Speech and Audio Processing - Assignment

CE827 Speech and Audio Processing
Assignment I

Submitted By:
Shah Ankit Parag - 11EC86
Prakash Pali - 11EC67

Department Of Electronics & Communication Engineering
National Institute Of Technology Karnataka, Surathkal
Srinivasnagar 575025 Karnataka India

October 2014

Contents

1 Familiarity with the signal

Question 1

Record a short segment of speech corresponding to the word "zero",
Plot the signal and Compute the signal mean, standard deviation, variance, average power, average magnitude, and number of zero crossings.

Matlab Script

```
%-------------------------------------------------------------
%   Recording Zero for the first time and saving it.
%-------------------------------------------------------------

%recObj = audiorecorder;
%disp('Start speaking.')
%recordblocking(recObj, 2);
%disp('End of Recording.');
%zero=getaudiodata(recObj,'double');
%audiowrite('zero.wav',zero,8000);
%play(recObj);

%-------------------------------------------------------------
%       Read zero.wav file and remove the silent part
%-------------------------------------------------------------
speech_signal = read_remove('zero.wav');

%-------------------------------------------------------------
%       Calculating and displaying number of samples
%       after removal of silence
%-------------------------------------------------------------
no_of_samples = length(speech_signal);
disp('Number of Samples =');disp(no_of_samples);

%-------------------------------------------------------------
%       Finding Mean and displaying it
%-------------------------------------------------------------
mean = sum(speech_signal)/no_of_samples;
disp('Mean =');disp(mean);
```

```
%---------------------------------------------------------------
%       Finding variance and displaying it
%---------------------------------------------------------------
var = sum((speech_signal-mean).^2)/no_of_samples;
disp('Variance = ');disp(var);

%---------------------------------------------------------------
%       Finding standard deviation displaying it
%---------------------------------------------------------------
std_dev = sqrt(var);
disp('Standard deviation = ');
disp(std_dev);

%---------------------------------------------------------------
%       Finding average power displaying it
%---------------------------------------------------------------
avg_power = sum(speech_signal.^2)/no_of_samples;
disp('Average power = ');
disp(avg_power);

%---------------------------------------------------------------
%       Finding average magnitude displaying it
%---------------------------------------------------------------
avg_magnitude = sum(abs(speech_signal))/no_of_samples;
disp('Average magnitude =');
disp(avg_magnitude);

%---------------------------------------------------------------
%       This loop finds the number of zero crossings and
%       displaying it
%---------------------------------------------------------------
zero_cross = 0;
for j = 2 : no_of_samples ;

    if (speech_signal(j)*speech_signal(j-1) < 0 )
        zero_cross = zero_cross + 1;

    end
end

disp('Zeros Crossings = ');
disp(zero_cross);
```

```
%---------------------------------------------------------------
%       Plot the time domain graph of zero
%---------------------------------------------------------------
plot(speech_signal);
xlabel('Time')
ylabel('Amplitude')
title('Time Domain Plot of Zero')
```

Results

Number of Samples = 7200
Mean = -0.0037
Variance = 0.1981
Standard deviation = 0.4450
Average power = 0.1981
Average magnitude = 0.2735
Zeros Crossings = 463

Plots

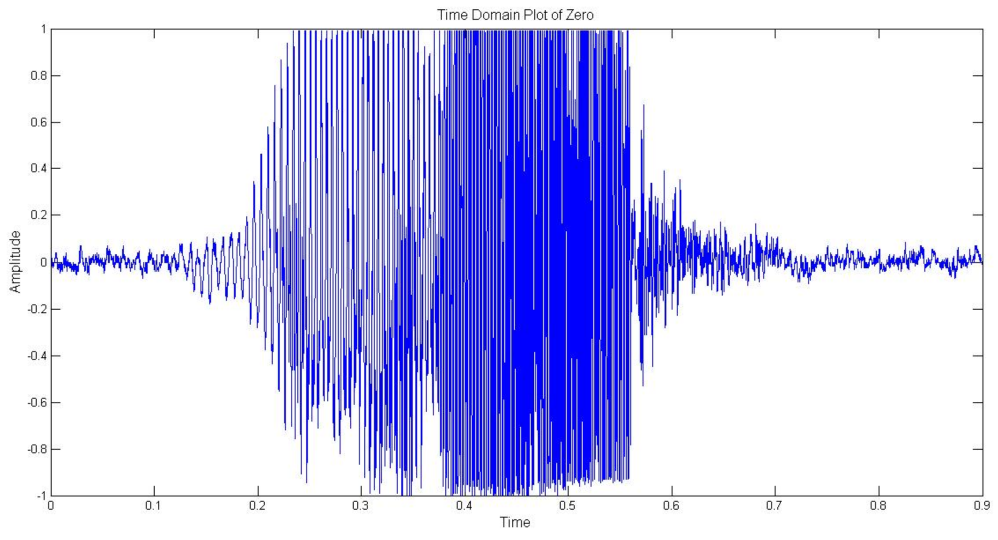


Figure 1: Time Domain Plot of Zero

Question 2

Take a speech signal and pass it through a lowpass filter, bandpass filter and highpass filter. Listen to the filtered signals and comment on the characteristics of the signal such as pitch and quality. Simulate a telephone channel frequency response and use it to the filter the signal. Listen to the filtered signal. What are your observations?

Matlab Script

```
%-------------------------------------------------------------
%       Reading zero.wav and removing silence
%-------------------------------------------------------------
speech= read_remove('zero.wav');

%-------------------------------------------------------------
%       Passing the signal through a LPF
%-------------------------------------------------------------
[numerator,denominator] = butter(10,0.4,'low');
f_low = filter(numerator,denominator,speech);

%-------------------------------------------------------------
%       Passing the signal through a bandpass
%-------------------------------------------------------------
[numerator,denominator] = butter(10,[0.1 0.6],'bandpass');
f_band = filter(numerator,denominator,speech);

%-------------------------------------------------------------
%       Passing the signal through a highpass
%-------------------------------------------------------------
[numerator,denominator] = butter(10,0.6,'high');
f_high = filter(numerator,denominator,speech);

%-------------------------------------------------------------
%       Plotting signal plassed through LPF
%-------------------------------------------------------------
subplot(3,1,1);
plot(f_low);
axis([0,length(speech),-1,1]);
text(5000,-1,'Press Blank Space','BackgroundColor',[.7 .9 .7])
legend('LPF');
xlabel('Time')
```

```
ylabel('Amplitude')
title('Time Domain Plot')
sound(f_low);
pause

%--------------------------------------------------------------
%       Plotting signal plassed through HPF
%--------------------------------------------------------------
subplot(3,1,2);
plot(f_high);
axis([0,length(speech),-1,1]);
text(5000,-1,'Press Blank Space','BackgroundColor',[.7 .9 .7])
legend('HPF');
xlabel('Time')
ylabel('Amplitude')
title('Time Domain Plot')
sound(f_high);
pause

%--------------------------------------------------------------
%       Plotting signal passed through BPF
%--------------------------------------------------------------
subplot(3,1,3);
plot(f_band);
axis([0,length(speech),-1,1]);
legend('BPF');
xlabel('Time')
ylabel('Amplitude')
title('Time Domain Plot')
sound(f_band);
```

Comments

Thus we see that on passing through LPF noise gets cancelled and in High Pass Sound signal gets attenuated and in Band Pass Only Sound signal is passed.
Since most the speech is in 300Hz to 3.4Khz , the telephone simulation channel provides less distortions.

Plots for different filtering operation

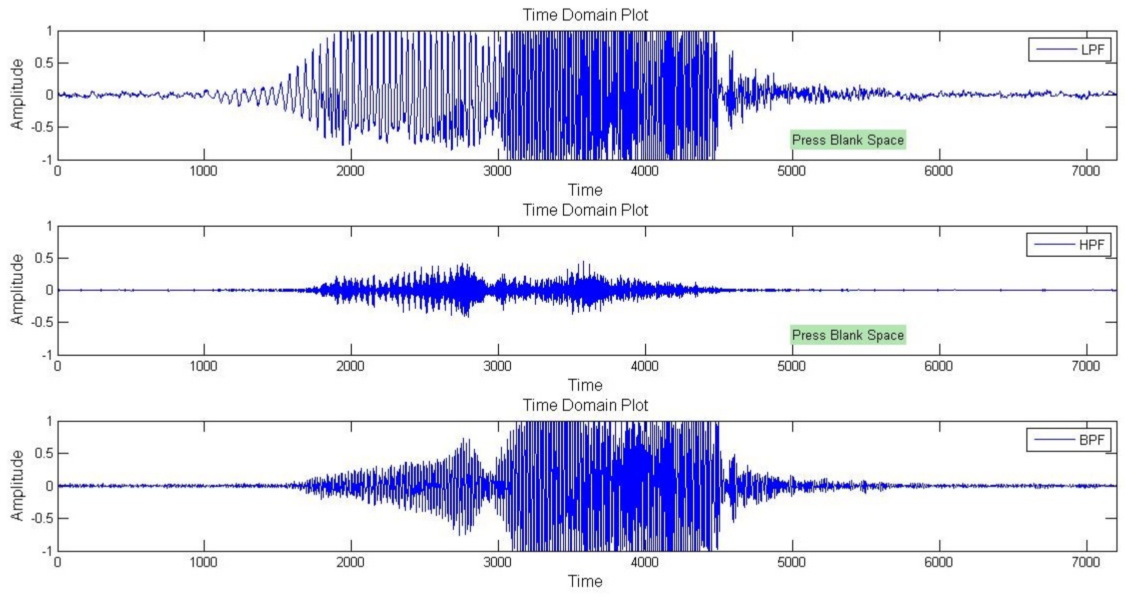


Figure 2: Plots for different filtering operation

Question 3

Record a short segment of speech corresponding to the word ’asa’, Identify the voiced (i.e., /a/) and unvoiced (i.e., /s/) segments of the signal. Determine the pitch and try to identify the gender of the speaker using the pitch information.

Matlab Script

```
%------------------------------------------------------------
%   Recording /asa/ for the first time and saving it.
%-------------------------------------------------------------
%recObj = audiorecorder;
%disp('Start speaking.')
%recordblocking(recObj, 2);
%disp('End of Recording.');
%asa=getaudiodata(recObj,'double');
%audiowrite('asa.wav',asa,8000);
%play(recObj);
```

```
%--------------------------------------------------------------
%       Reading asa.wav and removing silence
%--------------------------------------------------------------
[speech, Fs]= read_remove('asa.wav');

%-------------------------------------------------------------
%       Remove Voiced and Unvoiced speech from actual input
%-------------------------------------------------------------
[voiced, unvoiced] = voiced_unvoiced(speech);
[autocor,lags] = xcorr(voiced);
autocor = autocor(find(lags==0):end);
auto=autocor(200:250);
  max1=0;
  for uu=1:length(auto)
    if(autocor(uu)>max1)
      max1=auto(uu);
      sample_no=uu;
    end
  end

  pitch_period_To=(20+sample_no)*(1/Fs);
  pitch_freq_Fo=1/pitch_period_To;
  disp(pitch_freq_Fo);
```

Results

Pitch is 121.2121
For a pitch frequency lesser than 200 Hz we can say that it is a males voice
For a pitch frequency in range of 170 and 300 Hz it is a females voice.

Question 4

Take about 20s of speech and obtain a histogram of signal amplitudes (for a reasonable bin resolution). How does the histogram look like? Compute the mean and variance.

Matlab Script

```
% This program finds the histogram of the speech amplitudes
%--------------------------------------------------------------
%   Recording /asa/ for the first time and saving it.
%--------------------------------------------------------------
%recObj = audiorecorder;
%disp('Start speaking.')
%recordblocking(recObj, 2);
%disp('End of Recording.');
%asa=getaudiodata(recObj,'double');
%audiowrite('asa.wav',asa,8000);
%play(recObj);

%--------------------------------------------------------------
%       Reading twenty_sec.wav
%--------------------------------------------------------------
input = read_remove('twenty_sec.wav');

%--------------------------------------------------------------
%       hist() function to plot histogram of input
%--------------------------------------------------------------
hist(input);

%--------------------------------------------------------------
%       Calculate mean of input
%--------------------------------------------------------------
mean = 0;
for i=1:length(input)
    mean = mean + input(i);
end
mean = mean/length(input);
disp('mean = ');disp(mean);

%--------------------------------------------------------------
%       std() function calculates standard deviation of input
```

```
%--------------------------------------------------------------
%std = std(input);
deviation=0;
for i=1:length(input)
    deviation = deviation + (mean-input(i))^2;
end
deviation=deviation/length(input);
std=sqrt(deviation);
%--------------------------------------------------------------
%       Variance is square of standard deviation
%--------------------------------------------------------------
variance = std^2;
disp('variance = ');
```

Results

Histograms give a rough sense of the density of the data.
It represents frequencies of the different amplitudes of the signal.
Mean = 1.0757e-04
Variance = 0.1333

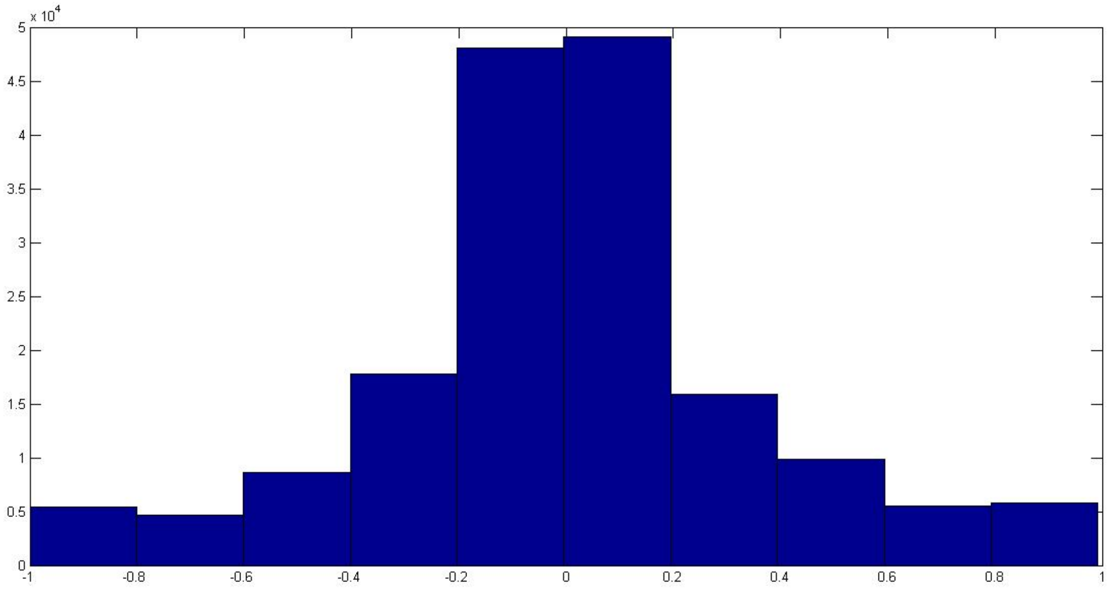


Figure 3: Plots of Histogram

Question 5

Take a short vowel segment and plot its DFT spectrum on a linear scale and log scale. Compare the two representations. Do you find reasons for preferring the latter to the former?

Matlab Script

```
%This program shows the plot of the spectrum in both linear and
%logarithmic scale

%------------------------------------------------------------
%       Reading a.wav
%------------------------------------------------------------
input = read_remove('a.wav');

%------------------------------------------------------------
%       plotting frequency response of input
%------------------------------------------------------------
subplot(2,1,1);
plot(abs(fft(input)));
xlabel('Frequency')
ylabel('Amplitude')
title('Frequency Domain Plot')

%------------------------------------------------------------
%       Plotting log magnitudes of frequency response of input
%------------------------------------------------------------
subplot(2,1,2);
plot(log(abs(fft(input))));
xlabel('Frequency')
ylabel('Log Amplitude')
title('Frequency Domain Plot')
```

Results

The range of the fourier coefficients is too large in linear scale and its difficult to interpret
In this case , linear scale is from 0 to 300
In log scale the range is reduced and the resolution is increased which becomes easy to interpret
In this case, log scale is from -4 to 6

Plot

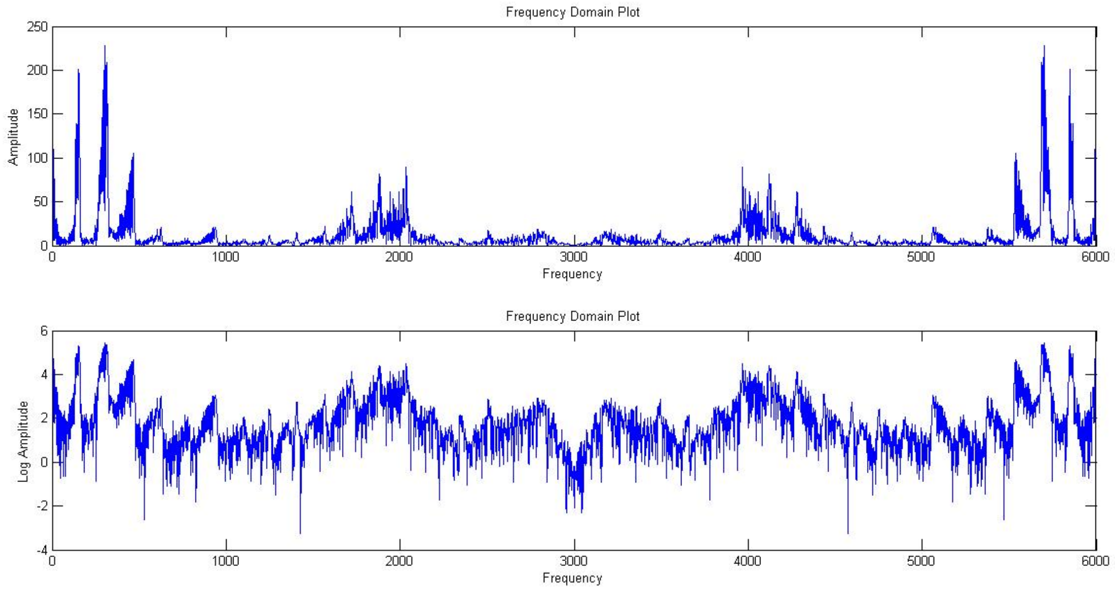


Figure 4: Comparison of Plot of Amplitude vs Frequency in Linear and Log Scale

2 Spectrogram

Question 6

Take a speech signal and compute the following on a linear frequency scale: (i) narrow-band spectrogram (ii) wide-band spectrogram. Repeat the above for a bark scale plot. How do these two spectrographs differ? Experiment with different window lengths and compare the spectrographs obtained.

Matlab Script

```
% Sampling frequency in Hz (for the data used in this demo
% i.e TIMIT database signals)
%--------------------------------------------------------------
%       Reading and removing silence part from zero.wav
%--------------------------------------------------------------
speech = read_remove('zero.wav');
Fs = 16000;
%--------------------------------------------------------------
%       Long window in time - Narrowband
%--------------------------------------------------------------
t_window_narrowband = .05; % window - time
t_overlap_narrowband = .001;

window_narrowband = t_window_narrowband*Fs;     % window samples
noverlap_narrowband = t_overlap_narrowband*Fs;
nfft_narrowband = 1024;

window = window_narrowband;
noverlap = noverlap_narrowband;
nfft = nfft_narrowband;

%--------------------------------------------------------------
%       Plotting narrowband spectrogram
%--------------------------------------------------------------
subplot(2,1,1);
spectrogram(speech,window,noverlap,nfft,Fs,'yaxis');
xlabel('Time (sec)','fontsize',16);
ylabel('Frequency (Hz)','fontsize',16);
```

```
%--------------------------------------------------------------
%       Short window in time - wideband
%--------------------------------------------------------------
t_window_wideband = .005; % window - time
t_overlap_wideband = .001;
window_wideband = t_window_wideband*Fs; % window samples
noverlap_wideband = 1; %t_overlap_wideband*Fs;
nfft_wideband = 8192;

window = window_wideband;
noverlap = noverlap_wideband;
nfft = nfft_wideband;

%--------------------------------------------------------------
%       Plotting widebandband spectogram
%--------------------------------------------------------------
subplot(2,1,2)
spectrogram(speech,window,noverlap,nfft,Fs,'yaxis');
xlabel('Time (sec)','fontsize',16);
ylabel('Frequency (Hz)','fontsize',16);
```

Plot

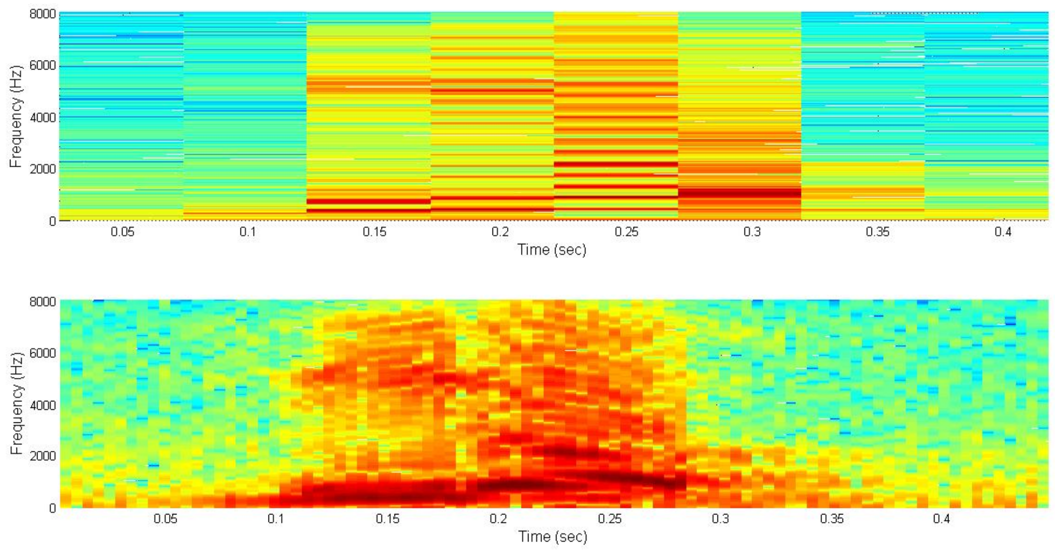


Figure 5: Comparison of Plot of Amplitude vs Frequency in Linear and Log Scale

3 LP modeling

Question 7

Implement the Levinson-Durbin algorithm for obtaining the AR coefficients for any model order 'p'.

Matlab Script

```
%----------------------------------------------------------------------
%       Asking for order and autocorrelation values
%----------------------------------------------------------------------
p = input('Enter the order of the filter : ');
r0 = input('Enter the autocorrelation value r(0) only (eg - 1.0) : ');
r = input('Enter the auto-correlation values in a row matrix
(eg - [-1.5650 1.0857 -0.7127 0.5722 -0.5007 0.5067 -0.4765 0.6040
-0.3675 0.0861] ):');
%----------------------------------------------------------------------
%       Initialize other coefficients
%----------------------------------------------------------------------
km = - r(1)/r0;
E_prev = (1 - km^2)*r0;
a_prev(1) = km;
k(1) = km;

%----------------------------------------------------------------------
%       Calculate other coefficients
%----------------------------------------------------------------------
for m = 2:p-1
    rm_1(1:m-1) = r(1:m-1);
    rm_1b(1:m-1) = r(m-1:-1:1);
    km = -((r(m) + rm_1b*a_prev')/E_prev);
    a(m) = km;
    k(m) = km;
    for j = 1:m-1
        a(j) = a_prev(j) + km*a_prev(m-j);
    end
    E_prev = E_prev*(1 - km^2);
    a_prev = a;
```

```
end

%-------------------------------------------------------------------
%       Print the coefficients
%-------------------------------------------------------------------
disp('AR coefficients are : ');disp(a);
```

Results

AR coefficients are :
-0.2039 -0.3055 0.7289 -0.8515 0.6208 -0.1295 -0.3791 0.7295 -0.6551

Question 8

Write a function that accepts the LP coefficients and outputs the LAR and LSF coefficients.

Matlab Script

```
a = input('Enter the values of LPS coefficients a interms of row matrix : ');
function [lsf,lar] = qs8(a)
    atemp = a(2:end);
    p1 = length(atemp);
    rf = zeros(p1,1);
    for k = p1:-1:1
        rf(k) = atemp(k);
        for i = 1:1:k-1
            atemp(i) = (atemp(i) - rf(k)*atemp(k-i))/(1-rf(k)^2);
        end
    end
    lar = -2*atanh(-rf);
    if a(1) ~= 1.0,
       a = a./a(1);
    end;

    p  = length(a)-1;  % The leading one in the polynomial is not used
    a1 = [a 0];
    a2 = a1(end:-1:1);
    P1 = a1-a2;        % Difference filter
    Q1 = a1+a2;        % Sum Filter
    if rem(p,2),  % Odd order
     P = deconv(P1,[1 0 -1]);
     Q = Q1;
    else          % Even order
     P = deconv(P1,[1 -1]);
     Q = deconv(Q1,[1  1]);
    end

    rP  = roots(P);
    rQ  = roots(Q);

    aP  = angle(rP(1:2:end));
    aQ  = angle(rQ(1:2:end));

    lsf = sort([aP;aQ]);
end

[lar,lsf] = LAR_LSF(k,a);
disp('lar = ');
disp(lar);
disp('lsf = ');
disp(lsf);
```

Results

Enter the values of LPS coefficients a interms of row matrix

1.0 -1.5650 1.0857 -0.7127 0.5722 -0.5007 0.5067 -0.4765 0.6040 -0.3675 0.0861

lar = 0.2702 0.3454 0.5721 0.9424 1.1450 1.4602 1.7848 2.0201 2.4429 2.6452

lsf = -3.1472 1.6455 -0.1067 0.6599 0.1861 0.5826 0.1324 0.3472 -0.4816 0.1726

Question 9

Take a short vowel segment. inverse filter it with an optimum 10th order AR system. Plot the following:
(i) short-time spectrum.
(ii) spectrum obtained by AR fitting.
(iii) inverse filtered speech signal.
(iv) spectrum of inverse-filtered speech signal.
(v) formant locations, bandwidths and pitch estimates.
Repeat the exercise for an unvoiced sound. What differences do you observe? Try experimenting with different orders.

Matlab Script

```
fs=16000;
aw=diff(n);
a320=aw(8001:8320);
plot([1:320]/16000,a320);
title('Short term time signal');
xlabel('time(ms)');
ylabel('amplitude');
%%
f_a320=freqz(a,1);
figure; plot([1:512].*8000/512,abs(f_a320));
title('Freq Resp of short-term signal');
xlabel('time(ms)');
ylabel('amplitude');

%%
ham=hamming(320);
a320ham=a320.*ham;
a320hamspec=fft(a320ham,1024);
y=abs(a320hamspec.*a320hamspec);
logy=10*log10(y);
figure;plot([1:512]*8000/512,logy(1:512));grid;
title('Short Term spectrum');
xlabel('frequency(Hz)');
ylabel('amplitude');
%%
ak6=lpc(a320,6);
lpspec6=freqz(1,ak6); %order = 6
```

```
ak10=lpc(a320,10);
lpspec10=freqz(1,ak10); %order = 10
ak14=lpc(a320,14);
lpspec14=freqz(1,ak14); % order = 14
y6 = 10*log10(abs(lpspec6.*lpspec6)) ;
y10 = 10*log10(abs(lpspec10.*lpspec10));
y14 = 10*log10(abs(lpspec14.*lpspec14));
figure;
subplot(3,1,1); plot( [1:512]*8000/512, (y6))  %in log scale
subplot(3,1,2); plot( [1:512]*8000/512, (y10))
subplot(3,1,3); plot( [1:512]*8000/512,(y14))
grid;
title('lp spectrum');
xlabel('Frequency');
ylabel('amplitude');
%%
invspec6=freqz(ak6,1);
invspec10=freqz(ak10,1);
invspec14=freqz(ak14,1);
logy6 = 10*log10(abs(invspec6.*invspec6)) ;
logy10 = 10*log10(abs(invspec10.*invspec10));
logy14 = 10*log10(abs(invspec14.*invspec14));
figure;
subplot(3,1,1); plot( [1:512]*8000/512, (logy6))  %in log scale
subplot(3,1,2); plot( [1:512]*8000/512, (logy10))
subplot(3,1,3); plot( [1:512]*8000/512,(logy14))
title('Inverse filter Spectrum');
xlabel('Frequency');
ylabel('amplitude');
%%
res6=filter(1,ak6,a320);
res10=filter(1,ak10,a320);
res14=filter(1,ak14,a320);
figure;
subplot(3,1,1); plot( [1:320]/16000, real(res6))
subplot(3,1,2); plot( [1:320]/16000, real(res10))
subplot(3,1,3); plot( [1:320]/16000,real(res14))
title('Inverse filtered speech signal.');
xlabel('time(ms)');
ylabel('amplitude');
%%
```

```
figure; plot([1:512].*8000/512,abs(freqz(res,1)));
subplot(3,1,1); plot( [1:512]*8000/512,abs(freqz(res6,1)))
subplot(3,1,2); plot( [1:512]*8000/512,abs(freqz(res10,1)))
subplot(3,1,3); plot( [1:512]*8000/512,abs(freqz(res14,1)))
title('spectrum of inverse-filtered speech signal.');
xlabel('time(ms)');
ylabel('amplitude');
%%
%%%  http://deploy.virtual-labs.ac.in/labs/cse16/exp08/indexie.html
%% formants
% get Linear prediction filter
ncoeff=2+fs/1000;           % rule of thumb for formant estimation
pol=lpc(a320,ncoeff);
 % find frequencies by root-solving
r=roots(pol);                  % find roots of polynomial a
r=r(imag(r)>0.01);           % only look for roots >0Hz up to fs/2
ffreq=sort(atan2(imag(r),real(r))*fs/(2*pi));
                             % convert to Hz and sort
for i=1:length(ffreq)
    fprintf('Formant %d Frequency %.1f\n',i,ffreq(i));
end

%% pitch
ms20=fs/50;
ms2=fs/500;
r=xcorr(a,ms20,'coeff');
r=r(ms20+1:2*ms20+1);
[rmax,tx]=max(r(ms2:ms20));
p=fs/(ms2+tx-1);
fprintf('Pitch Frequency %.1f\n',p);
```

Plot of Unvoiced Speech

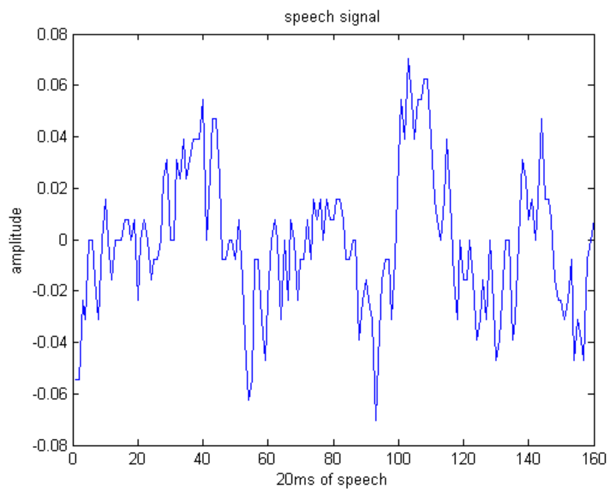


Figure 6: Speech Signal 20 ms Frame

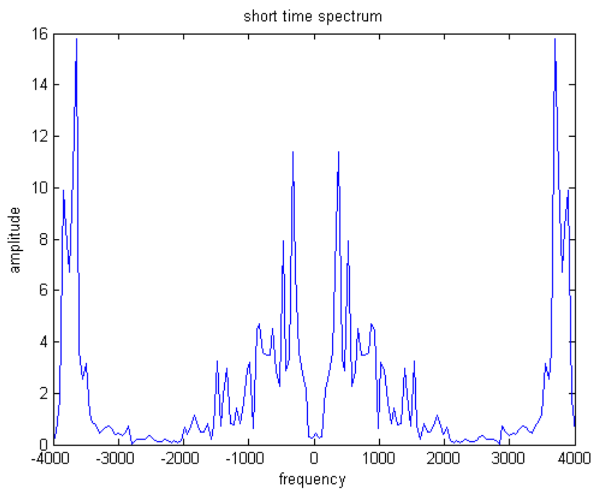


Figure 7: Spectrum of 20 ms Frame

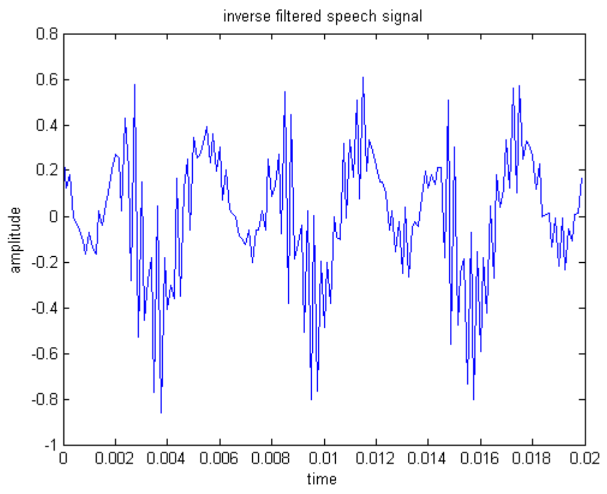


Figure 8: Inverse Filtered Speech Signal

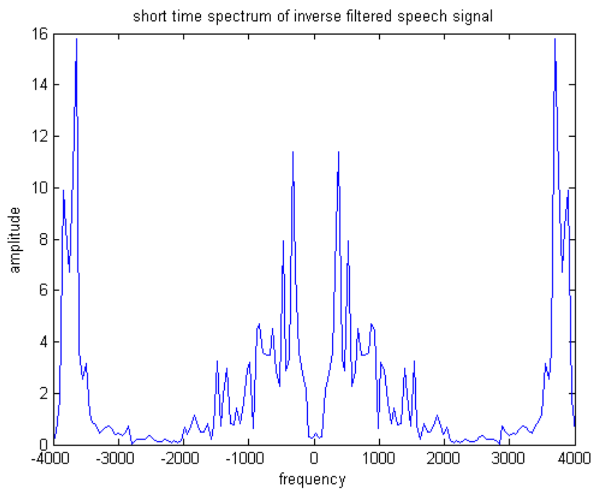


Figure 9: Short Time Spectrum of Inverse Filtered Speech Signal

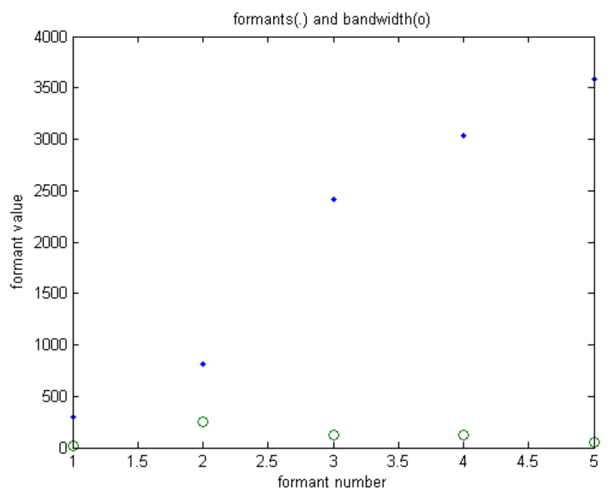


Figure 10: Formants and Bandwidth

Plot for Voiced Speech

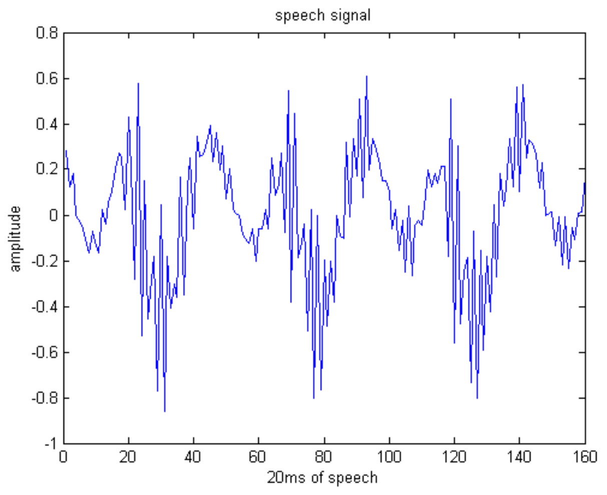


Figure 11: Speech Signal 20 ms Frame

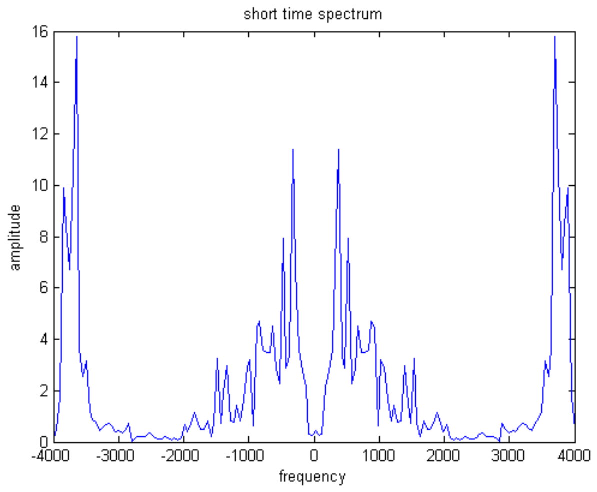


Figure 12: Spectrum of 20 ms Frame

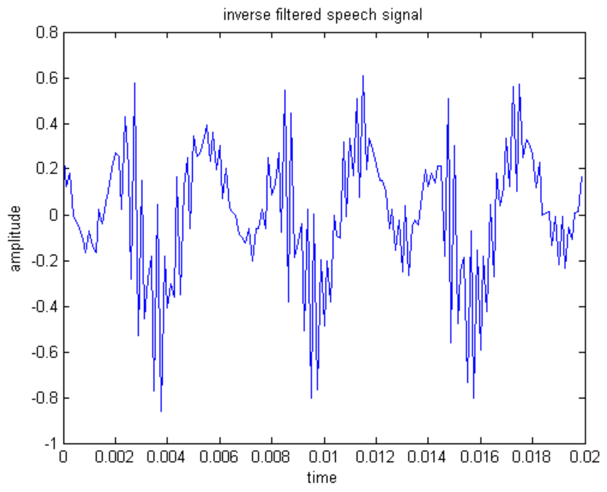


Figure 13: Inverse Filtered Speech Signal

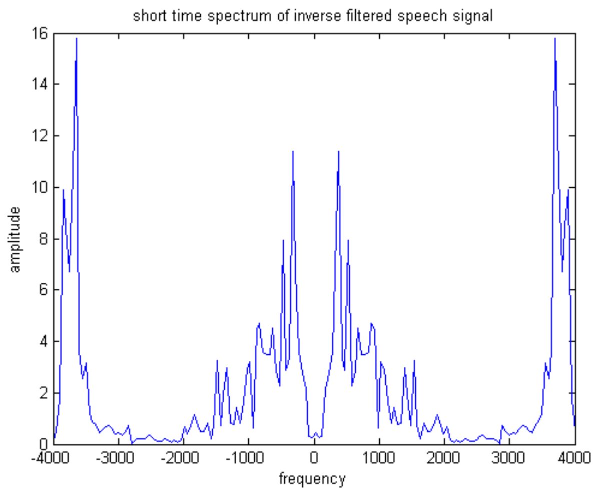


Figure 14: Short Time Spectrum of Inverse Filtered Speech Signal

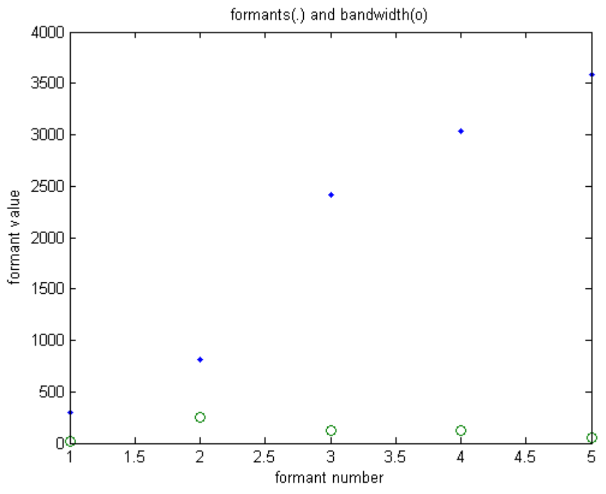


Figure 15: Formants and Bandwidth

Question 10

Take a vowel segment. Observe the following:
(i) prediction error as a function of the AR model order.
(ii) all-pole mode fit as a function of the model order.
Repeat the exercise for an unvoiced sound and comment on the results.

Matlab Script

```
%--------------------------------------------------------------
%   Prediction error as a function of AR model order with plot
%--------------------------------------------------------------
voiced = read_remove('a.wav');
[r1,lags1] = xcorr(voiced);
r1 = r1(find(lags1==0):end);
unvoiced = read_remove('non_vowel.wav');
[r2,lags2] = xcorr(unvoiced);
r2 = r2(find(lags2==0):end);

for i = 1:15
[x1,error1(i)]=levinson(r1,2*i);
[x2,error2(i)]=levinson(r2,2*i);
end
p = 2:2:30;
subplot(2,1,1); plot(p,error1)
title('Prediction error v/s order : unvoiced');
xlabel('order');
ylabel('error');
subplot(2,1,2); plot(p,error2)
title('Prediction error v/s order : voiced');
xlabel('order');
ylabel('error');
```

Plots

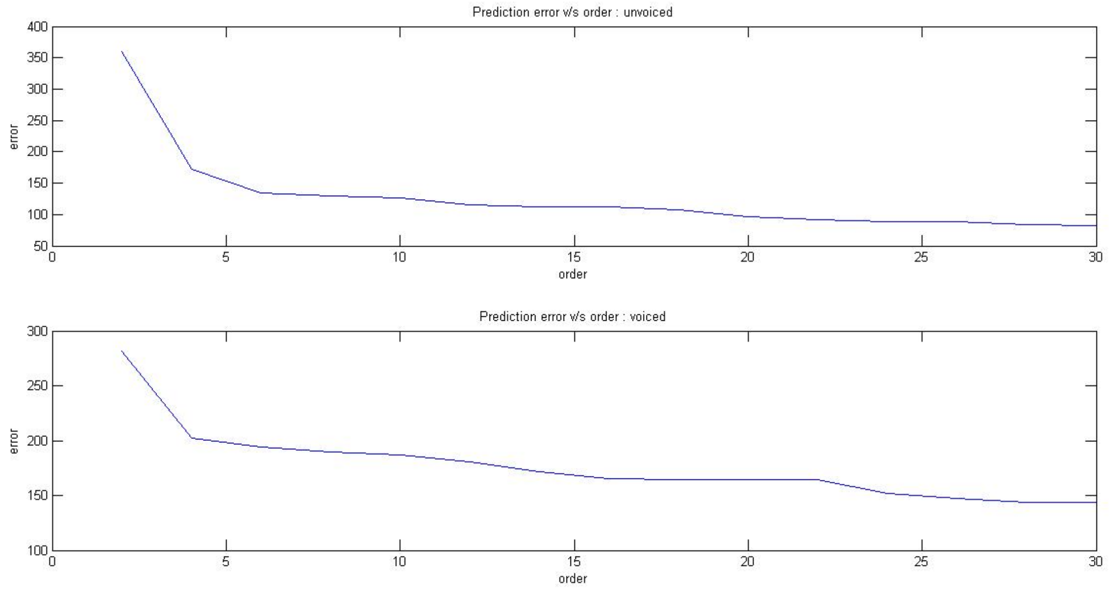


Figure 16: Comparison of Plot of Amplitude vs Frequency in Linear and Log Scale

Question 11

Take an all voiced sentence (for eg: 'We owe you a yo-yo.'). Using a frame-by-frame LP analysis, obtain the formant tracks and pitch contours. Also obtain the LP spectrogram.

Matlab Script

```
%--------------------------------------------------
% get a section of vowel
%--------------------------------------------------
[x,fs]=wavread('a.wav');
% resample to 10,000Hz (optional)
x=resample(x,10000,fs);
fs=10000;
%--------------------------------------------------
% plot waveform
%--------------------------------------------------
```

```
t=(0:length(x)-1)/fs;    % times of sampling instants
subplot(2,1,1);
plot(t,x);
legend('Waveform');
xlabel('Time (s)');
ylabel('Amplitude');
%----------------------------------------------------
% get Linear prediction filter
%----------------------------------------------------
ncoeff=2+fs/1000;     % rule of thumb for formant estimation
a=lpc(x,ncoeff);
%----------------------------------------------------
% plot frequency response
%----------------------------------------------------
[h,f]=freqz(1,a,512,fs);
subplot(2,1,2);
plot(f,20*log10(abs(h)+eps));
legend('LP Filter');
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
%----------------------------------------------------
% find frequencies by root-solving
%----------------------------------------------------
r=roots(a);         % find roots of polynomial a
r=r(imag(r)>0.01);  % only look for roots >0Hz up to fs/2
ffreq=sort(atan2(imag(r),real(r))*fs/(2*pi));
 % convert to Hz and sort
for i=1:length(ffreq)
    fprintf('Formant %d Frequency %.1f\n',i,ffreq(i));
end
%----------------------------------------------------
%     Part 2 : Pitch Contours
%----------------------------------------------------
%------------------------------------------------------------------
%       Record sound
%-----------------------------------------------------------------
clc;
clear;
 %soundClip = audiorecorder(8000,16,1);
 %recordblocking(soundClip, 5);
 %input = getaudiodata(soundClip);
```

```
%----------------------------------------------------------------
%       Remove silence
%----------------------------------------------------------------
 %x = silence_removal1(input);
 x=read_remove('twenty_sec.wav');
 N = length(x);
 [formants,bandwidth] = find_formant(x,10);
 fs = 8000;
 t = (0:N-1)/fs;
 f=0:fs/(N-1):fs;

 frame_length = 30;
 frame_overlap = 20;
 nsample = round(frame_length  * fs / 1000); % convert ms to points
 noverlap = round(frame_overlap * fs / 1000); % convert ms to points
 window   = eval(sprintf('%s(nsample)', 'hamming')); % e.g., hamming(nfft)
%Pitch detection for each frame
 pos = 1; i = 1;
 while (pos+nsample < N)
     frame = x(pos:pos+nsample-1);
     C(:,i) = cepstrumResponse(frame, fs);
     F0(i) = (pitchCepstrum(C(:,i), fs));
     pos = pos + (nsample - noverlap);
     i = i + 1;
 end
 T = (round(nsample/2):(nsample-noverlap):N-1-round(nsample/2))/fs;
  subplot(3,1,1);
  t = (0:N-1)/fs;
  plot(t, x);
  legend('Waveform');
  xlabel('Time (s)','fontsize',12);
  ylabel('Amplitude','fontsize',12);
  xlim([t(1) t(end)]);
   title('Time domain waaveform','fontsize',12);

  subplot(3,1,2);
  plot(T,F0,'.');
  legend('pitch track');
  xlabel('Time (s)','fontsize',12);
  ylabel('Frequency (Hz)','fontsize',12);
  xlim([t(1) t(end)]);
  title('Pitch Contour obtained by cepstral analysis','fontsize',12);
```

```
%------------------------------------------------------------------
%       Pitch contour by Autocorrelation method
%------------------------------------------------------------------
y=x;
Fs=8000;
sample_no=0;
mtlb_t=0;
data=y;
Frame_size = 30;
Frame_shift = 10;
max_value=max(abs(y));
y=y/max_value;
window_period=Frame_size/1000;
shift_period=Frame_shift/1000;
window_length = window_period*Fs;
sample_shift = shift_period*Fs;
sum1=0;energy=0;autocorrelation=0;
for i=1:(floor((length(y))/sample_shift)-ceil(window_length/sample_shift))
  k=1;yy=0;
  for j=(((i-1)*sample_shift)+1):(((i-1)*sample_shift)+window_length)
    yy(k)=y(j);
    k=k+1;
  end

  for l=0:(length(yy)-1)
    sum1=0;
    for u=1:(length(yy)-l)
      s=yy(u)*yy(u+l);
      sum1=sum1+s;
    end
    autocor(l+1)=sum1;

  end

  auto=autocor(21:240);
  max1=0;
  for uu=1:220
    if(auto(uu)>max1)
      max1=auto(uu);
      sample_no=uu;
    end
    if max1 == 0,
     pitch_freq(i) = 0;
     else
          pitch_period_To=(20+sample_no)*(1/fs);
          pitch_freq(i)=1/pitch_period_To;
    end
  end
  pitch_freq(i)=1/((20+sample_no)*(1/Fs));
end
rows=length(yy);
cols=(floor((length(y))/sample_shift)-ceil(window_length/sample_shift));
kkk=1/Fs:shift_period:(cols*shift_period);
subplot(3,1,3);
plot(kkk,pitch_freq,'.');
legend('pitch track');
xlabel('Time','fontsize',12)
ylabel('Cepstral Amplitude','fontsize',12);
xlim([t(1) t(end)]);
title('Pitch Contour obtained by autocorrelation of speech signal','fontsize',12);
```

Results

Formant 1 Frequency 390.2
Formant 2 Frequency 630.8
Formant 3 Frequency 2265.1
Formant 4 Frequency 2605.3
Formant 5 Frequency 3304.7
Formant 6 Frequency 3716.9

Plots

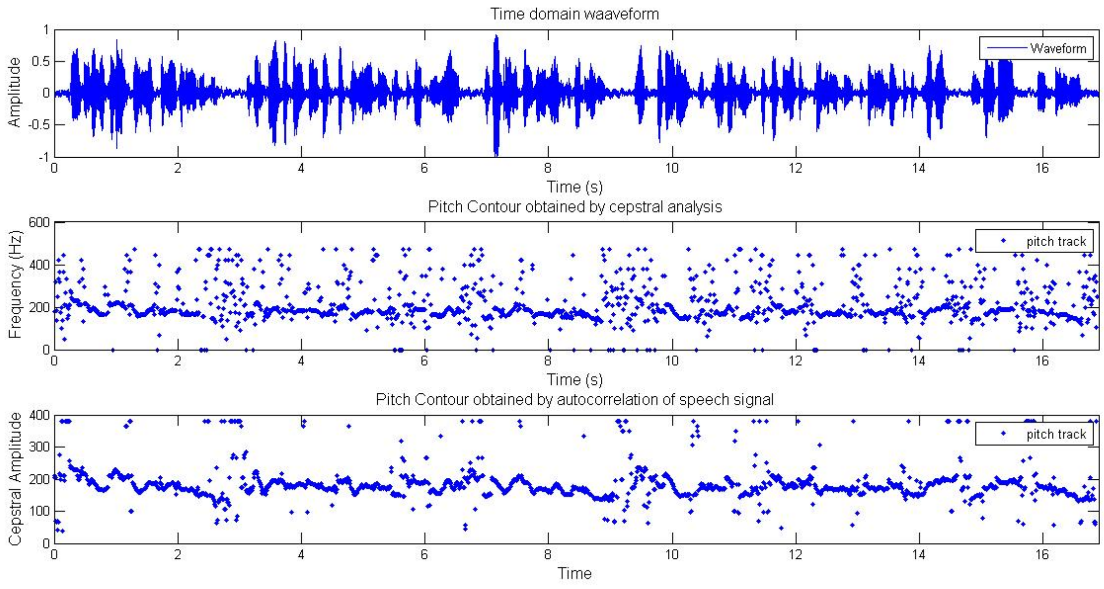


Figure 17: Plot of Pitch Contours

Question 12

Take a stable 10th order AR model. Excite it with a periodic pitch train and with a unit variance Gaussian white noise sequence. Plot the output signal and its spectrum. Experiment with different AR models, model orders and different pitch periods. Synthesize a sufficiently long signal and listen to the output signal. Does this experiment give a clue as to how certain speech sounds can be synthesized.

Matlab Script

```
clc;
clear;
%---------------------------------------------------------------------
%       reading a wav file and removing silence
%---------------------------------------------------------------------
x=read_remove('asa.wav');
%---------------------------------------------------------------------
```

```
%       Initialization
%------------------------------------------------------------------------
N=length(x);
Fs = 8000;
freq = 1000;
t = 0.02;
f=0:Fs/(N-1):Fs;

%------------------------------------------------------------------------
%       Generating an Impulse train of the required duration
%------------------------------------------------------------------------
imp = zeros(1,Fs*t);
noise = awgn(imp,0);
for i = 1:1:t*Fs,
    if mod(i,Fs/freq) == 0,
        imp(i) = 1;
    end
end
time = 0:1/Fs:(length(x)-1)/Fs;

%------------------------------------------------------------------------
%   Excitation of a 10th order AR Filter with an impulse train and AWGN
%------------------------------------------------------------------------
N = length(x);
discard = mod(N,16);
voiced = [];
unvoiced = [];
for frame_idx = 1:160:(N-discard),
    a = lpc(x(frame_idx:frame_idx+159),10);
    voiced = [voiced filter(1,a,imp)];
    unvoiced = [unvoiced filter(1,a,noise)];
    %frame_idx;
end

otpt_v_spec = abs(fft(voiced));
otpt_uv_spec = abs(fft(unvoiced));
otpt_x = abs(fft(x));

%------------------------------------------------------------------------
%       Plotting the output
%------------------------------------------------------------------------
 subplot(3,2,1); plot(time,x);%axis([0 2.5 -1 1])
```

```
title('Original signal','fontsize',12);
sound(x);
subplot(3,2,2); plot(f,otpt_x);%axis([0 2.5 -1 1])
title('FFT of Original signal','fontsize',12);
text(2000,700,'Press Blank Space','BackgroundColor',[.7 .9 .7])
pause
subplot(3,2,3); plot(time,voiced);%axis([0 2.5 -1 7])
title('Impulse excitation','fontsize',12);
sound(voiced);
subplot(3,2,4); plot(f,otpt_v_spec);%axis([0 2.5 -100 10000])
title('Impulse excitation','fontsize',12);
text(2000,25000,'Press Blank Space','BackgroundColor',[.7 .9 .7])
pause
subplot(3,2,5); plot(time,unvoiced);%axis([0 2.5 -1 1])
title('AWGN excitation','fontsize',12);
sound(unvoiced);
subplot(3,2,6); plot(f,otpt_uv_spec);%axis([0 2.5 -10 10000])
title('AWGN excitation','fontsize',12);
```

Results

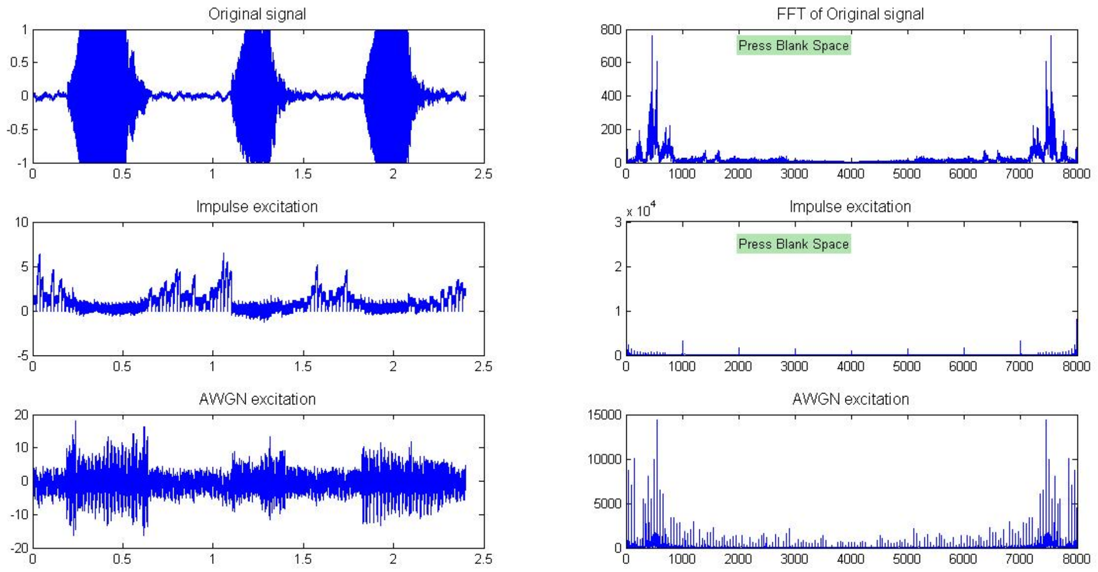


Figure 18: Comparison of Plot of Amplitude vs Frequency in Linear and Log Scale

4 Cepstrum

Question 13

Take a vowel segment. Obtain its real cepstrum. Perform lowpass and highpass liftering operations and show the corresponding log spectra.

Matlab Script

```
%-------------------------------------------------------------------------------
%time=5;
%Fs=8000;
%signal=audiorecorder;
%signal.StartFcn = 'disp(''Start speaking.'')';
%signal.StopFcn = 'disp(''End of recording.'')';
%recordblocking(signal, time);
%voice_signal = getaudiodata(signal,'double');
%audiowrite('q13.wav',voice_signal,Fs)
%[voice_signal,fs,nbits]=wavread('q13.wav'); % read file into memory *

%-------------------------------------------------------------------------------
%       Reading and removing silence from q13.wav
%-------------------------------------------------------------------------------
[voice_signal,Fs]=read_remove('q13.wav');
t=0:1/Fs:(length(voice_signal)-1)/Fs;
sound(voice_signal);

%-------------------------------------------------------------------------------
%       Plot of normal voice and its cepstrum
%-------------------------------------------------------------------------------
subplot(3,2,1);
plot(t,voice_signal,'b');
xlabel('Time')
ylabel('Amplitude')
title('Time Domain Plot')
ceps_voice_signal=rceps(voice_signal);
subplot(3,2,2);
plot(t,ceps_voice_signal,'g');
xlabel('Time')
ylabel('Cepstral Amplitude')
```

```
title('Cepstral Domain Plot')

%-------------------------------------------------------------------------------
%       Fourier Transform of voice signal and its cepstrum
%-------------------------------------------------------------------------------
n=length(voice_signal)-1;
f=0:Fs/n:Fs;
wavefft=abs(fft(voice_signal)); % perform Fourier Transform
subplot(3,2,3);
plot(f,wavefft,'b'); % plot Fourier Transform
xlabel('frequency')
ylabel('Amplitude')
title('F-Domain Plot of normal signal')
wavefft=abs(fft(ceps_voice_signal)); % perform Fourier Transform
subplot(3,2,4);
plot(f,wavefft,'g');
xlabel('frequency')
ylabel('Amplitude')
title('F-Domain Plot of cepstral signal')

%-------------------------------------------------------------------------------
% Passing normal signal and its cepstral through low pass Lifter and plotting it
%-------------------------------------------------------------------------------
frame=ceps_voice_signal(1:200);
w=hamming(200);
w=w.';
frame=frame.*w;
l=zeros(1,length(frame));
l(1:30)=1;
low_lifted_ceps_signal=real(frame.*l);
subplot(3,2,5);
hold on
plot(l,'b');
plot(low_lifted_ceps_signal,'g');
hold off
xlabel('frequency')
ylabel('Amplitude')
title('Low pass Liftering of Cepstrum')

%-------------------------------------------------------------------------------
%       Passing normal signal and its cepstrum through HPF and plotting it
%-------------------------------------------------------------------------------
```

```
h=zeros(1,length(frame));
 h(30:length(frame))=1;
 high_lifted_ceps_signal=real(frame.*h);
subplot(3,2,6);
hold on
plot(h,'b');
plot(high_lifted_ceps_signal,'g');
hold off
xlabel('frequency')
ylabel('Amplitude')
title('High pass Liftering of Cepstrum')
```

Plot

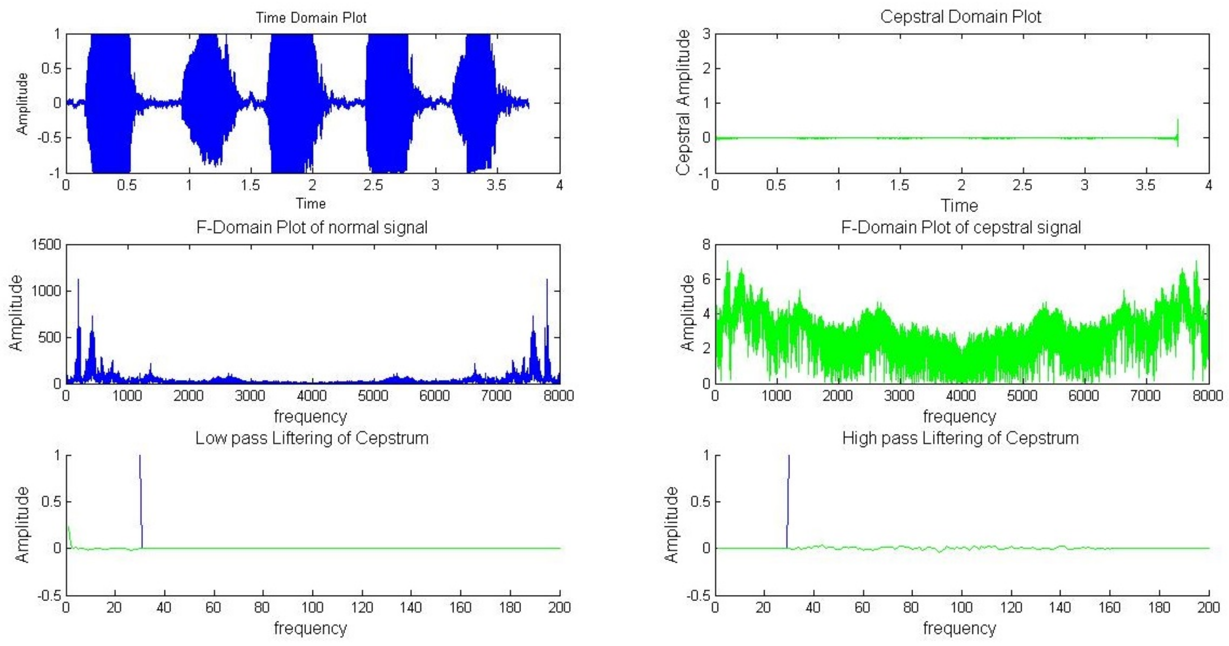


Figure 19: Real Cepstrum and Low and High pass Liftering Operation in a Log scale

Question 14

Take an all voiced sentence and obtain the pitch contour using real cepstrum. Compare this with the one obtained with LP analysis. Should they match?

Matlab Script

```
 soundClip = audiorecorder(8000,16,1);
 recordblocking(soundClip, 5);
 y_s = getaudiodata(soundClip);
 x = silence_removal1(y_s);
 N = length(x);
 fs = 8000;
 frame_length = 30;
 frame_overlap = 20;
 nsample = round(frame_length  * fs / 1000); % convert ms to points
 noverlap = round(frame_overlap * fs / 1000); % convert ms to points
 window   = eval(sprintf('%s(nsample)', 'hamming')); % e.g., hamming(nfft)
%Pitch detection for each frame
 pos = 1; i = 1;
 while (pos+nsample < N)
     frame = x(pos:pos+nsample-1);
     C(:,i) = cepstrumResponse(frame, fs);
     F0(i) = (pitchCepstrum(C(:,i), fs));
     pos = pos + (nsample - noverlap);
     i = i + 1;
 end
 T = (round(nsample/2):(nsample-noverlap):N-1-round(nsample/2))/fs;
  subplot(2,1,1);
  t = (0:N-1)/fs;
  plot(t, x);
  legend('Waveform');
  xlabel('Time (s)');
  ylabel('Amplitude');
  xlim([t(1) t(end)]);

  % plot F0 track
  subplot(2,1,2);
  plot(T,F0,'.');
  legend('pitch track');
  xlabel('Time (s)');
  ylabel('Frequency (Hz)');
```

```
xlim([t(1) t(end)]);
```

Plot

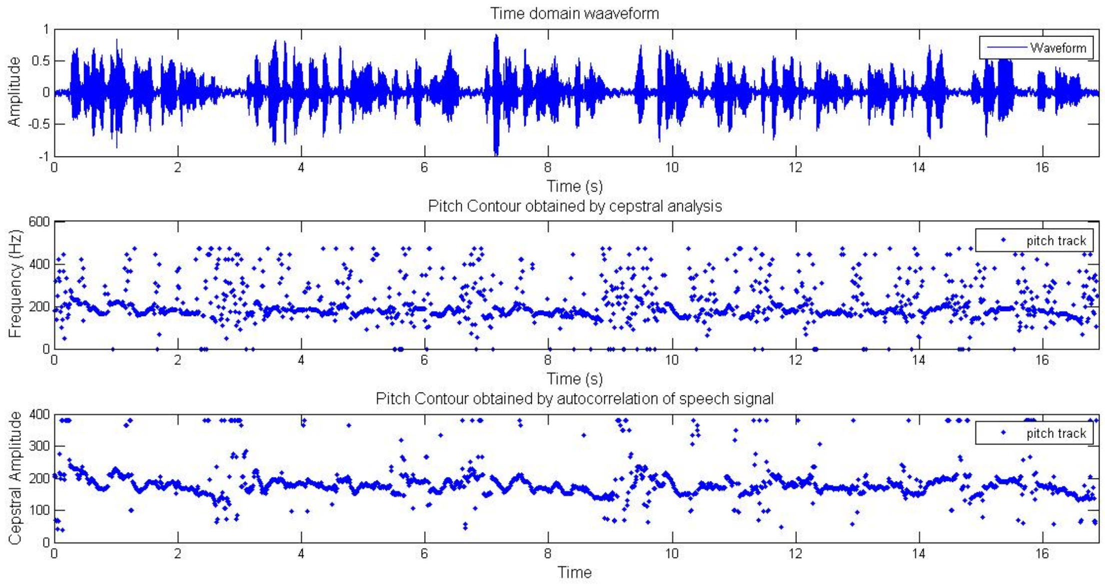


Figure 20: Plot for Pitch Contours

5 GUI

Question 15

Make a GUI with the following features. Provision to record from the microphone or select a speech/audio file from a play list. If you are recording you should be able to specify the sampling frequency and start and stop the recording

1) Provision to play the speech or audio at different sampling rates
2) Display the time domain waveform in a window, zoom in & out, select portion of signal to play
3) Display the spectrum of selected portion of time domain signal in a second window
4) Display the pitch (fundamental frequency) of voiced speech if needed. Use autocorrelation to compute the pitch
5) Display the LP spectrum and label formant frequencies
6) You may add more features which could be helpful when you are analyzing speech or audio

Matlab Script

```
function varargout = GUI2(varargin)
% GUI2 MATLAB code for GUI2.fig
%      GUI2, by itself, creates a new GUI2 or raises the existing
%      singleton*.
%
%      H = GUI2 returns the handle to a new GUI2 or the handle to
%      the existing singleton*.
%
%      GUI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI2.M with the given input arguments.
%
%      GUI2('Property','Value',...) creates a new GUI2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help GUI2

% Last Modified by GUIDE v2.5 26-Oct-2014 22:29:08

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI2_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI2 is made visible.
function GUI2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI2 (see VARARGIN)
handles.audio = audiorecorder;
%handles.a=0;
% Choose default command line output for GUI2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = GUI2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%handles.a = 0;
%[filename pathname] = uigetfile({'*.wav','File Selector'});
%handles.a = audioread(filename);
%fullpathname = strcat(pathname,filename);
%axes(handles.axes1);
%plot(handles.a);
%axes(handles.axes2);
%specgram(handles.a, 1024, 8000);
%title('Spectrogram of Original Signal'); %guidata(hObject,handles);
[FileName,PathName] = uigetfile({'*.wav'},'Load Wav File');
    [x,fs] = wavread([PathName '/' FileName]);
    handles.x = x;
    handles.fs = fs;
    axes(handles.axes1);
    time = 0:1/fs:(length(handles.x)-1)/fs;
    plot(time,handles.x);
    title('Original Signal');
    axes(handles.axes2);
    specgram(handles.x, 1024, handles.fs);
    title('Spectrogram of Original Signal');
guidata(hObject,handles);



% --- Executes on button press in pushbutton2.
```

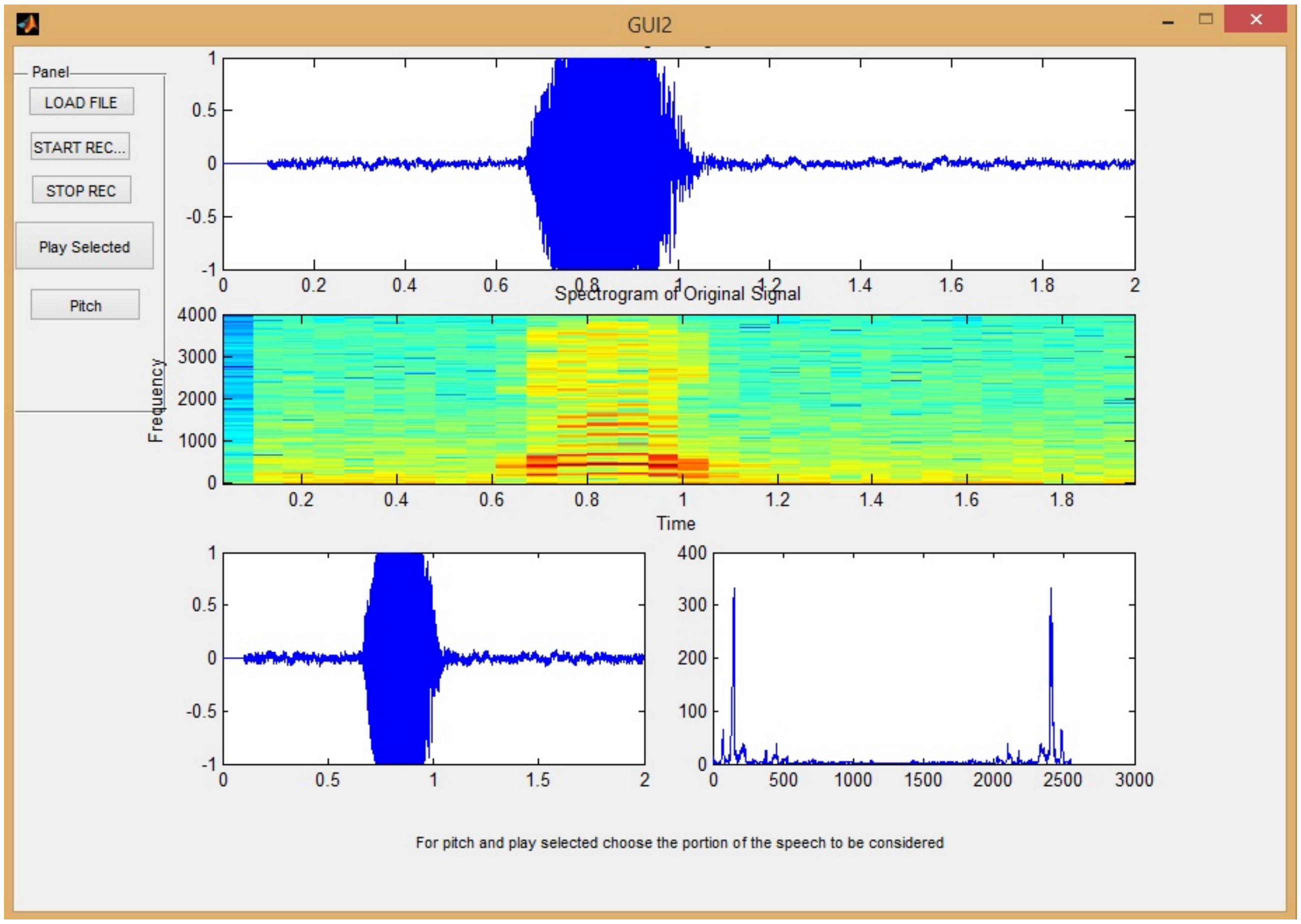
```
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

record(handles.audio);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

stop(handles.audio);
a = getaudiodata(handles.audio);
axes(handles.axes1);
plot(a);


% --- Executes on button press in pushbutton5.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fs=8000;
axes(handles.axes3);
time = 0:1/fs:(length(handles.x)-1)/fs;
plot(time,handles.x); RECT = (ginput(2));
xmin = RECT(1);
xmax = RECT(2);
% zoom in on the time data
axis([xmin xmax -0.5 0.5]);
s=handles.x(xmin:xmax);
title('Zoomed Signal');
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
%handles.FSQ = (get(hObject,'Value'));

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fs=8000;
axes(handles.axes3);
time = 0:1/fs:(length(handles.x)-1)/fs;
plot(time,handles.x);
RECT = (ginput(2));
xmin = RECT(1);
xmax = RECT(2);
% zoom in on the time data
axis([xmin xmax -0.5 0.5]);
s=handles.x(xmin*fs:xmax*fs);
sound(s, fs);
axes(handles.axes4);
```

```
plot(abs(fft(s)));


% --- Executes on button press in pushbutton7. function pushbutton7_Callback(hObject, ev
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fs=8000;
axes(handles.axes3);
time = 0:1/fs:(length(handles.x)-1)/fs;
plot(time,handles.x);
RECT = (ginput(2));
xmin = RECT(1);
xmax = RECT(2);
% zoom in on the time data
axis([xmin xmax -0.5 0.5]);
x=handles.x(xmin*fs:xmax*fs);
frame_length = 8*20;
m = length(x);
no_frames = m / frame_length;
count = 0;count1 = 0;
for k = 1 : no_frames
    frame = x((k-1)*frame_length + 1 : k * frame_length);
    max_val = max(frame);
    min_val = min(frame);

    if(max_val > 0.3 && min_val < -0.3)
        count1 = count1 + 1;
        voiced_speech((count1-1)*frame_length + 1 : count1 * frame_length) = frame;
    else
        count = count + 1;
        unvoiced_speech((count-1)*frame_length + 1 : count * frame_length) = frame;
    end
end
y = voiced_speech;
[autocor,lags] = xcorr(y);
autocor = autocor(find(lags==0):end);
auto=autocor(21:160);

max1=0;
  for uu=1:140
    if(auto(uu)>max1)
```

```
      max1=auto(uu);
      sample_no=uu;
    end
  end

  pitch_period_To=(20+sample_no)*(1/fs);
  pitch_freq_Fo=1/pitch_period_To;
  disp('OBSERVATIONS');
  disp('Pitch :')
  disp(pitch_freq_Fo);
```

Results



GUI2
Panel
LOAD FILE
START REC...
STOP REC
Play Selected
Pitch
Spectrogram of Original Signal
Frequency
Time
For pitch and play selected choose the portion of the speech to be considered

Figure 21: Graphical User Interface

6 Function Used

Detect Voiced

```
function [segments, fs] = detectVoiced(wavFileName,t
% This file is by the name detect_voiced.m and it is used for read_remove
% function [segments, fs] = detectVoiced(wavFileName)
% Theodoros Giannakopoulos
% http://www.di.uoa.gr/~tyiannak
% (c) 2010
%
% This function implements a simple voice detector. The algorithm is
% described in more detail, in the readme.pdf file
%
% ARGUMENTS:
%  - wavFileName: the path of the wav file to be analyzed
%  - t: if provided, the detected voiced segments are played and some
%  intermediate results are also ploted
%
% RETURNS:
%  - segments: a cell array of M elements. M is the total number of
%  detected segments. Each element of the cell array is a vector of audio
%  samples of the respective segment.
%  - fs: the sampling frequency of the audio signal
%
% EXECUTION EXAMPLE:
%
% [segments, fs] = detectVoiced('example.wav',1);
%
% Check if the given wav file exists:
fp = fopen(wavFileName, 'rb');
if (fp<0)
fprintf('The file %s has not been found!\n', wavFileName);
return;
end
fclose(fp);

% Check if .wav extension exists:
if  (strcmpi(wavFileName(end-3:end),'.wav'))
    % read the wav file name:
```

```
    [x,fs] = audioread(wavFileName);
else
    fprintf('Unknown file type!\n');
    return;
end
% Convert mono to stereo
if (size(x, 2)==2)
x = mean(x')';
end
% Window length and step (in seconds):
win = 0.050;
step = 0.050;

%-----------------------------------------
%  THRESHOLD ESTIMATION
%-----------------------------------------

Weight = 5; % used in the threshold estimation method

% Compute short-time energy and spectral centroid of the signal:
Eor = ShortTimeEnergy(x, win*fs, step*fs);
Cor = SpectralCentroid(x, win*fs, step*fs, fs);

% Apply median filtering in the feature sequences (twice), using 5 windows:
% (i.e., 250 mseconds)
E = medfilt1(Eor, 5); E = medfilt1(E, 5);
C = medfilt1(Cor, 5); C = medfilt1(C, 5);
% Get the average values of the smoothed feature sequences:
E_mean = mean(E);
Z_mean = mean(C);
% Find energy threshold:
[HistE, X_E] = hist(E, round(length(E) / 10));  % histogram computation
[MaximaE, countMaximaE] = findMaxima(HistE, 3);
% find the local maxima of the histogram
if (size(MaximaE,2)>=2)
% if at least two local maxima have been found in the histogram:
    T_E = (Weight*X_E(MaximaE(1,1))+X_E(MaximaE(1,2))) / (Weight+1);
    % ... then compute the threshold as the weighted average between
     the two first histogram's local maxima.
else
    T_E = E_mean / 2;
end
```

```
% Find spectral centroid threshold:
[HistC, X_C] = hist(C, round(length(C) / 10));
[MaximaC, countMaximaC] = findMaxima(HistC, 3);
if (size(MaximaC,2)>=2)
    T_C = (Weight*X_C(MaximaC(1,1))+X_C(MaximaC(1,2))) / (Weight+1);
else
    T_C = Z_mean / 2;
end

% Thresholding:
Flags1 = (E>=T_E);
Flags2 = (C>=T_C);
flags = Flags1 & Flags2;

if (nargin==2) % plot results:
clf;
subplot(3,1,1); plot(Eor, 'g'); hold on; plot(E, 'c');
legend({'Short time energy (original)', 'Short time energy (filtered)'});
    L = line([0 length(E)],[T_E T_E]); set(L,'Color',[0 0 0]);
    set(L, 'LineWidth', 2);
    axis([0 length(Eor) min(Eor) max(Eor)]);

    subplot(3,1,2); plot(Cor, 'g'); hold on; plot(C, 'c');
    legend({'Spectral Centroid (original)', 'Spectral Centroid (filtered)'});
L = line([0 length(C)],[T_C T_C]); set(L,'Color',[0 0 0]);
set(L, 'LineWidth', 2);
    axis([0 length(Cor) min(Cor) max(Cor)]);
end

%-------------------------------------------------------
%  SPEECH SEGMENTS DETECTION
%-------------------------------------------------------
count = 1;
WIN = 5;
Limits = [];
while (count < length(flags)) % while there are windows to be processed:
% initilize:
curX = [];
countTemp = 1;
% while flags=1:
while ((flags(count)==1) && (count < length(flags)))
```

```
if (countTemp==1) % if this is the first of the current speech segment:
Limit1 = round((count-WIN)*step*fs)+1; % set start limit:
if (Limit1<1) Limit1 = 1; end
end
count = count + 1;  % increase overall counter
countTemp = countTemp + 1; % increase counter of the CURRENT speech segment
end

if (countTemp>1) % if at least one segment has been found in the current loop:
Limit2 = round((count+WIN)*step*fs); % set end counter
if (Limit2>length(x))
            Limit2 = length(x);
        end

        Limits(end+1, 1) = Limit1;
        Limits(end,   2) = Limit2;
    end
count = count + 1; % increase overall counter
end

%---------------------------------------------
% POST - PROCESS      %
%---------------------------------------------

% A. MERGE OVERLAPPING SEGMENTS:
RUN = 1;
while (RUN==1)
    RUN = 0;
    for (i=1:size(Limits,1)-1) % for each segment
        if (Limits(i,2)>=Limits(i+1,1))
            RUN = 1;
            Limits(i,2) = Limits(i+1,2);
            Limits(i+1,:) = [];
            break;
        end
    end
end

% B. Get final segments:
segments = {};
for (i=1:size(Limits,1))
    segments{end+1} = x(Limits(i,1):Limits(i,2));
```

```
end

if (nargin==2)
    subplot(3,1,3);
    % Plot results and play segments:
    time = 0:1/fs:(length(x)-1) / fs;
    for (i=1:length(segments))
        hold off;
        P1 = plot(time, x); set(P1, 'Color', [0.7 0.7 0.7]);
        hold on;
        for (j=1:length(segments))
            if (i~=j)
                timeTemp = Limits(j,1)/fs:1/fs:Limits(j,2)/fs;
                P = plot(timeTemp, segments{j});
                set(P, 'Color', [0.4 0.1 0.1]);
            end
        end
        timeTemp = Limits(i,1)/fs:1/fs:Limits(i,2)/fs;
        P = plot(timeTemp, segments{i});
        set(P, 'Color', [0.9 0.0 0.0]);
        axis([0 time(end) min(x) max(x)]);
        sound(segments{i}, fs);
        clc;
        fprintf('Playing segment %d of %d. Press any key to continue...',
         i, length(segments));
        pause
    end
    clc
    hold off;
    P1 = plot(time, x); set(P1, 'Color', [0.7 0.7 0.7]);
    hold on;
    for (i=1:length(segments))
        for (j=1:length(segments))
            if (i~=j)
                timeTemp = Limits(j,1)/fs:1/fs:Limits(j,2)/fs;
                P = plot(timeTemp, segments{j});
                set(P, 'Color', [0.4 0.1 0.1]);
            end
        end
        axis([0 time(end) min(x) max(x)]);
    end
end
```

Read Remove

```
function [removed_signal, fs] = read_remove(wavFileName)
[segments, fs] = detectVoiced(wavFileName);
m=length(segments);
removed_signal=0;
x=0;
hold off
for j=1:m
    y=length(segments{j});
    removed_signal(x+1:x+y)=segments{j};
    x=length(removed_signal);
end
 %plot(removed_signal);
 %sound(removed_signal);
end
```

Cepstrum Response

```
function c = cepstrumResponse(x, fs)
     N = length(x);
     x = x(:); % assure column vector
     w = hamming(N);
     x = x(:) .* w(:);
     y = fft(x, N);
     c = ifft(log(abs(y)));
end
```

Find Maxima

```
function [Maxima, countMaxima] = findMaxima(f, step)
%
% MAXIMA ESTIMATION
%
% function [Maxima, countMaxima] = findMaxima(f, step);
%
% This function estimates the local maxima of a sequence
%
% ARGUMENTS:
% f: the input sequence
% step: the size of the "search" window
%
% RETURN:
% Maxima: [2xcountMaxima] matrix containing:
%         1. the maxima's indeces
%         2. tha maxima's values
% countMaxima: the number of maxima
%
%
% STEP 1: find maxima:
%
countMaxima = 0;
for (i=1:length(f)-step-1) % for each element of the sequence:
    if (i>step)
        if (( mean(f(i-step:i-1))< f(i)) && ( mean(f(i+1:i+step))< f(i)))
            % IF the current element is larger than its neighbors (2*step window)
            % --> keep maximum:
            countMaxima = countMaxima + 1;
            Maxima(1,countMaxima) = i;
            Maxima(2,countMaxima) = f(i);
        end
    else
        if (( mean(f(1:i))<= f(i)) && ( mean(f(i+1:i+step))< f(i)))
            % IF the current element is larger than its neighbors (2*step window)
            % --> keep maximum:
            countMaxima = countMaxima + 1;
            Maxima(1,countMaxima) = i;
            Maxima(2,countMaxima) = f(i);
        end
    end
end
%
% STEP 2: post process maxima:
%
MaximaNew = [];
countNewMaxima = 0;
i = 0;
while (i<countMaxima)
    % get current maximum:
    i = i + 1;
    curMaxima = Maxima(1,i);
    curMavVal = Maxima(2,i);

    tempMax = Maxima(1,i);
    tempVals = Maxima(2,i);

    % search for "neighbourh maxima":
    while ((i<countMaxima) && ( Maxima(1,i+1) - tempMax(end) < step / 2))
        i = i + 1;
        tempMax(end+1) = Maxima(1,i);
        tempVals(end+1) = Maxima(2,i);
    end
    % find the maximum value and index from the tempVals array:
    %MI = findCentroid(tempMax, tempVals); MM = tempVals(MI);

    [MM, MI] = max(tempVals);

    if (MM>0.02*mean(f)) % if the current maximum is "large" enough:
        countNewMaxima = countNewMaxima + 1;   % add maxima
        % keep the maximum of all maxima in the region:
        MaximaNew(1,countNewMaxima) = tempMax(MI);
        MaximaNew(2,countNewMaxima) = f(MaximaNew(1,countNewMaxima));
    end
    tempMax = [];
    tempVals = [];
end

Maxima = MaximaNew;
countMaxima = countNewMaxima;
```

LSF LAR

```
function [lsf,lar] = lsf_lar(a)
    atemp = a(2:end);
    p1 = length(atemp);
    rf = zeros(p1,1);
    for k = p1:-1:1
        rf(k) = atemp(k);
        for i = 1:1:k-1
            atemp(i) = (atemp(i) - rf(k)*atemp(k-i))/(1-rf(k)^2);
        end
    end
    lar = -2*atanh(-rf);
    if a(1) ~= 1.0,
       a = a./a(1);
    end;

    p  = length(a)-1;  % The leading one in the polynomial is not used
    a1 = [a 0];
    a2 = a1(end:-1:1);
    P1 = a1-a2;        % Difference filter
    Q1 = a1+a2;        % Sum Filter
    if rem(p,2),  % Odd order
     P = deconv(P1,[1 0 -1]);
     Q = Q1;
    else          % Even order
     P = deconv(P1,[1 -1]);
     Q = deconv(Q1,[1  1]);
    end

    rP  = roots(P);
    rQ  = roots(Q);

    aP  = angle(rP(1:2:end));
    aQ  = angle(rQ(1:2:end));

    lsf = sort([aP;aQ]);
end
```

Short Time Energy

```
function E = ShortTimeEnergy(signal, windowLength,step);
signal = signal / max(max(signal));
curPos = 1;
L = length(signal);
numOfFrames = floor((L-windowLength)/step) + 1;
%H = hamming(windowLength);
E = zeros(numOfFrames,1);
for (i=1:numOfFrames)
    window = (signal(curPos:curPos+windowLength-1));
    E(i) = (1/(windowLength)) * sum(abs(window.^2));
    curPos = curPos + step;
end
```

Plot Spectrum

```
function  plot_spectrum(x)
N = length(x);
fs = 8000;
f = -fs/2 : fs/(N-1) : fs/2;
plot(f,abs(fft(x)));
end
```

Pitch Cepstrum

```
function f0 = pitchCepstrum(c, fs)
    ms2=floor(fs*0.002); % 2ms
    ms20=floor(fs*0.02); % 20ms
    [maxi,idx]=max(abs(c(ms2:ms20)));
    if idx == 1,
       f0 = 0;
    else
       f0 = fs/(ms2+idx-1);
end
```

Spectral Centroid

```
function C = SpectralCentroid(signal,windowLength, step, fs)

% function C = SpectralCentroid(signal,windowLength, step, fs)
%
% This function computes the spectral centroid feature of an audio signal
% ARGUMENTS:
%  - signal: the audio samples
%  - windowLength: the length of the window analysis (in number of samples)
%  - step: the step of the window analysis (in number of samples)
%  - fs: the sampling frequency
%
% RETURNS:
%  - C: the sequence of the spectral centroid feature
%

signal = signal / max(abs(signal));
curPos = 1;
L = length(signal);
numOfFrames = floor((L-windowLength)/step) + 1;
H = hamming(windowLength);
m = ((fs/(2*windowLength))*[1:windowLength])';
C = zeros(numOfFrames,1);
for (i=1:numOfFrames)
    window = H.*(signal(curPos:curPos+windowLength-1));
    FFT = (abs(fft(window,2*windowLength)));
    FFT = FFT(1:windowLength);
    FFT = FFT / max(FFT);
    C(i) = sum(m.*FFT)/sum(FFT);
    if (sum(window.^2)<0.010)
        C(i) = 0.0;
    end
    curPos = curPos + step;
end
C = C / (fs/2);
```

Voice Unvoiced

```
%This is function is to separate out the voiced and the unvoiced sounds
function [voiced,unvoiced] = voice_unvoice(x)
voiced=0;
unvoiced=0;
m = length(x);
frame_length = m*20/1000;
no_frames = m / frame_length;
count = 0;count1 = 0;
for k = 1 : no_frames
    frame = x((k-1)*frame_length + 1 : k * frame_length);
    max_val = max(frame);
    min_val = min(frame);

    if(max_val > 0.3 && min_val < -0.3)
        count1 = count1 + 1;
        voiced((count1-1)*frame_length + 1 : count1 * frame_length) = frame;
    else
        count = count + 1;
        unvoiced((count-1)*frame_length + 1 : count * frame_length) = frame;
    end
end
```

References

[1] http://www.di.uoa.gr/ tyiannak

[2] http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/doc/voicebox/bark2frq.html

[3] http://iitg.vlab.co.in/?sub=59&brch=164&sim=1012&cnt=2

[4] http://web.eecs.utk.edu/ qi/teaching/ece310s01/project/pitch.m

[5] http://matlabsproj.blogspot.in/2012/05/voice-conversion-in-matlab.html

[6] https://catalog.ldc.upenn.edu/LDC93S1

[7] http://www.phon.ucl.ac.uk/courses/spsci/matlab/lect10.html