ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Μάθημα: Ψηφιακές Επικοινωνίες Ι

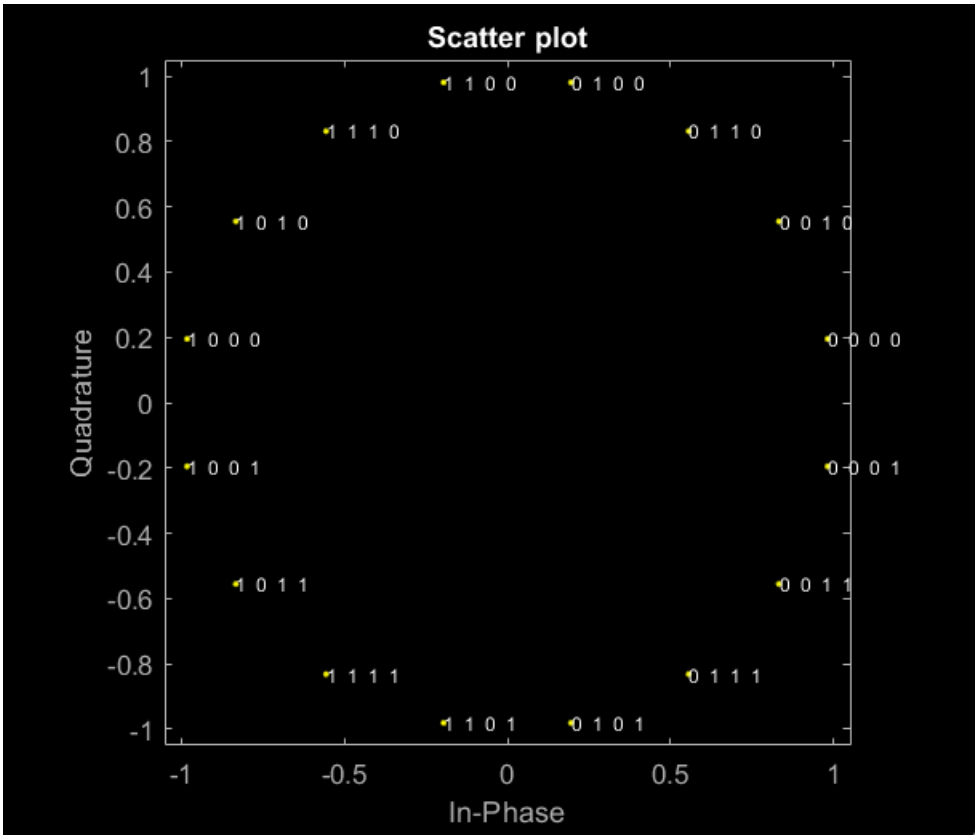
Ονοματεπώνυμο: Ειρήνη Δόντη

Α.Μ: 03119839

5η Σειρά Ασκήσεων

Αθήνα 2022

1. Σχεδιάζουμε σηματικό αστερισμό 16-PSK, με σημειωμένες δυαδικές λέξεις δίπλα σε κάθε σημείο του σε κωδικοποίηση Gray. Κάθε φορά, υποδιπλασιάζουμε τη γωνία και μετά καθρεπτίζουμε γύρω από τον φανταστικό άξονα Παρακάτω, απεικονίζεται ο ζητούμενος αστερισμός:



Για το παραπάνω διάγραμμα, δημιουργήσαμε και εκτελέσαμε το παρακάτω πρόγραμμα:

```
%Eirini Donti 03119839
function psk_constellation (k)
  % k is the number of bits per symbol
  % mapping is the vector of psk points, in the gray-coding order
  % i.e. mapping(1)<->00...00, mapping(2)<->00...01,
  % mapping(3)<->00...10, ..
  % For 16-PSK, set k=4;
  L= 2^k;
  ph1=[pi/4];
  theta=[ph1; -ph1; pi-ph1; -pi+ph1];
  mapping=exp(1j*theta);
  if(k>2)
  for j=3:k
  theta=theta/2;
  mapping=exp(1j*theta);
  mapping=[mapping; -conj(mapping)];
  theta=angle(mapping);
  end
  end
  scatterplot(mapping);
  text(real(mapping),imag(mapping),num2str(de2bi([0:L-1].',k,'left-msb')),'FontSize',7, 'Color','white');
end
```

Εκτελούμε το πρόγραμμα για 16-PSK. Οπότε, γράφουμε στο Command Window την εντολή:

```
>> psk_constellation(4)
```

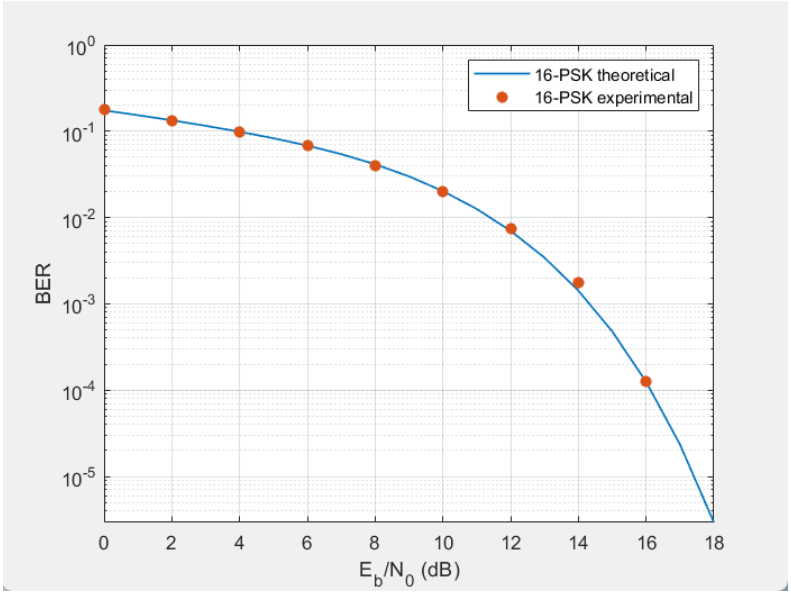
2. Το εύρος ζώνης είναι W = 6 – 4 = 2 MHz και ο ρυθμός μετάδοσης R = 6.2 Mbps. Γνωρίζουμε, από τη σχέση $\log_2 M \geq \frac{R}{W}(1+\alpha)$, $0 < \alpha \leq 1$ ή $\log_2 M \geq 3.1 \cdot (1+\alpha)$. Η μικρότερη ακέραια τιμή του M που επαληθεύει την προηγούμενη σχέση είναι M = 16. Για να εκμεταλλευτούμε όλο το εύρος ζώνης, βρίσκουμε τη μέγιστη τιμή α από τη σχέση:

$$(1+\alpha) \leq \frac{\log_2 M}{3.1} \text{ ή } \alpha \leq 0.29$$

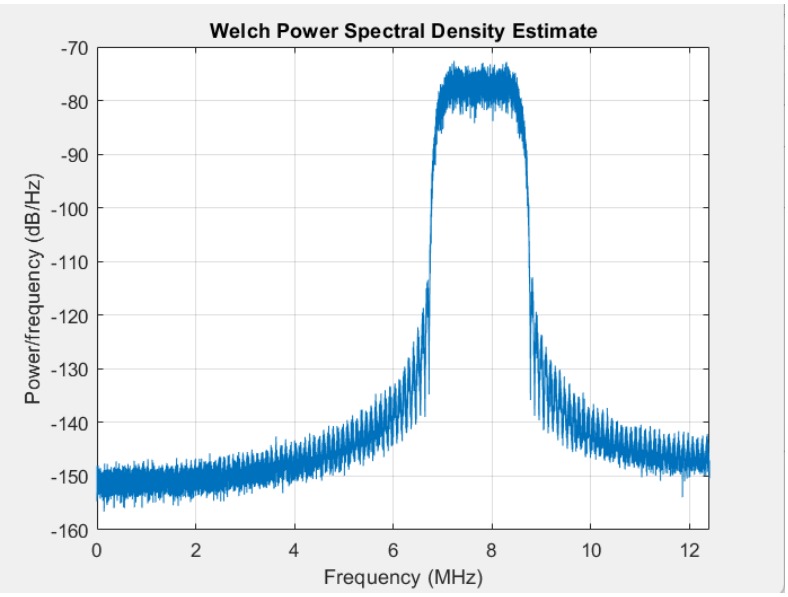
Οπότε, το Baud Rate $\frac{1}{T} = \frac{R}{\log_2 M} = 1.55\ MHz$

Θεωρούμε $f_c = \frac{5}{T}$, $F_s = \frac{16}{T} = 24.8\ MHz$ ώστε να μην έχουμε aliasing. Για να μην έχουμε aliasing, πρέπει $F_s > 2f_{max}$ ή $F_s > 12\ MHz$.

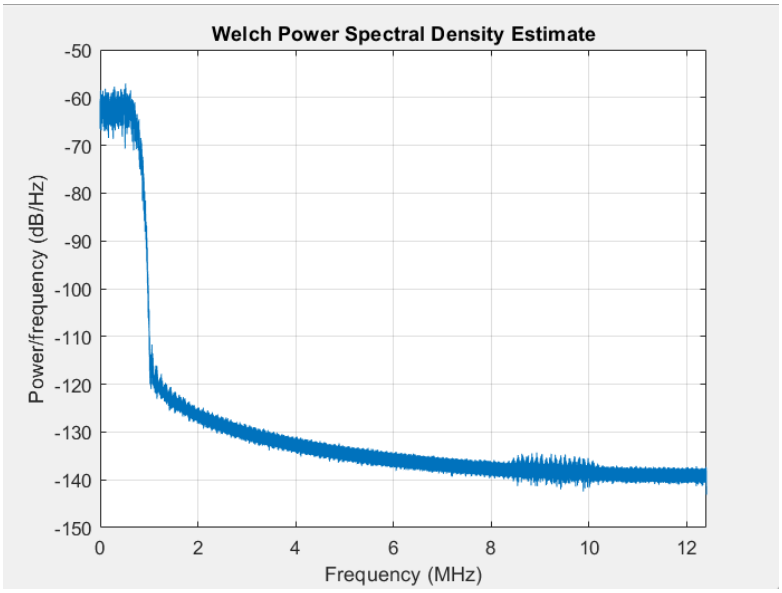
Εκτελούμε την εξομοίωση μέσω του εξομοιωτή BERTool. Οπότε, λαμβάνουμε την παρακάτω εξομοίωση:



Παρακάτω, απεικονίζεται η πυκνότητα φάσματος ισχύος του εκπεμπόμενου σήματος:



Παρακάτω, απεικονίζεται η πυκνότητα φάσματος ισχύος του λαμβανόμενου σήματος:



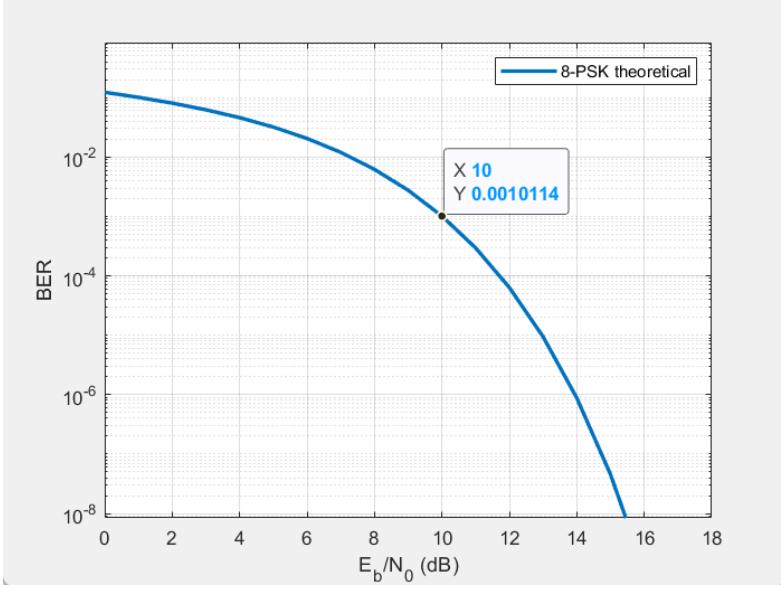
Ο κώδικας που χρησιμοποιήθηκε για το συγκεκριμένο υποερώτημα είναι το παρακάτω:

```
%Eirini Donti 03119839
function errors=ask_Nyq_psk(k,Nsymb,nsamp,EbNo,rolloff)
%k=4; Nsymb=10000; nsamp=16; EbNo=10;
rolloff=0.29;
L=2^k;
%% Gray Mapping
ph1=[pi/4];
theta=[ph1; -ph1; pi-ph1; -pi+ph1];
mapping=exp(1j*theta);
if(k>2)
 for j=3:k
 theta=theta/2;
 mapping=exp(1j*theta);
 mapping=[mapping; -conj(mapping)];
 theta=angle(mapping);
 end
end
%% Convert Random Bits to Symbols
x=floor(2*rand(k*Nsymb,1));
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb')';
y=[];
for i=1:length(xsym)
 y=[y mapping(xsym(i)+1)];
end
%% Filter Creation
delay=8;
filtorder = delay*nsamp*2;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
%% Transmitter
y=upsample(y,nsamp);
ytx = conv(y,rNyquist);
R=6200000; % 6.2 Mbps % R=4575000
Fs=R/k*nsamp;
fc=5;  % fc/1/T
m=(1:length(ytx));
s=real(ytx.*exp(1j*2*pi*fc*m/nsamp));
%figure(1); pwelch(s,[],[],[],Fs);
%% Noise
SNR=EbNo-10*log10(nsamp/2/k);
Ps=10*log10(s*s'/length(s)); % dB
Pn=Ps-SNR; % dB
n=sqrt(10^(Pn/10))*randn(1,length(ytx));
snoisy=s+n;
clear ytx xsym s n;
%% Receiver
yrx=2*snoisy.*exp(-1j*2*pi*fc*m/nsamp); clear s;
yrx = conv(yrx,rNyquist);
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay);
%figure(2); pwelch(real(yrx),[],[],[],Fs);
yr = downsample(yrx,nsamp);
%% Calculate Errors
xr=[];
q=[0:1:L-1];
for n=1:length(yr)
 [m,j]=min(abs(theta-angle(yr(n))));
 yr(n)=q(j);
 xr=[xr; de2bi(q(j),k,'left-msb')'];
end
errors=sum(not(x==xr));
```

Για την εξομοίωση μέσω BERTool, κάναμε χρήση της συνάρτησης ask_ber_func_psk() που παρουσιάζεται παρακάτω:

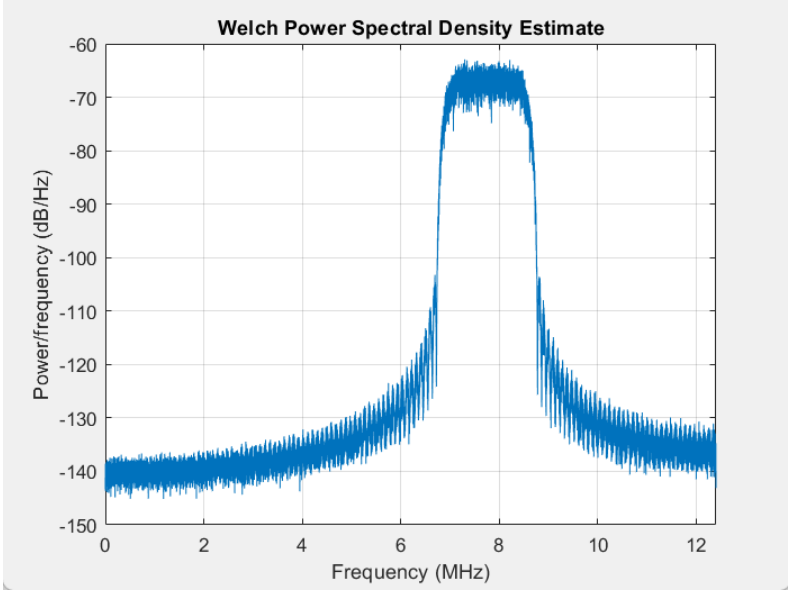
```
%Eirini Donti 03119839
function [ber,numBits] = ask_ber_func_psk(EbNo, maxNumErrs, maxNumBits,varargin)
% Import Java class for BERTool.
import com.mathworks.toolbox.comm.BERTool.*;
% Initialize variables related to exit criteria.
totErr = 0; % Number of errors observed
numBits = 0; % Number of bits processed
% A. --- Set up parameters. ---
% --- INSERT YOUR CODE HERE.
k=4; % number of bits per symbol
Nsymb=10000; % number of symbols in each run
nsamp=16; % oversampling,i.e. number of samples per T
% Simulate until number of errors exceeds maxNumErrs
% or number of bits processed exceeds maxNumBits.
while((totErr < maxNumErrs) && (numBits < maxNumBits))
 % Check if the user clicked the Stop button of BERTool.
 if (isBERToolSimulationStopped(varargin{:}))%BERTool.getSimulationStop)
 break;
 end
 % B. --- INSERT YOUR CODE HERE.
 %errors=ask_errors(k,Nsymb,nsamp,EbNo);
 errors=ask_Nyq_psk(k,Nsymb,nsamp,EbNo);
 % Assume Gray coding: 1 symbol error ==> 1 bit error
 totErr=totErr+errors;
 numBits=numBits + k*Nsymb;
end % End of loop
% Compute the BER
ber = totErr/numBits;
```

3. Σχεδιάζουμε, μέσω της εξομοίωσης BERTool, τη θεωρητική καμπύλη για 8-PSK και λαμβάνουμε την παρακάτω θεωρητική προσομοίωση:

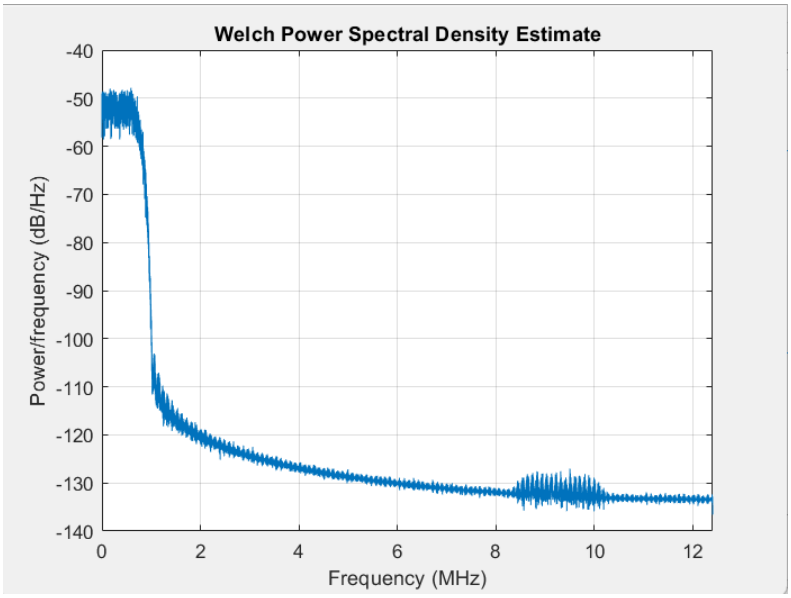


Παρατηρούμε ότι για EbNo=10 dB, όπως αναφέρεται ρητά στην εκφώνηση, ο κωδικοποιητής διαύλου απαιτεί πιθανότητα εσφαλμένου bit 10^{-3}. Ο μέγιστος ρυθμός μετάδοσης $R_{max} = \frac{1}{T} log_2 M$ = 1.55*3 = 4.65 Mbps.

Παρακάτω, απεικονίζεται η πυκνότητα φάσματος ισχύος του εκπεμπόμενου σήματος:



Παρακάτω, απεικονίζεται η πυκνότητα φάσματος ισχύος του λαμβανόμενου σήματος:

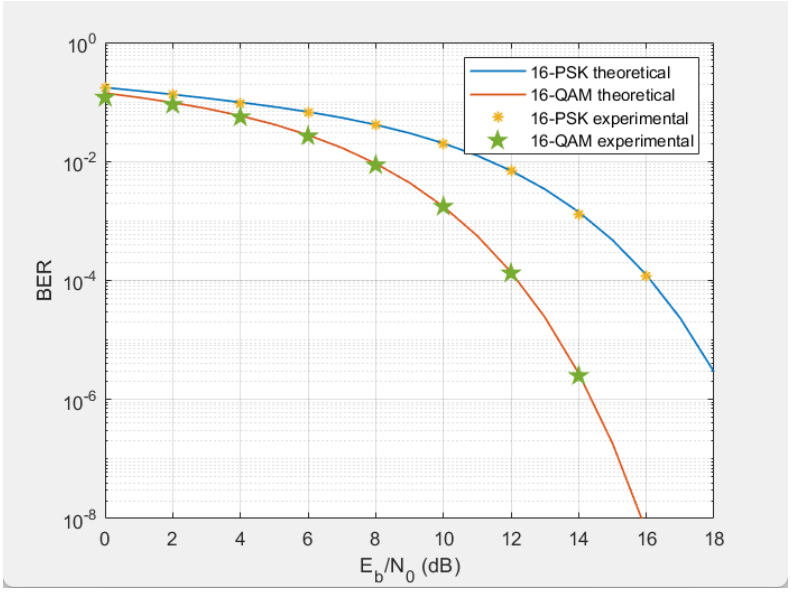


Παρατηρούμε ότι η πυκνότητα φάσματος ισχύος του εκπεμπόμενου και λαμβανόμενου σήματος είναι ίδια με πριν, αφού εξαρτάται από τις παραμέτρους α, Τ των οποίων η τιμή δεν μεταβλήθηκε.

4. Με παράμετρο α’ = 0.29*0.5 = 0.145, υπολογίζουμε ότι R’ $= \frac{1}{T_{'}} log_2 M' =$ $\frac{W}{1+\alpha'} log_2 M' = \frac{2}{1+0.145} 3 = 5.24\ Mbps$. Οπότε, η μείωση της τιμής α θα επιφέρει την αύξηση του λόγου $\frac{1}{T_{'}}$ και συνεπώς θα αυξηθεί η τιμή R’.

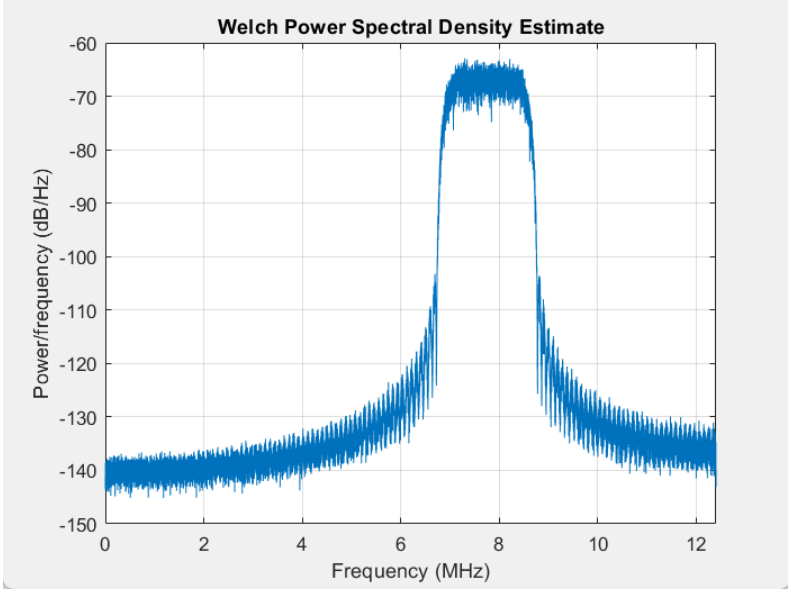
   Οπότε, ο ρυθμός μετάδοσης πρέπει να αυξηθεί κατά ΔR = 5.24-4.575 = 0.665 Mbps σε σχέση με το τρίτο ερώτημα.

5. (α) Εξομοιώνουμε το σύστημα QAM, με τις ίδιες παραμέτρους από το σύστημα PSK του ερωτήματος 2.



   Το QAM σύστημα έχει καλύτερο BER από εκείνου του PSK. Αυτό είναι λογικό, καθώς στο σύστημα QAM, τα σύμβολα απέχουν περισσότερο μεταξύ τους σε σχέση με το PSK σύστημα. Οπότε, η πιθανότητα λάθους στο σύστημα QAM είναι μικρότερη από το σύστημα PSK.

(β) Παρακάτω, φαίνεται η πυκνότητα φάσματος ισχύος του εκπεμπόμενου σήματος:



Παρατηρούμε ότι και στα δύο συστήματα PAM και QAM χρειάζονται το ίδιο εύρος ζώνης W. Αυτό είναι λογικό, καθώς οι τιμές, που εξαρτάται το εύρος ζώνης W, α και T δεν μεταβάλλονται.

Ο κώδικας που χρησιμοποιήθηκε για το συγκεκριμένο υποερώτημα είναι το παρακάτω:

```
%Eirini Donti 03119839
function errors=ask_Nyq_qam(M,Nsymb,nsamp,EbNo,rolloff)
%M=16; Nsymb=10000; nsamp=16; EbNo=10;
rolloff=0.29;
L=sqrt(M);l=log2(L); k=log2(M);
%% Gray Mapping
core=[1+1j;1-1j;-1+1j;-1-1j];
mapping=core;
if(l>1)
 for j=1:l-1
 mapping=mapping+j*2*core(1);
 mapping=[mapping;conj(mapping)];
 mapping=[mapping;-conj(mapping)];
 end
end;

%% Convert Random Bits to Symbols
x=floor(2*rand(k*Nsymb,1));
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb')';
y=[];
for i=1:length(xsym)
 y=[y mapping(xsym(i)+1)];
end

%% Filter Creation
delay=8;
filtorder = delay*nsamp*2;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
%% Transmitter
ytx=upsample(y,nsamp);
ytx = conv(ytx,rNyquist);
R=6200000;
Fs=R/k*nsamp;
fc=5;  %carrier frequency / baud rate
m=(1:length(ytx));
s=real(ytx.*exp(1j*2*pi*fc*m/nsamp));  % shift to desired frequency band
```

```
%figure(1); pwelch(s,[],[],[],Fs);

%% Noise
SNR=EbNo-10*log10(nsamp/2/k);
Ps=10*log10(s*s'/length(s)); % dB
Pn=Ps-SNR; % dB
n=sqrt(10^(Pn/10))*randn(1,length(ytx));
snoisy=s+n;
clear ytx xsym s n;

%% Receiver
yrx=2*snoisy.*exp(-1j*2*pi*fc*m/nsamp); clear s;
yrx = conv(yrx,rNyquist); %filter
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay);
% figure(2); pwelch(real(yrx),[],[],[],Fs);
yrx = downsample(yrx,nsamp);

%% Calculate Errors
yi=real(yrx); yq=imag(yrx);
xrx=[];
q=[-L+1:2:L-1];
for n=1:length(yrx)
    [m,j]=min(abs(q-yi(n)));
    yi(n)=q(j);
    [m,j]=min(abs(q-yq(n)));
    yq(n)=q(j);
end
errors=sum(not(y==(yi+1j*yq)));
```

Για την εξομοίωση μέσω BERTool, κάναμε χρήση της συνάρτησης ask_ber_func_qam() που παρουσιάζεται παρακάτω:

```
%Eirini Donti 03119839
function [ber,numBits] = ask_ber_func_qam(EbNo, maxNumErrs, maxNumBits,varargin)
% Import Java class for BERTool.
import com.mathworks.toolbox.comm.BERTool.*;
% Initialize variables related to exit criteria.
totErr = 0; % Number of errors observed
numBits = 0; % Number of bits processed
% A. --- Set up parameters. ---
% --- INSERT YOUR CODE HERE.
M=16; % M-QAM
L=sqrt(M);l=log2(L); k=log2(M);
Nsymb=10000; % number of symbols in each run
nsamp=16; % oversampling,i.e. number of samples per T
rolloff=0.29; % rolloff for raised cosine filter
% Simulate until number of errors exceeds maxNumErrs
% or number of bits processed exceeds maxNumBits.
while((totErr < maxNumErrs) && (numBits < maxNumBits))
% Check if the user clicked the Stop button of BERTool.
if(isBERToolSimulationStopped(varargin{:}))
break;
end
% B. --- INSERT YOUR CODE HERE.
errors=ask_Nyq_qam(M,Nsymb,nsamp,EbNo,rolloff);
% Assume Gray coding: 1 symbol error ==> 1 bit error
totErr=totErr+errors;
numBits=numBits + k*Nsymb;
end % End of loop
% Compute the BER
ber = totErr/numBits;
```