```matlab
% % -------------------------------------------
% Name : Manan Vyas
% USCID: 7483-8632-00
% Email: mvyas@usc.edu
% EE519 : Speech Recognition : Final Exam : P3
% % --------------------------------------------

% Setup
clc;
clear all;
close all;

Fs = 10000;   % Hz - Sampling frq
load('final2014_p3.mat'); % we have normalized Speech signal here
n = length(speech);
% Apply a 25ms Hamming window ...
% 10k samples/sec => 25msec = 250 samples => Length of Hamming window
wLen = 250;
window = hamming(wLen);
% Take first 250 samples of the Speech signal
Speech = speech(1:250);
wSpeech = Speech.*window' ; % Windowed Speech

% Computing Cepstrum
wSpeechFFT = fft(wSpeech,1024); % 1024 point FFT of the windowed Speech
wSpeechFFT_log = log(abs(wSpeechFFT)); % Log Mag spectrum of the FFT
wSpeechFFT_cepstrum = real(ifft(wSpeechFFT_log,1024)); % Defintion of Cepstrum

% Part (A) All plots
figure;
subplot(3,1,1); plot(Speech); title('250 samples of original Speech sample');
xlabel('n ->'); ylabel('Speech');

subplot(3,1,2); plot(wSpeech);
title('Windowed version of above Speech signal using Hamming Window');
xlabel('n ->'); ylabel('s[n].w(hamming)[n]');

subplot(3,1,2); plot(wSpeech);
title('Windowed version of above Speech signal using Hamming Window');
xlabel('n ->'); ylabel('s[n].w(hamming)[n]');

subplot(3,1,3); plot(wSpeechFFT_cepstrum(1:200)); axis([1 200 -1 1.5]);
title('Real cepstrum');
xlabel('Quefrency ->'); ylabel('Real ceps amplitude');

% Part (B) : Pitch Estimation
%Liftering

wSpeechFFT_cepstrum_coEff = wSpeechFFT_cepstrum(1:length(wSpeechFFT_cepstrum));
L = zeros(1,length(wSpeechFFT_cepstrum_coEff));
% Liftering Window
L(20:140) = 1;
% Low time lifted cepstrum
yOp = real(wSpeechFFT_cepstrum_coEff.*L);

%Finding peak in lifted cepstrum
[peak_val,peak_loc] = max(yOp);
pitch_period = peak_loc;
pitch_frq = (1/pitch_period)*Fs;

% Part (C) : Uniform Quantizer
pitchMin = 50;
pitchMax = 300;
B = 7 ; % Bit precision size
```

```
% The quantized value of the pitch frequency using a 7bit uniform quantizer
quantPitchFrqValue = quant(pitch_frq,B,pitchMin,pitchMax)
quantValue = quant(pitch_frq,B,pitchMin,pitchMax);

% Part (D): Non uniform quantizer
% take the initial 28 co-efficients of the cepstrum co-eff
cepstrumCoEff28 = wSpeechFFT_cepstrum(1:28);

% Unifromly quantize these
for i=1:1:28
    cepstrumCoEff28_UniQ(i) = quant(cepstrumCoEff28(i), 7,
min(cepstrumCoEff28),max(cepstrumCoEff28));
end
figure
subplot(2,1,1)
plot(cepstrumCoEff28,'--r');
hold on
stem(cepstrumCoEff28_UniQ);
title('Uniformly Quantized initial 28 cepstrum co-effs');
xlabel('n ->'); ylabel('c[n]'); axis([1 28 min(cepstrumCoEff28_UniQ)
max(cepstrumCoEff28_UniQ)]);
legend('Original Cepstrum Coeffs','Uniformly Quantized Cepstrum Coeff(7bit)');
hold off

% Non Uniform Quantization
cepstrumCoEff28_NonUniQ = nonUniformQuant(cepstrumCoEff28);
subplot(2,1,2)
plot(cepstrumCoEff28,'--r');
hold on
stem(cepstrumCoEff28_NonUniQ);
title('Non-Uniformly Quantized initial 28 cepstrum co-effs');
xlabel('n ->'); ylabel('_c[n]'); axis([1 28 min(cepstrumCoEff28_NonUniQ)
max(cepstrumCoEff28_NonUniQ)]);
legend('Original Cepstrum Coeffs','Non-Uniformly Quantized Cepstrum Coeff');
hold off

% Part (E): Minimum phase reconstruction
% Right sided Cepstral Lifter
Lifter(1) = 1;
Lifter(2:28) = 2;
% Lifter the non uniformly quantized values
cepstrumCoEff28_UniQ_Liftered = cepstrumCoEff28_UniQ.*Lifter;
minimumPhaseRecostruction = fft(cepstrumCoEff28_UniQ_Liftered,1024);

% Part (F) : Sampled log magnitude and phase and signal reconstruction
figure
% We need to map the x-axis value to reflect frequency in Hz.
% To do that all you need to remember that the length of the
% FFT corresponds to the sampling rate Fs for continuous frequencies and
% corresponds to 2? for discrete frequency. Therefore
% to get the positive and negative frequencies
scale = -(1024/2):(1024/2-1);
subplot (2,1,1)
plot(scale*Fs/1024,fftshift(20*log10(abs(minimumPhaseRecostruction)))); % to get the frquency
scale in Hz
title('Log Magnitude plot of minimum phase recosntruction');xlabel('f(Hz)-
>');ylabel('20*log|fft(Ceps,1024)| db scale');
subplot(2,1,2);
plot(scale*Fs/1024,fftshift((angle(minimumPhaseRecostruction))));
title('Phase plot of minimum phase recosntruction');xlabel('f(Hz)-
>');ylabel('<fft(Ceps,1024)');
% Now sample the log magnitude and phase values
toBeSampled = log(abs(minimumPhaseRecostruction));
diff = [5000,-ones(1,100)];
```

```
index = 1;
newHarms = 1;
for i=1:1:512 %because the spectrum FFT is symmetrical
    index = index + 1;
    newfreq = i*Fs/1024;
    tempdiff = abs(quantValue - (i*Fs/(1024)));
    diff(index) = quantValue - (i*Fs/(1024));
    if(abs(diff(index)) > abs(diff(index-1)))
%         sampledLogMag(i) = 20*log(abs(minimumPhaseRecostruction((quantValue+diff(index-
1))*1024/(Fs*(i)))));
        sampledLogMag(newHarms) = toBeSampled(i) ;%log(abs(minimumPhaseRecostruction(i)));
        sampledPhase(newHarms) = angle(minimumPhaseRecostruction(i));
        selectedHarmonicFrequencies(newHarms) = newfreq;
        selectedHarmonicFrequenciesIndex(newHarms) = i;
        quantValue = quantValue + quantPitchFrqValue;
        newHarms = newHarms + 1;
        diff = [5000,-ones(1,100)];
%          i = 1;
        if(i > 1024)
           break;
        end
        index = 1;
    end
end
selectedHarmonicFrequencies
selectedHarmonicFrequenciesIndex
sampledLogMag
sampledPhase

% Part (G) : Resynthesis
reconLogMag = exp(sampledLogMag);
reconPhase = exp(sampledPhase);
reconstructedSpeech = sinewave(reconLogMag,selectedHarmonicFrequencies,reconPhase,1000);
% Normalize
reconstructedSpeech = reconstructedSpeech./max(reconstructedSpeech) ;
figure
subplot(2,1,1)
plot(reconstructedSpeech); title('Reconstructed Speech'); xlabel('n ->'); ylabel('y[n]');
subplot(2,1,2);
plot(speech);title('Original Speech'); xlabel('n ->'); ylabel('x[n]');
% MSE Calculation
sqDiff = (double(reconstructedSpeech) - double(speech)).^2;
MSE = sum(sqDiff)/(length(reconstructedSpeech))
```

```
function [ quantValue ] = quant(inValue,B, minVal, maxVal)

% 7 bit quantizer on the input value 'inValue' by B bits
% minVal and maxVal denotes the minimum and maximum values of the input
% data :: quantValue
% Reference : http://www.mathworks.com/help/matlab/ref/bsxfun.html
% Reference : https://courses.engr.illinois.edu/ece420/lab3/prelab3.html

% Normalize such that the maxValue of the input comes in [-1 to 1] range
% divide by the smallest power of two such that the resulting absolute value of the largest
number
% is less than or equal to one
% This is an easy but fairly reasonable approximation of how numbers
% outside the range of -1 to 1 are actually handled on the DSP.
N = nextpow2(maxVal-minVal);
inValue = inValue/(2^N);
% Next, quantize to B bits of precision by first multiplying them by 2^{B} rounding to the
```

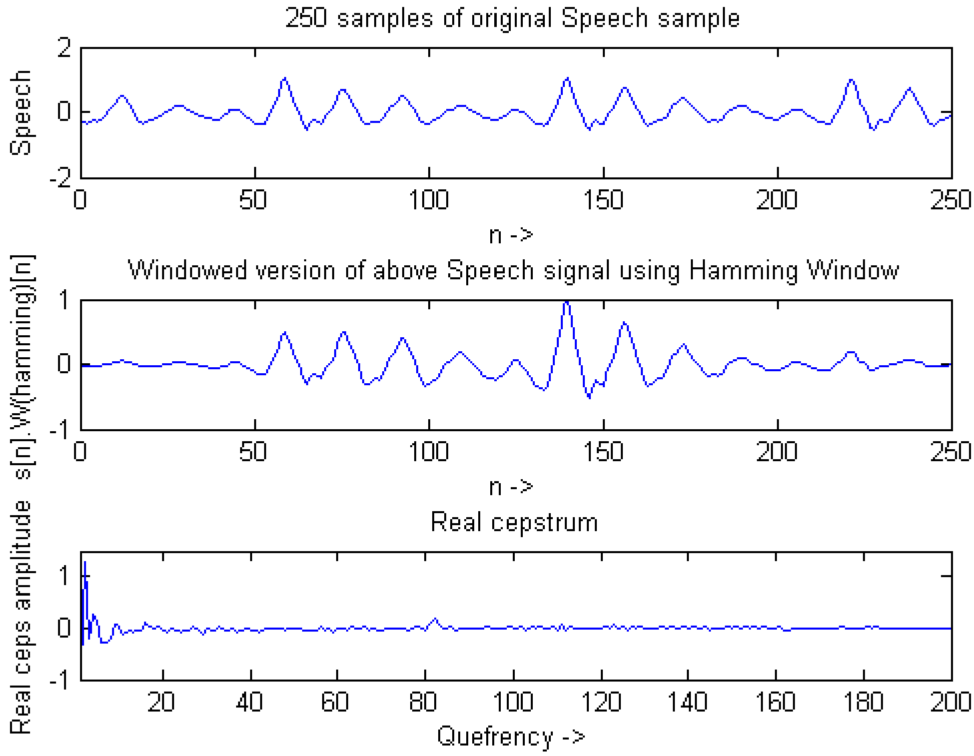
```
nearest integer
Qval = (round((inValue)*(2^(B))))/(2^(B));

% %Set up Quantization levels as a kind of lookup
% lvl = linspace(minVal,maxVal,2^B);
% % Level nearest to the quantized lookup -gives the index from the lookup
% [~,Idx] = min(abs(bsxfun(@minus,inValue,lvl.')));
% quantValue = inValue(Idx);

% Recover orginigal equivalent
quantValue = Qval*(2^N);

end
```

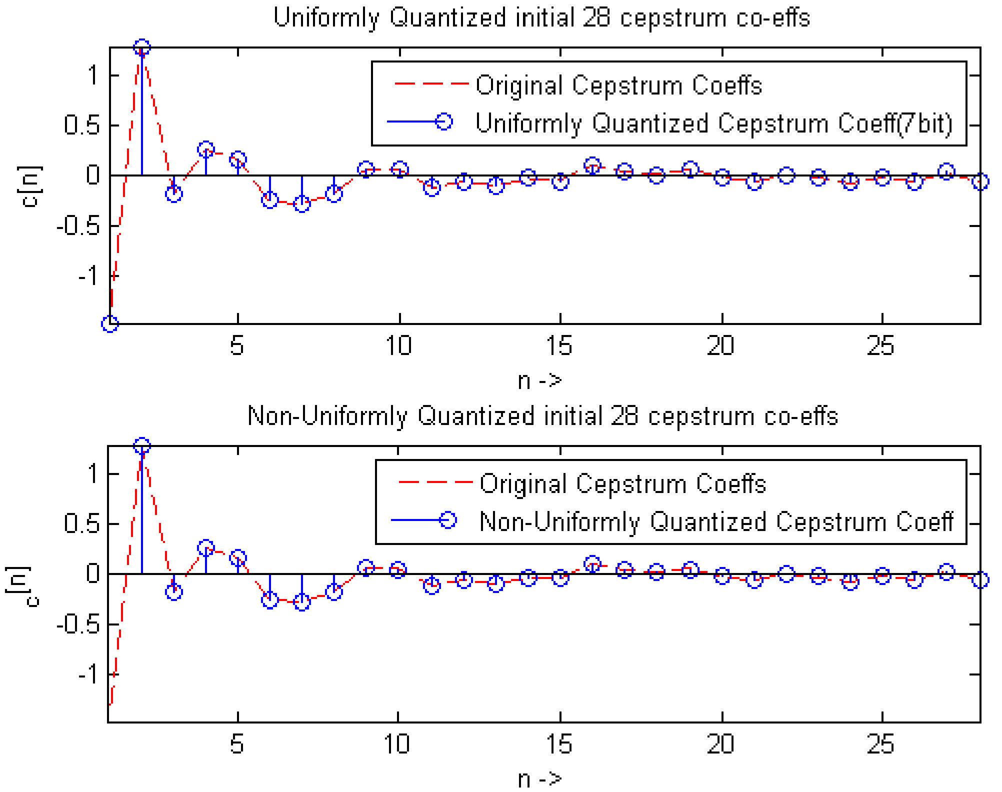
```
function [ out ] = nonUniformQuant( in )

% Custom non uniform Quantizer
% selects the bit depth of current co-efficient depending upon the dynamic
% range around its neighbours and the average dynamic range

% Since we have the pre-calculated co-efficients we can calculate average
% dynamic range of the whole set

sumDiff = 0;
for i=2:1:length(in)
   diff = abs(in(i) - in(i-1));
   sumDiff = sumDiff + diff;
end
avgDiff = sumDiff/(length(in)-1); % average dynamic range of the whole inp

% Max value of the array of co-effs
maxVal = max(in);
minVal = min(in);

% Based on the Local Dynamic range and the avg dynamic range make a
% decision to fix B
B(1) = 32;
out(1) = quant(in(1),B(1),minVal,maxVal);
for i=2:1:length(in)
   if (abs(in(i)-in(i-1)) > 3*avgDiff)
       B(i) = 64;
       out(i) = quant(in(i),B(i),minVal,maxVal);
   elseif (abs(in(i)-in(i-1)) > 2*avgDiff)
       B(i) = 32;
       out(i) = quant(in(i),B(i),minVal,maxVal);
   elseif (abs(in(i)-in(i-1)) > avgDiff)
       B(i) = 16;
       out(i) = quant(in(i),B(i),minVal,maxVal);
   else
       B(i) = 8;
       out(i) = quant(in(i),B(i),minVal,maxVal);
   end
end
```

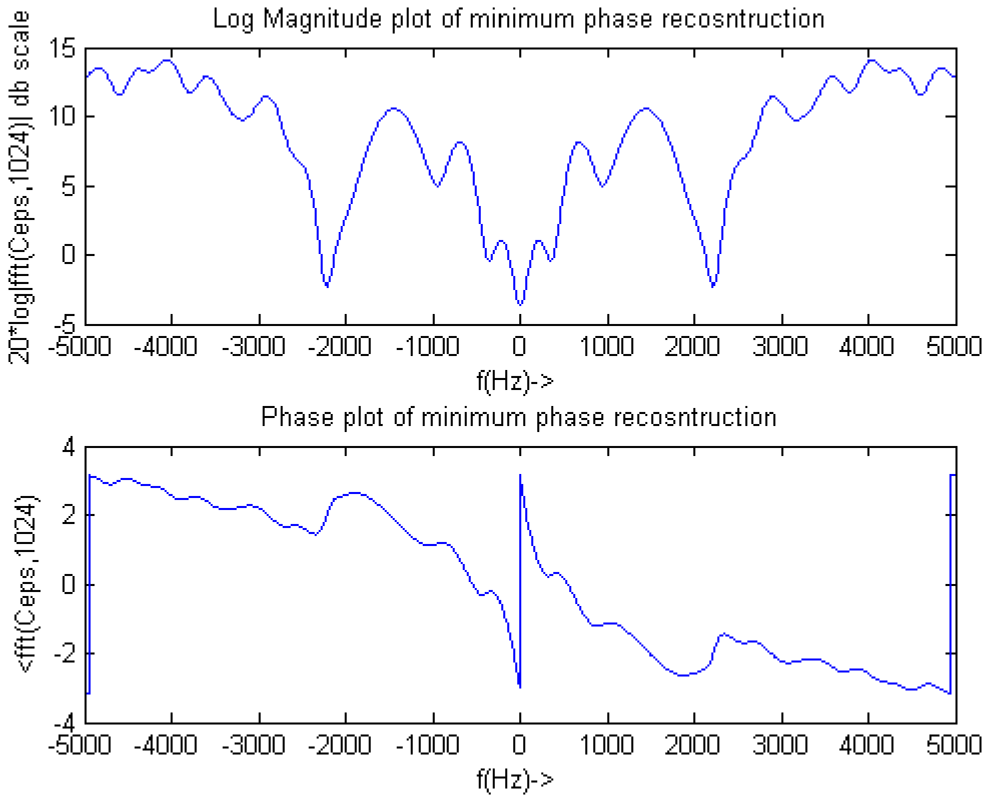
quantPitchFrqValue = 122 (Hz)

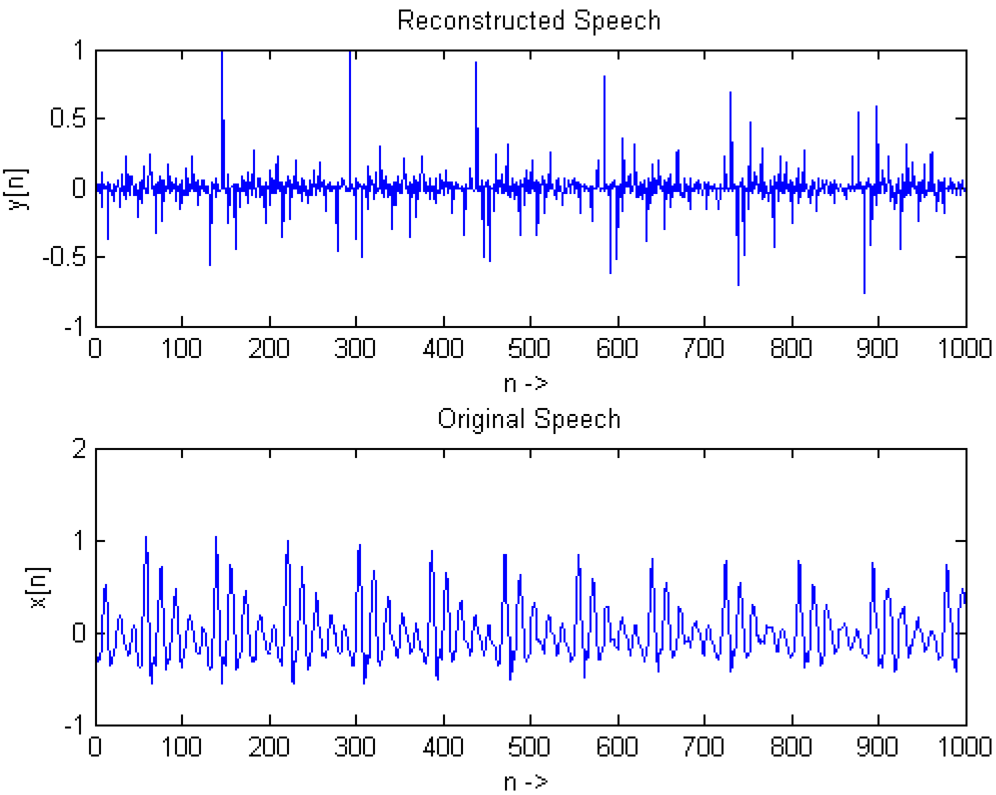
```
selectedHarmonicFrequencies (Hz) =

   1.0e+03 *

  Columns 1 through 7

    0.1270    0.2539    0.3711    0.4980    0.6152    0.7422    0.8594

  Columns 8 through 14

    0.9863    1.1035    1.2305    1.3477    1.4746    1.5918    1.7188

  Columns 15 through 21

    1.8359    1.9629    2.0801    2.2070    2.3242    2.4512    2.5684

  Columns 22 through 28

    2.6953    2.8125    2.9395    3.0566    3.1836    3.3008    3.4277

  Columns 29 through 35

    3.5449    3.6719    3.7891    3.9160    4.0332    4.1602    4.2773

  Columns 36 through 40

    4.4043    4.5215    4.6484    4.7656    4.8926

selectedHarmonicFrequenciesIndex =

  Columns 1 through 13

    13    26    38    51    63    76    88   101   113   126   138   151   163

  Columns 14 through 26

   176   188   201   213   226   238   251   263   276   288   301   313   326

  Columns 27 through 39

   338   351   363   376   388   401   413   426   438   451   463   476   488

  Column 40

   501

sampledLogMag =

  Columns 1 through 7

   -0.0430    0.1008   -0.0405    0.4758    0.8666    0.9158    0.7047

  Columns 8 through 14

    0.5885    0.8005    1.0298    1.1696    1.2122    1.1389    0.9593

  Columns 15 through 21

    0.7074    0.4022    0.1664   -0.2376    0.1562    0.6689    0.7955
```

```
  Columns 22 through 28

    0.9662    1.2272    1.3150    1.2105    1.1228    1.1680    1.3126

  Columns 29 through 35

    1.4602    1.4511    1.3502    1.4879    1.6208    1.5687    1.5242

  Columns 36 through 40

    1.5419    1.4171    1.3465    1.5151    1.5435


sampledPhase =

  Columns 1 through 7

    1.4426    0.4545    0.2507    0.2163   -0.2908   -0.8790   -1.1968

  Columns 8 through 14

   -1.1372   -1.1372   -1.3343   -1.5987   -1.9301   -2.2208   -2.4749

  Columns 15 through 21

   -2.6211   -2.6105   -2.5213   -2.1442   -1.4671   -1.5560   -1.6892

  Columns 22 through 28

   -1.6638   -1.8015   -2.0915   -2.2577   -2.2352   -2.1803   -2.1718

  Columns 29 through 35

   -2.2993   -2.5007   -2.5071   -2.4501   -2.6215   -2.8080   -2.8473

  Columns 36 through 40

   -2.9579   -3.0591   -2.9238   -2.9294   -3.1028


MSE =    0.0943
```


250 samples of original Speech sample
Speech
n ->
Windowed version of above Speech signal using Hamming Window
s[n].W(hamming)[n]
n ->
Real cepstrum
Real ceps amplitude
Quefrency ->


Uniformly Quantized initial 28 cepstrum co-effs
Original Cepstrum Coeffs
Uniformly Quantized Cepstrum Coeff(7bit)
c[n]
n ->
Non-Uniformly Quantized initial 28 cepstrum co-effs
Original Cepstrum Coeffs
Non-Uniformly Quantized Cepstrum Coeff
c[n]
n ->





Published with MATLAB® R2013a