Ψηφιακές Τηλεπικοινωνίες

1η Εργαστηριακή Άσκηση
Κωδικοποίηση PCM και Κωδικοποίηση Huffman

Παπαρροδοπούλου Αναστασία ΑΜ 3873
10/01/14

Εισαγωγή:

Το περιεχόμενο της εργαστηριακής άσκησης είναι η μέλετη βασικών μεθόδων για το σχεδιασμό συστημάτων κωδικοποίησης κυματομορφής. Τα συστήματα αυτά σχεδιάζονται ώστε να επιτρέπουν την αναπαραγωγή στον προορισμό της κυματομορφής εξόδου της πηγής με όσο τον δυνατό μικρότερη παραμόρφωση. Η παλμοκωδική διαμόρφωση (PCM), που είναι και το αντικείμενο της άσκησης, είναι ένα σύστημα ψηφιακής διαμόρφωσης αναλογικών δεδομένων.

Ειδικότερα ασχοληθήκαμε με την κατασκευή του κβαντιστή του PCM συστήματος. Φτιάξαμε συναρτήσεις τόσο για ομοιόμορφο κβαντιστή (ομοιόμορφο PCM) όσο και για μη ομοιόμορφο κβαντιστή (μη ομοιόμορφο PCM). Μελετήσαμε επίσης και την περίπτωση του PCM που χρησιμοποιεί μη ομοιόμορφο κβαντιστή, τον οποίο σχεδιάσαμε υλοποιώντας τον αλγόριθμο Lloyd-Max.

Για κάθε ένα από τα παραπάνω μοντέλα κωδικοποιήσαμε ένα σήμα φωνής, αυτό στο αρχείο speech.wav που δίνεται μαζί με την εκφώνηση της άσκησης.

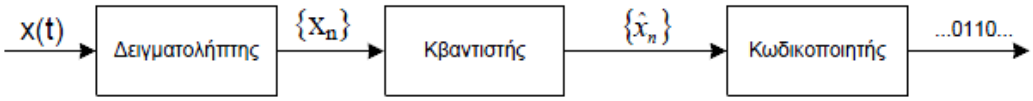
Χρησιμοποιώντας τον υλοποιημένο Huffman αλγόριθμο, κωδικοποιήσαμε την πηγή (για N = 2, 4 και 8 bits) και υπολογίσαμε την αποδοτικότητα του συγκεκριμένου αλγόριθμου κωδικοποίησης για κάθε ένα από τα σχήματα PCM (ομοιόμορφο και μη-ομοιόμορφο, Lloyd-Max).

Τέλος υλοποιήσαμε μία συνάρτηση που να επιστρέφει το αλφάβητο του κειμένου στην περίπτωση της πηγής B, του αρχείου keimeno.txt.

Ερώτημα 1:

(a)

Ένα σύστημα PCM αποτελείται από τρία βασικά τμήματα : ένα δειγματολήπτη, ένα κβαντιστή και ένα κωδικοποιητή, όπως φαίνεται στο ακόλουθο σχήμα:



Διάγραμμα βαθμίδων ενός συστήματος PCM

Αρχικά λοιπόν υλοποιήσαμε το σχήμα του ομοιόμορφου PCM. Στις εφαρμογές ομοιόμορφου PCM υποθέτουμε ότι οι τιμές των δειγμάτων της εισόδου βρίσκονται στο διάστημα [min_value,max_value] και ότι το πλήθος των σταθμών κβάντισης N είναι δύναμη του 2, με αποτέλεσμα το εύρος κάθε περιοχής κβάντισης να είναι σταθερό και να ισούται με $\text{max\_value}/2^{v-1}$.

Στο ομοιόμορφο PCM ως τιμές κβάντισης επιλέγονται τα μέσα των περιοχών κβάντισης και επομένως το σφάλμα ισούται με x - Q(x) ,όπου Q(x) η συνάρτηση κβάντισης, είναι μία τυχαία μεταβλητή που λαμβάνει τιμές στο διάστημα (-Δ/2,Δ/2].

Για την υλοποίηση του ομοιόμορφου PCM κατασκευάσαμε τον ομοιόμορφο βαθμωτό κβαντιστή N bits, δηλαδή 2^N επιπέδων. Ο ορισμός της συνάρτησης σε matlab που υλοποιεί τον κβαντιστή είναι ο παρακάτω(έχουμε συμπεριλάβει από τώρα, μιας και θα χρειαστούν στη συνέχεια, το διάνυσμα πιθανοτήτων p εμφάνισης κάθε συμβόλου και το διάνυσμα D της παραμόρφωσης):

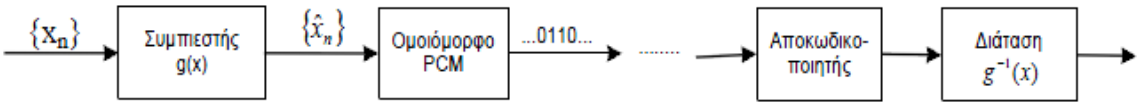
```
function [xq,centers,p,D] = my_quantizer(x,N,max_value,min_value)

% Uniform Quantizer N bits, 2^N levels
%     input:
%           x           : dianysma deigmatwn shmatos eisodoy
%           N           : o arithmos twn bits
%           max_value   : megisti timi dinamikis perioxis[min_value,max_value]
%           min_value   : elaxisti timi dinamikis perioxis[min_value,max_value]
%
%     output:
%           xq          : dianysma me to kvadismeno shma eksodou
%           centers     : dianysma me ta kedra twn perioxwn
%                             kvadisis
%           p           : dianysma me tis pithanotites emfanisis
%                             kathe kedrou toy kvantisti
%           D           : h Paramorfwsi
```

Η λειτουργία του ομοιόμορφου κβαντιστή συνοπτικά είναι, αρχικά να θέτει όλες τις τιμές του δείγματος εισόδου εντός δυναμικής περιοχής [min_value,max_value] και να υπολογίζει το εύρος τις κάθε ζώνης. Στην συνέχεια υπολογίζει τα όρια των ζωνών κβαντισμού καθώς και κέντρα τις κάθε ζώνης. Κάθε τιμή του δείγματος εισόδου κβαντίζεται με το κέντρο τις περιοχής στην οποία αυτή βρίσκεται. Ο αναλυτικός κώδικας μαζί με τα test αρχεία επισυνάπτονται στο τέλος της αναφοράς.

(b)

Στην μη ομοιόμορφη κβάντιση το εύρος κάθε περιοχής δεν είναι σταθερό σε αντίθεση με την ομοιόμορφη κβάντιση. Το σχήμα του ομοιόμορφου PCM λειτουργεί ικανοποιητικά όταν τα δείγματα του σήματος εισόδου ακολουθούν μία ομοιόμορφη κατανομή. Αντίθετα σε σήματα όπου η κατανομή των δειγμάτων δεν ακολουθεί την ομοιόμορφη κατανομή (π.χ σήμα φωνής) το ομοιόμορφο PCM δεν έχει ικανοποιητική συμπεριφορά. Σε ένα σήμα ομιλίας για παράδειγμα όπου τα μικρότερα πλάτη εμφανίζονται με μεγαλύτερη πιθανότητα απ' ότι τα μεγάλα χρειάζεται ενας κβαντιστής με περισσότερες περιοχές κβάντισης στα μικρότερα πλάτη και λιγότερες στα μεγάλα πλάτη. Συνεπώς ο κβαντιστής θα έχει περιοχές κβάντισης ποικίλου εύρους.

Στο μη ομοιόμορφο PCM χρησιμοποιείται η τεχνική του companding (compressing-expanding): τα δείγματα εισόδου διέρχονται αρχικά από ένα φίλτρο προκειμένου να συμπιεστούν τα μεγάλα πλάτη – μείωση της δυναμικής περιοχής του σήματος - και στην συνέχει κβαντίζονται ομοιόμορφα. Στην μεριά του δέκτη γίνεται η αποσυμπίεση του κβαντισμένου σήματος για ανάκτηση των δειγμάτων. Οι βαθμίδες ενός μη-ομοιόμορφου PCM φαίνονται στο παρακάτω σχήμα:



Διάγραμμα βαθμίδων ενός συστήματος μη ομοιόμορφου PCM

Στην περίπτωση που μας ζητήθηκε η κατασκευή ενός μη-ομοιόμορφου κβαντιστή, αυτός έγινε με την χρήση του αλγορίθμου Lloyd-Max ο οποίος επιτρέπει την σχεδίαση βέλτιστου κβαντιστή για οποιδήποτε αριθμό επιπέδων. Πρόκειται για έναν επαναληπτικό αλγόριθμο όπου σε κάθε επανάληψή του υπολογίζει την διαμόρφωση της τρέχουσας επανάληψης και την συγκρίνει με αυτή της προηγούμενης. 'Οταν η διαφορά συγκλίνει, τότε ο αλγόριθμος τερματίζει.

Ο ορισμός της συνάρτησης σε matlab που υλοποιεί το μη ομοιόμορφο κβαντιστή(αλγόριθμος Lloyd-Max) είναι ο παρακάτω:

```
function [xq,centers,D,iters] = Lloyd_Max(x,N,max_value,min_value)

% Mi Omiomorfos Kvadistis - Lloyd Max
%     input:
%           x           : dianysma deigmatwn shmatos eisodoy
%           N           : o arithmos twn bits
%           max_value   : megisti timi dinamikis perioxis[min_value,max_value]
%           min_value   : elaxisti timi dinamikis perioxis[min_value,max_value]
%
%     output:
%           xq          : dianysma me to kvadismeno shma eksodou
%           centers     : dianysma me ta kedra twn perioxwn
%                             kvadisis
%           D           : dianysma pou periexei tis times [D1:Dk_max] me Di h
%                             mesh paramorfwsi se kathe epanalipsi i tou
%                             algorithmou
```

Οι συναρτήσεις λοιπόν που κατασκευάσαμε για τα μοντέλα του ερωτήματος 1, και τα αντίστοιχα scripts για τον έλεγχο της λειτουργίας τους, παραθέτονται στο παράρτημα με τους κώδικες στη συνέχεια.

Η δειγματοληψία του σήματος ομιλίας που βρίσκεται στο αρχείο speech.wav και δειγματοληψία έγινε με ρυθμό f_s=8KHz. Για να φορτώσουμε το σήμα στην MATLAB χρησιμοποισαμε την εντολή:

```
>> x = wavread('speech.wav');
```

Ερώτημα 2:

Στην συνέχεια κωδικοποιήσαμε τα δείγματα x της πηγής για κάθε ένα απο τα παραπάνω σχήματα κωδικοποίησης PCM για N=2, 4 και 8 bits. Τα αποτελέσματα και οι εξόδοι κάθενος από τα παραπάνω σχήματα κωδικοποίησης αξιολογήθηκαν με βάση:

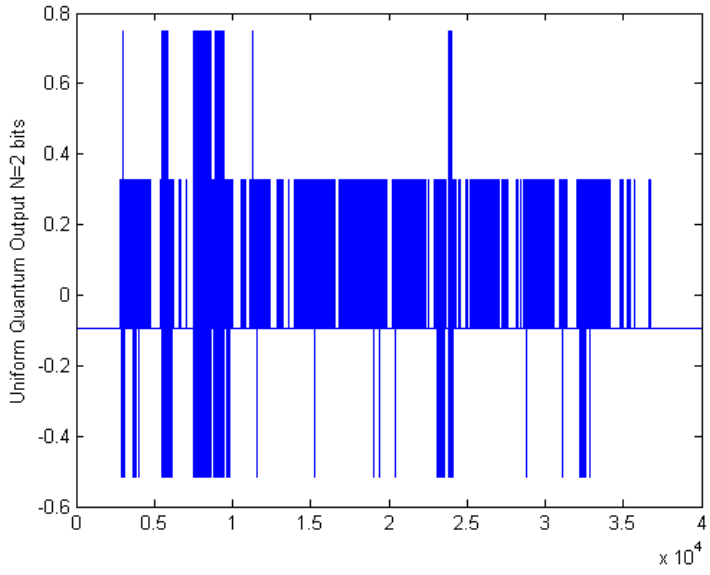
- τις τιμές του SQNR – λόγος σήματος προς θόρυβο κβάντισης ο οποίος ορίζεται ως $\text{SQNR} = \frac{E[X^2]}{E[(X-Q(X))^2]}$
- τις κυματομορφές εξόδου του κάθε σχήματος και
- του ακουστικού αποτελέσματος της κάθε μεθόδου με την χρήση της sound()

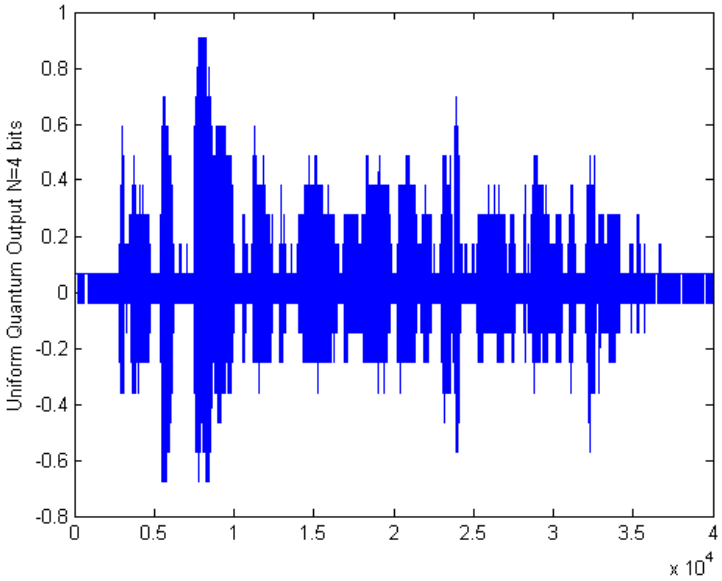
Τα αποτελέσματα στις τιμές SQNR για κάθε μια μέθοδο παρουσιάζονται στον παρακάτω πίνακα:

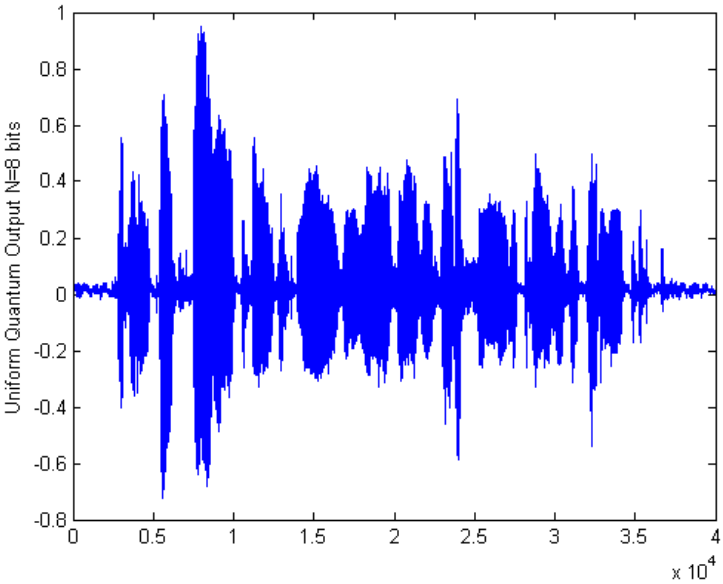
	N=2	N=4	N=8
SQNR	1.36	15.3	697

Για τις παραπάνω υλοποιήσεις οι αντίστοιχες κυματομορφές που παράγονται από τη Matlab, είναι οι ακόλουθες:

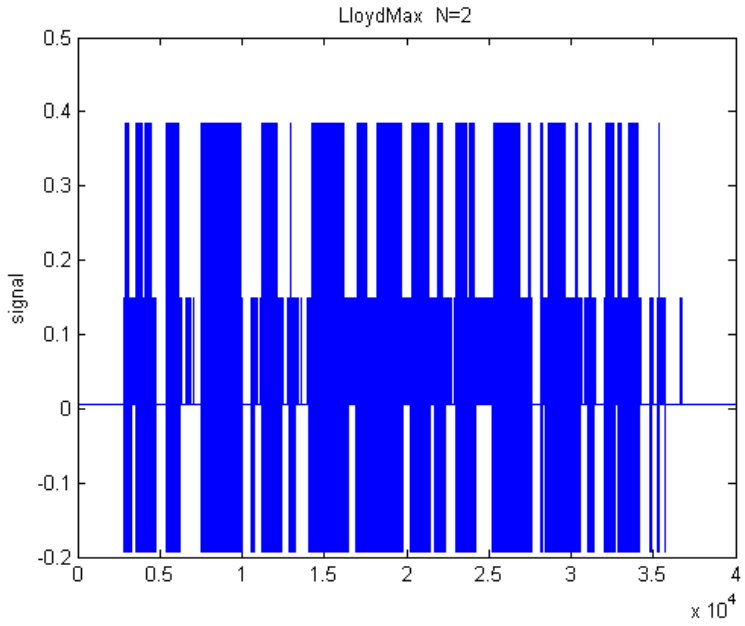
Αρχικά λοιπόν για τον my_quantizer, έχουμε τις εξής:

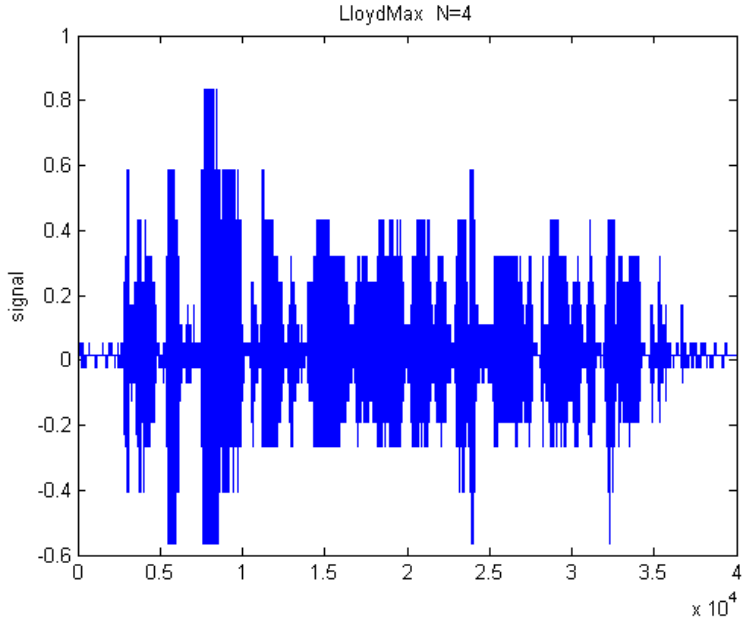


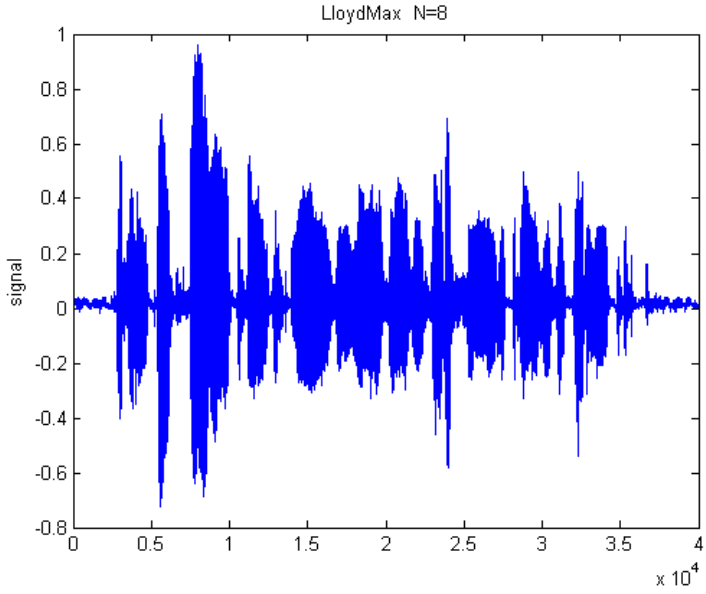

Uniform Quantum Output N=4 bits
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3
3.5
4
x 10^4

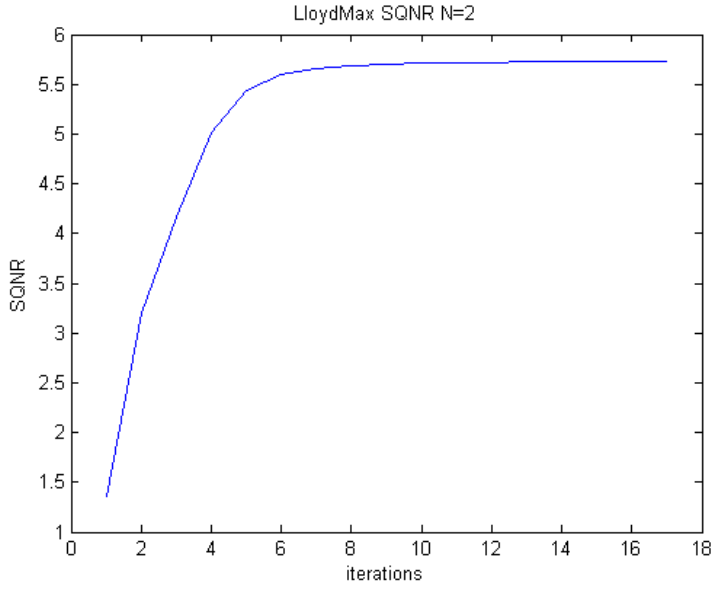

Uniform Quantum Output N=8 bits
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3
3.5
4
x 10^4

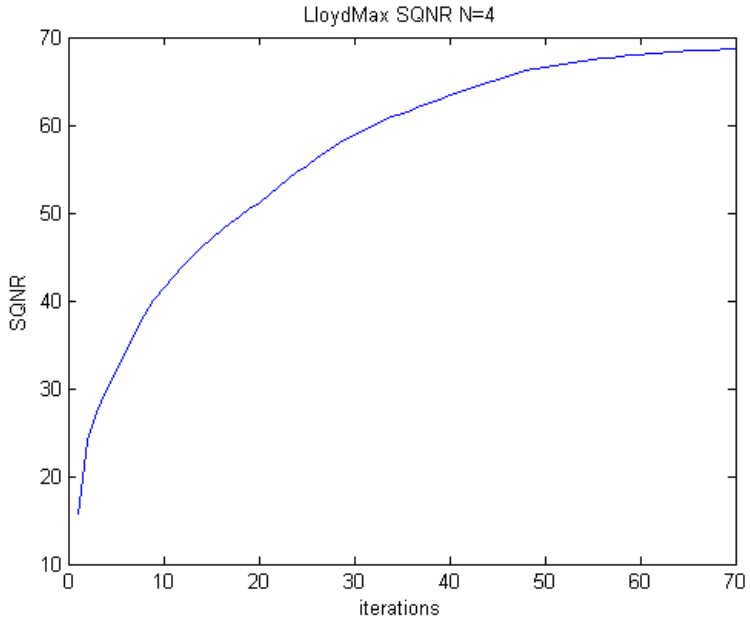
Στη συνέχεια για το Lloyd_Max, έχουμε:

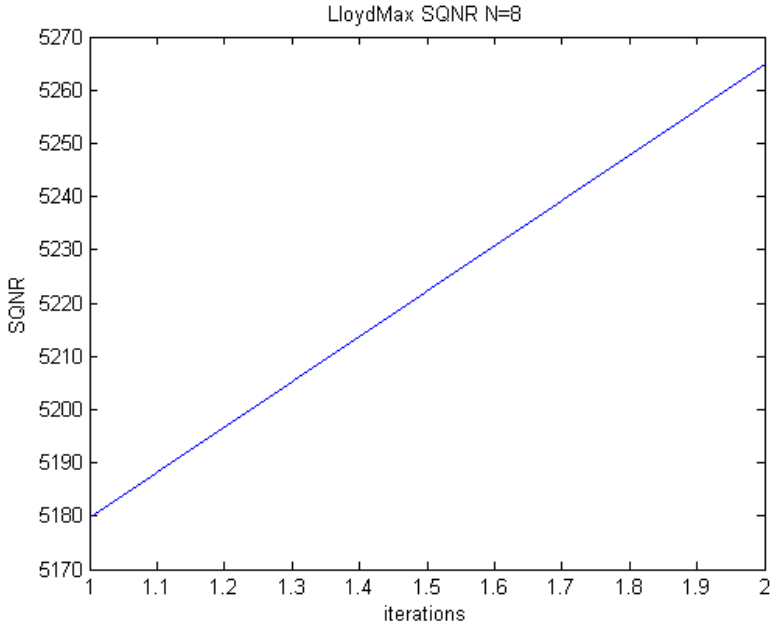





LloydMax N=8
signal
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3
3.5
4
x 10^4


LloydMax SQNR N=2
SQNR
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0
2
4
6
8
10
12
14
16
18
iterations


LloydMax SQNR N=4
SQNR
iterations
70
60
50
40
30
20
10
0
10
20
30
40
50
60
70


LloydMax SQNR N=8
SQNR
iterations
5270
5260
5250
5240
5230
5220
5210
5200
5190
5180
5170
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2

Ερώτημα 5:

Όπως έχουμε σχολιάσει εξ'αρχής, οι κώδικες έχουν κατασκευαστεί έτσι, ούτως ώστε να επιστρέφουν τις πιθανότητες εμφάνισης κάθε στάθμής, και την παραμόρφωση του σήματος.

Στο ερώτημα αυτό σκοπός μας είναι να μελετήσουμε την αποδοτικότητα ενός αλγορίθμου κωδικοποίησης πηγής, του κώδικα Huffman. Χρησιμοποιήσαμε τον υλοποιημένο κώδικα Huffman που μας δίνεται για να κωδικοποιήσουμε τις εξόδους καθενός από τα παραπάνω σχήματα PCM για N=4 και 8 bits.

Αφού κωδικοποιήσουμε την έξοδο υπολογίσαμε την αποδοτικότητα του κώδικα σε κάθε ένα από τα παραπάνω σχήματα. Η αποδοτικότητα υπολογίζεται ως εξής :

$$\text{Αποδοτικότητα} = \text{εντροπία}/\text{μέσο μήκος κώδικα} \leq 1$$

Κατασκευάσαμε δύο επιμέρους συναρτήσεις για τον υπολογισμό της εντροπίας και του μέσου μήκους κώδικα. Η εντροπία υλοποιείται από την συνάρτηση:

```
function H = edropia(p)
% Ypologismos tis Edropias tis pigis
% input p: dianysma pithanotitwn emfanisis symvolwn
```

Επίσης η συνάρτηση που βρίσκει το μέσο μήκος κώδικα είναι η παρακάτω:

```
function L_meso = mikos_kwdika(p,len)
% Ypologizei to meso mikos kwdika
% input
% p : dianysma pithanotitwn emfanisis
% len : dianusma me ta length tou kathe
% kwdikopoihmenoy symvolou
```

Το script το οποίο εξωμοιώνει το ερώτημα αυτό είναι το test_q4.m.

Ερώτημα 6:

Η συνάρτηση που υλοποιήσαμε για να επιστρέφει το αλφάβητο του κειμένου στην περίπτωση της πηγής Β, με τη χρήση της συνάρτησης Huffman, είναι η ακόλουθη:

```
fid=fopen('keimeno.txt');
G=fscanf(fid,'%c');

T2=zeros(1,127);
for j=1:127
      T2(j)=j;
end

T3=zeros(1,127);

T1=abs(G);
for j=1: length(G)
      for k=1:127
            if (T1(j)==T2(k))
                  T3(k)=T3(k)+1;
            end
      end
end

T3;
T3=T3/length(G)
```

Στη συνέχεια, για τον υπολογισμό της εντροπίας της πηγής, δημιουργήσαμε το ακόλουθο κομμάτι κώδικα:

```
T4=zeros(1,127);

for w=1:127
      if(B3(w)==0)
            T4(w)=0;
      else T4(w)=T3(w)*log2(1./T3(w));
      end
end

entropia = sum(T4)
```

Η εντροπία λοιπόν υπολογίζεται σε 4.274.

Τέλος, με τη χρήση της συνάρτησης Huffman, ο κώδικας που χρησιμοποιήθηκε είναι ο εξής:

```
[code, len] = huffman(T3);
P = sum(len.*T3);
profitability = entropia./P
```

Η αποδοτικότητα Huffman υπολογίζεται σε 0.9935.

Παράρτημα με Κώδικες:

Στο σημείο αυτό παρουσιάζουμε τους κώδικες που υλοποιήσαμε για κάθε ένα διαφορετικό σχήμα PCM αλλά και διάφορα script αρχεία για έλεγχο.

Ομοιόμορφο PCM

my_quantizer.m

```
function [xq,centers,p,D] = my_quantizer(x,N,max_value,min_value)
% Uniform Quantizer N bits, 2^N levels
%     input:
%           x           : dianysma deigmatwn shmatos eisodoy
%           N           : o arithmos twn bits
%           max_value   : megisti timi dinamikis perioxis[min_value,max_value]
%           min_value   : elaxisti timi dinamikis perioxis[min_value,max_value]
%
%     output:
%           xq          : dianysma me to kvadismeno shma eksodou
%           centers     : dianysma me ta kedra twn perioxwn
%                             kvadisis
%           p           : dianysma me tis pithanotites emfanisis
%                             kathe kedrou toy kvantisti
%           D           : h Paramorfwsi

lenx = length(x);
stathmes = 2^N;
b = (max_value+abs(min_value))/stathmes; %euros statmis
a = zeros(stathmes+1,1);                %desmeusi gia to dianisma twn ai
centers = zeros(stathmes,1);            %desmeusi gia to dianisma twn kedrwn
p = zeros(stathmes,1);                  %desmeusi gia to dianisma pithanotitwn
c = zeros(stathmes,1);                  %dianisma gia to plithos emfanisis tou
kathe kedrou

%% ----- Ypologismos ai ----- %%
a(1) = min_value;
for i=2:stathmes+1
    a(i) = a(1) + (i-1)*b;
end

%% ----- Ypologismos kedrwn ----- %%
for i=1:stathmes
    centers(i) = ( a(i) + a(i+1) ) / 2;
end


%% Kvadisi dld Q(x) = center(i) gia kathe x e Ri %%
for i=1:lenx
```

```
    quantum = 0;
    j=1;
    while (j<=stathmes) && (quantum==0)

        if  ( ( x(i)>=a(j) ) && ( x(i)<a(j+1)) )
                xq(i) = centers(j);

                c(j) = c(j) + 1;
                quantum = 1;
        else
                j = j+1;
        end

    end
end
xq = xq';

%% Ypologismos Pithanotitas %%
for i=1:stathmes
    p(i) = c(i) / length(xq);
end

%% Paramorfwsi %%
D = mean((x-xq).^2);
```

test_my_quant.m

Στο αρχείο test.m τρέχουμε τον ομοιόμορφο κβαντιστή για κάθε N και εξάγουμε τα αποτελέσματα για τον λόγο SQNR.

```
x = wavread('speech.wav');
max_value = max(x);
min_value = min(x);

N = 2;
[xq,centers,p,D] = my_quantizer(x,N,max_value,min_value);

SQNR = mean(x.^2)/D;
sprintf('Distortion: %0.3g \nSQNR: %0.3g',D,SQNR)

figure;
plot(xq);
ylabel('Uniform Quantum Output N=2 bits');

N = 4;
[xq,centers,p,D] = my_quantizer(x,N,max_value,min_value);

SQNR = mean(x.^2)/D;
sprintf('Distortion: %0.3g \nSQNR: %0.3g',D,SQNR)
```

```
figure;
plot(xq);
ylabel('Uniform Quantum Output N=4 bits');

N = 8;
[xq,centers,p,D] = my_quantizer(x,N,max_value,min_value);

SQNR = mean(x.^2)/D;
sprintf('Distortion: %0.3g \nSQNR: %0.3g',D,SQNR)

figure;
plot(xq);
ylabel('Uniform Quantum Output N=8 bits');
```

Μη Ομοιόμορφο PCM

Lloyd-Max.m

```
function [xq,centers,D,iter] = Lloyd_Max(x,N,max_value,min_value)

% Mi Omiomorfos Kvadistis - Lloyd Max
%     input:
%           x           : dianysma deigmatwn shmatos eisodoy
%           N           : o arithmos twn bits
%           max_value   : megisti timi dinamikis perioxis[min_value,max_value]
%           min_value   : elaxisti timi dinamikis perioxis[min_value,max_value]
%
%     output:
%           xq          : dianysma me to kvadismeno shma eksodou
%           centers     : dianysma me ta kedra twn perioxwn
%                         kvadisis
%           D           : dianysma pou periexei tis times [D1:Dk_max] me Di h
%                         mesh paramorfwsi se kathe epanalipsi i tou
%                         algorithmou

levels=2.^N;                %arithmos epipedwn kvadisti
centers=zeros(levels,1);    %dianisma me ta kedra kathe stathmis kvadisis
T = zeros(levels+1,1);      %dianisma me ta akra kathe perioxis kvadisis
vima = (max_value+abs(min_value))/(levels); %euros kathe perioxis

%arxikopoihsh twn kedrwn me ta kedra tou omoiomorfou kvadisti
centers = min_value + vima/2 : vima :max_value-(vima/2);

Dprev=0;                    %Paramorfwsi prohgoumenis epanalipsis
diff=1;                     %diafora diadoxikwn paramorfwsewn -arxika 1
iter=1;                     %counter gia ta iteration tou Lloyd-Max
threshold = 1e-7;           %katofli syglisis
```

```
%loop oso to diff den exei ftasei to katwfli
while(diff>threshold)


    % arxikopoihsh twn akrwn twn perioxwn kvadisis
    T(1) = -inf;
    for j=2:levels
        T(j)= 0.5*( centers(j-1)+centers(j) );
    end
    T(levels+1) = inf;


    %kvadisi simatos
    for i=1:length(x)
        for k=1:levels
           if ((x(i)<=T(k+1)) && (x(i)>T(k)))
                   xq(i)=centers(k);
           end
        end
    end

    %ypologismos mesis paramorfwsis tis trexousas epanalipsis
    d = mean((x-xq').^2);
    D(iter) = d;

    %ypologismos tis diaforas Paramorfwsis
    diff = abs(D(iter)-Dprev);
    Dprev = D(iter);


    %Nea kedra kvadismou: einai ta kedroeidi twn zwnwn
    for k=1:levels
        centers(k) = mean ( x ( x>T(k) & x<=T(k+1) ));
    end

    %increase ton counter twn iterations
    iter = iter + 1;

end

end
```

test_lloyd_max.m

```
x = wavread('speech.wav');
```

```
max_value = max(x);
min_value = min(x);

N=2;
[xq,centers,D,iter] = Lloyd_Max(x,N,max_value,min_value);
SQNR = mean(x.^2)./D;
figure;
plot(SQNR);
display(iter);
title( [ 'LloydMax SQNR N=', num2str(N)]);
ylabel('SQNR');
xlabel('iterations');
figure;
plot(xq);
title( [ 'LloydMax  N=', num2str(N)]);
ylabel('signal');

N=4;
[xq,centers,D,iter] = Lloyd_Max(x,N,max_value,min_value);
SQNR = mean(x.^2)./D;
figure;
plot(SQNR);
display(iter);
title( [ 'LloydMax SQNR N=', num2str(N)]);
ylabel('SQNR');
xlabel('iterations');
figure;
plot(xq);
title( [ 'LloydMax  N=', num2str(N)]);
ylabel('signal');

N=8;
[xq,centers,D,iter] = Lloyd_Max(x,N,max_value,min_value);
SQNR = mean(x.^2)./D;
figure;
plot(SQNR);
display(iter);
title( [ 'LloydMax SQNR N=', num2str(N)]);
ylabel('SQNR');
xlabel('iterations');
figure;
plot(xq);
title( [ 'LloydMax  N=', num2str(N)]);
ylabel('signal');
```

edropia.m

```
function H = edropia(p)
% Ypologismos tis Edropias tis pigis
% input p: dianysma pithanotitwn emfanisis symvolwn
h=0;
len = length(p);
for i=1:len
if (p(i)~=0)
h = h + (p(i) * log2(p(i)));
```

```
end
end
H = (-1)*h;
End
```

mikos_kwdika.m

```
function L_meso = mikos_kwdika(p,len)
% Ypologizei to meso mikos kwdika
% input
% p: dianysma pithanotitwn emfanisis
% len:dianusma me ta length tou kathe
% kwdikopoihmenoy symvolou
disp(p);
L_meso=0;
for i=1:length(p)
L_meso = L_meso +p(i)*len(i);
end
end
```

test_q4.m

```
x = wavread('speech.wav');
max_value = 1;
display('Huffman encoding');
display('====================');
display('Uniform N=4');
N=4;
[xq,p] = my_quantizer(x,N,max_value,min_value);
[code,len] = huffman(p);
%Apodotikotita = H(X)/L_meso
%H(X): endropia pigis , L_meso: meso mikos kwdika
L_meso = mikos_kwdika(p,len);
H = edropia(p);
perf_u4 = H/L_meso;
display(perf_u4);
display('Uniform N=6');
N=6;
[xq,p] = my_quantizer(x,N,max_value,min_value);
[code,len] = huffman(p);
%Apodotikotita = H(X)/L_meso
%H(X): endropia pigis , L_meso: meso mikos kwdika
L_meso = mikos_kwdika(p,len);
H = edropia(p);
perf_u6 = H/L_meso;
display(perf_u6);
display('PCM m-type N=4');
N=4;
[x_new,D,SQNR,p] = test_nonuni_m(N);
[code,len] = huffman(p);
L_meso = mikos_kwdika(p,len);
H = edropia(p);
perf_m4 = H/L_meso;
display(perf_m4);
display('PCM m-type N=6');
N=6;
[x_new,D,SQNR,p] = test_nonuni_m(N);
[code,len] = huffman(p);
L_meso = mikos_kwdika(p,len);
```

```
H = edropia(p);
perf_m6 = H/L_meso;
display(perf_m6);
display('PCM A-type N=4');
N=4;
[xq,p] = compress_a(N);
[code,len] = huffman(p);
L_meso = mikos_kwdika(p,len);
H = edropia(p);
perf_a4 = H/L_meso;
display(perf_a4);
display('PCM A-type N=6');
N=6;
[xq,p] = compress_a(N);
[code,len] = huffman(p);
L_meso = mikos_kwdika(p,len);
H = edropia(p);
perf_a6 = H/L_meso;
display(perf_a6);
display('Lloyd-Max N=4');
N=4;
[xq, centers, D] = Lloyd_Max(x,N,max_value,min_value);
p = Lloyd_probs(xq);
[code,len] = huffman(p);
L_meso = mikos_kwdika(p,len);
H = edropia(p);
perf_ld4 = H/L_meso;
display(perf_ld4);
display('Lloyd-Max N=6');
N=6;
[xq, centers, D] = Lloyd_Max(x,N,max_value,min_value);
p = Lloyd_probs(xq);
[code,len] = huffman(p);
L_meso = mikos_kwdika(p,len);
H = edropia(p);
perf_ld6 = H/L_meso;
display(perf_ld6);
```