ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ

2014 – 2015

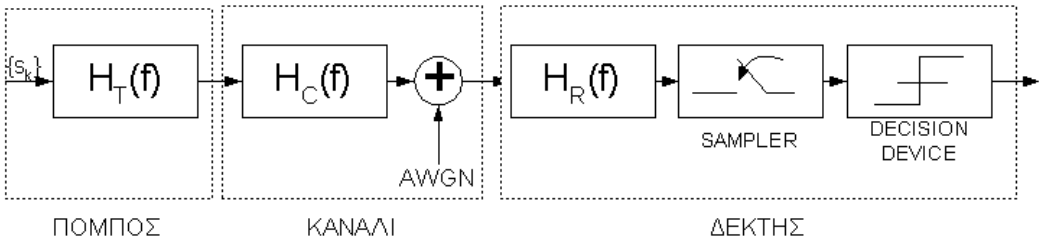
2η Εργαστηριακή Άσκηση

Γεώργιος-Κάρολος Λύκος 4758

(6ο έτος)

Όλοι κώδικες που αναφέρονται ονομαστικά στα παρακάτω ζητούμενα υλοποιήθηκαν στο περιβάλλον της Matlab και παραθέτονται στο τμήμα ***ΚΩΔΙΚΕΣ***

Στην άσκηση αυτή εξομοιώνω



ένα τηλεπικοινωνιακό σύστημα βασικής ζώνης και εξετάζω την επίδοση του για διάφορα είδη διαμόρφωσης σε ιδανικό και μη ιδανικό κανάλι.

Στις συναρτήσεις *epsk.m , eqam.m* εξομοιώνεται ένα τηλεπικοινωνιακό σύστημα ,όπως στο παραπάνω σχήμα ,όπου στην είσοδο του πομπού εφαρμόζονται κυματομορφές 8-PSK, 8-QAM αντίστοιχα υπό μορφή διανυσμάτων και το κανάλι είναι ιδανικό. Αντίστοιχα για μη ιδανικό κανάλι δημιουργήθηκαν οι συναρτήσεις *epsk_channel.m ,eqam_channel.m* .
Πριν το σύστημα του πομπού υπονοείται ότι υπάρχει ένας ψηφιακός διαμορφωτής που αντιστοιχεί δυαδικές ακολουθίες σε σύμβολα.

Στα πλαίσια της άσκησης παράγεται τυχαία ακολουθία δυαδικών συμβόλων η οποία μετατρέπεται σε σύμβολα 8PSK διαμόρφωσης με την συνάρτηση *btos.m* ή σε σύμβολα 8QAM διαμόρφωσης με την συνάρτηση *btoq.m .*

Έπειτα τα σύμβολα μπαίνουν ως είσοδο στον πομπό ο οποίος λειτουργεί ως φίλτρο τετραγωνικής ρίζας ανυψωμένου συνημιτόνου και στην ουσία κάνει συνέλιξη το διάνυσμα εισόδου με το διάνυσμα που παράγεται από την συνάρτηση *rcosfir.m* με παράγοντα αναδίπλωσης $\beta = 0.3$, χρονική έκταση φίλτρου διάρκειας 6 περιόδων και υπερδειγματοληψία του φίλτρου με 4 δείγματα ανά περίοδο.

Έτσι προέκυψε το ζήτημα αφού γίνεται εξομοίωση των φίλτρων πομπού και δέκτη με υπερδειγματοληψία κατά 4 κάνουμε το ίδιο και για την ακολουθία συμβόλων στην είσοδο του πομπού και για την κρουστική απόκριση του καναλιού με την χρήση της συνάρτησης upsample() της matlab βάζοντας 3 μηδενικά μετά από κάθε σύμβολο, και μετέπειτα αφαίρεση των παραπάνω τιμών με την downsample() .

Στην συνέχεια στις συναρτήσεις *one.m ,two.m, three.m* η κυματομορφή , αφού αλλοιωθεί από προσθετικό λευκό Gaussian θόρυβο με τη συνάρτηση *mynoise.m* για 8PSK και *mynoise_complex.m* για 8QAM, περνάει κατευθείαν στην είσοδο του δέκτη λόγω ότι το κανάλι είναι ιδανικό και δεν εισάγει κάποια παραμόρφωση.

Από την άλλη μεριά για το μη ιδανικό κανάλι στις συναρτήσεις *epsk_channel.m ,*

eqam_channel.m η κυματομορφή , αφού αλλοιωθεί από προσθετικό λευκό Gaussian θόρυβο , δεν περνάει κατευθείαν στην είσοδο του δέκτη λόγω ότι το κανάλι είναι μη ιδανικό αλλά υπόκειται κάποια παραμόρφωση από την συνέλιξη της εξόδου του πομπού με την κρουστική απόκριση του καναλιού που μας δίνεται.

Εδώ να τονίσουμε ότι αφού και η είσοδος συμβόλων και τα φίλτρα είναι υπερδειγματοληπτημένα κατά 4 , το ίδιο γίνεται και για την κρουστική απόκριση του καναλιού βάζοντας 3 μηδενικά μετά από κάθε στοιχείο με την συνάρτηση upsample() της matlab .

Στην συνέχεια η κυματομορφή εισέρχεται στην είσοδο του δέκτη που και αυτός λειτουργεί ως φίλτρο τετραγωνικής ρίζας ανυψωμένου συνημιτόνου και γίνεται συνέλιξη στην ουσία της εισόδου με το διάνυσμα που μας επιστρέφει η rcosfir() .

Έπειτα επειδή τα φίλτρα εισάγουν κάποια καθυστέρηση στην ακολουθία της εισόδου πρέπει να πετάξουμε κάποιες αρχικές τιμές ανάλογα αν το φίλτρο είναι ιδανικό και μη ιδανικό. Πιο συγκεκριμένα για το ιδανικό φίλτρο πετάμε τις 48 πρώτες τιμές (24 για το φίλτρο πομπού και 24 για το φίλτρο δέκτη) όπως και τις τελευταίες 48 τιμές ώστε να είναι ευθυγραμμισμένα τα στοιχεία της εισόδου με την έξοδο χωρίς καθυστερήσεις. Για το μη ιδανικό φίλτρο πετάμε τις 78 πρώτες και 78 τελευταίες τιμές (24 για το φίλτρο πομπού 24 για το φίλτρο καναλιού και 24 για το φίλτρο δέκτη).

Αμέσως μετά το σήμα μας περνάει από τον δειγματολήπτη ο οποίος παίρνει 1 δείγμα κάθε 4 στοιχεία του διανύσματος της κυματομορφής και μετά αφαιρούμε τις παραπάνω τιμές που βάλαμε για την υπερδειγματοληψία με την συνάρτηση downsample() της matlab.

Μετά το διάνυσμα εξόδου περνάει από τον φωρατή ο οποίος καλείται να πάρει κάποια απόφαση για το ποιο είναι το σύμβολο που στάλθηκε με βάση την τιμή του στοιχείου εισόδου του , λειτουργώντας με βάση το ML κριτήριο. Ο φωρατής λειτουργεί για τα 3 διαφορετικά σήματα διαμόρφωσης με μία μεταβλητή στην είσοδο ορισμάτων της συνάρτησης *foratis.m* που την ονομάζω pam. Αν epilogh = 1 σημαίνει ότι αναφερόμαστε στο 8PSK σύστημα. Αν epilogh = 2 αναφερόμαστε στο 8QAM.

Τέλος αφού μετατρέψουμε τα σύμβολα πάλι σε δυαδική μορφή με τις συναρτήσεις *stob.m stoq.m* (για 8PSK και 8QAM αντίστοιχα) ,συγκρίνουμε την ακολουθία εισόδου με την ακολουθία εξόδου και υπολογίζουμε πόσο διαφορετικές είναι στα στοιχεία τους (Bit error Rating). Τα δεδομένα αυτά επιστρέφονται ώστε να γίνει

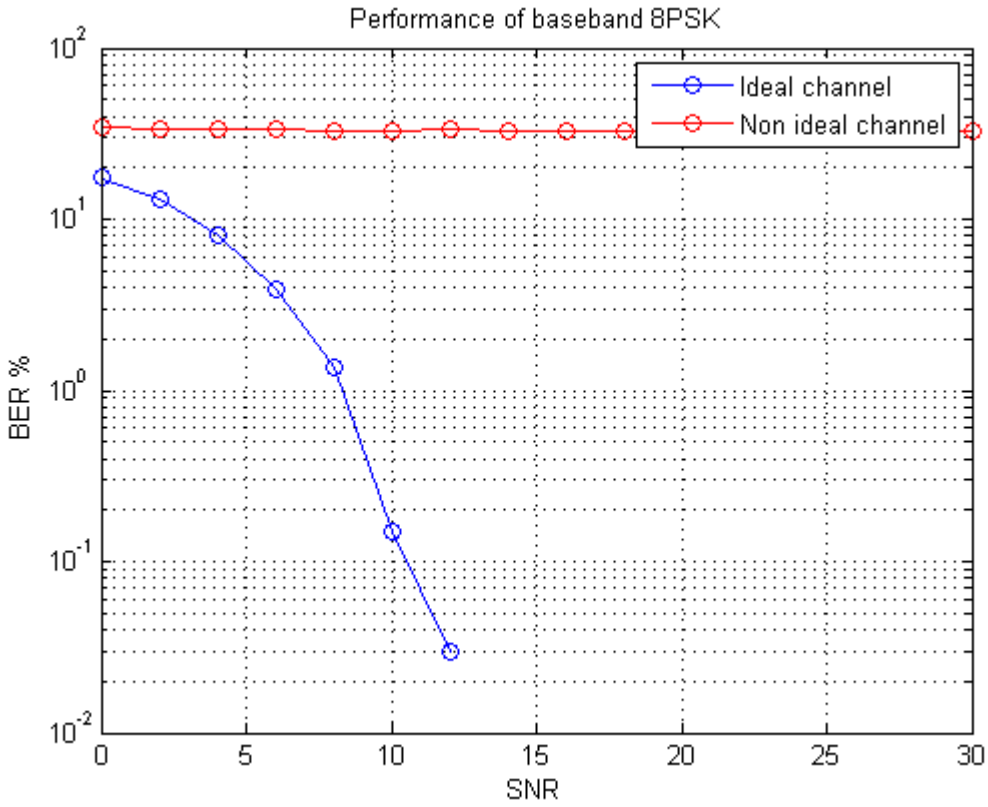
σύγκριση για τα διάφορα είδη διαμόρφωσης σε ιδανικό και μη ιδανικό κανάλι. Οι γραφικές παραστάσεις που ζητούνται για την σύγκριση υλοποιούνται στην συνάρτηση *sximata.m*.

Ειδικότερα

Για τον **θόρυβο** ο οποίος είναι AWGN ρυθμίζουμε την ισχύ του επαναληπτικά ανάλογα με το λόγο ισχύος του σήματος προς θόρυβο που θέλουμε (SNR).

Για την περίπτωση της 8QAM διαμόρφωσης ο θόρυβος όπως υποδεικνύεται προστίθεται και στο πραγματικό και στο φανταστικό μέρος των προς μετάδοση συμβόλων.
Ως αναφορά τον **φωρατή** είναι απλός και λειτουργεί σύμφωνα με τον κριτήριο ML ώστε για να πάρει απόφαση για το σύμβολο που στάλθηκε ως εξής : Υπολογίζεται η Ευκλείδια απόσταση από το στοιχείο του διανύσματος εξόδου και τα διάφορα σύμβολα της εκάστοτε διαμόρφωσης και αντιστοιχίζεται σε αυτό το σύμβολο από το οποίο έχει την ελάχιστη απόσταση.

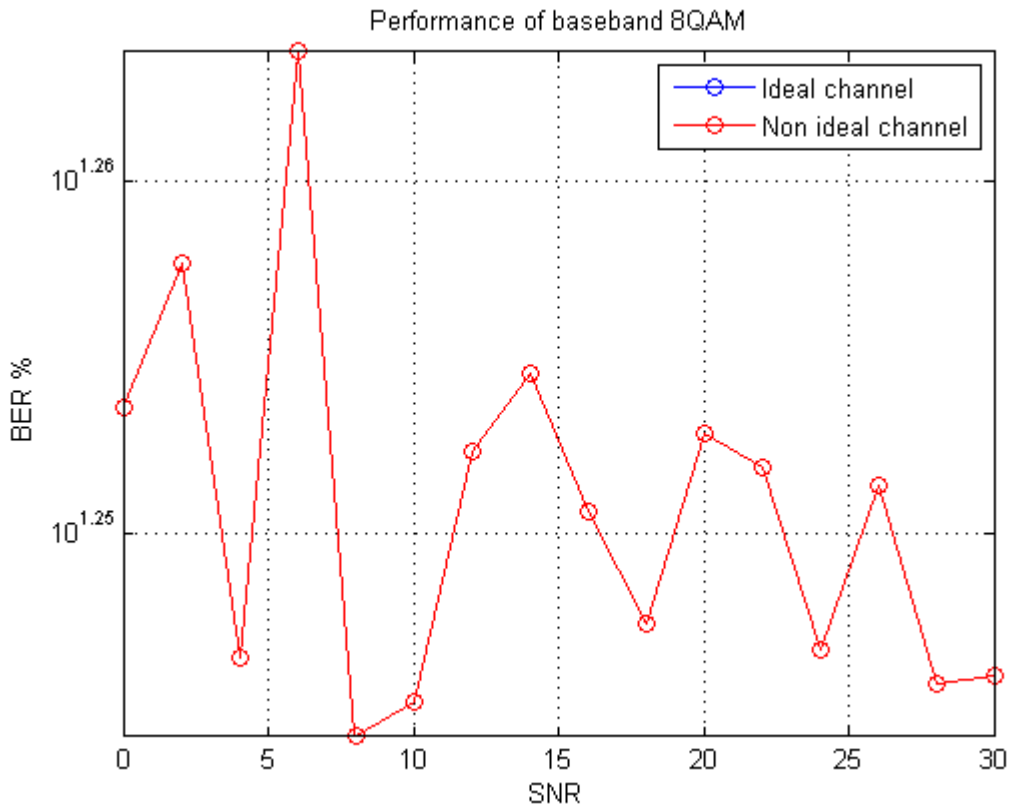
Αποτελέσματα



Παρατηρήσεις:

Παρατηρούμε ότι στο ιδανικό κανάλι αυξάνοντας την ισχύ του σήματος προς τον θόρυβο γίνονται λιγότερα λάθη κατά την μετάδοση των δυαδικών ψηφίων από το ψηφιακό σύστημα το οποίο είναι λογικό. Στην ουσία πρέπει η ενέργεια του θορύβου να είναι τέτοια ώστε το +1 να το κάνει τουλάχιστον κάτι λιγότερο από 0.

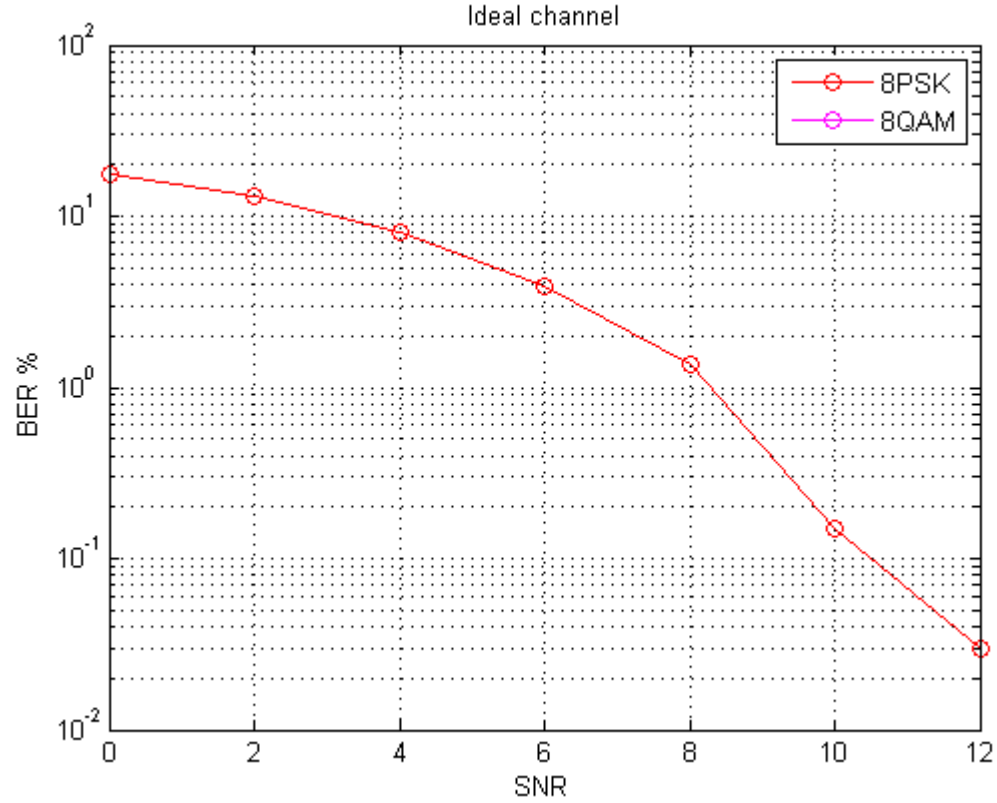
Για το μη ιδανικό κανάλι παρατηρούμε ότι τα λάθη στα δυαδικά ψηφία είναι περισσότερα απ' ότι στο μη ιδανικό, κάτι το οποίο είναι λογικό αν σκεφτούμε ότι το κανάλι εισάγει παραμόρφωση στην κυματομορφή εισόδου του.

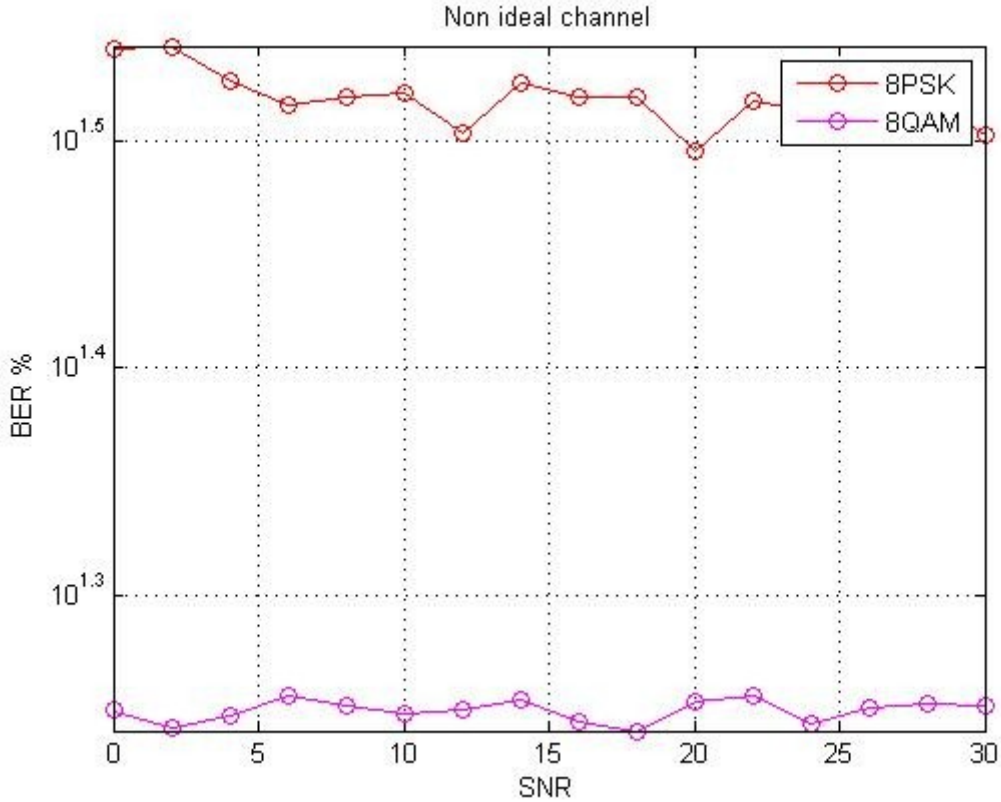


Εδώ είναι αξιοσημείωτο ότι δεν υπάρχει η γραφική παράσταση για το ιδανικό κανάλι επειδή το Bit Error Rating είναι 0 για όλες τις τιμές του SNR.

Αυτό σημαίνει ότι το 8QAM είναι ανεκτικό στον θόρυβο με 100% ακρίβεια στην ανακατασκευή του σήματος εισόδου στην έξοδο για οποιαδήποτε ισχύ του θορύβου AWGN.
Επίσης για την αντιστοιχία των δυαδικών συμβόλων σε σύμβολα του 8QAM με βάση την υπόδειξη χρησιμοποιήται κώδικας GRAY ώστε 2 γειτονικά σύμβολα να διαφέρουν κατά ένα μόνο bit.

Ας δούμε τώρα όλες της διαμορφώσεις στην ίδια γραφική παράσταση για ιδανικό και μη ιδανικό κανάλι αντίστοιχα.


Ideal channel
8PSK
8QAM
BER %
SNR
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
0
2
4
6
8
10
12


Non ideal channel
8PSK
8QAM
BER %
SNR
$10^{1.5}$
$10^{1.4}$
$10^{1.3}$
0
5
10
15
20
25
30

ΚΩΔΙΚΕΣ

function sximata()
close all
clear all

```
clc
```

% Signal to noise ratio

```
snr = 0:2:30;

% 8PSK

ber8psk = epsk() ;

ber8psk_channel = epsk_channel() ;



figure(2)
```

semilogy(snr,ber8psk,'-ob',snr,ber8psk_channel,'-or')
legend('Ideal channel','Non ideal channel')
title('Performance of baseband 8PSK')
xlabel('SNR')
ylabel('BER % ')
grid on

```

```

% 8QAM

```
ber8qam = eqam() ;

ber8qam_channel = eqam_channel() ;



figure(3)
```

semilogy(snr,ber8qam,'-ob',snr,ber8qam_channel,'-or')
legend('Ideal channel','Non ideal channel')
title('Performance of baseband 8QAM')
xlabel('SNR')
ylabel('BER % ')
grid on

```

```

% Ideal channel

```
figure(4)
```

```
semilogy(snr,ber8psk,'-or',snr,ber8qam,'-om')
legend('8PSK','8QAM')
title('Ideal channel')
xlabel('SNR')
ylabel('BER % ')
grid on
```

```
% Non ideal channel
```

```
 figure(5)
```

```
semilogy(snr,ber8psk_channel,'-or',snr,ber8qam_channel,'-om')
legend('8PSK','8QAM')
title('Non ideal channel')
xlabel('SNR')
ylabel('BER % ')
grid on
```

```
end

function [ s ] = foratis(r,epilogh)
```

```
% 8PSK symbols
if epilogh == 1
```

```
 sym = [-3/sqrt(5) -1/sqrt(5) 1/sqrt(5) 3/sqrt(5)] ;

 s = zeros(1,length(r)) ;
```

```
for i = 1: length(r)
```

```
    d(1) = (r(i) - sym(1)).^2 ;

    d(2) = (r(i) - sym(2)).^2 ;

    d(3)= (r(i) - sym(3)).^2 ;

    d(4) = (r(i) - sym(4)).^2 ;


    m = min(d);
```

```
    if m==d(1)
        s(i) = sym(1);

    elseif m==d(2)
        s(i) = sym(2);

      elseif m==d(3)
        s(i) = sym(3);

    else
        s(i) = sym(4);


    end
end
end
```

```
% 8QAM symbols
if epilogh == 2


  sym = [(-1-1i)/sqrt(2) (-1+1i)/sqrt(2) (1-1i)/sqrt(2) (1+1i)/sqrt(2)] ;

  s = zeros(1,length(r)) ;

for i = 1: length(r)


    d(1) = (r(i) - sym(1)).^2 ;

    d(2) = (r(i) - sym(2)).^2 ;

    d(3)= (r(i) - sym(3)).^2 ;

    d(4) = (r(i) - sym(4)).^2 ;



    m = min(d);

    if m==d(1)
        s(i) = sym(1);

    elseif m==d(2)
```

```
        s(i) = sym(2);

      elseif m==d(3)
        s(i) = sym(3);

   else
        s(i) = sym(4);



   end
end
end


end


ffunction [ r ] = demodulatorPSK( tr_sig )

% The function demodulates the received PSK signals.
% Input args:
%   tr_sig : Transmited signals as received from the channel.
% Output args:
%   r: Vector with the demodulated components of the received signals.

Es = 1;
Tsymbol = 40;
Tsample = 1;
Tc = 4;
Fc = 1/4;
Gt = sqrt((2 * Es) / Tsymbol);

[rows,cols] = size(tr_sig);
y1 = zeros(cols,1); y2 = zeros(cols,1);

for t = 1:cols
   y1(t,1) = Gt * cos(2 * pi * Fc * t);
   y2(t,1) = Gt * sin(2 * pi * Fc * t);
end

r = [tr_sig * y1, tr_sig * y2];

end

function [ s ] = modulatorPSK( symbols )

% The function creates 4-PSK modulator.
% Input args:
%   symbols: The symbol sequence created by the sequence of bits given to
%            the input of the PSK transmitter.
% Output args:
%   s: the PSK modulated signals created by the sequence of symbols given
%      to the input of modulator.
```

```
s = zeros(length(symbols),40);
Es = 1;
Tsymbol = 40;
Tsample = 1;
Tc = 4;
Fc = 1/4;
Gt = sqrt((2 * Es) / Tsymbol);

% PSK Modulation

for i = 1:length(symbols)
   for t = 1:Tsymbol
      s(i,t) = cos((2*pi*symbols(i)) / 4) *Gt* cos(2*pi*Fc*t) + ...
            sin((2*pi*symbols(i)) / 4) *Gt* sin(2*pi*Fc*t);
   end
end

end
function [ rec_sym ] = detectorPSK( r )

% The function chooses which symbol is most likely to have been transmitted
% based on the minimun euclidean distance from all the existing symbols.
% Input args:
%   r : Vector with demodulated signal.
% Output args:
%   rec_sym: Signals as recognised by the receiver.

[rows,cols] = size(r);
sym = zeros(4,1);

for i = 1:4
  sym(i,1) = cos(2 * pi * (i - 1) / 4);
  sym(i,2) = sin(2 * pi * (i - 1) / 4);
end

for i = 1:rows
   for j = 1:4
      comp(j,1) = norm(r(i,:) - sym(j,:));
   end
   [Y,Index] = min(comp);
   rec_sym(i,1) = Index-1;
end

end

function [ber8qam] = eqam()

counter =1 ;
for snr = 0:2:30
%Random Binary sequence
source = rand(1,10^4);
source(source>0.5) = 1;
source(source<=0.5) = 0;

% Convert to 4QAM sequence
sym = btoq(source) ;

p = upsample(sym,4);

% transmitter
rc = rcosfir(0.3, 6, 4, 0.25, 'sqrt');
t= conv(p,rc);
```

```
% thoruvos
t_noise = mynoise_complex(t,snr);

% receiver
r = conv(t_noise,rc);

r_clean = r(49:length(r)-48) ;

% Deigmatolipsia
data = zeros(1,length(r_clean)) ;
for i = 1: length(r_clean)

   if mod(i-1,4) == 0
   data(i) =  r_clean(i);
   else
      data(i) = 0;
   end

end

clean_data = downsample(data,4);

  %  ML decision making
output = foratis(clean_data,2) ;

% Convert to binary sequence
 outsym = qtob(output);

 % Bit Error Rate
BER(counter) = 100*length(source(source~=outsym))/length(source) ;

counter  = counter + 1 ;
end
%return ber values
ber8qam = BER ;

end

function [ber8qam_channel] = eqam_channel()

counter =1 ;
for snr = 0:2:30

% Random Binary sequence
source = rand(1,10^4);
source(source>0.5) = 1;
source(source<=0.5) = 0;

% Convert to 4QAM symbols sequence
sym = btoq(source) ;

p = upsample(sym,4);

% transmitter
rc = rcosfir(0.3, 6, 4, 0.25, 'sqrt');
t= conv(p,rc);

%kanali
 h = [ 0.01 0.04 -0.05 0.06 -0.22 -0.5 0.72 0.36 0 0.21 0.04 0.08 0.02] ;
 h = upsample(h,4);
 t_channel = conv(h(1:49),t);

% thoruvos
```

```
t_noise = mynoise_complex(t_channel,snr);

% receiver
r = conv(t_noise,rc);

r_clean = r(73:length(r)-72) ;

% Deigmatolipsia
data = zeros(1,length(r_clean)) ;
for i = 1: length(r_clean)

   if mod(i-1,4) == 0
   data(i) =  r_clean(i);
   else
      data(i) = 0;
   end

end

clean_data = downsample(data,4);
  %  ML decision making
output = foratis(clean_data,2) ;

% Convert to binary sequence
  outsym = qtob(output);

 % Bit Error Rate
BER(counter) = 100*length(source(source~=outsym))/length(source) ;

counter  = counter + 1 ;
end
%return ber values
ber8qam_channel = BER;

end

function h = mynoise(x,snr)
SNR = snr ;
   Ps = mean(x.^2);
   sigma = sqrt( Ps/(10^(SNR/10)) );
   N = sigma.*randn(1,length(x));
   h = x + N;
end

function h = mynoise_complex(x,snr)
SNR = snr ;
   Ps = mean(x.^2);
   sigma = sqrt( Ps/(10^(SNR/10)) );
   N = sigma.*randn(1,length(x))/sqrt(2) + imag((sigma.*rand(1,length(x))/sqrt(2)));
   h = x + N;

end
```