**Submitter By: Arade Tariku  Tegen**

# Audio Signal Processing Project Report

UNIVERSITY OF TRENTO FACULTY OF ENGINEERING DEPARTMENT OF TELECOMMUNICATION ENGINEERING

**SUMBITTED TO:- MAURIZIO OMOLOGO**

**PIERGIORGIO SVAIZER**

Date: December 30, 2011

UNIVERSITY
OF TRENTO - Italy

# Assignment Report

On the assignment given we were required to perform analysis on a given speech signal (sx119.pcm speech in raw PCM format, with sampling frequency of 16 kHz, and samples represented as 16-bit integers (shorts). We will perform the analysis according to the given specifications. The analysis was done using MATLAB.

1. The first question we were asked to perform is LPC Analysis according to the following specifications:

- Window type: Hamming
- Window length (frame length): 20 milliseconds
- LPC analysis step: 5 milliseconds (i.e. 200 Hz analysis rate)
- LPC method: autocorrelation
- LPC prediction order: 16

**1a.** The first question is to organize the input signal into *N* analysis frames which can be realized in a structure of a matrix with N columns.

To start the analysis, we first imported the signal and determine the length of our signal using MATLAB. Here we put the signal on matrix '*rw*' and then length of matrix will be put to variable '*lenght*' in our case the length of our signal is 50278. Then we will initialize the LPC prediction order, Window length, LPC analysis step and Window type according to the specification given above. Here the window size can be calculated as follows (*16 KHz * 20ms=320samples)*. The analysis stapes can also be calculated in the same way (*16 KHz *ms=80samples*)

The important thing to notice in our code is, we define all our coding based on variables which are initialized at the beginning of the code. This will allow us to simply vary our input parameters with no need to go and adjust the parameters line by line in our code.

In addition our code will calculate the number of frames dynamically from our input signal. This will make our code applicable for variable type of inputs (i.e. because we don't hardcoded the frame numbers and other calculated inputs the code will automatically do this analysis based on the input parameters and the input signal)

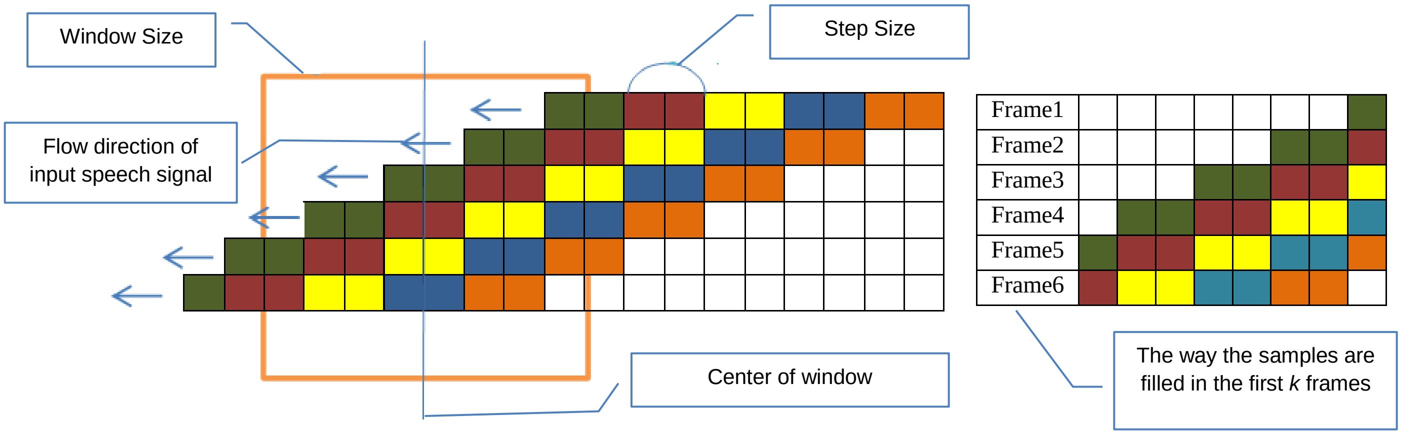
Before starting the analysis, to make our input raw signal fit to our analysis window in our last frames we will zero pad the raw data at the end until our last index is an integer multiple of our analysis window (in our case 80). Then we will update '*lenght*' to our new length.

Then based on the new length of our raw data, after zero padding at the end, we will calculate the max number of frame we are going to use in our analysis and put the value to 'max'. While calculating this length we consider both the initial and final frames which will not be fully occupied by out speech signal samples. Detail will be discussed later. Then we have declared some important matrixes which are indexed to max number of our analysis frame.

The next stapes will be putting the raw data in to our frames for analysis. This can be done assuming that we have input flow of speech signal from some source. We begin with a '*for*' loop which starts from the first frame represented by the index *k =1* and loops through to the end of the signal which is represented by *k = max*.

Here there are three cases to consider during framing of our raw speech signal. *Case 1:* when we are taking the first samples. *Case 2:* is when we are taking samples in the middle of our sample. *Case 3:* is when we are taking samples at the end of our speech signal.

We have designed three conditions for each of this cases which will enable us to zero pad the missing spots in our frame. Generally what we will do is we move the input signal starting from the first index to the last in stapes of 80samples. The only exception to this will be when taking the first frame, since we want to give emphasis for our 80 samples on our frame analysis we will initially take 40 samples, this will align the first 80 sample to the center of the window after moving two analysis stapes. This can be demonstrated using the fig 1.1 below



Window Size
Step Size
Flow direction of input speech signal
Frame1
Frame2
Frame3
Frame4
Frame5
Frame6
Center of window
The way the samples are filled in the first *k* frames

*Fig 1.1 Graphical representation of flow of our source speech signal in the window and how it is sampled*

A point to remember here is that the window cannot function properly at the beginning and end of the signal. This can demonstrate the effect of delay which is introduced during speech signal analysis in real time. In our case the valid LPC prediction coefficient is expected in the 3rd frame (i.e. 200 samples delay). This will get higher as we increase the size of our window.

1b. According to what is asked in question 1b we Derive for each frame corresponding N-column matrices Containing: The LPC predictor coefficients , The predictor gain G and The linear prediction error (residual).the step is described as follows.

The frame we obtain is multiplied with hamming window of size *320(which is our frame size)* to get windowed signals which is stored in a new matrix of the windowed signals called `afh(1:ws,k)`for the corresponding frame *k*. Next we will compute the frequency response of our frame using `fft()`command the value is stored in `afhfft(1:512,k)`.The next line use the windowed signal of each frame to determine the LPC coefficients. In that case a MATLAB function called `lpc((afh(1:ws,k)),p)` is used ,which takes two arguments, `(afh(1:ws,k))` that is the windowed signal of kth frame and the prediction order *p*. Thus, the output of the function, the LPC coefficients, are assigned to a matrix called `coff(1:(p+1),k)`which is the prediction coefficients for kth frame. After that, the autocorrelation of our kth frame signal is computed. In order to do that the MATLAB function called `xcorr(afh(1:ws,k),p)`is used which takes two arguments, the windowed signal `afh(1:ws,k)` and the maximum lag which is equivalent to the prediction order *p*. This function returns autocorrelation of the signal for lag of *–p to p*. But we are interested in the autocorrelation values from 0 lag to the p, therefore we will truncate the autocorrelation values and store it in to a new vector `trcorr(1:(p+1),k)` *p+1 to 2p+1* of our *xcorr* matrix.

The next step is computing the gain of each frame from transpose of truncated auto correlated signal and the lpc coefficients i.e. we used this variables to compute the gain as a square root of the matrix product of the `coff` matrix and the transpose of `trcorr` matrix. In the next lines the prediction errors is calculated from lpc coefficient and input framed signal by using MATLAB function of *filter()*. Finally we compute the impulse and the frequency responses for each frame. This computation will be done for all the frames and when it reach the last frame it will exit the *'for'* loop.

Finally for simplicity of checking the result, we have developed a plotting procedure you can find it at the end of the Main MATLAB procedure list. Here by simply putting the index number of the frames that we want to in the k matrix we can plot raw frame, windowed frame, prediction error, truncated correlation, frequency response of our framed speech signal with the LPC estimated frequency response, then impulse response of our LPC, and the frequency response of the truncated impulse responses compared with the frequency response of the our LPC.

1c. For this question we selected four voices and four unvoiced sample frames, and we do complete analysis to compare the LPC envelopes obtained using the parameters found in b with the short-time Fourier analysis.

For our observation, we tried to pick the voiced and unvoiced frames by examining the signal using 'sptool' in such a way that it is selected from different part of speech signal, *sx119.pcm*. The frames we selected for our analysis are presented in the table below with reference index and time:

| Voiced | | | | | |
|---|---|---|---|---|---|
| Frame index | Start index in raw data | End index in raw data | Start time(ms) | End time(ms) | Letter it correspond |
| 157th | 12281 | 12600 | 767.5 | 787.5 | misquote |
| 327th | 25881 | 26200 | 1617.5 | 1637.5 | retracted |
| 414th | 32841 | 33160 | 2052.5 | 2072.5 | with |
| 502th | 39881 | 40200 | 2492.5 | 2512.5 | apology |

| Unvoiced | | | | | |
|---|---|---|---|---|---|
| Frame index | Start index in raw data | End index in raw data | Start time(ms) | End time(ms) | Letter it correspond |
| 104th | 8041 | 8360 | 502.5 | 522.5 | misquote |
| 195th | 15321 | 15640 | 957.5 | 977.5 | misquote |
| 255th | 20121 | 20440 | 1257.5 | 1277.5 | was |
| 308th | 24361 | 24680 | 1522.4 | 1542.4 | retracted |

**Plot of windowed signal:** In figure 1.2 we have plotted the windowed signal two unvoiced and two voiced frames stated above in order to analyze their characteristics.

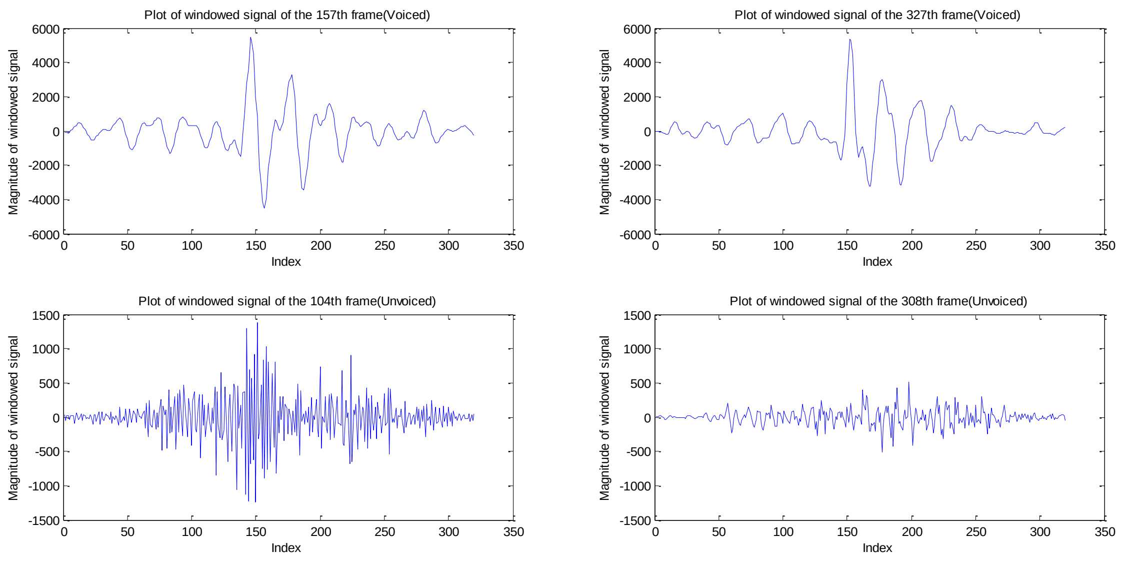


*Fig 1.2 Windowed signal of 4 selected frames in the voiced and unvoiced part of the speech*

**The LPC Envelope:** plot of Figure 1.3 correspond to the LPC envelopes of the windowed signals plotted above. For comparison, we have plotted the envelope along with the Fourier Transform of our windowed signal for selected frames above. The envelopes are plotted blue and the FFT is plotted red. The figure shows that a linear prediction model with p = 16 matches the general shape of the short-time spectrum, but does not represent all its local peaks and valleys, and this is exactly what is desired.[1]

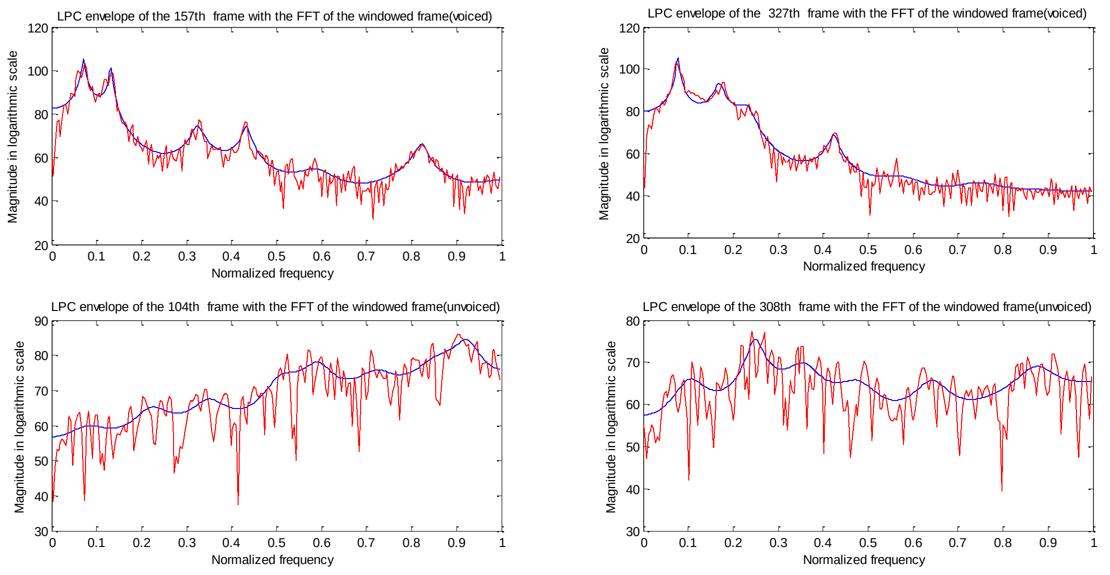


*Fig 1.3 Frequency response of the LPC envelopes for 4 selected frames in the voiced and unvoiced part of the speech*

**Explanations**: As can be seen from the above plotted graphs, the envelopes generally follow the FFT both in the *voiced* as well as the *unvoiced* part of the windowed signal. Evidently, the linear predictive spectra tend to favor the peaks of the short-time Fourier transform. But the level at which they follow is much strict for the voiced signal than the unvoiced signal. This shows that using a prediction order p= 16, we can approximates with a good tradeoff between complexity and spectral detail necessary for our analysis. We can also notice that the peaks of our envelope occurred near the average of the local peaks of the original signals FFT. This shows that the envelope approximates the wave form of our signal and can be used to predict that signal while ignoring unnecessary details. We can also see that the lpc envelop matches the signal spectrum much more closely in the regions of large signal energy than lower energies and also the magnitude of the FFT is bigger than the that of the envelop at the peaks rather than the valleys.

Looking separately on our *voiced signals* it can be shown that the spectrum will have more energy, typically, in the low frequency region, which can be explained by the high energy of our low frequency component produced by our vocal fold vibration (We will further elaborate this in the part where we speak about pitch). The spectrum will also decrease starting from low frequency and moving to higher ones. It is also good to mention here that the higher peaks stand for the formants of the voiced segment. Also it can be observed in the spectrum that the frequency components of the voiced signal repeating at regular intervals indicating the presence of harmonic structure.

For the *unvoiced signals* the original spectrum is characterized by random noise like pattern which of course is difficult for our LPC to follow. But the LPC will try to follow the spectrum by considering the average of the local peaks. We can also see that unvoiced speech signal have relatively low energy compared to voiced one. In addition we don't have the harmonic structure for the unvoiced part. Further, the spectrum will have more energy, typically, in the high frequency region. The spectrum will also have a constant or upward trend starting from low frequency and moving upwards.

**Prediction error: -** In figure 1.4 we have plotted the signal and prediction error for some of the unvoiced and voiced frames stated above in order to analyze their characteristics.

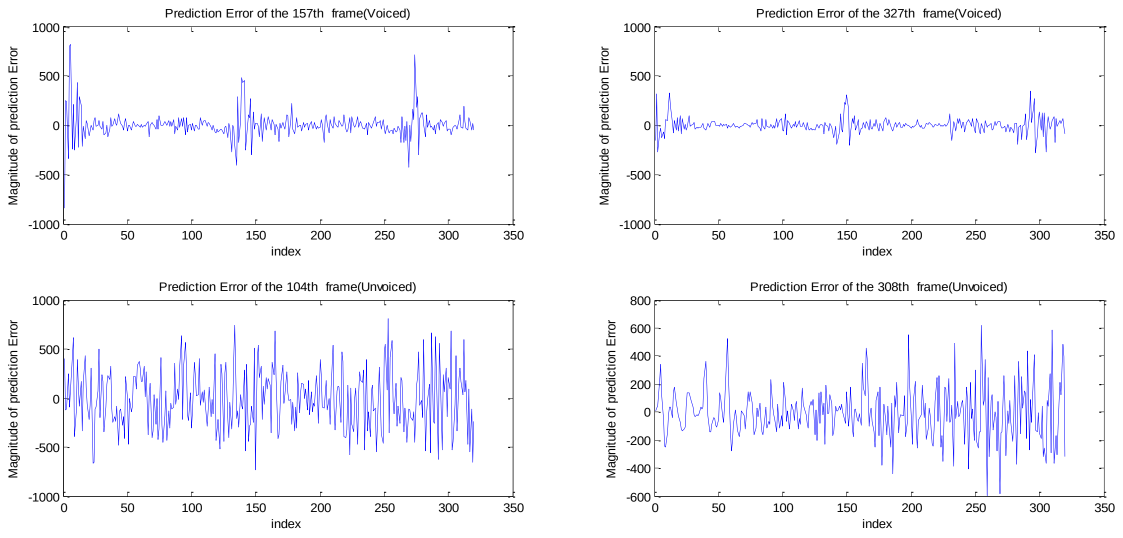


*Fig 1.4 Prediction error for voiced and unvoiced part of the speech*

**Explanations:** As we can see on Fig 1.4, generally we can see that the Prediction error exhibit a random characteristic. But if we see for the *voiced* part of our error signal we can notice the periodicity of the signal which corresponds to the picks of out original signal. This period is more commonly termed as **pitch period** i.e. one can approximate the pitch of a speech using these periodic peaks. Also we can see that the magnitude of the prediction error is much less than the original signal for the voiced one. This shows that we need fewer bits to represent the error than the original signal.

For the unvoiced part of our signal completely exhibits random noise characteristics. Also we can see that the magnitude of the prediction *error is not* significantly less than the original signal for the unvoiced one. This shows that we need higher bits to represent the error than our voiced counterpart.

**Further explanation on the formants and pitch:** We also tried to calculate the formant for the voiced frames by looking at the frequency corresponding to the two peaks of our LPC envelop and we try to map our result on the vowel triangle (for English male). We also try to cross check our result by extracting our formant using sptool. The result is presented in the following table. We also present our sptool analysis result

| Voiced | | | | | |
|---|---|---|---|---|---|
| Frame index | First formant(hz) | Second formant(hz) | VT map | End time(ms) | Letter it correspond |
| 157th | 562 | 1062 | UH | 787.5 | misquote |
| 327th | 625 | 1320 | AH | 1637.5 | retracted |
| 414th | 440 | 970 | UH | 2072.5 | with |
| 502th | 660 | 1125 | AA | 2512.5 | apology |

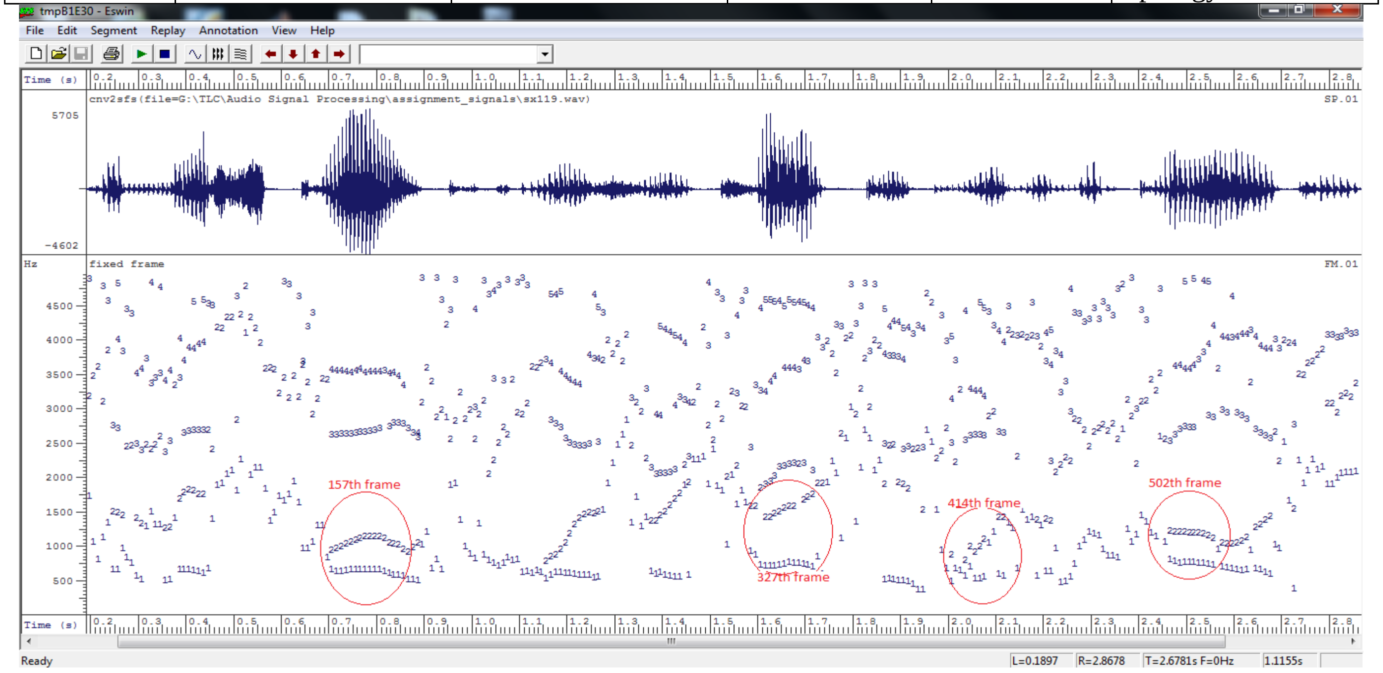


*Fig 1.5 sptool analysis of our signals formant*

We also try to calculate the pitch period of our voiced frames form the peak period of the prediction error plot. We also try to cross check our result by extracting our formant using sptool. The result is presented in the following table. We also present our sptool analysis result.

| Voiced | | | | | |
|---|---|---|---|---|---|
| Frame index | First Pick index no | Second Pick index no | Period(ms) | Frequency(hz) | Letter it correspond |
| 157th | 139 | 274 | 8.43 | 118.6 | misquote |
| 327th | 150 | 293 | 8.93 | 111.98 | retracted |
| 414th | 57 | 224 | 10.43 | 95.8 | with |
| 502th | 58 | 242 | 11.5 | 86.95 | apology |

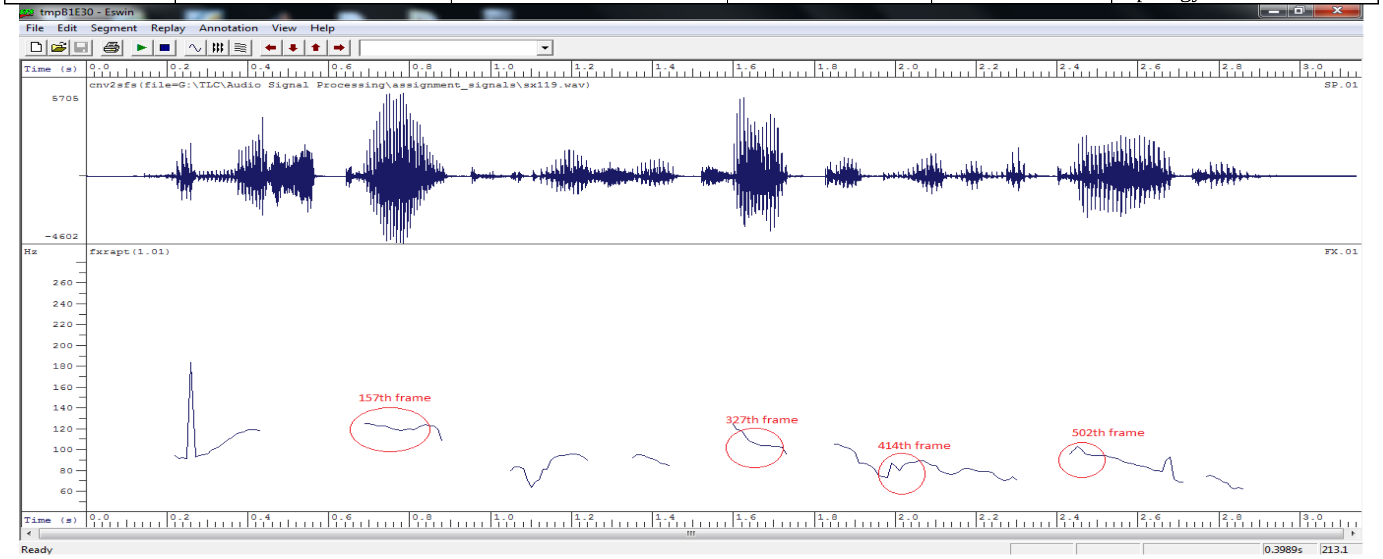

tmpB1E30 - Eswin
File Edit Segment Replay Annotation View Help
Time (s)
cnv2sfs(file=G:\TLC\Audio Signal Processing\assignment_signals\sx119.wav)
SP.01
5705
-4602
Hz
fxrapt(1.01)
FX.01
157th frame
327th frame
414th frame
502th frame
Ready
0.3989s
213.1

*Fig 1.6 sptool analysis of our signals pitch*

It can be seen that we have approximated the formant and the pitch with minor difference from what we get in the sptool analysis.

**The impulse response of our LPC all-pole model:** Figure 1.7 shows the plots of the impulse response of the predictor for voiced and unvoiced parts of the speech signal. Here, as you can see on the plots we observed that the impulse response of the voiced frames show periodicity. However, impulse responses of the unvoiced frames show noisy characteristics.

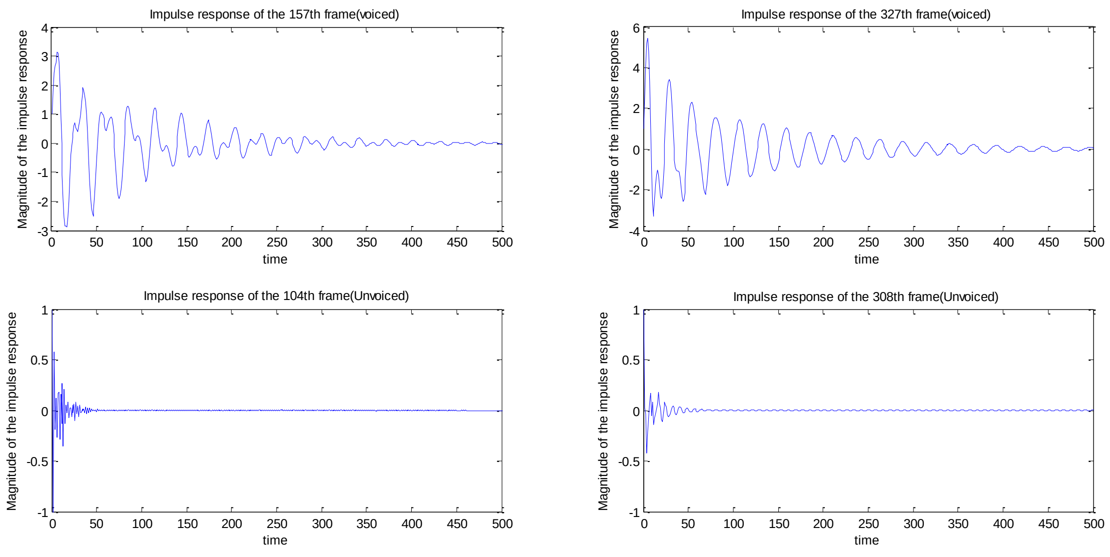


*Fig 1.7 Impulse responses for the earlier selected frames speech signal*

1d. In our third question it was asked to compute from our LPC predictor coefficients a corresponding impulse response with truncated versions of length 5 ms (*i.e. 5ms*16kz=80 sample*) of the infinite-length impulse responses of the LPC synthesis filters.

This can be simply done by MATLAB function impz() which computes the impulse response of the filter with numerator coefficients b and denominator coefficients a. Since our LPC is all pole filters our b value will be 1. We have previously calculated this value for every frame buy taking from the infinite impulse response 1000 elements and store it in the vector h with frame index k using this code `[h(1:1000,k),t(1:1000,k)]=impz(1,coff(1:(p+1),k),1000);` So what will be done next is to truncate the first 80 samples of this impulse response and put it in matrix '*IR*'. Then we drive the frequency response for this truncated signal using MATLAB function of *fft()* and put it on matrix '*HIR*' for plotting. Then we will compute '*IR2*' and '*IR3*' using the same procedure but having larger window size(for *IR2*) and also using hamming window to truncate our impulse response(for *IR3*), the reason for this will be discussed later.

We will also find the frequency response for '*IR2'* and '*IR3'* using *fft()* and put it in matrix *'HIR2'and ' HIR3'*. We will plot our LPC envelop and the three frequency response we just calculated on the same plot and compare our result. We will do this analysis on one voiced and one unvoiced frame (`327th and 104th`). The Code to implement this is given at the end of this report.

Fig 1.7 is the plot of the frequency response of LPC predictor filter in *red*, our 80sample truncated impulse response HIR2 in *green,* our 320 truncated in *black* and our 320 truncated with hamming window in *blue*.

We can see that for the unvoiced signal both our truncated impulse responses have perfectly approximated our IIR LPC predictor. But when we see the case of the voiced signal we can see that our 80 sample truncated FIR filters was not able to simulate the IIR filter fully. The 320 sample truncated signal do follow the frequency response of LPC with much better approximation but we still see some fluctuation its frequency response. For the last case we can see that when we use hamming window the overall gain will be reduced because of suppressed energies of our impulse response on the borders, but we can see that a smooth curvature of our formants . But we have to also notice the wide-ranging of in the lower part which is typical characteristic of these windows. Here during our analysis we have seen that generally we get good approximation of our IIR filter if we use more a window size of more than 200. But for our demonstration we choose 320.

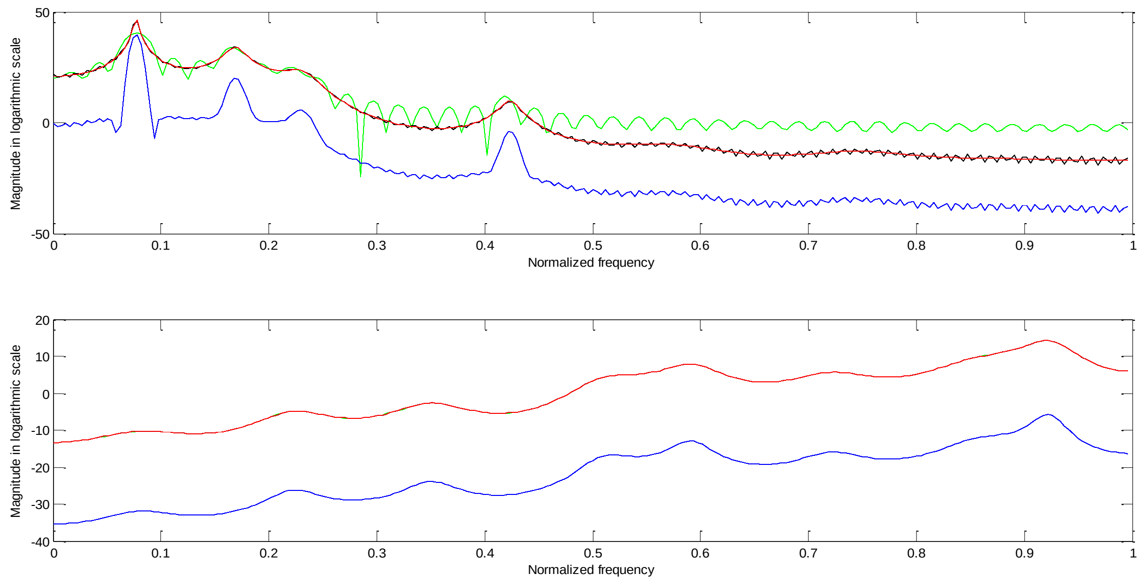


*Fig 1.8 plot of the frequency response of LPC predictor filter and frequency response of truncated impulse responses*

**Explanations:** What we are trying to do here is that to approximate an FIR filter from its IIR counterpart by windowing. So we put our knowledge of effect window size and window type in approximating the FIR filter from IIR one .It states long window size is desirable in order to have a narrow main lobe which allows one to obtain an accurate reproduction of the desired response. It is known that increasing window size will make us better approximate our IIR filter but this comes with the increased complexity and increased delay of higher order FIR filter(By delay it means our group delay will increase as we increase our order of FIR filter ). So what we try to do is to get optimal window size which will give fair approximate of our IIR LPC prediction filter and also minimize the complexity and delay of our FIR filter, which in our case is 320sample. In addition we might also consider using other type of windows instead of rectangle because in the case of rectangular window, the main lobe and the side lobe peaks grow with increasing N. Using the other special windows (Hamming , Blackman) the height of the side lobes can be diminished but with the cost of increased band in the frequency domain.

2.a The second question is calculating LPC coefficients using different windows types and repeat the analysis in 1a. and 1b. The windows are Hanning, Rectangular, Blackman and Bartlett. To implement the above question with MATLAB we simply make additional '*for*' loop which will do the whole lpc analysis computation of each frames five times i.e. for each window .During each of our five loops we store the our calculated gain in matrix `G=zeros(5,max)`. Finally we will plot the resulting Gain of each window. We can easily manipulate our code with this simple modification. The whole code is given at the end of this report.

2b. In this part of the question we were asked to Plot the gain we calculate in the previous question on a dB scale and compare the predictor gain G obtained in the various cases of window types along the whole utterance. In order to do this we will call our gain matrix index by '*j*' (`G(j,:)`)and plot them on the same plane for comparison. The code to plot our gain of different window together is given at the back of the report.

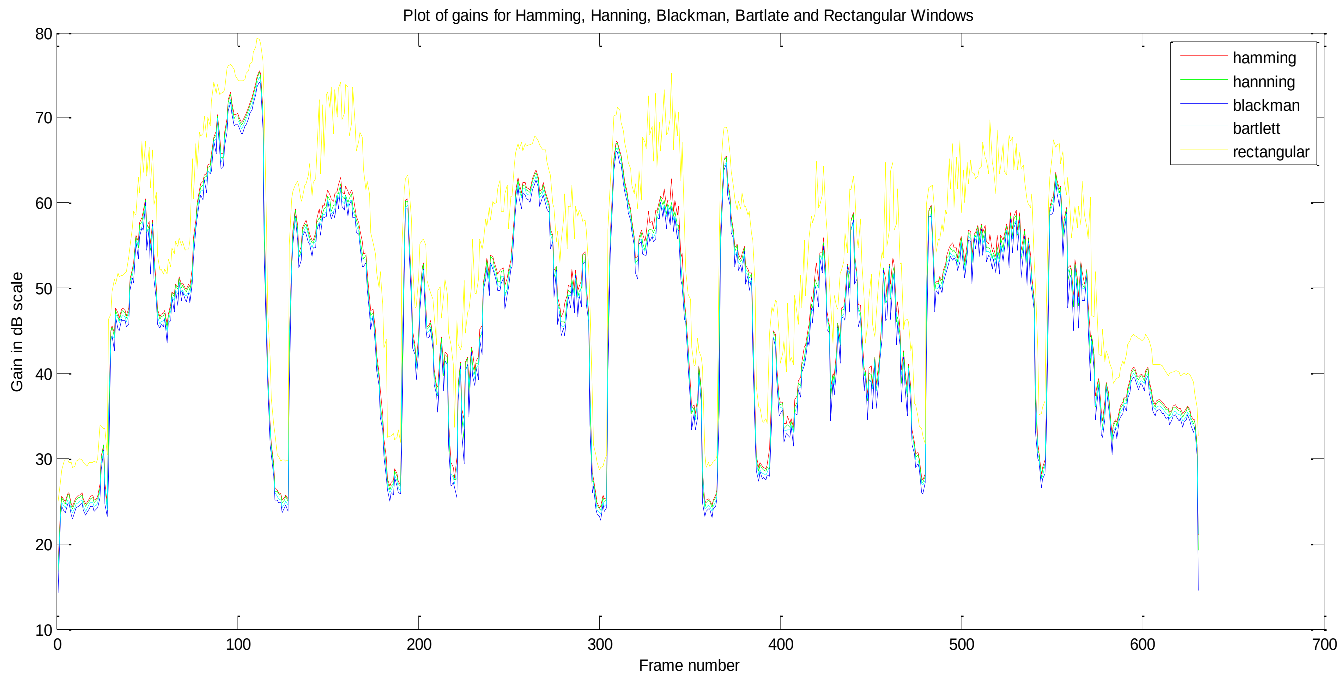


*Fig 1.9 plot of gain of LPC predictor filter using different type of window*

**Explanation:** The plot of gain for rectangular window in yellow line is the most jagged and the highest than all the other windows, this is because Rectangular window gives maximum sharpness but large side-lobes (ripples) .on the other hand the gain is much smoother and low for Blackman in blue line, this is because this window blurs in frequency but produces much less leakage, this properties is reflected on all the other windows.

It must be noted here that rectangular window exhibits the Gibbs phenomenon (i.e. the ripple of 8.9% near the discontinuities, corresponds to 1dB in the pass-band and 20 dB in the stop-band) in its frequency response. Using the other windows the height of the side lobes can be diminished, while the main lobe width increases; hence, one obtains a wider (less steep) transition at the discontinuity. This is a very important characteristic because since we are framing these signals for our LPC filter analysis we have to make sure that our lpc filter has a stable frequency response. This can be demonstrated by our plot of lpc for this different kinds of windows.

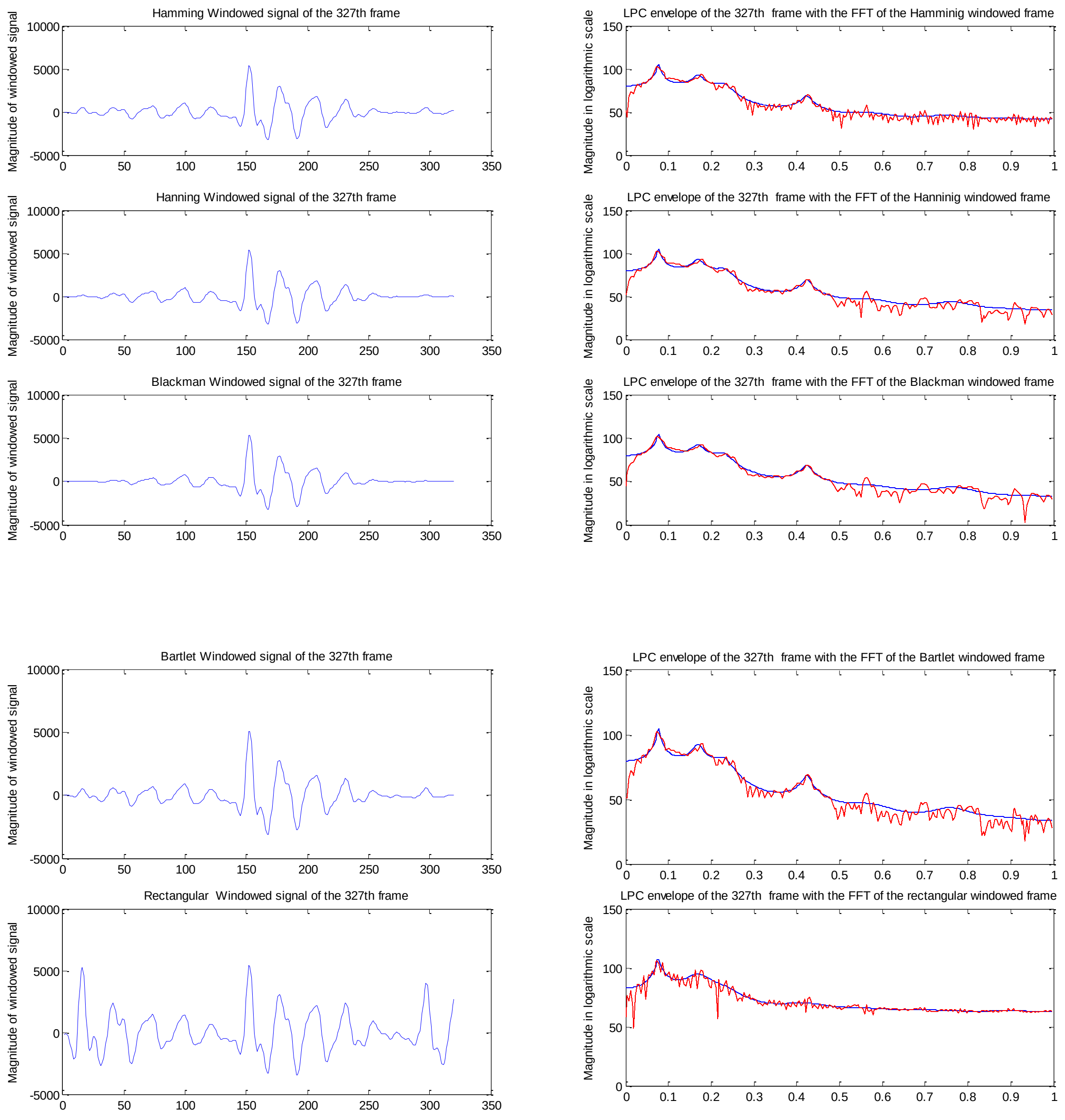


*Fig 1.10 plot of Frequency response of the LPC envelopes with their respective windowed signals of the five windows*

3.a The third question is calculating LPC coefficients using different windows size and repeat the analysis in 1a. and 1b. The windows sizes given are *10, 30, 40, 60 and 80* milliseconds. Converting this to number of samples yields (*10ms * 16 KHz* = `160`), (*30ms * 16 KHz* = `480`), (*40ms * 16 KHz* = `640`), (*60ms * 16 KHz* = `960`) and (*80ms * 16 KHz* = `1280`)

To implement the above question with MATLAB we simply make additional for loop which will do the whole LPC analysis computation of each frame six times for each window size like we did in question number two. The difference here is that here we have to do all the matrix size initialization during every loop this is because we are changing our frame size which will affect most of the matrix size during our computation. In other word the varying size of window will affect the computation of frames and every matrix related to this frame must be dynamically adjusted.

Putting this in mind when we did our first MATLAB coding we define every element based on variables which adjust their numbers dynamically based on the situation at hand. Therefor our code will take values which are defined in the initial conditions (frame size analysis step and window type). This will make our program flexible for any kind of input parameters and input signal.

During each of our six loops we will initialize our variables depending on the window size at hand. Then the second '*for'* loop will compute the lpc coefficients and gains as usual and store the calculated gain in 6x (total no of frames) indexed matrix. Finally we will plot the resulting Gain of each window. Because the code is quite similar to the first one we will not explain the details here. we will try to point out only the points we make changes.

We define our first '*for'* loop just after reading the input signal. Then all the other values defined are based on the specific value of the window size which is indexed by '*ws(j)'* . So every time the loop will redefine the whole parameters of our analysis calculation except our gain and some other variable which are not dependent on the size of our window. This parameters are only defied on the first loop and protected from redefining by *'if'* condition. The full code is given at the end of the document.

3b. In this part of the question we were asked to Plot the gains we calculated above on a dB scale and compare the predictor gain G obtained in the various cases of window size along the whole utterance. In order to do this we will call our gain matrix by index and plot them on the same plane for comparison. The code to plot our gain of different window together is given at the end of the report.

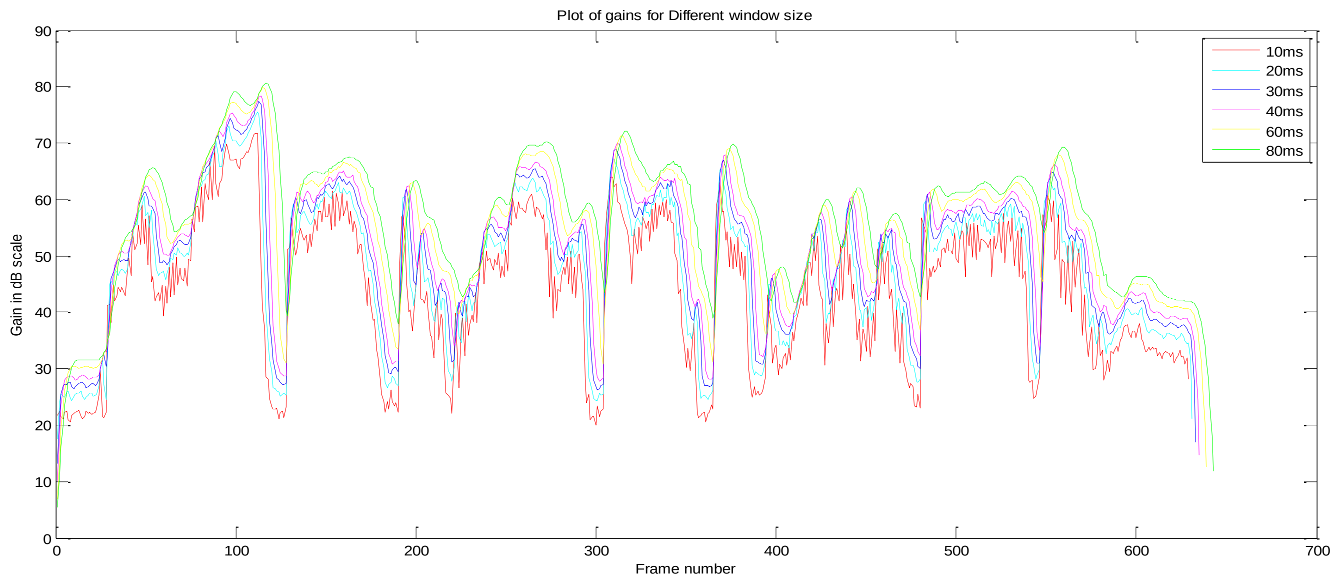


*Fig 1.9 plot of gain of LPC predictor filter using different size of window*

**Explanation:** Before talking about our observations lets point out some important points about window size and its impact on STFT(short time Fourier transform). Because of the slowly varying nature of the speech signal, it is common to process speech in frames over which the properties of the speech waveform can be assumed to remain relatively constant or better quasi-stationary. [1]So we have to choose our window size in order to maintain this quasi-stationary nature. In addition to this we must also consider using larger window size will introduce delay in our speech signal analysis.

On the other hand we have inverse relationship between window size in time domain and resolution in the frequency dimension. As a result of the short analysis window, each individual pitch period is resolved in the time dimension, but the resolution in the frequency dimension is poor .Conversely, when the window length is long we have a good frequency resolution and poor time resolution. [1] One also has to consider the average pitch period when setting window size. I.e. we must have enough data points to more accurately capture the pitch and its harmonics. So it is important to point out here that, we must decide length of our window by considering the tradeoff between all these facts.

Coming back to our observation  the green line of gain was computed with a window length of 1280 samples, corresponding to 80 ms time duration(which is the highest window length given). This window length is on the order of several speech periods of the waveform. As a result, the gain no longer displays grooves since several periods are included in the window no matter where it is placed on the waveform in the vicinity of the analysis time. As a result, the gain is not as sensitive to rapid time variations, but the resolution in the frequency dimension is much better.

The red line of gain was computed with a window length of 160 samples, corresponding to 10ms time duration. Here we can see that the gain is the grooviest of all, plus with lowest magnitude compared to all the others. The same explanation applies here, when we have smaller window size the gain is very sensitive to rapid time variations but the resolution in the frequency dimension is very low.

We also can see that the plot of gain have basically the same shape but a clear delay is observed when we go from small window size to larger window size. This can be explained as follows; since we have the same analysis step it will take more time for larger windows to calculate the corresponding gain for a portion of our signal. I.e. if we consider 80 sample of input speech signal it will only take 1 analysis step for our 10ms window to reach the center, but in case of the 80ms window it will take 8 analysis steps. This will clearly explain the delay we see in our plot of gain for different window sizes.

List of MATLAB  Procedure for question 1

```
clear all;
clc;
fid=fopen('sx119.pcm','r');
[rw,lenght]=fread(fid,inf,'short');
fclose(fid);
p=16;                                 % Initialization of order of LPC
ws=320;                               %Initialization of window size of analysis frame. Possible values
160, 320, 480, 640, 960, 1280
as=80;                                % Initialization of analysis step
window=hamming(ws);                   % possible options: hamming(ws),hann(ws),blackman(ws),bartlett(ws),rectwin(ws)
rw(lenght+1:1:ceil(lenght/as)*as)=0;  %Zero padding in order to have the last analysis frame
lenght=length(rw);                    % Here we calculate the length of our speech signal after padding
i=ceil(lenght/as);
max=ceil(lenght/as)+ceil(ws/as)-2;    %we find the number of our analysis frames
af=zeros(ws,max);                     % Initialization of matrix for each analysis frames
afh=zeros(ws,max);                    % Initialization of matrix for each analysis frames after windowing
err=zeros(ws,max);                    % Initialization of matrix for each linear prediction error
coff=zeros(p,max);                    % Initialization of matrix for LPC coefficients of each frame
corr=zeros((2*p+1),max);              % Initialization of matrix for signal after correlation
trcorr=zeros(p,max);                  % Initialization of matrix for correlated signal after truncation
G=zeros(1,max);                       % Initialization of matrix for gain of each  analysis frame
h=zeros(1000,max);                    % Initialization of matrix for impulse response of LPC of each  analysis frame
t=zeros(1000,max);                    % Initialization of matrix for time
H=zeros(256,max);                     % Initialization of matrix for frequency response of LPC of analysis frame
w=zeros(256,max);                     % Initialization of matrix for omega
afhfft=zeros(512,max);                % Initialization of matrix for FFT of each analysis frame
for k=1:max
     if (((k*as)/ws)<1)
        af((ws-(as*k)-(as/2)+1):ws,k)=rw(1:(as*k)+(as/2)) ;
     end;
    if (((k*as)/ws)>=1 && k<i)
        af(1:ws,k)=rw(((as*k)+(as/2))-(ws-1):((as*k)+(as/2)));
    end;
    if (k>=i)
        af(1:ws-(((k-i)*as)+(as/2)),k)=rw(((as*k)+(as/2))-(ws-1):lenght);
    end;
        afh(1:ws,k)=af(1:ws,k).*window;
        afhfft(1:512,k)=fft(afh(1:ws,k),512);
        coff(1:(p+1),k)=lpc((afh(1:ws,k)),p);
        corr(1:(2*p+1),k)=xcorr(afh(1:ws,k),p);
        trcorr(1:(p+1),k)=corr(p+1:2*p+1,k);
        G(1,k)=sqrt(trcorr(1:(p+1),k)'*coff(1:(p+1),k));
        err(1:ws,k)=filter(coff(1:(p+1),k),1,af(1:ws,k));
        [h(1:1000,k),t(1:1000,k)]=impz(1,coff(1:(p+1),k),1000);
        [H(1:256,k),w(1:256,k)]=freqz(G(1,k),coff(1:(p+1),k), 256);
end;
```

```
%Windowed signal of 4 selected frames in the voiced and unvoiced part of the
%speech
k=[157 327 104 308];
figure;
subplot(2,2,1);
    plot(afh(1:ws,k(1)));
subplot(2,2,2);
    plot(afh(1:ws,k(2)));
subplot(2,2,3);
    plot(afh(1:ws,k(3)));
subplot(2,2,4);
    plot(afh(1:ws,k(4)));
```

```
%Frequency response of the LPC envelopes for 4 selected frames in the
%voiced and unvoiced part of the speech
figure;
k=[157 327 104 308];
subplot(2,2,1);
    plot(w/pi,20*log10(abs(H(1:256,k(1)))),'b');
    hold on;
    plot(w/pi,20*log10(abs(afhfft(1:256,k(1)))),'r');
    hold off;
subplot(2,2,2);
    plot(w/pi,20*log10(abs(H(1:256,k(2)))),'b');
    hold on;
    plot(w/pi,20*log10(abs(afhfft(1:256,k(2)))),'r');
    hold off;
subplot(2,2,3);
    plot(w/pi,20*log10(abs(H(1:256,k(3)))),'b');
    hold on;
    plot(w/pi,20*log10(abs(afhfft(1:256,k(3)))),'r');
    hold off;
subplot(2,2,4);
    plot(w/pi,20*log10(abs(H(1:256,k(4)))),'b');
    hold on;
    plot(w/pi,20*log10(abs(afhfft(1:256,k(4)))),'r');
    hold off;
```

```
%Impulse responses for the earlier selected frames speech signal
k=[157 327 104 308];
figure;
subplot(2,2,1);
    plot(h(1:1000,k(1)));
subplot(2,2,2);
    plot(h(1:1000,k(2)));
subplot(2,2,3);
    plot(h(1:1000,k(3)));
subplot(2,2,4);
    plot(h(1:1000,k(4)));
    k=[157 327 104 308];

%Prediction error for voiced and unvoiced part of the speech
k=[157 327 104 308];
figure;
subplot(2,2,1);
    plot(err(1:ws,k(1)));
subplot(2,2,2);
    plot(err(1:ws,k(2)));
subplot(2,2,3);
    plot(err(1:ws,k(3)));
subplot(2,2,4);
    plot(err(1:ws,k(4)));
```

```
%plot of the frequency response of LPC predictor filter and frequency
%response of truncated impulse responses
k=[327 104];
for i=1:length(k)
      IR=h(1:80,k(i));
      IR2=h(1:320,k(i));
      IR3=h(1:320,k(i)).*hamming(320);
      HIR=fft(IR,512);
      HIR2=fft(IR2,512);
      HIR3=fft(IR3,512);

      subplot(2,1,i);
      plot(w/pi,20*log10(HIR(1:256)),'g');
       hold on;
      plot(w/pi,20*log10(HIR2(1:256)),'k');
       hold on;
      plot(w/pi,20*log10(HIR3(1:256)),'b');
       hold on
      plot(w/pi,20*log10(abs(H(1:256,k(i)))/G(1,k(i))),'r');
       hold on;
end
 hold off;
```

```
% Code to check characteristic of any frame
 k=[628 629 630 631];
for i=1:length(k)
    figure;
subplot(3,2,1);
    plot(af(1:ws,k(i)));
subplot(3,2,2);
    plot(afh(1:ws,k(i)));
subplot(3,2,3);
    plot(err(1:ws,k(i)));
subplot(3,2,4);
    plot(corr(1:2*p+1,k(i)));
subplot(3,2,5);
    plot(w/pi,20*log10(abs(H(1:256,k(i)))),'b');
    hold on;
    plot(w/pi,20*log10(abs(afhfft(1:256,k(i)))),'r');
    hold off;
subplot(3,2,6);
    plot(h(1:1000,k(i)));

figure;
      Hh=fft(h(1:80,k(i)),512);
      Hhh=fft(h(1:160,k(i)),512);
```

```
     plot(w/pi,20*log10(Hhh(1:256)),'b');
      hold on;
     plot(w/pi,20*log10(Hh(1:256)),'g');
      hold on;
     plot(w/pi,20*log10(abs(H(1:256,k(i)))/G(1,k(i))),'r');
      hold off;
end;
```

## List of MATLAB Procedure for question 2

```
clear all;
clc;
fid=fopen('sx119.pcm','r');
[rw,lenght]=fread(fid,inf,'short');
fclose(fid);
p=16;                                  % Initialization of order of LPC
ws=320;                                %Initialization of window size of analysis frame. Possible values 160, 320,
480, 640, 960, 1280
as=80;                                 % Initialization of analysis step
window(1:ws,1)=hamming(ws);
window(1:ws,2)=hann(ws);
window(1:ws,3)=blackman(ws);
window(1:ws,4)=bartlett(ws);
window(1:ws,5)=rectwin(ws);            % possible options: hamming(ws),hann(ws),blackman(ws),bartlett(ws),rectwin(ws)
rw(lenght+1:1:ceil(lenght/as)*as)=0;   %Zero padding in order to have the last analysis frame
lenght=length(rw);
i=ceil(lenght/as);
max=ceil(lenght/as)+ceil(ws/as)-1;     %we find the number of our analysis frames
af=zeros(ws,max);                      % Initialization of matrix for each analysis frames
afh=zeros(ws,max);                     % Initialization of matrix for each analysis frames after windowing
err=zeros(ws,max);                     % Initialization of matrix for each linear prediction error
coff=zeros(p,max);                     % Initialization of matrix for LPC coefficients of each frame
corr=zeros((2*p+1),max);               % Initialization of matrix for signal after correlation
trcorr=zeros(p,max);                   % Initialization of matrix for correlated signal after truncation
G=zeros(5,max);                        % Initialization of matrix for gain of each  analysis frame
h=zeros(1000,max);                     % Initialization of matrix for impulse response of LPC of each  analysis frame
t=zeros(1000,max);                     % Initialization of matrix for time
H=zeros(256,max);                      % Initialization of matrix for frequency response of LPC of analysis frame
w=zeros(256,max);                      % Initialization of matrix for omega
afhfft=zeros(512,max);                 % Initialization of matrix for FFT of each analysis frame
for j=1:5
for k=1:max
     if (((k*as)/ws)<1)
        af((ws-(as*k)-(as/2)+1):ws,k)=rw(1:(as*k)+(as/2)) ;
     end;
    if (((k*as)/ws)>=1 && k<i)
        af(1:ws,k)=rw(((as*k)+(as/2))-(ws-1):((as*k)+(as/2)));
    end;
    if (k>=i)
        af(1:ws-(((k-i)*as)+(as/2)),k)=rw(((as*k)+(as/2))-(ws-1):lenght);
    end;
        afh(1:ws,k)=af(1:ws,k).*window(1:ws,j);
        afhfft(1:512,k)=fft(afh(1:ws,k),512);
        coff(1:(p+1),k)=lpc((afh(1:ws,k)),p);
        corr(1:(2*p+1),k)=xcorr(afh(1:ws,k),p);
        trcorr(1:(p+1),k)=corr(p+1:2*p+1,k);
        G(j,k)=sqrt(trcorr(1:(p+1),k)'*coff(1:(p+1),k));
        err(1:ws,k)=filter(coff(1:(p+1),k),1,af(1:ws,k));
        [h(1:1000,k),t(1:1000,k)]=impz(1,coff(1:(p+1),k),1000);
        [H(1:256,k),w(1:256,k)]=freqz(G(j,k),coff(1:(p+1),k), 256);
end
figure;
 u=[327];
for a=1:length(u)
    subplot(2,1,1);
    plot(afh(1:ws,u(a)));
    subplot(2,1,2);
    plot(w/pi,20*log10(abs(H(1:256,u(a)))),'b');
    hold on;
    plot(w/pi,20*log10(abs(afhfft(1:256,u(a)))),'r');
    hold off;

end;
end
figure;
     plot(20*log10(abs(G(1,:))),'r');%hamming
     hold on;
     plot(20*log10(abs(G(2,:))),'g');%hann
     hold on;
     plot(20*log10(abs(G(3,:))),'b');%blackman
     hold on;
     plot(20*log10(abs(G(4,:))),'c');%bartlett
     hold on;
     plot(20*log10(abs(G(5,:))),'y');%rectwin
     hold off;
```

## List of MATLAB Procedure for question 3

```
clear all;
clc;
fid=fopen('sx119.pcm','r');
[rw,lenght]=fread(fid,inf,'short');
fclose(fid);
for j=1:6
ws=[160 320 480 640 960 1280];             %Initialization of window size of analysis frame. Possible values 160, 320, 
480, 640, 960, 1280
as=80;                                     % Initialization of analysis step
window=hamming(ws(j));                     % possible options:hann(ws),blackman(ws),bartlett(ws),rectwin(ws)
max=ceil(lenght/as)+ceil(ws(j)/as)-1;      %we find the number of our analysis frames
af=zeros(ws(j),max);                       % Initialization of matrix for each analysis frames
afh=zeros(ws(j),max);                      % Initialization of matrix for each analysis frames after windowing
err=zeros(ws(j),max);                      % Initialization of matrix for each linear prediction error
if (j==1)
p=16;                                      % Initialization of order of LPC
rw(lenght+1:1:ceil(lenght/as)*as)=0;       %Zero padding in order to have the last analysis frame
lenght=length(rw);
i=ceil(lenght/as);
coff=zeros(p,max);                         % Initialization of matrix for LPC coefficients of each frame
corr=zeros((2*p+1),max);                   % Initialization of matrix for signal after correlation
trcorr=zeros(p,max);                       % Initialization of matrix for correlated signal after truncation
G=zeros(6,max);                            % Initialization of matrix for gain of each  analysis frame
h=zeros(1000,max);                         % Initialization of matrix for impulse response of LPC of each  analysis 
frame
t=zeros(1000,max);                         % Initialization of matrix for time
H=zeros(256,max);                          % Initialization of matrix for frequency response of LPC of analysis frame
w=zeros(256,max);                          % Initialization of matrix for omega
afhfft=zeros(512,max);                     % Initialization of matrix for FFT of each analysis frame
end
for k=1:max
     if (((k*as)/ws(j))<1)
        af((ws(j)-(as*k)-(as/2)+1):ws(j),k)=rw(1:(as*k)+(as/2)) ;
     end;
    if (((k*as)/ws(j))>=1 && k<i)
        af(1:ws(j),k)=rw(((as*k)+(as/2))-(ws(j)-1):((as*k)+(as/2)));
    end;
    if (k>=i)
        af(1:ws(j)-(((k-i)*as)+(as/2)),k)=rw(((as*k)+(as/2))-(ws(j)-1):lenght);
    end;
        afh(1:ws(j),k)=af(1:ws(j),k).*window;
        afhfft(1:512,k)=fft(afh(1:ws(j),k),512);
        coff(1:(p+1),k)=lpc((afh(1:ws(j),k)),p);
        corr(1:(2*p+1),k)=xcorr(afh(1:ws(j),k),p);
        trcorr(1:(p+1),k)=corr(p+1:2*p+1,k);
        G(j,k)=sqrt(trcorr(1:(p+1),k)'*coff(1:(p+1),k));
        err(1:ws(j),k)=filter(coff(1:(p+1),k),1,af(1:ws(j),k));
        [h(1:1000,k),t(1:1000,k)]=impz(1,coff(1:(p+1),k),1000);
        [H(1:256,k),w(1:256,k)]=freqz(G(j,k),coff(1:(p+1),k), 256);
end
end
figure;
    plot(20*log10(abs(G(1,:))),'r');%160
     hold on
     plot(20*log10(abs(G(2,:))),'c');% 320
     hold on
     plot(20*log10(abs(G(3,:))),'b');% 480
     hold on
     plot(20*log10(abs(G(4,:))),'m');% 640
     hold on
     plot(20*log10(abs(G(5,:))),'y');% 960
     hold on
     plot(20*log10(abs(G(6,:))),'g');% 1280
     hold off
```

## REFERENCES

[1] L. Rabiner and R. Shafer, *Digital Processing of Speech Signals*, Englewood Cliffs: Prentice Hall, 1978.