ATILIM UNIVERSITY

EE519 SPEECH PROCESSING AND APPLICATIONS

Assignment- 3

Instructor: **Assist. Prof. Dr. BARAN USLU**

Student: **NURAY GUL – 130505009**

11.01.2015

Introduction:

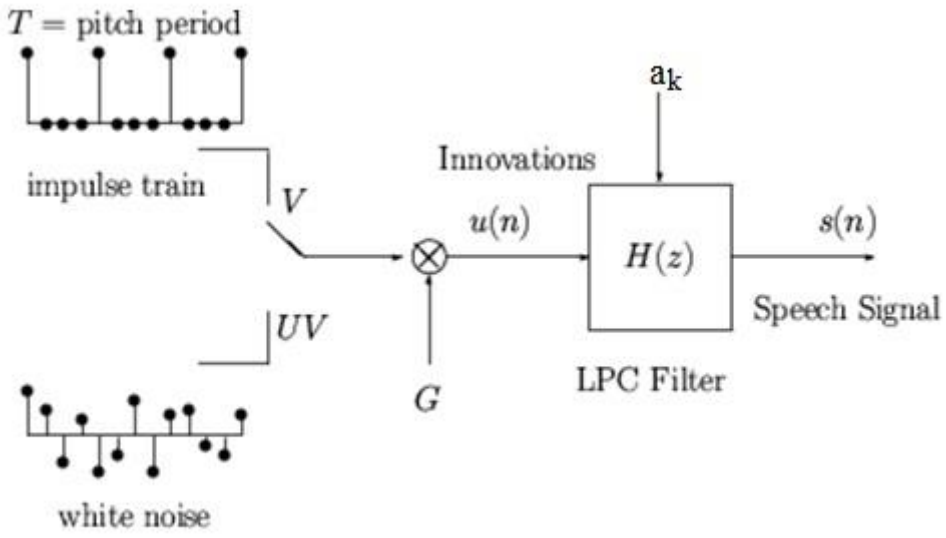
This project aims to do Linear Predictive Coding analysis and synthesis for two different speech signals. We used 2 utterances which are mono recorded speeches.We used the Matlab's own **'lpc.m'** function to analyze the speech files firstly. This analyze gives us the LPC coefficients and gains and we wrote a function **'syn_lpc.m'** to synthesize this speech by using these LPC coefficients and gains. LPC is good for characterizing the general spectral shape of a signal, which may be time-varying as in speech sounds. For synthesis, any source can be filtered, allowing the general spectral shape of one signal (used in analysis) to be applied to any source. For these reasons we try to do implementation of LPC of speech.

Background:

The linear predictive coding (LPC) method for speech analysis and synthesis is based on modeling the Vocal tract as a linear All-Pole (IIR) filter having the system transfer function:

$$H(z) = \frac{G}{\prod_{k=1}^{p} (1 - a_k)(1 - a_k^*)}$$

Simple speech production



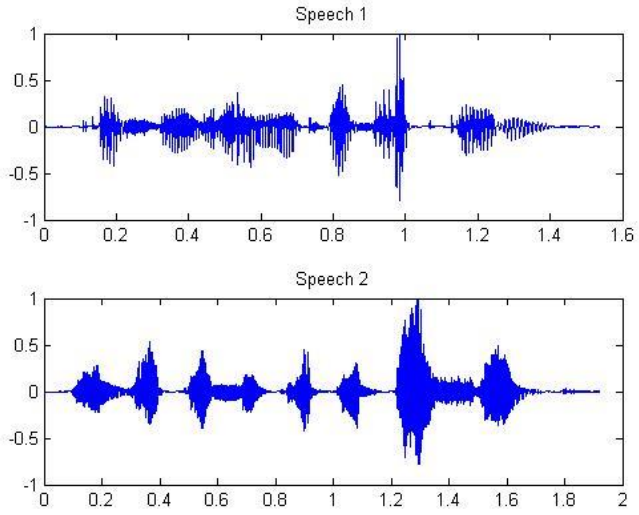
Where **p** is the order (number of poles), G is the filter Gain, and a[k] are the coefficients that determine the poles. There are two mutually exclusive ways excitation functions to model voiced and unvoiced speech sounds. For a short time-basis analysis, voiced speech is considered periodic with a fundamental frequency of Fo, and a pitch period of 1/Fo, which depends on the speaker. Hence, Voiced speech is generated by exciting the all pole filter model by a periodic impulse train. On the

other hand, unvoiced sounds are generated by exciting the all-pole filter by the output of a random noise generator.

Methodology:

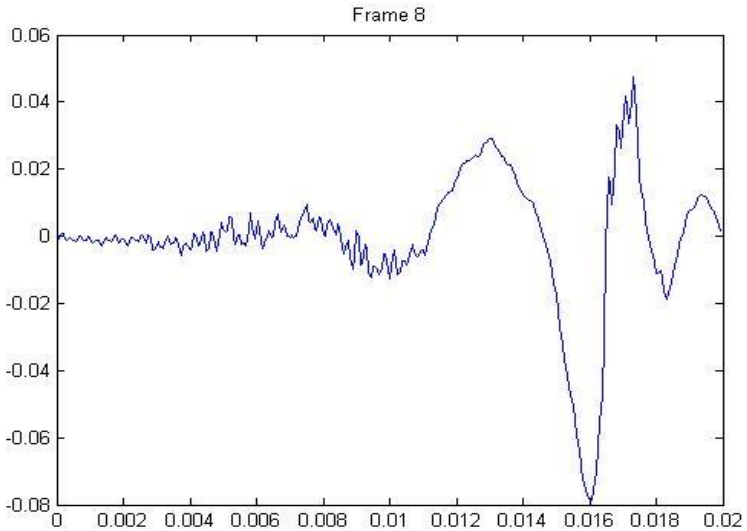
According to background part we can start our project now.

1) Firstly we choose 2 utterances which are recorded as mono speech signals as shown below.



2) Speech signals are highly non-stationary signals. This makes difficult to analyze the signals correctly. So we need to segment signals in short durations. We divide speech signals in 20 msec short frames by applying Hamming window .

- **`win = hamming(FrameLen);`**



One of the frame sample(20 msec)

3) After windowing we can start our LPC analyze. Firstly we define LPC order number which calculated as:

- $p \approx 2\times$Bandwidth of signal (in kHz)$+[2,3,4]$
- e.g. BW=4 kHz, then

  $p = 2\times4 + [2,3,4] \in [10,11,12]$

In our Project we choose **P=18** because our sampling frequency Fs=16000 Hz so we think that (according to Nyquist) Bandwidth = Fs/2=8 KHz. So,

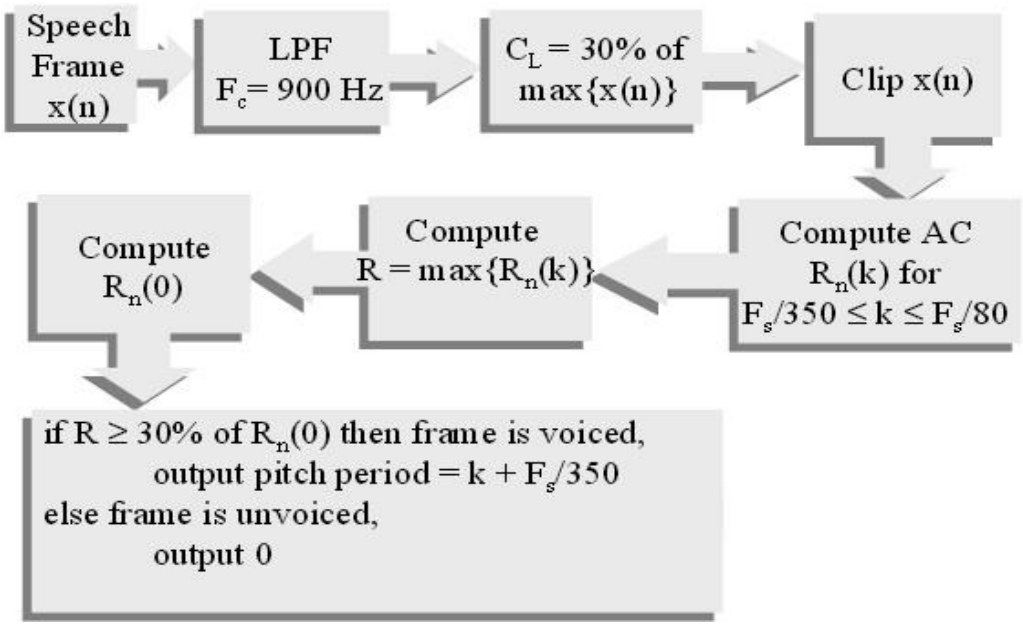
P=2*BW+[2,3,4]=[18,19,20]

4) Then we calculate A and G by using 'lpc.m' which is MATLAB's own function.

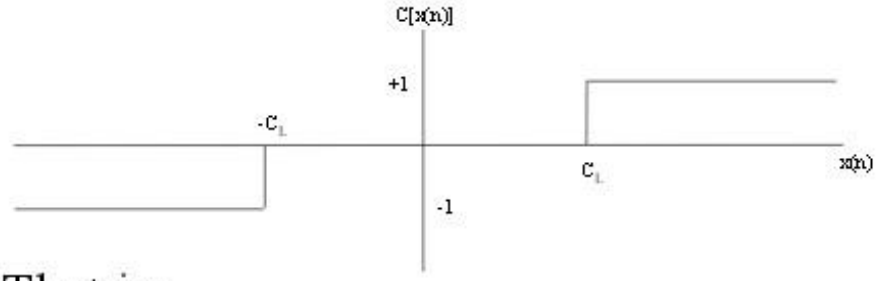
```
%% Calculation of LPC coefficient (A) and Gain (G)
  [A(:,i+1),G(i+1)] = lpc(speech1,P);

A=Filter coefficient Matrix (P+1xnumberoffframes)
G=Gain vector(Length=numberoffframes)
```

5) After these operations according to background part we need to decide frames type if they are voiced or unvoiced. If they includes voiced parts we need to calculate pitch period of them to generate impulse train. So we firstly make Pitch Detection according to following algorithm.



6) For these reasons firstly we clip the frames using 3-level center clipping function:



That is:

$$C[x(n)] = \begin{cases} +1 & if \quad x(n) > C_L \\ -1 & if \quad x(n) < -C_L \\ 0 & otherwise \end{cases}$$

7) We wrote a MATLAB Script as follows:

```
function speech1= clip_center(speech1)
N=length(speech1);
cliplevel = max(speech1).*0.3; % 30 percent of max(speech1)
for j = 1:N
  if (speech1(j)) > cliplevel
    speech1(j) = 1;
  elseif (speech1(j)) < cliplevel
      speech1(j) = -1;
  else
      speech1(j) = 0;
  end
end
end
```

8) And we use this function and clip the frames

```
% Clipping the signal to Pitch Detection
speech2= clip_center(speech1);
```

9) Then we use autocorrelation method to find pitch periods for each frames. But we define a modified autocorrelation function for our Project.

$$R_n(k) = \sum_{m=0}^{N-k-1} x(m)x(m+k)$$

- where $x(m)x(m+k)$ can have only 3 different values:

$$x(m)x(m+k) = \begin{cases} +1 & if \quad x(m) = x(m+k) \\ -1 & if \quad x(m) \neq x(m+k) \\ 0 & if \quad x(m) = 0 \; or \; x(m+k) = 0 \end{cases}$$

10) We wrote a function code as follows

```
function [ Rn ] = modified_autocorrelation( speech1 )
N=length(speech1);

for k=0:N
for m=1:N-k
    a(m,k+1)=speech1(m)*speech1(m+k);
  if speech1(m)==speech1(m+k)
     a(m,k+1)=1;
  elseif speech1(m)~=speech1(m+k)
     a(m,k+1)=-1;
  elseif speech1(m)==0 || speech1(m+k)==0
    a(m,k+1)=0;

  end
  Rn(k+1)=sum(a(:,k+1));
end
end
end
```

11) And we use this function to calculate all values of autocorrelation frame by frame.

- We don't need to compute $R_n(k)$ for all values of k (i.e. $0 \le k \le N$)

	F_0 (Hz) min	F_0 (Hz) max
men	80	200
women	150	350

- Thus we only need to look in the range:

$$80\ \text{Hz} \le F_0 \le 350\ \text{Hz}$$

12) So we just calculate F0 between 80 Hz to 350 Hz. We apply the Pitch Decection Algorithms and we calculate all pitch frequencies which are necessary for us and decide the voiced and unvoiced parts. Pitch Period is '0' for unvoiced parts.

13) Until now we try to analyze our speech signals. Now we can pass the synthesis part. So firstly we need to define impulse train and random noise generators. We wrote 2 different function for them.

```
function [ y ] = impulsetrain( srate,f0,frame_dur)
  f=f0; %pitch frequency of a frame
  fs=srate; % sample rate
  framelen=fs*frame_dur/1000;
  n=0:1:framelen;% time vector
  y=zeros(size(n));
  y(1:f:length(y))=1;
end
```

```
function [ noise ] = random_noise( srate,frame_dur )
Frame_Len=srate.*frame_dur./1000;
noise = wgn(Frame_Len, 1, 10, 'real');
end
```

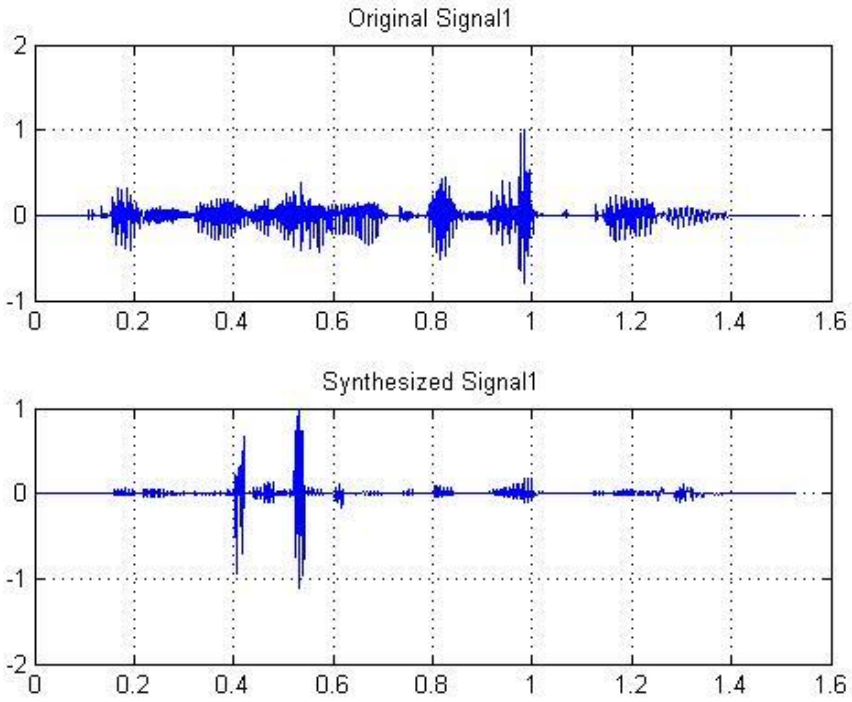
14) After these functions we wrote another function which is doing lpc synthesis by filtering impulse train generator's output (if the frame is voiced) or random noise generator's output (if the frame is unvoiced) with LPC coefficients A and gains G.

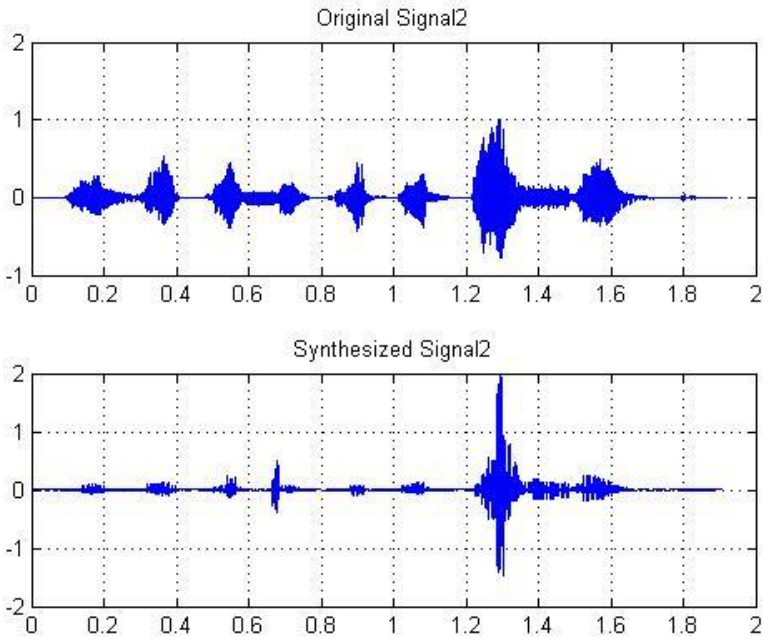
```
function [signal,t]=syn_lpc(srate,f0,frame_dur,A,G)
if f0~=0
    y  = impulsetrain( srate,f0,frame_dur);%voiced parts
else
    y=random_noise( srate,frame_dur);% unvoiced parts
    y=y'; % make all of them column vectors
end
signal= filter(G,A',y); %filters the data in vector y with the
                        %filter described by vectors A' and G to create the filtered data signal.
t =(0:length(signal)-1)/srate;
end
```

15) Then we call it in our main script to synthesize the signal

```
[signal,t]=syn_lpc(fs,pitchFrequency(i+1),frame_dur,A(:,i+1),G(i+1));
%syntehize the signal by using syn_lpc function
 signalnew=[signalnew;signal'];
% syntehized the signal by collect all frames
```

16) And we finally plot the synthesized speech signals as below,





17) As a result,to listen the synthized speech signals we increase the gain and when we listen it we hear a robotic sound. This means that we don't get the same as original but we get the general characterized properties of that speech and it's useful for a lot of areas such as speech coding, speech synthesis, speech recognition, speaker recognition and verification and for speech storage.

Appendix : All codes

```
% Clipping function
function speech1= clip_center(speech1)
N=length(speech1);
cliplevel = max(speech1).*0.3; % 30 percent of max(speech1)
for j = 1:N
  if (speech1(j)) > cliplevel
    speech1(j) = 1;
  elseif (speech1(j)) < cliplevel
      speech1(j) = -1;
  else
      speech1(j) = 0;
  end
end
end
```

```
% Moddified Autocorrelation Function
function [ Rn ] = modified_autocorrelation( speech1 )
N=length(speech1);

for k=0:N
```

```
for m=1:N-k
    a(m,k+1)=speech1(m)*speech1(m+k);
  if speech1(m)==speech1(m+k)
     a(m,k+1)=1;
  elseif speech1(m)~=speech1(m+k)
     a(m,k+1)=-1;
  elseif speech1(m)==0 || speech1(m+k)==0
    a(m,k+1)=0;

  end
  Rn(k+1)=sum(a(:,k+1));
end
end
end
```

```
% Impulse Train Generator
function [ y ] = impulsetrain( srate,f0,frame_dur)
 f=f0; %pitch frequency of a frame
 fs=srate; % sample rate
 framelen=fs*frame_dur/1000;
 n=0:1:framelen;% time vector
 y=zeros(size(n));
 y(1:f:length(y))=1;
end
```

```
% Random_noise Generator
function [ noise ] = random_noise( srate,frame_dur )
Frame_Len=srate.*frame_dur./1000;
noise = wgn(Frame_Len, 1, 10, 'real');
end
```

```
% LPC Synthesizer
function [signal,t]=syn_lpc(srate,f0,frame_dur,A,G)
if f0~=0
    y  = impulsetrain( srate,f0,frame_dur);%voiced parts
else
    y=random_noise( srate,frame_dur);% unvoiced parts
    y=y'; % make all of them column vectors
end
signal= filter(G,A',y); %filters the data in vector y with the
                        %filter described by vectors A' and G
to create the filtered data signal.
t =(0:length(signal)-1)/srate;
end
```

```
% ---------------------%
% EE519 Assignment 3  %
% NURAY GUL-130505009 %
%---------------------%
%% This script is finding LPC coefficients and gains for a
speech signal
% And it resynthesizes the speech by using these values %
%%
clear all;
clc;
close all;
[speech,fs,nbits] = wavread('word1Anger.wav'); % Read Mono
speech
                                                % fs=16000
Hertz
sound(speech,fs)    % Listen it
%% Normalization
speech=(speech-mean(speech)./(max(abs((speech -
mean(speech))))));%Normalize the signal
%% Frame Properties
frame_dur=20; %20 ms
FrameLen=frame_dur*fs/1000; %Length of frame
y=mod(length(speech),FrameLen);% last frame's length
m=length(speech);%length of speech
bolum=(m-y)/FrameLen; % number of frames
%% Applying Hamming Window
win = hamming(FrameLen);
%% LPC Order
P=35;% P is the order of LPC and it calculated as
     % P~Bandwidth(in KHz)*2+[2,3,4]
     % if BW=8 KHz P=2*8 +[2,3,4]=[18,19,20]
pitchFrequency=[]; % pitch periods for each frames
signalnew=[]; %resynthesized speech
tson=[]; % time for resynthesized speech
%% Loop
for i=0:1:bolum-1
speech1 = win.*speech((i)*FrameLen+1:(i+1)*FrameLen,1);
%signal for one frame
tframe=(0:length(speech1)-1)/fs;
%figure,plot(tframe,speech1)
%title(['Frame ', num2str(i+1)])
%% Calculation of LPC coefficient (A) and Gain (G)
[A(:,i+1),G(i+1)] = lpc(speech1,P);
%% Pitch Detection
% Clipping the signal to Pitch Detection
speech2= clip_center(speech1);
% Moddified Autocorrelation Function is used to compute
Autocorrelation
Rn(:,i+1)= modified_autocorrelation( speech2); % All values of
Autocorrelation :Rn(k) for 0<=k<=N
```

```
a=length(round(fs/350):round(fs/80)); % length of
Fs/350<=k<=Fs/80 we take these interval
                                      % because we only need
to look in range
                                      % 80 Hz<=F0<=350Hz
                                      % (80<=F0men<=200 and
150<=F0women<=350 )
Rnk(1:a,i+1)=Rn(round(fs/350):round(fs/80),i+1); % Rn(k) for
Fs/350<=k<=Fs/80
[R(i+1),indis(i+1)]=max(Rnk(1:a,i+1)); %find max values at
80<f0<350 R=max{Rn(k)}
indis(i+1)=round(fs/350)-1+indis(i+1);%indices of max R values
R0=Rn(1,:); % Rn(0)
Rn0=R0*0.3; %if R>=30% of Rn(0) then frame is voiced and pitch
period= k+ Fs/350 else pitch period=0;
 k(i+1)=indis(i+1);
 if R(i+1)>=Rn0
       pitch(i+1)=(k(i+1)+fs/350); %frame is voiced
    else
       pitch(i+1)=0; % frame is unvoiced
    end
pitchFrequency=[pitchFrequency; pitch(i+1)]; %all pitch
frequency
G(i+1)=G(i+1).*0.5.*10^3; % increase the gain to hear the new
speech
[signal,
t]=syn_lpc(fs,pitchFrequency(i+1),frame_dur,A(:,i+1),G(i+1));
%syntehize the signal by using syn_lpc function
signalnew=[signalnew;signal'];% syntehized the signal by
collect all frames
tson=(0:length(signalnew)-1)/fs; %time duration for syntehized
the signal

end
%figure,
subplot(2,1,1)
t1=(0:length(speech)-1)/fs; %time for the original signal
plot(t1,speech) %original signal
grid on
title ('Original Signal2')
subplot(2,1,2)
plot(tson,signalnew) %synthesized Signal
grid on
title('Synthesized Signal2')
sound(signalnew,fs) %listen the synthesized Signal
```

References:

- http://www.seas.ucla.edu/~ingrid/ee213a/speech/vlad_present.pdf
- http://eecs.oregonstate.edu/education/docs/ece352/CompleteManual.pdf
- http://sail.usc.edu/~lgoldste/Ling582old/
- http://www.seas.ucla.edu/spapl/projects/ee214aW2002/1/report.html