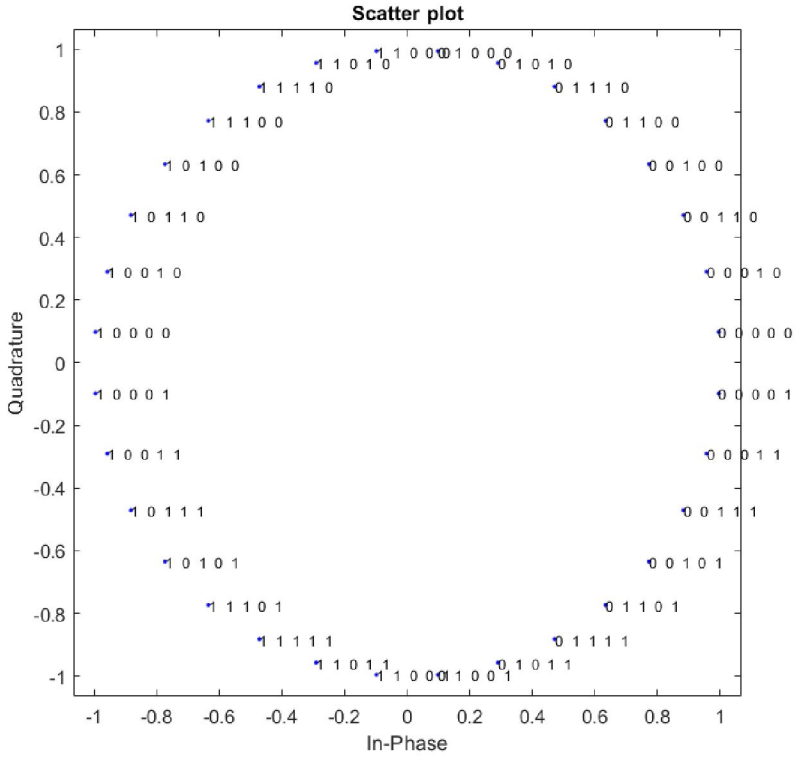
ΗΜΜΥ ΕΜΠ

6ο Εξάμηνο

Ψηφιακές Επικοινωνίες Ι

Lab 5

1. Φτιάχνουμε τη mapping όπως τον αλγόριθμο των σημειώσεων: σε κάθε βήμα υποδιπλασιάζουμε τη γωνία και μετά καθρεφτίζουμε γύρω από τον φανταστικό άξονα.



Κώδικας για άσκηση 1: Αρχείο *psk_scatterplot.m* (συνάρτηση με παράμετρο k)

```
function psk_scatterplot(k)
% k is the number of bits per symbol
M=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:M-1].',k,'left-
msb')), 'FontSize', 6);
end
```

2. Έχουμε εύρος ζώνης $W = 7 - 5 = 2\ MHz$ και ρυθμό μετάδοσης $R = 6Mbps$

Από τη σχέση $\log_2 M \geq \frac{R}{W}(1+a) \Rightarrow \log_2 M \geq 3(1+a)$, με $0 < a < 1$ βρίσκουμε $M = 16$ η μικρότερη τιμή που την ικανοποιεί.

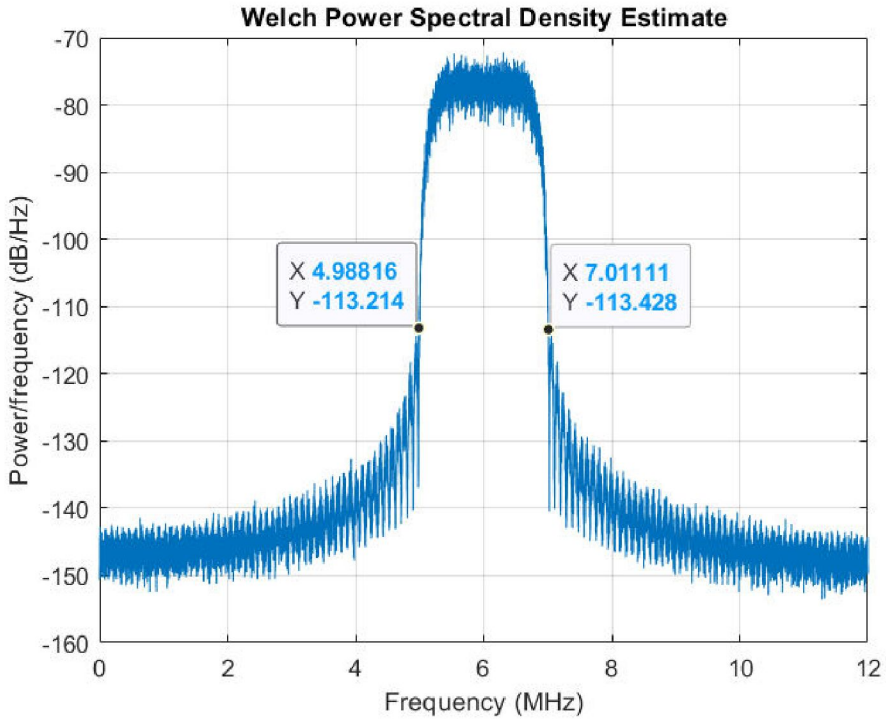
Επιλέγουμε το μεγαλύτερο α για να εκμεταλλευτούμε όλο το εύρος ζώνης:

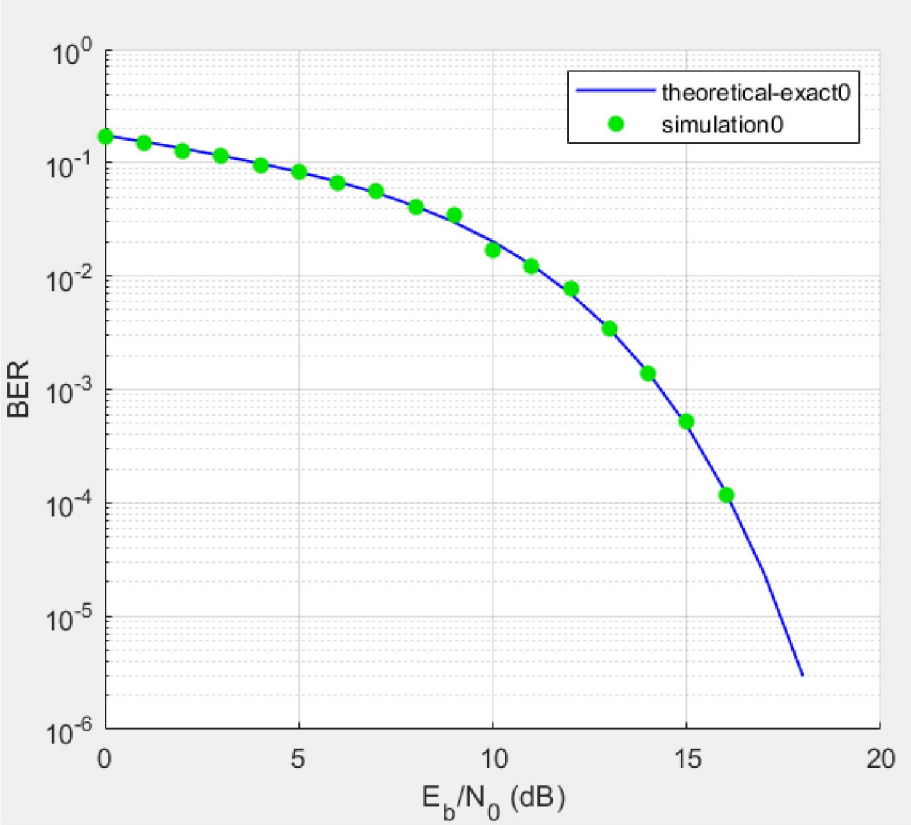
$$(1+a) \leq \frac{\log_2 M}{3} \quad \Rightarrow \quad a \leq 0.33$$

Έτσι προκύπτει baud rate $\frac{1}{T} = 1.5\ MHz$

$f_c = 6\ MHz = \frac{4}{T}$ και χρησιμοποιούμε nsamp=16 άρα $F_s = \frac{16}{T} = 24\ MHz$, αρκετό για να μην έχουμε aliasing (πρέπει να είναι μεγαλύτερο από 14MHz, δηλαδή 2 φορές το άνω όριο του εύρους ζώνης).

Για την εξομοίωση του συστήματος καλούμε το bertool με το αρχείο *ask_ber_func_psk_nyq.m*, το οποίο καλεί τη συνάρτηση *ask_nyq_psk.m* με παραμέτρους k, Nsymb, nsamp, EbNo, rolloff. Ρυθμίζουμε τις παραμέτρους k=4, Nsymb=2000, nsamp=16, rolloff=0.33. Συνίσταται επίσης να μπουν σε comment οι γραμμές 38,53 της *ask_nyq_psk.m* πριν τη χρήση bertool, καθώς ο σχεδιασμός των γραφικών παραστάσεων σε κάθε προσομοίωση καθυστερεί αρκετά τη διαδικασία.



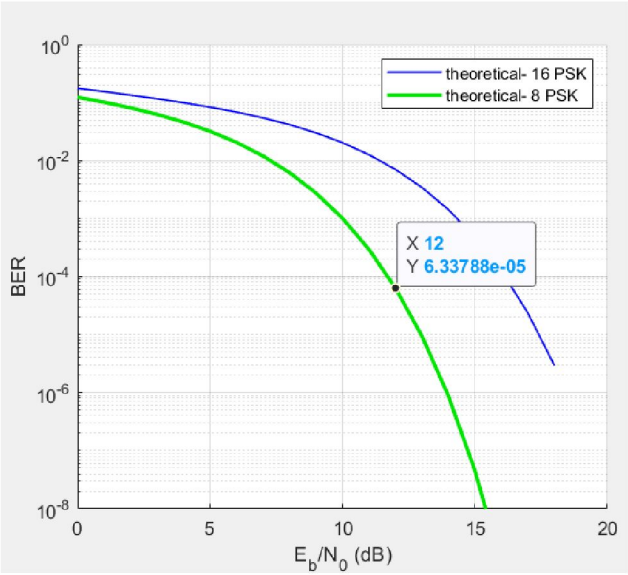


Κώδικας για άσκηση 2: συνάρτηση *ask_nyq_psk.m*, καλείται από bertool από την *ask_ber_func_psk_nyq.m*

```
function errors=ask_nyq_psk(k,Nsymb,nsamp,EbNo,rolloff)
%k=4; Nsymb=10000; nsamp=16; rolloff=0.33;
L=2^k;

%% Grey encoding vector
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

%% Random bits -> symbols
x=floor(2*rand(k*Nsymb,1));
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb')';
```

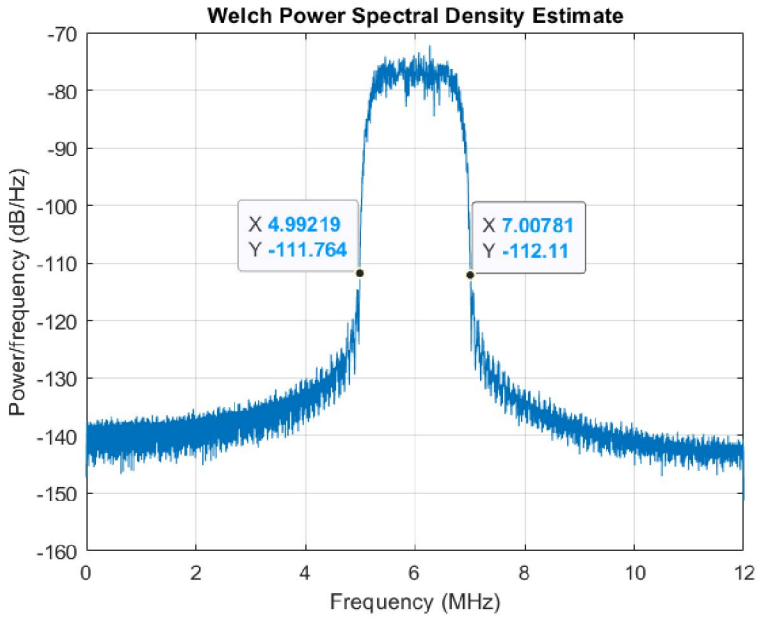
```
y=[];
for i=1:length(xsym)
 y=[y mapping(xsym(i)+1)];
end

%% Filter parametres
delay=8;
filtorder = delay*nsamp*2;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);

%% Transmitter
y=upsample(y,nsamp);
ytx = conv(y,rNyquist);
R=6000000;
Fs=R/k*nsamp;
fc=4;  %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); % COMMENT FOR BERTOOL

%% Noise
SNR=EbNo-10*log10(nsamp/2/k);
Ps=10*log10(s*s'/length(s)); %signal power (db)
Pn=Ps-SNR; %noise power (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; %shift to 0 frequency
yrx = conv(yrx,rNyquist); %filter
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay);
%figure(2); pwelch(real(yrx),[],[],[],Fs); % COMMENT FOR BERTOOL
yr = downsample(yrx,nsamp);

%% Error counting
xr=[];
q=[0:1:L-1];
for n=1:length(yr)
 [m,j]=min(abs(theta-angle(yr(n))));    % compare angle to theta
 yr(n)=q(j);
 xr=[xr; de2bi(q(j),k,'left-msb')'];
end
errors=sum(not(x==xr));
```

3. Σχεδιάζοντας με το bertool την θεωρητική καμπύλη για 8-PSK παρατηρούμε ότι για EbNo=12 η πιθανότητα εσφαλμένου bit είναι οριακά αποδεκτή.



Ο μέγιστος ρυθμός μετάδοσης είναι τώρα $R' = \frac{1}{T}\log_2 M' = 4.5\ Mbps$.

Η πυκνότητα φάσματος ισχύος του εκπεμπόμενου σήματος είναι ίδια με πριν αφού εξαρτάται από τις παραμέτρους a, T οι οποίες δεν άλλαξαν.



4. Με $a' = 0.17$ έχουμε $R'' = \frac{1}{T''}\log_2 M' = \frac{W}{1+a''}k = \frac{2}{1.17}3 = 5.19\ Mbps$. Για να πάρουμε το ίδιο εύρος ζώνης με μικρότερο α πρέπει να αυξήσουμε το $1/T$, άρα και το R.

Κώδικας για άσκηση 3: συνάρτηση *ask_nyq_psk_34.m* (ίδια με την *ask_nyq_psk.m* από το μέρος 2 αλλά έχει επιπλέον παράμετρο το R)

Καλείται ως εξής:

```
>> ask_nyq_psk_34(3,2000,16,12,0.33,4500000)
```

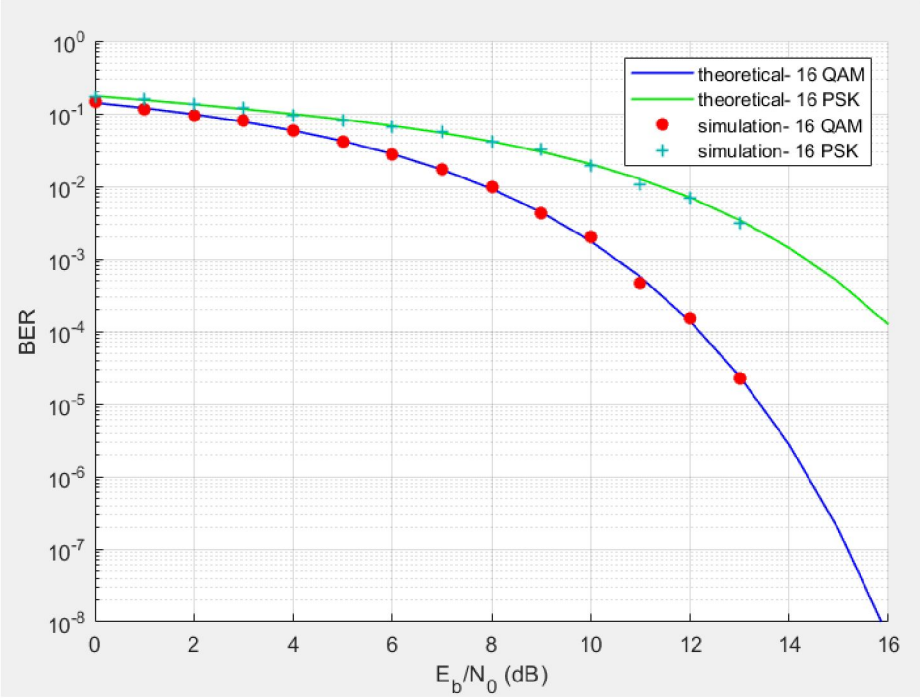
Ο κώδικας:

```
function errors=ask_nyq_psk_34(k,Nsymb,nsamp,EbNo,rolloff,R)
% k=3; Nsymb=2000%; nsamp=16; EbNo=14; rolloff=0.33; R=4500000;
L=2^k;

%% Grey encoding vector
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=log(mapping)/1j;
 end
end

%% Random bits -> symbols
x=floor(2*rand(k*Nsymb,1));
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb')';
y1=[];
for i=1:length(xsym)
 y1=[y1 mapping(xsym(i)+1)];
end

%% Filter parametres
delay=8;
filtorder = delay*nsamp*2;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);

%% Transmitter
y=upsample(y1,nsamp);
ytx = conv(y,rNyquist);
Fs=R/k*nsamp;
fc=4;  %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); % COMMENT FOR BERTOOL

%% Noise
SNR=EbNo-10*log10(nsamp/2/k);
```

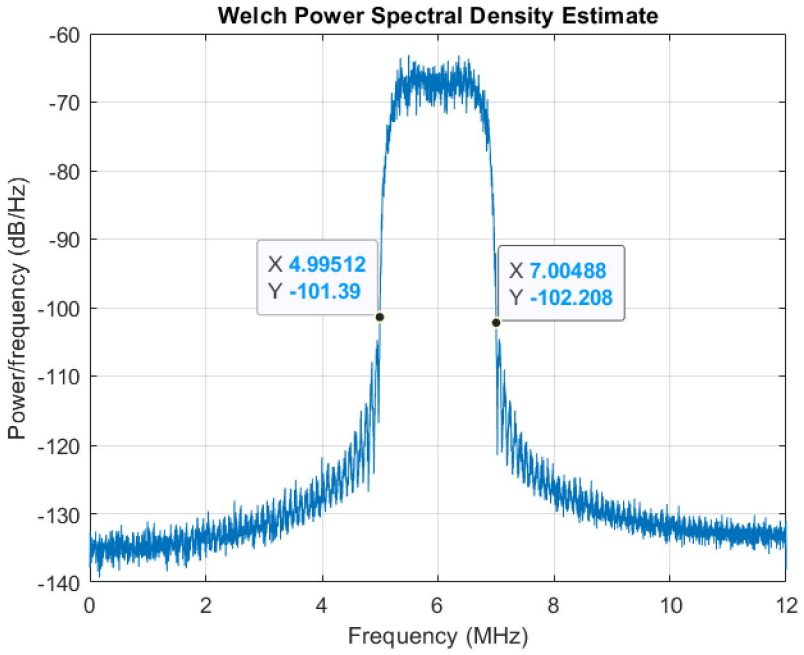
```
Ps=10*log10(s*s'/length(s)); %signal power (db)
Pn=Ps-SNR; %noise power (db)
n=sqrt(10^(Pn/10))*randn(1,length(ytx));
snoisy=s+n;
clear ytx xsym s n;

%% Receiver
yrx=2*snoisy.*exp(-1j*2*pi*fc*m/nsamp); clear s; %shift to 0 frequency
yrx = conv(yrx,rNyquist); %filter
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay);
figure(2); pwelch(real(yrx),[],[],[],Fs); % COMMENT FOR BERTOOL
yr = downsample(yrx,nsamp);

%% Error counting
xr=[];
q=[0:1:L-1];
for n=1:length(yr)
 [m,j]=min(abs(theta-angle(yr(n))));    % compare angle to theta
 yr(n)=q(j);
 xr=[xr; de2bi(q(j),k,'left-msb')'];
end
errors=sum(not(x==xr));
```

5. Συγκρίνουμε το 16-QAM με το 16-PSK.

(α) Παρατηρούμε ότι το QAM σύστημα έχει καλύτερο BER, επειδή τα σύμβολα απέχουν περισσότερο μεταξύ τους απ'ότι στο PSK σύστημα οπότε τα λάθη είναι λιγότερα.



(β) Πετυχαίνουμε ίδιο εύρος ζώνης όσο δίνεται, λογικό αφού δεν αλλάζουν οι παράμετροι a, T.



Κώδικας για άσκηση 5: συνάρτηση *ask_nyq_qam.m* με παραμέτρους M, Nsymb, nsamp, EbNo, rolloff.

Για να δούμε το εύρος ζώνης εκτελούμε στο command line:

```
>> ask_nyq_qam(16,2000,16,12,0.33);
```

Καλούμε το bertool, παράγουμε τις θεωρητικές καμπύλες, και για τις προσομοιώσεις καλούμε τα αρχεία *ask_ber_func_qam_nyq.m* και *ask_ber_func_psk_nyq.m,* τα οποία καλούν αντίστοιχα τις *ask_nyq_qam.m*, *ask_nyq_psk.m* με τις κατάλληλες παραμέτρους. Συνίσταται πάλι να μπουν σε σχόλιο οι γραμμές 36, 51 στο *ask_nyq_qam.m* και η γραμμή 38 στο *ask_nyq_psk.m* για να μην σχεδιάζονται οι γραφικές παραστάσεις του εύρους ζώνης σε κάθε προσομοίωση.

Συνάρτηση *ask_nyq_qam.m*

```
function errors=ask_nyq_qam(M,Nsymb,nsamp,EbNo,rolloff)
% M=16; Nsymb=2000%; nsamp=16; EbNo=12; rolloff=0.33;
L=sqrt(M); l=log2(L); k=2*l;

%% Grey encoding vector
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;

%% Random bits -> 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 parametres
delay=8;
filtorder = delay*nsamp*2;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);

%% Transmitter
ytx=upsample(y,nsamp);
ytx = conv(ytx,rNyquist);
R=6000000;
Fs=R/k*nsamp;
fc=4;  %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); % COMMENT FOR BERTOOL

%% Noise
SNR=EbNo-10*log10(nsamp/2/k);
Ps=10*log10(s*s'/length(s)); %signal power (db)
Pn=Ps-SNR; %noise power (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; %shift to 0 frequency
yrx = conv(yrx,rNyquist); %filter
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay);
%figure(2); pwelch(real(yrx),[],[],[],Fs); % COMMENT FOR BERTOOL
yrx = downsample(yrx,nsamp);

%% Error counting
yi=real(yrx); yq=imag(yrx);
xrx=[];
q=[-L+1:2:L-1];
for n=1:length(yrx) % compare in phase and quadrature components
    [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)));
```