EEE 431

Project 1

Report

Outline

Introduction

In this project, uniform and non-uniform quantization are studied. In the first part, standard normally distributed source outputs are handled by both uniform and non-uniform quantizers. In the second part, a speech signal is taken as source and again its behavior over uniform and non-uniform quantizers is examined. In the last part, a 10 level non-uniform quantizer for a typical normally distributed source is designed by Lloyd-Max algorithm.

Work Done

Part I

In the first part, I obtained standard normally distributed random variables truncated between the values of -7 to 7 as source output as follows:

```
pd=makedist('Normal');

t=truncate(pd,-7,7);

X=random(t,1,10^6);
```

1.

The source power is calculated by averaging the samples' squares. It is found as follows:

```
SourcePower =

          1.0002
```

As the source power is equal to the expectation of source outputs' squares and as the source is standard normally distributed here, its square's expectation is estimated as 1. It is theoretically consistent.

2.

Power of the quantization errors can be named as the mean square error (MSE) of the difference between original source outputs and their quantized versions. It is calculated and found as follows:

```
QuantizationErrors(n)=X(n)-quantizedX(n);

MSE=mean(QuantizationErrors.^2);

MSE =

    0.0042
```

where X(n) is the samples of source output and quantizedX(n) is their quantized version.

To plot the histogram of the quantization errors they are normalized by being divided to themselves' norm. The following histogram is obtained:

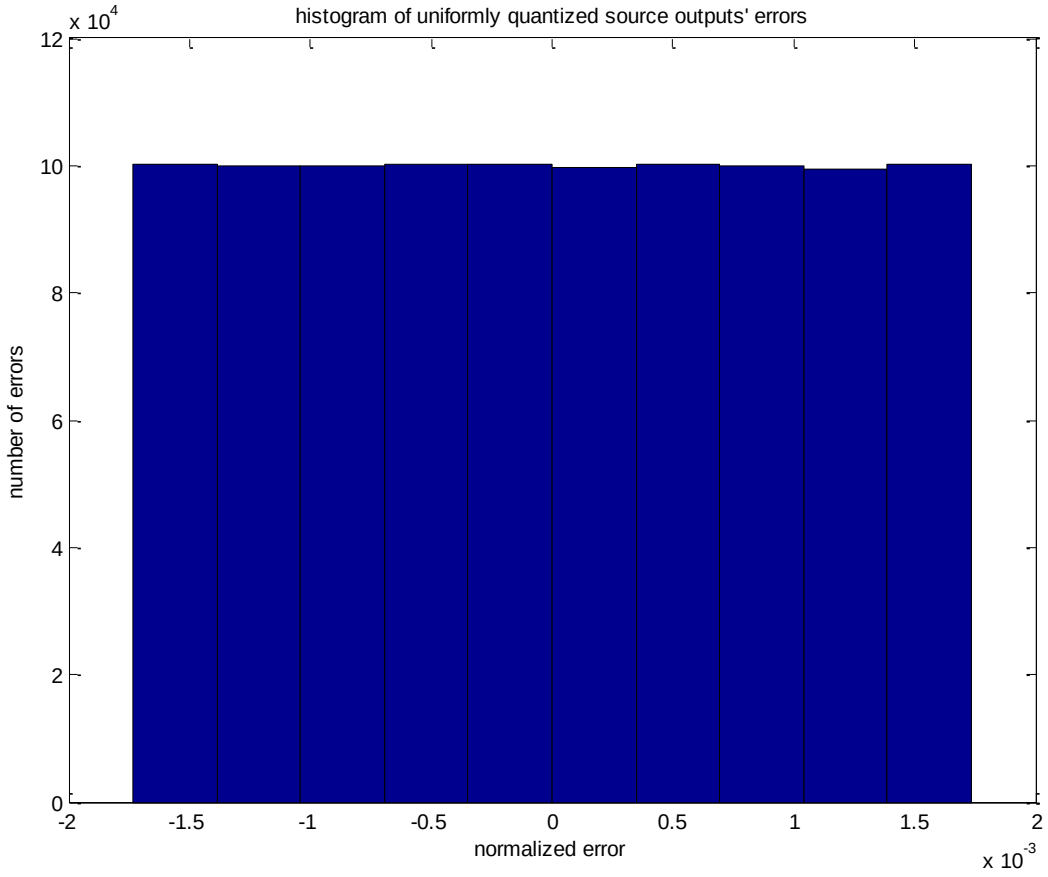

Figure 1: histogram of uniformly quantized source outputs' quantization errors

Here, as uniform quantization is applied to the source output, a uniform distribution among quantization errors are observed. This distribution is consistent with theory as uniform variables have such a PDF graph.

Estimated signal to quantization noise ratio is abbreviated as SQNR. It is the ratio of expectation of source output's square and expectation of quantization errors' squares which means the ratio of source power to mean square error. It is calculated and found as follows:

```
SQNR=ExpectationXsquare/MSE;

SQNR =

  235.3479
```

To express it as in dB the following formula is applied:

```
SQNR_dB=10*log10(SQNR)
```

and the following result is obtained:

```
SQNR_dB =

   23.7171
```

To understand how good a quantizer is it can be checked whether the mean square error is low or SQNR is high. Our MSE and SQNR values here are quite satisfactory.

3.

Here, a non-uniform quantization is made by applying both mu-compander and A-compander. Before quantizing the source output it is compressed by the compander and then the quantized values are expanded. First, 255 is given to the parameter of mu-compander, then 87.6 is given for the A compander. The resulting SQNR values in dB was not satisfactory. At last, 1 is given to the parameter of mu-compander which gives out the following MSE and SQNR values for non-uniform (nu) quantization:

```
nuMSE_mu1 =

    0.0028

nuSQNR_mu1_dB =

   25.5332
```

It can be seen here that the mean square error is decreased and as a result SQNR in dB is increased. This means non-uniform quantization is better for our source output as it improves the SQNR value.

The histogram of the errors of non-uniform quantization here comes as follows:

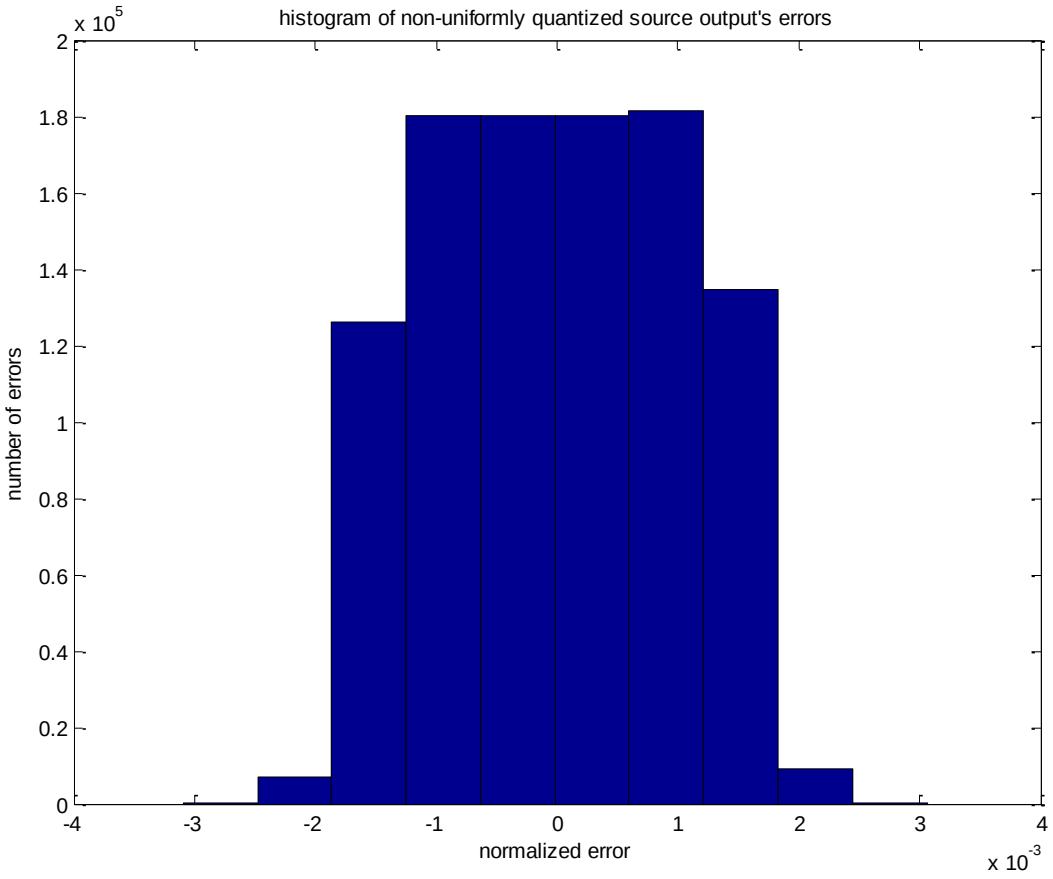

Figure 2: histogram of non-uniformly quantized source outputs' quantization errors

The histogram above displays again normalized errors versus their range. This time the distribution is not uniform. As the non-uniform quantization fits better to the standard normally distributed source outputs, the quantization errors displays a shape that looks more likely to the PDF of the source.

4.

In this step, the quantization level is increased from 64 to 128 and 256 respectively. It is expected to see an improvement in quantization as there are more levels. This means a decrease in mean square errors and an increase in SQNR values. The results meet the expectations. SQNR values obtained for 128 level quantization are as follows:

```
SQNR_dB =

    23.9171

nuSQNR_mu1_dB =

    26.9553
```

And SQNR values obtained for 256 level quantization are as follows:

```
SQNR_dB =

   23.9191

nuSQNR_mu1_dB =

   28.9829
```

As it can be seen above, although there is not a great change in SQNR values, an improvement can be observed.

Part II

In this part, I recorded a speech signal with a greater frequency than Nyquist rate, which is 16 kHz for human voice, to avoid aliasing. To do that the following method is used in MATLAB:

```
recObj=audiorecorder(20*10^3,8,1);%20kHz>16kHz

recordblocking(recObj,50);

Y=getaudiodata(recObj);
```

Here, the parameter 20*10^3 of the audiorecorder function shows the sampling frequency and the parameter 50 of the recordblocking function shows the time that we need for obtaining 10^6 samples. After the recording period, I get the recorded data to Y matrix as seen above.

1.

The plot the normalized histogram of the samples of speech, record matrix is first normalized as follows:

```
normalizedY=Y/norm(Y);
```

Then histogram of normalized Y matrix is drawn.

```
hist(normalizedY);
```

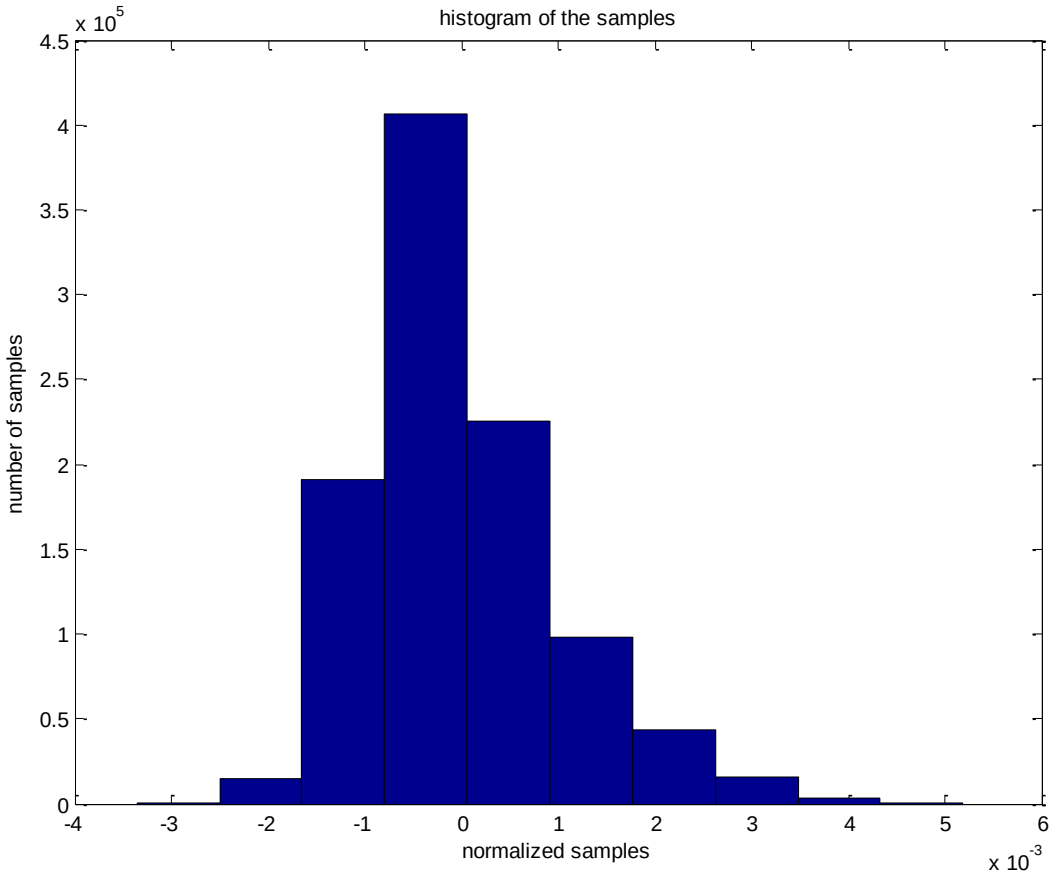

Figure 3: histogram of the samples of recorded speech

The histogram above shows a distribution similar to the standard normal distribution. From that observation the PDF of the speech signal can be estimated as standard normal.

2.

The source power in this part is obtained by the same way as it is obtained in the first part. It is the average of samples' squares and it is found as follows:

```
SourcePower =

    0.0236
```

3.

The quantization error in this question is again calculated by taking the difference of speech signal and its quantized version thorough the sample length. Its normalized histogram comes as follows:

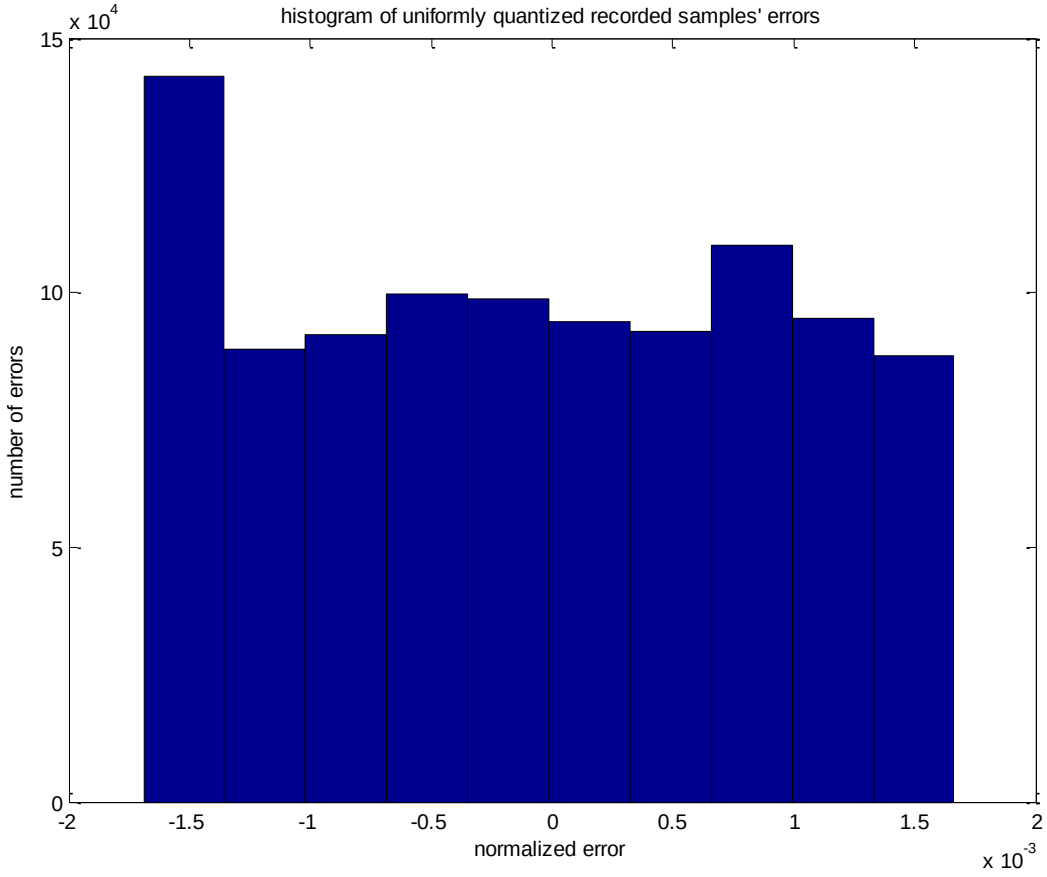

Figure 4: histogram of uniformly quantized recorded samples' quantization errors

Here, again a distribution looking like uniform is seen. This means the uniformly quantized speech signal has a uniform PDF.

SQNR in this question is again the ratio of source power and mean square error. It is found similarly in dB as follows:

```
SQNR_dB =

    24.1320
```

Here, the value 24 is consistent with the theoretical expectations.

4.

Non-uniform quantizer in this question is again made by mu-compander. For the mu parameter of 255 SQNR value does not improve. But the value of 1 gives the expected result. Non-uniform mean square error and SQNR values in this question are as follows:

```
nuMSE_mu1 =

    6.0596e-05

nuSQNR_mu1_dB =

    25.9010
```

It can be seen that mean square error here is dramatically decreased and SQNR is increased which means that non-uniform quantization is a better quantization for speech.

The histogram of the errors for non-uniform quantization is as follows:

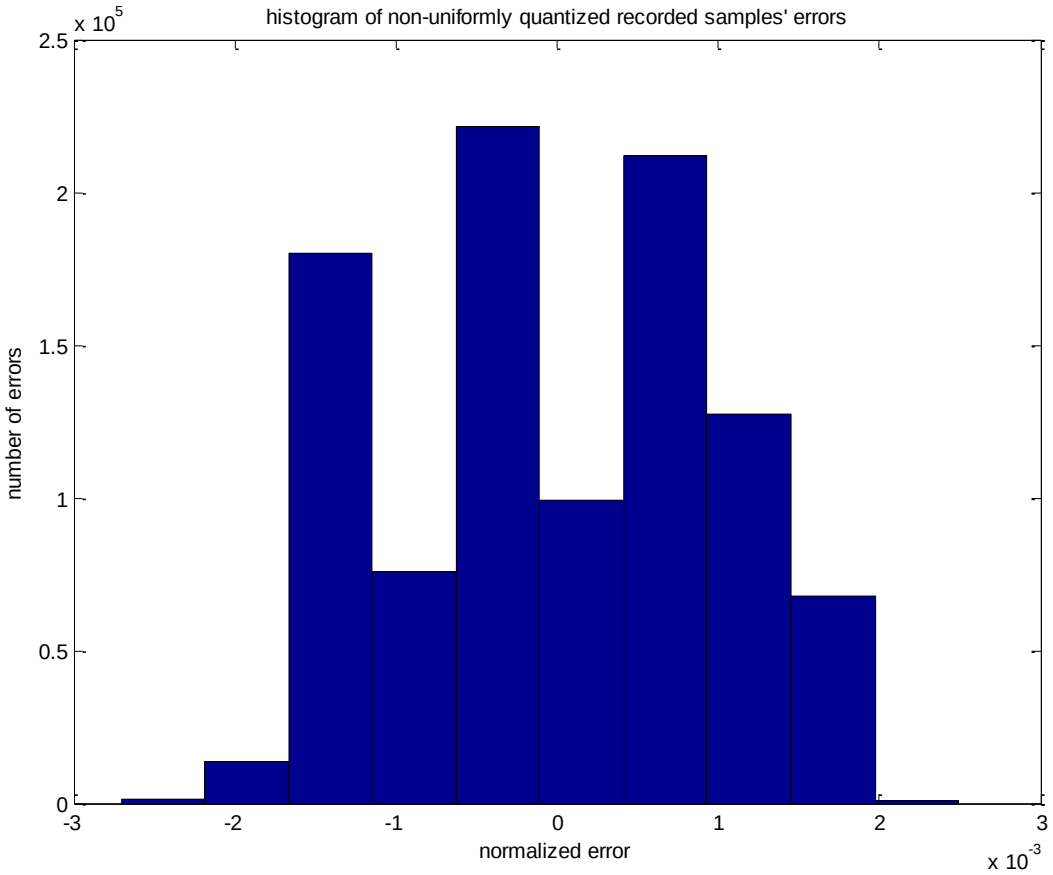

Figure 5: histogram of non-uniformly quantized recorded samples' quantization errors

This histogram also has a non-uniform distribution and the distribution here most likely has the PDF of standard normal.

Part III

In this part a 10 level non-uniform quantizer for a standard normally distributed source is designed by using Lloyd-Max algorithm. The initial boundaries of the quantization regions are selected as -5 and 5 as these values give finite levels. Then the levels in each interval calculated from the formula of the expectation of x given the boundaries of that region. That expectation integrates to the x times PDF of x divided by the probability of x being in that interval from lower boundary of the region to the upper boundary. When the levels are calculated, boundaries are iterated to the x values. This iteration process is continued until the distortion of a randomly produced source output falls below 0.05.

The following boundary and level matrices are obtained as a result:

a=[-1.6847 -1.2635 -0.8423 -0.4211 -7.2555e-12 0.4211 0.8423 1.2635 1.6847]

x=[-2.2243 -1.5407 -1.1004 -0.6602 -0.2200 0.2200 0.6602 1.1004 1.5407 2.2243]

And the resulting distortion is found as follows:

```
D =

    0.0261
```

which is close to the optimal value defined in the text book.

Conclusion

In this project, uniform and non-uniform quantization is worked on in the first two parts. In the first part, mean square error and signal to quantization noise ratio are calculated and PDF of the sources estimated among their error histograms. In the second part, the same thing is done for a speech signal instead of a standard normal source. In the last part, a 10 level non-uniform quantizer is designed by Lloyd-Max algorithm.

Appendix

Part I

```
pd=makedist('Normal');
t=truncate(pd,-7,7);
X=random(t,1,10^6);
%X=linspace(-7,7,10^6);
Y=normpdf(X);
N=256;%64,128,256
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for n=1:(10^6)
    Xsquare(n)=(X(n).^2);
end
squaresum=sum(Xsquare);
SourcePower=squaresum/10^6;
%m=mean(X);

L=linspace(-7,7,N-1);
delta=L(2)-L(1);
for i=1:N
    for i=1:(N-1)
        Q(i)=L(i)-(delta/2);
    end
    Q(64)=7+(delta/2);
end

for n=1:(10^6)
    for i=2:(N-1)
        if X(n)>=L(i-1)&&X(n)<=L(i)
            quantizedX(n)=Q(i);
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for n=1:(10^6)
    QuantizationErrors(n)=X(n)-quantizedX(n);
end

hist(QuantizationErrors/norm(QuantizationErrors));

ExpectationXsquare=SourcePower;
MSE=mean(QuantizationErrors.^2);
SQNR=ExpectationXsquare/MSE
SQNR_dB=10*log10(SQNR)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
comp_mu=compand(X,255,max(X),'mu/compressor');
for n=1:(10^6)
    for i=2:(N-1)
        if comp_mu(n)>=L(i-1)&&comp_mu(n)<=L(i)
            nuquantizedX_mu(n)=Q(i);
        end
    end
end
pand_mu=compand(nuquantizedX_mu,255,max(nuquantizedX_mu),'mu/expander');
```

```
for n=1:(10^6)
    nuQuantizationErrors_mu(n)=X(n)-pand_mu(n);
end

%hold on
hist(nuQuantizationErrors_mu/norm(nuQuantizationErrors_mu));

%ExpectationXsquare=sum(Xsquare.*Y);
nuMSE_mu=mean(nuQuantizationErrors_mu.^2);
nuSQNR_mu=ExpectationXsquare/nuMSE_mu
nuSQNR_mu_dB=10*log10(nuSQNR_mu)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
comp_A=compand(X,87.6,max(X),'A/compressor');
for n=1:(10^6)
    for i=2:(N-1)
        if comp_A(n)>=L(i-1)&&comp_A(n)<=L(i)
            nuquantizedX_A(n)=Q(i);
        end
    end
end
pand_A=compand(nuquantizedX_A,87.6,max(nuquantizedX_A),'A/expander');

for n=1:(10^6)
    nuQuantizationErrors_A(n)=X(n)-pand_A(n);
end

%hold on
hist(nuQuantizationErrors_A/norm(nuQuantizationErrors_A));

%ExpectationXsquare=sum(Xsquare.*Y);
nuMSE_A=mean(nuQuantizationErrors_A.^2);
nuSQNR_A=ExpectationXsquare/nuMSE_A
nuSQNR_A_dB=10*log10(nuSQNR_A)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
comp_mu1=compand(X,1,max(X),'mu/compressor');
for n=1:(10^6)
    for i=2:(N-1)
        if comp_mu1(n)>=L(i-1)&&comp_mu1(n)<=L(i)
            nuquantizedX_mu1(n)=Q(i);
        end
    end
end
pand_mu1=compand(nuquantizedX_mu1,1,max(nuquantizedX_mu1),'mu/expander');

for n=1:(10^6)
    nuQuantizationErrors_mu1(n)=X(n)-pand_mu1(n);
end

%hold on
hist(nuQuantizationErrors_mu1/norm(nuQuantizationErrors_mu1));

%ExpectationXsquare=sum(Xsquare.*Y);
nuMSE_mu1=mean(nuQuantizationErrors_mu1.^2);
nuSQNR_mu1=ExpectationXsquare/nuMSE_mu1
nuSQNR_mu1_dB=10*log10(nuSQNR_mu1)
```

Part II

```
%recObj=audiorecorder(20*10^3,8,1);%20kHz>16kHz
%recordblocking(recObj,50);
Y=getaudiodata(recObj);
Y=transpose(Y);
plot(Y);
normalizedY=Y/norm(Y);
%Y=normalizedY;
hist(normalizedY);
N=64;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for n=1:(10^6)
    Ysquare(n)=(Y(n).^2);
end
squaresum=sum(Ysquare);
SourcePower=squaresum/10^6;
%m=mean(Y.^2);

L=linspace(-1,1,N-1);
delta=L(2)-L(1);
for i=1:N
    for i=1:(N-1)
        Q(i)=L(i)-(delta/2);
    end
    Q(64)=1+(delta/2);
end

for n=1:(10^6)
    for i=2:(N-1)
        if Y(n)>=L(i-1)&&Y(n)<L(i)
            quantizedY(n)=Q(i);
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for n=1:(10^6)
    QuantizationErrors(n)=Y(n)-quantizedY(n);
end

hist(QuantizationErrors/norm(QuantizationErrors));

ExpectationYsquare=SourcePower;
MSE=mean(QuantizationErrors.^2);
SQNR=ExpectationYsquare/MSE
SQNR_dB=10*log10(SQNR)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
comp_mu=compand(Y,255,max(Y),'mu/compressor');
for n=1:(10^6)
    for i=2:(N-1)
        if comp_mu(n)>=L(i-1)&&comp_mu(n)<=L(i)
            nuquantizedY_mu(n)=Q(i);
        end
    end
end
```

```
pand_mu=compand(nuquantizedY_mu,255,max(nuquantizedY_mu),'mu/expander');

for n=1:(10^6)
    nuQuantizationErrors_mu(n)=Y(n)-pand_mu(n);
end

%hold on
hist(nuQuantizationErrors_mu/norm(nuQuantizationErrors_mu));

%ExpectationYsquare=sum(Ysquare.*normpdf(Y));
nuMSE_mu=mean(nuQuantizationErrors_mu.^2);
nuSQNR_mu=ExpectationYsquare/nuMSE_mu
nuSQNR_mu_dB=10*log10(nuSQNR_mu)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
comp_mu1=compand(Y,1,max(Y),'mu/compressor');
for n=1:(10^6)
    for i=2:(N-1)
        if comp_mu1(n)>=L(i-1)&&comp_mu1(n)<=L(i)
            nuquantizedY_mu1(n)=Q(i);
        end
    end
end
pand_mu1=compand(nuquantizedY_mu1,1,max(nuquantizedY_mu1),'mu/expander');

for n=1:(10^6)
    nuQuantizationErrors_mu1(n)=Y(n)-pand_mu1(n);
end

%hold on
hist(nuQuantizationErrors_mu1/norm(nuQuantizationErrors_mu1));

%ExpectationYsquare=sum(Ysquare.*normpdf(Y));
nuMSE_mu1=mean(nuQuantizationErrors_mu1.^2);
nuSQNR_mu1=ExpectationYsquare/nuMSE_mu1
nuSQNR_mu1_dB=10*log10(nuSQNR_mu1)
```

Part III

```
pd=makedist('Normal');
t=truncate(pd,-5,5);
X=random(t,1,10);

%fun1=@(v) v.*(normpdf(v)./(0.5/sqrt(2).*erfc(v/sqrt(2))));
fun1=@(v) v.*normpdf(v);
%fun2=@(v) v.*(normpdf(v)./(0.5/sqrt(2).*erf(v/sqrt(2))));
fun2=@(v) v.*normpdf(v);
bound=[-5 5];

for m=1:10
a=linspace(bound(1),bound(2),9);
for i=1:10
    for i=2:5
        x(i)=0.75*(1/((0.5/sqrt(2).*erfc(a(i-1)/sqrt(2)))-
(0.5/sqrt(2).*erfc(a(i)/sqrt(2))))).*integral(fun1,a(i-1),a(i));
    end
```

```
    for i=6:9
        x(i)=0.75*(1/((0.5/sqrt(2).*erf(a(i)/sqrt(2)))-(0.5/sqrt(2).*erf(a(i-
1)/sqrt(2))))).*integral(fun2,a(i-1),a(i));
    end
    x(1)=0.75*(1/((0.5/sqrt(2).*erfc(-inf/sqrt(2)))-
(0.5/sqrt(2).*erfc(a(1)/sqrt(2))))).*integral(fun1,-inf,a(1));
    x(10)=0.75*(1/((0.5/sqrt(2).*erf(inf/sqrt(2)))-
(0.5/sqrt(2).*erf(a(9)/sqrt(2))))).*integral(fun2,a(9),inf);
end
bound(1)=x(2);
bound(2)=x(9);

for n=1:10
for k=2:9
    if X(n)>=a(k-1)&&X(n)<a(k)
        quantizedX(n)=x(k);
    end
end
QuantizationErrors(n)=X(n)-quantizedX(n);
end
D=mean(QuantizationErrors.^2);
if D>0.05
    continue
else
end
end
```