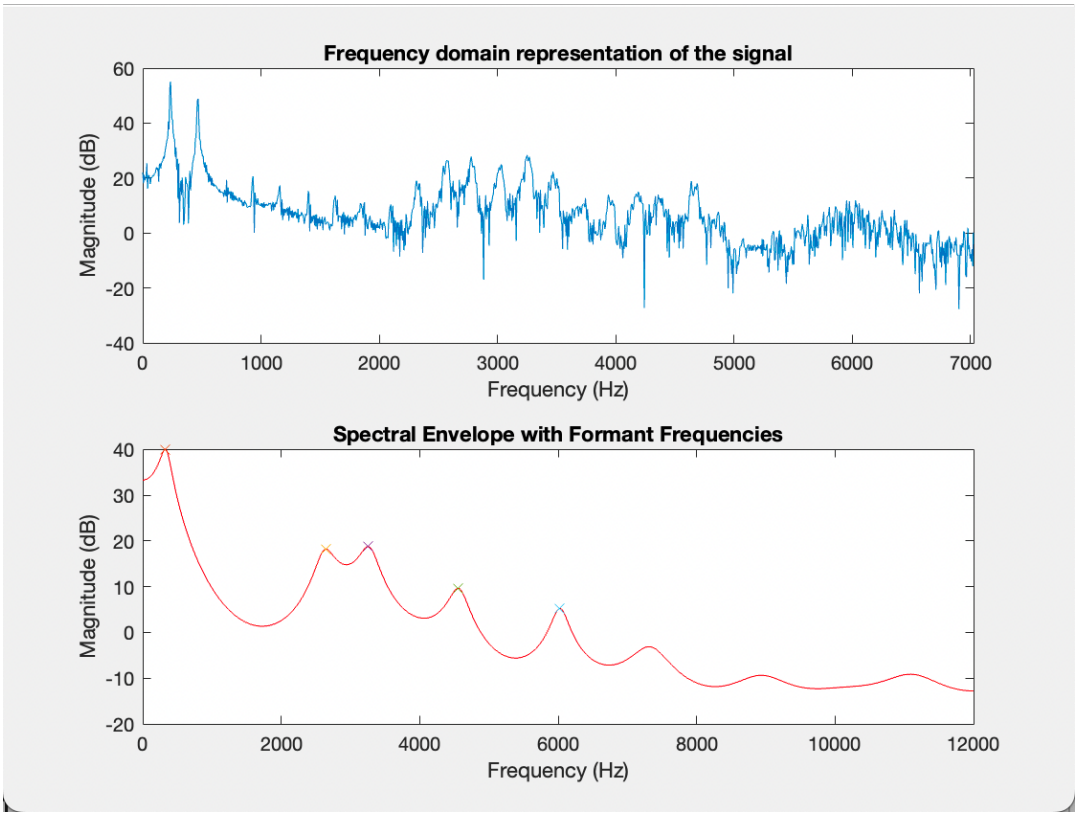


Figure 7: Spectrum and Spectral Envelope for heed_f

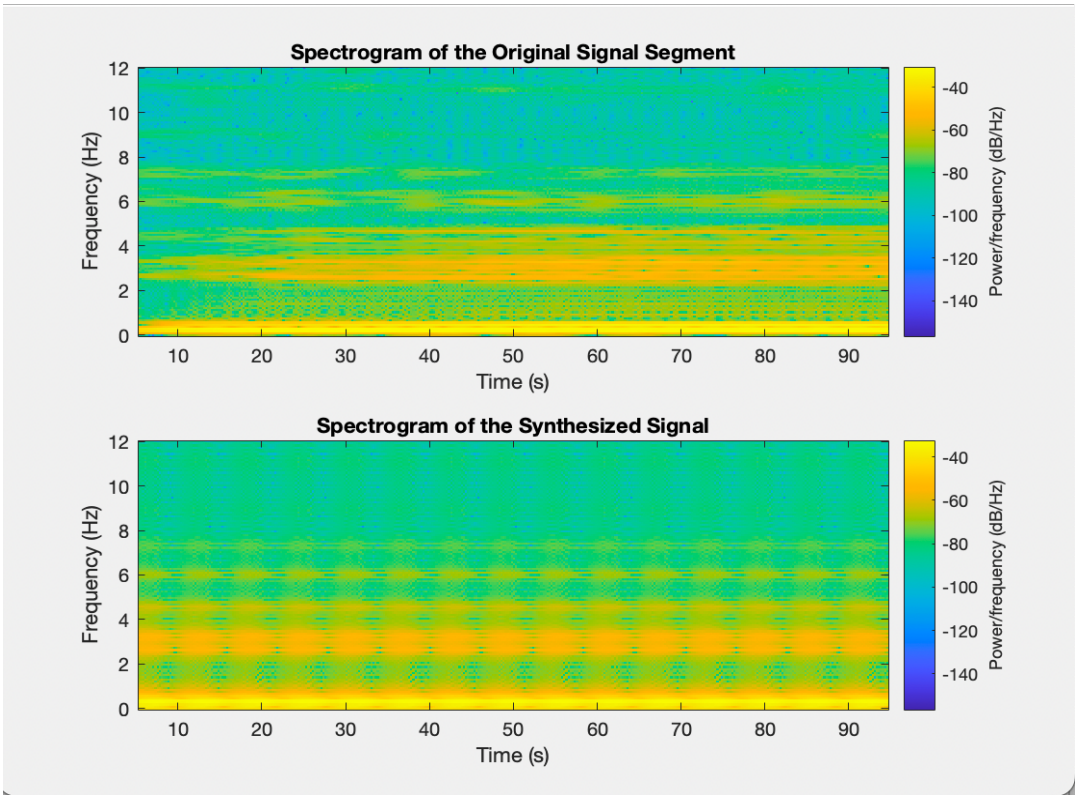


Figure 8: Spectrogram of the original signal and synthesised signal for heed_f

SOURCE CODE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% READ IN SOUND SAMPLES AND DEFINE SEGMENT LENGTHS%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Male speech voice for hood_m
% MALE
% [y, fs] = audioread('hood_m.wav');

% Female speech voice for heed_f
% FEMALE
[y, fs] = audioread('heed_f.wav');

% Define the segment length (in seconds)
segment_length = 0.1; % 100ms

% Calculate the number of samples for the segment
segment_samples = round(segment_length * fs);

% Choose the starting point of the segment (you can change this as needed)
start_sample = 1;

%I use this variable to manipulate the length of the synthesized signal I
%play at the end
desired_duration = 1; % in seconds


desired_samples = round(desired_duration * fs);

% Extract the 100ms segment from the speech signal
segment = y(start_sample:start_sample+segment_samples-1);

% Instead of using the pitch() function built-in I opt for the correlation
% function which I discussed in the report
autocorr_segment = xcorr(segment);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%%%%%%%%%% SPECIFICATIONS FOR BOTH MALE AND FEMALE FREQUENCY
ESTIMATIONS%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%

% Define the expected F0 range (80-150 Hz) for male
% MALE
% min_male_f0 = 80;
% max_male_f0 = 150;

% FEMALE
%Define the expected F0 range (165-255 Hz) for female
min_female_f0 = 165;
max_female_f0 = 255;

% Calculate the lag values corresponding to the F0 range male
% MALE
% min_male_lag = round(fs / max_male_f0);
% max_male_lag = round(fs / min_male_f0);
```

```
% Calculate the lag values corresponding to the F0 range female
% FEMALE
min_female_lag = round(fs / max_female_f0);
max_female_lag = round(fs / min_female_lag);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% FUNDAMENTAL FREQUENCY ESTIMATION%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Find peaks in the autocorrelation within the expected lag range
[peaks, locs] = findpeaks(autocorr_segment);

% Filter peaks within the expected lag range for male
% MALE
% expected_peaks_male = locs(locs >= min_male_lag & locs <= max_male_lag);

% FEMALE
% Filter peaks within the expected lag range for female
expected_peaks_female = locs(locs >= min_female_lag & locs <=
max_female_lag);

% Calculate fundamental frequencies (F0) in Hz for male
% MALE
% fundamental_frequencies_male = fs ./ expected_peaks_male;

% FEMALE
% Calculate fundamental frequencies (F0) in Hz for female
fundamental_frequencies_female = fs ./ expected_peaks_female;

% Calculate the mean F0 for male
% MALE
% mean_male_f0 = mean(fundamental_frequencies_male);

% FEMALE
% Calculate the mean F0 for female
mean_female_f0 = mean(fundamental_frequencies_female);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% LINEAR PREDICTIVE CODING IMPLEMENTATION (THIS INCLUDES FINDING
FORMANTS)%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

% Using LPC to estimate formant frequencies in both speech
order = 70; % Change this to better fit the signal
%The lpc coefficiencts are stored in the first variable in the vector. For
%now I am ignoring the second variable which is the prediction error
[lpc_coeffs, ~] = lpc(segment, order);
[response, norm_ang_freq] = freqz(1, lpc_coeffs,
2^nextpow2(length(segment)), fs);%with the freqz() function, I can plot
the frequency response of the digital signal

% Find peaks in the LPC spectrum that correspond to formants
[pks, freq_locs] = findpeaks(abs(response));

% Select the first three peaks as the first three formants
```

```
% and ensure they are within the expected range for human speech formants
formant_freqs = norm_ang_freq(freq_locs);
valid_formants_idx = formant_freqs > 90 & formant_freqs < 8000; % This
should be the estimated formant frequency for human speech
formant_frequencies = formant_freqs(valid_formants_idx);
formant_frequencies = sort(formant_frequencies(1:0), 'ascend'); % I sort
the first 3 formants but plot five formants

fprintf('Formant Frequencies (1st-3rd):\n');
for i = 1:min(3, length(formant_frequencies))
    fprintf('Formant %d: %.2f Hz\n', i, formant_frequencies(i));
end
% Calculate the mean of the first three formant frequencies
mean_formants = mean(formant_frequencies(1:0));


%Print out both the mean of the fundamental frequencies and the mean
%formants (which might be unnecessary?)
% MALE
% fprintf('Mean F0: %.2f Hz\n', mean_male_f0);
% fprintf('Mean Formants (1st-3rd): %.2f Hz\n', mean_formants);

% FEMALE
fprintf('Mean F0: %.2f Hz\n', mean_female_f0);
fprintf('Mean Formants (1st-3rd): %.2f Hz\n', mean_formants);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% SIGNAL SYNTHESIS PROCESS%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This is used to calculate the period of the impulse train used for the
% synthesised signal play at the end

% MALE
% impulse_train_period = round(fs / mean_male_f0);
% impulse_train = zeros(size(segment));
% impulse_train(1:impulse_train_period:end) = 1;

% FEMALE
% impulse_train_period = round(fs / mean_female_f0);
% impulse_train = zeros(size(segment));
% impulse_train(1:impulse_train_period:end) = 1;

% FOR MALE
% impulse_train_period = round(fs / mean_male_f0);
%
% % FOR FEMALE
impulse_train_period = round(fs / mean_female_f0);
%
%
%
impulse_train_samples = min(desired_samples, impulse_train_period);
impulse_train = zeros(1, desired_samples);
impulse_train(1:impulse_train_samples:end) = 1;

synthesized_signal = filter(1, lpc_coeffs, impulse_train);

% Normalize the synthesized signal
```

```
synthesized_signal = synthesized_signal / max(abs(synthesized_signal));

% Play the signal
sound(synthesized_signal,fs);

% MALE WRITE OUT
% path_male = fullfile('Speech_Main_Assignment',
'synthesized_speech_male.wav');
% audiowrite(path_male, synthesized_signal, fs);

% FEMALE WRITE OUT
% path_female = fullfile('Speech_Main_Assignment',
'synthesized_speech_female.wav');
% audiowrite(path_female, synthesized_signal, fs);

% Plot the signal's frequency domain representation
Y = fft(segment, length(response)); % Compute the FFT of the signal
f = (0:length(Y)-1) * fs / length(Y); % Frequency vector

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% PLOTTING ALL THE GRAPHS %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This is where I start with all the graph plots for the spectrum and the
%spectral envelope.
figure;
subplot(2,1,1);
plot(f, 20*log10(abs(Y)));
title('Frequency domain representation of the signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
%this zooms in the plot to find the formants clearly
xlim([0 max(formant_frequencies) + 1000]);

% Plot the spectral envelope on the same graph
subplot(2,1,2);
plot(norm_ang_freq, 20*log10(abs(response)), 'r');
title('Spectral Envelope of the Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
hold on;

for i = 1:length(formant_frequencies)
    freq_index = find(norm_ang_freq >= formant_frequencies(i), 1);
    plot(formant_frequencies(i), 20*log10(abs(response(freq_index))),
'x');
end
title('Spectral Envelope with Formant Frequencies');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');

% Mark the formant frequencies on the spectral envelope. I tried using 'x'
% as a marker but this wasn't really visible
% plot(formant_frequencies,
20*log10(abs(h(freq_locs(valid_formants_idx)))), 'x');

%GRAPHS
figure;
plot(f, 20*log10(abs(Y(1:length(f)))), 'k');
```

```
hold on;
plot(norm_ang_freq, 20*log10(abs(response)), 'r', 'LineWidth', 1.5);
title('Frequency Domain Representation and LPC Spectral Envelope. Order =
26 Segment Length: 170ms');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');

% Mark the formant frequencies on the plot. Use pretty big green circles
% because it wasn't very clear with x on the same plot as the signal
% spectrum
for i = 1:length(formant_frequencies)
    freq_index = find(norm_ang_freq >= formant_frequencies(i), 1);
    plot(formant_frequencies(i), 20*log10(abs(response(freq_index))),
'go', 'MarkerSize', 8, 'MarkerFaceColor', 'g');
end

% Spectograms
legend('Signal Spectrum', 'LPC Spectral Envelope', 'Formant Frequencies');
hold off;

figure;
subplot(2,1,1);
spectrogram(segment, 256, 250, 256, fs, 'yaxis');
title('Spectrogram of the Original Signal Segment');
xlabel('Time (s)');
ylabel('Frequency (Hz)');

% Plot the spectrogram of the synthesized signal
subplot(2,1,2);
spectrogram(synthesized_signal, 256, 250, 256, fs, 'yaxis');
title('Spectrogram of the Synthesized Signal');
xlabel('Time (s)');
ylabel('Frequency (Hz)');

% sound(y,fs);
```