Ψηφιακές Τηλεπικοινωνίες
1ο Σετ Ασκήσεων

Ακαδημαϊκό Έτος 2022-2023

Κόλλιας Ιωάννης 1064886

Μέρος 1

Για τη μη ομοιόμορφη κβάντιση του διανύσματος εισόδου θα πρέπει να ορίσουμε τη συνάρτηση *Lloyd-Max* στο περιβάλλον της MATLAB.

Lloyd_Max.m

```
%%Lloyd-Max
function [xq, kentra, delta] = Lloyd_Max(X, N, min_value, max_value)
count=0;
for q=1:N %%number of bits
  q_levels = 2^N;

    minv = min_value;
    maxv = max_value;

    len=(-1*minv+maxv)/q_levels;
    m=zeros(1,q_levels+1);
    for i=1:q_levels+1
        m(i)=minv+(i-1)*len;
    end
    sig=X;
    sig=sort(sig);
    lu=zeros(1,q_levels);
    sk=zeros(1,q_levels);

    for i=2:1000
        for k=1:q_levels
            [sk(k),lu(k)]=cent(m(k),m(k+1),sig,k,q_levels);
            kentra(k)=sk(k)/lu(k);
        end

        for k=2:q_levels
            m(k)=(kentra(k-1)+kentra(k))/2;
        end

        for h=1:q_levels
            for t=1:1:length(X)
                if(X(t)<m(h+1) && X(t)>=m(h))
                    xq(t)=kentra(h);
                end
            end
        end

        e=10^(-6);

        a=X-xq';
        b=a.^2;
        mse1(i)=sum(b)/length(X);

        if (abs(mse1(i)-mse1(i-1))<e)
            delta = mse1(i-1);
            break;
        end
        delta=mse1(i);
    end
end
end
```

Όπου ορίζω για τη λειτουργία του και τις συναρτήσεις **mew** και **cent.**

cent.m

```
function [ sk,lu ] = cent( mint,mfin,sig,k,levels )
lu=0;
sk=0;
for j=1:10000
    if(sig(j)<mfin && sig(j)>=mint)
        lu=lu+1;
        sk=sk+sig(j);
    end
end
if lu==0
    sk=(mint+mfin)/2;
    lu=1;
end
end
```

mew.m

```
function [ mu ] = mew( min,max )
ts=0.01;
intn = 0;
intd = 0;
for i=min:ts:max
    intn = intn+ts*i*exp(-i*i);
    intd = intd+ts*exp(-i*i);
end
mu = intn/intd;
end
```

Για τις απαντήσεις των ερωτημάτων του 1ου μέρους (κωδικοποίηση της πηγής Α αλλά και του αρχείου εικόνας) φτιάξαμε ένα **main** αρχείο που θα συγκεντρώσει τις απαντήσεις του πρώτου αλλά και του δεύτερου ερωτήματος.

main.m

```
%1ο ΜΕΡΟΣ - ΑΠΑΝΤΗΣΕΙΣ ΕΡΩΤΗΜΑΤΩΝ 1.1, 1.2
clc;
clear;
close all;

fprintf('\nΜΕΡΟΣ 1 - ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΡΩΤΗΜΑΤΟΣ 1.1\n');
[s1,s2] = srcA(10000);
[SQNR_sound_ADM_1_2, SQNR_sound_PCM_1_2, SQNR_sound_ADM_2_2,
SQNR_sound_PCM_2_2]= solution1_1(s1,s2,2);
[SQNR_sound_ADM_1_4, SQNR_sound_PCM_1_4, SQNR_sound_ADM_2_4,
SQNR_sound_PCM_2_4]= solution1_1(s1,s2,4);
[SQNR_sound_ADM_1_8, SQNR_sound_PCM_1_8, SQNR_sound_ADM_2_8,
SQNR_sound_PCM_2_8]= solution1_1(s1,s2,8);
fprintf('\nΜΕΡΟΣ 1 - ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΡΩΤΗΜΑΤΟΣ ΜΕΡΟΣ 1.2 (Εντροπία Ήχου)\n');
[sound_entropy1]=solution1_2(s1);
[sound_entropy2]=solution1_2(s2);

fprintf('\nΜΕΡΟΣ 1 - ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΡΩΤΗΜΑΤΟΣ 2.1\n');
[SQNR_photo_PCM_2, SQNR_photo_PCM_4]= solution2_1();
fprintf('\nΜΕΡΟΣ 1 - ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΡΩΤΗΜΑΤΟΣ 2.2 (Εντροπία Εικόνας)\n');
[image_entropy]= solution2_2('cameraman.mat');
```

Ερώτημα 1

Αρχικά φτιάξαμε τις δύο διαδικασίες AR(1) και AR(2) όπου πηγές s_1 και s_2 για α_1=0.9 και α_1=0.01 αντίστοιχα.

srcA.m

```
function [s1 ,s2] = srcA(L)

x = randn(L,1);
s1 = filter(1,[1, -0.9],x);
s2 = filter(1,[1, -0.01],x);
```

Ερώτημα 1.1

Για να απαντήσουμε το ερώτημα αυτό εκτελούμε διαδοχικά τον κώδικα στο script solution1_1 και είσοδο βάζουμε τα bits κβάντισης που επιθυμούμε σε κάθε μια περίπτωση. Η συγκεκριμένη συνάρτηση μας υπολογίζει το SQNR τόσο στο ADM όσο και στο PCM σύστημα. Ο κώδικας για το script δίνεται παρακάτω:

solution1_1.m

```
function [SqnrAdm1, SqnrPcm1, SqnrAdm2, SqnrPcm2]= solution1_1(s1,s2,bits)
    SqnrAdm1 = admquantizer(s1,bits);
    SqnrPcm1 = pcmquantizer(s1,bits);
    SqnrAdm2 = admquantizer(s2,bits);
    SqnrPcm2 = pcmquantizer(s2,bits);
End
```

admquantizer.m

```
function [ADM]= admquantizer(s,N)

fprintf('\nΑποτελέσματα κβάντισης ADM %d bits\n',N);

k=ceil(N);
if k<4
   k=4;
end

s=interp(s,k);
if (nargin==2)
    samples=length(s);
    h=1;
end
if (nargin==3)
    h=1;
end

s=s(1:samples);

StepSize=1/(5*k);


[a,cn]=adm_encoder(s, StepSize);

Sn=adm_decoder(a,cn);
Sa=filter(100, .1, Sn);
e=s'-Sa;

figure
subplot(4,1,1)
```

```
plot(s,'k');
title(['Σήμα Εισόδου ADM-' num2str(N)]);

subplot(4,1,2)
plot(Sa,'b');
title(['Σήμα Εξόδου ADM-' num2str(N)]);

Ps=sum(s(1:samples).^2)/samples;
Pn=sum(e(1:samples).^2)/samples;
ADM=10*log10(Ps/Pn);
fprintf('\t\tΙσχύς Σήματος(S):\t%f\n',Ps);
fprintf('\t\tΙσχύς Θορύβου(N):\t%f\n',Pn);
fprintf('\t\tSQNR ADM-%d:\t%f[dB]\n',N,ADM);
end
```

adm_decoder.m

```
function [Sn]=adm_decoder(StepSizeArray, cn)
samples = length(cn);
samples = length(cn)
output(1) = 0;
for i=1:samples
   if(cn(i)==1)
      output(i+1) = output(i) + StepSizeArray(i);
  elseif (cn(i)==-1)
      output(i+1) = output(i) - StepSizeArray(i);
   end
end

Sn=output(2:samples+1);
end
```

LPF.m

```
function Sa=LPF(tap, cf, Sn)
b=fir1(tap,cf);
Sa = conv2(Sn,b,'same');
end
```

adm_encoder.m

```
function [StepSizeArray,cn]=adm_encoder(x, StepSize)
samples = length(x);
accum(1) = 0;
StepSizeArray(1)=StepSize;
StepSizeArray(2)=StepSize;
for i=1:samples
    if(x(i)>=accum(i))
        output(i)=1;
            if i>1
        if output(i)==output(i-1)
            StepSizeArray(i)=1.5*StepSizeArray(i-1);
        else
            StepSizeArray(i)=(1.5^-1)*StepSizeArray(i-1);
        end
            end
        accum(i+1) = accum(i) + output(i) * StepSizeArray(i);
    else
        output(i)=-1;
            if i>1
        if output(i)==output(i-1)
            StepSizeArray(i)=1.5*StepSizeArray(i-1);
```

```
        else
            StepSizeArray(i)=(1.5^-1)*StepSizeArray(i-1);
        end
            end
        accum(i+1) = accum(i) + output(i) * StepSizeArray(i);
    end
end
cn = output;
end
```

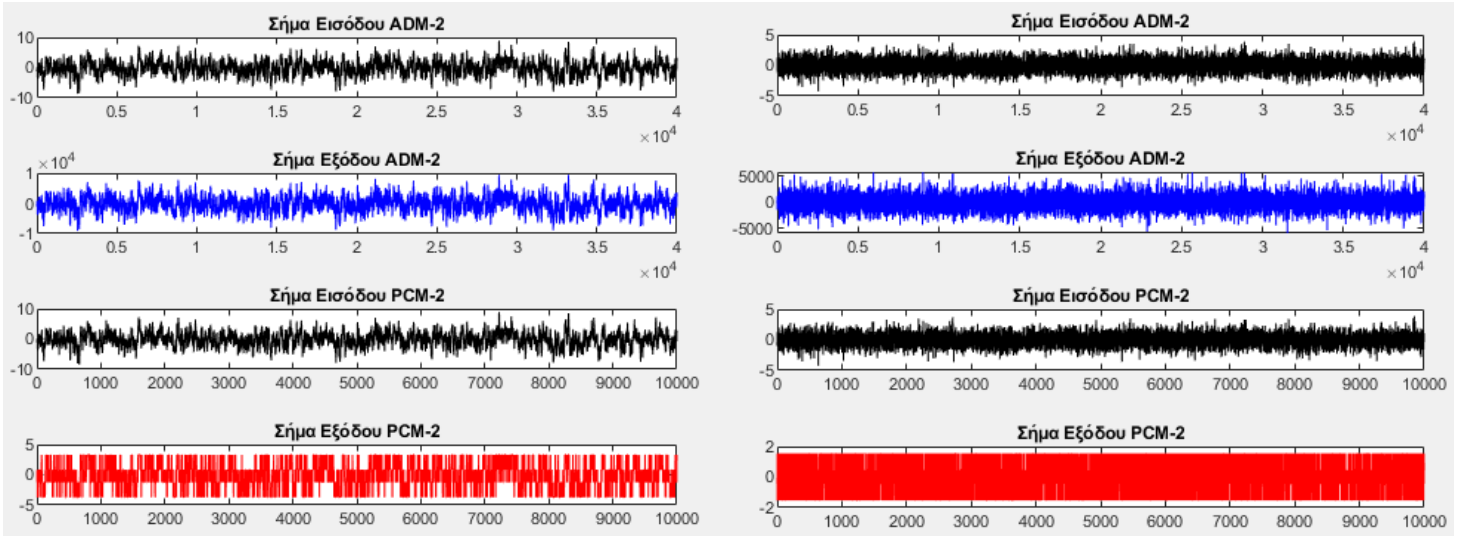
pcmquantizer.m

```
function [PCM]=pcmquantizer(x1,N)

fprintf('\nΑποτελέσματα κβάντισης PCM %d bits\n',N);

if (nargin==2)
    k=1;
end

N=ceil(N);

if nargin==2
    samples=length(x1)
end


lx1=samples;

min_value = min(x1);
max_value = max(x1);

    out=[];
        [xq, centers, MSE] = Lloyd_Max(x1, N, min_value, max_value);
       for i=1:1:lx1
              out(i)=xq(i);
        end
       out=out(:);
  lo = length(out);

rec_out = out;
rec_out=rec_out-mean(rec_out);
lxq=length(xq);
nxq=0:lxq-1;
e=x1-xq(1:lx1);
ne=0:length(e)-1;


 if(N==2 || N==4 || N==8 )

    subplot(4,1,3)
    plot(x1,'k');
    title(['Σήμα Εισόδου PCM-' num2str(N)]);


    subplot(4,1,4)
    plot(xq,'r')
     title(['Σήμα Εξόδου PCM-' num2str(N)]);
     tmpstr=['PCM_',num2str(N)];
 end
```

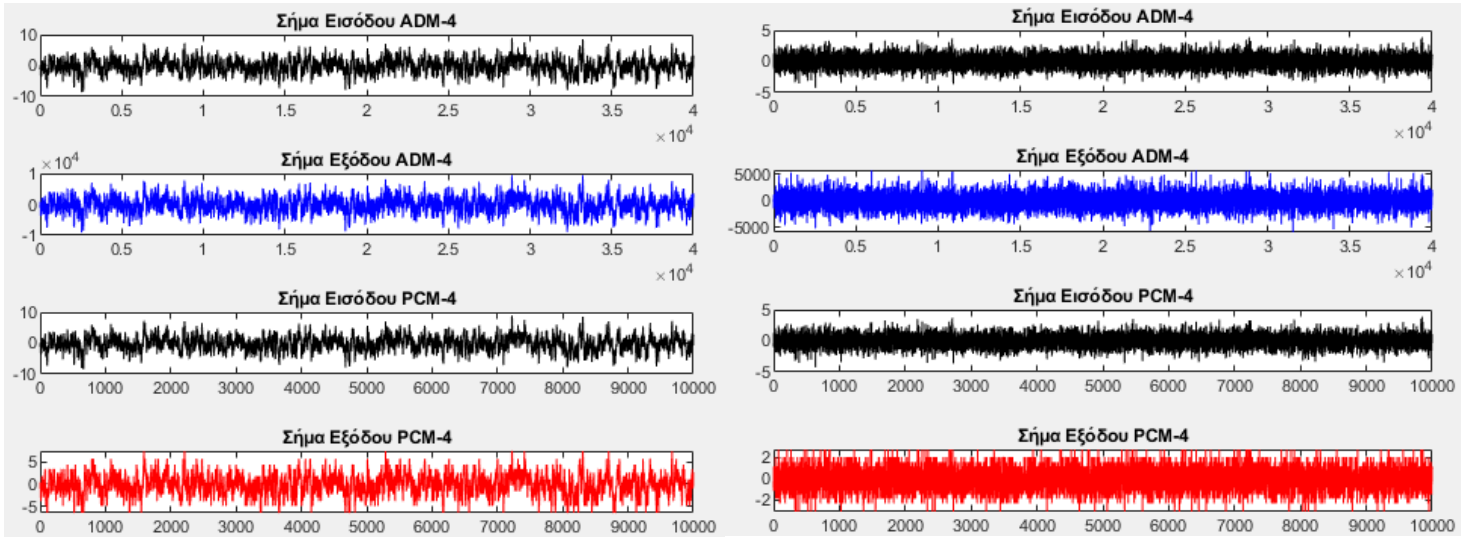
```
    Ps = mean(x1.^2);
    Pn = mean((x1-out).^2);
    PCM = 10*log10(Ps./Pn);

fprintf('\t\tIσχύς Σήματος(S):\t%f\n',Ps);
fprintf('\t\tIσχύς Θορύβου(N):\t%f\n',Pn);
fprintf('\t\tSQNR PCM-%d:\t%f[dB]\n',N,PCM);
end
```

Στις εικόνες που ακολουθούν για κάθε N στο αριστερό μέρος έχουμε την πηγή με α_1=0.9 ενώ στο δεξιό α_2=0.01.

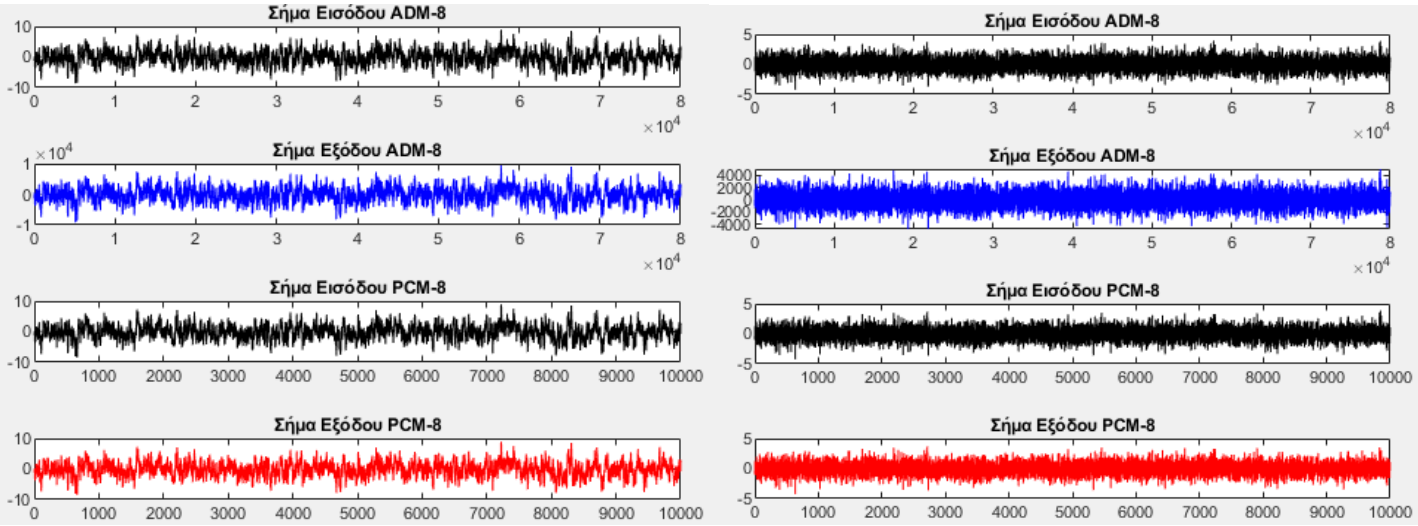
Για N=2bits.



Για N=4bits.



Για N=8bits.



Με τα αποτελέσματα του SQNR να είναι:

ΓΙΑ ΤΗΝ S1:

Αποτελέσματα κβάντισης ADM 2 bits

samples =40000

Ισχύς Σήματος(S): 5.538352

Ισχύς Θορύβου(N): 5675827.060803

SQNR ADM-2: -60.106486[dB]

Αποτελέσματα κβάντισης PCM 2 bits

samples =10000

Ισχύς Σήματος(S): 5.575666

Ισχύς Θορύβου(N): 0.650753

SQNR PCM-2: 9.328809[dB]

Αποτελέσματα κβάντισης ADM 4 bits

samples =40000

Ισχύς Σήματος(S): 5.538352

Ισχύς Θορύβου(N): 5675827.060803

SQNR ADM-4: -60.106486[dB]

Αποτελέσματα κβάντισης PCM 4 bits

samples =10000

Ισχύς Σήματος(S): 5.575666

Ισχύς Θορύβου(N): 0.053495

SQNR PCM-4: 20.179803[dB]

Αποτελέσματα κβάντισης ADM 8 bits

samples =80000

Ισχύς Σήματος(S): 5.537488

Ισχύς Θορύβου(N): 5587149.910110

SQNR ADM-8: -60.038775[dB]

Αποτελέσματα κβάντισης PCM 8 bits

samples =10000

Ισχύς Σήματος(S): 5.575666

Ισχύς Θορύβου(N): 0.000363

SQNR PCM-8: 41.867667[dB]

ΓΙΑ ΤΗΝ S2:

Αποτελέσματα κβάντισης ADM 2 bits

samples =40000

Ισχύς Σήματος(S): 0.902985

Ισχύς Θορύβου(N): 1218837.575077

SQNR ADM-2: -61.302651[dB]

Αποτελέσματα κβάντισης PCM 2 bits

samples =10000

Ισχύς Σήματος(S): 1.023844

Ισχύς Θορύβου(N): 0.123625

SQNR PCM-2: 9.181265[dB]

Αποτελέσματα κβάντισης ADM 4 bits

samples =40000

Ισχύς Σήματος(S): 0.902985

Ισχύς Θορύβου(N): 1218837.575077

SQNR ADM-4: -61.302651[dB]

Αποτελέσματα κβάντισης PCM 4 bits

samples =10000

Ισχύς Σήματος(S): 1.023844

Ισχύς Θορύβου(N): 0.011382

SQNR PCM-4: 19.540280[dB]

Αποτελέσματα κβάντισης ADM 8 bits

samples =80000

Ισχύς Σήματος(S): 0.900083

Ισχύς Θορύβου(N): 998927.919627

SQNR ADM-8: -60.452516[dB]

Αποτελέσματα κβάντισης PCM 8 bits

samples =10000

Ισχύς Σήματος(S): 1.023844

Ισχύς Θορύβου(N): 0.001586

SQNR PCM-8: 28.099236[dB]

Αυτό που παρατηρούμε και στις 2 περιπτώσεις είναι πως όσο μεγαλώνει η τιμή του N τόσο καλύτερη ανάλυση της πληροφορίας έχουμε επομένως και καλύτερα αποτελέσματα σ' ένα μη ομοιόμορφο κβαντιστή. Στη περίπτωση ενός συστήματος ADM, ακόμα και με χαμηλό N επιτυγχάνεται καλύτερο άκουσμα.

Ερώτημα 1.2

Για τον υπολογισμό της εντροπίας σ' ένα PCM σύστημα υλοποιήθηκε το script **solution1_2.m**.

solution1_2.m

```
function [H] = solution1_2(s)
       H=0;
       count=1;
       while(2^count<=8)
         signal = s;
         [xq, centers, D]=Lloyd_Max(signal,2^count,min(signal),max(signal));
         for j=1:2^(2^count)
             POS=sum((xq==centers(j)))/length(signal);
             if (POS~=0)
                 H= H +(-POS*log2(POS));
             end
         end
         fprintf('Source-A [%d bits quantization] = %f \t\n',2^count,H);
         count=count+1;
       end
```

Με τα αποτελέσματα στην έξοδο να είναι:

ΓΙΑ ΤΗΝ S1:

Source-A [2 bits quantization] = 1.926746

Source-A [4 bits quantization] = 5.613605

Source-A [8 bits quantization] = 12.765530

ΓΙΑ ΤΗΝ S2:

Source-A [2 bits quantization] = 1.907351

Source-A [4 bits quantization] = 5.624723

Source-A [8 bits quantization] = 12.644826

Ομοίως με την περίπτωση του SQNR παρατηρούμε πως όσο μεγαλώνει το N, μεγαλώνει και η εντροπία.

Ερώτημα 2

Ερώτημα 2.1

Για την εύρεση του SQNR σε ένα σύστημα PCM που πηγή εισόδου έχει το αρχείο εικόνας cameraman.mat για N=2, N=4 bits κβάντισης υλοποιήθηκε το παρακάτω script.

solution2_1.m

```
function [PcmSqnr_2, PcmSqnr_4] = solution2_1()
    [PcmSqnr_2, PcmSqnr_4] = pcmquantizer2();
end
```

pcmquantizer2.m

```
function [SQNR_2, SQNR_4]=pcmquantizer2()

i=load('cameraman.mat');
i=cell2mat(struct2cell(i));
x=i(:);
x=(x-128)/128;

figure
subplot(3,1,1);
imshow(uint8(i))
title('ΧΩΡΙΣ PCM');


[xq centers D]=Lloyd_Max(x,2,min(x),max(x));

 xnew=xq*128+128;
 i_after=reshape(xnew,256,256);
 subplot(3,1,2);
 imshow(uint8(i_after));
 title('2-PCM');

 SQNR_2 = 10*log10(mean(x.^2)/mean((x-xq').^2));
 fprintf('PCM-SQNR(db) [%d bits quantization]= %f\n',2,SQNR_2);

 [xq centers D]=Lloyd_Max(x,4,min(x),max(x));

 xnew=xq*128+128;
 i_after=reshape(xnew,256,256);
 subplot(3,1,3);
 imshow(uint8(i_after));
 title('4-PCM');
 SQNR_4 = 10*log10(mean(x.^2)/mean((x-xq').^2));

 fprintf('PCM-SQNR(db) [%d bits quantization]= %f\n',4,SQNR_4);

end
```

ΧΩΡΙΣ PCM

2-PCM

4-PCM

Στην εικόνα αριστερά φαίνονται τα τρία στάδια επεξεργασίας που υπεβλήθη η εικόνα **cameraman.mat.**

Εκτελώντας N=2bits λάβαμε την ανάλυση της εικόνας με τίτλο 2-PCM και SQNR σε dB.

PCM-SQNR(db) [2 bits quantization]= 10.617080

Ενώ εκτελώντας N=4bits λάβαμε την ανάλυση της εικόνας με τίτλο 4-PCM και SQNR σε dB.

PCM-SQNR(db) [4 bits quantization]= 23.458592

Παρατηρούμε ότι και στην πηγή εισόδου του ήχου έτσι και στην περίπτωση της εικόνας που η πηγή μας είναι διακριτή φαίνεται πως όσο μεγαλώνει η τιμή του Ν τόσο καλύτερη ποιότητα ανάλυσης έχουμε.

Ερώτημα 2.2

Για τον υπολογισμό της εντροπίας σε ένα PCM σύστημα το οποίο λαμβάνει ως σήμα εισόδου το αρχείο εικόνας και πραγματοποιεί ανάλυση για N=2, N=4 bits υλοποιήθηκε το script **solution2_2.m**

solution2_2.m

```
function [H_image] = solution2_2(image)
             i=load (image);
             i=cell2mat(struct2cell(i));
             photo=i(:);
             photo=(photo-128)/128;
             H_image=0;
             count=1;
      while(2^count<=4)

      [xq, centers, D]=Lloyd_Max(photo,2^count,min(photo),max(photo));
                     for j=1:2^(2^count)
                             POS=sum((xq==centers(j)))/length(photo);
                                if (POS~=0)
                                       H_image= H_image +(-POS*log2(POS));
                                end
                     end
      fprintf('Source-B[%d bits quantization] = %f \t\n',2^count,H_image);
      count=count+1;

      end
end
```

Με τις εξόδους να είναι:

Source-B[2 bits quantization] = 1.487349

Source-B[4 bits quantization] = 4.682383

Έτσι παρατηρούμε και εδώ πως όσο μεγαλώνει η τιμή των bits κβάντισης, τόσο μεγαλύτερη εντροπία παρουσιάζεται.

Μέρος 2

Για τις απαντήσεις του 2ου μέρους φτιάχτηκε το script **main2.m.**

```
%2ο ΜΕΡΟΣ - ΑΠΑΝΤΗΣΕΙΣ ΓΙΑ SER/BER 4-PAM ΚΑΙ 8-PAM
clear all;
sol_PAM();
```

Το ομόδυνο σύστημα ζωνοπερατού PAM παράγει κυματομορφές που αντιπροσωπεύουν σύμβολα, καθένα από τα οποία έχει συγκεκριμένο πλάτος «A» σε δεδομένη χρονική στιγμή. Τα σύμβολα αυτά μεταφέρουν ένα ορισμένο ποσό ενέργειας με βάση τη συνάρτηση παλμού που χρησιμοποιείται για τον ορισμό τους. Όταν η πληροφορία μεταδίδεται μέσω του καναλιού ως ψηφιακό σήμα, αποδιαμορφώνεται στο άκρο του δέκτη ώστε να είναι κατανοητά τα αρχικά σύμβολα. Το πρότυπο για τα συστήματα M-PAM χρησιμοποιείται ως βάση για την εφαρμογή συγκεκριμένων συστημάτων 4-PAM και 8-PAM. Το μέγεθος, M, αναφέρεται στον αριθμό των διακριτών συμβόλων στο σύστημα (και δεν είναι το ίδιο) με 4 ή 8.

Ο κώδικας **init.m** ορίζει διάφορες μεταβλητές που σχετίζονται με τον χρονισμό ενός σήματος σε ένα σύστημα PAM.

init.m

```
%symbol period
T_symbol=40;
%sampling period
T_sample=1;
%duration of one period of the carrier frequency
Tc=4;
%carrier frequency
fc=1/Tc;
%rectangular pulse
g=sqrt(2/T_symbol);
%amplitude of the signal
A=1/sqrt(5);
```

Ο κώδικας **g_symbols.m** ορίζει δύο διαφορετικά σύνολα συμβόλων για δύο διαφορετικά συστήματα PAM, 4-PAM και 8-PAM.

- Εάν η μεταβλητή "M" έχει οριστεί ως 4, ο κώδικας δημιουργεί έναν πίνακα που ονομάζεται "symbol" και περιέχει τέσσερις γραμμές, καθεμία από τις οποίες αντιπροσωπεύει ένα διαφορετικό σύμβολο στο σύστημα 4-PAM. Τα σύμβολα αυτά αντιπροσωπεύονται από τις δυαδικές τιμές [00,01,10,11].
- Εάν η μεταβλητή "M" έχει οριστεί ως 8, ο κώδικας δημιουργεί έναν άλλο πίνακα που ονομάζεται "symbol" και περιέχει οκτώ γραμμές, καθεμία από τις οποίες αντιπροσωπεύει ένα διαφορετικό σύμβολο στο σύστημα 8-PAM. Τα σύμβολα αυτά αντιπροσωπεύονται από τις δυαδικές τιμές [000,001,010,011,100,101,110,111]

Σε κάθε περίπτωση, αυτός ο πίνακας συμβόλων θα χρησιμοποιηθεί για τη δημιουργία του ψηφιακού σήματος στην πλευρά του πομπού.

g_symbols.m

```
% 4-PAM symbols 00,01,10,11
if(M==4)
     symbol=[0 0; 0 1; 1 0; 1 1];
end
% 8-PAM symbols 000,001,010,011,100,101,110,111
if(M==8)
     symbol=[0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1];
end
```

Έπειτα ο κώδικας **mdl.m** εκτελεί τη διαμόρφωση ενός ψηφιακού σήματος εισόδου για τη δημιουργία ενός διαμορφωμένου σήματος, το οποίο θα σταλεί μέσω ενός καναλιού. Συγκεκριμένα κάνει διαμόρφωση διαμόρφωσης πλάτους παλμού (PAM).

mdl.m

```
%initializiing variable k as the index of output signal

k=1;
%PAM modulation
for i=1:number
    %finds the int m that represents the symbol according to the symbols
matrix
    for j=1:M
        if (sum(input(i,:)==symbol(j,:))==log2(M))
            m=j;
        end
    end
    %calculation of the transmitted series cos(t)
    for t=0:T_symbol-1
      %waveforms M-PAM
      signal(k)=(2*m-1-M)*A*g*cos(2*pi*fc*t);
      k=k+1;
    end
end
```

Τέλος, ο κώδικας **demdl.m** εκτελεί την αποδιαμόρφωση του λαμβανόμενου σήματος για την ανάκτηση του αρχικού ψηφιακού σήματος εισόδου. Αυτό το κάνει εκτελώντας αποδιαμόρφωση PAM.

demdl.m

```
k=1;
for i=1:number
    %Calculation of symbol with demodulation function
    for t=0:T_symbol-1
    rr(t+1)=r(k)*g*cos(2*pi*fc*t);
    k=k+1;
    end
    r_symbol=sum(rr);
    for m=1:M
    sv=(2*m-1-M)*A;
    %symbol distance
    d(m)=sqrt(sum((r_symbol-sv).^2));
    end
    %demaper by pronunciation for symbol output
    [mnm l]=min(d);
    output(i,:)=symbol(l,:);
end
```

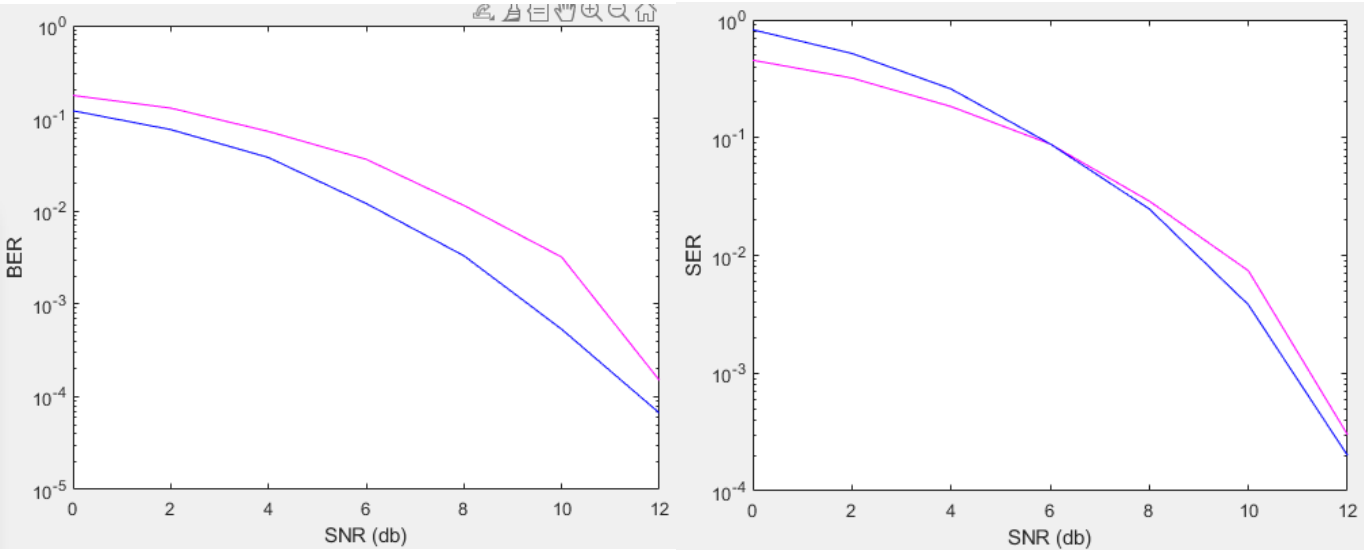
Για τον σχεδιασμό των γραφικών του ερωτήματος 2 (BER) και 3 (SER) υλοποιήθηκε ο κώδικας.

sol_PAM.m

```
clear;
init();
% #symbols as bits 1st 4-PAM and 2nd 8-PAM
% so 1st smbl=2 and next smbl=3bits
smbl=2;
while (smbl<4)
% depending on the value of the symbol the code for the M-PAM
% will be configured i. e. either 4-PAM configuration or 8-PAM
% respectively
M=2^smbl;
eb=1/log2(M);
% initialization SNR=1 and as it changes the corresponding
% BER and SER values of the signal will be recorded
SNR_i=1;
% produced bits
Lb=10000*log2(M);
xbits=randsrc(Lb,1,[0 1]);
input=reshape(xbits,Lb/log2(M),log2(M));
output=input*0;
ybits=xbits*0;
number=Lb/log2(M);
% output signal
signal=zeros(Lb/log2(M)*40,1);
% produced Gray symbols for each PAM configuration
g_smblbols();
% For each SNR we perform modulation and demodulation
% and measure the noise values and calculate the BER, SER
for snr=0:2:20
    mdl();
% calculation of the noise level and addition of noise to s
    No=eb/(10^(snr/10));
    noise = sqrt(No/2)*randn(number*40,1);
    r=noise+signal;
    demdl();
% calculation BER, SNR for each SNR
ybits=output(:);
    BER(1,SNR_i)=sum(abs(ybits-xbits))/Lb;
    SER(1,SNR_i)=sum(abs(bi2de(output)-bi2de(input)))/number;
% increase SNR value for repetition having different SNR value
    SNR_i=SNR_i+1;
end
% BER / Symbol Error Rate (SER) Graph
   figure(1)
   SNR=0:2:20;
   %4-PAM magenta
   if (M==4)
       semilogy(SNR,BER,'m');
       hold on
   else
       %8-PAM blue
       semilogy(SNR,BER,'b');
   end
       xlabel('SNR (db)');
       ylabel('BER');
```

```
    figure(2)
    SNR=0:2:20;
    %4-PAM magenta
    if (M==4)
       semilogy(SNR,SER,'m');
       hold on
    else
        %8-PAM blue
       semilogy(SNR,SER,'b');
    end
           xlabel('SNR (db)');
           ylabel('SER');
     smbl = smbl + 1;
end
```

Ακολουθούν οι γραφικές παραστάσεις. Με magenta η γραφική παράσταση 4-PAM ενώ με μπλε για THN 8-PAM.



Σύμφωνα με το διάγραμμα SER-SNR συμπεραίνουμε πως σε ένα σύστημα 4-PAM υπάρχει μεγαλύτερη καθυστέρηση στη μετάδοση όμως είναι καλύτερο σε θέμα απόδοσης. Το ίδιο συμπεραίνουμε και από τις κυματομορφές του διαγράμματος BER-SNR.