Computer Engineering & Informatics Department
University of Patras

Digital Telecommunications
Academic Year 2016-2017

Laboratory Exercise

E-mail: marinou@ceid.upatras.gr

Index

Part A

1. Implementation of the DPCM encoding / decoding system

2. Comment on results for different P and N

3. Average square prediction error with N

4. Representation of an original-rebuilt signal to the receiver and annotations of results

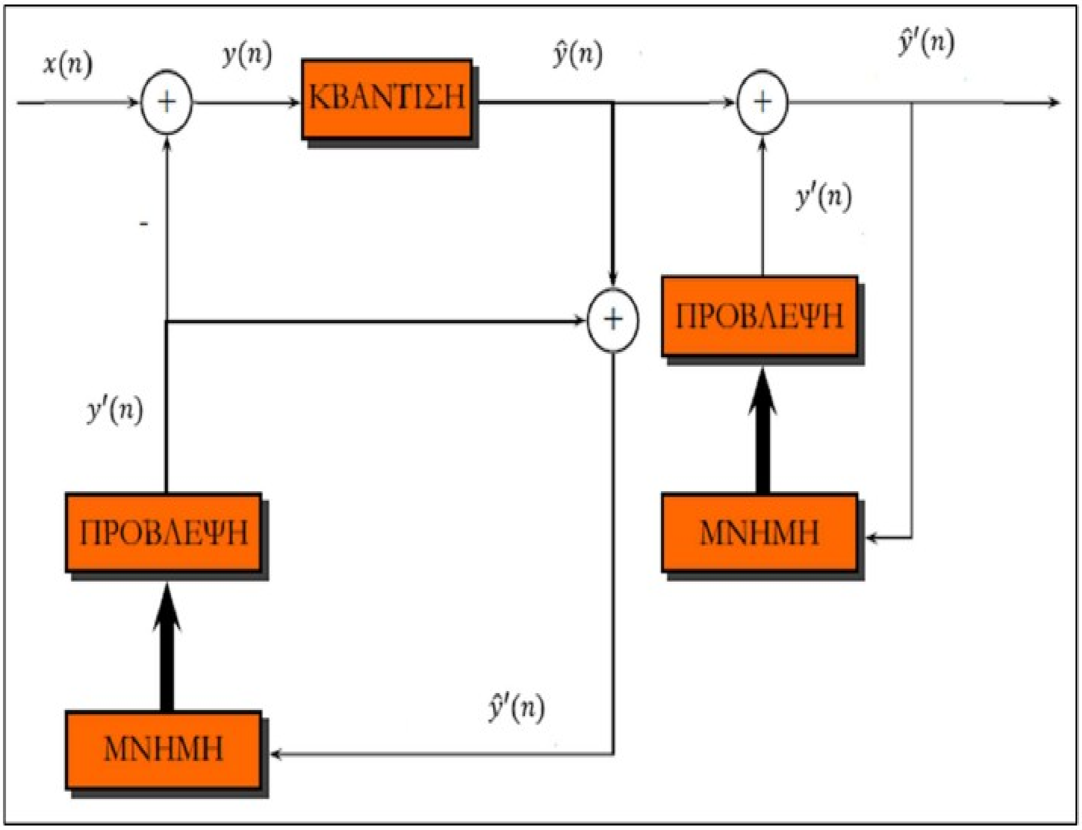
Part B

1. Implementation of the M-PSK system

2. Error probability and BER curve design

Part A

1. Implementation of the DPCM encoding / decoding system

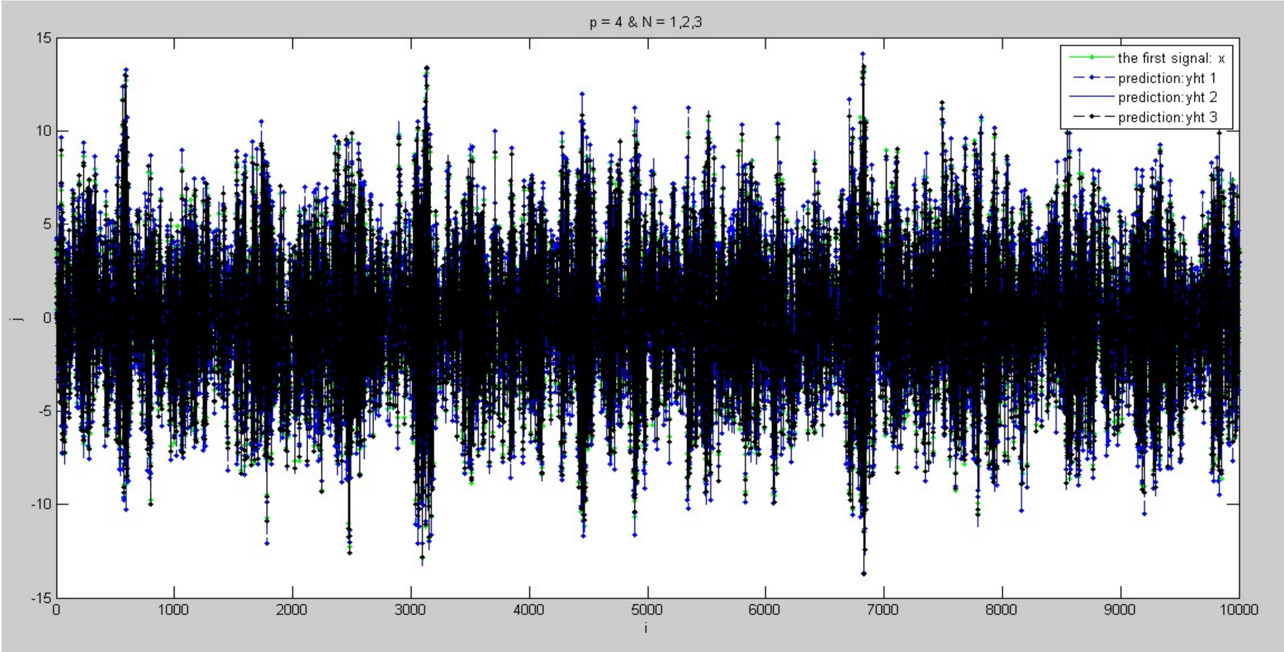


Σχήμα 1. Κωδικοποιητής και Αποκωδικοποιητής DPCM

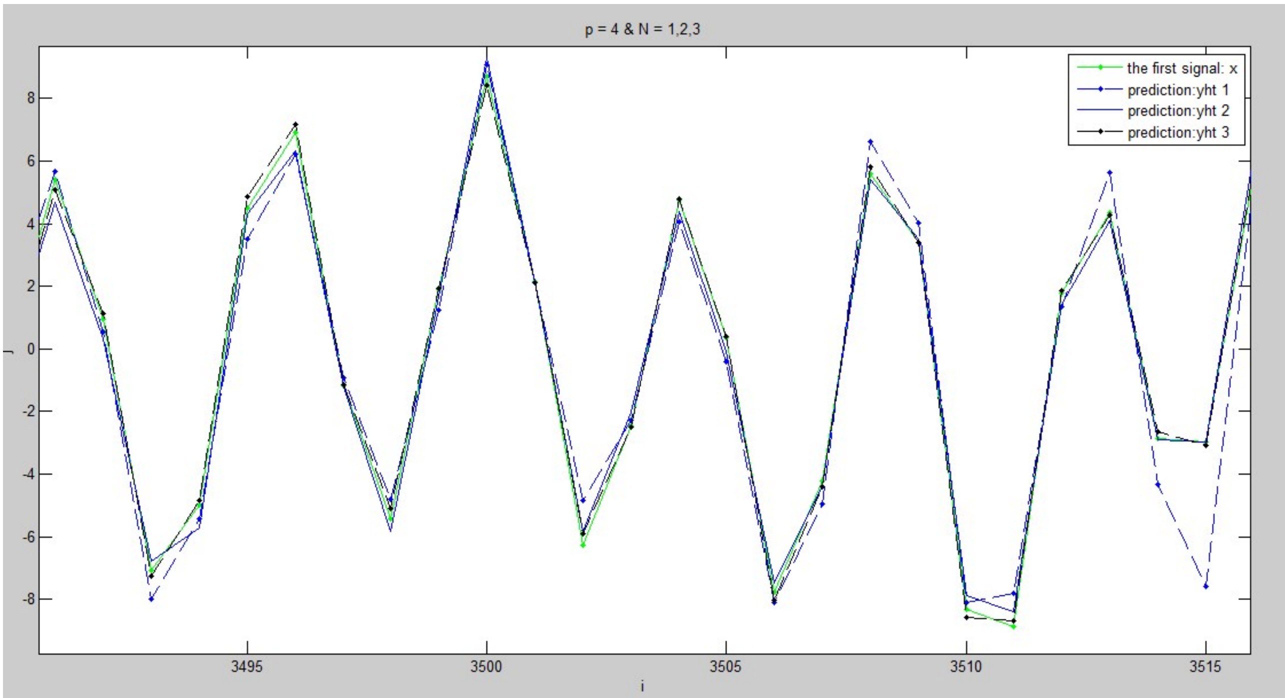
2. Comment on results for different P and N

I choose two values of p and for N = 1,2,3 I have:

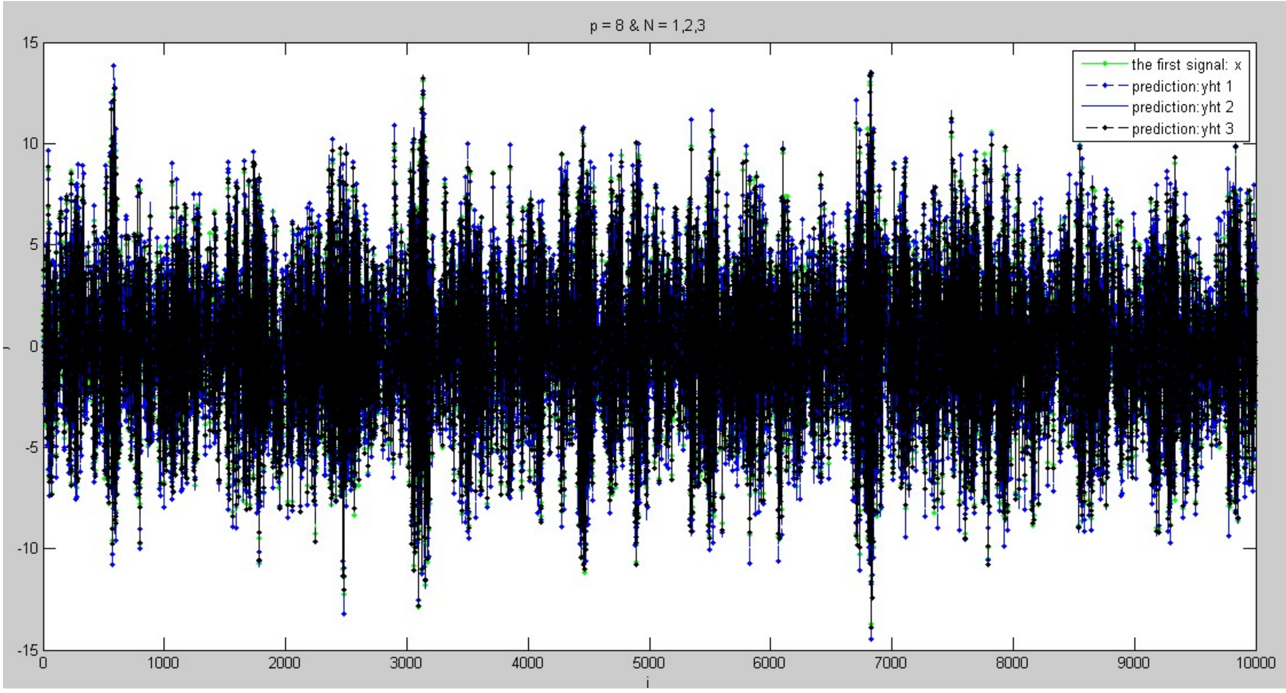
p=4



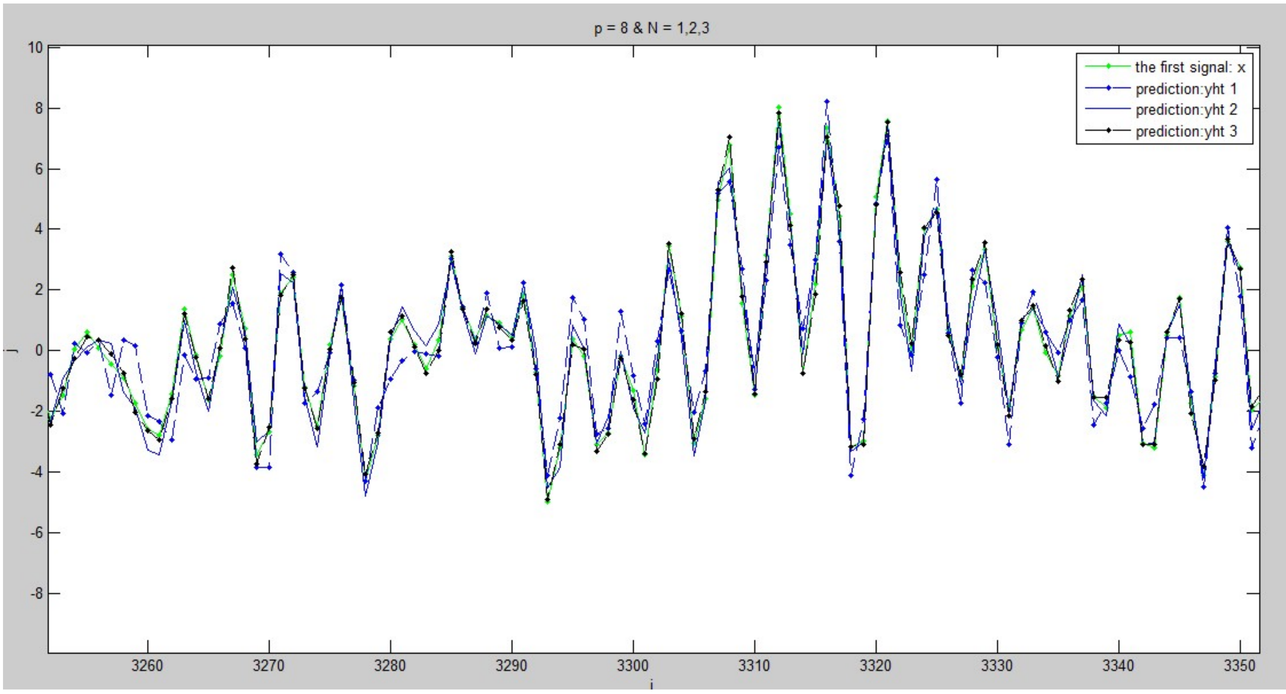
Zoom in the same image:



p=8



Μεγέθυνση της ίδιας εικόνας:



For different N I have a different deviation between the original signal and the prediction signal. The larger the N so smaller the difference between them. Accordingly, the larger the p, the two signals coincide, one tends to approach the other.

3. Average square prediction error with N

Below we give the mean square error of the original signal and the reconstructed signal and we determine the conclusion of the second part: The larger the number of Bits and P is, the smaller the deviation between them.

- The value of p=4 -

- num = 1 => E_y_2 = 4.8069
- num = 2 => E_y_2 = 2.0907
- num = 3 => E_y_2 = 1.3129

- The value of p= 8 -

- num = 1 => E_y_2 = 4.7128
- num = 2 => E_y_2 = 2.0894
- num = 3 => E_y_2 = 1.2975

► For each P = 4: 8 and for N = 1: 3, we recorded the predictor coefficient values which are as follows:

The value of p= 4

-The value of N= 1
a =[1.3889; -1.5214; 1.2124; -0.3017]
//---
-The value of N= 2
a =[1.3889; -1.521; 1.2124; -0.3017]
//---
-The value of N= 3
a =[1.3889; -1.5214; 1.2124;-0.3017]

The value of p= 5

-The value of N= 1
a =[1.3883; -1.5186; 1.2088; -0.2984; -0.0025]
//--
-The value of N= 2
a =[1.3883; -1.5186; 1.2088; -0.2984; -0.0025]
//--
-The value of N= 3
a =[1.3883; -1.5186; 1.2088; -0.2984; -0.0025]

The value of p= 6

-The value of N= 1

a =[1.3882; -1.5193; 1.2118; -0.3023; 0.0011; -0.0026]

//---

-The value of N= 2

a =[1.3882; -1.5193; 1.2118; -0.3023; 0.0011; -0.0026]

//---

-The value of N= 3

a =[1.3882; -1.5193; 1.2118; -0.3023; 0.0011; -0.0026]

The value of p= 7

-The value of N= 1

a =[1.3881; -1.5195; 1.2110; -0.2975; -0.0051; 0.0033; -0.0045]

//---

-The value of N= 2

a =[1.3881; -1.5195; 1.2110; -0.2975; -0.0051; 0.0033; -0.0045]

//---

-The value of N= 3

a =[1.3881; -1.5195; 1.2110; -0.2975; -0.0051; 0.0033; -0.0045]

The value of p= 8

-The value of N= 1

a =[1.3880; -1.5193; 1.2108; -0.2998; 0.0050; -0.0095; 0.0073; -0.0085]

//---

-The value of N= 2

a =[1.3880; -1.5193; 1.2108; -0.2998; 0.0050; -0.0095; 0.0073; -0.0085]

/---

-The value of N= 3

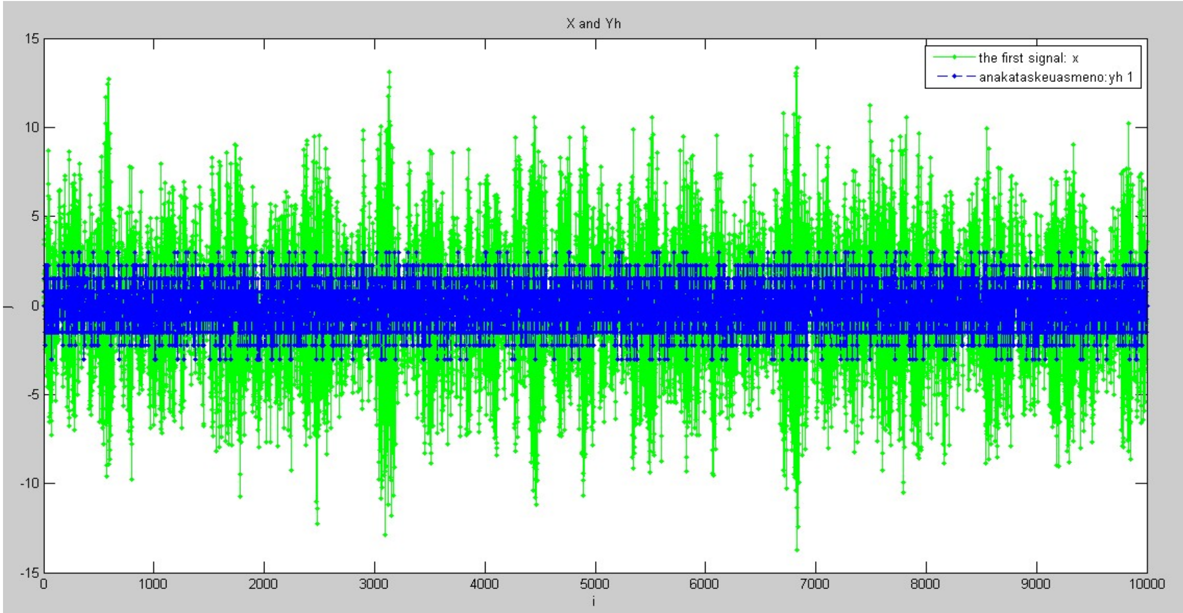
a =[1.3880; -1.5193; 1.2108; -0.2998; 0.0050; -0.0095; 0.0073; -0.0085]

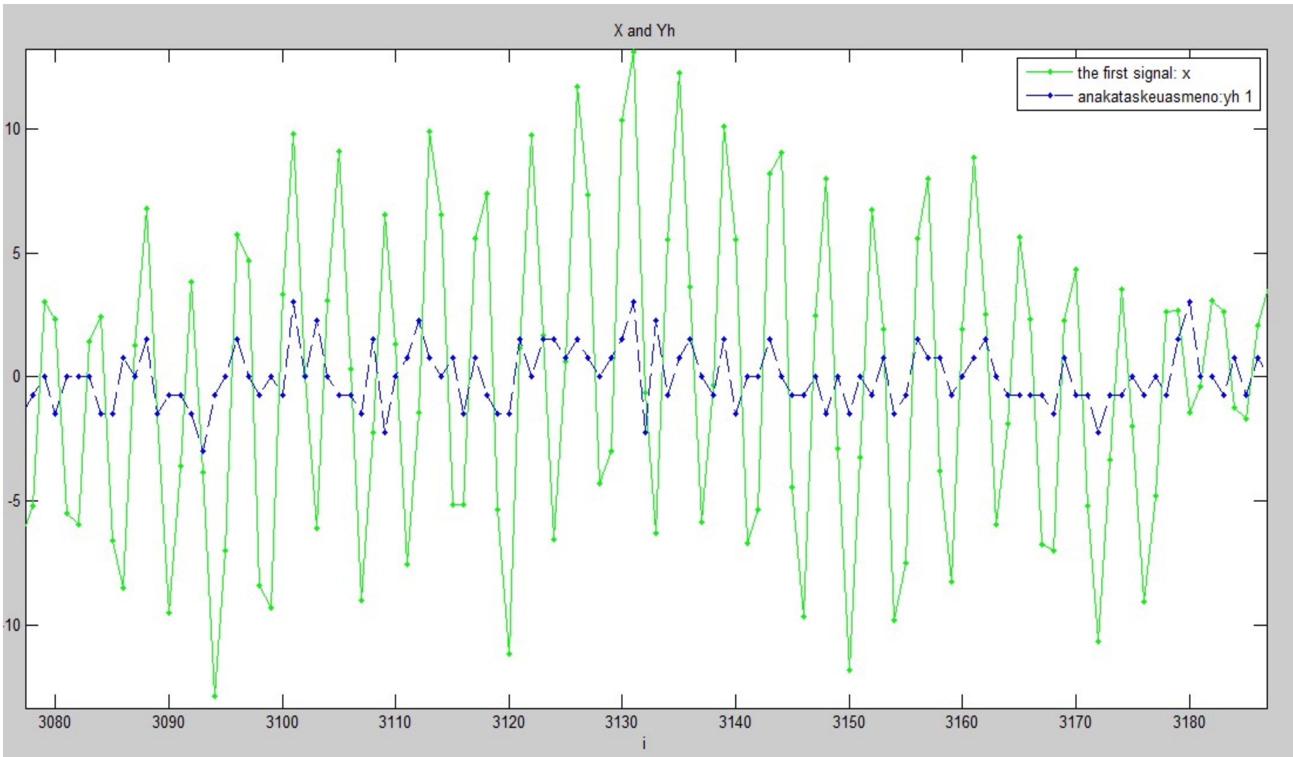
We notice that for each P the coefficients are the same, regardless of the digits we have set. To find the coefficients we used a uniform quantizer with a dynamic range [-2,2], the code-function of which performs the process has the name quantizer and is shown below:

```
function [ x_q ] = quantizer( x, N, min_v, max_v )

    if (x > max_v)
        x = max_v;
    elseif (x < min_v)
        x = min_v;
    end

    lvl = 2^N;

    D = ( max_v - min_v )/lvl;
    x_q = D * floor((x/D) + 0.5);

end
```

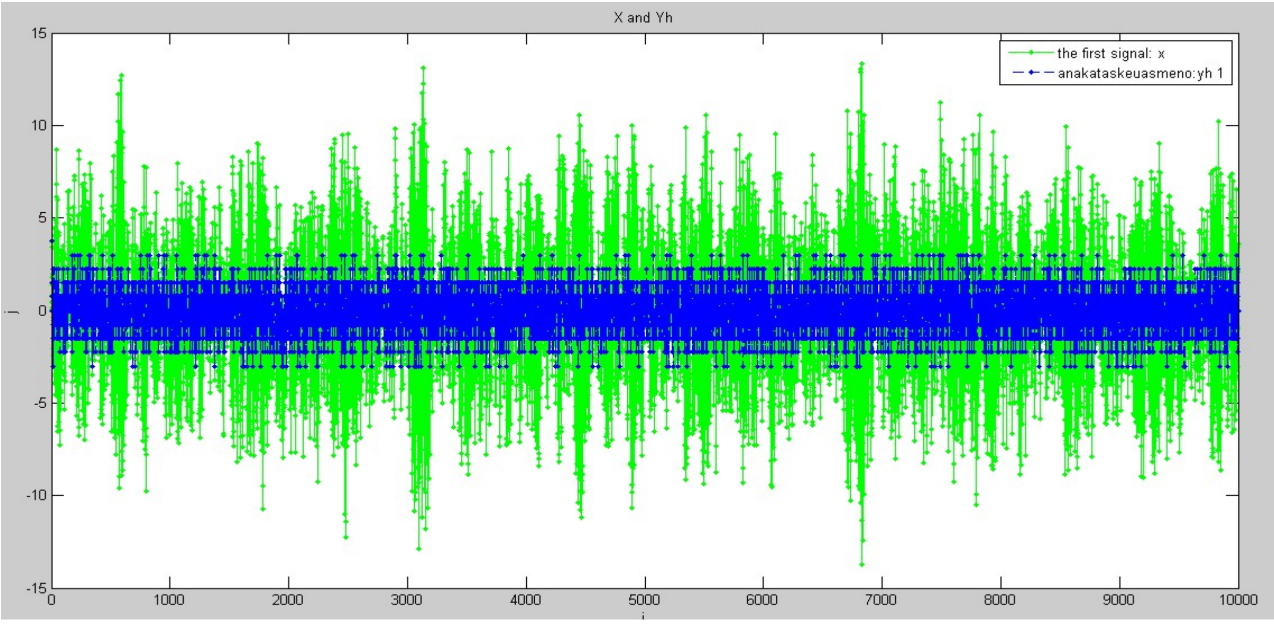
4. Representation of an original-rebuilt signal to the receiver and annotations of results

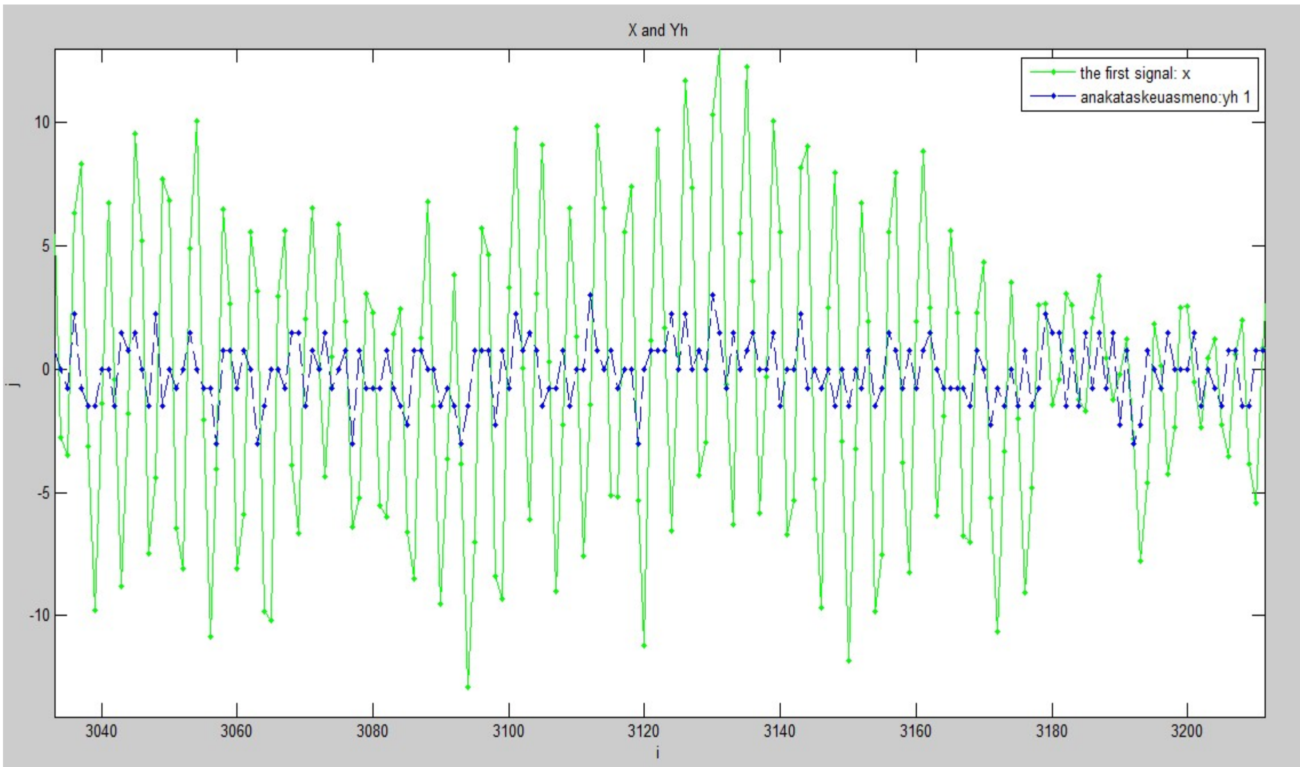
For p=4





For p=8:



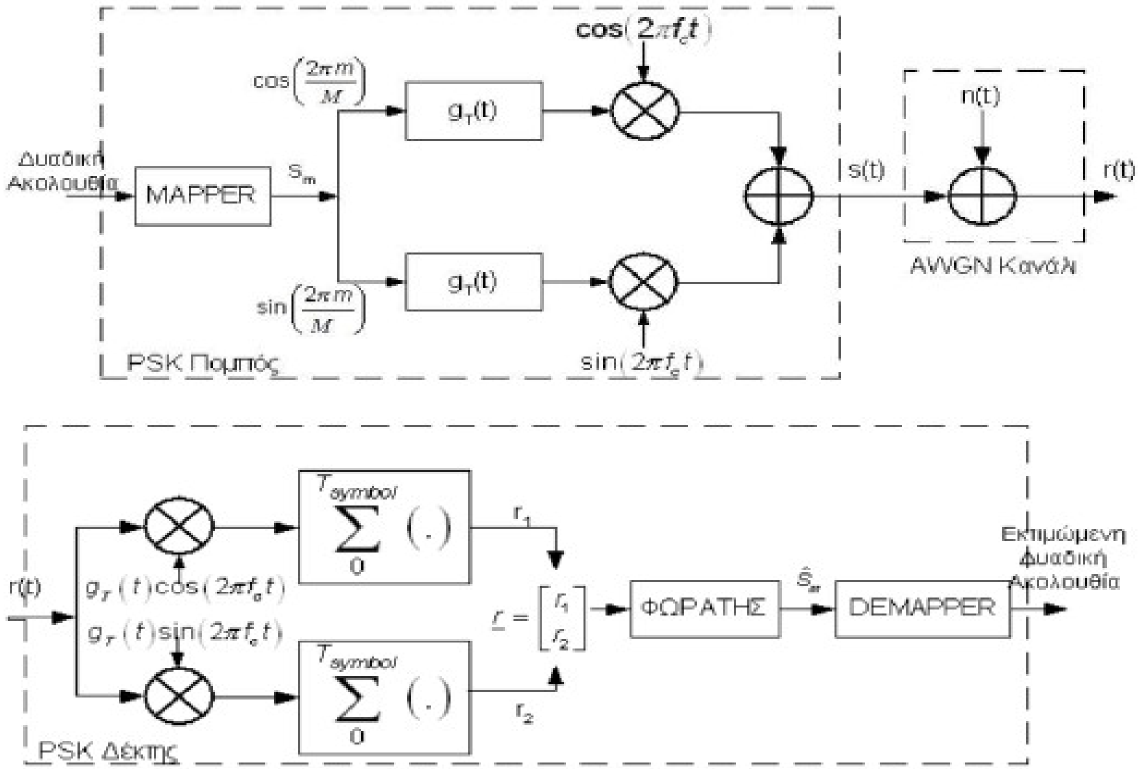


We notice that in the second case where p is larger, the reconstruction is on the original signal, it takes its values with respect to smaller p.

```
function [ ] = main2 ( )

load source.mat
leng = length(x);
data = zeros(leng,3);
data2 = zeros(3,4);
E_y_2 = zeros(3,4);
for p=4:8
   for num=1:3


   %Number of bits we are going to use

   %Allocating memory
   yn = zeros(leng,1);
   yht = zeros(leng,1);
   yh = zeros(leng,1);
   yt = zeros(leng,1);


   %Calculating a
   a = Calculating_a(p, leng,x );

   aq = quantizer(a, 8, -2, 2);

   yn(1:p) = x (1:p);
   yh(1:p) = quantizer(yn(1:p), num, -3, 3);
   yht(1:p) = yt(1:p) + yh(1:p);


   for i = p+1:leng
   yt(i) = sum(aq(1:p).*yht(i-1:-1:i-p));
   yn(i) = x(i) - yt(i);
   yh(i) = quantizer(yn(i), num, -3, 3);
   yht(i) = yt(i) + yh(i);
   %disp('Predicted!')
   end

   data(:,num) = yht;

   disp('The value of N= ')
   num
   disp('The value of p=')
   p

   E_y_2(num,p-3) = mean((x - yt).^2);
```

```
  end
end
%E_y_2
figure
subplot(1,1,1)
i=[1:10000];
%plot(i,x, 'g.-', i,data(:,1), 'b.--',i,data(:,2),i,data(:,3), 'k.--' );
%legend('the first signal: x', 'prediction:yht 1','prediction:yht 2','prediction:yht 3');
%xlabel('i');
%ylabel('j');
%title('p = 8 & N = 1,2,3');

plot(i,x, 'g.-', i,yh, 'b.--');
legend('the first signal: x', 'anakataskeuasmeno:yh 1');
xlabel('i');
ylabel('j');
title('X and Yh');

%N = [2 4 6];
%figure(3);


end




%-------------------------------------------------------------------------------------


function [a ] = Calculating_a (p, leng,x )
r = zeros(p, 1);
R = zeros(p, p);
for i = 1:p
  r(i) = 1/(leng - p) * sum(x(p+1:leng).*x(p+1-i:leng-i));
end

%Calculating R
for i = 1:p
  for j = 1:p
    R(i, j) =  1/(leng - p + 1) * sum(x(p+1-j:leng-j).*x(p+1-i:leng-i));
  end
end
a = R\r;
end



%-------------------------------------------------------------------------------------
```

Part B



Σχήμα 3. Ομόδυνο M-PSK

1. Implementation of the M-PSK system

We enter the input sequence in the M-PSK, that is the bits containing the information we want to send. This sequence is generated by the command:

randsrc (1, 10000, [0 1; 0.5 0.5])

that is, a vector of 10000 elements is created with 0 and 1 which are equidistant.

The mapper converts the bits into symbols.
When configuring, we create a register where its lines are the symbols and columns of the 40 samples we get for each symbol based on the type given in the speech.

In order to include noise, we use the type of speech

SNR = 10 * log10 (Eb / N0) = 10

to calculate NO, with the SNR being given at the input and then creating the vector of noise, which we convert it with the reshape command into a register of the same dimensions as the register sent to the receiver to add registers.

In demodulation, we create a register where its lines are the symbols and each of its two columns is the components of the signal and contains the sum of the samples multiplied by the rectangular pulse and the carrier.

The detector takes the components of each symbol received and, after calculating its distance from each source symbol, selects the distance that is the shortest distance. The Euclidean distance is used as the metric

$$root\ [(x2 - x1)\ \hat{}\ 2 + (y2 - y1)\ \hat{}\ 2]$$

, where (x1, y1) and (x2, y2) the coordinates of the points.

Finally, **the demapper** performs the inverse process from the mapper, ie the 8-PSK simply assigns each symbol value to 3-bits, while for the 4-PSK assigns each symbol to a block of 2 bits, taking into account the encoding option gray .

The Bit Error Rate is calculated by counting the number of different elements between the input and output sequences and dividing it by the number of input sequence elements.

```
function [ ] = main_psk ()
BER_0=zeros(8,1);
BER_1=zeros(8,1);

   Lin=randsrc(1, 10000, [0 1; 0.5 0.5]);
          M=8;

        k=0;
        l=1;
        for i=0:2:16
                SNR=i;
        Gray=k;
        [BER_0(l), Pb0(l)]=my_psk(Lin, M, SNR, Gray);
        l = l+1;
     end
 %------------------------------------------
        k=1;
     l=1;
        for i=0:2:16
        SNR=i;
        Gray=k;
        [BER_1(l), Pb1(l)]=my_psk(Lin, M, SNR, Gray);
        l = l+1;
     end
Pb1
Pb0
BER_0
BER_1

figure
subplot(1,1,1)
i=[1:9];
plot(i,BER_0, 'k.-', i,BER_1, 'b.--',i,Pb1, 'y.--',i,Pb0, 'g.--');
legend('Without Gray:BER0', 'With gray: BER1', 'Pb0', 'Pb1');
xlabel('i');
ylabel('j');
title('With and Without Gray the BER signal is : ');

end

%-------------------------------------------------------------------------------------

function [BER, Pb] = my_psk (Lin, M, SNR, Gray)
% M - PSK
%       Lin: dianusma symvolwn pou tha xrhsimopoiithei gia metadosh
%       M: to plithos twn symvolwn tou alfavhtou
%       SNR: o logos isxuos pros thoryvo ana bit
%       Gray: 1 gia kwdikopoihsh Gray, alliws 0
%       BER: to bit error rate
%   Pb: thewrhtikh pithanothta sfalmatos
```

```
    if M ~= 8 && M ~= 4
        fprintf('Wrong input\n')
        return
    end

    % MAPPER
    Lb = length(Lin); % Mhkos ths arxikhs akolouthias
    out = zeros(Lb / log2(M), 1); % Arxikopoihsh tou pinaka pou tha periexei ta sumvola pros metadosh
    if M == 8 % Se auth thn periptwsh kathe stoixeio ths akolouthias eisodou antistoixei se sumvolo
        out = Lin;
    elseif M == 4 % Se auth thn periptwsh 2 stoixeia ths akolouthias eisodou antistoixoun se sumvolo
        for a = 0 : length(out) - 1
            out(a+1) = bin2dec(num2str(Lin(2*a+1 : 2*a+2))); % Sairnoume 2 sunexomena stoixeia kai ta metatrepoume se sumvolo me thn bin2dec
        end
        if Gray == 1
            out = gray2bin(out, 'psk', M); % Metatroph ths akolouthias se gray kwdikopoihsh an zhteitai
        end
    end


    % Modulation PSK
    Es = 1;
    Tsymbol = 40;
    Tc = 4;
    Tsample = Tc / 4;
    fc = 1 / Tc;
    g = sqrt(2 * Es / Tsymbol); % Orthogwnios palmos
    samples = Tsymbol / Tsample; % Ypologismos tou arithmou twn deigmatwn
    s = zeros(length(out), samples); % Arxikopoihsh tou pinaka pou tha periexei ta deigmata gia kathe sumvolo
    for i = 1 : length(out)
        m = out(i);
        for j = 1 : samples
            s(i, j) = cos(2 * pi * m / M) * g * cos(2 * pi * fc * j) + sin(2 * pi * m / M) * g * sin(2 * pi * fc * j);
        end
    end


    % AWGN
    No = Es / (log2(M) * power(10, SNR / 10)); % Sunarthsh No(SNR), opou to SNR(ana bit) dinetai sthn eisodo
    s2 = No / 2;
    noise = sqrt(s2) * randn(length(out) * 40, 1); % Paragwgh twn deigmatwn thoruvou
    noise = reshape(noise, length(out), samples); % Metatroph tou dianusmatos se disdiastato pinaka gia kateutheian prosthesh me to arxiko mhtrwo
    r = s + noise; % Paragwgh tou mhtrwou pou tha stalei sto dekth
```

```
% Demodulation PSK
siz = size(r);
r_  = zeros(siz(1), 2); % To siz(1) dinei ton arithmo twn grammwn tou dianusmatos r, dld to plithos twn sumvolwn. Gia kathe sumvolo exoume 2 sunistwses
for i = 1 : siz(1) % Kathe sumvolo pollaplasiazetai me th ferousa kai ton orthogwnio palmo ki ustera athroizontai ta deigmata
    for j = 1 : samples
        r_(i, 1) = r_(i, 1) + r(i, j) * g * cos(2 * pi * fc * j);
        r_(i, 2) = r_(i, 2) + r(i, j) * g * sin(2 * pi * fc * j);
    end
end


% Fwraths
s1 = zeros(M, 1); % Oi sunistwses(suntagmenes) kathe sumvolou tou alfavhtou
s2 = zeros(M, 1);
for m = 1 : M
    s1(m) = sqrt(Es) * cos(2 * pi * (m - 1) / M);
    s2(m) = sqrt(Es) * sin(2 * pi * (m - 1) / M);
end
dist = zeros(M, 1); % Arxikopoihsh pinaka pou tha periexei gia kathe sumvolo thn apostash tou apo
symb = zeros(1, siz(1)); % To siz(1) dinei ton arithmo twn grammwn tou dianusmatos r_
for i = 1 : siz(1) % Gia kathe symvolo eisodou
    for j = 1 : M %Ypologismos ths apostashs tou apo to kathe symvolo tou alfavhtou
        dist(j) = sqrt((r_(i, 1) - s1(j))^2 + (r_(i, 2) - s2(j))^2); %Xrhsimopoieitai h eukleidia apostash
    end
    [a, b] = min(dist); % Euresh tou elaxistou, dhladh tou symvolou pou apexei ligotero apo thn eisodo tou dekth
    symb(i) = b - 1; % Apothikeush tou stoixeiou me thn elaxisth apostash meiwmenou kata 1 gt to m pairnei times [0, M-1]
end


% Demapper
if Gray == 1
    symb = bin2gray(symb, 'psk', M); % Metatroph apo gray se duadiko se periptwsh pou exei ginei kwdikopoihsh Gray
end
Lout = zeros(1, length(symb) * log2(M)); % Arxikopoihsh pinaka pou tha periexei ta bits ths ektimwmenhs akolouthias
if M == 8 % Se auth thn periptwsh kathe sumvolo antistoixizetai se stoixeio ths akolouthias eksodou
    Lout = symb;
elseif M == 4 % Se auth thn periptwsh ena sumvolo antistoixizetai se 2 stoixeia ths akolouthias eksodou
    for i = 1 : length(symb)
        if symb(i) == 0
            Lout(2*(i-1)+1) = 0;
            Lout(2*i) = 0;
        elseif symb(i) == 1
            Lout(2*(i-1)+1) = 0;
```

```
            Lout(2*i) = 1;
        elseif symb(i) == 2
            Lout(2*(i-1)+1) = 1;
            Lout(2*i) = 0;
        elseif symb(i) == 3
            Lout(2*(i-1)+1) = 1;
            Lout(2*i) = 1;
        end
    end
  end

  v = Lin(Lin ~= Lout); % To dianusma v periexei osa stoixeia einai diaforetika metaksu twn Lin,Lout
  BER = length(v)/Lb; % Diairesh tou plithous twn diaforetikwn stoixeiwn me to sunoliko arithmo stoixeiwn wste na prokupsei to BER
  Eb = Es/ log2(M);
  Pb = 1/2*erfc(sqrt(Eb/No)); % Ypologismos ths pithanothtas sfalmatos

end
```

2. Error probability and BER curve design

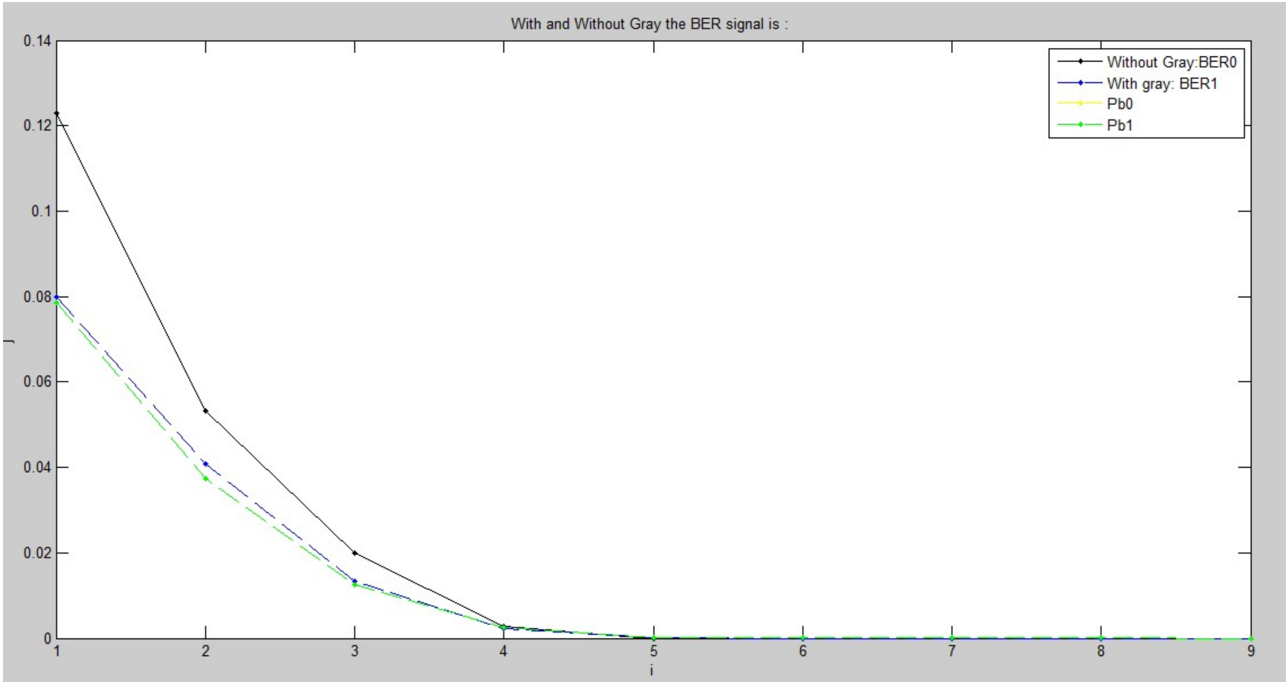
The theoretical calculation of the probability of error is given by the formula:

$$P_b = Q\left(\sqrt{\frac{2*E_b}{N_0}}\right) = \frac{1}{2}\,\mathrm{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right).$$

In PSK, the increase in M does not significantly affect the Bit Error Rate for M under 8, where this value increases sharply.

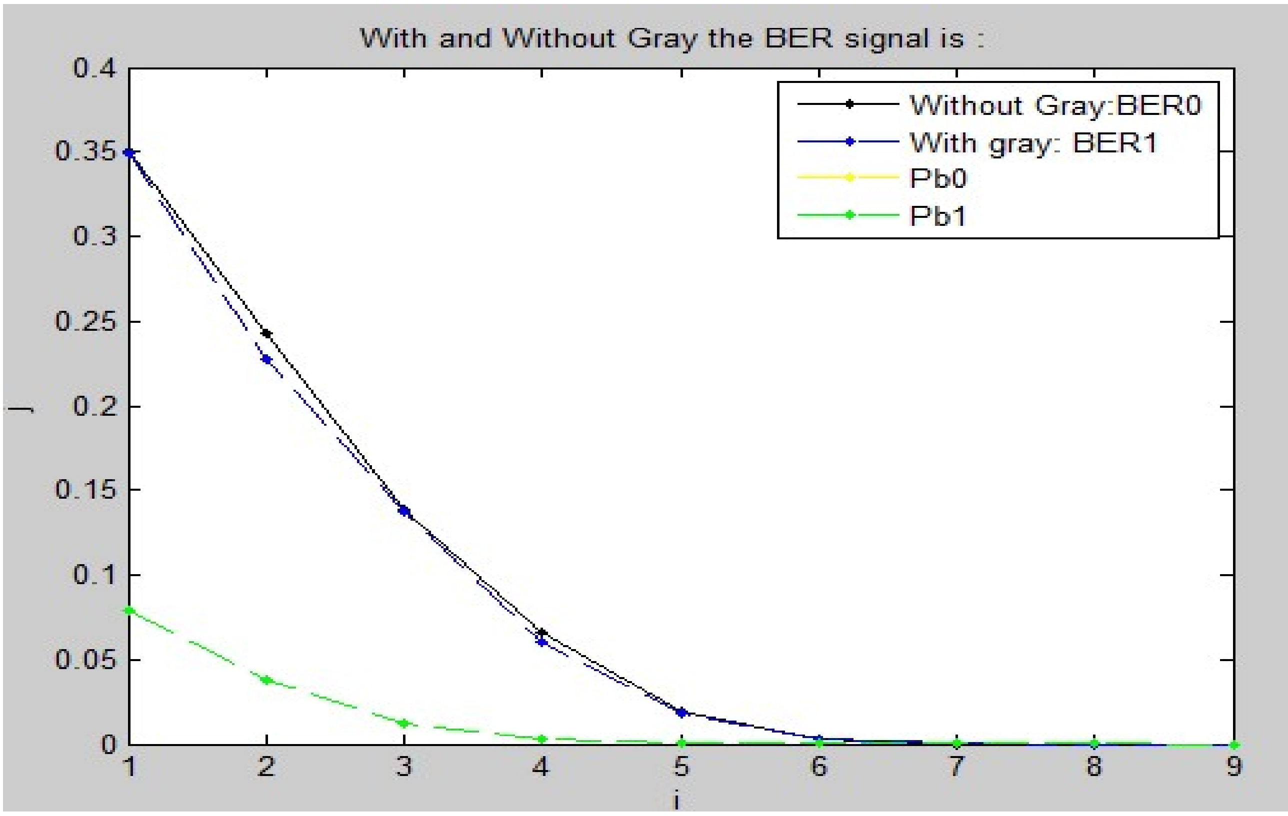
The experimental BER without Gray coding is slightly above the experimental Gray encoding BER, which is logical since we expected more errors in Gray's binary encoding.

M=4



With and Without Gray the BER signal is :
Without Gray:BER0
With gray: BER1
Pb0
Pb1
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1
2
3
4
5
6
7
8
9
i

M=8



With and Without Gray the BER signal is :
Without Gray:BER0
With gray: BER1
Pb0
Pb1
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
9
i