ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ

1η Εργασία

Παναγιώτης Καββαδίας

ΑΜ:1054350

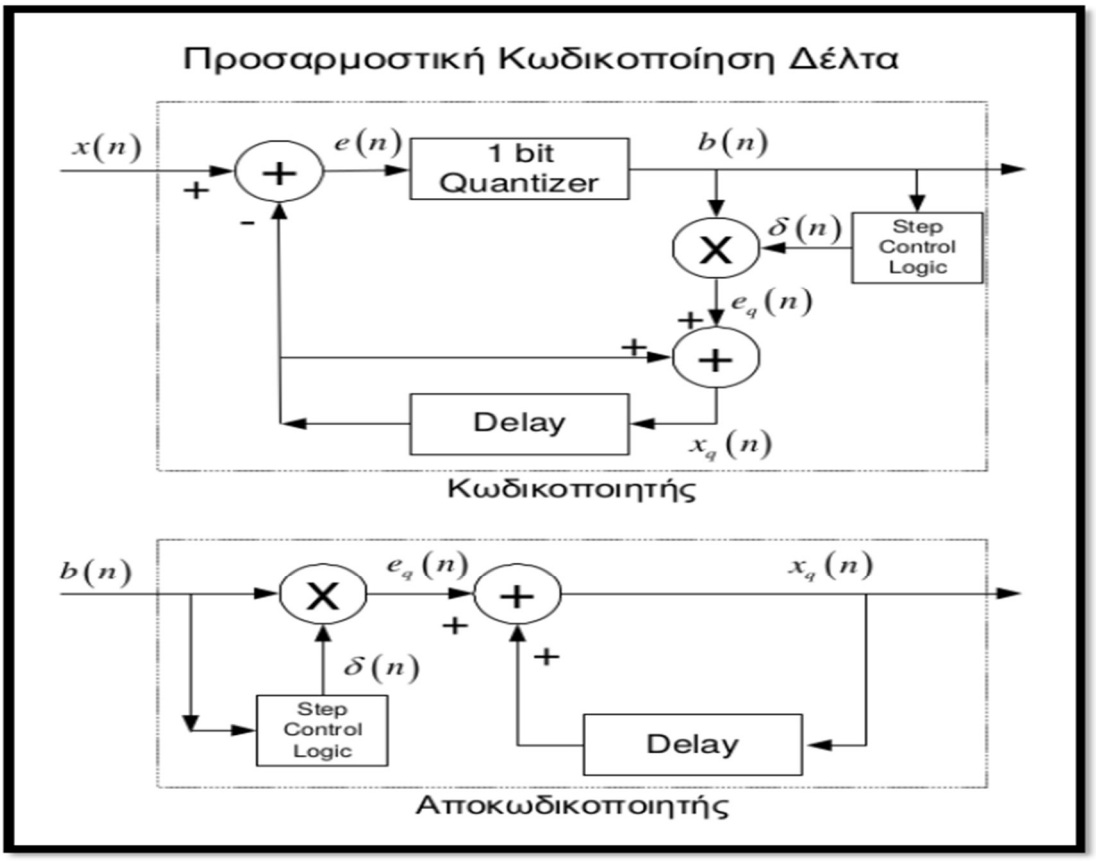
Έτος 2019-2020

Καθηγητής: Κ.Μπερμπερίδης

Μέρος 1

Στο πρώτο μέρος μας ζητήθηκε να υλοποιήσουμε έναν παλμοκωδικό διαμορφωτή(PCM) με μη ομοιόμορφο κβαντιστή. Η PCM είναι μια μέθοδος κωδικοποίησης κυματομορφής, η οποία μετατρέπει ένα αναλογικό σήμα σε ψηφιακά δεδομένα. Έχει τρία βασικά μέρη: Δειγματολήπτη, κβαντιστή και κωδικοποιητή. Για την άσκηση υλοποιήθηκε ένας μη ομοιόμορφος κβαντιστής N bits, δηλαδή 2^N επιπέδων.

Επίσης κληθήκαμε να υλοποιήσουμε προσαρμοστική διαμόρφωση Δέλτα(ADM). Η DM είναι είναι μια απλοποιημένη μορφή της DPCM όπου ο κβαντιστής διαθέτει 2 στάθμες κβάντισης. Η μόνη διαφορά της ADM από την DM είναι ότι η ADM χρησιμοποιεί μεταβαλλόμενο βήμα. Συγκεκριμένα, σε περιοχές που η κυματομορφή του σήματος εμφανίζει απότομη κλήση, η ADM αυξαίνει το βήμα. Αντίθετα, σε περιοχές σταθερής τιμής του σήματος, το βήμα μικραίνει, ώστε να αποφευχθεί ο κοκκώδης θόρυβος.



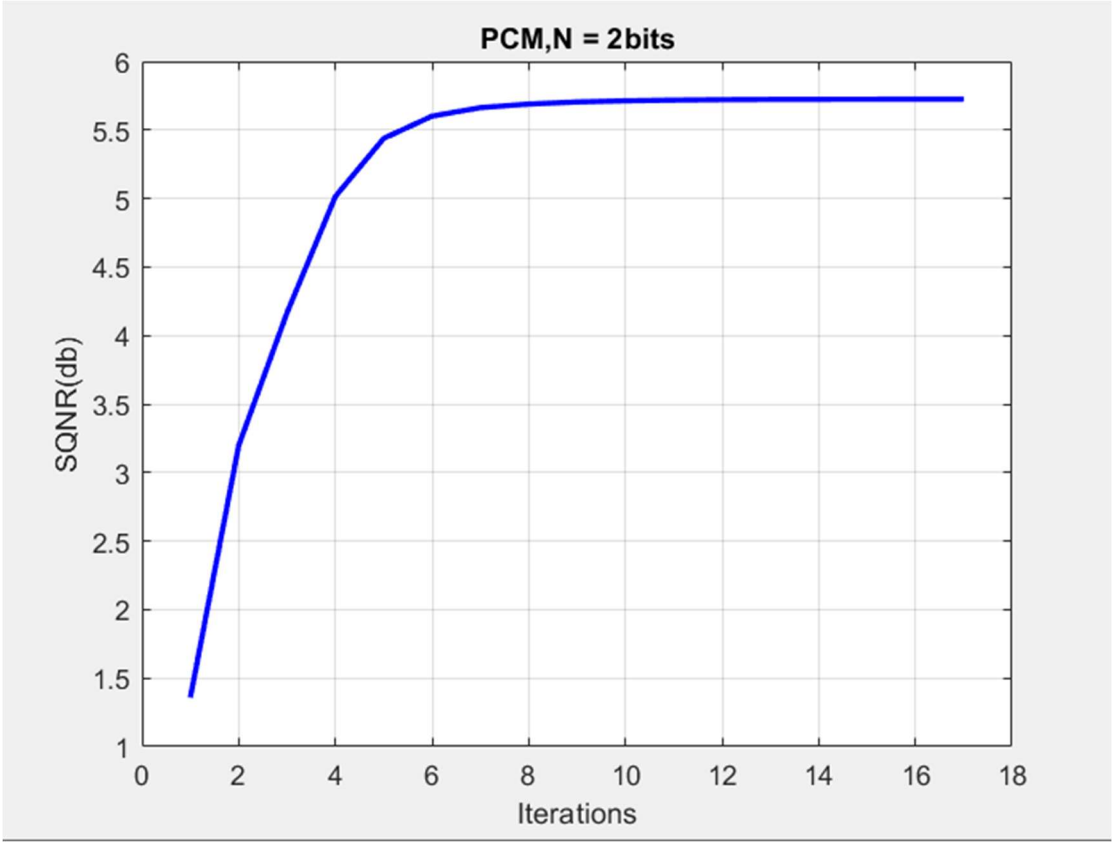
Για τις μετρήσεις μας έπρεπε να χρησιμοποιήσουμε 2 πηγές. Η πρώτη ήταν μια πηγή ήχου και συγκεκριμένα πηγή ομιλίας το οποίο εκτείνεται συχνοτικά μέχρι τα 4KHz περίπου. Στο αρχείο που μας δόθηκε περιέχονταν δείγματα σήματος φωνής με ρυθμό δειγματοληψίας fs=8KHz κβαντισμένα με N=16bits.

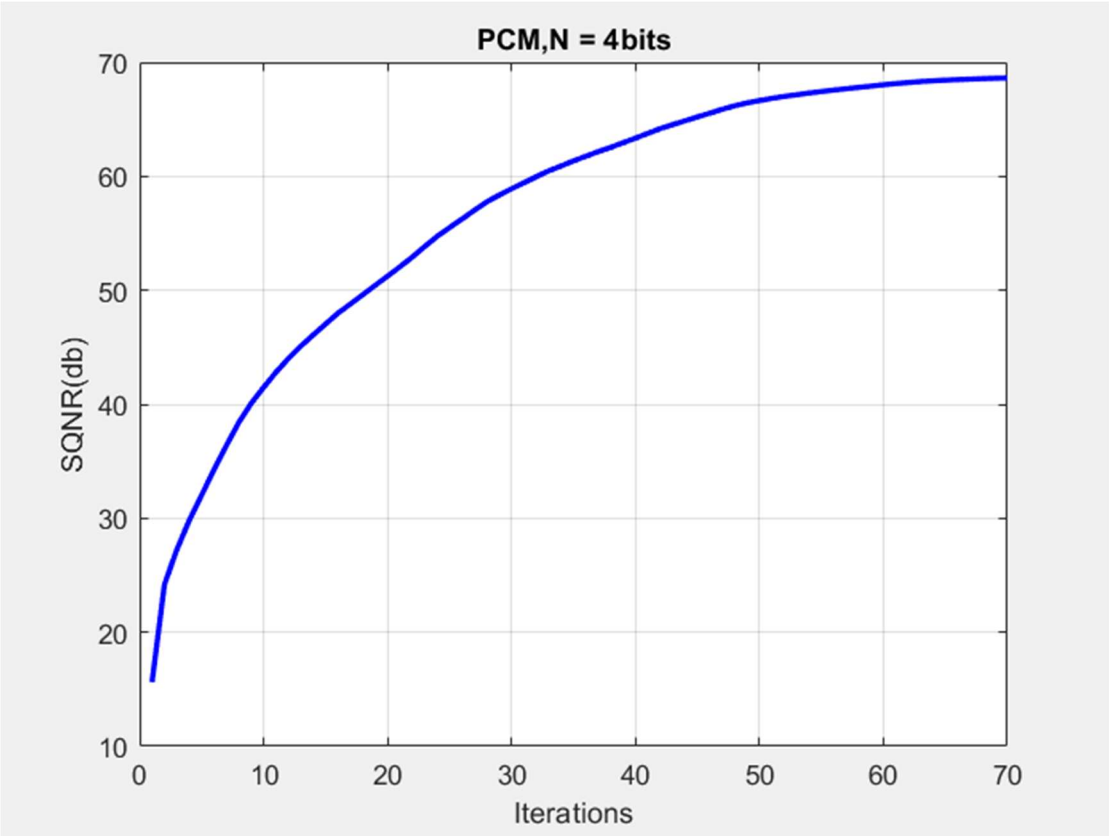
Η πηγή Β ήταν τα εικονοστοιχεία (pixels) μιας grayscale εικόνας. Μια τέτοια εικόνα μπορεί να θεωρηθεί ως ένας πίνακας με εικονοστοιχεία, όπου κάθε εικονοστοιχείο αντιστοιχεί σε ένα byte πληροφορίας. Κάθε εικονοστοιχείο επομένως λαμβάνει μια τιμή στο δυναμικό εύρος [0:255] η οποία αντιστοιχεί σε ένα από τα 256 επίπεδα φωτεινότητας.

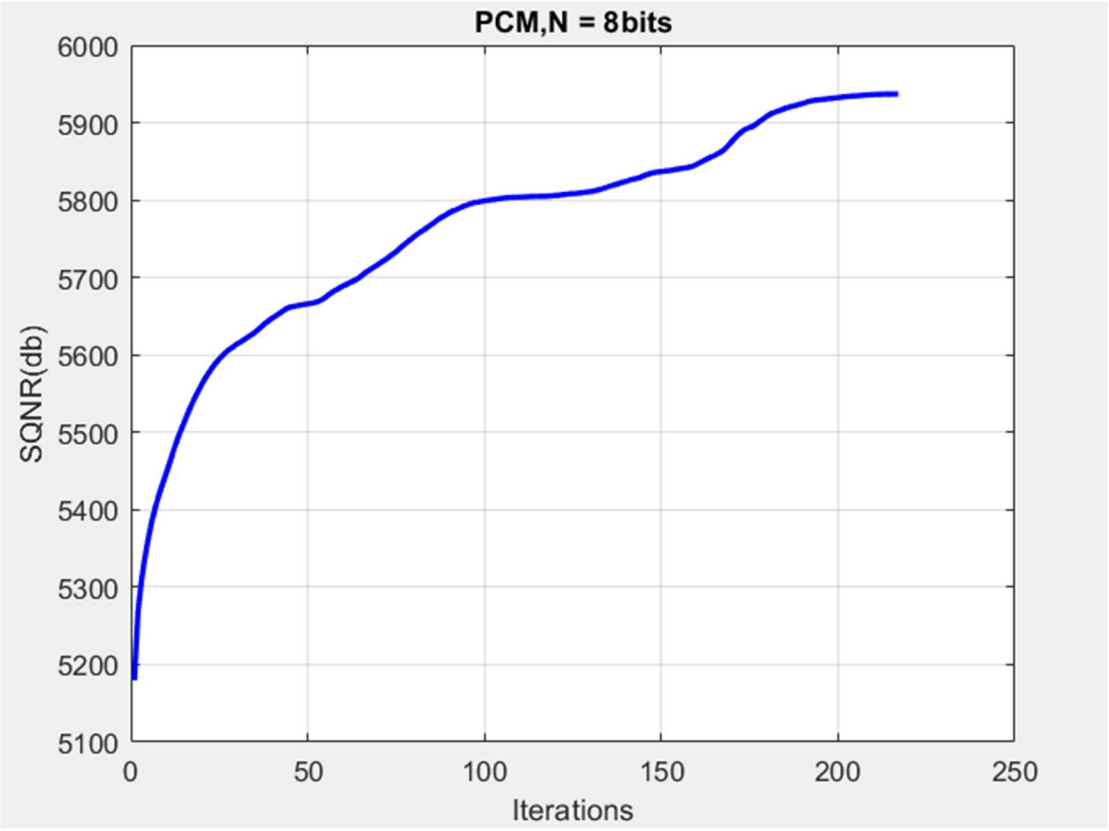
Η PCM υλοποιήθηκε στο αρχείο PCM.m, η ADM στο ADM.m και για την λήψη όλων των απαραίτητων μετρήσεων οι συναρτήσεις καλούνται από το Script.m

Οι πηγαίοι κώδικες δίνονται στο παράρτημα.

Για την πηγή ήχου παρατίθενται οι γραφικές παραστάσεις μεταβολών του SQNR συναρτήσει του αριθμού επαναλήψεων του αλγορίθμου Lloyd-Max.

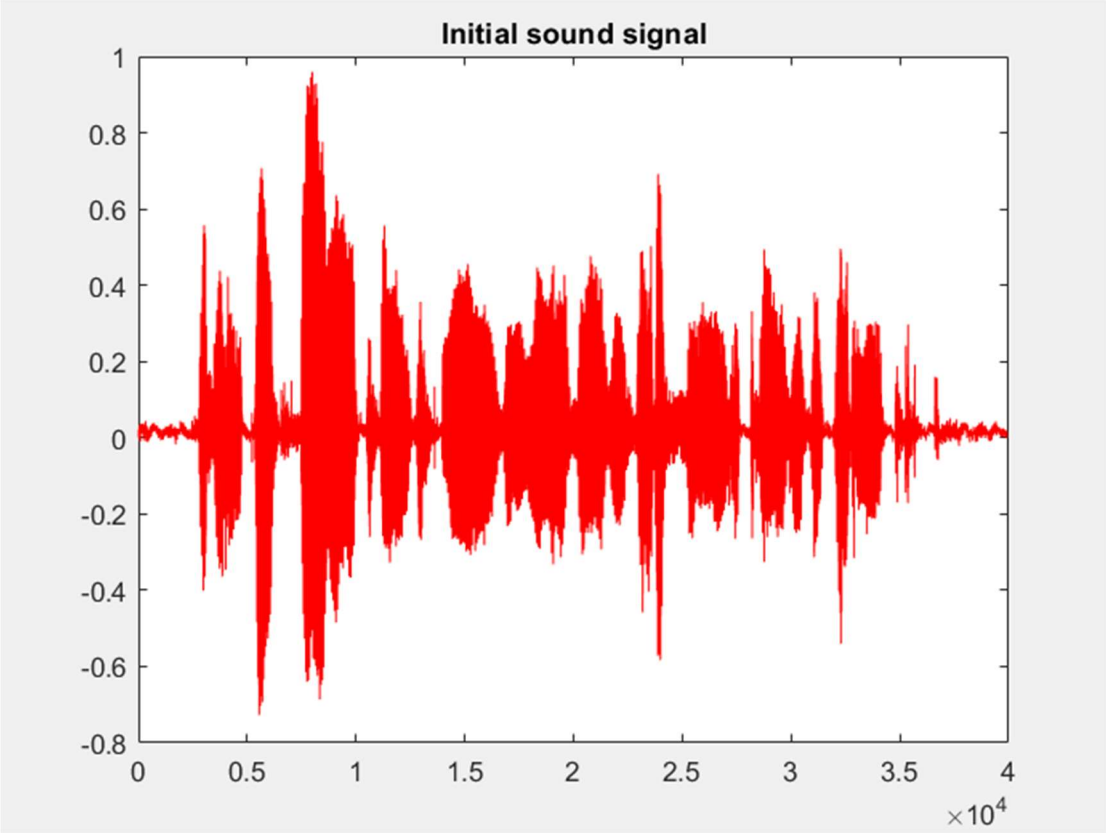




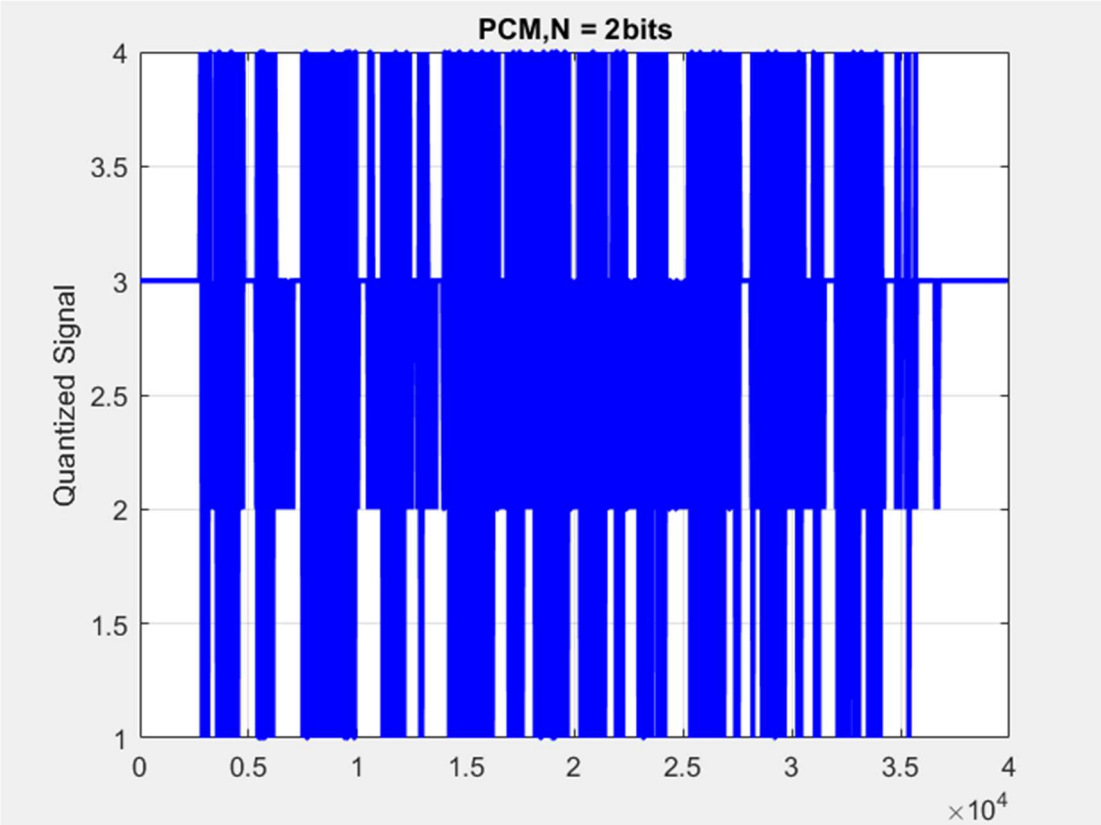


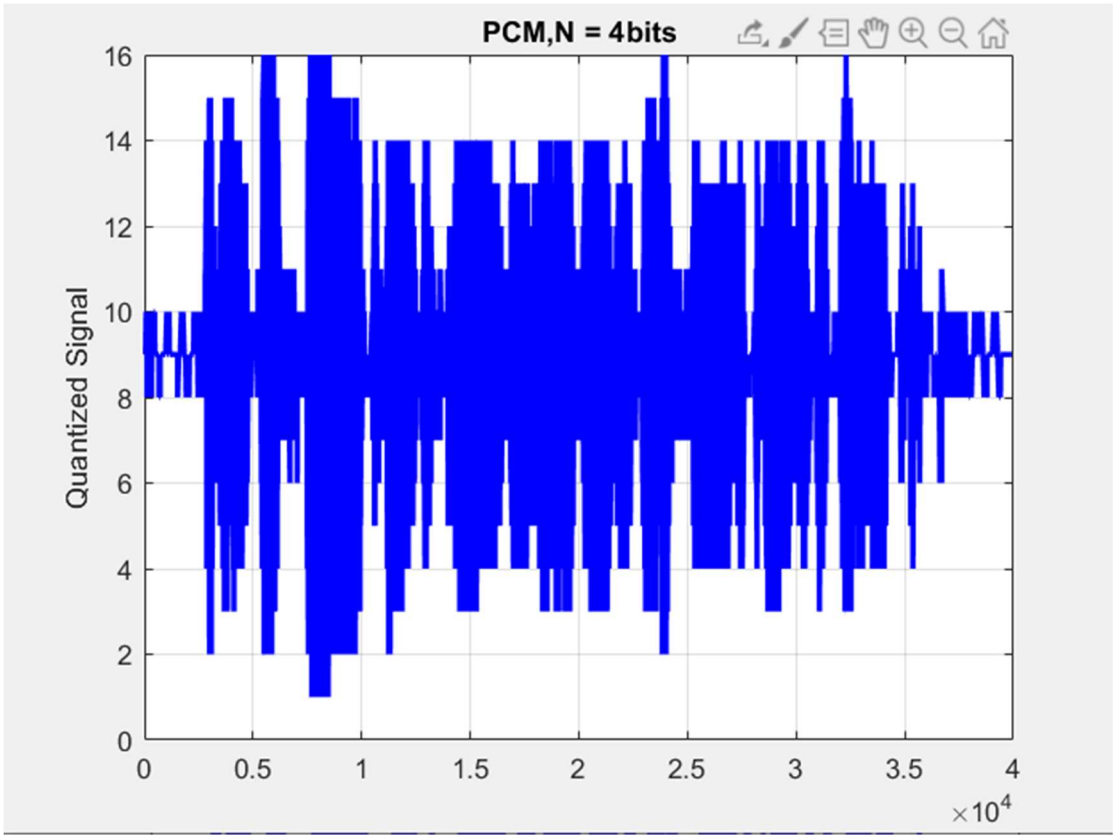
Η ADM για την πηγή ήχου μετρήθηκε ότι έχει SQNR 6.2282dB

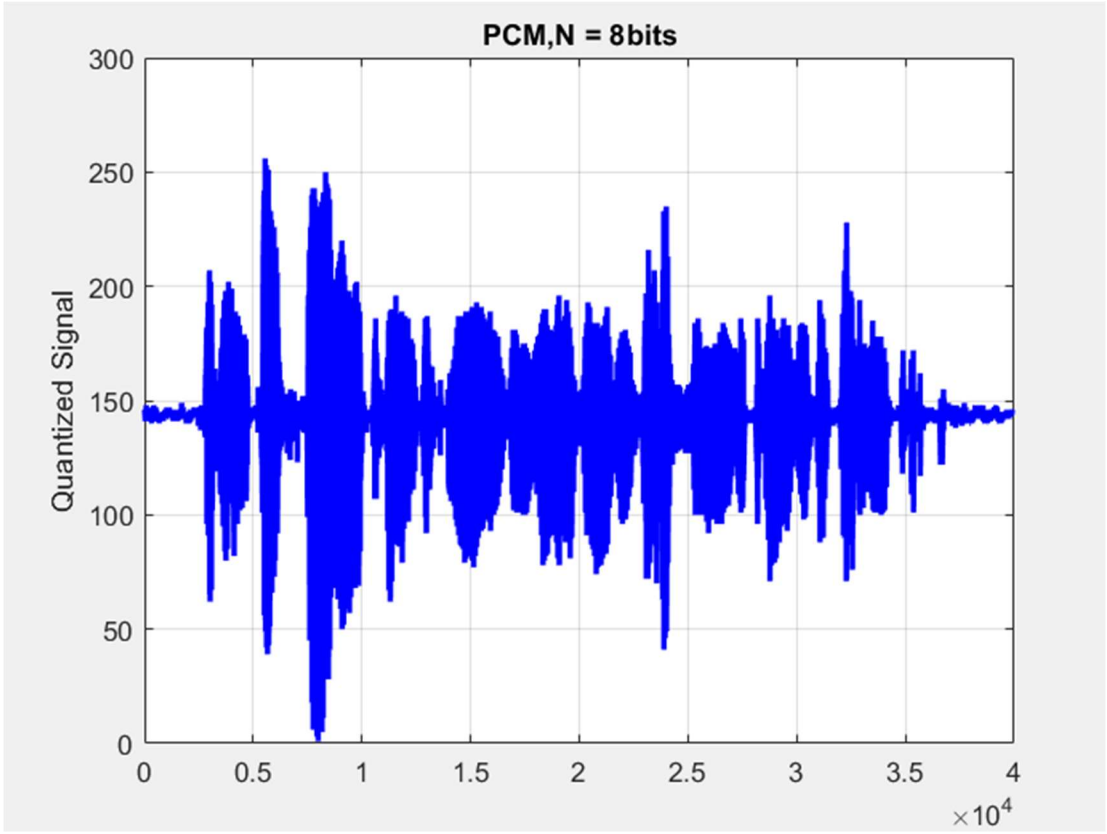
Παρακάτω φαίνεται το αρχικό σήμα:

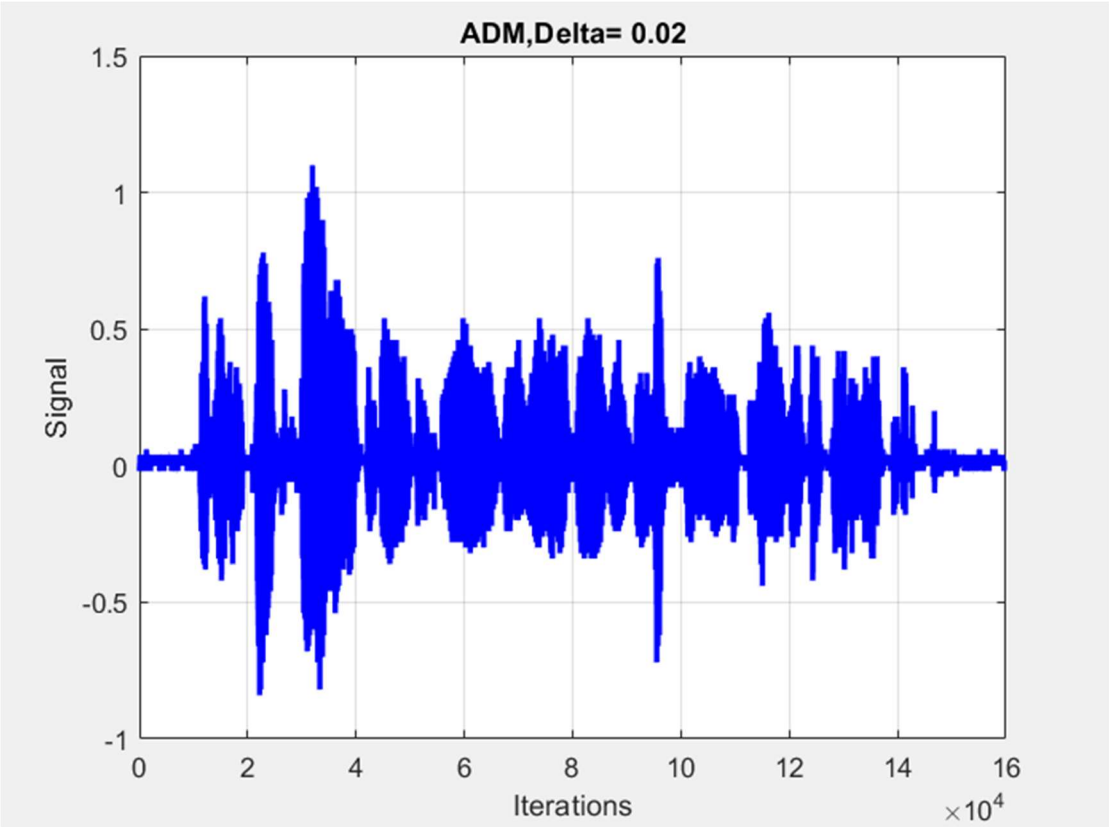


Ακολουθούν τα κβαντισμένα σήματα για PCM με N = 2,4,8 και ADM.




PCM,N = 4bits
Quantized Signal
16
14
12
10
8
6
4
2
0
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4


PCM,N = 8bits
Quantized Signal
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4



Με βάση τις τιμές του SQNR η PCM για N=8 bits είναι καλύτερη, ακολουθείται από την PCM για N=4 bits, μετά από την ADM και τέλος από την PCM για N=2 bits

Με βάση τις κυματομορφές εξόδου οι καλύτερες είναι οι ADM και PCM για N=8 bits και ακολουθούνται από την PCM για N=4 bits και N=2bits.

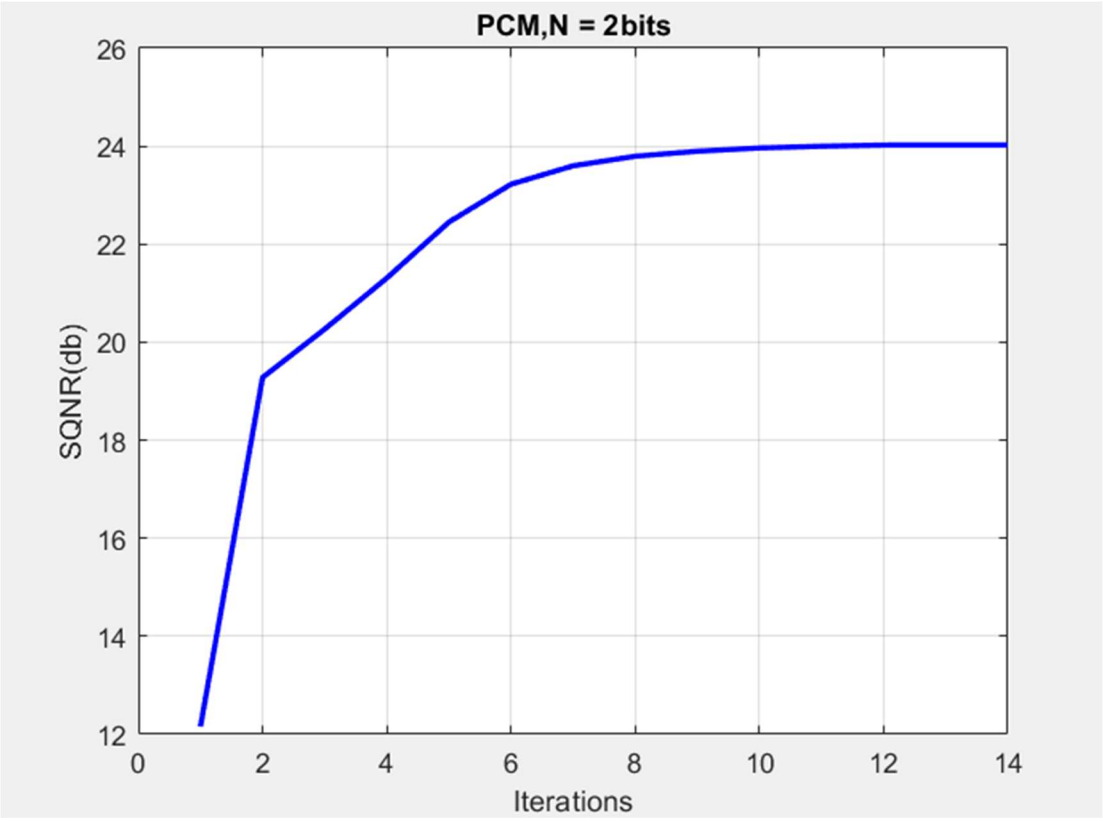
Ακουστικά η κατάταξη είναι η ίδια με την PCM για N=8 bits να είναι οριακά καλύτερη της ADM. Το αποτέλεσμα της PCM για N=2 bits και N=4 bits έχει πολύ μεγάλο ποσοστό θορύβου κάτι που κάνει το ακουστικό αποτέλεσμα εξαιρετικά κακής ποιότητας.

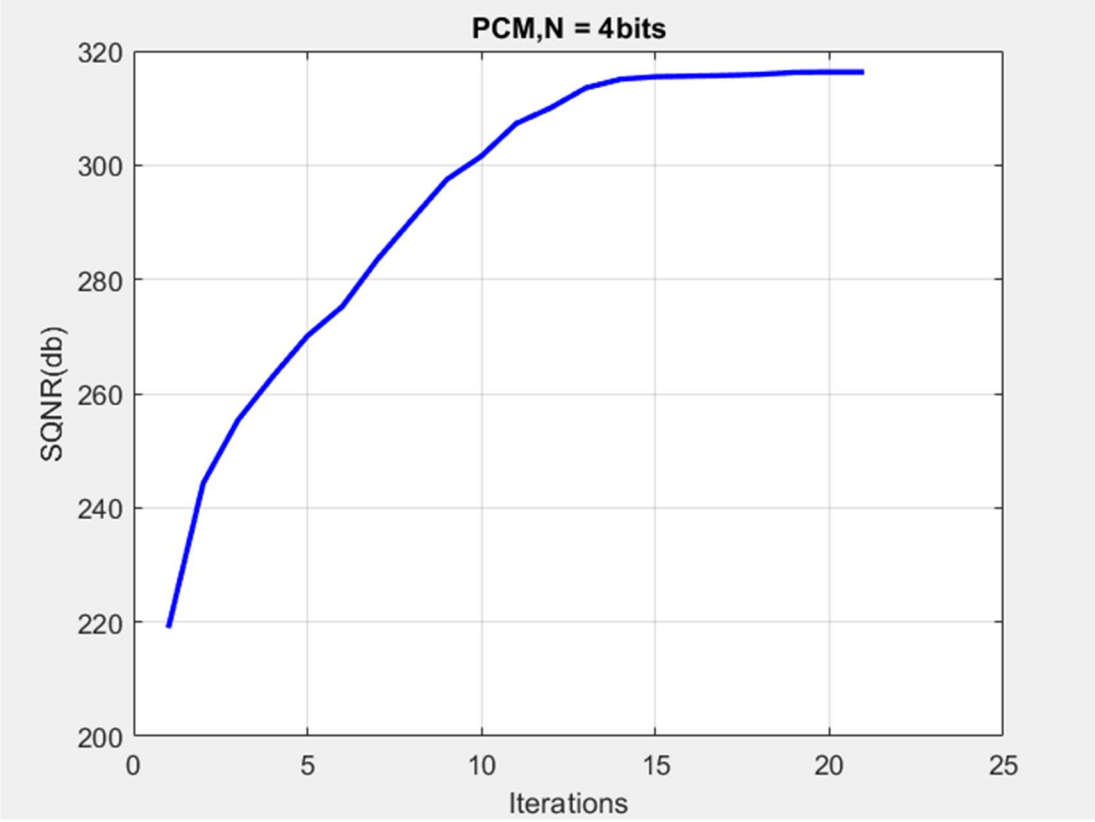
Για την PCM υπολογίστηκε η εντροπία στην έξοδο του κβαντιστή και παρατίθεται παρακάτω με στρογγυλοποίηση στο τέταρτο δεκαδικό

PCM bits	Εντροπία
2	1.4355
4	3.0979
8	6.1246

Η ίδια διαδικασία έγινε για την πηγή Β, δηλαδή την εικόνα

Για την εικόνα παρατίθενται οι γραφικές παραστάσεις μεταβολών του SQNR συναρτήσει του αριθμού επαναλήψεων του αλγορίθμου Lloyd-Max.





Με βάση το SQNR βλέπουμε ότι για N=4 bits η PCM είναι σημαντικά καλύτερα απ'ότι για N=2 bits. Αυτό επιβεβαιώνεται και από το οπτικό αποτέλεσμα όταν ανακατασκευαστούν οι εικόνες

Για N=2 bits η εντροπία στην έξοδο του κβαντιστή στρογγυλοποιημένη στο τέταρτο δεκαδικό είναι 1.9687 ενώ για N=4 bits είναι 3.4151.

Για να υπολογιστεί πειραματικά και θεωρητικά η πιθανότητα εμφάνισης κάθε στάθμης στην έξοδο του κβαντιστή έχει γραφτεί κατάλληλος κώδικας που τις υπολογίζει ο οποίος βρίσκεται μέσα στο Script.

Οι θεωρητικές πιθανότητες παρατίθενται στον παρακάτω πίνακα:

Στάθμη	Θεωρητική πιθανότητα
1	0.0407
2	0.2779
3	0.4695
4	0.2120

Οι πειραματικές πιθανότητες παρατίθενται στον παρακάτω πίνακα:

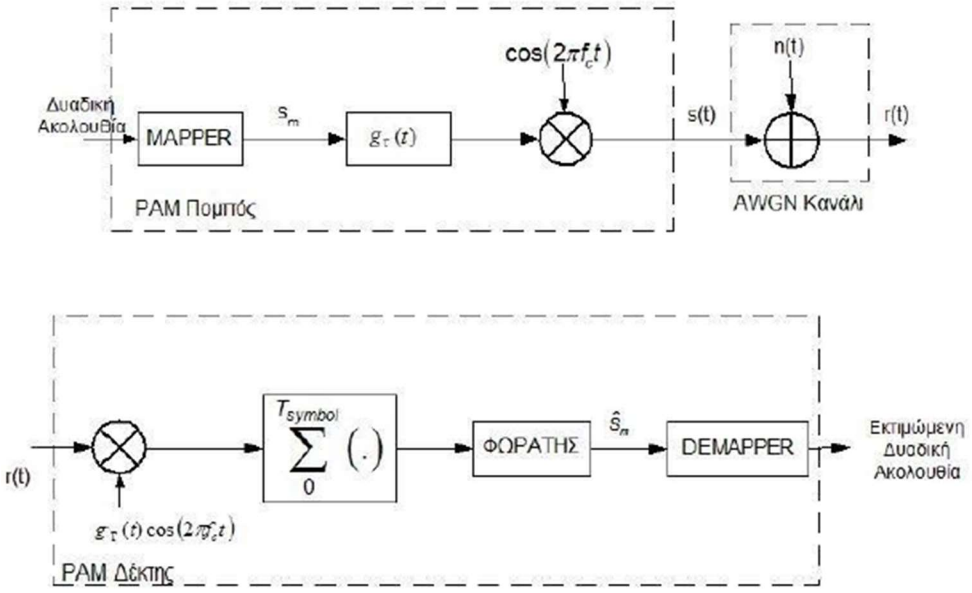
Στάθμη	Πειραματική πιθανότητα
1	0.0418
2	0.1654
3	0.6454
4	0.1474

Παρατηρώ ότι υπάρχει κάποια απόκλιση ανάμεσα στις θεωρητικές και τις πειραματικές τιμές, κάτι που είναι αναμενόμενο.

ΜΕΡΟΣ 2

Για το δεύτερο μέρος έπρεπε να υλοποιηθεί προσομοίωση τηλεπικοινωνιακού συστήματος M-PAM ώστε να μετρηθεί η απόδοση για M=4 και M=8 αντίστοιχα.

Σε ένα σύστημα M-PAM ο πομπός δέχεται ως είσοδο μια δυαδική ακολουθία, τη μετατρέπει σε σύμβολα, την πολλαπλασιάζει με τον ορθογώνιο παλμό, και κατόπιν το σήμα μεταφέρεται στη ζώνη μετάδοσης μέσω του διαμορφωτή. Στο σήμα που στάλθηκε προστίθεται AWGN θόρυβος, και φθάνει στο δέκτη του M-PAM συστήματος. Εκεί αποδιαμορφώνεται και προκύπτει ένα μονοδιάστατο διάνυσμα το οποίο εισάγεται στο φωρατή όπου και αποφασίζεται ποιο σύμβολο στάλθηκε. Τέλος, ο demapper κάνει την αντίστροφη αντιστοίχιση από σύμβολα σε bits.



Σαν είσοδο στο σύστημα δίνω μια ακολουθία εισόδου των 10^5 bits η οποία δημιουργείται τυχαία μέσω της συνάρτησης randsrc.

Ο mapper είναι ένας μετατροπέας από bits σε σύμβολα. Κάθε σύμβολο αντιστοιχεί σε μια συγκεκριμένη ακολουθία $\log_2 M$ bits. Δηλαδή ο mapper για κάθε log2M bits εξάγει ένα από τα σύμβολα της διαμόρφωσης M-PAM. Αντίστοιχα ο demapper δέχεται ως είσοδο σύμβολο και εξάγει μια ακολουθία log2M bits για κάθε σύμβολο.

Στην κωδικοποίηση Gray αν δυο σύμβολα είναι γειτονικά τότε σε αυτά ανατίθενται διατάξεις bits που διαφέρουν μόνο κατά 1 bit μεταξύ τους.

Στην υλοποίηση μου χρησιμοποιώ τις συναρτήσεις bi2de και de2bi αντιστοίχως σε mapper και demapper ενώ για την περίπτωση κωδικοποίησης Gray χρησιμοποιώ τις bin2gray και gray2bin.

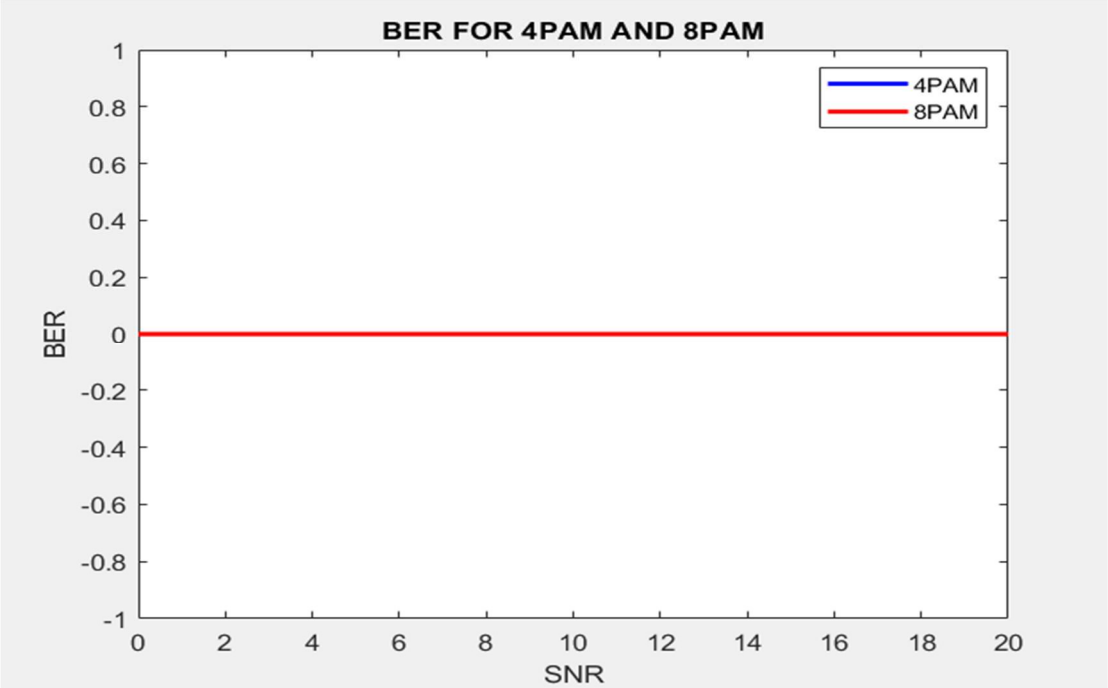
Η μετάδοση γίνεται μέσω ορθογώνιου παλμού τον οποίο προσομοιώνουμε. Για την δημιουργία του ορθογώνιου παλμού χρησιμοποιείται στο modulation ο τύπος g_T που έχει δοθεί στην εκφώνηση.

Το ζωνοπερατό σήμα που εκπέμπει ο πομπός διέρχεται μέσα από ένα ιδανικό κανάλι προσθετικού θορύβου. Ο θόρυβος είναι λευκός και ακολουθεί Gaussian κατανομή μέσης τιμής. Τον παράγω με κλήση της συνάρτησης randn.

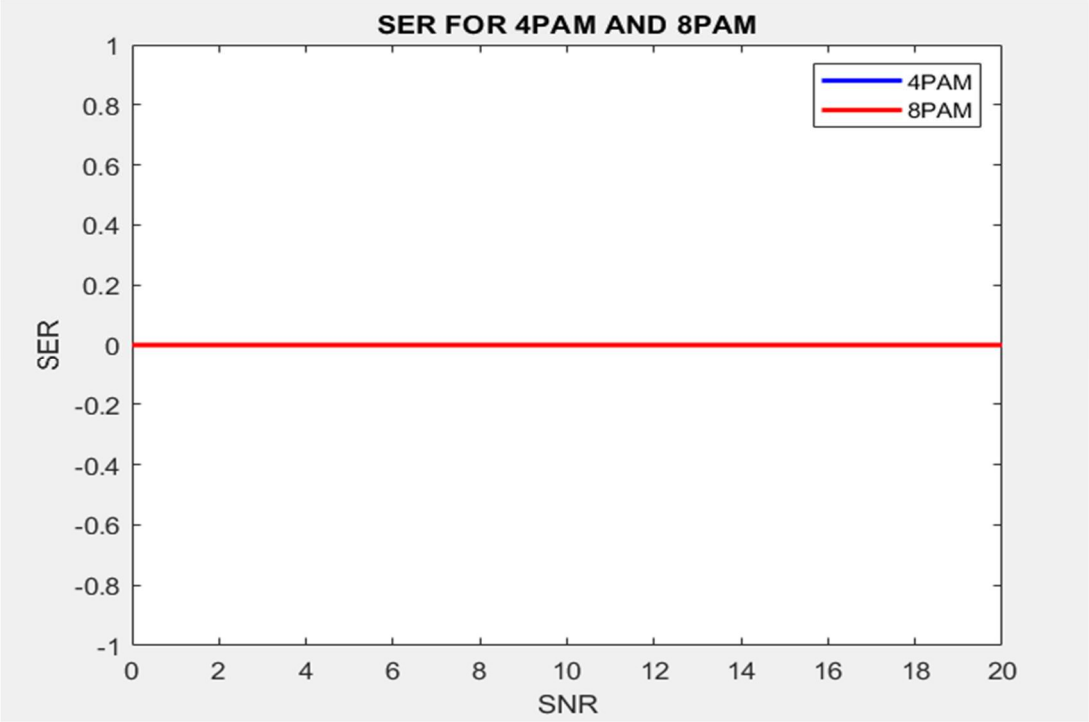
Ο αποδιαμορφωτής του συστήματος M-PAM συσχετίζει το ληφθέν σήμα με τη φέρουσα και τον ορθογώνιο παλμό. Η συσχέτιση γίνεται στα χρονικά πλαίσια μιας περιόδου συμβόλου. Ο αποδιαμορφωτής συσχετίζει το σήμα με την συνιστώσα της φέρουσας οπότε προκύπτει το διάνυσμα r που είναι η εκτιμηθείσα τιμή του τρέχοντος συμβόλου.

Ο φωρατής με βάση την τιμή r αποφασίζει σε ποιο σύμβολο βρίσκεται πιο κοντά. Το σύμβολο που θα έχει την μικρότερη απόσταση από το r είναι τελικά το σύμβολο που στάλθηκε.

Ακολουθούν οι καμπύλες BER για M = 4 για απλή κωδικοποίηση για τιμές του SNR=0:2:20db και για τις ίδιες τιμές SNR για M=8 σε κωδικοποίηση Gray στο ίδιο γράφημα



Ακολουθούν οι καμπύλες SER για M = 4,8 για απλή κωδικοποίηση για τιμές του SNR=0:2:20db



SER και BER, πιθανώς λόγω προβλήματος στην υλοποίηση, είναι 0 σε όλες τις περιπτώσεις.

ΠΑΡΑΡΤΗΜΑ

Εδώ παρατίθενται οι κώδικες Matlab για τα ζητούμενα.

PCM.M

```
function [xq,centers,D] = PCM(x,N,min,max)
% INPUT ------------------------------------------------------------------
%   x: Input signal in vector format
%   N: Number of bits
%   min: Minimum acceptable value of input signal
%   max: Maximum acceptable value of input signal
% OUTPUT -----------------------------------------------------------------
%   xq: Encoded vector of output signal
%   centers: Centers of quantization segments
%   D: Vector of signal's distortion at every repetition
% ------------------------------------------------------------------------

% Number of bits used.
v = N;
% Levels of quantization.
quant_levels = 2^v;
% Initialization of vector xq.
xq = zeros(length(x),1);
% Quantization step.
quant_step = (abs(min)+max)/quant_levels;
% Calculation of centers.
centers = zeros(quant_levels,1);
for i=1:quant_levels
    centers(i) = max-(2*(i-1)+1)*(quant_step/2);
end
% Loop for Lloyd-Max.
T = zeros((quant_levels+1),1);
counter = 1;
previous_distortion = 0;
```

```
while 1
    %Calculation of new segments' limits.
    T(1) = max; % Higher level limit.
    for i=2:quant_levels % Middle levels.
        T(i) = (centers(i)+centers(i-1))/2;
    end
    T(quant_levels+1) = min; % Lower level limit.

    %Output signal
    q_max_value = 1; q_min_value = quant_levels;
    for i=1:length(x)
        if (x(i) >= max)
            xq(i) = q_max_value;
        elseif (x(i) <= min)
            xq(i) = q_min_value;
        else
            for n=1:quant_levels
                if ((x(i) <= T(n)) && (x(i) > T(n+1)))
                    xq(i) = n;
                end
            end
        end
    end

    % Check if there are any zeros.
    if (all(xq)) % If not, calculate the distortion.
        D(counter) = mean((x-centers(xq)).^2);
    end
```

```
    SQNR(counter)=mean(x.^2)/D(counter);
    %Criterion check. If fulfilled, stop the loop.
    difference = abs(D(counter)-previous_distortion);
    if (difference < eps('single')&&N<=4)
        break;
    elseif (difference < eps()&&N>4)
        break;
    else % Else store distortion for the next comparison.
        previous_distortion = D(counter);
    end

    %New levels of quantization.
    temp_sum = zeros(quant_levels,1);
    temp_counter = zeros(quant_levels,1);
    for n=1:quant_levels
        for i=1:length(x)
            % Check if x(i) belongs to n level.
            if (x(i) <= T(n) && x(i) > T(n+1))
                temp_sum(n) = temp_sum(n) + x(i);
                temp_counter(n) = temp_counter(n) + 1;
            % If x(i) is greater than max_value.
            elseif ((x(i) > T(n)) && (n == 1))
                temp_sum(n) = temp_sum(n) + T(n);
                temp_counter(n) = temp_counter(n) + 1;
            % If x(i) is smaller than min_value.
            elseif ((x(i) < T(n+1)) && (n == quant_levels))
                temp_sum(n) = temp_sum(n) + T(n+1);
                temp_counter(n) = temp_counter(n) + 1;
            end
```

```
        end
        % Calculating the new center for n level.
        if (temp_counter(n) > 0) % If greater than zero, calculate.
            centers(n) = temp_sum(n)/temp_counter(n);
        end

    end

    counter = counter + 1;
end
stoixeio = unique(xq);


[emfaniseis,~] = hist(xq,stoixeio);

pithanotita =(emfaniseis/length(xq)) ;

Entropia = -pithanotita*log2(pithanotita)';

fprintf('H entropia gia N = %d einai %d \n',N,Entropia);
figure;
plot(SQNR,'-b','LineWidth',2);
title(['PCM,N = ',num2str(N),'bits']);
ylabel('SQNR(db)')
xlabel('Iterations')
grid on;

figure;
plot(xq,'-b','LineWidth',2);
```

```
plot(xq,'-b','LineWidth',2);
title(['PCM,N = ',num2str(N),'bits']);
ylabel('Quantized Signal')
hold off;
grid on;
```

ADM.M

```
function [ADMout] = ADM(sig_in, Delta, td, ts)
% INPUT -------------------------------------------------------------------
%   sig_in: Input signal in vector format
%   Delta: Minimum step size
%   td: The original sampling period of the input signal, sig_in
%   ts: The required sampling period for ADM output
% OUTPUT ------------------------------------------------------------------
%   ADMout:Encoded vector of output signal


    if (round(ts/td) >= 2)
        M = round(ts/td);    %Nearest integer
        xsig = interp(sig_in,M);
        Lxsig = length(xsig);

        cnt1 = 0;
        cnt2 = 0;
        sum = 0;
        for i=1:Lxsig

            if (xsig(i) == sum)
            elseif (xsig(i) > sum)
                if (cnt1 < 2)
                    sum = sum + Delta;  %Step up by Delta
                elseif (cnt1 == 2)
                    sum = sum + 2*Delta;    %Double the step size after
                                            %first two increase
                elseif (cnt1 == 3)
                    sum = sum + 4*Delta;    %Double step size
```

```
                else
                    sum = sum + 8*Delta; %Still double and then stop

                end
                if (sum < xsig(i))
                    cnt1 = cnt1 + 1;
                else
                    cnt1 = 0;
                end
            else
                if (cnt2 < 2)
                    sum = sum - Delta;
                elseif (cnt2 == 2)
                    sum = sum - 2*Delta;
                elseif (cnt2 == 3)
                    sum = sum - 4*Delta;
                else
                    sum = sum - 8*Delta;
                end
                if (sum > xsig(i))
                    cnt2 = cnt2 + 1;
```

```
                else
                    cnt2 = 0;
                end
            end
             D(Lxsig)= mean(abs(xsig - sum).^2);
            SQNR(Lxsig-1)=mean(xsig.^2)/D(Lxsig);
            SQNR(Lxsig-1)=10*log10(SQNR(Lxsig-1));
            ADMout(((i-1)*M + 1):(i*M)) = sum;

        end
    end
fprintf('To SQNR ths ADM einai %d \n',SQNR(Lxsig-1));
figure;
plot(ADMout,'-b','LineWidth',2);
title(['ADM,Delta= ',num2str(Delta)]);
ylabel('Signal')
xlabel('Iterations')
grid on;
```

SCRIPT.M

Με εκτέλεση αυτού γίνονται αυτόματα οι υπολογισμοί του πρώτου μέρους

```
clear;
[y,fs] = audioread('speech.wav');
load cameraman.mat;
x = i(:);
x = (x-128)/128;

figure;
plot(y,'-r');
title('Initial sound signal');

[xq_2,centers_2,D_2] = PCM(y,2,min(y),max(y));
[xq_4,centers_4,D_4] = PCM(y,4,min(y),max(y));
[xq_8,centers_8,D_8] = PCM(y,8,min(y),max(y));

[ADM]= ADM(y,0.02,8,16);
fprintf('Source 2 \n');

figure;
plot(x,'-r');
title('Initial image signal');

[xq_2_image,centers_2_image,D_2_image] = PCM(x,2,min(x),max(x));
[xq_4_image,centers_4_image,D_4_image] = PCM(x,4,min(x),max(x));

%Gia therhtiko kai peiramatiko ypologismo emfanishs kathe stathmhs sthn
%eksodo kvantisth gia N=2

```

```
28      % Ypologismos Delta
29 -    Del = (max(y)-min(y))/2^2 ;
30
31
32 -    m = -0.04;
33
34 -    d = sqrt(0.11);
35
36
37      % Ypologismos oriwn
38 -    limit1 = max(y) - Del;
39 -    limit2 = max(y) - 2*Del;
40 -    limit3 = max(y) - 3*Del;
41 -    limit4 = inf;
42
43      % Pdf function
44 -    p4 = normcdf(limit3,m,d) ;
45 -    p3 = normcdf(limit2,m,d) - p4;
46 -    p2 = normcdf(limit1,m,d) -p3-p4;
47 -    p1 = normcdf(limit4,m,d) -p2-p3-p4;
48
49      % Anathesi timwn se ena dianusma
50 -    theoritical_prop = [p1 p2 p3 p4];
51
52      % Ypologismos peiramatikwn timwn
53      % gia sunarthsh puknotitas pithanotitas
54 -     experimental_prop = hist(xq_2, max(xq_2) - min(xq_2) + 1) / length(xq_2) ;
```

MPAM.M

```
function [out,SER,BER] = MPAM(lin,M,SNR,encoding)
    if ~strcmp(encoding,'bin') && ~strcmp(encoding,'gray')
        error('Encoding should be bin or gray');
    end
%-------------------------MAPPER-----------------------------------------
    k=log2(M);
    Fc=2.5*10^6;
    Tsym=4*10^(-6);
    Tsample=Tsym/40;
    gt=sqrt(1/2)*10^3;
    sm =zeros(length(lin),1);
    blocks_in=reshape(lin,[],k);
    if strcmp(encoding,'bin')
        m = bi2de(blocks_in,'left-msb')+1; % lowest value of sm shoul be 1
    else
        m = bi2de(blocks_in,'left-msb');
        m=bin2gray(m,'pam',M)+1;           % lowest value of sm should be 1
    end
%-------------Modulation PAM-----------------------------------------------
    Es=1;
    if M==4
    A=1/sqrt(5);
    elseif M==8
    A=1/sqrt(21);
    end
    sm=(m.*2-1-M)*A;
    symbols=size(sm,1);
    s=zeros(symbols*40,1);
    t=(0:Tsample:(Tsym-Tsample))';
```

```
    for i=1:symbols
        for j=1:40
            s((i-1)*40+j)=sm(i)*gt*cos(2*pi*Fc*t(j));
        end
    end
%-----------------------------AWGN-----------------------------------------
    No=Es/(log2(M)*power(10,SNR/10));
    s2=No/2;
    noise=sqrt(s2)*randn(length(m)*40,1);
    r=s+noise;

%--------------------------Demodulation-------------------------------------
    for i=1:length(r)
        r(i)=r(i)*gt*cos(2*pi*Fc*(i-1)*Tsample)*Tsample;
    end

    demodulated=zeros(symbols,1);
    for i=1:symbols
        ind=((i-1)*40)+1;
        demodulated(i)=sum(r(ind:(ind+39)));
    end
```

```
%------------------Decision--------------------------------------------------
    constellation=zeros(M,1);
    for i=1:M
        constellation(i)=((2*i)-1-M)*A;
    end
    decision=zeros(symbols,1);
    for i=1:symbols
        dist=abs(constellation-demodulated(i));
        [~,ind]=min(dist);
        decision(i)=ind;
    end
    errorsym=0;
    totalsymb=0;
    for i=1:symbols
        if decision(i)~=m(i)
            errorsym=errorsym+1;
        end
        totalsymb=totalsymb+1;
    end
%---------------------------------Demapper---------------------------------
    if strcmp(encoding,'bin')
        blocks_out=de2bi(decision-1,k,'left-msb');
    else
        decision=gray2bin(decision-1,'pam',M);
        blocks_out=de2bi(decision,k,'left-msb');
    end

    output_sequence=reshape(blocks_out,[],1);

```

```

    k=lin(lin~=output_sequence);
    BER=length(k)/length(lin);
    out=output_sequence;
    SER=errorsym/totalsymb;
end
```

SCRIPTPAM.M

Με εκτέλεση αυτού γίνονται αυτόματα οι υπολογισμοί του δεύτερου μέρους

```
input=randsrc(10^5,1,[0 1]);
for SNR=0:2:20
    i=i+1;
    [~,SER,BER]=MPAM(input,4,SNR,'bin');
    SERFINAL4(i)=SER;
    BERFINAL4(i)=BER;
    [~,SER,BER]=MPAM(input,8,SNR,'gray');
    SERFINAL8(i)=SER;
    BERFINAL8(i)=BER;
    [~,SER,BER]=MPAM(input,8,SNR,'bin');
    SERFINAL8bin(i)=SER;
    BERFINAL8bin(i)=BER;
end
figure
plot(0:2:20,BERFINAL4,'-b','LineWidth',2);
hold on;
plot(0:2:20,BERFINAL8,'-r','LineWidth',2);
legend('4PAM','8PAM');
hold off;
title('BER FOR 4PAM AND 8PAM');
xlabel('SNR');
ylabel('BER');

figure
plot(0:2:20,SERFINAL4,'-b','LineWidth',2);
hold on;
plot(0:2:20,SERFINAL8,'-r','LineWidth',2);
legend('4PAM','8PAM');
hold off;
```

```
hold off;
title('SER FOR 4PAM AND 8PAM');
xlabel('SNR');
ylabel('SER');
```