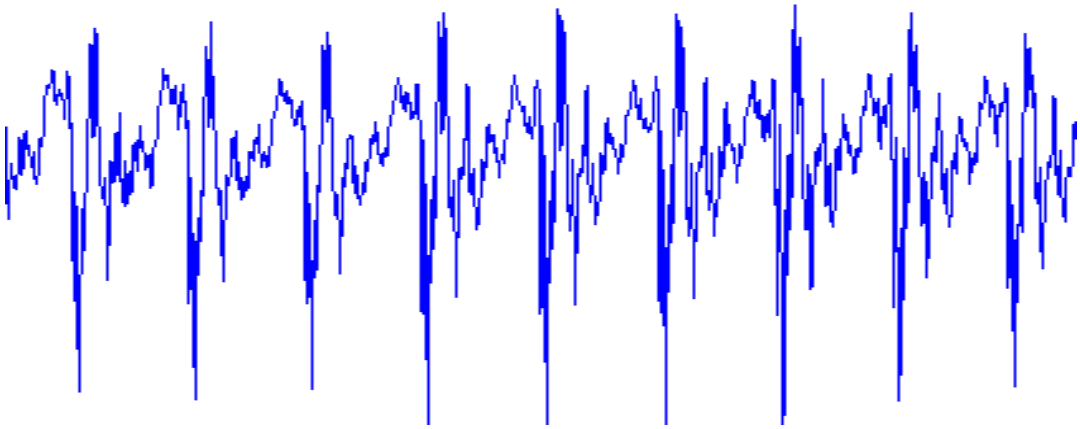

Linear Predictive Analysis on real speech samples

SYNTHESIS OF SOME VOWELS THROUGH THE LPC METHOD

Mihalas E Mrs (PG/T - Elec Electronic Eng) | Computer Vision | November 24, 2015

This report addresses the issue of synthesis of vowels given some real speech samples. It is required to use the Linear Predictive Coding (LPC) in order to estimate the all-pole filter coefficients that provide a reasonable approximation of the original vowels when convolved with a periodic impulse train. Moreover, it is requested to visualize the frequency response of the vocal tract by computing the frequency response of the all-pole filter, which holds the information about the formant structure of sounds.

The main purpose of this paper is to use the speech processing tools provided by MATLAB language in order to extract important features from the audio signal representation and to synthesize the original sound by using the source-filter model of speech production.

Introduction

The speech signal is produced at the vocal cords when air is forced through the vocal tract and then released from the mouth. The vocal tract forms a resonator and produces the characteristic vocal formants, which are regions of high-amplitude harmonics in the sound spectrum. It is common practice to examine mainly the first two formants frequencies when analyzing the quality of vowels though the number of formants in the spectral envelope always correspond to half the number of poles in the all-pole filter.

Speech sounds are divided into three classes: voiced sounds, unvoiced sounds and plosive sounds. This partition is mainly due to the source periodicity characteristics. In fact, the voiced sounds have a periodic source, which is generated by the vocal cords vibration, while the unvoiced sounds are produced by a noisy and aperiodic source. However, this paper discusses only the voiced sounds class, namely it focuses on temporal and spectral characteristics of vowels.

In vowel production, the rate of the cord vibration determines the pitch of the sound. The pitch period is variable and it is different for each person. Typically, the range is 80-155 Hz for male and 160-255 Hz for female. The fundamental frequency (f0) is the inverse of the pitch period and the harmonics are the multiples of f0.

The audio signal must be stationary and linear in order to be processed. Generally, the speech signal is non-stationary but it is assumed quasi-stationary over short periods of time, namely over 20-30ms. Moreover, as the vocal tract shape and its mode of excitation change relatively slowly, it is considered linear and time-invariant (LTI).

In speech processing, the Linear Predictive Coding method is robust and accurate in estimating the speech parameters. It uses *p-th* order linear predictor to predict the current samples $\hat{s}\,[n]$ from *p* previous samples.

$$\hat{s}\,[n] = -\sum_{k=1}^{p} (a_k \mathrm{s[n-k]})$$

This method uses the least square autocorrelation criterion in order to compute the LP coefficients a_k, which minimize the prediction error e[n].

$$e[n] = s[n] - \hat{s}[n] = s[n] \quad + \sum_{k=1}^{p} (a_k \mathrm{s[n-k]})$$

The above equation transferred in the frequency domain becomes:

$$E(z) = \mathrm{S(z)} + \sum_{k=1}^{p} \left(a_k * \mathrm{S(z)} * z^{-k}\right)$$

$$A(z) = \frac{E(z)}{S(z)} = 1 + \sum_{k=1}^{p} \left(a_k * z^{-k}\right)$$

The prediction error is also called the LP residual and it is obtained through the inverse filtering, namely the inverse operation is applied to LP spectrum H(z), which is given by the “lpc” Matlab function, then the result is multiplied by the vowel spectrum and transferred to the time domain.

Speech is modelled as the output of a linear, time-varying system, excited by periodic impulses. The time-varying digital filter represents the combined effects of the glottal pulse shape (spectral details), the vocal tract (spectral envelope) and the radiation at the lips (amplitude spectrum of the uttered vowel).

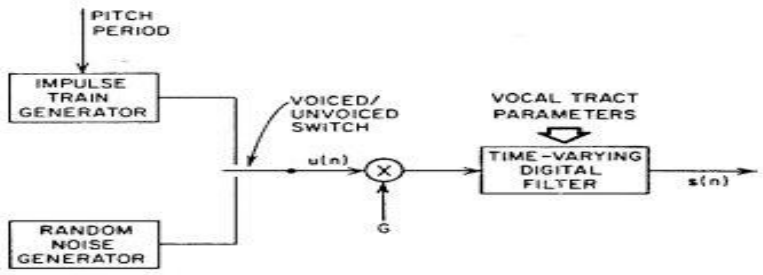

Fig. 1 The source filter model of speech production used to synthesize the samples of vowels.

Procedure

The processing of the vowel phoneme samples was done in Matlab by using the in-built functions for signal processing, such as *fft, filter, freqz, lpc, xcorr, spectrogram* ecc.

Two phoneme samples were selected from the folder, namely the “**hood_m.wav**” for male and “**hood_f.wav**” for female. A script file was created to hold the entire programming code. The code was designed to hold the input variables at the beginning of the code and to process only one audio signal at a time. Therefore, just one sample was uploaded on

Matlab using the "*wavread*" function, which read the file and extracted the sampling frequency (fs) from the phoneme sample. The vowel time trend was visualized through the "*plot*" function and the time axis was defined with the sampling step of `1/fs`. A quasi-stationary segment was chosen from the middle of the graph in both cases, namely 100ms segment. For each of these segments was performed the model estimation and the synthesized speech.

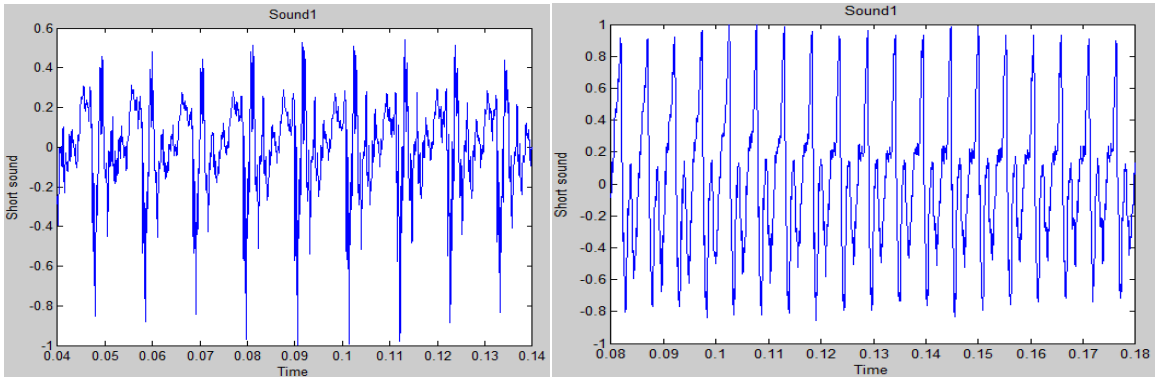

Fig.2 Two quasi-stationary segments of vowel "oo" utterance (100ms). On the left is represented the male utterance and on the right the female utterance.

The model estimation was implemented using the "*lpc*" function, which have used the autocorrelation method to estimate the coefficients of the all-pole filter. It was assigned a filter order of 30 for the male sample and of 50 for the female sample. Then the pole-zero plot of the transfer function was plotted on the screen.

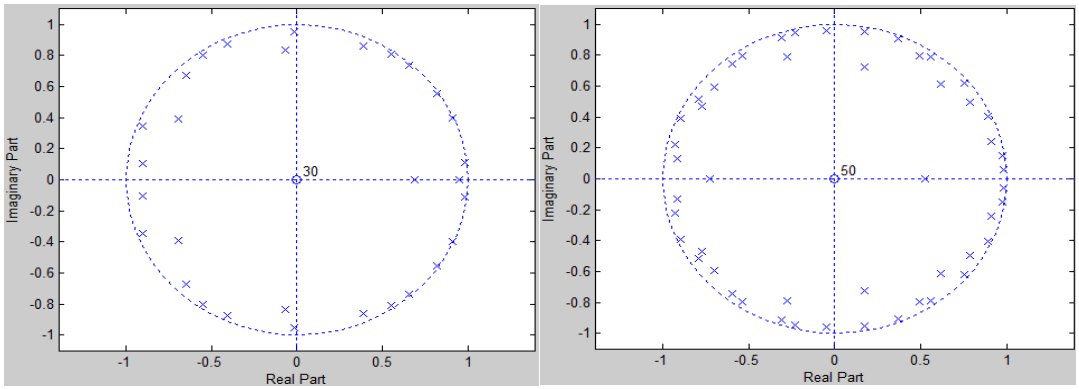

Fig.3 The pole-zero plot for the transfer function of the male vowel (on the left) and of the female vowel (on the right).

After the frequency axis was set, the frequency response of the LPC filter was computed using the "*freqz*" function and plotted over the amplitude spectrum (in dB) of the speech segment, which was obtained using the "*fft*" function.

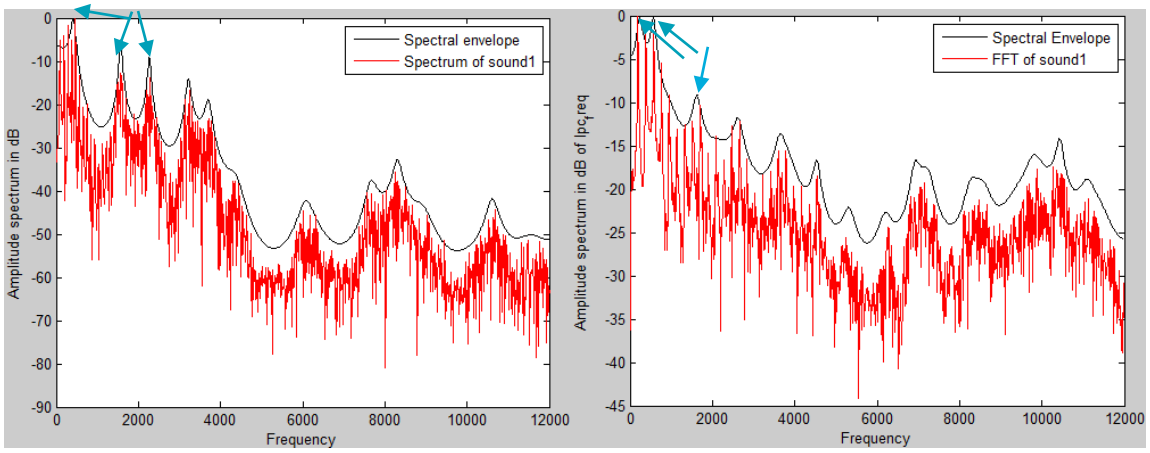

Fig.4 The spectral envelope and the amplitude spectrum of male vowel (on the left) and of female vowel (on the right).

The formant frequencies were determined by computing the roots of the transfer function denominator and by extracting the phase angles from the roots. Once the angles were obtained, the formant frequencies were estimated by using the following formula: `formants = (theta./(2*pi))*fs.` However, in order to select only the first three frequencies from the formants vector, the negative conjugates and the null elements were removed from the vector and the final vector, populated only with positive elements, was sorted in ascending order.
The LP residual was computed by applying the inverse filtering. Then the pitch detection was performed using the "*xcorr*" function, which has computed the autocorrelation of the residual over the range [60 : 250] Hz. The pitch period, instead, was determined by finding the index of the first peak after the zero lag, while the fundamental frequency was obtained by dividing the sampling frequency by the length of the lag.

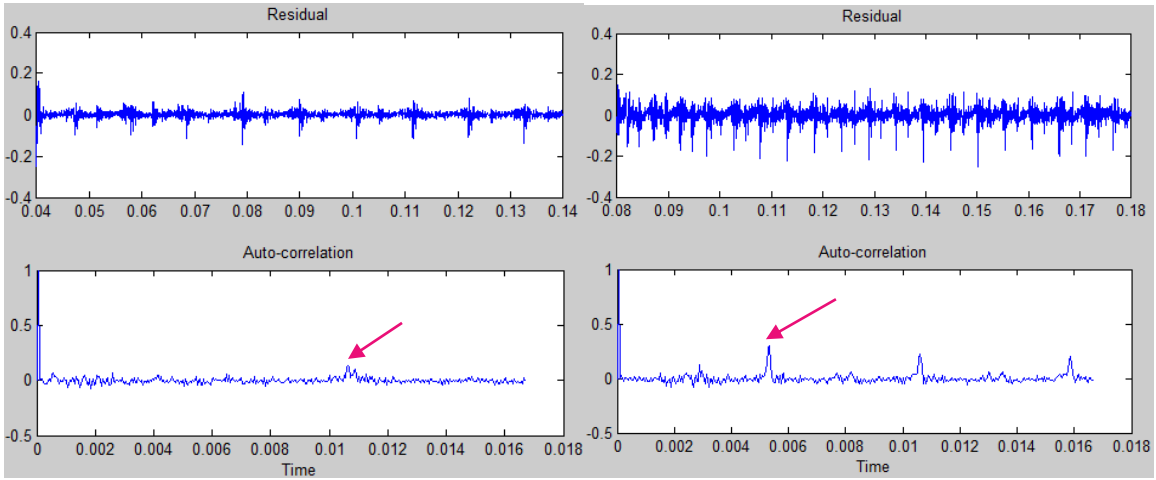

Fig.5 The autocorrelation function of the male residual signal (on the left) and of the female residual signal (on the right).

The source filter model was then applied in order to synthesize the vowels. Thus it was generated a periodic impulse train, with the impulses spaced by the pitch period, and then it was filtered using the LPC filter and the Matlab "*filter*" function.

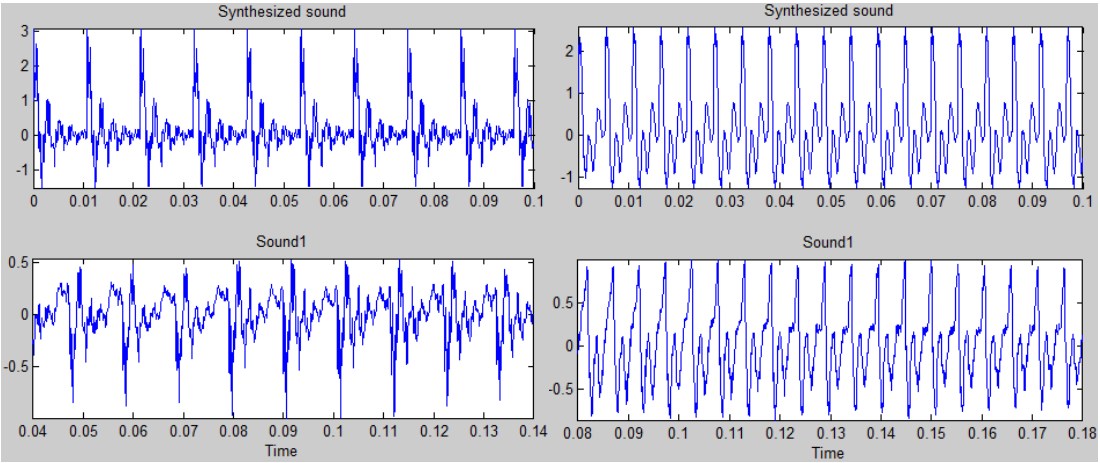

Fig.6 The time representation of both original vowel and the synthesized vowel for male (on the left) and for female (on the right) utterance.

Finally, the synthesized speech was transferred onto an audio file called “Synthesized_f.wav” and “Synthesized_m.wav”.

Results

The results that are discussed here refer to the following vowel samples: the “**hood_m.wav**” (male voice) and “**hood_f.wav**” (female voice).

Both utterances show a good periodicity. The audio signal has a duration of around 180ms for the male sample and 240ms for the female sample, and a sampling frequency of 24 kHz. In order to make the signal quasi-stationary a period of 100ms was chosen from the middle of the graph, namely from 40ms to 140ms for the male sample and from 80ms to 180ms for the female sample, which roughly corresponds to the vowel “oo” utterance.

It was required to use the LPC modelling because it is reliable and accurate especially for the non-nasal voiced speech. In fact, in both cases it was obtained a very good fit of the estimated coefficients to the amplitude spectrum with only a filter order of 30 for the male utterance and 50 for the female utterance. It was decided to increase the filter order for the female utterance because it was necessary to discriminate better the first two formant frequencies responsible for the synthesized speech quality.

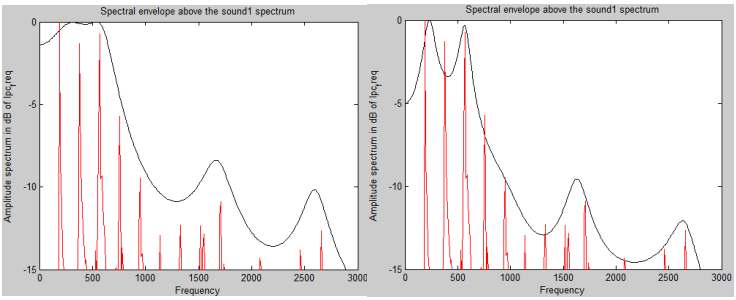

Fig.7 The spectral envelope trend for the female utterance determined by the all-pole filter of order 30 (on the left) and of order 50 (on the right).

By looking at the pole-zero plot it can be seen that the LPC method generates finite filters which satisfy the bounded input bounded output (BIBO) stability criterion, namely the transfer function gathers all the poles and zeros within the unit circle. Therefore, the spectral envelope has a decreasing trend and a finite energy.

The values of the first three formant frequencies are very different between the two utterances. In fact, the female formant frequencies values are 231.36, 569.53, 933.96, while the male formant frequencies are 421.61, 1571.7, 2269. It can be noticed that the female formants are very close to each other. This proximity between the formants is due to women's high-pitch voice and it is difficult to manage in order to obtain a synthesized voice that seems natural. Moreover, the formants risk to be assembled when the filter has a low number of poles. On the other hand, the male's formant frequencies are well spaced and this determines the good quality of the synthesized vowel.

The pitch period was computed by spotting the second peak in the autocorrelation function of the residual signal. The first peak was ignored because it defines the residual signal's energy. Therefore, the pitch period value for the male utterance is 10.67ms and for the female utterance is 5.37ms, namely it was found a fundamental frequency of 93.75 Hz and of 186.05Hz respectively. Thus the female utterance has a fundamental frequency twice as high as the male utterance.

Once the transfer function parameters and the fundamental frequency were computed, the synthesized vowels were produced by convolving the parameters with the excitation signal, namely a train of impulses spaced by the pitch period. The train of impulses was created with the option of making it longer than the original sound in case the synthesized sound has not been heard well. If, instead, it is necessary to make the vowel sound louder, it is suggested to increase the value of the gain of the transfer function.

The main steps of speech production are summarized by the below block diagram.

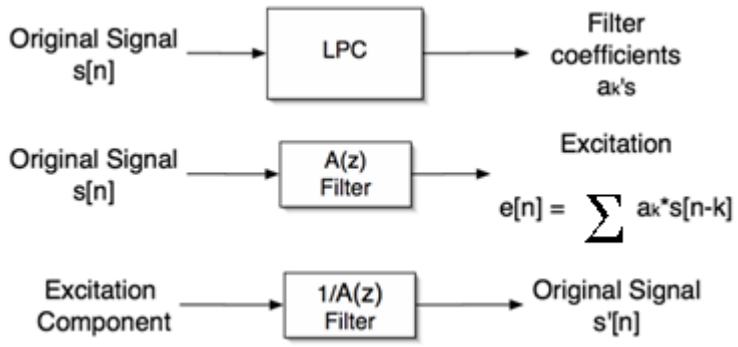

Fig.8 The main steps in speech production when implementing the LPC analysis.

Conclusion

The above procedure has provided interesting results. For example, it was noticed that by increasing the number of poles of the transfer function the prediction error decreases and the spectral envelope fits better to the amplitude spectrum of the original sound. However only the first few formants frequencies are important in the modelling process because they encompass all the essential information related to the signal quality.

As regards the synthesized sound, it was observed that the vowel “oo” has been perceived clearly during the male utterance while it has not sounded human and natural during the female utterance. The poor quality of the synthesized sound for female voice may be related with the fundamental frequency value, which is much higher than the male’s fundamental frequency, or even with the all-pole filter assumption, which does not estimate the valleys while fitting the spectral envelope to the speech segment amplitude spectrum. Therefore, it could be worth trying to implement the cepstral analysis for the female vowel, in order to discover if the synthesized sound’s quality could be improved by a transfer function which models both peaks and valleys of the sound amplitude spectrum.

References

- W. Wang, *Speech and Audio Processing and Recognition*, speech and audio processing (part 2) Lecture Notes, Guildford: University of Surrey, 2015.
- W. Wang, *Speech and Audio Processing and Recognition*, speech and audio processing (part 3) Lecture Notes, Guildford: University of Surrey, 2015.
- A. M. Kondoz, Digital Speech. West Sussex: John Wiley & Sons, ch. 3, 1994.
- L. R. Rabiner and R. W, Schafer, Digital Processing of Speech Signals. New Jersey: Prentice-Hall, ch.8, 1978

Appendix

The Matlab script, containing the code that implements the speech processing discussed above, is attached in this appendix.

```
% I have to estimate one male vowel and one female vowel from a set of
% samples and then to synthesize them

clear all          % clear the workspace
close all          % close all the previous figures

%input variables

tmin=0.08;         % the min and max of my new time frame
tmax=0.18;         % t=(tmax-tmin)sec
rank_filter=50;   % order of the lpc filter -> 20:30 usually is sufficient
to model
                   %         |                 % the first few formants
                   %         |-> as rank_filter increases the
                   %         estimated coefficients result in a closer
                   %         fit to the actual spectral shape
name1='hood_f.wav';      % The name of the original sound
name2='Synthesized_f';  % The name of the synthesized sound

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Read, plot and play the speech signal %%%%%%%%%%%%%%%%%%%%%%%%%%

[sound,fs]=wavread(name1);    % Read the female

N=length(sound);              % number of samples in the time frame
dt=1/fs;                      % time increment per sample
t=0:dt:dt*(N-1);              % time axis

figure;
plot(t, sound)
xlabel('Time-original time frame')
ylabel('Original sound')
%s = audioplayer(sound, fs);  % play sound
%play(s)
%wavplay(sound,fs)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Selection of the stationary segment %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

idxmin=find(t==tmin);
idxmax=find(t==tmax);          % index at which t=tmin and t=tmax
t1=t(idxmin:idxmax);           % stationary segment of 100ms
sound1=sound(idxmin:idxmax);
N1=length(sound1);             % number of samples in the new time frame

figure;
plot(t1, sound1)               % Plot the stationary segment
xlabel('Time')
```

```
ylabel('Short sound')
title('Sound1')
wavplay(sound1,fs)
%s0 = audioplayer(sound1, fs);  % play sound
%play(s0)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% LPC coefficient estimation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[coeff,e]=lpc(sound1, rank_filter); % LPC calculates the coefficients of
an
                                    % all-pole filter from real speech
                                    % signal. It provides the transfer
function
                                    % of the vocal tract


figure;
zplane(1, coeff, 'r')      % pole-zero plot -> shows if the system is
stable


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% The amplitude spectrum of the speech stationary segment %%%%%%%

sound1_freq=fft(sound1);                          % fourier transform of
sound1
df=fs/N1;                                        % frequency increment per
sample
f=0:df:df*(N1-1)/2;                               % half of the frequency
range for
                                                  % data -> the signal
repeats itself
                                                  % after the Nyquist
frequency, which is fs/2
M=10*log10(abs(sound1_freq./max(sound1_freq)));  % normalised Magnitude
of sound_freq

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Spectral Envelope %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5%%%%%

G=1;                                             % The gain of the transfer
function

lpc_freq=freqz(G,coeff,f,fs);                    % fourier transform of lpc
filter
M0=10*log10(abs(lpc_freq./(max(lpc_freq))));  % normalised Magnitude of
the lpc filter
figure;
plot(f, M0,'k-')
hold on                                         % overlap of the sound_freq
and lpc_freq
                                                % magnitude spectrums
```

```
plot(f,M(1:(length(M)+1)/2), 'r-')
xlabel('Frequency')
ylabel('Amplitude spectrum in dB of lpc_freq')
title('Spectral envelope above the sound1 spectrum')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% The first three formant frequencies estimation %%%%%%%%%%%%%%%

poles=roots(coeff);

theta=angle(poles);

formants=(theta./(2*pi))*fs;       % find the formant frequencies
positive=(formants>0);               % ignore the negative conjugates of
the formant frequency
formants=formants.*positive;
formants=formants(find(formants));  %  select  only  the  positive  values
inside the vector
formants=sort(formants);           % sort the values inside the vector in
ascending order


first_formant=formants(1,1)
second_formant=formants(2,1)
third_formant=formants(3,1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  Fundamental  frequency  (f0)  or  pitch  estimation  throught  the
maximization of
%%% the normalised autocorrelation of sound1  %%%%%%%%%%%%%%%%%%%%%%%%%


% Pitch frequency occurs between 60Hz for low-pitched male speech and up
to
% 250Hz for women: [60:250]->m=fs/f0=[24000/250:24000/60]=[96:400]

residual=filter(coeff,G, sound1);    % estimate the residual by using the
                                      % inverse filtering A(z)=[coeff,1]
figure;
subplot(2, 1, 1); plot(t1, sound1);
title('Sound1');
subplot(2, 1, 2); plot(t1, residual);
title('Residual');

Mmin=96;            % lowest lag period
Mmax=400;           % highest lag period


auto_corr = xcorr(residual, Mmax, 'coeff' );    % xcorr computes the
                                                %  autocorrelation  over
the range -Mmax:Mmax
                                                %
length(auto_corr)=2*Mmax+1
```

```
auto_corr = auto_corr(floor(length(auto_corr)/2):end);   % half is just
mirror for real signals
[max_ac,idx]=max(auto_corr(Mmin:end));                    % Search for
maximum between 60Hz:250Hz
                                                  % -> i exclude zero-lag
because it determines
                                                  % the average power in the
signal, not the fundamental frequency
                                                  % idx is the index of the
maximum value of the autocorrelation
f0 = fs/(Mmin+idx-1)                              % fs divided by the length
of the lag

figure;
subplot(2, 1, 1); plot(t1, residual);
title('Residual');
d=(0:Mmax+1)/fs;                                  % Convert the samples into
time
subplot(2, 1, 2); plot(d,auto_corr);
xlabel('Time')
% ylabel('Amplitude spectrum in dB')
title('Auto-correlation')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Generation of the synthesized sound %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dt_pitch=1/f0;
idx_pitch=round(dt_pitch/dt);    % computes how many dt are contained in
a dt_pitch

imp_length=N1;                  % I may want to make the synthesized sound
longer
impulse=zeros(1,imp_length);     % train of impulses with T=dt_pitch
i=1;
while i<=imp_length
    impulse(i)=1;
    i=i+idx_pitch;
end



synthesized=filter(G,coeff,impulse);   % convolution of the excitation
signal
                                      % and the transfer function of the
vocal tract
N2=length(synthesized);
t2=0:dt:dt*(N2-1);

figure;
subplot(2,1,1)
plot(t2, synthesized)
title('Synthesized sound')
axis tight
```

```
subplot(2,1,2)
plot(t1, sound1)
xlabel('Time')
title('Sound1')
axis tight

s1 = audioplayer(synthesized, fs);                % play sound
play(s1)

%wavwrite(synthesized,fs,name2);
%synthesized_sound=wavread(strcat(name2, '.wav'));
%wavplay(synthesized_sound,fs)
```