Linear Predictive Speech Synthesizer

Andy Pack

EEEM030

November 2020

Department of Electrical and Electronic Engineering
Faculty of Engineering and Physical Sciences
University of Surrey

Abstract

A system implementing the source-filter model of speech is presented and evaluated using vowel segments as subjects. Linear predictive coding is used to estimate the formant frequencies of the samples while the cepstrum is used to identify the fundamental frequency. Comparisons of the LPC filter spectrum with the original audio spectrum are provided. A periodic impulse train of the same pitch period is used to synthesise vowel samples, a subjective analysis of the segment quality is given. Evaluations of various parameter variations are also presented.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Speech analysis and processing is an ever-expanding space with applications from data compression to speech recognition. The latter is a particularly relevant and popular area, presenting an important domain for AI and machine learning applications.

Prior to these, however, the ability to analyse, transform and identify key parameters for a speech signal are important tools that will be explored herein.

1.1 Brief

The aim of this report is to demonstrate how digital signal processing techniques can be used to analyse, model and synthesise speech. The task will be considered as two areas of concern, that of modelling and synthesis.

The modelling stage will utilise Linear Predictive Coding[1] and the source-filter model of speech[2] to construct an all-pole filter that acts similarly to the vocal tract's effect on sound produced by the vocal chords. Comparisons of the frequency response for both the estimated filter and the original sound will be presented, the effect of different filter orders will also be demonstrated. Relevant parameters of the original vowel speech segment will be presented including the fundamental frequency[3] and formant frequencies.

The synthesis stage will complete the source-filter model of speech by creating a suitable periodic sound source to be modulated by the previous filter. With a complete source-filter model, artificial vowel sounds will be synthesised and analysed. Subjective assessments will be made as to the differences between the original sound and the final product of the model when system parameters are varied.

2 Implementation

The implementation of this system was completed using MATLAB with aid from functions in the digital signal processing toolbox among others. Following loading a vowel sample, a segment of given length (100ms was typical) was clipped for processing. The investigations were conducted on two samples, `hood_m.wav` and `head_f.wav`, any results from other samples are identified as such.

2.1 Modelling

In order to estimate the filter state of the vocal tract, the linear predictive coding coefficients of varying orders were calculated using the `lpc(signal, order)` function. In order to compare the frequency response of the LPC filter with the original signal, the Fourier transform of the signal was calculated. The frequency domain representation of the LPC filter was found using the `freqz(b, a, n, f)` function and co-plotted with the original signal. This frequency plot of the LPC filter constitutes the spectral envelope of the signal and the vowel formant frequencies can be found at the maxima of the spectrum. The smooth profile of the LPC spectrum allowed the formant frequencies to be estimated by identifying the local maxima[4], [5] of the function.

	1	2	3	4	5	6	7	8	9	10	11
head_f	1	-1.8275	0.6130	0.7424	-0.2264	-1.0744	0.8921	0.4595	-0.8184	-0.3913	1.2207
hood_m	1	-1.9166	0.9014	0.1898	0.5570	-0.9309	-0.4874	1.0068	0.0966	-0.4469	0.1029

	12	13	14	15	16	17	18	19	20	21
head_f	-0.3812	-0.5842	0.2820	0.7351	-0.8951	0.1172	0.4359	-0.1220	-0.3546	0.1977
hood_m	-0.6152	0.7490	0.1002	-0.3020	-0.1184	-0.0494	0.6293	-0.3474	-0.2172	0.2164

Table 1: Order 20 LPC coefficients for both investigated samples, source segments taken from the first 100ms of each vowel sample

In order to find the fundamental frequency of the signal, the cepstrum[6] was used. Regular periodic frequencies in the time domain present as peaks in the quefrency domain, these can also be identified with an auto-corelation function. The use of a low-pass filter was investigated in order to smooth the cepstrum before programmatically finding pitch period candidates by applying x and y thresholds. Firstly, local maxima of the cepstrum function were found using the `islocalmax(x)` function[5]. A minimum quefrency threshold of 20 was applied to ignore the transient-like oscillations at small x values. Lowering the quefrency corresponds to an increase in frequency, thus it is reasonable to discard these values when 20 samples represents 1200Hz when sampled at 24kHz, a frequency higher than that of the fundamental frequency being investigated. Additionally a minimum cepstrum threshold of 0.075 was used, from here the quefrency candidate with the highest value was used as the pitch period.

2.2 Synthesis

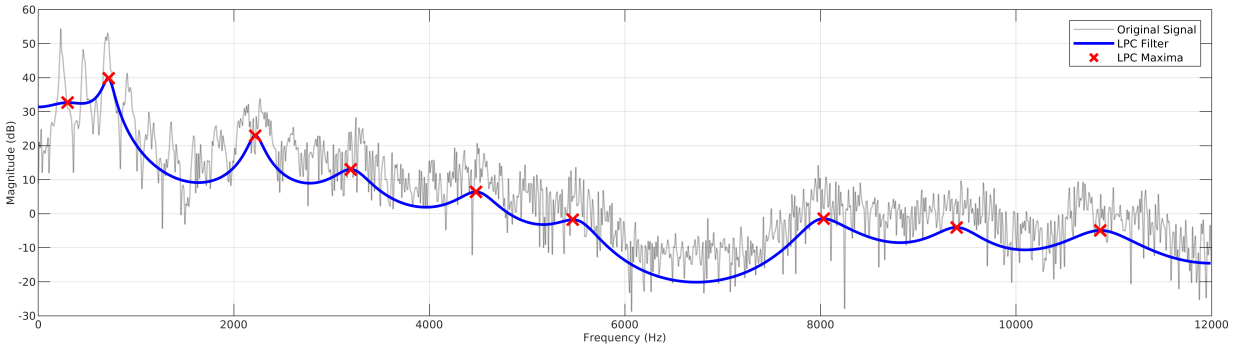
In order to synthesise speech, a periodic impulse train at the identified fundamental frequency of the original vowel was generated. As the fundamental frequency of speech is far lower than a typical audio signal would be sampled at, a carrier signal of the same sampling frequency as the original sound was modulated by the lower frequency impulse train, see listing 7. In order to produce the final synthesised speech, the generated impulse train must be convolved (in the time domain) with the transfer function of the LPC filter representing the vocal tract[2]. In MATLAB this can be completed with the `filter(b, a, x)` which takes the provided coefficients (a, b) and applies the transfer function these describe. This final signal was written to disk and played for comparison to the original.

3 Results

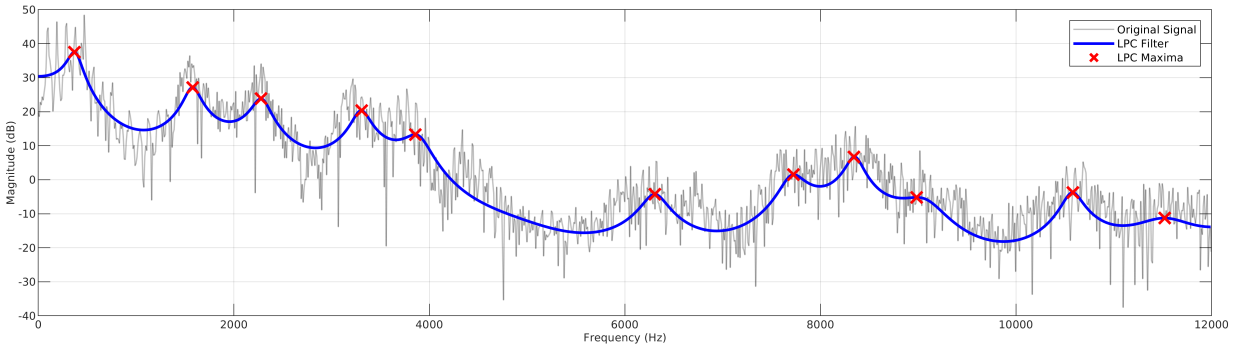
3.1 LPC Filter

LPC filter coefficients of varying orders were calculated, the values for each vowel sample at order 20 can be seen in table 1. The frequency response for similar filters of order 25 can be seen in figure 1, as described in section 2.1 the local maxima of the filter response were also plotted as red crosses.

As the spectra are plotted with the same frequency axes bounds, the peaks of the filter response corresponding to estimations of the formant frequencies can be compared between the male and females voice. In general the male's formant frequencies are lower than for the female's sample, this can be seen specifically with the



(a) `head_f`, order 25



(b) `hood_m`, order 25

Figure 1: LPC filter and vowel segment spectra for both investigated samples

first few peaks. It's worth noting that the first local maxima identified in the `head_f` sample does not appear to have identified a peak that would be considered a formant.

3.1.1 Order Variation

The effect of increasing the order of the LPC filter can be seen in figure 2, the order of the `hood_m` filter is repeatably incremented by 5. In general, as the order of the filter is increased, the spectral response of the LPC filter closer fits the spectrum of the original vowel segment. At lower orders, the filter's response can smooth over multiple peaks and valleys in the original signal as can be seen at order 6, whereas by order 36 the LPC spectrum follows all of the major motions of the speech signal.

3.1.2 Source Segment Length Variation

Figure 3 presents the speech sample and LPC filter spectral response for different source sample lengths. As the source sample length increases the spectral profile becomes less smooth with higher peaks and deeper troughs throughout. Additionally the mid to higher frequencies are affected more, the first few formants are less affected.

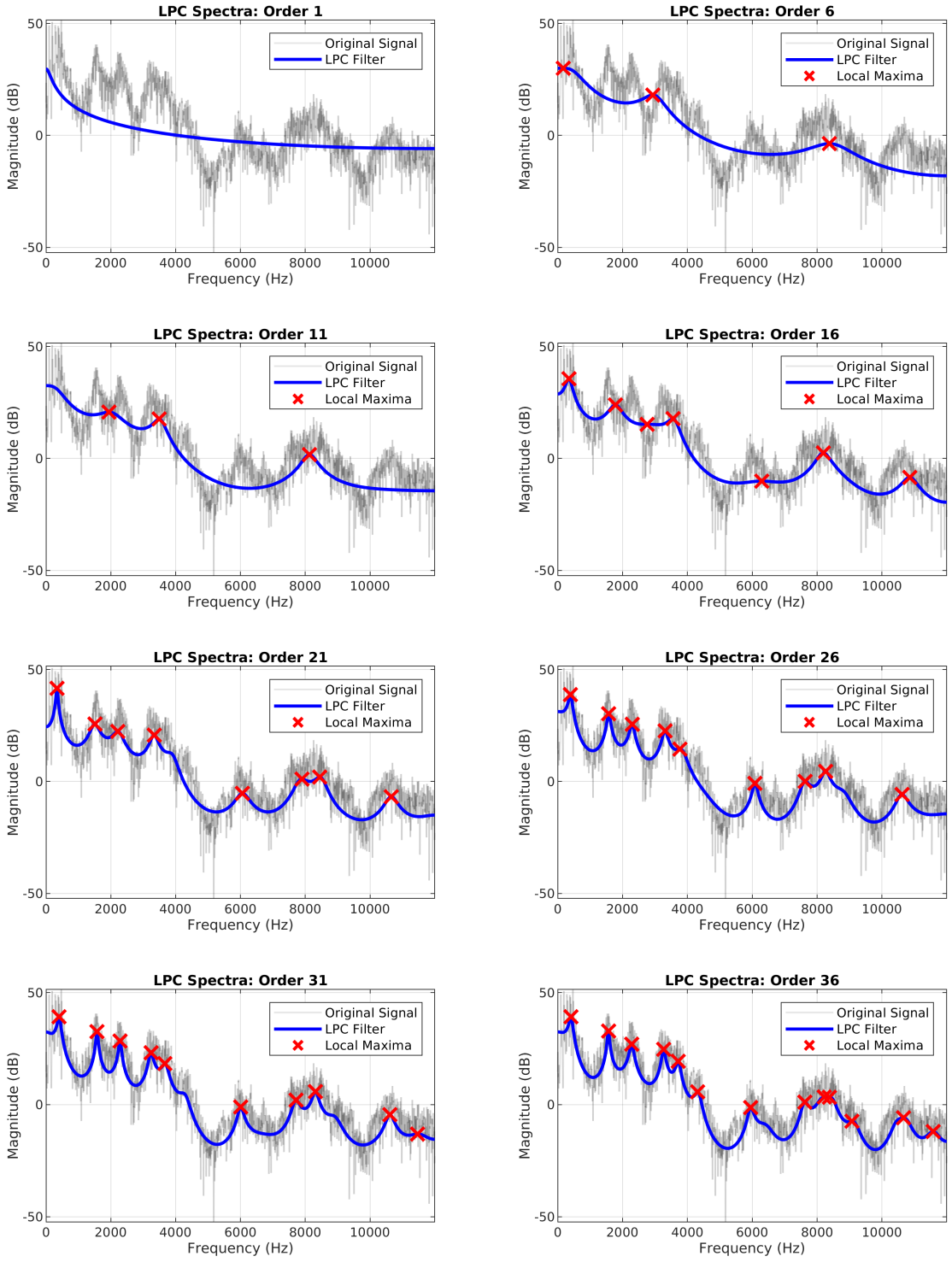


Figure 2: Effect of increasing LPC filter order on the `hood_m` sample

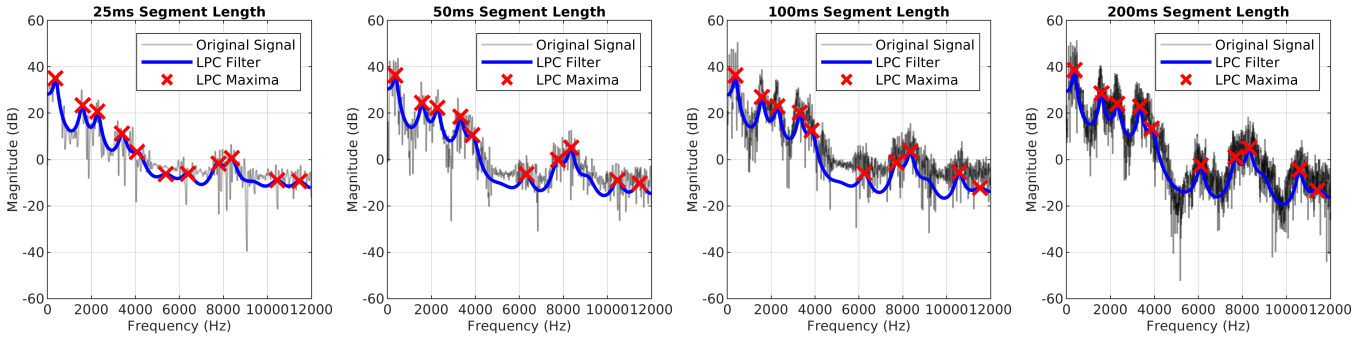


Figure 3: Increasing source segment lengths for the `hood_m` sample

	`head_f`	`hood_m`
f_1	719.4	369.7
f_2	2,218.2	1,578.7
f_3	3,197.3	2,278.1
$f_2 - f_1$	1,498.8	1,209.0

Table 2: First formant frequencies at order 25, Hz

3.2 Spectral Analysis

3.2.1 Formant Frequencies

As described previously, the smooth profile of the LPC filter spectra allows the local maxima to be used as reasonable estimations of the peaks. The first three formants for the order 25 filters seen in figure 1 can be seen in table 2, as described above the first local maxima for the female `head_f` sample was not included as f_1 as it did not refer to a maximum that would indicate a formant.

Table 3 presents average formant frequencies for the investigated vowel sounds as displayed in [7]. The percentage difference between these averages and the calculated estimations are also presented. The female sample was closer to the averages than the male sample.

3.2.2 Cepstrum Smoothing

The effect of smoothing the cepstrum with a low-pass filter is presented in figure 4, [1 -0.7] were used as coefficients. When employing smoothing, the peak corresponding to the pitch period has been amplified compared to the unsmoothed curve where the pitch period does not reach far beyond the noise of the rest of the function. As a result of this, smoothing was employed in the following when identifying the fundamental frequency.

		Average Frequency[7]			Measured % Difference		
Sample	Vowel	f_1	f_2	$f_2 - f_1$	f_1	f_2	$f_2 - f_1$
`head_f`	/ɛ/	731	2,058	1,327	1.6	7.8	12.9
`hood_m`	/ʊ/	469	1,122	653	21.2	40.7	85.1

Table 3: Relevant IPA vowels and their average formant frequencies[7]

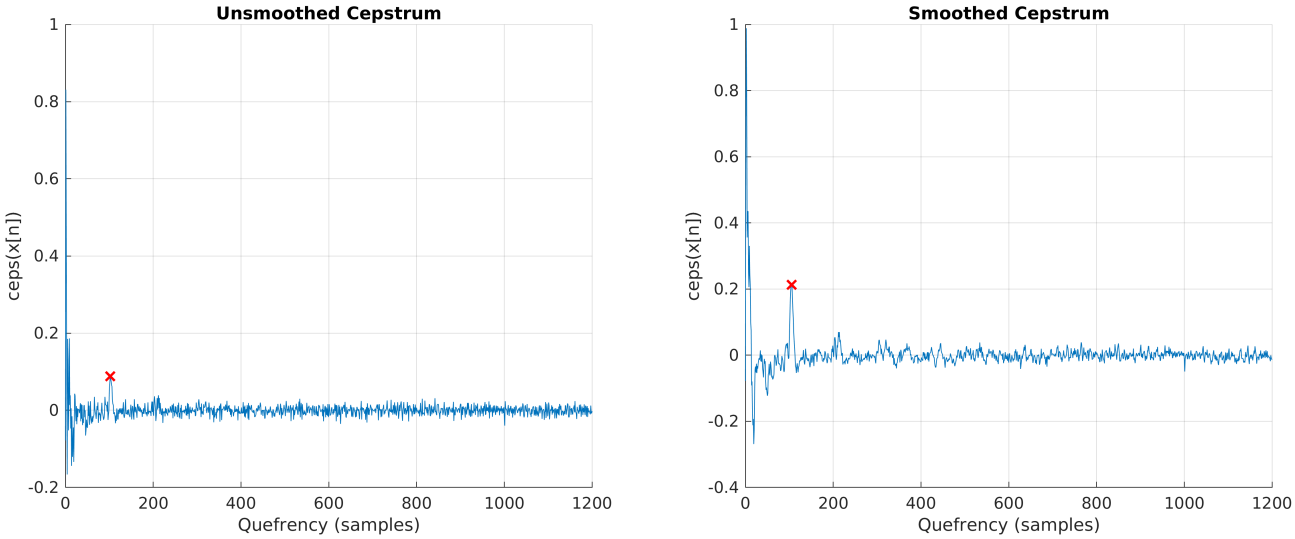


Figure 4: Real cepstrum for `head_f` with and without low-pass filtering, thresholded local maxima crossed, smoothing coefficients: [1 -0.7]

	`head_f`	`hood_m`
Pitch Period, samples	105	255
Fundamental Frequency, Hz	228.57	94.12

Table 4: Pitch period and fundamental frequency as calculated from the real cepstrum

3.2.3 Fundamental Frequency

The fundamental frequency was calculated by identifying the pitch period in the real cepstrum. The cepstrums for either sample were thresholded and the candidates can be seen in figure 5. The identified quefrency pitch period, q_p, and the corresponding fundamental frequency, f_f, can be seen in table 4. f_f was calculated using the following where f_s is the sample frequency,

$$f_f = \frac{1}{q_p / f_s}$$

3.3 Synthesis

Following the convolution of the impulse train and the LPC filter, the spectrograms for the original and synthesised sound can be seen in figure 6. The circled areas highlight similar portions, the formant frequencies can be seen as bright horizontal lines in both. Despite being quasi-stationary, some variation in time can be seen throughout the original signal. The stationary synthesised signal, however, has a flat profile in time.

At lower filter orders (< 10), the synthesised speech has a *buzzy* quality resembling a sawtooth wave of the same pitch as the original voice sample. At these orders, the synthesised sound can not accurately be discerned as speech. As the filter order increases, the tone of the sound becomes less harsh and by around order 20 the sample could be identified as being of a voice. By order 40, much of the harsh tone has been

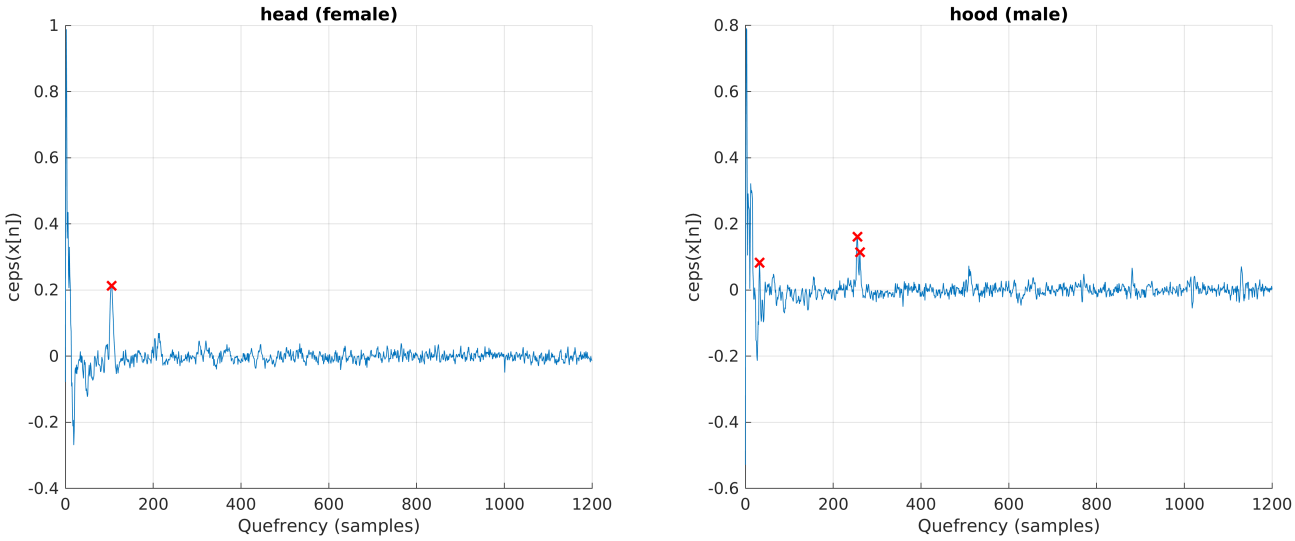


Figure 5: Real cepstrums with candidate pitch periods highlighted

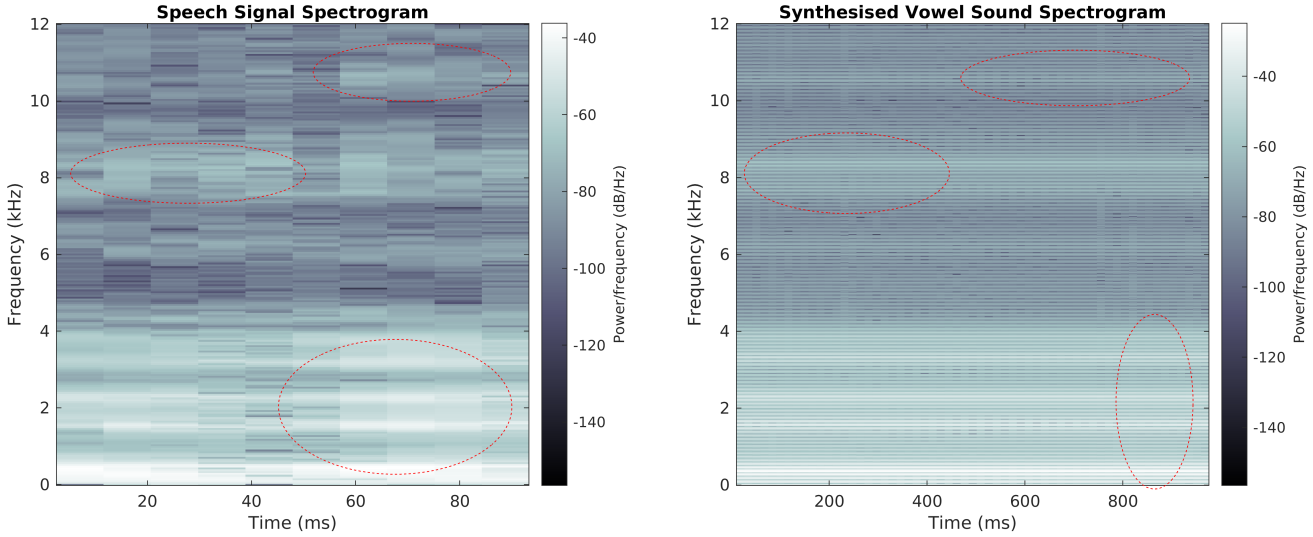


Figure 6: Spectrograms for the original and synthesised vowel segment, areas of comparison highlighted

smoothed and the sample subjectively sounds as close to human speech as could be achieved. Beyond this order, although the sound does change and smooth, it does not appear to further approach the quality of the original sound.

4 Discussion

As presented, the order of the LPC filter is a critical parameter for audio quality. An order that is too low will not allow the filter to accurately map to the desired vowel spectrum leaving a sound that, although at the right pitch, does not appreciably sound like the source segment. At the other end, increasing the order beyond a certain complexity can result in diminishing returns. Although the sound sounded smoother, beyond around order 40 it did not noticeably further approach the original sound. Subjectively, an order of 30 provided a good approximation of the input sound with acceptable quality for low bandwidth transmission.

The use of low-pass filtering on the cepstrum when identifying the fundamental frequency was effective in accentuating the peak corresponding to the pitch period. With this, a higher y threshold could be used that would be further from the noise of the function while still consistently identifying the correct peak.

The relative frequencies for male and female speech was as expected with the male speech segment having both lower fundamental frequencies and formant frequencies.

5 Conclusion

Within this work, a complete source-filter model of speech has been presented, analysing vowel samples and re-synthesising them while compressing the data representation. The effect of changing the complexity of this representation was investigated by varying the order of the LPC filter and describing the effect on the final audio sample. Various statistics about the original samples were calculated including the formant frequencies and the fundamental frequency. With a sufficient filter order, sound samples comparable to human speech were generated.

References

[1] H.-S. Kim. (2014). "Linear predictive coding is all-pole resonance modeling," Center for Computer Research in Music and Acoustics, Stanford University, [Online]. Available: https://ccrma.stanford.edu/~hskim08/lpc.

[2] R. Mannell. (Mar. 2020). "Source-filter theory of speech production," Department of Linguistics, Macquarie University, [Online]. Available: https://www.mq.edu.au/about/about-the-university/faculties-and-departments/medicine-and-health-sciences/departments-and-centres/department-of-linguistics/our-research/phonetics-and-phonology/speech/acoustics/acoustic-theory-of-speech-production/source-filter-theory.

[3] T. Bäckström. (Aug. 2020). "Fundamental frequency (f0)," Aalto University, [Online]. Available: https://wiki.aalto.fi/pages/viewpage.action?pageId=149890776.

[4] Whitman College. (). "Maxima and minima," [Online]. Available: https://www.whitman.edu/mathematics/calculus_online/section05.01.html.

[5] MathWorks. (). "Islocalmax, Find local maxima," MathWorks, [Online]. Available: https://www.mathworks.com/help/matlab/ref/islocalmax.html.

[6] A. Oppenheim and R. Schafer, "From frequency to quefrency: A history of the cepstrum," *Signal Processing Magazine, IEEE*, vol. 21, pp. 95–106, Oct. 2004. DOI: 10.1109/MSP.2004.1328092. [Online]. Available: https://www.researchgate.net/publication/3321562_From_Frequency_to_Quefrency_A_History_of_the_Cepstrum.

[7] S. Scherer, G. Lucas, J. Gratch, A. Rizzo, and L.-P. Morency, "Self-reported symptoms of depression and ptsd are associated with reduced vowel space in screening interviews," *IEEE Transactions on Affective Computing*, vol. 7, pp. 1–1, Jan. 2015. DOI: 10.1109/TAFFC.2015.2440264. [Online]. Available: https://www.researchgate.net/publication/279164505_Self-Reported_Symptoms_of_Depression_and_PTSD_Are_Associated_with_Reduced_Vowel_Space_in_Screening_Interviews.

A Source Code

While much of the code was developed in individual scripts in order to experiment with separate aspects of the system, for collecting results a script which constitutes the entire system was written, `lpss.m`.

Additional helper functions were written to plot and manipulate data.

Listing 1: Main script including source-filter model and spectral analysis

```
%% lpss.m
%%
%% Coursework script

close all;clear all;clc;

NAME = 'hood_m';
% NAME = 'head_f';

SEGMENT_LENGTH = 100; % ms
SEGMENT_OFFSET = 20; % ms from start

LPC_ORDER = 30;
AC_DISP_SAMPLES = 1000; % autocorrelation display samples
WINDOW_NUMBER = 10; % number of windows for spectrogram
WINDOW_OVERLAP = 10; % ms
SYNTH_WINDOW_NUMBER = 60; % number of windows for spectrogram
SYNTH_WINDOW_OVERLAP = 20; % ms

PREEMPHASIS_COEFFS = [1 -0.9]; % first order zero coeff for pre-emphasis

F0 = 60; % low-pitched male speech
% F0 = 600; % children

% flags for selective running
PREEMPHASIS = false;
CEPSTRUM_LOW_PASS = true; % smooth cepstrum for fund. freq. isolation
CEPSTRUM_LOW_PASS_COEFFS = [1 -0.7];

FREQ_RESPONSE = true;
AUTOCORRELATION = false;

CEPSTRUM_COMPLEX = false; % else real cepstrum
CEPSTRUM_PLOT = true;
```

```
CEPSTRUM_THRESHOLD = 0.075; % threshold for isolating peaks in cepstrum

ORIG_LPC_T_COMPARE = false;

ORIG_SPECTROGRAM = true;
SYNTH_SPECTROGRAM = true;

SYNTHESISED_SOUND_LENGTH = 100; % ms

WRITE = ~true;
PLAY = ~false;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% READ SIGNAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[y, Fs] = audioread(strcat('samples/', NAME, '.wav'));
% take segment of sample for processing
y = clip_segment(y, Fs, SEGMENT_LENGTH, SEGMENT_OFFSET);
y_orig = y;

if PREEMPHASIS
    y = filter(PREEMPHASIS_COEFFS, 1, y);
end

L = length(y); % number of samples

max_lag = Fs/ F0; % for autocorrelation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LPC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a = lpc(y, LPC_ORDER) % signal, filter order
est_y = filter(0.02, a, y);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% COMPARE ORIGINAL SIGNAL WITH LPC (T DOMAIN)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ORIG_LPC_T_COMPARE
x = 1:AC_DISP_SAMPLES;
AC_DISP_SAMPLES = min([AC_DISP_SAMPLES L]);
```

```
% plot t domain for original signal and estimation using LPC coeffs

figure(1)
plot(x, y(end-AC_DISP_SAMPLES+1:end), x, est_y(end-AC_DISP_SAMPLES+1:end),
    '--')

gridh
xlabel('Sample Number')
ylabel('Amplitude')
legend('Original signal','LPC estimate')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% T DOMAIN PREDICTION ERROR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t_domain_err = y - est_y; % residual?

if AUTOCORRELATION
figure(2)
[acs, lags] = autocorr(t_domain_err, max_lag, true, Fs);
title('Autocorrelation of error in time domain')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FREQUENCY RESPONSE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if FREQ_RESPONSE
figure(3)

%% ORIGINAL FFT
[freq_dom_freqs, freq_dom_vals] = fft_(y, Fs);

orig_freq_plot = plot(freq_dom_freqs, 20*log10(abs(freq_dom_vals)), 'black'
    );
orig_freq_plot.Color(4) = 0.25;
orig_freq_plot.LineWidth = 1;
hold on

%% LPC FILTER RESPONSE
[filter_vals, filter_freqs] = freqz(1, a, length(freq_dom_freqs), Fs);
filter_vals_db = 20*log10(abs(filter_vals));
```

```
lpc_freq_plot = plot(filter_freqs, filter_vals_db, 'b');
lpc_freq_plot.LineWidth = 2;

% MAXIMA
% estimate formant frequencies from maxima of LPC filter freq response
maxima = islocalmax(filter_vals_db);
maxima_freqs = filter_freqs(maxima)
maxima_db = filter_vals_db(maxima);

maxima_plot = plot(maxima_freqs, maxima_db, 'rx');
maxima_plot.MarkerSize = 12;
maxima_plot.LineWidth = 2;

%% PRE_FILTER LPC
if PREEMPHASIS
    [prefilter_vals, prefilter_freqs] = freqz(1, lpc(y_orig, LPC_ORDER),
       length(freq_dom_freqs), Fs);

    prefilter_plot = plot(prefilter_freqs, 20*log10(abs(prefilter_vals)), '
       g');
    prefilter_plot.Color(4) = 0.8;
    prefilter_plot.LineWidth = 1.5;
end

%% PLOT
hold off
grid
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
if PREEMPHASIS
    legend('Original Signal', 'LPC Filter', 'LPC Maxima', 'LPC No Pre-
       emphasis')
else
    legend('Original Signal', 'LPC Filter', 'LPC Maxima')
end
title('Frequency Response For Speech Signal and LPC Filter')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CEPSTRUM
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if CEPSTRUM_COMPLEX
    cep = cceps(y);
else
    cep = rceps(y);
end
cep_filt = filter(1, CEPSTRUM_LOW_PASS_COEFFS, cep);

if CEPSTRUM_PLOT % plot cepstrum in t domain
ceps_t = (0:L - 1);

if CEPSTRUM_LOW_PASS
    c = cep_filt;
else
    c = cep;
end

figure(4)
hold on
plot(ceps_t(1:round(L / 2)), c(1:round(L / 2)))

%% MAXIMA
% value threshold
c(c < CEPSTRUM_THRESHOLD) = 0;

% local maxima
cep_maxima_indexes = islocalmax(c);
cep_maxima_times = ceps_t(cep_maxima_indexes);
c = c(cep_maxima_indexes);

% quefrency threshold
cep_time_indexes = 20 < cep_maxima_times;
cep_maxima_times = cep_maxima_times(cep_time_indexes);
c = c(cep_time_indexes);

% 1st half
cep_half_indexes = cep_maxima_times <= round(L / 2);
cep_maxima_times = cep_maxima_times(cep_half_indexes);
c = c(cep_half_indexes);

maxima_plot = plot(cep_maxima_times, c, 'rx');
```

```
maxima_plot.MarkerSize = 8;
maxima_plot.LineWidth = 1.5;

grid
xlabel('Quefrency (samples)')
ylabel('ceps(x[n])')
xlim([0 L / 2])
title('Speech Signal Cepstrum')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE FUNDAMENTAL FREQUENCY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CEPSTRUM
if CEPSTRUM_PLOT && length(cep_maxima_times) >= 1
    pitch_period = cep_maxima_times(c == max(c))
    fundamental_freq = 1 / (pitch_period / Fs)
else
    disp('pitch periods not identified')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% GENERATE SIGNAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if exist('fundamental_freq')
    excitation = get_impulse_train(fundamental_freq, Fs,
       SYNTHESISED_SOUND_LENGTH);

    synth_sound = filter(1, a, excitation);

    if WRITE
    audiowrite(strcat('synthed/', NAME, '_o', num2str(LPC_ORDER), '_',
       num2str(SEGMENT_LENGTH), '_', num2str(SEGMENT_OFFSET), 'ms.wav'),
       synth_sound, Fs);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SPECTROGRAM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ORIG_SPECTROGRAM
```

```
figure(6)
spectro(y, Fs, WINDOW_NUMBER, WINDOW_OVERLAP);
colormap bone
title('Speech Signal Spectrogram')
end

if SYNTH_SPECTROGRAM
figure(7)
spectro(synth_sound, Fs, SYNTH_WINDOW_NUMBER, SYNTH_WINDOW_OVERLAP);
colormap bone
title('Synthesised Vowel Sound Spectrogram')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PLAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if PLAY
sound(y, Fs);
pause(1);
if exist('synth_sound')
    sound(synth_sound, Fs);
end
end
```

Listing 2: Spectrogram plotting wrapper function

```
function spectro(signal, sample_frequency, windows, overlap_interval)

sample_overlap = ms_to_samples(overlap_interval, sample_frequency);

%window_size = round(sample_size(1) / ((windows + 1)/2))

% Turn windows into window width in samples, take into account overlap
window_size = round(...
                (length(signal) + (windows + 1) * sample_overlap) ...
                    / ...
                (windows+1) ...
            );

spectrogram(signal, window_size, round(sample_overlap), [],
   sample_frequency, 'yaxis');
```

```
end
```

Listing 3: Fast Fourier transform wrapper function

```
function [frequencies, values] = fft_(signal, sample_frequency)

L=length(signal);

Y = fft(signal);
P2 = abs(Y); % two-sided spectrum
% P2 = abs(Y/L); % two-sided spectrum
P1 = P2(1:floor(L/2+1)); % single-sided spectrum
P1(2:end-1) = 2*P1(2:end-1);
frequencies = sample_frequency*(0:(L/2))/L;
values = P1;

end
```

Listing 4: Autocorrelation plotting wrapper function

```
function [cep_autocorr, cep_lags] = autocorr(signal, max_lags, time, Fs)

% [cep_autocorr, cep_lags] = xcorr(signal, round(max_lags), 'coeff');
[cep_autocorr, cep_lags] = xcorr(signal, 'coeff');

if time
    cep_lags = 1000*cep_lags/Fs; % turn samples into ms
end

plot(cep_lags, cep_autocorr)
grid
if time
    xlabel('Delay␣(ms)')
else
    xlabel('Delay␣(samples)')
end
ylabel('Normalized␣Autocorrelation')
title('Autocorrelation')
xlim([min(cep_lags) max(cep_lags)]);

end
```

Listing 5: Retrieve a segment of the original speech signal

```
function output = clip_segment(signal, Fs, seg_length, offset)

signal_length_samples = length(signal);
seg_length_samples = min(ms_to_samples(seg_length, Fs),
   signal_length_samples);
offset_samples = max(ms_to_samples(offset, Fs), 1);

seg_length_samples = min(seg_length_samples, signal_length_samples - 1);

if signal_length_samples < seg_length_samples + offset_samples
    offset_samples = signal_length_samples - seg_length_samples;
end

output = signal(offset_samples:offset_samples + seg_length_samples);

end
```

Listing 6: Transform time in milliseconds into the respective number of samples

```
function samples = ms_to_samples(time_in, sample_freq)
    samples = (time_in / 1000) * sample_freq;
end
```

Listing 7: Generate an impulse rate of given fundamental frequency at a provided sampling frequency for a given length of time

```
%% get_impulse_train.m
%%
%% Generate periodic impulse train for use in speech synth
%%
%% Signal of pitch fundamental_freq sampled at sampling_freq
%% for time length_ms
function signal = get_impulse_train(fundamental_freq, sampling_freq,
   length_ms)

if fundamental_freq > sampling_freq
    disp('Fundamental frequency greater than sampling_freq')
    signal = [];
    return
end

required_samples = ms_to_samples(length_ms, sampling_freq);
```

```
pitch_period = 1 / fundamental_freq;
sample_period = 1 / sampling_freq;

cell_length = round(pitch_period / sample_period);

% cell to be repeated into periodic signal
pitch_cell = [1 zeros(1, cell_length - 1)];
required_cells = ceil(required_samples / cell_length);

signal = repmat(pitch_cell, 1, required_cells);
signal = signal(1:required_samples);
end
```