Communication Lab II

(ELC3940)

Experiment - 01

Object : Record your own speech signal (say “hello”) using single channel with a sampling frequency of 16 kHz. Draw the power spectrum of the signal. Quantize the signal and generate a bit sequence using a uniform quantizer of 16 levels.

Name: Maha Z. Khan

S.No.: A3EL-02

Faculty No.: 19ELB056

Date of Performing: 11|01|2022

Date of Submission: 18|01|2022

Software Used :

MATLAB R2021A

Program :

We started with converting a .m4a file into a .wav file for processing, and added it to matlab's path

Using audioinfo function, information about the .wav file is loaded. audioread reads the file and outputs the signal and sampling frequency

```
info = audioinfo('C:\Users\Maha Khan\Downloads\Hello5.wav')
```

```
info = struct with fields:
             Filename: 'C:\Users\Maha Khan\Downloads\Hello5.wav'
    CompressionMethod: 'Uncompressed'
          NumChannels: 2
           SampleRate: 48000
         TotalSamples: 199680
             Duration: 4.1600
                Title: 'Recording (7)'
              Comment: []
               Artist: []
        BitsPerSample: 16
```

```
[sig , fs1] = audioread('C:\Users\Maha Khan\Downloads\Hello5.wav'); % Voice signal sig with dua
                                          % and fs1 as the sampling frequency
```

This section can be skipped; At frequency higher than the sampling frequency, the audio sounded significanly higher pitched and vise versa for lower sampling frequency.

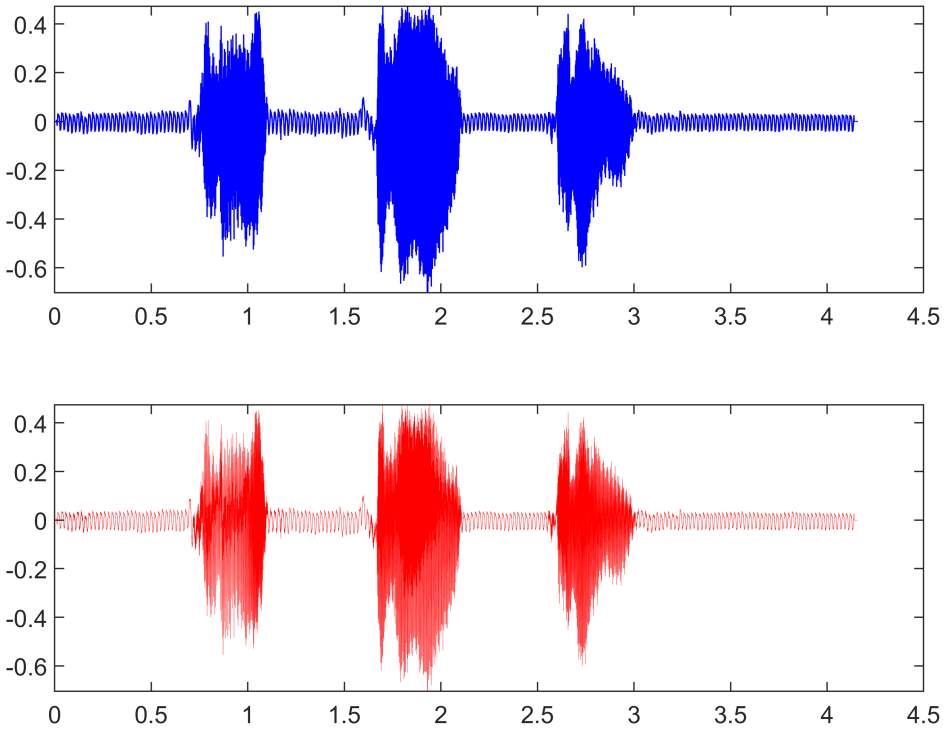
```
sound(sig,fs1)  % To play the given voice signal
sound(sig,fs1+5000)  % To play the given voice signal
sound(sig,fs1-5000)  % To play the given voice signal
```

Storing signal information in the respective variable

```
dur = info.Duration;         % Signal duration in seconds
Ts1 = info.TotalSamples;     % Total no. of samples in time-domain
n1 = linspace(0,dur,Ts1);
y1 = sig(:,1);               % Channel 1 of the signal sig
```

This section can be skipped; I compared the plot of the two channels of the signal, there isn't much difference in the signal recorded from the microphone use, however microphones with stereo depth have significant depth to it due to which plots will e different.

```
y1 = sig(:,1);                % Channel 1 of the signal sig
z1 = sig(:,2);                % Channel 2 of the signal sig

subplot(2,1,1);
plot(n1,y1,'b');
subplot(2,1,2);
plot(n1,z1,'r','LineWidth',0.1);
```



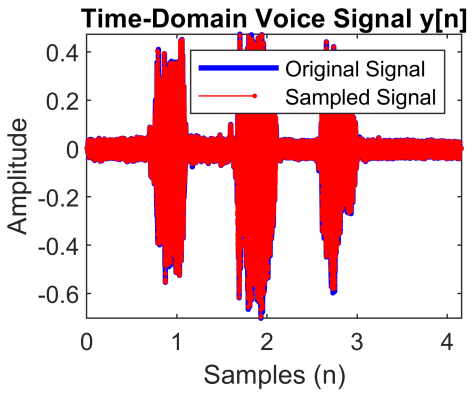
1.)Sampling:

I sampled the signal at the given sampling frequency of 16000Hz by interpolating the signal at the original sampling frequency to the required sampling frequency

```
fs = 16000;                   % Sampling frequency given in the Experiment
k = 4;                        % bit-rate for 16 bit quantizer
l = 2^k;                      % 16-level
n = 0:1/fs:dur;
y = interp1(n1,y1,n);         % Signal Waveform with sampling frequency fs
Ns = length(y);
```

Plotting the discrete-time sampled signal and comparing it to the original voice signal

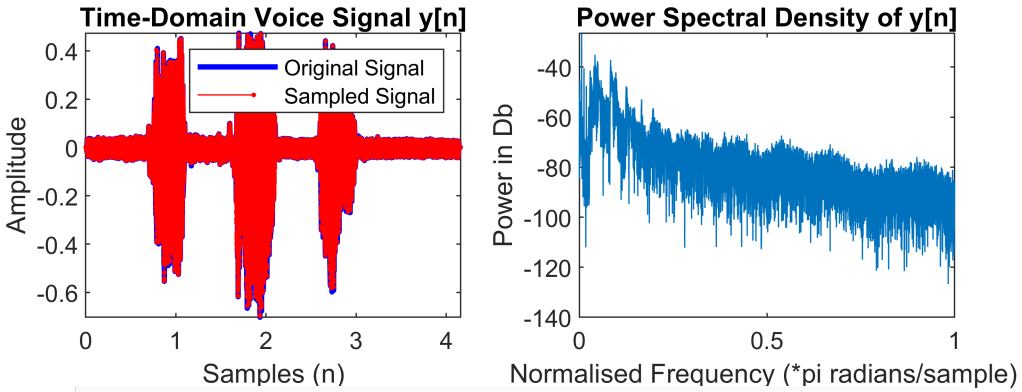
```
figure('NumberTitle', 'off', 'Name', 'Pulse Code Modulation of Voice Signal');
subplot(2,2,1);          % Plot of Discrete time-domain signal y
plot(n1,y1,'b','LineWidth',2);
hold on

s = stem(n,y);
s.Marker = '.';
s.Color = 'r';
% ymax = max(y);
% ymin = min(y);
% yline([ymax ymin],'--',{'Max','Min'})
xlabel('Samples (n)');
ylabel('Amplitude');
title('Time-Domain Voice Signal y[n]');
legend('Original Signal','Sampled Signal','Location','northeast');
```



Calculating the power spectrum by using periodogram function

```
subplot(2,2,2);
[Pxx,F] = periodogram(y,[],length(y),fs);  % Using periodogram function for power spectral dens
f = (2/fs)*F;                              % Normalised frequency vector
plot(f,10*log10(Pxx))                      % Logarithmic plot of of psd vs normalised frequency
xlabel('Normalised Frequency (*pi radians/sample) ');
ylabel('Power in Db');
```

```
title('Power Spectral Density of y[n]')
```



Comparing generated spectrum with Signal analyser

2.) Quantizing :

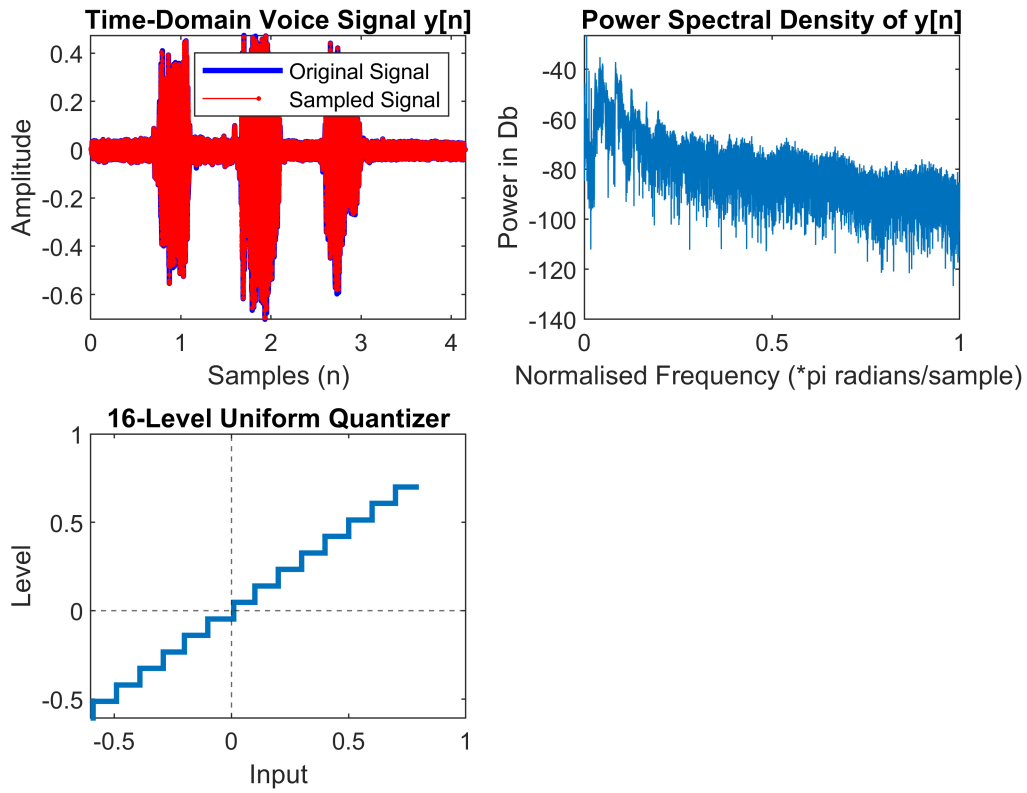
Used quantiz function to generate a 16 level uniform quantizer with range specified in the codebook from -0.7 to +0.7. The index of a particular sample is stored in index variable and quants contains the quantized vales of the sampled signal

```
partition = [linspace(-0.7,0.7,l-1)];
codebook = [linspace(-0.7,0.7,l)];

% Transfer Function of quantizer
input = -0.6:0.01:0.8;
[indexinp,tf] = quantiz(input,partition,codebook);
```

The transfer function plot of the quantizer is plotted below

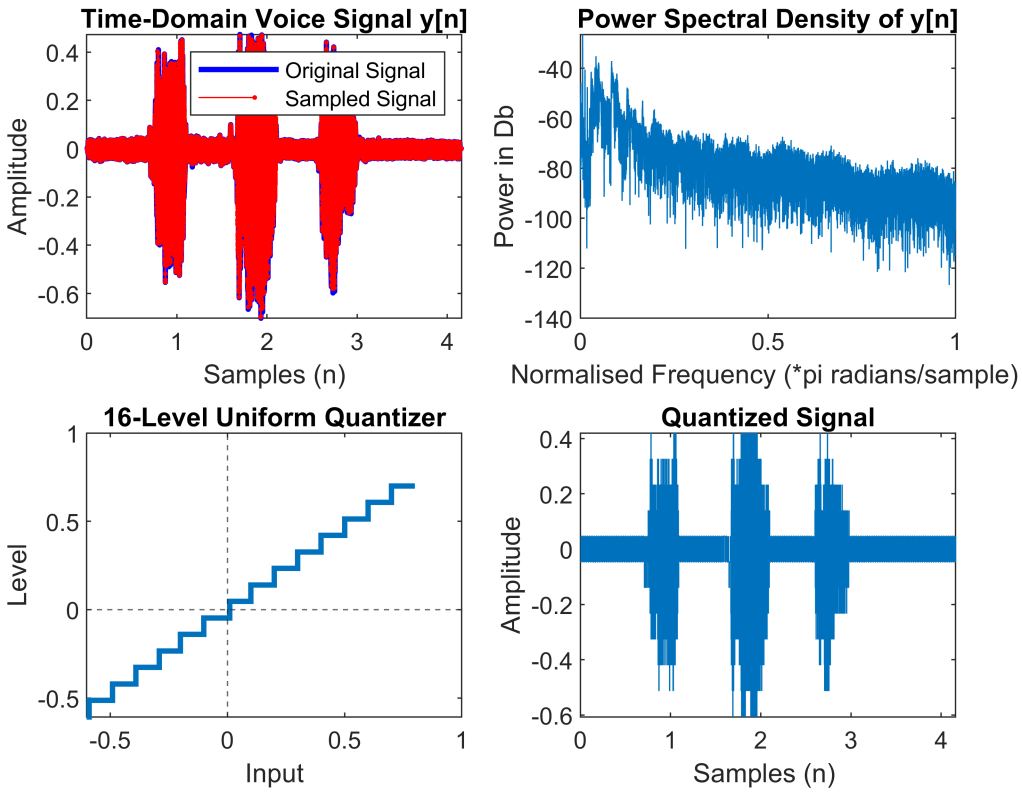
```
subplot(2,2,3);
stairs(input,tf,'LineWidth',2);
xline(0,'--');
yline(0,'--');
xlabel('Input');
```

```
ylabel('Level');
title('16-Level Uniform Quantizer')
```



Below is the Plot of the quantized signal quants

```
[index,quants] = quantiz(y,partition,codebook); % Quantized Signal
% sound(quants',fs)

% Quantized Signal Plot
subplot(2,2,4);
stairs(n,quants);
xlabel('Samples (n) ');
ylabel('Amplitude');
title('Quantized Signal');
```



3.) Encoding :

To encode each value of the quantized signal quants, I encoded using a 4-bit encoder for each value of the 16-leveled the quantized signal using the index of the particular partition the value belongs to thus generating a pcm signal whose length is given by k*Total no. of samples ; where k is the bit rate of the modulated signal.

```
coder = [0 0 0 0;       % 4-bit Coder
         0 0 0 1;
         0 0 1 1;
         0 0 1 0;
         0 1 1 0;
         0 1 1 1;
         0 1 0 1;
         0 1 0 0;
         1 1 0 0;
         1 1 0 1;
         1 1 1 1;
         1 1 1 0;
         1 0 1 0;
         1 0 1 1;
         1 0 0 1;
         1 0 0 0];

pcm = linspace(0,0,Ns*k); % Array to store pcm signal
i = 1;
```

```
c = 1;
    % Encoding Quantized Signal
     while i < length(pcm)
         pcm(i:i+3) = coder(index(c)+1,:);
         i = i+4;
         c = c+1;
     end
```

Experimental Decoder code and PCM Plot

```
%  %PCM Signal Plot
%  subplot(2,3,6);
%  stairs(1:5000,pcm(1:5000));
%  xlabel('N');
%  ylabel('Amplitude');
%  title('PCM Signal');

%  hold on
%
%  decoder =  [0 0 0 0;
%              0 0 0 1;
%              0 0 1 1;
%              0 0 1 0;
%              0 1 1 0;
%              0 1 1 1;
%              0 1 0 1;
%              0 1 0 0;
%              1 1 0 0;
%              1 1 0 1;
%              1 1 1 1;
%              1 1 1 0;
%              1 0 1 0;
%              1 0 1 1;
%              1 0 0 1;
%              1 0 0 0];
%   rpcm = pcm;
%   dindex = (linspace(0,0,length(rpcm)/4))';
%   i = 1;
%   c = 1;
%   while i<length(rpcm)
%       for j = 1:length(decoder)
%       sub = abs(rpcm(i:i+3)-decoder(j,:));
%       if sum(sub) == 0
%           dindex(c)= j-1;
%       end
%       end
%       i = i+4;
%       c = c+1;
%   end
%   subplot(2,2,3)
```

```
%    stairs(1:5000,rpcm(1:5000))
% dsig = (linspace(0,0,length(dindex)))';
% for i = 1:length(dsig)
%     dsig(i) = codebook(dindex(i)+1);
% end
% subplot(2,2,1)
% stairs(n,dsig,'yellow')
% %  sound(dsig,fs)
```

Discussion :

Playing and comparing the plot of the original voice signal and the quantized signal, we can observe that there is a loss of information and significant lowering of the quality of the signal after passing through the quantizer, it is an irreversible process.

Original Signal first channel:

```
sound(y,fs);
```

Quantized Signal first channel:

```
sound(quants,fs);
```