Bangladesh University of Engineering and Technology

Course Number: EEE 312

Course Title: Digital Signal Processing Laboratory

Experiment Number: 01

Name of the Experiment(s):

Study of Sampling, Quantization and Encoding

Assignment number: 01

Date of Performance: 23/11/2022

Date of Submission: 30/11/2022

Submitted By

Name : Tasmin Khan

Student ID : 1906055

Department : EEE

Section : A2

Problem 1

Question:

1. Consider the given signal below:

$$y(t) = sin(2\pi * 20 * R * t) + sin(2\pi * 60 * R * t) + sin(2\pi * 120 * R * t);\ \mathbf{0 < t < 1ms}$$

Here, R is the last two digits of your student ID. Now:

(i) Sample the signal with sampling frequencies of (240*R), (360*R), (640*R) and (810*R) samples/sec. From each discrete time signal, obtain the reconstructed signal, $\tilde{y}(t)$ and plot the error signal, $e(n) = \tilde{y}(t) - y(t)$. Plot/Stem every resulting signal and comment on your results.

(ii) Sample the signal by (1500*R) samples/sec. Then, quantize the signal with 3-bit to 8-bit uniform quantizers, separately. Obtain the SQNR for each case and compare them with the theoretical SQNR. From this, plot both experimental and theoretical SQNR with respect to bits in a single plot (use legends).

(i) MATLAB code:

```
clc
clear all
close all

%Analog signal generation
R=55;
Fs=240*R;
y=@(t) sin(2*pi*20*R*t)+sin(2*pi*60*R*t)+sin(2*pi*120*R*t);
t=0:0.000001:0.001;

%Analog signal plot
subplot(4,1,1)
plot(t,y(t),'Color','#008000')
xlabel('Time')
title("Analog Signal")

%Sampling the signal
ts=0:1/Fs:0.001;
ys=y(ts);

%Sampled signal plot
subplot(4,1,2)
stem(ts,ys)
xlabel('Time')
title("Sampled Signal")
```

```
%Reconstruction
yr=interp1(ts,ys,t);

%Reconstructed signal plot
subplot(4,1,3)
plot(t,yr,'k')
xlabel('Time')
title("Reconstructed Signal")

%Error calculation
err=(yr-y(t));

%Error plot
subplot(4,1,4)
plot(t,err,'Color','#C41E3A')
xlabel('Time')
title("Error Signal")
```

(i) Plot:

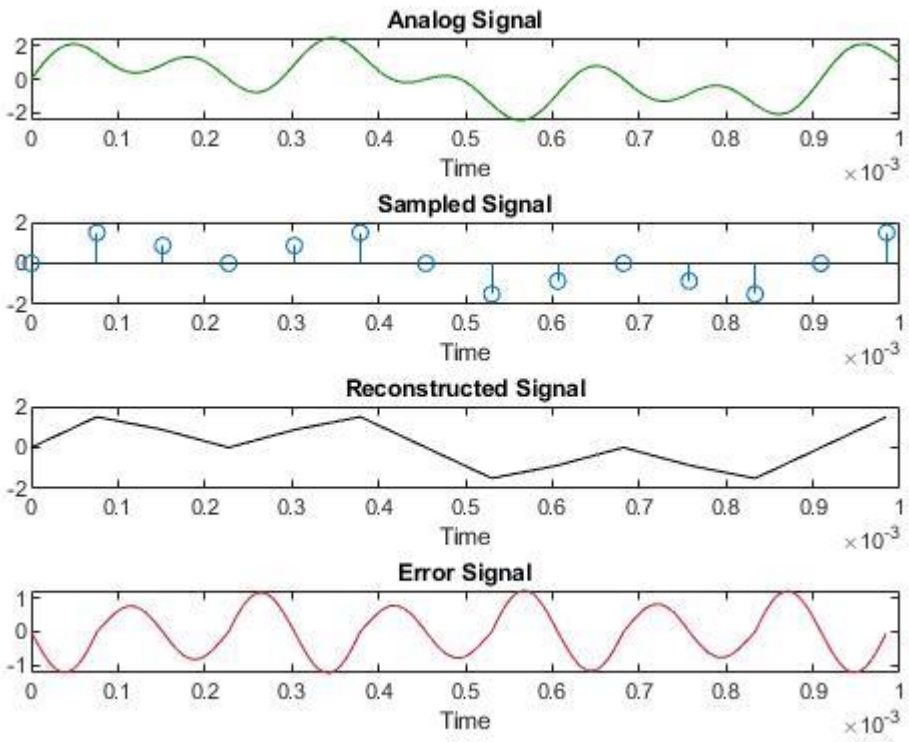


Fig 1.1: Plot for sampling frequency Fs = 240*R

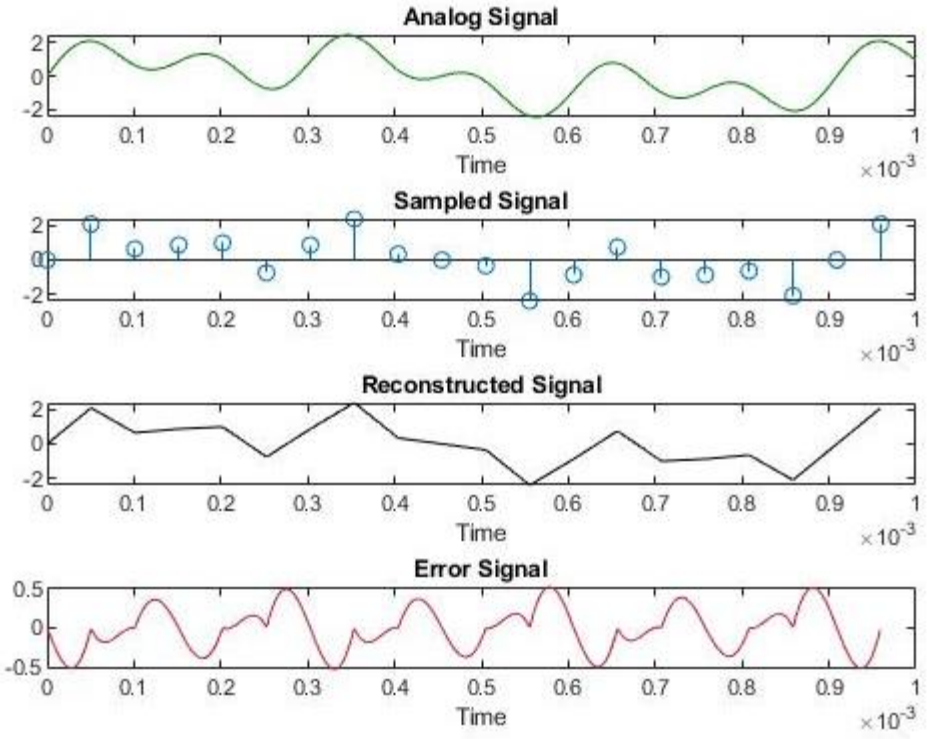


Fig 1.2: Plot for sampling frequency Fs = 360*R

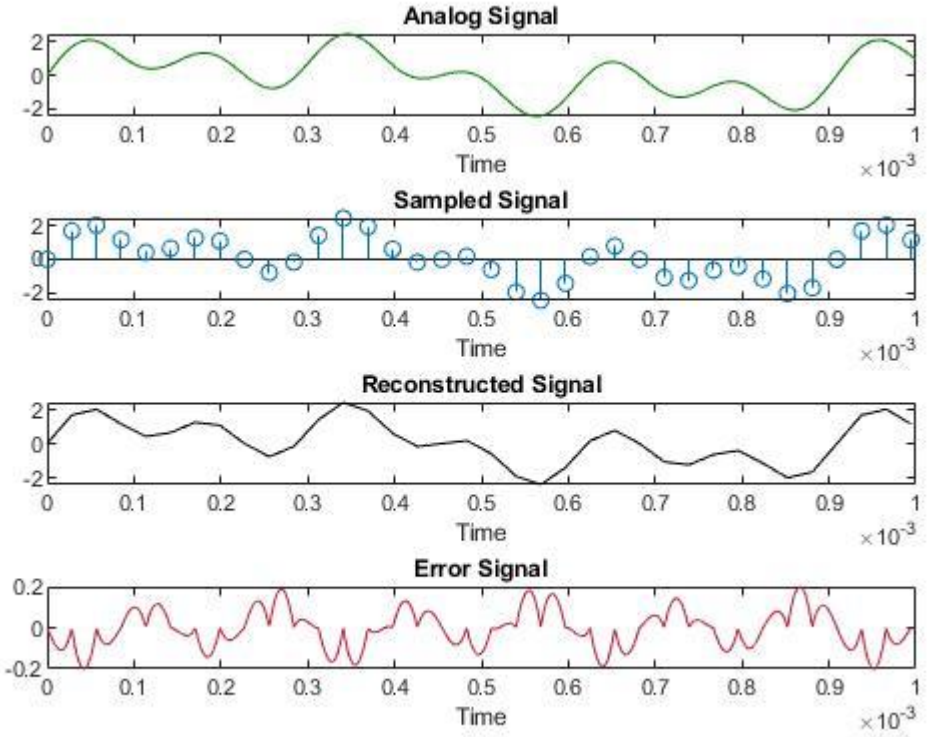


Fig 1.3: Plot for sampling frequency Fs = 640*R

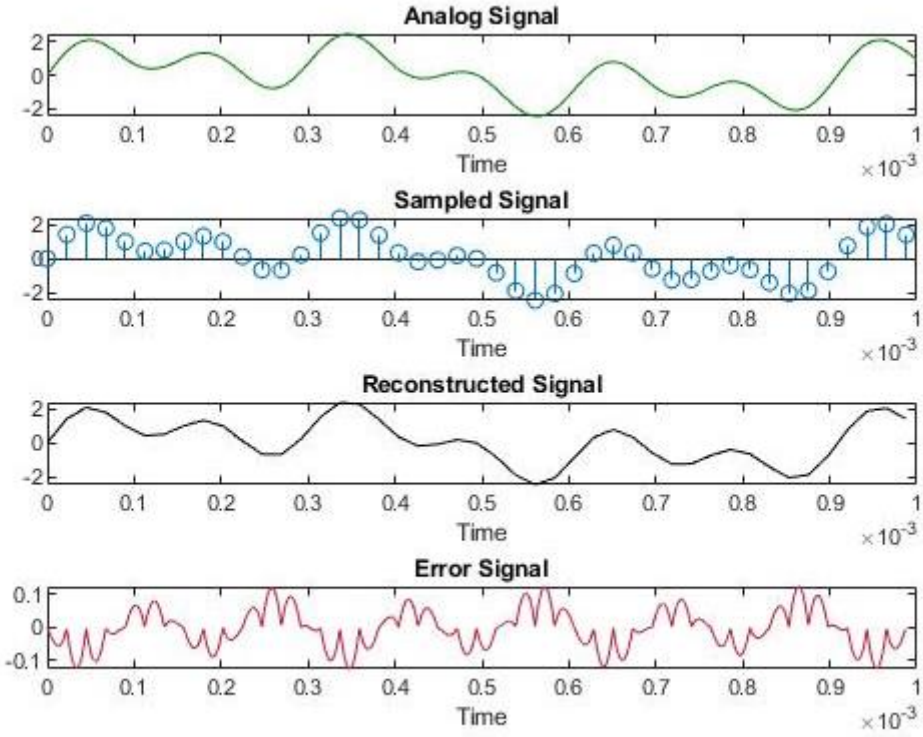


Fig 1.4: Plot for sampling frequency Fs = 810*R

Observation:

The original signal consists of sinusoidal components of three frequency: 1100 Hz, 3300 Hz and 6600 Hz. From Nyquist sampling theorem, Nyquist frequency is 13200 Hz.

For the first plot, we sampled it at 13200 Hz and hence we see no aliasing here. However, since the sampling frequency is very low, it is barely enough to reconstruct properly. Thus, the reconstructed signal fails to follow the original signal waveshape properly and the error signal that we get is quite large in magnitude ranging from -1 to +1

For the second plot, the sampling frequency is increased to 19800 Hz and here we see the reconstructed signal matches the original signal better. We see more peaks that were absent earlier. The error signal got smaller too, now laying in the range of -0.5 to +0.5

For the third plot, the sampling frequency is further increased to 35200. Now the reconstructed signal takes more of a sinusoidal shape with smaller linear segments representing the original signal better. Error has also been reduced to a range of -.02 to +0.2

For the fourth plot, the sampling frequency is highest at 44550 Hz. The reconstructed signal thus, is almost reassembling the original signal. The error signal is lowest now ranging from -0.1 to +0.1

So, with the highest sampling frequency as f=Fs/F decreases, we get the best result and minimum error given that it follows the Nyquist sampling theorem. Because of the fact that we used linear interpolation to plot the reconstructed signal, there are some errors even in the best result. Specially the peaks are not really in a sinusoidal shape, rather straight line is visible there. But it reserves the most amount of information from more frequent sampling producing the best matching signal output.

(ii) MATLAB Code:

```
clear all
close all

%Analog signal generation
R=55;
Fs=1500*R;
y=@(t) sin(2*pi*20*R*t)+sin(2*pi*60*R*t)+sin(2*pi*120*R*t);
t=0:0.000001:0.001;

%Analogn signal plot
subplot (4,2,1)
plot(t,y(t),'Color','#008000');
xlabel('Time');
ylabel('Stength');
title('Original signal')

%Sampling the signal
ts=0:1/Fs:0.001;
ys=y(ts);
ysvar=sum(ys.*ys)/length(ys);

%sampled points plot
subplot(4,2,2)
stem(ts*Fs,ys,'Color','#008000');
xlabel('Number of samples');
ylabel('Stength');
title('Sampled signal');

%Quantization
for bit=3:8
    count=bit-2;
    L=2.^bit;
    i=1:L;
    delta=(max(ys)-min(ys))/(L-1);
    q=min(ys)+(i-1)*delta;
    for j=1:length(ys)
        for k=2:L
            if (ys(j)>=q(k-1)) && (ys(j)<=q(k))
                if ys(j)<((q(k-1)+q(k))/2)
                    yq(j)=q(k-1);
                else
                    yq(j)=q(k);
                end
            end
        end
    end
    err=yq-ys;
    errvar=sum(err.*err)/length(err);
    SQNR_Exp(count)=10*log10(ysvar/errvar);
    SQNR_Theory(count)=4.77+(10*log10(ysvar/max(ys)^2))+6*bit;

    %bit wise plot
    subplot(4,2,bit)
    stairs(ts*Fs,yq)'
    xlabel('Number of samples');
    ylabel('Stength');
    title(['Quantization for ',num2str(bit),'bit']);
```

```
end

%% SQNR plot
bit=3:8;
plot(bit,SQNR_Exp,'-d');
hold on
plot(bit,SQNR_Theory,'-d');
legend('From Experiment','From Theory')
title("Plot of SQNR in dB")
xlabel('Bit number')
ylabel('SQNR(dB)')
xticks(bit)
```

(ii) Plot

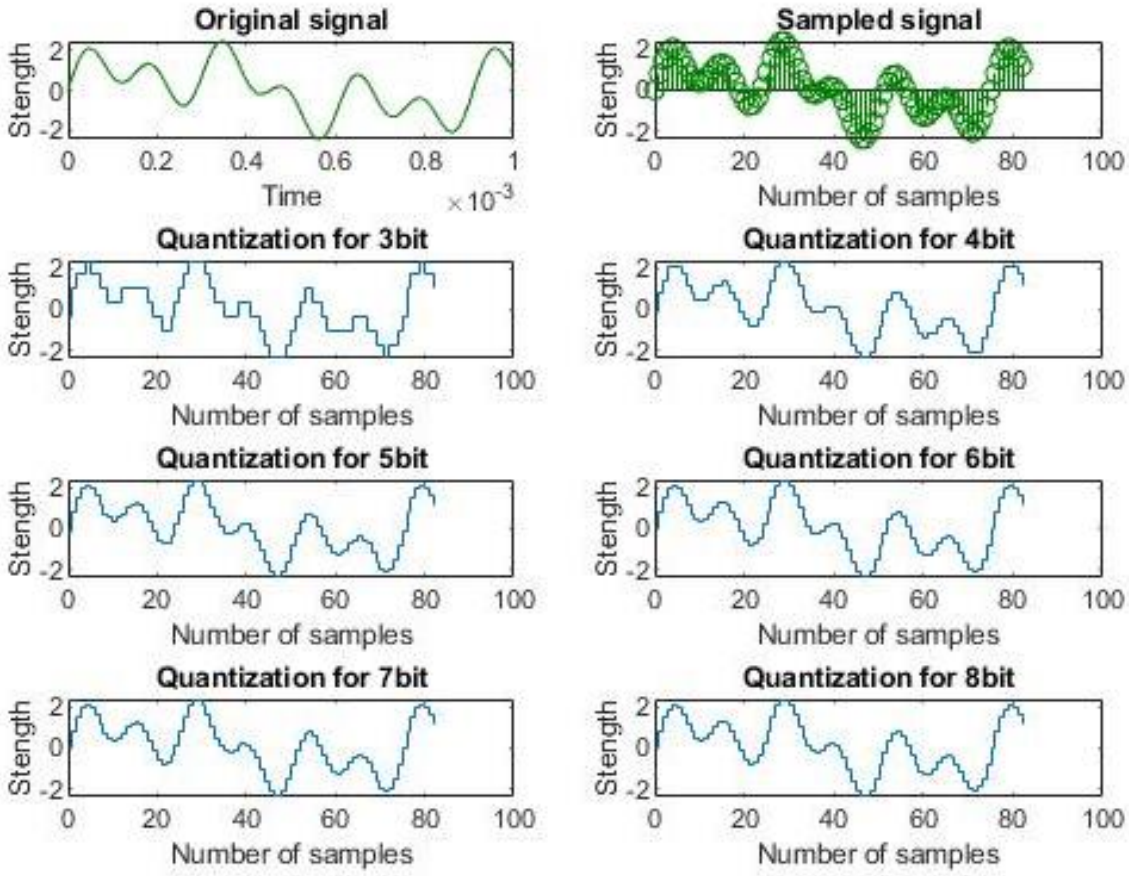


Fig 1.5: Plot of signals with varying bits of quantization

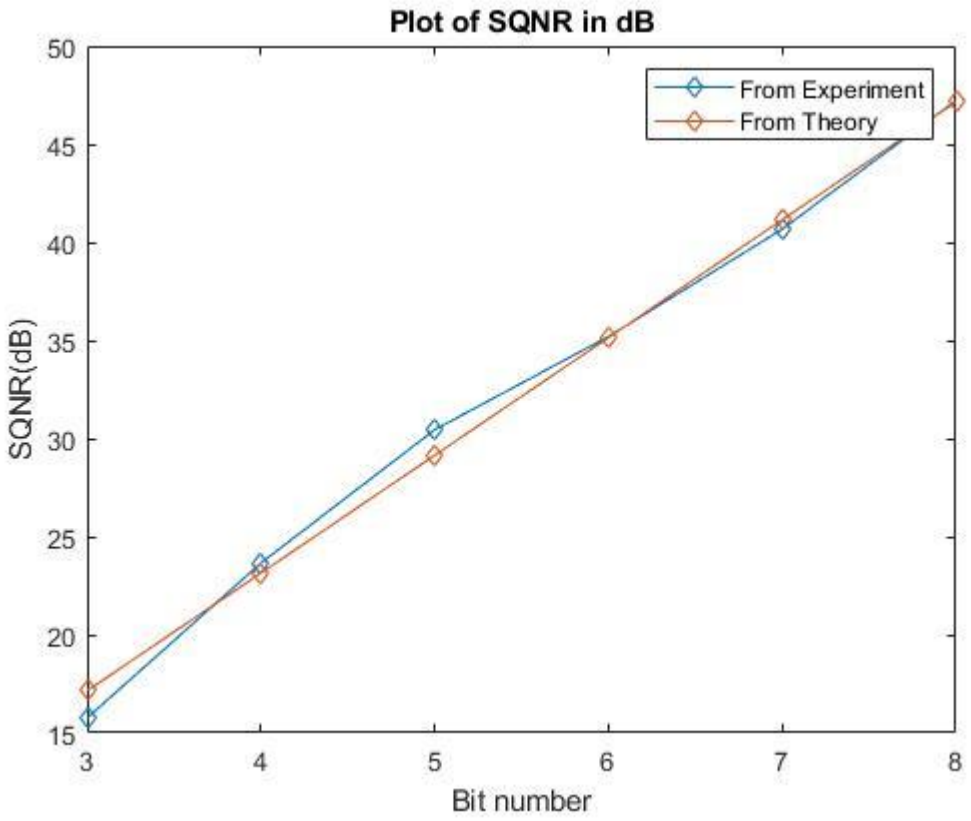


Fig 1.6: Plot of SQNR vs Number of bits

Table of SQNR values:

Number of bits	3	4	5	6	7	8
SQNR(dB) Experimental	15.757	23.651	30.445	35.194	40.671	47.223
SQNR(dB) Theoretical	17.143	23.143	29.143	35.143	41.143	47.143

Observation:

Number of bits determine the number of levels assigned in the quantization process. As bit number increases, the more of levels are uniformly distributed. As a result, the rounding process is more precise, and the quantized signal represent the original signal better. We can see this from the first plot that with increasing number of bits, the quantized signal is more accurate.

Now SQNR stands for signal to quantization ratio. Mathematically, it is the ratio of variance of signal (signal power) to the variance of error (error power):

$$SQNR = \frac{\sigma_x^2}{\sigma_e^2}$$

This formula is used to calculate the experimental value using every signal data point.

An approximate formula for SQNR is as below:

$$SQNR_{dB} = 4.77 + 10\log_{10}\left(\frac{\sigma_x^2}{V^2}\right) + 6b$$

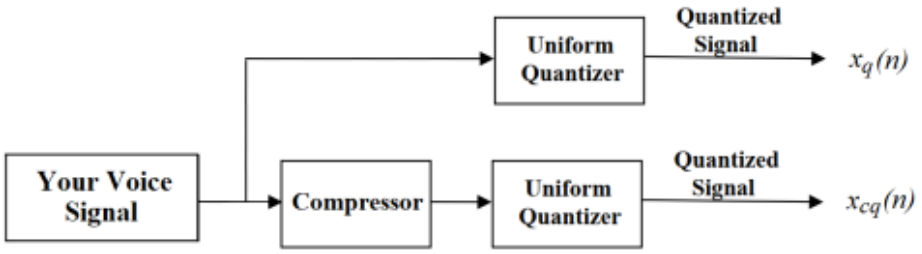
This formula is used to calculate the theoretical value where b=Number of bits.
As number of bits increases, error decreases and so does the error variance. But the signal variance stays constant. So, SQNR increases rapidly in value with increasing bits. We can see that holds true for both the experimental and theoretical value from the second plot and the table.

We can also see that the values calculated using the two methods are very close to each other for each number of bits. This justifies the experimental values. The slight deviation can be caused from the mathematical approximation of SQNR formula.

Problem 2

Question:

2. Record your voice as an **.mp3/.wav** file (any one sentence long audio clip of your choice) through your phone and import it to your pc. You can use the `audioread()` function to read the signal. With that function, you shall get a discrete time signal as well as a sampling frequency, F_s. Therefore, complete the following system:



Consider an 8-bit quantizer for the system. Based on the information above:

(i) First take a μ −law compressor (read lab sheet for specifications). Show waveshape of each block's output.

(ii) Listen to both quantized signals using `sound(signal, Fs)`. What difference do you hear? Comment on it.

(iii) Compute the signal to quantization-noise ratio for each case.

(iv) Repeat tasks (i)-(iii) using an A −law compressor.

(i) MATLAB code:

```
clc;
close all;
clear all;

%Audio signal input
[ya,Fs]=audioread("DSP audio signal.mp3");
ttotal=length(ya)/Fs;
ta=1:length(ya);
[samples,channel]=size(ya);
```

```
%Audio signal plot
subplot(4,1,1);
plot(ta,ya,'Color','#008000');
xlabel('Number of samples');
ylabel('Weight');
title('Input Audio signal');

%%
%sound(ya,Fs)

%% mu law compresion
mu=255;
ymu=(log(1+mu.*abs(ya))/log(1+mu)).*sign(ya);

%Compressed signal plot
subplot(4,1,2);
plot(ta,ymu,'color','#0072bb');
xlabel('Number of samples');
ylabel('Weight');
title("Compressed Signal");

%uniform quantization of compressed signal
bit=8;
L=2^bit;
delta=(max(ymu)-min(ymu))/(L-1);
for c=1:channel
    for i=1:L
        q(i,c)=min(ymu(:,c))+(i-1)*delta(c);
    end
end
for j=1:length(ymu)
    for k=2:L
        for c=1:channel
            if (ymu(j,c)>=q(k-1,c)) && (ymu(j,c)<q(k,c))
                if ymu(j,c)<((q(k-1,c)+q(k,c))/2)
                    yq(j,c)=q(k-1,c);
                else
                    yq(j,c)=q(k,c);
                end
            end
        end
    end
end

%compressed and quantized signal plot
subplot(4,1,3);
stairs(ta,yq);
xlabel('Number of samples');
ylabel('Weight');
title('Uniform Quantizatization after mu law compression');

%SQNR calculation
errCQ=yq-ymu;
errvarCQ=sum(errCQ.*errCQ)/length(errCQ);
ymuvar=sum(ymu.*ymu)/length(ymu);
for c=1:channel
    SQNR_CQ(c)=10*log10(ymuvar(c)/errvarCQ(c));
```

```
    fprintf('SQNR(dB) with mu compressor for channel%d =
%f\n',c,SQNR_CQ(c));
end

%%
%sound(yq,Fs);

%% Uniform Quantization without compression
for c=1:channel
    for i=1:L
        qq(i,c)=min(ya(:,c))+(i-1)*delta(c);
    end
end
for j=1:length(ya)
    for k=2:L
        for c=1:channel
            if (ya(j,c)>=qq(k-1,c)) && (ya(j,c)<qq(k,c))
                if ya(j,c)<((qq(k-1,c)+qq(k,c))/2)
                    yqq(j,c)=qq(k-1,c);
                else
                    yqq(j,c)=qq(k,c);
                end
            end
        end
    end
end

%compressed and quantized signal plot
subplot(4,1,4);
stairs(ta,yqq,'k');
xlabel('Number of samples');
ylabel('Weight');
title('Uniform Quantizatization without compression');

%SQNR calculation
errQ=yq-ymu;
errvarQ=sum(errQ.*errQ)/length(errQ);
yavar=sum(ya.*ya)/length(ya);
for c=1:channel
    SQNR_Q(c)=10*log10(yavar(c)/errvarQ(c));
    fprintf('SQNR(dB) without compression for channel%d =
%f\n',c,SQNR_Q(c));
end

%%
%sound(yqq,Fs)
```

(i) Plot:

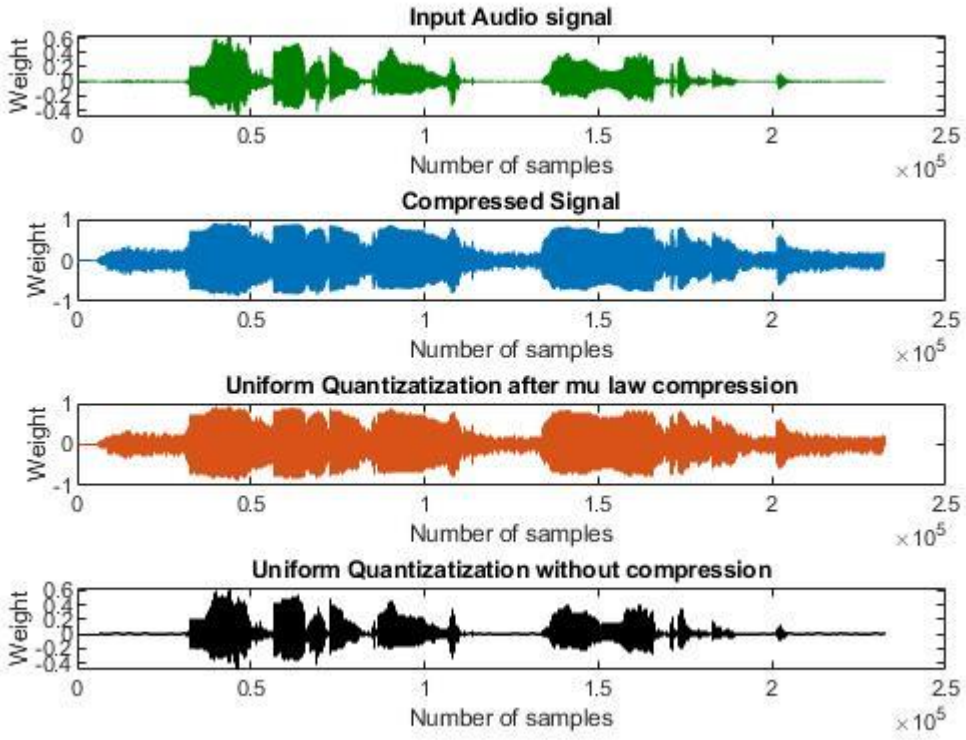


Fig 2.1: Signal plot comparison

```
Command Window
  SQNR(dB) with mu compressor for channel1 = 43.287272
  SQNR(dB) with mu compressor for channel2 = 43.287272
  SQNR(dB) without compression for channel1 = 30.936432
  SQNR(dB) without compression for channel2 = 30.936432
fx >>
```

Fig 2.2: SQNR value output from the command window

Observation:

μ law compressor compressed the signal amplifying the relatively small values the more.

The formula for such compression is:

$$\text{Compressed signal, } F(x) = \frac{\ln(1+\mu|x|)}{\ln(1+\mu)}\text{sgn(x) where } 0 \leq |x| \leq 1.$$

We took μ=255

For this audio signal, the relatively small values were the surrounding noise. As these get more amplified after μ law compression, we hear more noise in the sound from quantized signal. Since the compression is not uniform, the compressed signal is not exactly like the original audio and the signal may sound a bit distorted.

Without any μ law compression, the noises are not amplified as much. So the sound from just uniformly quantized signal sound smoother with less noise. So, for this experiment, the uniform quantization provides better sound quality.

Comparing the SQNR values in dB, we see that the value is higher for μ law compressor quantization (43.287 dB) than just uniform quantization (30.936 dB).

In the first process, compression ensures that there are more quantization levels distributed for relatively smaller values. This reduces the quantization error allocating more levels for small amplitudes. Since SQNR = signal power/ error power, SQNR value is increases with decreased error value.

In the second process no compression is done, and levels are allocated uniformly. So, the quantized values deviate more specially for small amplitude values. As a result, error is increased and SQNR value is decreased as we can see from the data.

Here, depending on the audio recorder, there can be multiple audio channels. Unless there is a significant difference in the left and right channel, the stereo intensity is nearly same for the whole signal and so is the SQNR values.

(ii) MATLAB code:

```
clc;
close all;
clear all;

%Audio signal input
[ya,Fs]=audioread("DSP audio signal.mp3");
ttotal=length(ya)/Fs;
ta=1:length(ya);
[samples,channel]=size(ya);

%Audio signal plot
subplot(4,1,1);
plot(ta,ya,'Color','#008000');
xlabel('Number of samples');
ylabel('Weight');
title('Input Audio signal');
%%
%sound(yin,Fs)

%% A law compresion
A=87.56;
for c=1:channel
    for i=1:length(ya)
        if abs(ya(i,c))>=1/A
            yA(i,c)=((1+log(A*abs(ya(i,c))))/(1+log(A)))*sign(ya(i,c));
        elseif abs(ya(i,c))<1/A
            yA(i,c)=((A*abs(ya(i,c)))/(1+log(A)))*sign(ya(i,c));
        end
    end
end

%Compressed signal plot
subplot(4,1,2);
plot(ta,yA,'color','#0072bb');
xlabel('Number of samples');
ylabel('Weight');
title("Compressed Signal");
%uniform quantization of compressed signal
bit=8;
L=2^bit;
delta=(max(yA)-min(yA))/(L-1);
for c=1:channel
    for i=1:L
        q(i,c)=min(yA(:,c))+(i-1)*delta(c);
    end
end
for j=1:length(yA)
    for k=2:L
        for c=1:channel
            if (yA(j,c)>=q(k-1,c)) && (yA(j,c)<q(k,c))
                if yA(j,c)<((q(k-1,c)+q(k,c))/2)
                    yq(j,c)=q(k-1,c);
                else
                    yq(j,c)=q(k,c);
                end
            end
        end
```

```
    end
end

%compressed and quantized signal plot
subplot(4,1,3);
stairs(ta,yq);
xlabel('Number of samples');
ylabel('Weight');
title('Uniform Quantizatization after A law compression');

%SQNR calculation
errCQ=yq-yA;
errvarCQ=sum(errCQ.*errCQ)/length(errCQ);
ymuvar=sum(yA.*yA)/length(yA);
for c=1:channel
    SQNR_CQ(c)=10*log10(ymuvar(c)/errvarCQ(c));
    fprintf('SQNR(dB) with A law compressor for channel%d =
%f\n',c,SQNR_CQ(c));
end

%%
%sound(yq,Fs);

%% Uniform Quantization without compression
for c=1:channel
    for i=1:L
        qq(i,c)=min(ya(:,c))+(i-1)*delta(c);
    end
end
for j=1:length(ya)
    for k=2:L
        for c=1:channel
            if (ya(j,c)>=qq(k-1,c)) && (ya(j,c)<qq(k,c))
                if ya(j,c)<((qq(k-1,c)+qq(k,c))/2)
                    yqq(j,c)=qq(k-1,c);
                else
                    yqq(j,c)=qq(k,c);
                end
            end
        end
    end
end

%compressed and quantized signal plot
subplot(4,1,4);
stairs(ta,yqq,'k');
xlabel('Number of samples');
ylabel('Weight');
title('Uniform Quantizatization without compression');
%SQNR calculation
errQ=yq-yA;
errvarQ=sum(errQ.*errQ)/length(errQ);
yavar=sum(ya.*ya)/length(ya);
for c=1:channel
    SQNR_Q(c)=10*log10(yavar(c)/errvarQ(c));
    fprintf('SQNR(dB) without compression for channel%d =
%f\n',c,SQNR_Q(c));
end
%sound(yqq,Fs)
```

(ii) Plot

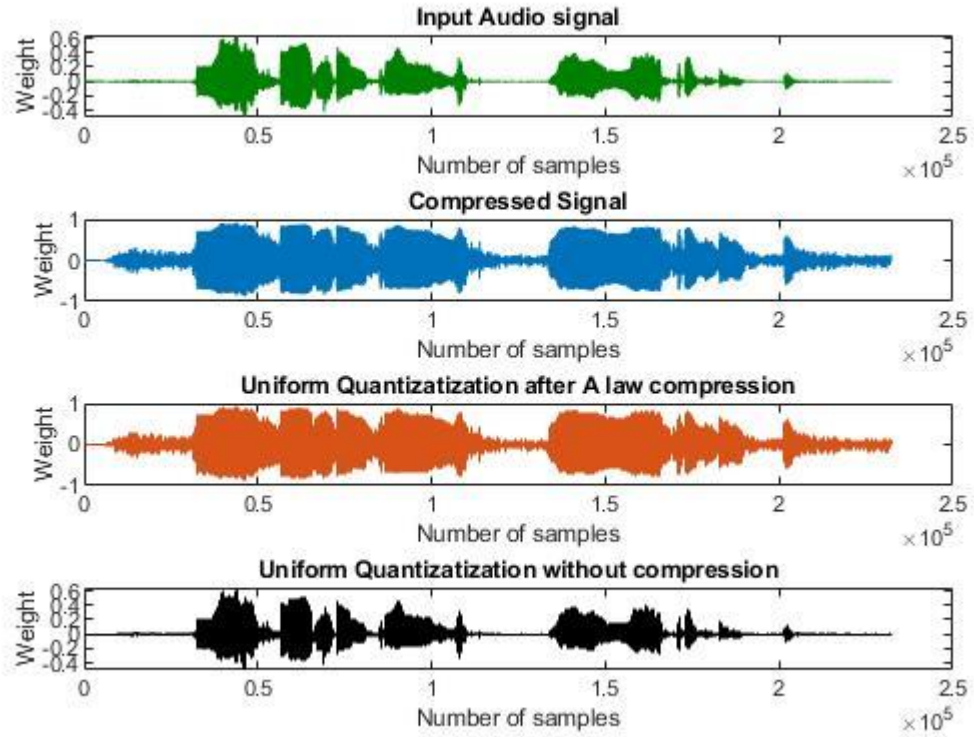


Fig 2.1: Signal plot comparison

```
Command Window
  SQNR(dB) with A law compressor for channel1 = 42.935270
  SQNR(dB) with A law compressor for channel2 = 42.935270
  SQNR(dB) without compression for channel1 = 30.961348
  SQNR(dB) without compression for channel2 = 30.961348
fx >>
```

Fig 2.2: SQNR value output from the command window

Observation:

Similarly for A law compression, compressed quantized signal has less error. Thus, SQNR value is higher for A law compression (42.935 dB) and lower for just uniform quantization (30.961 dB). The sound quality is also better for uniform quantization without any compression.

We used the following formula for A law compression:

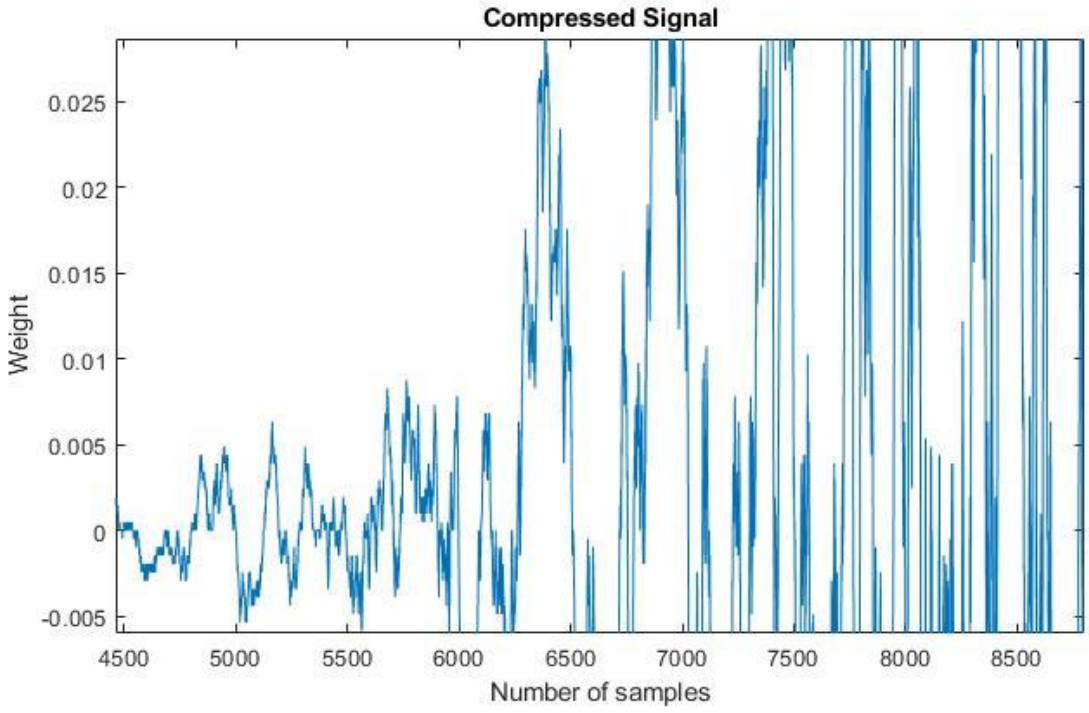
$$\text{Compressed signal, } F(x) = \begin{cases} \frac{1+\ln(A|x|)}{\ln(1+A)}\text{sgn(x)} \; ; \frac{1}{|A|} \leq |x| \leq 1 \\ \frac{A|x|}{\ln(1+A)}\text{sgn(x)}; 0 \leq |x| \leq \frac{1}{|A|} \end{cases}$$

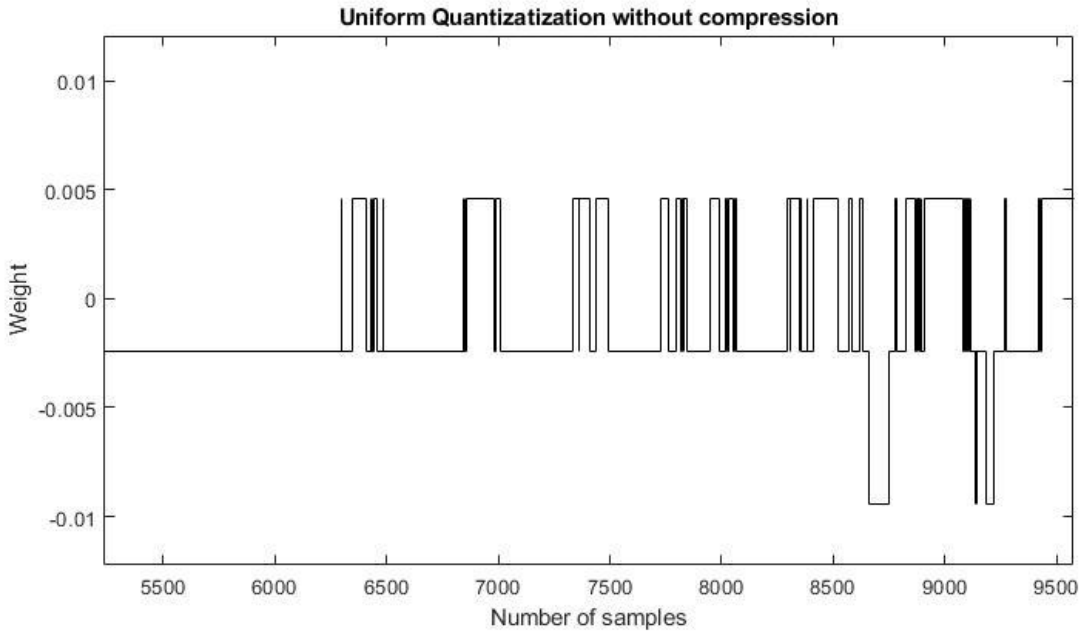
We took A = 87.56

The main difference is created by the fact that during compression, we allocate much more quantization levels to quantize small amplitude components of the signal than just uniform quantization.

Thus, the quantization error is less and so is the quantization noise.

This can easily be seen from the plot:





Discussion:

In problem 1, we experimented with sampling and reconstruction of sampled signal with multiple frequency components. We observed the variation of signal characteristics with sampling frequency. We also observed variation of SQNR with quantization bits.

In problem 2, we observed the effect of compression for uniform quantization for a random audio signal. We used two different kinds of laws to do the compression. We saw that compressed signals were louder, and otherwise very low sounds were heard making too.