Ψηφιακές Τηλεπικοινωνίες

Ακαδημαϊκό Έτος 2017/2018

Τουρλίδα Βάγια ΑΜ: 62233 (1041905)

Περιεχόμενα

1 Εισαγωγή

Στα πλαίσια της συγκεκριμένης εργασίας πραγματοποιείται η υλοποίηση και ανάλυση δύο βασικών συστημάτων των ψηφιακών τηλεπικοινωνιών. Συγκεκριμένα, τα συστήματα που περιγράφονται παρακάτω είναι τα εξής:

1. Σύστημα Συμπίεσης Διακριτής Πηγής με Χρήση της Κωδικοποίησης DPCM
2. Σύστημα Προσομοίωσης Ομόδυνου Ζωνοπερατού Συστήματος M-PAM

Στη συνέχεια παρουσιάζονται αναλυτικά :

1. οι αλγόριθμοι υλοποίησης των παραπάνω συστημάτων, καθώς
2. οι αντίστοιχες μετρήσεις/μετρικές για την αξιολόγηση της απόδοσής τους, υπό συγκεκριμένες συνθήκες εκτέλεσης.

2 Περιβάλλον Εκτέλεσης

Τα χαρακτηριστικά του περιβάλλοντος που χρησιμοποιήθηκε για τον υλοποίηση και την αξιολόγηση των διαφόρων συστημάτων περιγράφονται παρακάτω.

Λειτουργικό Σύστημα	**Γλώσσα Προγραμματισμού**
Windows 10 64-bit	MatlabR2017a

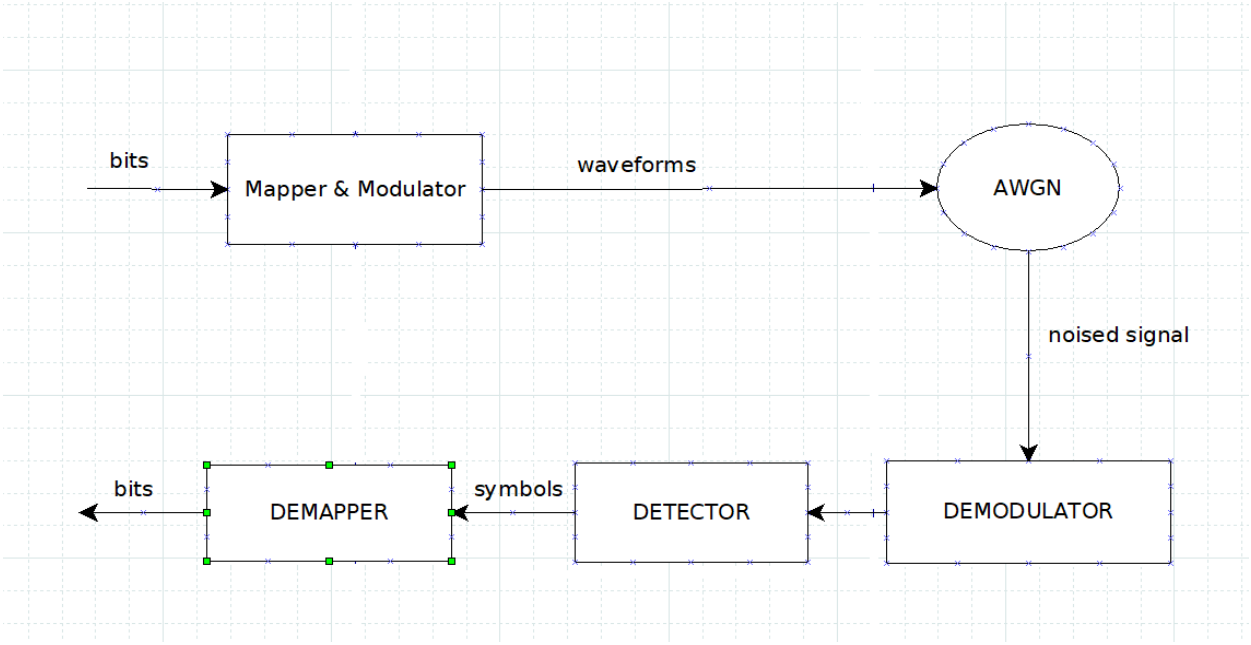
4.1 ΠΕΡΙΓΡΑΦΗ

Το **PAM** είναι μια μορφή διαμόρφωσης σήματος όπου η πληροφορία του μηνύματος κωδικοποιείται στο εύρος μιας σειράς παλμών σήματος. Πρόκειται για ένα πρόγραμμα αναλογικής διαμόρφωσης παλμών στο οποίο τα πλάτη μεταβάλλονται σύμφωνα με την τιμή δείγματος του σήματος μηνύματος. Η αποδιαμόρφωση γίνεται με ανίχνευση του επιπέδου πλάτους του φορέα σε κάθε μία περίοδο.

΄Οπως κάθε τηλεπικοινωνιακό σύστημα, έτσι και το PAMαποτελείται από τρία βασικά μέρη:

- **Δέκτη**: Αποτελείται από τον mapper και τον modulator.
- **Κανάλι**: Αποτελείται από ένα Additive white Gaussian noise (AWGN) κανάλι.
- **Πομπό**: Αποτελείται από τον demapper, τον decoder και τον demodulator.

Επομένως για την υλοποίηση της προσομοίωσης ενός συστήματος PAM απαιτήθηκε η υλοποίηση του καθενός υποσυστήματος ξεχωριστά, και στη συνέχεια η σύνδεση αυτών μεταξύ τους, όπως παρουσιάζεται στο παρακάτω σχήμα:



3.1.1 Mapper

- Αρχικά υλοποιήθηκε το υποσύστημα του Mapper, ο οποίος αναλαμβάνει την μετατροπή της δυαδικής ακολουθίας του σήματος σε M σύμβολα. Ο Mapper υλοποιήθηκε με τέτοιο τρόπο ώστε να υποστηρίζει δύο τρόπους κωδικοποίησης, την απλή και την Gray.Για να μπορέσουμε να πραγματοποιήσουμε το Mapping για την κωδικοποιήση κατά Gray για το 8-PAM μετατρέψουμε την αρχική ακολουθία που δημιουργούμε στην αντίστοιχη ακολουθία Gray υλοποιήσαμε την συνάρτηση **gray_encoder**.Αντίστοιχα για την αποκωδικοποίηση της ακολουθίας κάτα Gray και την αντιστοίχιση της σε δυαδική ακολουθία υλοποιήθηκε η συνάρτηση **decode_gray**.
- Επίσης, ανάλογα του μεγέθους του σήματος εισόδου υπάρχει πιθανότητα να υπάρξουν κάποια bits τα οποία να μην αρκούν ώστε να σχηματίσουν ομάδα των log2(M) bits. Για τον λόγο αυτό, κατά την κωδικοποίηση του σήματος, προστίθεται ο απαραίτητος αριθμός από bits μηδενικών στα MSB bits του σήματος, ώστε όλα τα bits να δημιουργούν αντίστοιχα ομάδα των log2(M) bits. Με αυτόν τον τρόπο όλη η δυαδική πληροφορία του σήματος κωδικοποιείται σωστά, δίχως να υπάρξει πιθανώς κάποια απώλεια των τελευταίων δειγμάτων.

Modulator

- Στη συνέχεια υλοποιήθηκε το υποσύστημα του διαμορφωτή, ο οποίος αναλαμβάνει να πολλαπλασιάσει το κωδικοποιημένο σήμα με τον ορθογώνιο παλμό και στη συνέχεια να μεταφέρει το σήμα στη ζώνη μετάδοσης, ώστε τελικά να προκύψει το ζωνοπερατό σήμα Sm.

Τα δυο υποσυστήματα υλοποιήθηκαν στην ίδια συνάρτηση **my_mapper**.

3.1.2 AWGN Κανάλι

Επιπλέον υλοποιήθηκε το υποσύστημα του AWGN καναλιού, μέσω του οποίου διέρχεται το σήμα. Συγκεκριμένα θεωρήθηκε ότι το σήμα διέρχεται μέσω ενός ιδανικού καναλιού. Επομένως εφόσον παράχθηκε ο τυχαίος θόρυβος, με βάση την αντίστοιχη τιμή του SNR, τότε προστέθηκε στις αντίστοιχες τιμές του διαμορφωμένου σήματος.

Τα δυο υποσυστήματα υλοποιήθηκαν στην ίδια συνάρτηση **add_awgn**.

Στην αρχική μας προσπάθεια να παράξουμε τυχαίο θόρυβο αντιμετώπισαμε αρκέτα προβλήματα το κυριότερο ήταν το γεγονός ότι η συνάρτηση randn γενά τιμές μεταξυ 0-1 κατά συνέπεια να προστίθεται κάποιος θόρυβος στο σήμα μας ο οποίος ναι μεν υπήρχε αλλά δεν ήταν επάρκης ώστε να προκαλέσει κάποια παραμόρφωση στο σήμα μας λόγω του ότι τα πλάτοι μας πολλαπλασιάζονται με την ενέργεια A η οποία έχει αρκετά μεγάλες τιμές.Για την αντιμετώπιση αυτού του προβλήματος βασιστήκαμε στο πλατός του Sm μας και προσθέσαμε στο θόρυβο μία μεταβλήτη αρκέτα μεγάλη ώστε η παραμόρφωση που θα υποστεί το σήμα να μπορεί να γίνει αντιληπτή.

3.1.3 Demodulator

Μετά υλοποιήθηκε το υποσύστημα του αποδιαμορφωτή, μέσω του οποίου το σήμα αποδιαμορφώνεται σε ένα διάνυσμα r. Το συγκεκριμένο διάνυσμα αντιστοιχεί στη θέση του ληφθέντος σήματος πάνω στον άξονα του M-PAM. Μέσω του συγκεκριμένου επιπέδου θα επιτευχθεί ο προσδιορισμός του αντίστοιχου συμβόλου της κάθε ομάδας των log2(M) bits.

3.1.4 Detector
Στη συνέχεια υλοποιήθηκε το υποσύστημα του φωρατή, ο οποίος αξιοποιώντας το επίπεδο του M-PAM και την αντίστοιχη θέση του διανύσματος r πάνω σε αυτό, αποφασίζει σε ποιο σύμβολο βρίσκεται εγγύτερα. Για να επιτευχθεί αυτό εξετάζονται όλες οι αποστάσεις μεταξύ των διανυσμάτων r και Sm, χρησιμοποιώντας την μετρική της ευκλείδιας απόστασης, και εφόσον βρεθεί η μικρότερη απόσταση, τότε το αντίστοιχο διάνυσμα Sm αντιστοιχεί στο σύμβολο που στάλθηκε.

3.1.5 Demapper
Τελευταίο υλοποιήθηκε το υποσύστημα του Demapper, ο οποίος δέχεται ως είσοδο το σύμβολο που έχει ανιχνεύσει ο φωρατής και στη συνέχεια παράγει την αντίστοιχη ομάδα από log2(M) bits. Επιπλέον ανάλογα με την αρχική κωδικοποίηση που έχει υποστεί το σήμα, επιλέγεται και ο αντίστοιχα κατάλληλος τρόπος αποκωδικοποίησής του.

3.1.6 Main
Για την ένωση όλων των υποσυστημάτων που υλοποιήθηκαν δημιουργήθηκε ένα τελευταίο κομμάτι κώδικα, που συμβάλει στην πλήρη και ομαλή λειτουργία της προσομίωσης του συστήματος M-PAM.
Συγκεκριμένα μέσω αυτού δημιουργείται αρχικά ένα τυχαίο δυαδικό σήμα εισόδου, με ισοπίθανες εμφανίσεις των 0 και 1. Στη συνέχεια αναλαμβάνει να χρησιμοποιήσει κατάλληλα τα αντίστοιχα υλοποιημένα υποσυστήματα, ώστε τελικά να προκύψει το αντίστοιχο δυαδικό σήμα εξόδου. Τέλος, αναλαμβάνει να αφαιρέσει τυχόν επιπλέον bits που προστέθηκαν μέσω του Mapper, τα οποία δεν προσφέρουν πια καμία πληροφορία στο σήμα.

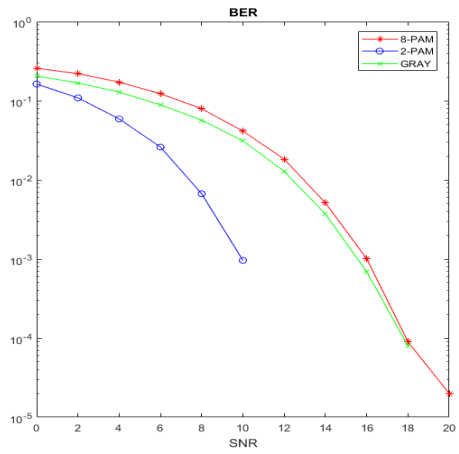
3.1.6 Μετρήσεις

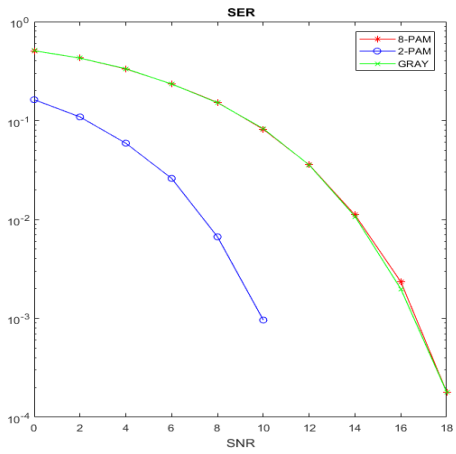
ΕΡΩΤΗΜΑ Β

Υπολογίσμος BER για M=2,M=8 με απλή κωδικοποίηση && κωδικοποίηση κατά Gray

Για να υπολογίσουμε το BER χρειάστηκε να συγκρίνουμε την ακολουθία των bit που παράχθηκαν ύστερα από την εκτέλεση της συναρτησης **my_demmaper** με ακολουθία εισόδου.Όμως επειδή το M παίζει ρόλο στον τρόπο με τον οποίο ομαδοποιούμε αρχικά τα bits της ακολουθίας εισόδου τα οποία τα ομαδοποιούμε ανά log2(M) bits γίνεται αντιληπτό ότι κατά το demmaping πρέπει να λάβουμε υπόψιν μας αυτή την αντιστοιχία ώστε να μπορέσουμε να κάνουμε σωστή αντιστοίχηση των δεκαδικών συμβόλων που προκύπτουν για τα διαφορετικά M και τις διαφορετικές κωδικοποιήσεις.Για τον λόγο αυτό υπολόγισαμε τα BER ξεχωριστά για κάθε διαφορετικο M και για κάθε διαφορετικη κωδικοποίηση.Η αντιστοιχία κάθε M και κάθε κωδικοποίησης με την αντίστοιχη μεταβλητή BER που χρησιμοποιήθηκε παρατίθεται παρακάτω :

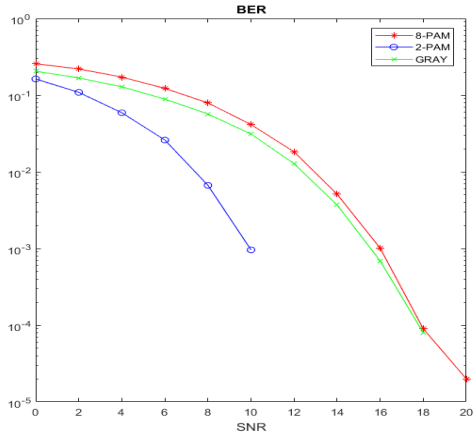
Μεταβλητές BER	M	Κωδικοποίηση
num_err_2PAM	2	απλή
num_err_8PAM	8	απλή
num_err_8PAM_Gray	8	Gray

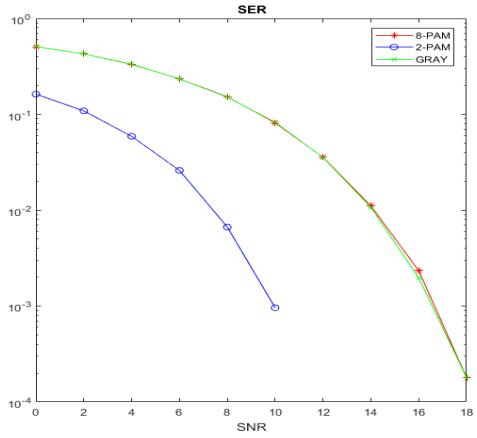




ΕΡΩΤΗΜΑ Γ

Υπολογίσμος SER για M=2,M=8 με απλή κωδικοποίηση && κωδικοποίηση κατά Gray





Για να υπολογίσουμε το SER χρειάστηκε να συγκρίνουμε την ακολουθία των συμβόλων που παράχθηκαν ύστερα από την εκτέλεση της συναρτησης **my_detector** με ακολουθία εισόδου.Όμως επειδή το M παίζει ρόλο στον τρόπο με τον οποίο ομαδοποιούμε αρχικά τα bits της ακολουθίας εισόδου τα οποία τα ομαδοποιούμε ανά log2(M) bits γίνεται αντιληπτό ότι κατά την αντιστοίχιση πρέπει ξανά να λάβουμε υπόψιν μας αυτή την αντιστοιχία ώστε να μπορέσουμε να κάνουμε σωστή αντιστοίχηση των δεκαδικών συμβόλων που προκύπτουν για τα διαφορετικά M και τις διαφορετικές κωδικοποιήσεις.Για τον λόγο αυτό υπολόγισαμε τα SER ξεχωριστά για κάθε διαφορετικο M και για κάθε διαφορετικη κωδικοποίηση.

Η αντιστοιχία κάθε M και κάθε κωδικοποίησης με την αντίστοιχη μεταβλητή SER που χρησιμοποιήθηκε παρατίθεται παρακάτω :

Μεταβλητές	**M**	**Κωδικοποίηση**
`signal_error_2PAM`	2	απλή
`signal_error_8PAM`	8	απλή
`signal_error_Gray`	8	Gray

Παρατηρωντας το διάγραμα του SER βλέπουμε ότι όταν αλλάζουμε το M από 2 σε 8 χρειαζόμαστε μεγαλύτερη ισχύ δηλαδή μεγαλύτερο SNR μέχρι να καταφέρουμε να πετύχουμε το ίδιο SER για την ακρίβεια κάθε φορά που διπλασιάζεται το M το SNR/bit θα πρέπει να αυξηθεί περισσότερο από 4dB για το ίδιο SER.

- **Πιθανότητα σφάλματος συμβόλου** (SER):

$$P_M = \frac{2(M-1)}{M} Q\left(\sqrt{\frac{2E_g}{N_0}}\right) = \frac{2(M-1)}{M} Q\left(\sqrt{\frac{6E_s}{(M^2-1)N_0}}\right)$$

Αυτό μπορούμε να το επαληθεύσουμε και αν στον παραπάνω τύπο αντικαταστήσουμε M=2 & M=8.

Επίσης παρατηρώντας και τα δυο διαγράμματα γίνεται αντιληπτό ότι η εκτέλεση του 8-PAM με απλή κωδικοποίηση είναι χειρότερη από την εκτέλεση του 8-PAM με κωδικοποίηση κατά Gray.Μέσα από τα διαγράμματα βλέπουμε ότι ενώ ενώ στην κωδικοποιήση κατά gray το SER είναι μεγαλύτερο από το SER στην απλή κωδικοποίηση στην κωδικοποίηση κατά gray έχουμε μικρότερο BER από την απλή κωδικοποίηση πράγμα το οποίο σημαίνει ότι με την κωδικοποίηση κατά gray ο δέκτης λαμβάνει λιγότερη λανθασμένη πληροφορία.

Κώδικες για το Μέρος Α:

my_modulator.m

```
%Υλοποιήση του διαμορφωτή
%input_symbols <-Συμβολική ακολουθία σήματος
%M <- Το πλήθος των συμβόλων
function [s_m] = my_modulator(input_symbols, M,Am)


%σε κάθε περίοδο φέρουσας κρατάμε 4 δείγματα και
%κάθε περίδο συμβόλου περιλαμβάνει 40 δείγματα
t_sumbol = 40;
t_c = 4;
f_c = 1/t_c;
%Ορθογώνιος παλμός
g_t = sqrt(2/t_sumbol);

%Υπολογισμός συνιστώσας για κάθε σύμβολο
for m = 1:M
    s(m+1,1) =Am(m)*cos(2*pi*m/M);
end

%Υπολογισμός του ζωνοπερατού σήματος
for i = 1:length(input_symbols)
    for t = 1:t_sumbol
        s_m(i,t) = s((input_symbols(i)+1),1)*g_t*cos(2*pi*f_c*t);
    end
end
end
```

generate_Am.m

```
function Am = generate_Am(M,energy_A )
%Δημιουργία πλατών
for i =1:1:M
   Am(i)=(2*i-(M+1))*energy_A;
end
end
```

calculate_energy.m

```
function energy= calculate_energy(M)
%=Υπολογισμός ενέργειας
Aaverage=1;
energy=(Aaverage*3)/(M^2-1);
end
```

gray_encoder.m

```
function gray_vector=gray_encoder(input_binary, n_bits)
%γεννά και επιστρέψειένα διάνυσμα gray των N bits
n=length(input_binary);
if mod(n, n_bits)~=0
    error('Vector size doesnt match word bits')
end
for i=1:n_bits:n-n_bits+1
    gray_vector(i)=input_binary(i);
    for j=1:n_bits-1
        gray_vector(i+j)=xor(input_binary(i+j),input_binary(i+j-1));
    end
end
end
```

main.m

```
clear;
%Προσωμοίωση Ομόδυννου ζωνοπερατού M-PAM
%input_size <- μέγεθος διανύσματος εισόδου
input_size= 10^5;
Rsymbol=25*power(10,4);
Tsymbol=1/Rsymbol;
fc=2.5*power(10,6);
Tc=1/fc;
Tsample=Tc/4;
fsample=1/Tsample;
samples=Tsymbol/Tsample;
Esymbol=1;

%Αλφάβητο εισόδου
alphabet = [0 1];
%Δημιουργία ακολουθίας με ισοπίθανη εμφάνιση συμβόλων αλφαβήτου
input_binary = randsrc(input_size , 1, alphabet);
%Υπολογισμός του μήκους του διανύσματος της πηγής
len=length(input_binary);
%Ελέγχουμε αν το διάνυσμα του σήματος μπορεί να κωδικοποιήθει ακριβώς με k=3
modulo=mod(len,log2(8));
 if(modulo~=0)
     temp=input_binary(len-modulo+1:len);
    input_binary(len-modulo+1:len)=0;
    input_binary(len:len+log2(8)-modulo)=temp;
 end
%κωδικοποίηση κατά gray
gray_vector = gray_encoder(input_binary,log2(8));
%Επανυπολογισμός μεγέθους εισόδου
input_size=length(input_binary);
%Διάνυσμα χρόνου
t=(0:Tsample:Tsymbol);
%Υπολογισμός πλατών για 2-PAM && 8-PAM
Am2=generate_Am(2,calculate_energy(2));
Am8=generate_Am(8,calculate_energy(8));
```

```
%Δημιουργία διανύσματος SNR
SNR=(0:2:20);

% Preallocation πινάκων για μείωση του χρόνου προσωμοίωσης
Sm_2PAM(1:input_size,1:length(t))=zeros;
Sm_8PAM(1:floor(input_size/3),1:length(t))=zeros;
SmAm2(1:2,1:length(t))=zeros;
SmAm8(1:8,1:length(t))=zeros;
symbol8(1:floor(input_size/3))=zeros;
Sm_gray(1:floor(input_size/3),1:length(t))=zeros;
symbol_gray(1:floor(input_size/3))=zeros;
bit_error8_ave(1:length(SNR))=zeros;
bit_error2_ave(1:length(SNR))=zeros;
bit_error_gray_ave(1:length(SNR))=zeros;
symbol_error8_ave(1:length(SNR))=zeros;
symbol_error_gray_ave(1:length(SNR))=zeros;


% Υπολογισμός συμβόλων κυματομορφων για μείωση του χρόνου προσωμοίωσης
 for i=1:2
    SmAm2(i,:)=my_mapper(Am2(i),t,Tsymbol,fc);
end
for i=1:8
    SmAm8(i,:)=my_mapper(Am8(i),t,Tsymbol,fc);
end

for snr=1:length(SNR)
   bits_error_8PAM = 0;
   bits_error_2PAM = 0;
   SER_2PAM=0;
   SER_8PAM=0;
   SER_Gray=0;
   bits_error_gray=0;
   bit_error_all=0;

 %Υποσύστημα Modulator & mapping
      for j=1:input_size

      %2-PAM modulation
      if input_binary(j)==0
         Sm_2PAM(j,:)=SmAm2(1,:);
      else
         Sm_2PAM(j,:)=SmAm2(2,:);

      end
      if((mod(j+2,3)==0)&&(j<(input_size-2)))
      %8-PAM modulation
       if (input_binary(j)==0 && input_binary(j+1)==0 && input_binary(j+2)==0)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(1,:);
        elseif (input_binary(j)==0 && input_binary(j+1)==0 &&
input_binary(j+2)==1)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(2,:);
        elseif (input_binary(j)==0 && input_binary(j+1)==1 &&
input_binary(j+2)==0)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(3,:);
        elseif (input_binary(j)==0 && input_binary(j+1)==1 &&
input_binary(j+2)==1)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(4,:);
        elseif (input_binary(j)==1 && input_binary(j+1)==0 &&
input_binary(j+2)==0)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(5,:);
```

```
 elseif (input_binary(j)==1 && input_binary(j+1)==0 && input_binary(j+2)==1)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(6,:);
        elseif (input_binary(j)==1 && input_binary(j+1)==1 &&
input_binary(j+2)==0)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(7,:);
        elseif (input_binary(j)==1 && input_binary(j+1)==1 &&
input_binary(j+2)==1)
            Sm_8PAM(floor(j/3)+1,:)=SmAm8(8,:);
       end
        %8-PAM modulation με κωδικοποιληση κατά gray
       if (gray_vector(j)==0 && gray_vector(j+1)==0 && gray_vector(j+2)==0)
            Sm_gray(floor(j/3)+1,:)=SmAm8(1,:);
        elseif (gray_vector(j)==0 && gray_vector(j+1)==0 && gray_vector(j+2)==1)
            Sm_gray(floor(j/3)+1,:)=SmAm8(2,:);
        elseif (gray_vector(j)==0 && gray_vector(j+1)==1 && gray_vector(j+2)==0)
            Sm_gray(floor(j/3)+1,:)=SmAm8(3,:);
        elseif (gray_vector(j)==0 && gray_vector(j+1)==1 && gray_vector(j+2)==1)
            Sm_gray(floor(j/3)+1,:)=SmAm8(4,:);
        elseif (gray_vector(j)==1 && gray_vector(j+1)==0 && gray_vector(j+2)==0)
           Sm_gray(floor(j/3)+1,:)=SmAm8(5,:);
        elseif (gray_vector(j)==1 && gray_vector(j+1)==0 && gray_vector(j+2)==1)
            Sm_gray(floor(j/3)+1,:)=SmAm8(6,:);
        elseif (gray_vector(j)==1 && gray_vector(j+1)==1 && gray_vector(j+2)==0)
            Sm_gray(floor(j/3)+1,:)=SmAm8(7,:);
        elseif (gray_vector(j)==1 && gray_vector(j+1)==1 && gray_vector(j+2)==1)
            Sm_gray(floor(j/3)+1,:)=SmAm8(8,:);
       end
      end
      end


       size_Sm_8PAM=size(Sm_8PAM);
       size_Sm_2PAM=size(Sm_2PAM);
       size_Sm_Gray_8PAM=size(Sm_gray);


 %Υπολογισμός του πλάτους της κυματομορφής για τον υπολογισμό του θορύβου.
calculation
         m2=4*max(Sm_2PAM(:));
         m8=max(Sm_8PAM(:));

       %Υπολογισμός θορύβου
       AWGN_8PAM= add_awgn(SNR(snr),size_Sm_8PAM,m8);
       AWGN_2PAM= add_awgn(SNR(snr),size_Sm_2PAM,m2);
       AWGN_8PAM_Gray= add_awgn(SNR(snr),size_Sm_Gray_8PAM,m8);

       %Υπολογισμός ληφθέντος σήματος
       %r(t)=Sm(t)+N(t)
       AWGN_signal_8PAM=Sm_8PAM+ AWGN_8PAM;
       AWGN_signal_2PAM=Sm_2PAM+ AWGN_2PAM;
       AWGN_signal_8PAM_Gray=Sm_gray+ AWGN_8PAM;

      %Υποσύστημα Demodulator
        received_signal_8PAM = my_demodulator(AWGN_signal_8PAM,t,Tsymbol,fc);
        received_signal_2PAM = my_demodulator(AWGN_signal_2PAM,t,Tsymbol,fc);
received_signal_8PAM_Gray=my_demodulator(AWGN_signal_8PAM_Gray,t,Tsymbol,fc);

     %Εξαγωγή συμβόλων από τα διανύσματα received
     received_symbols_8PAM = my_detector(received_signal_8PAM, Am8);
     received_symbols_2PAM = my_detector(received_signal_2PAM, Am2);
     received_symbols_8PAM_Gray = my_detector(received_signal_8PAM_Gray, Am8);
```

```
 %Δημιουργία δυαδικού διανύσματος από τα ληφθέντα σύμβολα
    output_vector_8PAM=my_demapper(received_symbols_8PAM, 8); %M=8
    output_vector_2PAM=my_demapper(received_symbols_2PAM, 2); %M=2
    output_vector_gray_und=my_demapper(received_symbols_8PAM_Gray, 8); %M=8 gray
   %Αποκωδικοποιήση gray ακολουθίας
    output_vector_gray=decode_gray(output_vector_gray_und,3);
         for i = 1:length(input_binary)
           if (input_binary(i) ~= output_vector_2PAM(i))
              bits_error_2PAM = bits_error_2PAM + 1;
           end
           if (input_binary(i) ~= output_vector_8PAM(i))
              bits_error_8PAM = bits_error_8PAM + 1;
           end
           if (input_binary(i) ~= output_vector_gray(i))
              bits_error_gray = bits_error_gray + 1;
           end
         end
        num_err_2PAM(snr)=bits_error_2PAM/length(input_binary);
        num_err_8PAM(snr)=bits_error_8PAM/length(input_binary);
        num_err_8PAM_Gray(snr)=bits_error_gray/length(input_binary);


         bit_error_all(1,snr)= num_err_8PAM(snr);
         bit_error_all(2,snr)=num_err_8PAM_Gray(snr);
         bit_error_all(3,snr)=num_err_2PAM(snr);

         symbol_8_binary = de2bi(received_symbols_8PAM,3,'left-msb');
         symbol_gray_binary=de2bi(received_symbols_8PAM_Gray,3,'left-msb');

         %Υπολογισμός SER
       for i=1:length(input_binary)
            % Υπολογισμός SER για 2-PAM
            if(~isequal([input_binary(i)],received_symbols_2PAM(i)))
                SER_2PAM= SER_2PAM+1;
            end
            if((mod(i+2,3)==0)&&(i<(length(input_binary)-2)))
             % Υπολογισμός SER για 8-PAM
            if(~isequal([input_binary(i) input_binary(i+1)
input_binary(i+2)],symbol_8_binary(floor(i/3)+1,:)))
                SER_8PAM= SER_8PAM+1;
            end

            % Υπολογισμός SER για 8-PAM για κωδικοποίηση κατά Gray
if(~isequal([gray_vector(i) gray_vector(i+1)
gray_vector(i+2)],symbol_gray_binary(floor(i/3)+1,:)))
                SER_Gray= SER_Gray+1;
            end
            end




        end
                    signal_error_2PAM(snr)= SER_2PAM/(input_size);
                    signal_error_8PAM(snr)= SER_8PAM/(input_size/3);
                    signal_error_Gray(snr)= SER_Gray/(input_size/3);


end
```

```
subplot(1,2,1);
semilogy(SNR, num_err_8PAM, 'r-*');
hold on
semilogy(SNR, num_err_2PAM, 'b-o');
semilogy(SNR, num_err_8PAM_Gray, 'g-x');
title('BER');
xlabel('SNR');
legend('8-PAM','2-PAM', 'GRAY' );

hold off
subplot(1,2,2);
semilogy(SNR, signal_error_8PAM, 'r-*');
hold on
semilogy(SNR, signal_error_2PAM, 'b-o');
semilogy(SNR, signal_error_Gray, 'g-x');
title('SER');
xlabel('SNR');
legend('8-PAM','2-PAM', 'GRAY' );

hold off
```

decode_gray.m

```
function output_vector_gray_decoded=decode_gray(output_vector_gray, n_bits)
%  Αποκωδικοποιέι ένα διάνυσμα των N bits
    n=length(output_vector_gray);
    for i=1:n_bits:n-n_bits+1
        output_vector_gray_decoded(i)=output_vector_gray(i);
        for j=1:n_bits-1

output_vector_gray_decoded(i+j)=xor(output_vector_gray(i+j),output_vector_gray_d
ecoded(i+j-1));
        end
    end
end
```

demapper.m

```
function output_vector=my_demapper(received_symbols, M)
% επιστρέφει μια δυαδική ακολουθία βασισμένη στα σύμβολα που λαμβάνει
    output_array=de2bi(received_symbols);
    temp=1;
    for i=1:length(received_symbols)
        if log2(M) == 3
            output_vector(temp) = output_array(i, 3);
            temp = temp+1;
            output_vector(temp) = output_array(i, 2);
            temp = temp+1;
            output_vector(temp) = output_array(i, 1);
            temp = temp+1;
        else
            output_vector(temp) = output_array(i);
            temp = temp+1;
        end
    end
end
```

my_demodulator.m

```
%Υλοποίηση του αποδιαμορφωτή
%signal_AWGN <- Ληφθέν σήμα
function r = my_demodulator( AWGN_signal,t,Tsymbol,fc)
%Ορθογώνιος παλμός
g_t=sqrt(2/Tsymbol);
%Συσχέτιση της φέρουσας με τον ορθογώνιο παλμό
r_1 =  AWGN_signal.*g_t.*cos(2*pi*fc*t).*t;
r=sum(r_1,2);
end
```

my_detector.m

```
%Υλοποίηση του φωρατή
%r <- ληφθέν σήμα
%M <- το πλήθος των συμβόλων
function output_symbols = my_detector(received, Am)
Am=20*Am;
%Εύρεση του συμβόλου που στάλθηκε
%Υπολογισμός της μικρότερης απόστασης μεταξύ των διανυσμάτων  r & s_m
%με χρήση της ευκλείδιας απόστασης
for i = 1:length(received)
    %Αρχικοποίηση με την μέγιστη τιμή
    min = realmax;
    %Αρχικοποίηση με την μικρότερη θέση
    pos = 1;
    for j = 0:length(Am)-1
        euclidean_dist(j+1) = sqrt((received(i, 1)-Am(j+1))^2);
        if (euclidean_dist(j+1) < min)
            min = euclidean_dist(j+1);
            pos = j;
        end
    end
output_symbols(i) = pos;
end
end
```

add_awgn.m

```
%Δημιουργία θορύβου
function noise= add_awgn(SNR_dB,sz,mm)
    SNR = power(10,(SNR_dB/10)); %linear to scale
    N0=1/SNR;
    sgma = sqrt(N0/2);
    noise= sgma*(randn(sz)+mm*randn(sz));%computed noise
end
```

my_mapper.m

```
%Υλοποίηση του mapper
%Αντιστοίχηση των log2(M) bits εισόδου σε M σύμβολα
function Sm= my_mapper(Am,t,Tsymbol,fc)
pulse=gt_pulse(Tsymbol,t);
for i=1:length(t)
    Sm(i)=Am*pulse(i)*cos(2*pi*fc*t(i));
end
end
```

ΜΕΡΟΣ Β

Συμπίεση Διακριτής Πηγής με Χρήση της Κωδικοποίησης DPCM

ΕΡΩΤΗΜΑ Α

Υλοποίηση συστήματος DPCM

Για την υλοποίηση του συστήματος DPCM αρχικά χρειάστηκε να υλοποιηθεί ένας ομοιόμοφος κβαντιστής N δυαδικών ψηφίων,δηλαδη 2^N επιπέδων.Στήν συνέχεια για την υλοποίηση του DPCM υλοποιήθηκε και το φίλτρο πρόβλεψης όπως αναφέρεται στην εκφώνηση.Το φίλτρο πρόβλεψης βασίζεται στις p προηγούμενες τιμές που είναι αποθηκευμένες στη μνήμη,παρ'όλα αυτά για την αρχικοποίηση του συστήματος θεωρήθηκε ότι οι p πρώτες τιμές μεταδίδονται χωρίς σφάλματα.

Για το ερώτημα αυτό δημιουργίθηκαν οι συναρτήσεις :

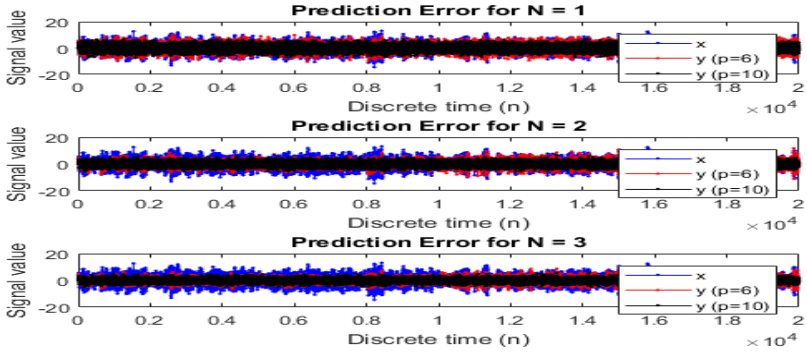
- my_quantizer(input, N, min_value, max_value):η συνάρτηση του κβαντιστή επιστρέφει την κβαντισμένη εκδοχή του διανύσματος εισόδο με βάση έναν ομοιόμορφο κβαντιστή 2^N επιπέδων και δυναμική περιοχή εύρους από max_value μέχρι min_value.

- DPCM(x, p, N): η συνάρτηση υπολογίζει αρχικά τις τιμές των συντελεστών πρόβλεψης που είναι απαραίτητοι για την πρόβλεψη μελλοντικών τιμών εισόδου

και στη συνέχεια υπολογίζει για το αρχικά διακριτό σήμα x ένα ανακατασκευασμένο σήμα y^' το οποίο δημιουργείται με πρόβλεψη.

ΕΡΩΤΗΜΑ Β

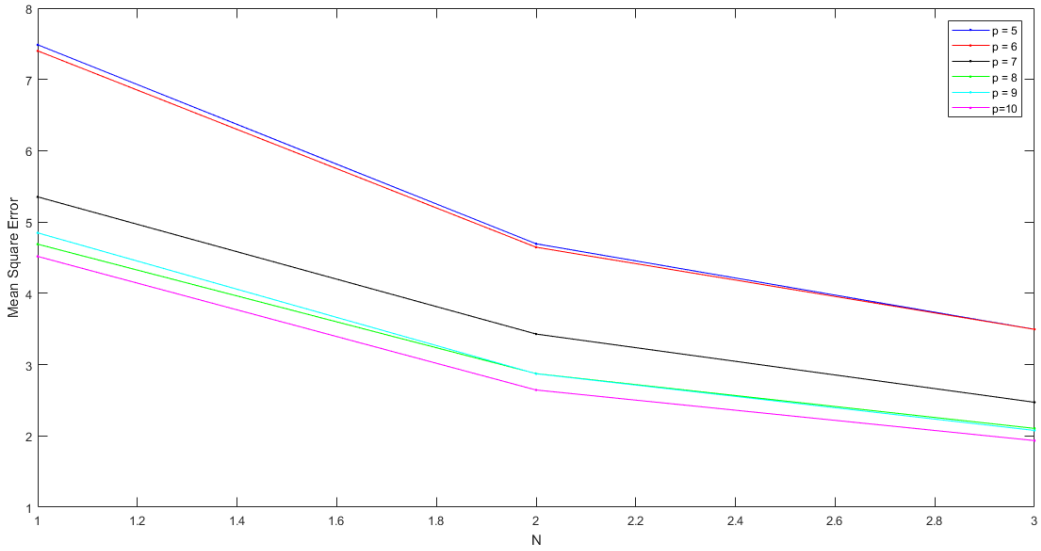
Για την αξιολόγηση της απόδοσης του προβλέπτη του συστήματος DPCM, λήφθηκαν οι αντίστοιχες μετρήσεις του σφάλματος πρόβλεψης για p = 6,10 και N = 1,2,3 bits. Συγκεριμένα παρουσιάζονται παρακάτω οι μετρήσεις του σφάλματος πρόβλεψης που λήφθηκαν σε σχέση με το αρχικό σήμα.
Παρατηρώντας την γραφική παράσταση της παρακάτω εικόνας βλέπουμε πως καθώς το N αυξάνεται το μέσο τετραγωνικό σφάλμα μειώνεται για κάκε ένα από τα διαφορετικά p, ενώ μάλιστα ο ρυκθός με τον οποίο φθίνει είναι μεγάλος.Το γεγονός αυτό οφείλεται στη μεγαλύτερη ακρίβεια που μας προσφέρουν τα επιπλέον bits κβάντιση.



ΕΡΩΤΗΜΑ Γ

Για την περαιτέρω αξιολόγηση του προβλέπτη του συστήματος DPCM, λήφθηκαν οι αντίστοιχες τιμές του προβλεπόμενου σήματος για p = 5 : 10και N = 1,2,3 bits. Συγκεκριμένα για κάθε συνδυασμό των παραμέτρων p και N υπολογίστηκε και απεικονίστηκε αντίστοιχα το μέσο τετραγωνικό σφάλμα πρόβλεψης ως:

$$E(y^2) = E((x-y')^2)$$



Από το παραπάνω γράφημα μπορούμε να συμπεράνουμε ότι η απόδοση του φίλτρου πρόβλεψης του σήματος επηρεάζεται σχεδόν αποκλειστικά από την ακρίβεια κβάντισης των δειγμάτων. Όσο περισσότερα bits κβάντισης χρησιμοποιούνται, τόσο περισσότερο αξιόπιστα προσεγγίζει το αρχικό σήμα ο προβλέπτης .

p=5	p=6	p=7	p=8	p=9	p=10
1.2853	1.2866	1.2651	1.0876	1.1432	1.1064
-1.5857	-1.5627	-1.5208	-1.3074	-1.2975	-1.1763
0.9902	0.9481	1.1886	0.9403	0.9285	0.8947
-0.5424	-0.4750	-0.9547	-0.6198	-0.5740	-0.5566
-0.0288	-0.0834	0.7069	0.2901	0.1918	0.2360
0	0.0425	-0.6081	-0.0750	0.0742	-0.0584
0	0	0.5056	0.0621	-0.1455	0.0693
0	0	0	0.3505	0.5233	0.2231
0	0	0	0	-0.1589	0.1058
0	0	0	0	0	-0.2314

Παρατηρούμε ότι για τα διαφορέτικα p οι συντελεστές α είναι αρκετά κοντά αυτό ωφείλεται στο γεγονός ότι σε οι συντελεστές είναι βασισμένοι στα x που δίνουμε σαν είσοδο τα οποία τα πολλαπλασιάζουμε συνάρτηση του p γιαυτό το λόγο οι τιμές τους είναι διαφορετικές.

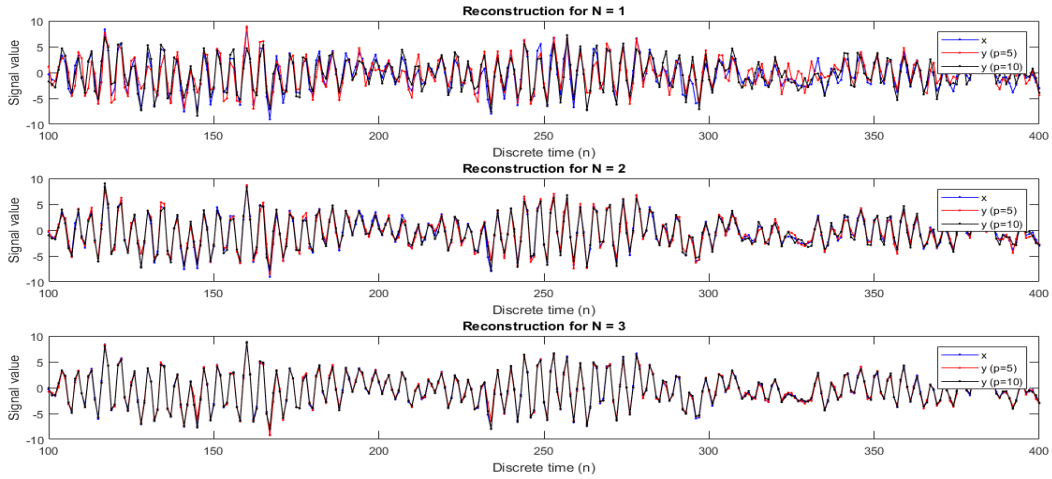
Κβαντισμένες τιμές αντίστοιχων συντελεστών πρόβλεψης:

p=5	p=6	p=7	p=8	p=9	p=10
1.2891	1.2891	1.2578	1.0859	1.1484	1.1016
-1.5859	-1.5703	-1.5234	-1.3047	-1.3047	-1.1797
0.9922	0.9453	1.1953	0.9453	0.9297	0.8984
-0.5391	-0.4766	-0.9609	-0.6172	-0.5703	-0.5547
-0.0234	-0.0859	0.7109	0.2891	0.1953	0.2422
0	0.0391	-0.6016	-0.0703	0.0703	-0.0547
0	0	0.5078	0.0547	-0.1484	0.0703
0	0	0	0.3516	0.5234	0.2266
0	0	0	0	-0.1641	0.1016
0	0	0	0	0	-0.2266

Για το ΕΡΩΤΗΜΑ 3 υλοποιήθηκε η συνάρτηση mean_sq(x) μέσω της οποία υλογίζουμε το μέσω τετραγωνικό σφάλμα.Η εκτέλεση του ερωτήματος γίνεται μέσω του script3.

ΕΡΩΤΗΜΑ Δ

Για την αξιολόγηση του σύστημα μας λήφθηκαν οι αντίστοιχες τιμές του αρχικού σήματος και του ανακατασκευασμένου σήματος στο δέκτη.Πιο συγκεκριμένα για τον συνδιασμό των τιμών p=6,10 και bits=1,2,3 απεικονίστηκε το αρχικό και το ανακατασκευασμένο σήμα.Για να είναι πιο ευδιάκριτες οι διαφορές περιοριστήκαμε σε ένας μέρος δειγμάτων καθώς το γράφημα ήταν πολύ πυκνό λόγω του μεγάλου όγκου των δεδομένων εισόδου.



Παρατηρώντας τις μετρήσεις γίνεται προφανές ότι όσα περισσότερα bits χρησιμοποιούνται τόσο καλύτερη απόδοση έχουμε,παρ'όλα αυτά σε μεγάλες μεταβολές του σήματος το σύστημα δεν ανταποκρίνεται κατάλληλα.

Κώδικες για το μέρος Β:

DPCM.m

```
%Υλοποίηση DPCM
function [y1,y2]=DPCM(x,p,bits)
%μέγεθος δείγματος
N=length(x);
r=zeros;
%υπολογισμός διανύσματος αυτοσυσχέτισης διάστασης(px1)
for i=1:p
    s=0;
    for n=p+1:N
        s=s+x(n)*x(n-i);
    end
    r(i)=(1/(N-p))*s;
end
r=r';
%υπολογισμός πίνακα αυτοσυσχέτισης διάστασης(pxp)
for i=1:p
    for j=1:p
        s=0;
        for n=p:N
            s=s+x(n-j+1)*x(n-i+1);
        end
        R(i,j)=(1/(N-p+1))*s;
    end
end
%υπολογισμός συντελεστών πρόβλεψης
a=R\r;
%υπολογισμός κβατισμένων συντελεστών πρόβλεψης
a_quantum = my_quantizer(a,8,-2,2)';
%θεωρούμε ότι τα p πρώτα δείγματ μεταδίδονται
%μη κβαντισμένα και χωρίς σφάλματα
%αποθήκευση p αρχικών τιμών στη μνήμη
mem(1:p)=x(1:p)';
y2(1:p)=mem(1:p)';
%Υλοποίηση του φίλτρου πρόβλεψης
for j=p+1:N
    sum = 0;
    for i = 1:p
        sum = sum + a_quantum(i)*mem(j-i);
    end
    %πρόβλεψη δείγματος
    prediction(j) = sum;
    %υπολογισμός σφάλματος πρόβλεψης
    y1(j)=x(j)-prediction(j);
    %κβάντιση του σφάλματος πρόβλεψης
    [y1_qua(j)]=my_quantizer(y1(j),bits,-3.5,3.5)';
    %ανακατασκεύη του δείγματος
    y2(j)=y1_qua(j)+prediction(j);
    mem(j)=y2(j);
end
end
```

my_quantizer.m

```
%Υλοποίηση κβαντιστή
%y<-- τρέχον δείγμα του σφάλματος πρόβλεψης
%N<-- ο αριθμός των bit που θα χρησιμοποιηθούν
function [ myqua ] = my_quantizer( y,N,min_value,max_value )
%Υπολογισμός κέντρων κβαντισμού
D=max_value/2^(N-1);
%Υπολογισμός των κέντρων κάθε περιοχής
centers(1) = max_value- D/2;
centers(2^N) = min_value+D/2;
for i = 2:(2^N-1)
    centers(i) = centers(i-1)-D;
end
%Υπολογισμός περιοχής στην οποία ανήκει το δείγμα
for j=1:length(y)
    if y(j)<=min_value
        myqua(j)=2^N;
    elseif y(j)>=max_value
        myqua(j)=1;
    else
        if y(j)<0
            y(j)=max_value+abs(y(j));
        elseif y(j)>=0
            y(j)=max_value-y(j);
        end
        myqua(j)=floor(y(j)/D)+1;
    end
    myqua(j)=centers(myqua(j));
end
end
```

mean_sq.m

```
function [ exp_sq ] = mean_sq(x)
%Υπολογισμός του μέσου τετραγωνικού σφάλματος
    exp_sq = 0;
    for i = 1:length(x)
        exp_sq = x(i)^2 + exp_sq ;
    end
    exp_sq = exp_sq / length(x);

end
```

script2.m

```
%Ερώτημα 2ο
load source.mat;
x=t;
n=1:length(x);
[y1a ,y2a]=DPCM(x,6,1);
[y1b ,y2b]=DPCM(x,10,1);

[y12a ,y22a]=DPCM(x,6,2);
[y12b, y22b]=DPCM(x,10,2);

[y13a, y23a]=DPCM(x,6,3);
[y13b,y23b]=DPCM(x,10,3);

figure
subplot(3,1,1)
plot(n,x,'b.-', n,y1a,'r.-',n ,y1b,'k.-');
    legend('x','y (p=6)','y (p=10)');
    xlabel('Discrete time (n)');
```

```
    ylabel ('Signal value');
    title('Prediction Error for N = 1');
    subplot(3,1,2)
plot(n,x,'b.-', n,y12a,'r.-',n ,y12b,'k.-');
    legend('x','y (p=6)','y (p=10)');
    xlabel('Discrete time (n)');
    ylabel ('Signal value');
    title('Prediction Error for N = 2');
    subplot(3,1,3)
plot(n,x,'b.-', n,y13a,'r.-',n ,y13b,'k.-');
    legend('x','y (p=6)','y (p=10)');
    xlabel('Discrete time (n)');
    ylabel ('Signal value');
    title('Prediction Error for N = 3');
```

script3.m

```
%Ερωτημα 3ο
clear;clc;
close all;
load source;
x=t;
p = 5:10;
N = 1:3;
M=length(x);

%Αρχικοποίηση των πινάκων για μείωση του χρόνου προσομοίωσης
a_hat = zeros(max(p),length(p));
Sq_Exp = zeros(length(p),length(N));

for i = 1:length(p)
    %Υπολογισμός μέσου τετραγωνικού σφάλματος για όλα τα p,N
    for j = 1:length(N)
        y = DPCM(x,p(i),N(j));
        Sq_Exp(i,j) = mean_sq(y);

    end
    p_2=p(i);
    for index=1:p_2
        s=0;
        for n=p_2+1:M
            s=s+x(n)*x(n-index);
        end
        r(index)=(1/(M-p_2))*s;
    end
```

```
r=r';
    for index3=1:max(p)
        if index3>length(r)
            test_r(index3,i)=0;
        else
           test_r(index3,i)=r(index3);
        end
    end
    for index=1:p_2
        for index2=1:p_2
            s=0;
                for n=p_2:M
                    s=s+x(n-index2+1)*x(n-index+1);
                end
                R(index,index2)=(1/(M-p_2+1))*s;
        end
    end
        b=(R\r);
        for index3=1:max(p)
            if index3>length(b)
            a(index3,i)=0;
            else
            a(index3,i)=b(index3);
            end
        end
    clear R r s;
            a_temp = my_quantizer(b,8,-2,2);
            for j = 1:p(i)
               a_hat(j,i) = a_temp(j);
            end
end

%Απεικόνιση τιμών
plot(N,Sq_Exp(1,:),'b.-', N,Sq_Exp(2,:),'r.-', N,Sq_Exp(3,:),'k.-',
N,Sq_Exp(4,:),'g.-', N,Sq_Exp(5,:),'c.-',N,Sq_Exp(6,:),'m.-');
    legend('p = 5','p = 6','p = 7','p = 8','p = 9','p=10');
    xlabel('Αριθμός bits κβάντισης')
    ylabel ('Μέσο Τετραγωνικό Σφάλμα Πρόβλεψης');

plot(N,Sq_Exp(1,:),'b.-', N,Sq_Exp(2,:),'r.-', N,Sq_Exp(3,:),'k.-',
N,Sq_Exp(4,:),'g.-', N,Sq_Exp(5,:),'c.-',N,Sq_Exp(6,:),'m.-');
    legend('p = 5','p = 6','p = 7','p = 8','p = 9','p=10');
    xlabel('Αριθμός bits κβάντισης')
    ylabel ('Μέσο Τετραγωνικό Σφάλμα Πρόβλεψης');
```

script4.m

```
%Ερώτημα 4ο
load source.mat
x=t;
n=100:400;

[y1a ,y2a]=DPCM(x,5,1);
[y1b ,y2b]=DPCM(x,10,1);

[y12a ,y22a]=DPCM(x,5,2);
[y12b, y22b]=DPCM(x,10,2);

[y13a, y23a]=DPCM(x,5,3);
[y13b,y23b]=DPCM(x,10,3);
```

```
figure
subplot(3,1,1);
    plot(n,x(100:400),'b.-', n,y2a(100:400),'r.-', n,y2b(100:400),'k.-');
    legend('x','y (p=5)','y (p=10)');
    xlabel('Discrete time (n)');
    ylabel ('Signal value');
    title('Reconstruction for N = 1');
subplot(3,1,2);
    plot(n,x(100:400),'b.-', n,y22a(100:400),'r.-', n,y22b(100:400),'k.-');
    legend('x','y (p=5)','y (p=10)');
    xlabel('Discrete time (n)');
    ylabel ('Signal value');
    title('Reconstruction for N = 2');
subplot(3,1,3);
    plot(n,x(100:400),'b.-', n,y23a(100:400),'r.-', n,y23b(100:400),'k.-');
    legend('x','y (p=5)','y (p=10)');
    xlabel('Discrete time (n)');
    ylabel ('Signal value');
    title('Reconstruction for N = 3');
```

Σπάσιμο της πηγής ανά 5000 δείγματα:

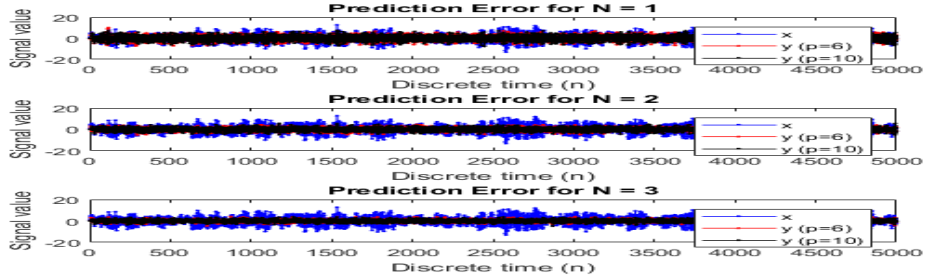
```
v=t;

Len_v=length(v);
Div_v=Len_v/4;
%0-5000
%x=t;
%x=v(1:Div_v);
%n=1:Div_v;
%5000-10000
%x=v(Div_v+1:2*Div_v);
%n=Div_v+1:2*Div_v;
%10000-15000
%x=v(2*Div_v+1:3*Div_v);
%n=2*Div_v+1:3*Div_v;
%15000-20000
%x=v(3*Div_v+1:4*Div_v);
%n=3*Div_v+1:4*Div_v);
```

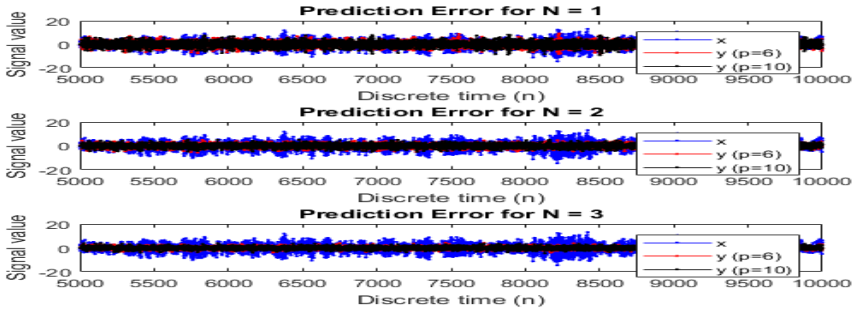
Για να υπολοποιήσουμε πάλια τα ερωτήματα προσθέσαμε τις παραπανω εντολές στα script2.m,script3.c,script4.m και πήραμε τις αντίστοιχες μετρήσεις που παρατίονται παρακάτω:

Ερώτημα 2ο

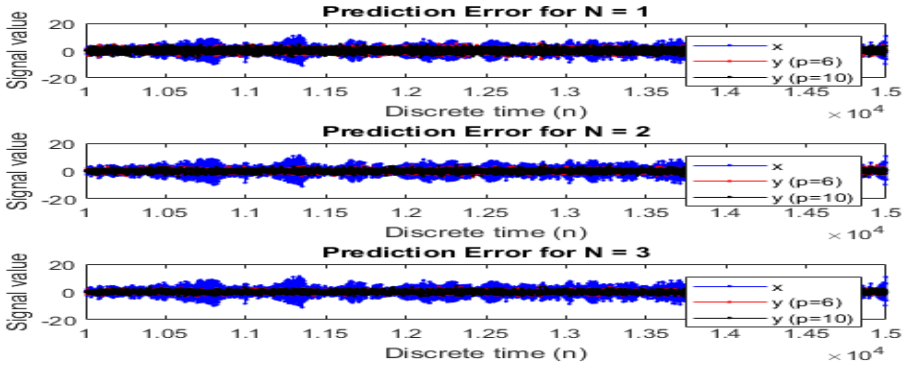
```
%0-5000
x=v(1:Div_v);
n=1:Div_v;
```



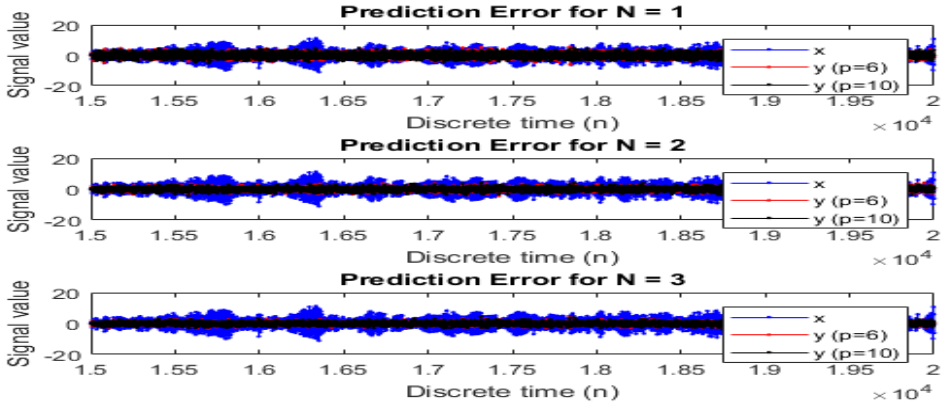
```
%5000-10000
x=v(Div_v+1:2*Div_v);
n=Div_v+1:2*Div_v;
```




```
%10000-15000
x=v(2*Div_v+1:3*Div_v);
n=2*Div_v+1:3*Div_v;
```



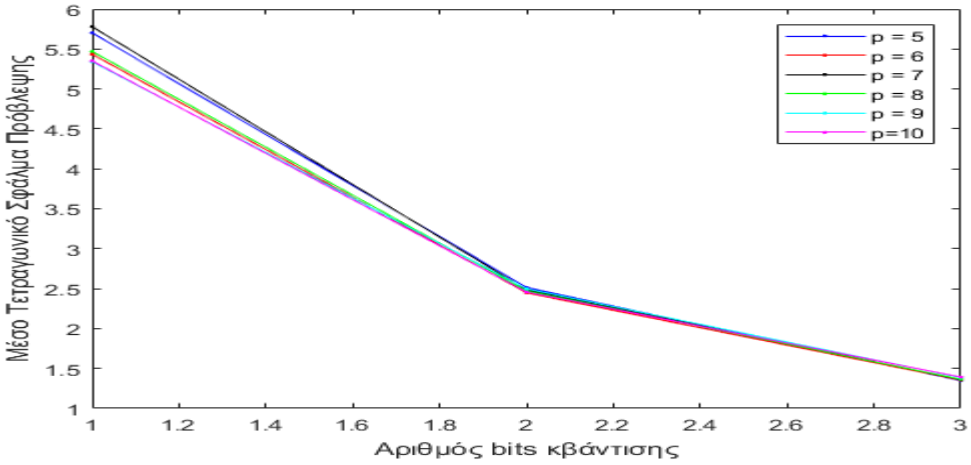

```
%15000-20000
x=v(3*Div_v+1:4*Div_v);
n=3*Div_v+1:4*Div_v);
```



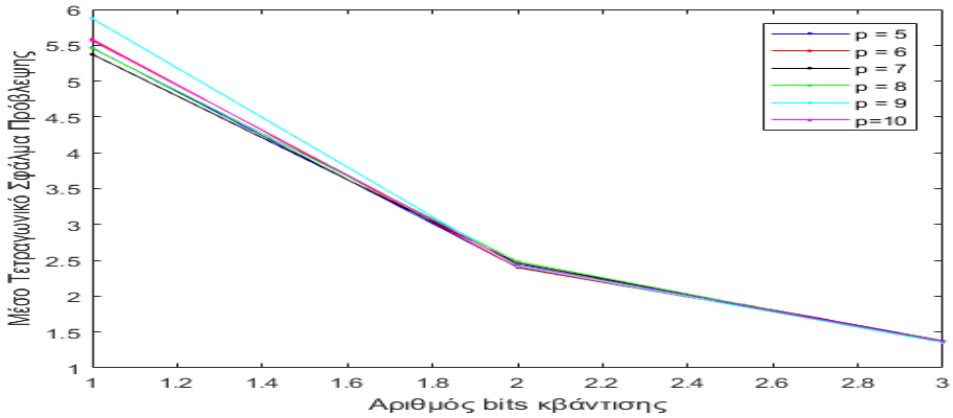
Παρατηρώντας την γραφική παράσταση τις παραπάνω γραφικές παραστάσεις και συγκρινοντάς τες με την γραφική παράσταση του 2ου ερωτήματος παρατηρούμε ότι το μέσω τετραγωνικό σφάλμα ελατώνεται με μεγαλύτερο ρυθμό καθώς αυξάνεται το N.

Ερώτημα 3ο

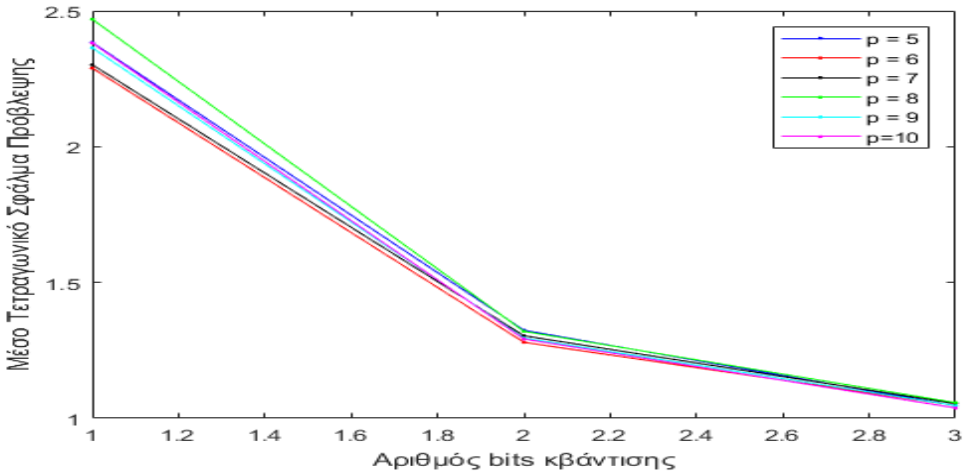
```
%0-5000
x=v(1:Div_v);
n=1:Div_v;
```



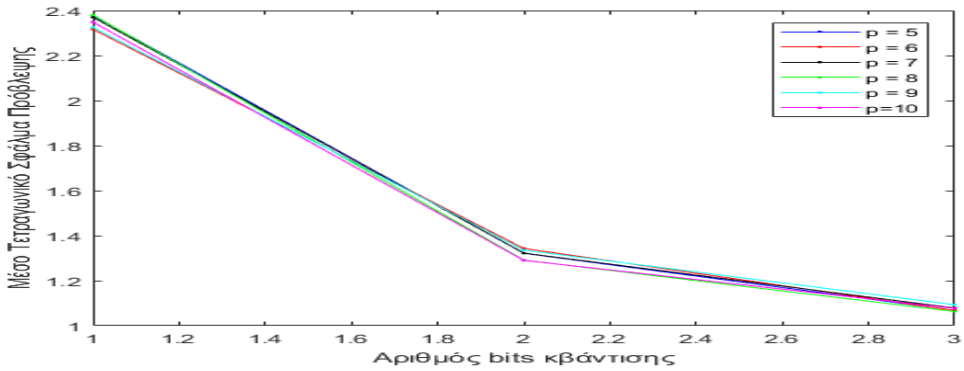

```
%5000-10000
x=v(Div_v+1:2*Div_v);
n=Div_v+1:2*Div_v;
```



```
%10000-15000
x=v(2*Div_v+1:3*Div_v);
n=2*Div_v+1:3*Div_v;
```




```
%15000-20000
x=v(3*Div_v+1:4*Div_v);
n=3*Div_v+1:4*Div_v);
```



Παρατηρώντας τις παραπάνω γραφικές παραστάσεις και συγκρίνωντας τες με τις αρχικές καταλήγουμε στο συμπέρασμα ότι το μέσω τετραγωνικό σφάλμα πρόβλεψης αλλάζει ανάλογα με τις διαφορετικές εισόδους και από τα διαφορετικά p.

Ερώτημα 4ο

```
%0-5000
x=v(1:Div_v);
n=1:Div_v
```