ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αναφορά 6ης Εργαστηριακής Άσκησης στις Ψηφιακές Επικοινωνίες Ι

«FSK-MSK»

Μπουφίδης Ιωάννης 03120162

Μέρος 1

Συμπληρώνοντας τον κώδικα 6.1 με σκοπό να εξομοιώνει τόσο σύμφωνη, όσο και ασύμφωνη FSK προκύπτουν δύο συναρτήσεις MATLAB, η fsk_coherent_errors και η fsk_non_coherent_errors αντίστοιχα. Ο κώδικας των δύο συναρτήσεων διαφέρει μόνο στον τρόπο με τον οποίο αποδιαμορφώνεται το σήμα στον δέκτη και στο πως υπολογίζεται η απόσταση μεταξύ των διαδοχικών συχνοτήτων. Ακολουθεί κοινή υλοποίηση για τις 2 συναρτήσεις με πλαισιωμένα τα παραπάνω σημεία, στα οποία διαφέρουν.

```
function errors=fsk_coherent_errors(bps,Nsymb,ns,EbNo) % comment for fsk_non_coherent_errors
function errors=fsk_non_coherent_errors(bps,Nsymb,ns,EbNo) % comment for fsk_coherent_errors
```

```
%% Input parameters
% bps: bits per symbol, Nsymb: numb of simulated symbols
% ns: number of samples per symbol (oversampling)
% EbNo: normalized signal-to-noise ratio, in db
M=2^bps; % number of different symbols
BR=1; % Baud Rate
fc=2*M*BR; % RF frequency
%% Derived parameters
nb=bps*Nsymb; % number of simulated data bits
T=1/BR; % one symbol period
Ts=T/ns; % oversampling period
```

```
% M frequencies in "coherent" distance (BR)
f=fc+BR/2*((1:M)-(M+1)/2); % comment for fsk_non_coherent_errors
% M frequencies in "non-coherent" distance (BR)
f=fc+BR*((1:M)-(M+1)/2); % comment for fsk_coherent_errors
```

```
% awgn channel
SNR=EbNo+10*log10(bps)-10*log10(ns/2); % in db
% input data bits
y=randi([0,1],1,nb); %
x=reshape(y,bps,length(y)/bps)';
t=[0:T:length(x(:,1))*T]'; % time vector on the T grid
tks=[0:Ts:T-Ts]';
%% FSK signal
s=[];
A=sqrt(2/T/ns);
for k=1:length(x(:,1))
 fk=f(bi2de(x(k,:))+1);
 tk=(k-1)*T+tks;
 s=[s; sin(2*pi*fk*tk)];
end
% add noise to the FSK (passband) signal
s=awgn(s,SNR, 'measured');
```

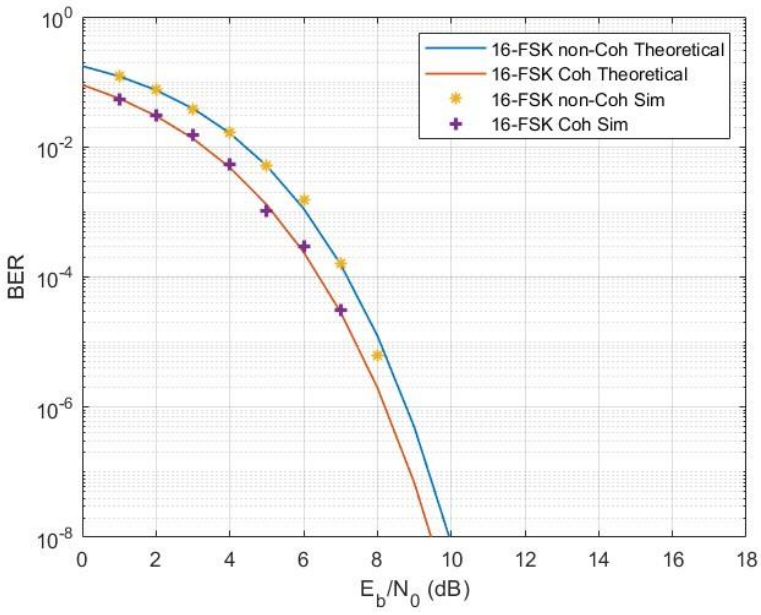
```
%% FSK receiver
```

```
% coherent demodulation
% comment for fsk_non_coherent_errors
th=0;
xr=[];
for k=1:length(s)/ns
    tk=(k-1)*T+tks;
    sk=s((k-1)*ns+1:k*ns);
    smi=[];
    for i=1:M
        si=sin(2*pi*f(i)*tk);
        smi(i)=sum(sk.*si);
    end
    [m,j]=max(smi);
    xr=[xr;de2bi(j-1,bps)];
end
% non-coherent demodulation
% comment for fsk_coherent_errors
xr=[];
for k=1:length(s)/ns
  tk=(k-1)*T+tks;
  sk=s((k-1)*ns+1:k*ns);
  smi=[];
  for i=1:M
    th=rand()*pi;
    si=sin(2*pi*(f(i)*tk+th));
    sq=cos(2*pi*(f(i)*tk+th));
    smi=sum(sk.*si);
    smq=sum(sk.*sq);
    sm(i)=sqrt(smi^2+smq^2);
  end
  [m,j]=max(sm);
  xr=[xr;de2bi(j-1,bps)];
end
```

```
% count errors
err=not(x==xr);
errors=sum(sum(err));
end
```

Μέρος 2

Μέσω του bertool, σχεδιάζουμε τις διπλανές καμπύλες Pb-EbNo, τόσο για σύμφωνη, όσο και για ασύμφωνη 16-FSK. Συγκεκριμένα, πέρα από τις δύο θεωρητικές καμπύλες, παραθέτονται και σημεία τα οποία προέκυψαν από εξομοιώσεις. Αυτές οι δύο εξομοιώσεις πραγματοποιήθηκαν στο bertool, μέσω της συνάρτησης ask_ber_func, η οποία με την σειρά της θέτει τα κατάλληλα χαρακτηριστικά για το σύστημα και υπολογίζει την πιθανότητα λάθους για διακριτές τιμές του EbNo χρησιμοποιώντας τις δύο παραπάνω συναρτήσεις.

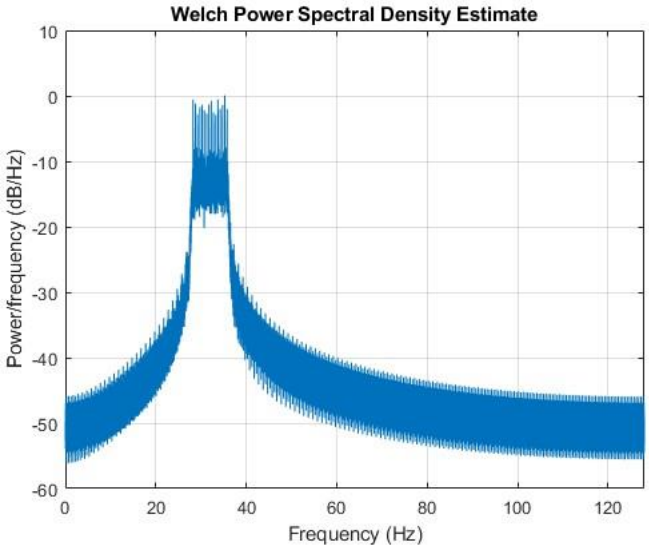


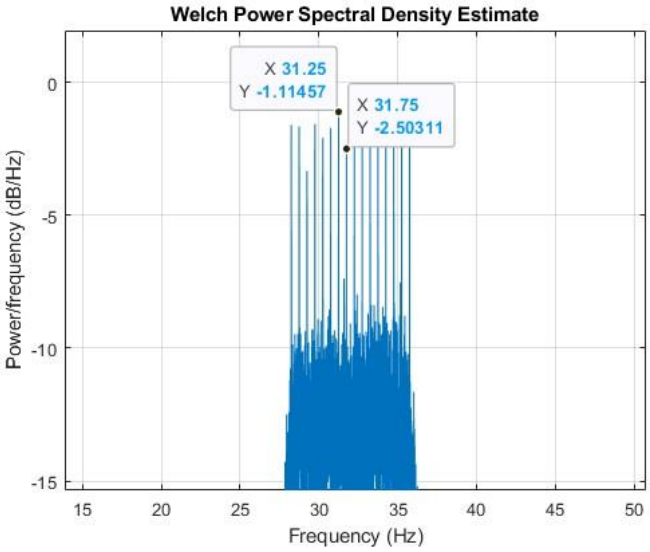
Παρατηρούμε ότι η καμπύλη της σύμφωνης 16-FSK βρίσκεται «κάτω» από αυτή της ασύμφωνης 16-FSK, πράγμα που σημαίνει ότι για τον ίδιο σηματοθορυβικό λόγο EbNo η πρώτη είναι πιο αποδοτική όσο αφορά την πιθανότητα λάθους. Αυτό συμβαίνει, καθώς στην σύμφωνη FSK ο δέκτης έχει γνώση της ακριβούς φάσης και συχνότητας των διαφορετικών συμβόλων FSK, σε αντίθεση με την ασύμφωνη FSK. Κάτι τέτοιο εχεί ως αποτέλεσμα, η πιθανότητα λάθους αναγνώρισης συμβόλου από τον δέκτη να είναι μικρότερη.

Μέρος 3

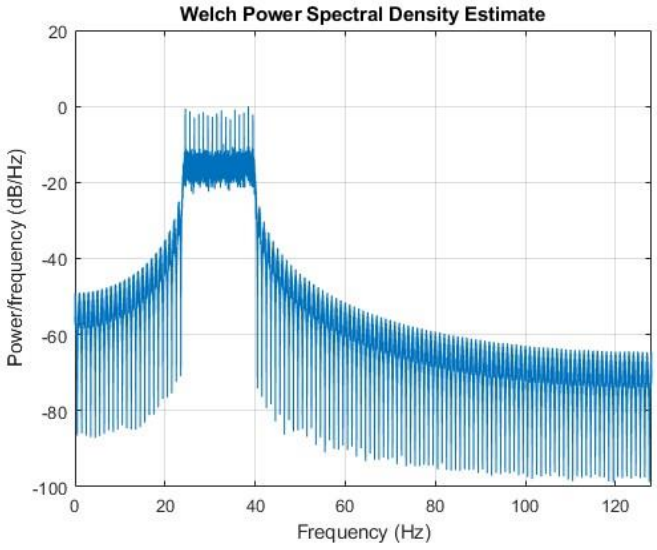
Παρακάτω φαίνεται το φάσμα του ζωνοπερατού σήματος τόσο για σύμφωνη, όσο και για ασύμφωνη 16-FSK (ερώτημα 2). Παρατηρούμε ότι στην σύμφωνη οι 16 συχνότητες απέχουν μεταξύ τους 1/2T=0.5Hz, ενώ στην ασύμφωνη απέχουν 1/T=1Hz. Έτσι, από την στιγμή που και στις δύο περιπτώσεις το συνολικό έυρος ζώνης είναι 17*(απόσταση μεταξύ διαδοχικών συχνοτήτων) γύρω από τα $fc = 2 * M * (1/T) = 32Hz$, αντιλαμβανόμαστε ότι με την σύμφωνη πετυχαίνουμε το μισό εύρος ζώνης.

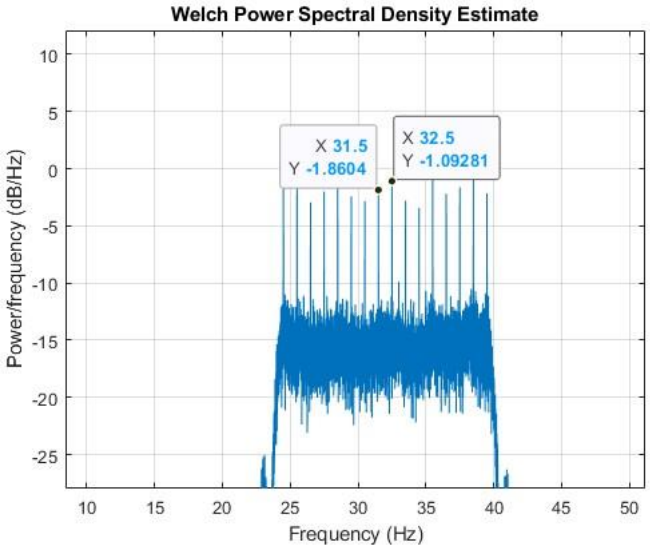
πυκνότητα φάσματος σύμφωνης 16-FSK





πυκνότητα φάσματος ασύμφωνης 16-FSK





Ο κώδικας ο οποίος χρησιμοποιήθηκε για την δημιουργία αυτών των σχημάτων είναι ο εξής:

```
close all; clear all; clc;
bps=4; Nsymb=2000; ns=256;

%% Input parameters
% bps: bits per symbol, Nsymb: numb of simulated symbols
% ns: number of samples per symbol (oversampling)
% EbNo: normalized signal-to-noise ratio, in db
M=2^bps; % number of different symbols
BR=1; % Baud Rate
fc=2*M*BR; % RF frequency

%% Derived parameters
nb=bps*Nsymb; % number of simulated data bits
T=1/BR; % one symbol period
Ts=T/ns; % oversampling period
% M frequencies in "coherent" distance (BR)
f_coh=fc+BR/2*((1:M)-(M+1)/2);
% M frequencies in "non-coherent" distance (BR)
f_ncoh=fc+BR*((1:M)-(M+1)/2);

% input data bits
y=randi([0,1],1,nb); %
x=reshape(y,bps,length(y)/bps)';
t=[0:T:length(x(:,1))*T]'; % time vector on the T grid
tks=[0:Ts:T-Ts]';

%% FSK signal
s_coh=[];
s_ncoh=[];
A=sqrt(2/T/ns);

% coherent signal
for k=1:length(x(:,1))
    fk_coh=f_coh(bi2de(x(k,:))+1);
    tk=(k-1)*T+tks;
    s_coh=[s_coh; sin(2*pi*fk_coh*tk)];
end
figure(1); pwelch(s_coh, [], [], [], ns); pause

% non-coherent signal
for k=1:length(x(:,1))
 fk_ncoh=f_ncoh(bi2de(x(k,:))+1);
 tk=(k-1)*T+tks;
 s_ncoh=[s_ncoh; sin(2*pi*fk_ncoh*tk)];
end
figure(2); pwelch(s_ncoh, [], [], [], ns);
```

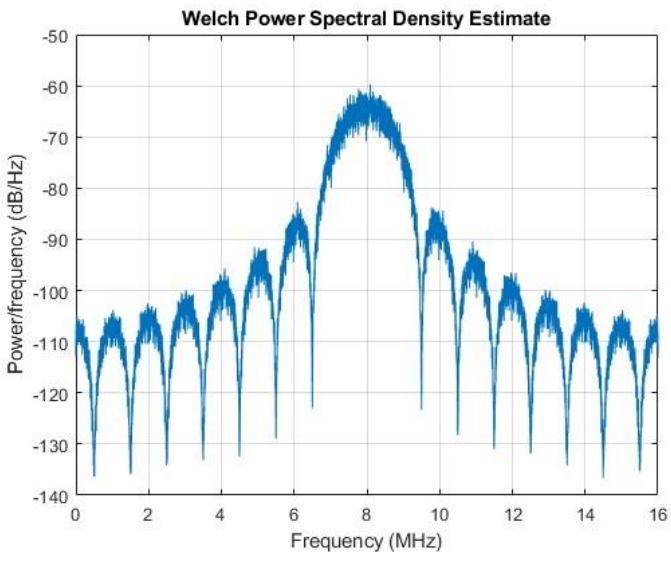
Μέρος 4

Τροποιώντας των κώδικα 6.2, έτσι ώστε να εξομοιωθεί σύστημα MSK μετάδοσης σε ζωνοπερατό δίαυλο με κεντρική συχνότητα 8 MHz και ρυθμό μετάδοσης 2 Mbps, προκύπτει το παρακάτω πρόγραμμα MATLAB:

```
clear all; close all; clc;
% Οριζόμενες παράμετροι
n=4000; % αριθμός data bits
R=2000000; % bit rate
```

```
fc=8/2*R; % φέρουσα συχνότητα
ns=16; % παράγοντας υπερδειγμάτισης

% Παραγόμενες παράμετροι
T=1/R; % περίοδος 1 bit (= βασική περίοδος)
Ts=T/ns; % συχνότητα δειγματοληψίας

% ακολουθία εισόδου
y=[1;sign(rand(n-1,1)-0.5)]; % random numbers, -1 ή 1

% προκωδικοποίηση
x(1)=1;
for i=2:length(y)
    x(i)=y(i)*y(i-1);
end
x=x';
g=ones(ns,1);
x_upsampled = upsample(x,ns);
xx=conv(x_upsampled,g); % δείγματα παλμοσειράς NRZ polar

% χρονικό πλέγμα
ts=[0:Ts:length(xx)*Ts]'; % μήκους ns*(n+1)

%% ΠΟΜΠΟΣ MSK
xs=xx;
theta=cumsum(xs)*pi/2/ns;
xs_i=cos(theta); % συμφασική συνιστώσα
xs_i=[xs_i; xs_i(length(xs_i))]; % επέκταση με ένα ακόμη δείγμα
xs_q=sin(theta); % εγκάρσια συνιστώσα
xs_q=[xs_q; xs_q(length(xs_q))]; % επέκταση με ένα ακόμη δείγμα

% Διαμόρφωση
s=xs_i.*cos(2*pi*fc*ts)-xs_q.*sin(2*pi*fc*ts);
figure(1); pwelch(s, [], [], fc, 1/Ts);
```

Μέσω αυτού του κώδικα σχεδιάζεται το φάσμα του ζωνοπερατού σήματος, το οποίο φαίνεται στο διπλανό σχήμα. Φαίνεται ότι είναι ζωνοπερατό με κεντρική συχνότητα 8MHz. Για να πετύχουμε την εξομοίωση του συστήματος MSK που ζητείται, αλλάξαμε τις εξής παραμέτρους στον κώδικα 6.2:



- `R=2000000; % bit rate`
- `fc=8/2*R; % φέρουσα συχνότητα`

Στην συνέχεια, προκειμένου να υπολογίσουμε το BER (θεωρητικά και με εξομοίωση) όταν EbNo=10dB, επεκτείνουμε τον παραπάνω κώδικα, ώστε να υπάρχουν δύο περιπτώσεις: με προκωδικοποίηση και χωρίς. Έτσι, προκύπτουν δύο νέες συναρτήσεις MATLAB, η msk_precoding_errors και η msk_noprecoding_errors, τις οποίες θα χρησιμοποιήσει η συνάρτηση ask_ber_func, ώστε να εξομοιώσει το δεδομένο σύστημα για τις δύο περιπτώσεις και να υπολογίσει τις αντίστοιχες τιμές του BER για διακριτές τιμές του EbNo μέσω του bertool.

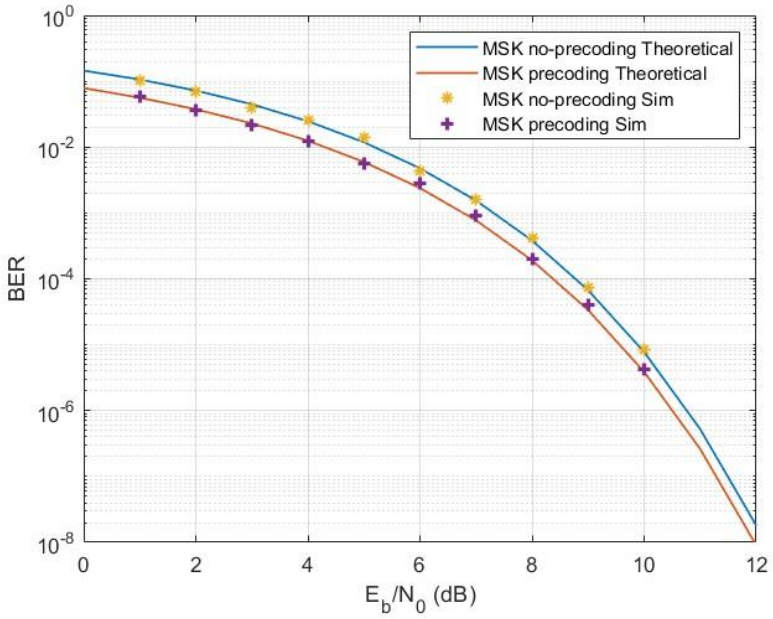
Ακολουθεί κοινή υλοποίηση για τις 2 συναρτήσεις με πλαισιωμένα τα σημεία, στα οποία διαφέρουν.

```
function errors=msk_precoding_errors(Nbits,nsamp,EbNo) % comment for msk_noprecoding_errors
function errors=msk_noprecoding_errors(Nbits,nsamp,EbNo) % comment for msk_precoding_errors
```

```
% Οριζόμενες παράμετροι
n=Nbits; % αριθμός data bits
R=2000000; % bit rate
fc=8/2*R; % φέρουσα συχνότητα
ns=nsamp; % παράγοντας υπερδειγμάτισης
%
% δίαυλος awgn
SNR=EbNo-10*log10(ns/2); % in db
% Παραγόμενες παράμετροι
T=1/R; % περίοδος 1 bit (= βασική περίοδος)
Ts=T/ns; % συχνότητα δειγματοληψίας
% ακολουθία εισόδου
y=[1;sign(rand(n-1,1)-0.5)]; % random numbers, -1 ή 1
%
% προκωδικοποίηση
x(1)=1;
for i=2:length(y)
 x(i)=y(i)*y(i-1);
end
x=x';
g=ones(ns,1);
xx=conv(upsample(x,ns),g); % δείγματα παλμοσειράς NRZ polar
% χρονικό πλέγμα
ts=[0:Ts:length(xx)*Ts]'; % μήκους ns*(n+1)
%
%% ΠΟΜΠΟΣ MSK
xs=xx;
theta=cumsum(xs)*pi/2/ns;
xs_i=cos(theta); % συμφασική συνιστώσα
xs_i=[xs_i; xs_i(length(xs_i))]; % επέκταση με ένα ακόμη δείγμα
xs_q=sin(theta); % εγκάρσια συνιστώσα
xs_q=[xs_q; xs_q(length(xs_q))]; % επέκταση με ένα ακόμη δείγμα
% Διαμόρφωση
s=xs_i.*cos(2*pi*fc*ts)-xs_q.*sin(2*pi*fc*ts);
% προσθήκη θορύβου
s=awgn(s,SNR, 'measured');
%% ΔΕΚΤΗΣ MSK
xs_i=s.*cos(2*pi*fc*ts);
xs_q=-s.*sin(2*pi*fc*ts);
% Φίλτρο LP (Parks-McClellan)
f1=0.75/ns; f2=4*f1;
order=4*ns;
fpts=[0 f1 f2 1];
mag=[1 1 0 0];
wt=[1 1];
b = firpm(order,fpts,mag,wt);
a=1;
len=length(xs_i);
dummy=[xs_i;zeros(order,1)];
dummy1=filter(b,a,dummy);
delay=order/2; % Δοκιμάστε με delay=0!
xs_i=dummy1(delay+(1:len));
dummy=[xs_q;zeros(order,1)];
dummy1=filter(b,a,dummy);
delay=order/2;
xs_q=dummy1(delay+(1:len));
```

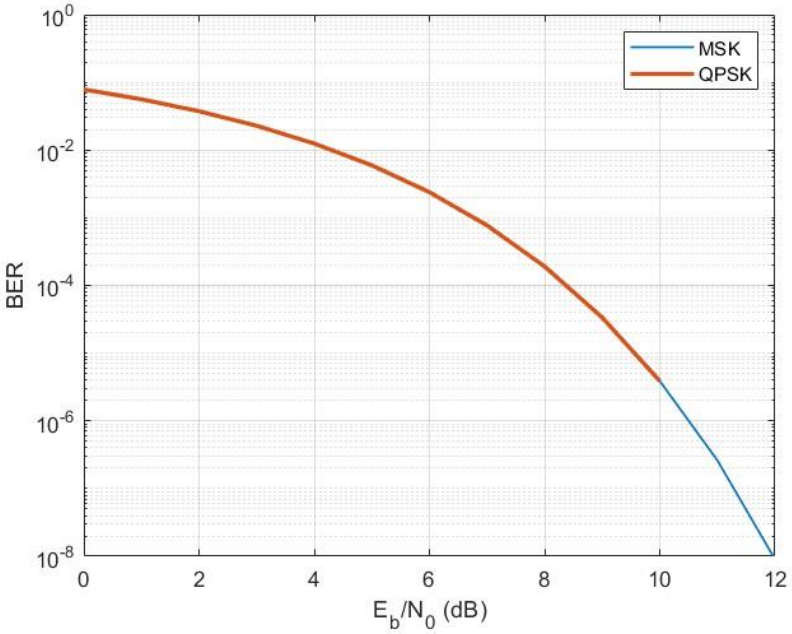
```
bi=1; xr_1=1;
for k=1:2:n-1
 li=(k*ns+1:(k+2)*ns)';
 lq=((k-1)*ns+1:(k+1)*ns)';
 xi=xs_i(li);
 xq=xs_q(lq);
 gmi=cos(pi/2/T*Ts*li); % παλμός matched-filter
 gmq=-gmi; % =sin(pi/2/T*Ts*lq);
 bi_1=bi;
 bi=sign(sum(xi.*gmi));
 bq=sign(sum(xq.*gmq));

 % προκωδικοποίηση % comment for msk_noprecoding_errors
 xr(k)=xr_1*bi_1*bq;
 xr(k+1)=xr(k)*bi*bq;
 xr_1=xr(k+1);
 % χωρίς προκωδικοποίηση % comment for msk_precoding_errors
 xr(k)=bi_1*bq;
 xr(k+1)=bi*bq;

end
xr=xr';
err=not(y==xr);
errors=sum(err);
end
```

Έπειτα, από τις προσομοιώσεις του bertool για τις δύο περιπτώσεις και τον σχεδιασμό των θεωρητικών καμπυλών πορκύπτει το διπλανό σχήμα. Για EbNo=10dB με προκωδικοποίηση, το BER προέκυψε **θεωρητικά ίσο με** $3.9 * 10^{-6}$ και **πειραματικά** $5.3 * 10^{-6}$. Αντίστοιχα, χωρίς προκωδικοποίηση το **θεωρητικό BER ισούται με** $7.75 * 10^{-6}$, ενώ **το πειραματικό υπολογίστηκε** $7.9 * 10^{-6}$.



Παρατηρούμε ότι η πιθανότητα λάθους για το σύστημα χωρίς προκωδικοποίηση είναι διπλάσια από αυτή του συστήματος με προκωδικοποίηση. Αυτό συμβαίνει, καθώς στην περίπτωση που δεν έχουμε προκωδικοποίηση, αν γίνει λάθος σε ένα ψηφίο, γίνεται και στο επόμενο του. Κάτι τέτοιο με την προκωδικοποίηση αποφεύγεται.

Μέρος 5



Σχεδιάζοντας μέσω του bertool της θεωρητικές καμπύλες BER-EbNo για MSK με προκωδικοποίηση και για QPSK (διπλανό σχήμα), παρατηρούμε ότι ταυτίζονται.

Για το σύστημα MSK

Μέσω του τύπου $\frac{1}{T} = \frac{R}{\log_2 M}$ υπολογίζουμε ότι το Baud Rate είναι ίσο με $\frac{1}{T} = \frac{2Mbps}{2} = 1MHz$ για την MSK.

Το σύστημα MSK έχει έυρος ζώνης ίσο με $3 * \frac{1}{T} = 3 * 1 = 3MHz$ όπως φαίνεται και στο ακριβώς από κάτω σχήμα.

Για το σύστημα QPSK

Όταν ένα σύστημα είναι QPSK σημαίνει ότι είναι 4-PSK.

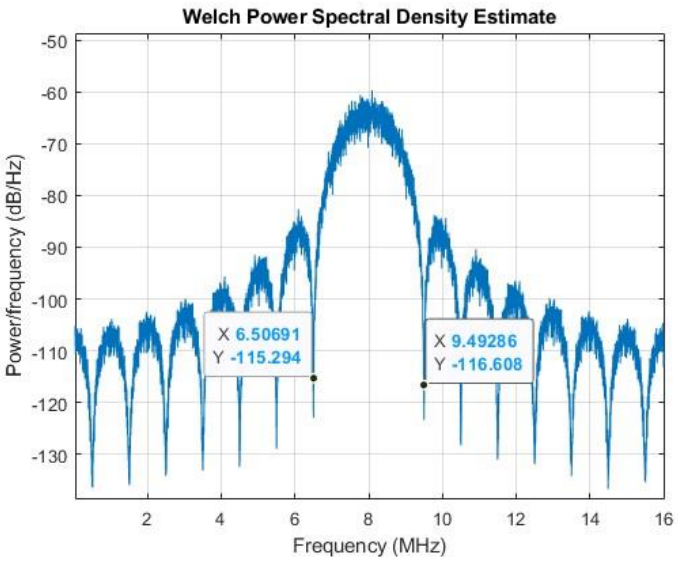
Θεωρούμε ότι το σύστημα έχει το ίδιο bit rate R με το παραπάνω MSK.

Κατά την εξομοίωση χρησιμοποιούμε φίλτρο Nyquist με roll-off factor ίσο με $\alpha = 0.5$. Σε ένα σύστημα QPSK με ζωνοπερατό Nyquist το εύρος ζώνης υπολογίζεται από τον εξής τύπο:

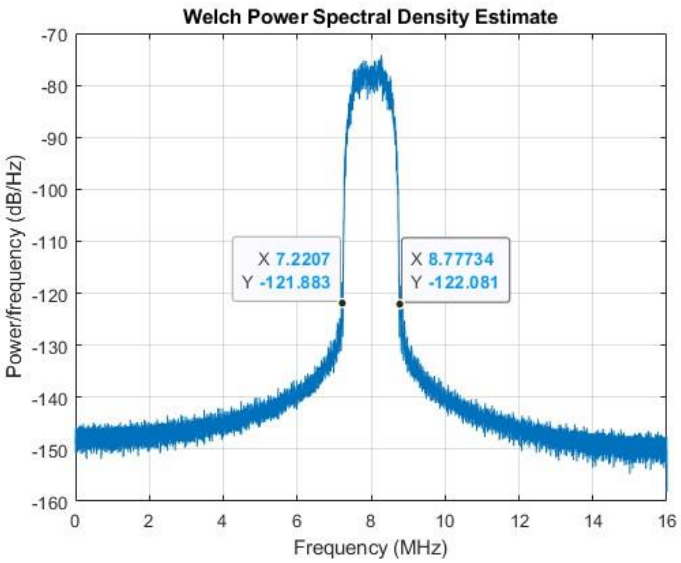
$$W = \frac{1}{T}(1+a) = \frac{R}{\log_2 M}(1+a) = \frac{2Mbps}{\log_2 4}(1+0.5) = 1.5MHz$$

πράγμα που επιβεβαιώνει και το παρακάτω σχήμα.

εύρος ζώνης συστήματος MSK



εύρος ζώνης συστήματος QPSK



Παρατηρούμε, λοιπόν, ότι για την ίδια πιθανότητα λάθους, στην QPSK απαιτείται το μισό εύρος ζώνης απ' ότι στην MSK.

Η πυκνότητα φάσματος της MSK προέκυψε από το προηγούμενο ερώτημα, ενώ αυτή της QPSK σχεδιάστηκε μέσω του παρακάτω κώδικα MATLAB:

```
close all; clear all; clc;
k=2; M=2^k; Nsymb=2000; nsamp=32;

%% 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;
rolloff=0.5;
filtorder = delay*nsamp*2;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);

%% Transmitter
y=upsample(y,nsamp);
ytx = conv(y,rNyquist);
R=2000000; % bit rate
Fs=R/k*nsamp;
fc=8; % carrier frequency in multiples of baud rate (1/T=1MHz)
m=(1:length(ytx));
s=real(ytx.*exp(1j*2*pi*fc*m/nsamp));  % shift to desired frequency band
%plots
%figure; pwelch(real(ytx),[],[],[],Fs); %before frequency shifting
figure; pwelch(s,[],[],[],Fs);
```