EE 519 Homework2

Samwoo Seong

Problem 4.

(a) sol)

For a certain function, to be an even function, following relationship is required.

$$f(x) = f(-x)$$

Let $R_x[m] = \sum_{n=-\infty}^{\infty} x[n]x[n+m] \quad \cdots ①$

Then, $R_x[-m] = \sum_{n=-\infty}^{\infty} x[n]x[n-m] \quad \cdots ②$

right hand side of ②

$$\sum_{n=-\infty}^{\infty} x[n]x[n-m]$$

Let $\Rightarrow n-m = t$

$$\Rightarrow \sum_{t=-\infty}^{\infty} x[t+m]x[t]$$

Let $t = n$

$$\Rightarrow \sum_{n=-\infty}^{\infty} x[n+m]x[n]$$

$= ①$'s right hand side $= R_x[m]$

Therefore, $R_x[-m] = R_x[m]$

$\Rightarrow R_x[m]$ is an even function

Q.E.D.

(b) $R_e[m] = \sum_{n=-\infty}^{\infty} e[n]\,e[n+m]$

Let $a_k = a[k]$

$$= \sum_{n=-\infty}^{\infty} \left(\sum_{k=0}^{p} a[k]\,x[n-k] \right) \cdot e[n+m]$$

Let $a[k] = \begin{cases} 0 & k<0 \text{ or } k>p \\ a_k & \text{otherwise} \end{cases}$

$$= \sum_{n=-\infty}^{\infty} \left\{ \left(\sum_{k=-\infty}^{\infty} a[k]\,x[n-k] \right) \cdot e[n+m] \right\}$$

Switch order of summations

$$= \sum_{k=-\infty}^{\infty} a[k] \sum_{n=-\infty}^{\infty} x[n-k] \cdot e[n+m]$$

$$= \sum_{k=-\infty}^{\infty} a[k] \sum_{n=-\infty}^{\infty} x[n-k] \cdot \sum_{l=-\infty}^{\infty} a[l]\,x[n+m-l]$$

$$= \sum_{k=-\infty}^{\infty} a[k] \sum_{l=-\infty}^{\infty} a[l] \sum_{n=-\infty}^{\infty} x[n-k]\,x[n+m-l]$$

Let $n-k = n'$

$$= \sum_{k=-\infty}^{\infty} a[k] \sum_{l=-\infty}^{\infty} a[l] \sum_{n'=-\infty}^{\infty} x[n']\,x[n'+k+m-l]$$

Let $k-l = -l'$

$$= \sum_{k=-\infty}^{\infty} a[k] \sum_{l'=-\infty}^{\infty} a[k+l'] \sum_{n'=-\infty}^{\infty} x[n']\,x[n'+m-l']$$

Let $k = n$, $l' = l$

$$= \sum_{l=-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} a[n]\,a[n+l] \right) \left(\sum_{n'=-\infty}^{\infty} x[n']\,x[n'+m-l] \right)$$

Let $n' = n$

$$= \sum_{l=-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} a[n]\,a[n+m] \right) \left(\sum_{n=-\infty}^{\infty} x[n']\,x[n'+m-l] \right)$$

$$= \sum_{l=-\infty}^{\infty} R_a[l] \cdot R_x[m-l]$$

Q.E.D.

(c) $F_s = 16000$ Hz

$\Rightarrow$ Interval = 2.5 - 0.7 = 1.8 sec

$\Rightarrow$ # of samples = $1.8 \times F_s$

= 28,800

let's say we have p order

$\Rightarrow a_k \Rightarrow p+1$

length of $x[n] = 28,800 = N_1$

$\Rightarrow$ Length of $R_x = 2N_1$

length of $R_a = 2(p+1)$

$\Rightarrow$ Length of $R_e[n] = 2N_1 + 2(p+1)$

= # of multiplication

$\therefore$ we need to multiply

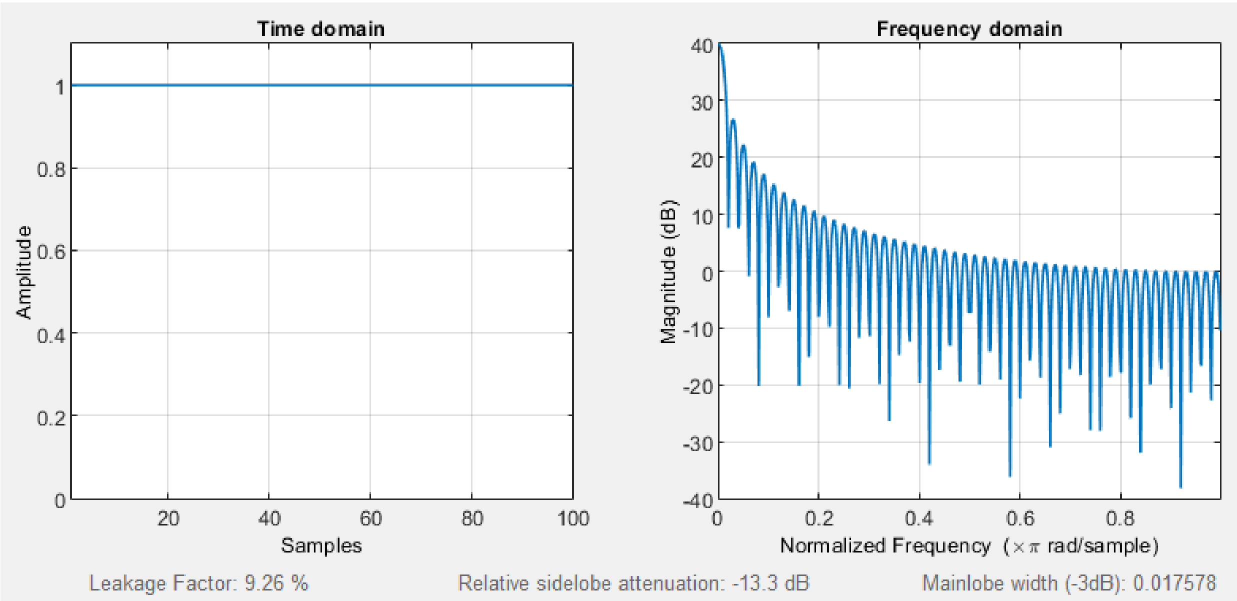
$2N_1 + 2(p+1)$ times.

problem 1,2,3 are in my report.

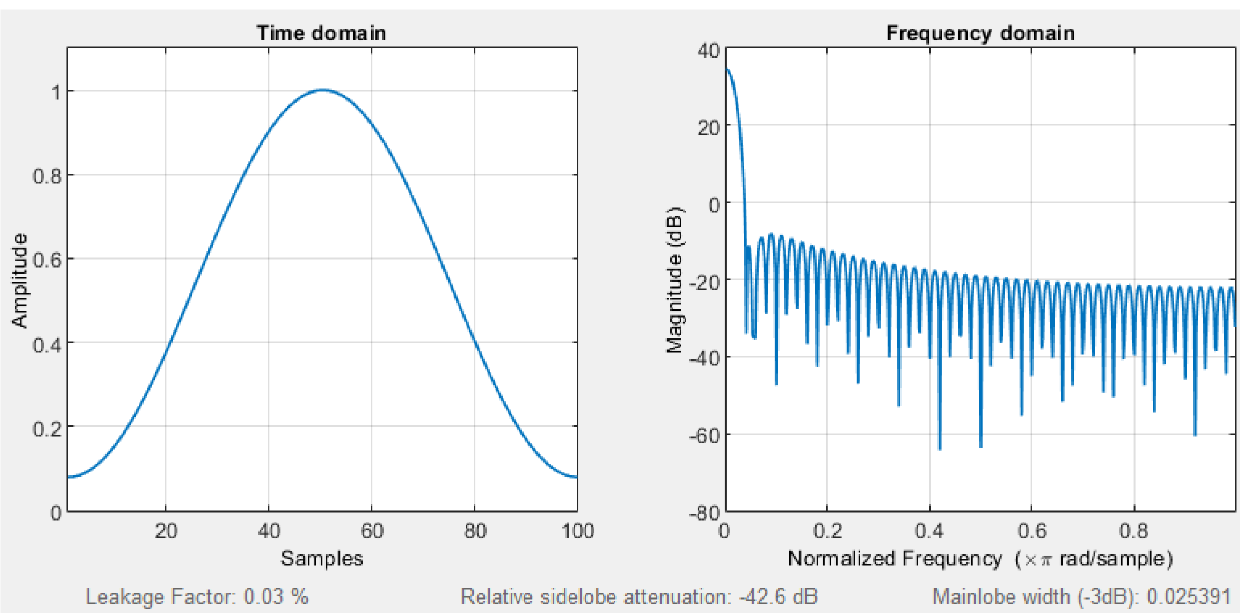
Problem1

(a)

(1) Common figures in the time and frequency domain for rectangular window



(2) Common figures in the time and frequency domain for Hamming window



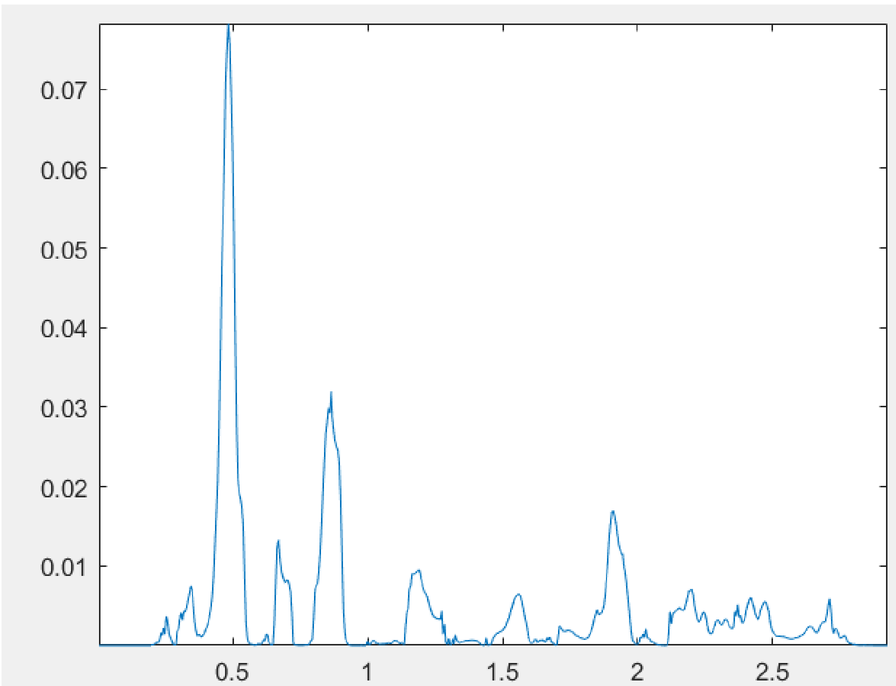
Answer: If we look at main lobe width closely, we can realize the main lobe for the Hamming window is wider. On the other hand, the main lobe for the rectangular window is narrow and sharp. In terms of side lobe, one from the rectangular window has much higher magnitude compared to Hamming window one. These characteristics of main lobe and side lobe have trade off relationship meaning, for example, if main lobe is sharp, then magnitude of side lobe is relatively big.

Also, we need to consider which window will be used depending on applications. For instance, in frequency estimation of unknown signal, we should select a window carefully because resulting frequencies could be different based on the chosen window.

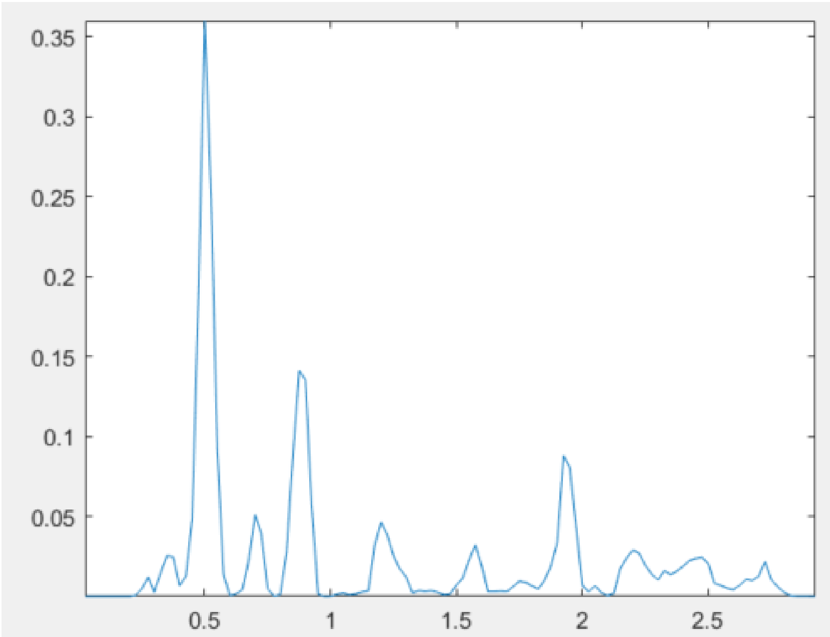
Ideally, we want a window that has sharp and narrow main lobe and flat side lobe.

(b)

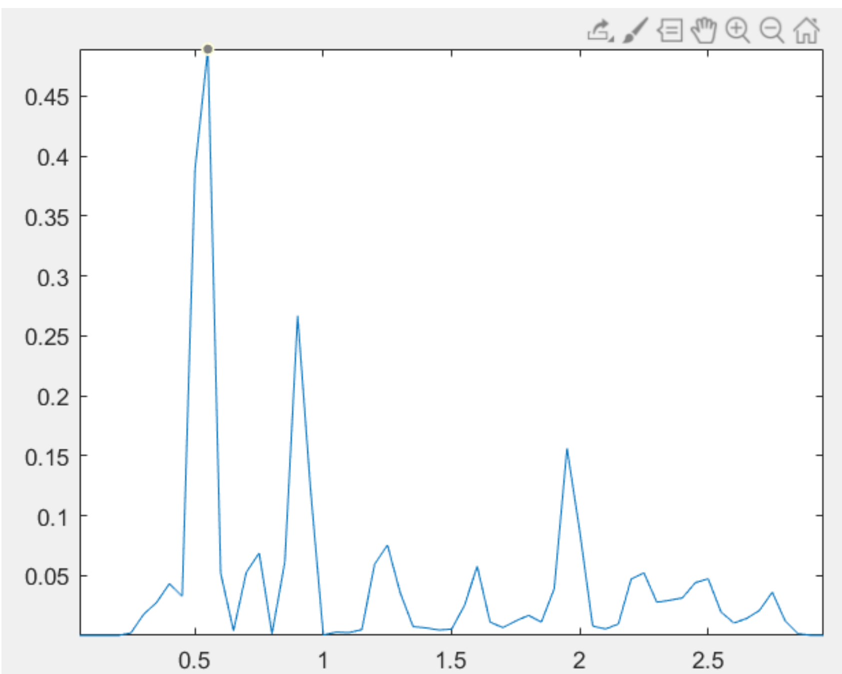
Window (10msec)



Window (50msec)



Window (100msec)



-What do you observe as the length of the window gets bigger?

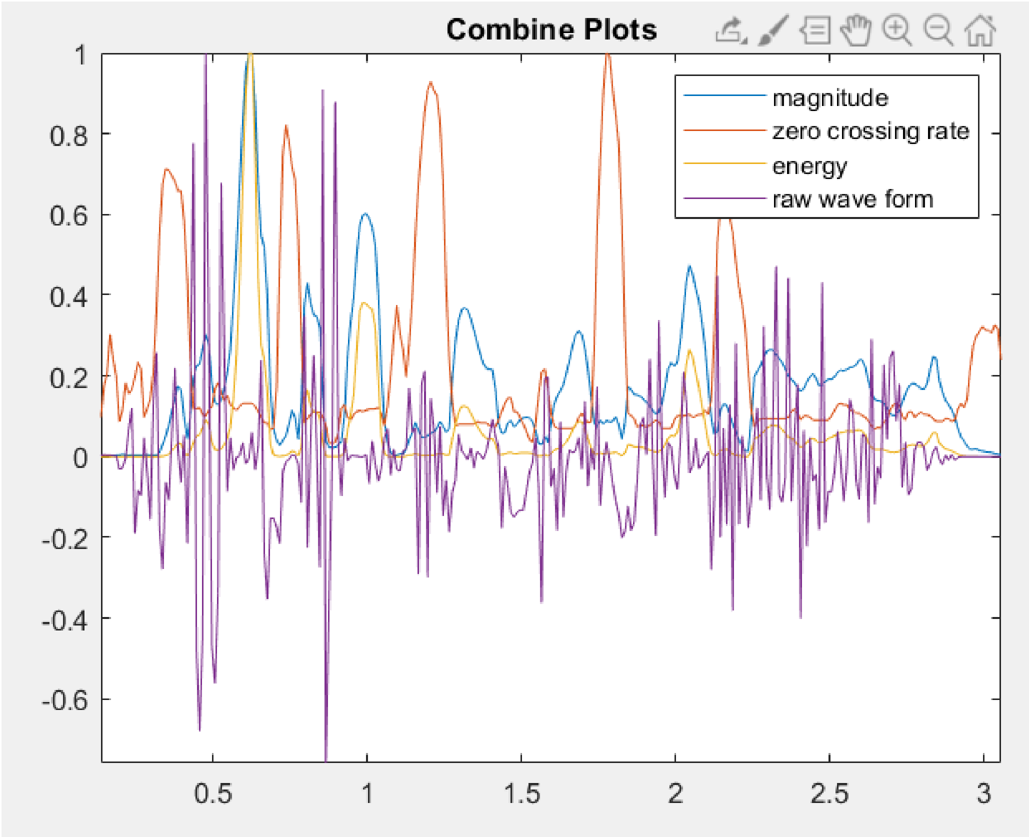
=>Generally, the shape of plot gets smoother as the length of the window gets bigger.

We can observe many little oscillations disappear in the third plot.

-What's the length of your energy vector in the three cases?

=>Window(10msec): 585, Window(50msec): 117, Window(100msec): 59

(c)



(d)Yes. We can use those measurements to distinguish between voiced and unvoiced sound. And between silence and speech.

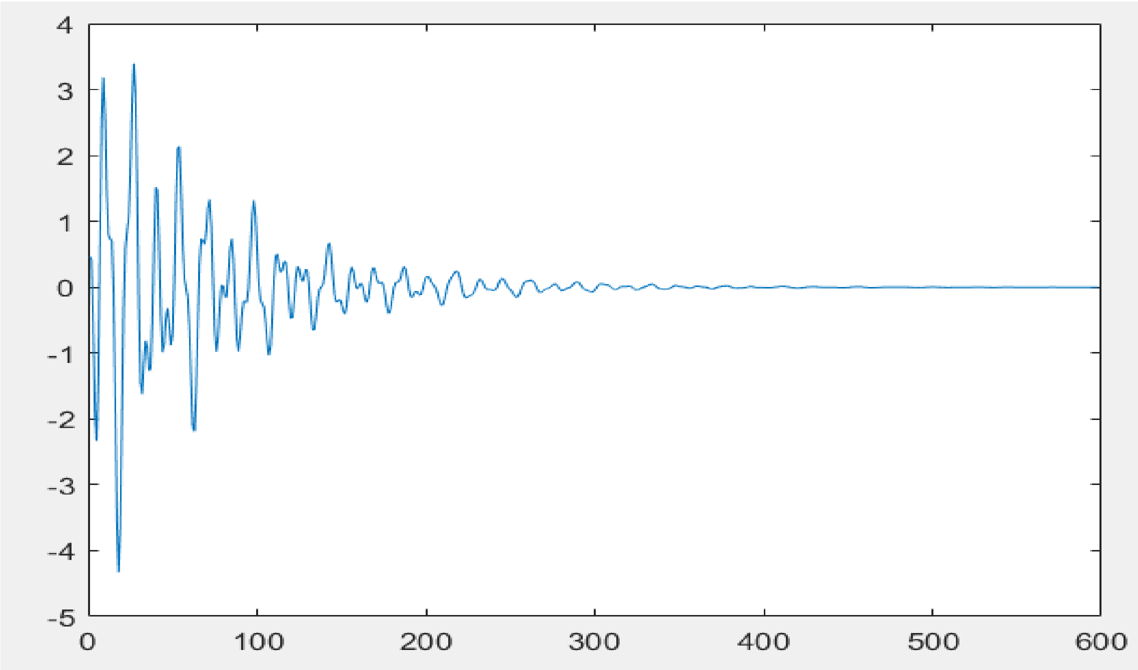
We can find out where voiced sound occurs by pointing out an area high magnitude/energy and low zero crossing rate. On the other hand, unvoiced sound occurs at the location where low magnitude/energy and high zero crossing rate.

For the silence we can pay attention to where value of energy/magnitude is small.

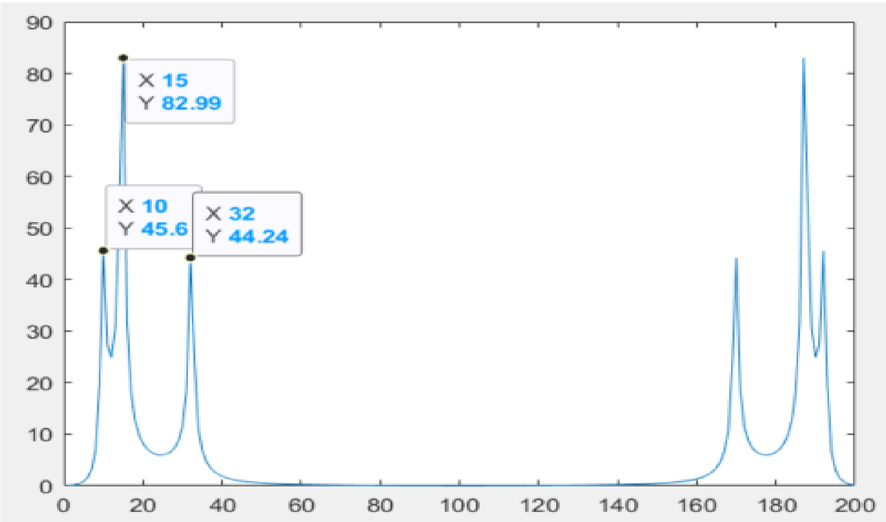
Problem2.

(a)

(a-1) Impulse Response.



(a-2) Magnitude of its frequency response



First frequency: 10 x Fs / N_dft = 10 x 16000 / 200 = 800 Hz

Second frequency: 15 x Fs / N_dft = 50 x 16000 / 200 = 1200 Hz

Third frequency: 32 x Fs / N_dft = 100 x 16000 / 200 = 2560 Hz

- ⇨ Yes, these converted frequencies are what I expect to see.

(b)

(b-1) f0 = 200 Hz

16000 samples in 1 second.

200 non-zero samples.

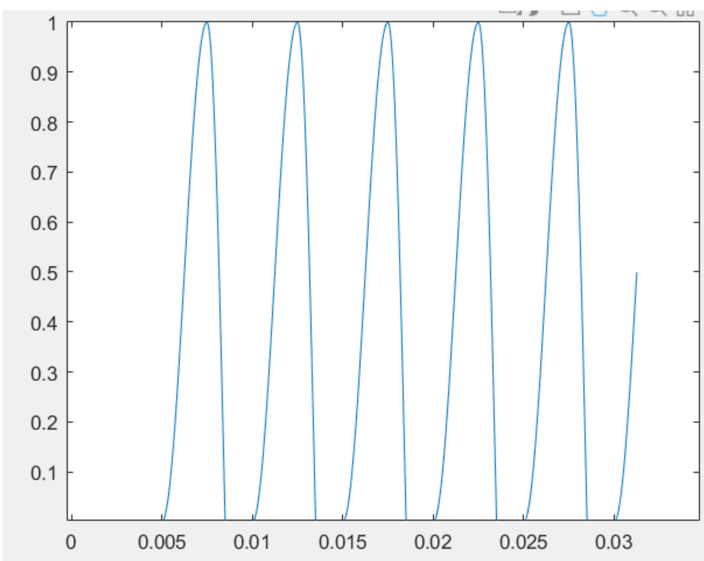
(b-2) f0 = 125 Hz

16000 samples in 1 second.

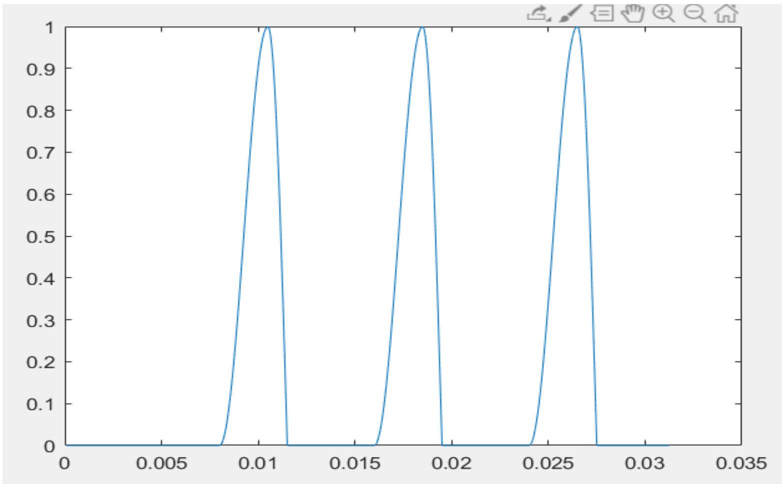
125 non-zero samples

(c)

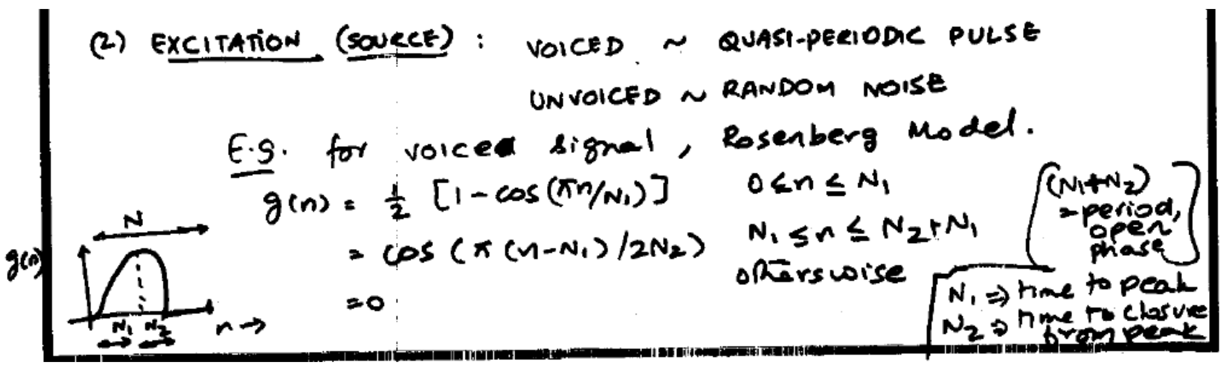
(c-1) f0 = 200 Hz



(c-2) f0 = 125 Hz



T1 represents time to peak, T2 represents time to closure from peak. Details are below


(2) EXCITATION (SOURCE) : VOICED ~ QUASI-PERIODIC PULSE

UNVOICED ~ RANDOM NOISE

E.g. for voiced signal, Rosenberg Model.

$g(n) = \frac{1}{2}\left[1 - \cos(\pi n / N_1)\right]$ $\quad 0 \le n \le N_1$

$= \cos(\pi (n - N_1)/2N_2)$ $\quad N_1 \le n \le N_2 + N_1$

$= 0$ $\quad$ otherswise

$(N_1 + N_2)$ = period, open phase

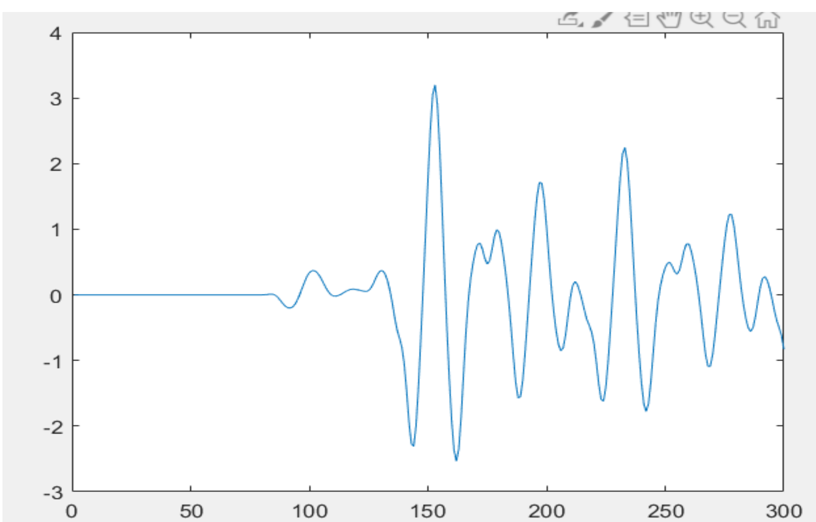
$N_1 \Rightarrow$ time to peak

$N_2 \Rightarrow$ time to closure from peak

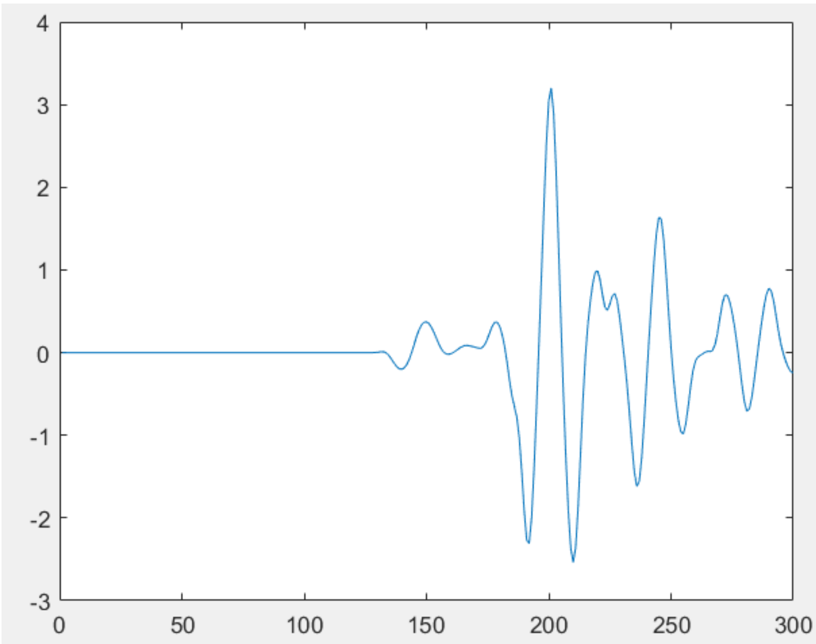
[This reference is from the lecture note week 2 in EE519]

(d)

(d-1) f0 = 200 Hz



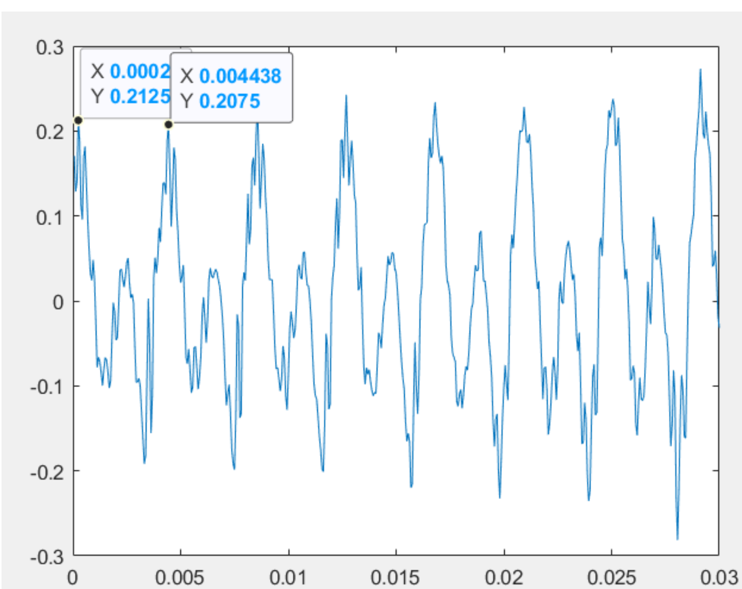
(d-2) f0=125 Hz



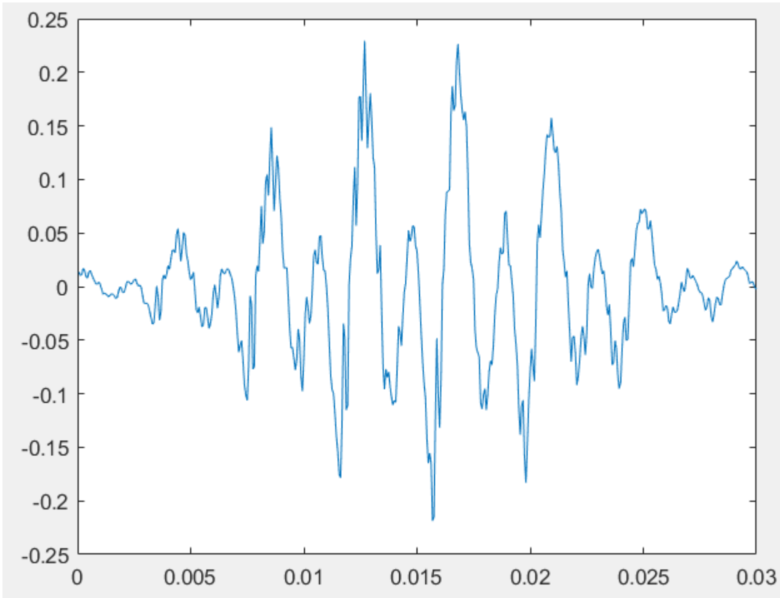
Sound created by f0 = 200 Hz has a little bit higher pitch.

Problem 3.

(i)Non windowed signal s[n]



(ii)

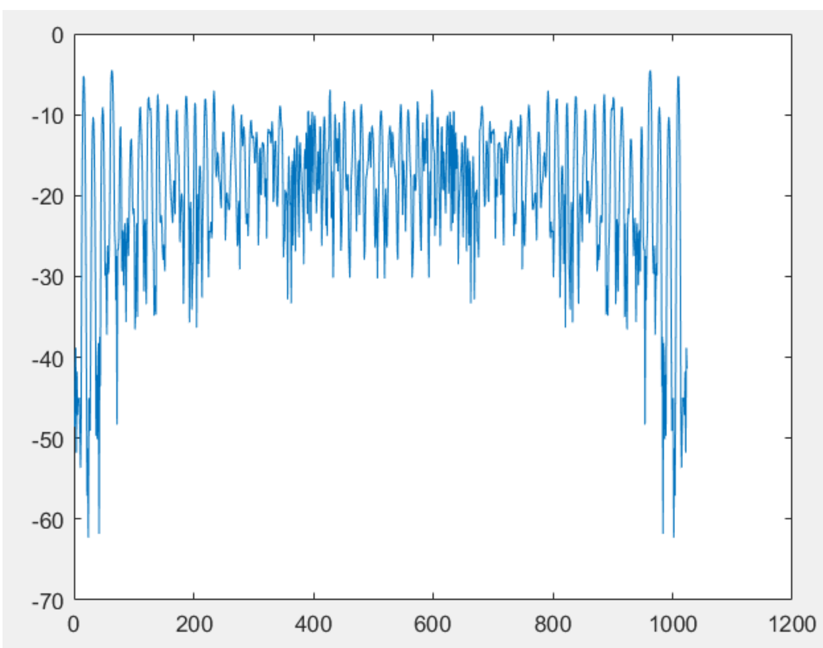


L = (0.0044381-0.00025) x Fs = 67 samples

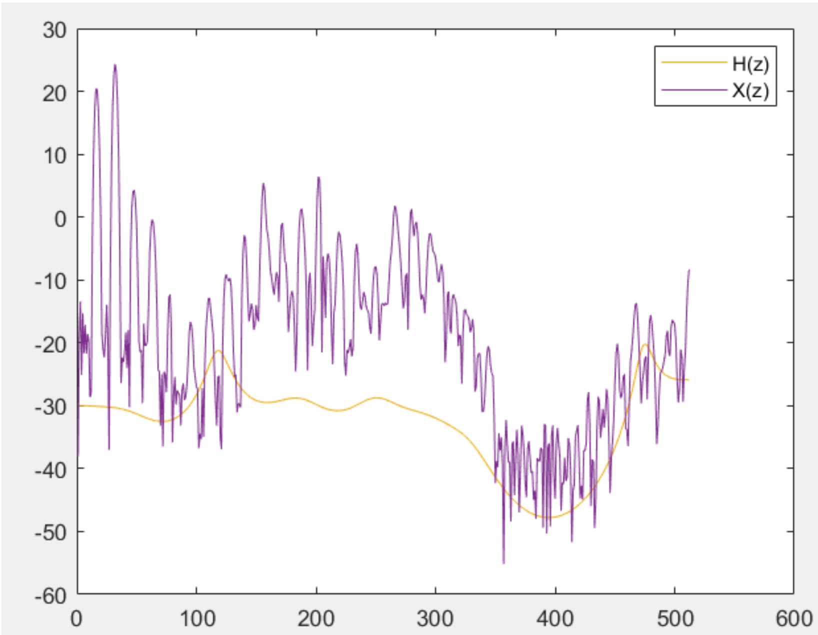
(b)

p = 16 + 4 = 20

(i)the spectrum of the error e[n]



(ii) the spectrum of the LPC model in a common figure with original spectrum of x[n]



- ⇨ I can observe LPC can capture the shape of output X(z) as p increases.

This is because order p is related to number of poles.

Appendix

*********************Pr1_HW1.m****************************

```
%% (a)Properties of window functions
%Create window functions. 100samples for each window
%Rectangular window
numOfSample = 100;
recWindow = rectwin(numOfSample);

%Hamming window
hammingWindow = hamming(numOfSample);

%Visualization for each window in time and frequency domain
wvtool(recWindow)

wvtool(hammingWindow)

%% (b)Short time power estimation
fileName = 'hw2_TIMIT_LDC93S1.wav';
[audioSignal,Fs ]= audioread(fileName);
audioSignal = audioSignal' ; % make it 1xlength form

%zero pad audio signal to avoid a case that the last window
overshoots the
%length of signal(original length = 46797 -> padded length
= 46800
paddedLength = 46800;
zeroPaddedAudioSignal = zeros(1,paddedLength);

for index = 1:size(zeroPaddedAudioSignal,2)
    if index <= size(audioSignal,2)
        zeroPaddedAudioSignal(index) = audioSignal(index);
    end
end

%Create windows with different sizes. e.g., 10 ,50, 100
lengthOfWindow10 = 160; %0.01*16000 = 160 samples
lengthOfWindow50 = 800; %0.05*16000 = 800 samples
lengthOfWindow100 = 1600; %0.1*16000 = 1600 samples

hammingWindow10 = hamming(lengthOfWindow10);
```

```
hammingWindow50 = hamming(lengthOfWindow50);
hammingWindow100 = hamming(lengthOfWindow100);

%Frame signal with 50% overlap
%length of 160 => 50% overlap is 5 samples
lengthOfOverlap_10 = 80;
framedZeroPaddedAudioSignal_10 =
buffer(zeroPaddedAudioSignal,lengthOfWindow10,lengthOfOverl
ap_10);
%transpose the framed signal
framedZeroPaddedAudioSignal_10 =
framedZeroPaddedAudioSignal_10' ;

%Calculate x_m with length10 window (x_m is a notation from
HW2 pr1(b)
%description
x_m_storage_10 = framedZeroPaddedAudioSignal_10;%initialize
for row = 1:size(framedZeroPaddedAudioSignal_10,1)
    for col = 1:size(framedZeroPaddedAudioSignal_10,2)
        x_m_storage_10(row,col) =
framedZeroPaddedAudioSignal_10(row,col)*hammingWindow10(col
);
    end
end

%Calculate E_m with length10 window (E_m is a notation form
the HW2 pr1(b)
%description
E_m_storage_10 =
zeros(size(x_m_storage_10,1),1);%initialize 9360x1 size
storage
for index = 1:size(E_m_storage_10,1)
    temp = 0;
    for col = 1:size(x_m_storage_10,2)
        %See the formula in HW2 pr1 (b)
        temp =
temp+x_m_storage_10(index,col)*x_m_storage_10(index,col);
    end
    E_m_storage_10(index) = temp;
end

%Transpose E_m_storage to plot propery ( time VS E_m )
E_m_storage_10 = E_m_storage_10';
```

```
%Plot time VS E_m graph
%Generate continuous time array for plotting
lengthInTimeUnit_10 = 0.01;%10msec
percentageOfOverlap_10 = 50;% 50%
numOfFrame_10 = size(framedZeroPaddedAudioSignal_10,1);
timeArray_10 =
continuousTimeGenerator(lengthInTimeUnit_10,percentageOfOve
rlap_10,numOfFrame_10);

figure(5)
plot(timeArray_10,E_m_storage_10)
axis tight


%length of 800 => 50% overlap is 400 samples
lengthOfOverlap_50 = 400;
framedZeroPaddedAudioSignal_50 =
buffer(zeroPaddedAudioSignal, lengthOfWindow50,
lengthOfOverlap_50);
%transpose the framed signal
framedZeroPaddedAudioSignal_50 =
framedZeroPaddedAudioSignal_50' ;

%Calculate x_m with length 800 window (x_m is a notation
from HW2 pr1(b)
%description
x_m_storage_50 = framedZeroPaddedAudioSignal_50;%initialize
for row = 1:size(framedZeroPaddedAudioSignal_50,1)
    for col = 1:size(framedZeroPaddedAudioSignal_50,2)
        x_m_storage_50(row,col) =
framedZeroPaddedAudioSignal_50(row,col)*hammingWindow50(col
);
    end
end

%Calculate E_m with length50 window (E_m is a notation form
the HW2 pr1(b)
%description
E_m_storage_50 =
zeros(size(x_m_storage_50,1),1);%initialize 1872x1 size
storage
for index = 1:size(E_m_storage_50,1)
    temp = 0;
    for col = 1:size(x_m_storage_50,2)
```

```
        %See the formula in HW2 pr1 (b)
        temp =
temp+x_m_storage_50(index,col)*x_m_storage_50(index,col);
    end
    E_m_storage_50(index) = temp;
end

%Transpose E_m_storage to plot propery ( time VS E_m )
E_m_storage_50 = E_m_storage_50';
%Plot time VS E_m graph
%Generate continuous time array for plotting
lengthInTimeUnit_50 = 0.05;%50msec
percentageOfOverlap_50 = 50;% 50%
numOfFrame_50 = size(framedZeroPaddedAudioSignal_50,1);
timeArray_50 =
continuousTimeGenerator(lengthInTimeUnit_50,percentageOfOve
rlap_50,numOfFrame_50);

figure(6)
plot(timeArray_50,E_m_storage_50)
axis tight



%length of 1600 => 50% overlap is 800 samples
lengthOfOverlap_100 = 800;
framedZeroPaddedAudioSignal_100 =
buffer(zeroPaddedAudioSignal, lengthOfWindow100,
lengthOfOverlap_100);
%transpose the framed signal
framedZeroPaddedAudioSignal_100 =
framedZeroPaddedAudioSignal_100' ;

%Calculate x_m with length 1600 window (x_m is a notation
from HW2 pr1(b)
%description
x_m_storage_100 =
framedZeroPaddedAudioSignal_100;%initialize
for row = 1:size(framedZeroPaddedAudioSignal_100,1)
    for col = 1:size(framedZeroPaddedAudioSignal_100,2)
        x_m_storage_100(row,col) =
framedZeroPaddedAudioSignal_100(row,col)*hammingWindow100(c
ol);
    end
```

```
end

%Calculate E_m with length100 window (E_m is a notation
form the HW2 pr1(b)
%description
E_m_storage_100 =
zeros(size(x_m_storage_100,1),1);%initialize 936x1 size
storage
for index = 1:size(E_m_storage_100,1)
    temp = 0;
    for col = 1:size(x_m_storage_100,2)
        %See the formula in HW2 pr1 (b)
        temp =
temp+x_m_storage_100(index,col)*x_m_storage_100(index,col);
    end
    E_m_storage_100(index) = temp;
end

%Transpose E_m_storage to plot propery ( time VS E_m )
E_m_storage_100 = E_m_storage_100';
%Plot time VS E_m graph
%Generate continuous time array for plotting
lengthInTimeUnit_100 = 0.1;%100msec
percentageOfOverlap_100 = 50;% 50%
numOfFrame_100 = size(framedZeroPaddedAudioSignal_100,1);
timeArray_100 =
continuousTimeGenerator(lengthInTimeUnit_100,percentageOfOv
erlap_100,numOfFrame_100);

figure(7)
plot(timeArray_100,E_m_storage_100)
axis tight

%% (c)Average Magnitude and average zero crossing
%Average Magnitude
%Create Hamming window
lengthOfHamming25 = 400; % 0.025*16000 = 400 samples
hammingWindow25 = hamming(lengthOfHamming25);

normalized_M_n_storage25 =
normalizedMnCalculator(zeroPaddedAudioSignal,hammingWindow2
5);

%Generate continuous time array for plotting
```

```
lengthInTimeUnit_25 = 0.025;%25msec
percentageOfOverlap_25 = 60;% 60%
numOfFrame_25 = size(normalized_M_n_storage25,2);
timeArray_25 =
continuousTimeGenerator_Overlap60(numOfFrame_25);



figure(8)
p1 = plot(timeArray_25,normalized_M_n_storage25);
axis tight
title('Combine Plots')
hold on

%Average Zero Crossing rate
%Create a defined rentagular window
lengthOfRecWindow25 = 400; % 0.025*16000 = 400 samples
originalRecWindow25 = rectwin(lengthOfRecWindow25);
definedRecWindow25 = (1/(2*lengthOfRecWindow25)) .*
originalRecWindow25;

normalized_ZeroCrossingRate_storage25 =
normalizedZeroCrossingRateCalculator(zeroPaddedAudioSignal,
definedRecWindow25);


p2 =
plot(timeArray_25,normalized_ZeroCrossingRate_storage25);
axis tight



%Frame signal with 50% overlap
%length of 160 => 50% overlap is 5 samples
lengthOfOverlap_25 = 240;
framedZeroPaddedAudioSignal_25 =
buffer(zeroPaddedAudioSignal, lengthOfHamming25,
lengthOfOverlap_25);
%transpose the framed signal
framedZeroPaddedAudioSignal_25 =
framedZeroPaddedAudioSignal_25';

%Calculate x_m with length25 window (x_m is a notation from
HW2 pr1(b)
```

```
%description
x_m_storage_25 = framedZeroPaddedAudioSignal_25;%initialize
for row = 1:size(framedZeroPaddedAudioSignal_25,1)
    for col = 1:size(framedZeroPaddedAudioSignal_25,2)
        x_m_storage_25(row,col) =
framedZeroPaddedAudioSignal_25(row,col)*hammingWindow25(col
);
    end
end

%Calculate E_m with length25 window (E_m is a notation form
the HW2 pr1(b)
%description
E_m_storage_25 =
zeros(size(x_m_storage_25,1),1);%initialize 9360x1 size
storage
for index = 1:size(E_m_storage_25,1)
    temp = 0;
    for col = 1:size(x_m_storage_25,2)
        %See the formula in HW2 pr1 (b)
        temp =
temp+x_m_storage_25(index,col)*x_m_storage_25(index,col);
    end
    E_m_storage_25(index) = temp;
end

%Transpose E_m_storage to plot propery ( time VS E_m )
E_m_storage_25 = E_m_storage_25';
%normalize
max_E_m = max(E_m_storage_25);
normalized_E_m_storage_25 = E_m_storage_25./max_E_m;

p3 = plot(timeArray_25,normalized_E_m_storage_25);
axis tight

%Sampling raw wave
sampledRawWave =
samplingOriginalSignal(zeroPaddedAudioSignal,timeArray_25,F
s);
%normalize sampled raw wave
max_sampled_raw_wave = max(abs(sampledRawWave));
normalized_sampled_raw_wave =
sampledRawWave./max_sampled_raw_wave;
```

```
p4=plot(timeArray_25,normalized_sampled_raw_wave);
axis tight

hold off

%Create Legends
h = [p1;p2;p3;p4];

legend(h,'magnitude','zero crossing rate', 'energy','raw
wave form');
```

*****************continuousTimeGenerator.m*******************

```
function continuousTimeArray =
continuousTimeGenerator(lengthInTimeUnit,OverlapPercent,num
OfFrame)
%CONTINUOUSTIMEGENERATOR Summary of this function goes here
%Generate time array that corresponds to short time
analysis result
%   Detailed explanation goes here
%lengthInTimeUnit : length in time unit e.g. 10msec => 0.01
sec
%OverlapPercent : how much percent each frame is overlapped
e.g., 50 => 50%
%numOfFrame: number of frame.

percentScaler = 0.01; %Change percentage scale e.g. 50% =>
0.5
initialTime = lengthInTimeUnit * percentScaler*
OverlapPercent;

continuousTimeArray = zeros(1,numOfFrame);

for col = 1 : numOfFrame
    continuousTimeArray(1,col) = initialTime*col;
end
```

**************************continuousTimeGenerator_Overlap60.m***************

```
function continuousTimeArray =
continuousTimeGenerator_Overlap60(numOfFrame)
%CONTINUOUSTIMEGENERATOR Summary of this function goes here
```

```
%Generate time array that corresponds to short time
analysis result
%   Detailed explanation goes here
%lengthInTimeUnit : length in time unit e.g. 10msec => 0.01
sec
%OverlapPercent : how much percent each frame is overlapped
e.g., 50 => 50%
%numOfFrame: number of frame.

initialTime = 0.125;
timeInterval = 0.01;
continuousTimeArray = zeros(1,numOfFrame);

for col = 1 : numOfFrame
    continuousTimeArray(1,col) =
initialTime+timeInterval*col;
end
```

**************************normalizedMnCalculator.m**********

```
function normalizedMnStorage = normalizedMnCalculator(x,w)
%NORMALIZEDMNCALCULATOR Summary of this function goes here
%Calculate normalized average magnitude(M_n) and save them
to storage
%x: original signal. 1 x length form
%w: window. 1 x window length form
%   Detailed explanation goes here

lengthOfWindow = size(w,1);%In this homework, it is 25

%Frame signal with 60% overlap
%Length of 400 => 60% overlap is 240 samples
lengthOfOverLap_25 = 240;
framedAudioSignal_25 = buffer(x,
lengthOfWindow,lengthOfOverLap_25);
%transpose the framed signal
framedAudioSignal_25 = framedAudioSignal_25' ;

%Calculate Mn with framed signal x and window
M_n_storage_25 =
zeros(size(framedAudioSignal_25,1),1); %initialize storage
for index = 1:size(M_n_storage_25,1)
    temp = 0;
    for col = 1:size(framedAudioSignal_25,2)
```

```
        %See the formula in HW2 pr1 (c)
        temp =
temp+abs(framedAudioSignal_25(index,col))*w(col);
    end
    M_n_storage_25(index) = temp;
end

%Transpose E_m_storage to plot propery ( time VS M_n )
M_n_storage_25 = M_n_storage_25';

%normalize with max
maxIn_M_n_storage = max(M_n_storage_25);

normalized_M_n_storage_25 = M_n_storage_25 ./
maxIn_M_n_storage ;

normalizedMnStorage = normalized_M_n_storage_25;

end
```

```
************************normalizedZeroCrossingRateCalculator.m

function normalized_ZeroCrossingRate_storage25 =
normalizedZeroCrossingRateCalculator(x,w)
%NORMALIZEDZEROCROSSINGRATECALCULATOR Summary of this
function goes here
%Calculater normalized zero crossing rates, save them into
storage, and
%return the storage
%x: original signal. 1 x length form
%w: window. 1 x window length form
%   Detailed explanation goes here
lengthOfWindow = size(w,1);%In this homework, it is 25

%Frame signal with 60% overlap
%Length of 400 => 60% overlap is 240 samples
lengthOfOverLap_25 = 240;
framedAudioSignal_25 = buffer(x,
lengthOfWindow,lengthOfOverLap_25);
%transpose the framed signal
framedAudioSignal_25 = framedAudioSignal_25' ;
```

```
%Calculate Mn with framed signal x and window
Z_n_storage_25 =
zeros(size(framedAudioSignal_25,1),1); %initialize storage

for index = 1:size(Z_n_storage_25,1)
    temp = 0;
    for col = 1:size(framedAudioSignal_25,2)
        %See the formula in HW2 pr1 (c)
        if (col-1) ~= 0
            temp =
temp+abs(sign(framedAudioSignal_25(index,col)) -
sign(framedAudioSignal_25(index,col-1)) )*w(col);
        elseif (col-1) ==0 %handle the case index is out of
boundary
            temp = temp +
abs( sign(framedAudioSignal_25(index,col) ) - 0) *w(col);
        end
    end
    Z_n_storage_25(index) = temp;
end

%Transpose Z_n_storage to plot propery ( time VS Z_n )
Z_n_storage_25 = Z_n_storage_25';

%normalize with max
maxIn_Z_n_storage = max(Z_n_storage_25);

normalized_Z_n_storage_25 = Z_n_storage_25 ./
maxIn_Z_n_storage ;

normalized_ZeroCrossingRate_storage25 =
normalized_Z_n_storage_25;

end




*************samplingOriginalSignal
function sampledRawWave =
samplingOriginalSignal(rawWave,timeArray, Fs)
%SAMPLINGORIGINALSIGNAL Summary of this function goes here
%   Detailed explanation goes here
```

```
indexStorageForRawWave = floor(timeArray*Fs);
sampledRawWave = zeros(1,size(indexStorageForRawWave,2));

for col = 1:size(indexStorageForRawWave,2)
    current_index = indexStorageForRawWave(1,col);
    if(current_index <= size(rawWave,2))
        sampledRawWave(1,col) = rawWave(1,current_index);
    end
end

end
```

```
*****************************Pr2_HW2.m

%% (a)
%Generate impulse response for /aa/
%Notations are from HW2 problem2 description
F1_aa = 700; %Fi_aa : ith formant frequency for /aa/ vowel
F2_aa = 1100;
F3_aa = 2500;
lengthOfImpulseResponse = 200; % choose the length that is
not too long
Fs = 16000; %Sampling frequency is 16000Hz


h1_aa = h_ith_Generator(F1_aa,lengthOfImpulseResponse,Fs);
h2_aa = h_ith_Generator(F2_aa,lengthOfImpulseResponse,Fs);
h3_aa = h_ith_Generator(F3_aa,lengthOfImpulseResponse,Fs);

%h_aa is a cascaded system of h1_aa, h2_aa, and h3_aa
%A cascaded system can be designed with a serial of
convolutions.
h_aa = conv(conv(h1_aa,h2_aa), h3_aa);

dft_h_aa = fft(h_aa,lengthOfImpulseResponse);

figure(1)
plot(abs(dft_h_aa))
```

```
figure(2)
plot(h_aa);
%wvtool(h_aa)

%% (b)
Fs = 16000;%sampling frequency is 16000Hz
%(1)
f0_1 = 200;%fundamental frequency is 200Hz
impulseTrain_f0_1 = impulseTrainGenerator(f0_1,Fs);
%(2)
f0_2 = 125;%fundamental frequency is 125Hz
impulseTrain_f0_2 = impulseTrainGenerator(f0_2,Fs);

%% (c)
%Generate g[n]
T1 = 40;
T2 = 16; %T1 and T2 are given in HW2 description
Ts =1/Fs;
g_n = RosenbergGlottalGenerator(T1,T2);


%generate glottal pulse train by convolving impulse train
from (b) and g[n]
%from (c)
glottalPulseTrain_f0_1 = conv(g_n,impulseTrain_f0_1);
glottalPulseTrain_f0_2 = conv(g_n,impulseTrain_f0_2);

lengthOfPartialPulseTrain = 500;
partOfGlottalPulseTrain_f0_1 =
glottalPulseTrain_f0_1(1,1:lengthOfPartialPulseTrain);
partOfGlottalPulseTrain_f0_2 =
glottalPulseTrain_f0_2(1,1:lengthOfPartialPulseTrain);

continuousTimeArray = [1:1:lengthOfPartialPulseTrain]*Ts;

%Plotting
figure(3)
plot(continuousTimeArray,partOfGlottalPulseTrain_f0_1)

figure(4)
plot(continuousTimeArray,partOfGlottalPulseTrain_f0_2)

%% (d)
```

```
%Excitate the filters built in (a) with the source from(c)
i.e., convolve
%the glottal pulse train with system from(a) to synthesize
the vowels

syntheticVowel_f0_1 = conv(glottalPulseTrain_f0_1,h_aa);
syntheticVowel_f0_2 = conv(glottalPulseTrain_f0_2,h_aa);

%Take the first 300 samples
lengthOfFirst300Samples = 300;

syntheticVowel_f0_1_First300 =
syntheticVowel_f0_1(1:lengthOfFirst300Samples);
syntheticVowel_f0_2_First300 =
syntheticVowel_f0_2(1:lengthOfFirst300Samples);

%Plotting
figure(5)
plot(syntheticVowel_f0_1_First300)

figure(6)
plot(syntheticVowel_f0_2_First300)


% %Sound Checking
% sound(syntheticVowel_f0_1,Fs)
%
% sound(syntheticVowel_f0_2,Fs)
```

```
******************************h_ith_Generator.m

function h_ith =
h_ith_Generator(formantFrequency,lengthOfImpulseResponse,Fs
)
%H_ITH_GENERATOR Summary of this function goes here
%Generate an impulse response with given length and formant
frequency
%   Detailed explanation goes here
%formantFrequency : formant frequency
%lenghtOfImpulseResponse: Length of impulse response to be
created
%Fs: Sampling frequency
```

```
h_ith = zeros(1, lengthOfImpulseResponse);%initialize h_ith
sequence
Ts = 1/Fs;
a_ith = 0.005*pi - 0.01*formantFrequency*Ts;
w_ith = 2*pi*formantFrequency/Fs;

for index = 1:lengthOfImpulseResponse
    %Refer to formula for h_ith in HW2 description
    h_ith(index) = exp(-a_ith*index)*cos(w_ith*index)*1;
end



end
```

```
*************************impulseTrainGenerator.m
function impulseTrainSequence =
impulseTrainGenerator(f0,Fs)
%IMPULSETRAINGENERATOR Summary of this function goes here
%Generate impulse train sequence based on the given
fundamental frequency
%f0: Fundamental frequency
%Fs: Sampling Rate

%   Detailed explanation goes here

impulseTrainSequence = zeros(1,Fs); %1sec long sequence
initialNonZeroLocation = Fs/f0;

for col= 1:Fs
    if(mod(col,initialNonZeroLocation)==0)
        impulseTrainSequence(1,col) = 1;
    else
        impulseTrainSequence(1,col) = 0;
    end
end

end
```

```
***********************RosenbergGlottalGenerator.m
function glottalSequence = RosenbergGlottalGenerator(T1,T2)
%ROSENBERGGLOTTALGENERATOR Summary of this function goes
here
%Generate a Rosenberg's glottal response using equation
from HW2
%description
%   Detailed explanation goes here
lengthOfTimeSequence = T1+T2+1;

timeSequence = [0:1:lengthOfTimeSequence];

glottalSequence = zeros(1,lengthOfTimeSequence);

for index = 1:lengthOfTimeSequence
    if(timeSequence(1,index)>=0 &&
timeSequence(1,index)<=T1)
        glottalSequence(1,index) = 0.5*(1-
(cos(2*pi*timeSequence(1,index)/(2*T1))));
    elseif(timeSequence(1,index)>T1 &&
timeSequence(1,index)<=T1+T2)
        glottalSequence(1,index) =
cos((2*pi*(timeSequence(1,index)-T1))/(4*T2) );
    end
end

end
```

```
****************************Pr3_HW2.m
%% (a)
fileName = 'hw2_TIMIT_LDC93S1.wav';
[audioSignal,Fs ]= audioread(fileName);
audioSignal = audioSignal' ; % make it 1xlength form

Ts = 1/Fs; %Sampling interval.
```

```
%Normalize the audio signal
abs_maxValueOfSignal = max(abs(audioSignal));
normalizedAudioSignal = audioSignal ./
(abs_maxValueOfSignal);

timeLength = 0.03; %30 msec
lengthOfPartialSignal = Fs*timeLength;
halfLengthOfPartialSignal = lengthOfPartialSignal/2;
timeLocationOfVoicedSound = 0.33; %Not sure this one yet
CenterTargetIndex = timeLocationOfVoicedSound*Fs;
indexes = [(CenterTargetIndex-
halfLengthOfPartialSignal+1):(CenterTargetIndex+halfLengthO
fPartialSignal)];

%s_n represents s[n] in HW2 descrition.
s_n = normalizedAudioSignal(indexes);

continuousTimeArray = [1:lengthOfPartialSignal].*Ts;

%Generate Hamming window
hammingWindow = hamming(lengthOfPartialSignal);
hammingWindow = hammingWindow'; % transpose

%Window s[n]
windowed_s_n = s_n .* hammingWindow;


%plotting
%(1)continuous time vs non-windowed s[n]
figure(1)
plot(continuousTimeArray, s_n)

%(2)continuous time vs windowed s[n]
figure(2)
plot(continuousTimeArray, windowed_s_n)

%% (b)
N_fft = 1024; %Length of DFT
orderOf_p = 20; % It can change depending on model you want
to estimate
% %portion of normalized original signal
% %Perform Linear Prediction Filter Coefficient
% a_nonWindowed = lpc(s_n,orderOf_p);
```

```matlab
% est_s_n_nonWindowed = filter([1 -
a_nonWindowed(2:end)],1,s_n); %
%
% error_nonWindowed = s_n - est_s_n_nonWindowed;
% gain_nonWindowed = gainEstimator(error_nonWindowed);
%
% %Calculate error spectrum
% dft_error_nonWindowed = fft(error_nonWindowed,N_fft);
% %Change unit to dB
% Magnitude_eft_error_nonWindowed_dB =
20*log10(abs(dft_error_nonWindowed));
% %Plot
% figure(3)
% plot(Magnitude_eft_error_nonWindowed_dB)
%
% %Find H(z) spectrum
% [h_nonWindowed,w] = freqz(gain_nonWindowed,[1 -
a_nonWindowed(2:end)],N_fft);
%
% figure(4)
% plot(abs(h_nonWindowed));

%portion of normalized windowed signal
a_windowed = lpc(windowed_s_n,orderOf_p);

est_s_n_windowed = filter([0 -
a_windowed(2:end)],1,windowed_s_n); %

error_windowed = windowed_s_n - est_s_n_windowed;
gain_windowed = gainEstimator(error_windowed);

%Calculate error spectrum
dft_error_windowed = fft(error_windowed,N_fft);
%Calculate error spectrum
dft_windowed_s_n = fft(windowed_s_n,N_fft);
%Change unit to dB
Magnitude_dft_error_windowed_dB =
20*log10(abs(dft_error_windowed));
Magnitude_dft_windowed_s_n_dB =
20*log10(abs(dft_windowed_s_n));

%Plot
figure(5)
plot(Magnitude_dft_error_windowed_dB)
```

```
%Find H(z) spectrum
[h_windowed,w] = freqz(gain_windowed,[1 -
a_windowed(2:end)],N_fft/2);

figure(6)
p1 = plot(20*log10(abs(h_windowed)));

hold on

p2 = plot(Magnitude_dft_windowed_s_n_dB(1:512));

h = [p1;p2];

legend(h,'H(z)','X(z)')
hold off
```

```
*****************************gainEstimator.m
function gain = gainEstimator(error)
%GAINESTIMATOR Summary of this function goes here
%Estimate gain of LPC based on the given error
%   Detailed explanation goes here

lengthOfError = size(error,2);

temp = 0;
for col = 1:lengthOfError
    temp = temp + error(1,col);
end
%Estimation Formula is in HW2 description
gain = sqrt(temp);


end
```