ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Μάθημα: Ψηφιακές Επικοινωνίες Ι

Ονοματεπώνυμο: Ειρήνη Δόντη

Α.Μ: 03119839

6η Σειρά Ασκήσεων

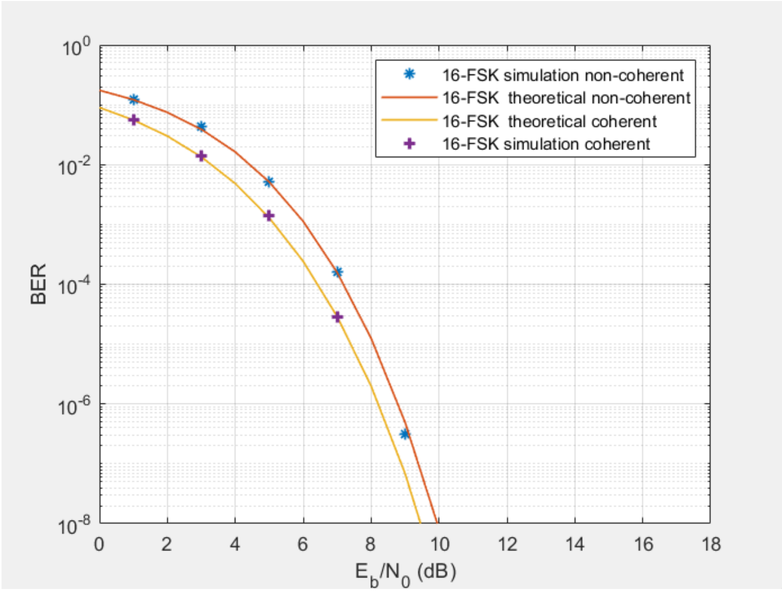
Αθήνα 2022

1. Συμπληρώνουμε τον κώδικα 6.1 των σημειώσεων ώστε να εξομοιώνει την ασύμφωνη FSK, όπως απεικονίζεται στον παρακάτω τροποποιημένο κώδικα:

```
function errors=fsk_errors_non_coherent(bps,Nsymb,ns,EbNo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 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 "non-coherent" distance (BR)
f=fc+BR*((1:M)-(M+1)/2);
% 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
% figure; pwelch(s);
% add noise to the FSK (passband) signal
s=awgn(s,SNR, 'measured');
%% FSK receiver
% non coherent demodulation
%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
th=rand()*pi; %
si=sin(2*pi*(f(i)*tk + th)); %
sq=sin(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
```

Δημιουργούμε τυχαία φάση th στο διάστημα (0,π) και υλοποιούμε ορθογωνικό δέκτη. Καταρχάς, πολλαπλασιάζουμε το λαμβανόμενο σήμα με ημίτονο και συνημίτονο ίδιας φάσης και δημιουργούμε δύο συνιστώσες smi και smq, τις οποίες τετραγωνίζουμε και αθροίζουμε.

2. Χρησιμοποιούμε τη νέα συνάρτηση για εξομοίωση συστήματος 16-FSK και σχεδιάζουμε τις ζητούμενες καμπύλες για σύμφωνη και ασύμφωνη αποδιαμόρφωση, όπως φαίνεται παρακάτω:



Παρακάτω, παραθέτουμε τον κώδικα για το σύστημα 16-FSK για σύμφωνη διαμόρφωση:

```
%Eirini Donti 03119839
function errors=fsk_errors_coherent(bps,Nsymb,ns,EbNo)
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %% 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
 %bps=4; Nsymb=10000; ns=256; EbNo=10;
 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 "non-coherent" distance (BR)
 f=fc+BR*((1:M)-(M+1)/2);
 % 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
 %figure; pwelch(s,[],[],[],ns);
 % add noise to the FSK (passband) signal
 s=awgn(s,SNR, 'measured');
 %% FSK receiver
 % coherent demodulation
 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
 % count errors
 err=not(x==xr);
 errors=sum(sum(err));
 end
```

Παρακάτω, παραθέτουμε τον κώδικα για την εξομοίωση του συστήματος 16-FSK για σύμφωνη διαμόρφωση:

```
%Eirini Donti 03119839
function [ber,numBits] = ask_ber_func_fsk_coherent(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.
bps=4; k=4;% number of bits per symbol
Nsymb=10000; % number of symbols in each run
ns=256; nsamp=256; % 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=fsk_errors_coherent(bps,Nsymb,ns,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;
```

Παρακάτω, παραθέτουμε τον κώδικα για το σύστημα 16-FSK για ασύμφωνη διαμόρφωση:

```
%Eirini Donti 03119839
function errors=fsk_errors_non_coherent(bps,Nsymb,ns,EbNo)
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %% 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
 %bps=4; Nsymb=10000; ns=256; EbNo=10;
 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 "non-coherent" distance (BR)
 f=fc+BR*((1:M)-(M+1)/2);
 % 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
 %figure; pwelch(s,[],[],[],ns);
 % add noise to the FSK (passband) signal
 s=awgn(s,SNR, 'measured');
 %% FSK receiver
 % non coherent demodulation
 %th=0;
 xr=[];
 for k=1:length(s)/ns
 tk=(k-1)*T+tks;
 sk=s((k-1)*ns+1:k*ns);
 smi=[];
```

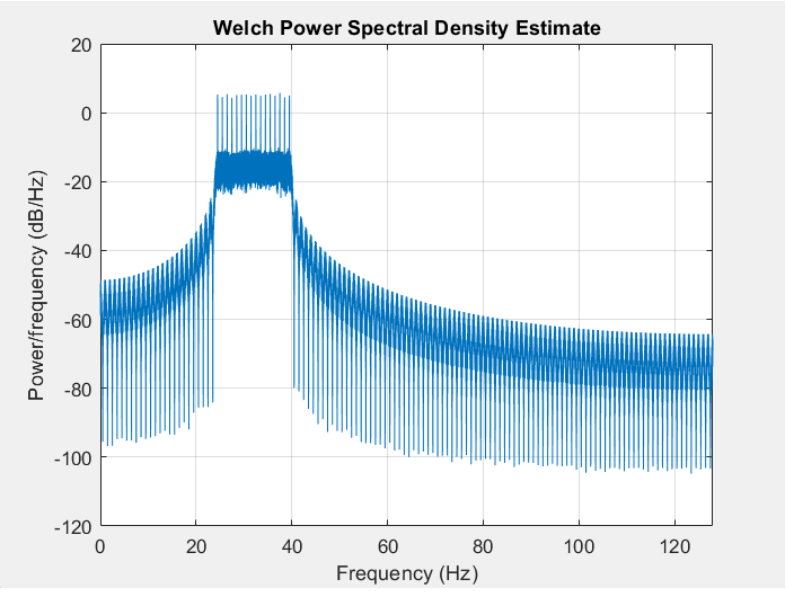
```
for i=1:M
th=rand()*pi; %
si=sin(2*pi*(f(i)*tk + th)); %
sq=cos(2*pi*(f(i)*tk + th)); %
smi=sum(sk.*si);%
smq=sum(sk.*sq); %
sm(i)=sqrt(smi^2+smq^2);
end
[m,j]=max(sm);
xr=[xr;de2bi(j-1,bps)];
end
% count errors
err=not(x==xr);
errors=sum(sum(err));
End
```

Παρακάτω, παραθέτουμε τον κώδικα για την εξομοίωση του συστήματος 16-FSK για ασύμφωνη διαμόρφωση:

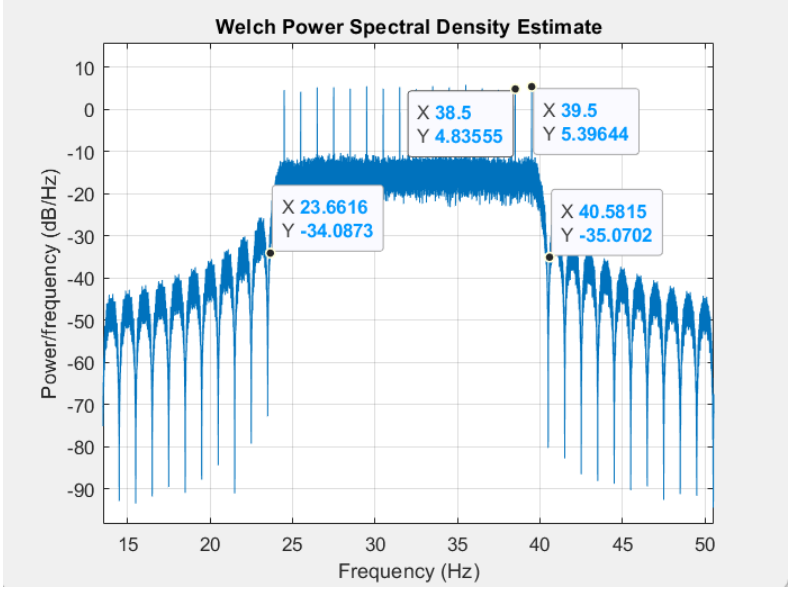
```
%Eirini Donti 03119839
function [ber,numBits] = ask_ber_func_fsk_non_coherent(EbNo, maxNumErrs, maxNumBits,varargin)
% Import Java class for BERTool.
import com.mathworks.toolbox.comm.BERTool.*;
% Initialize variables related to exit criteria.
totErr = 0; % Number of errors observed
numBits = 0; % Number of bits processed
% A. --- Set up parameters. ---
% --- INSERT YOUR CODE HERE.
bps=4; k=4; % number of bits per symbol
Nsymb=10000; % number of symbols in each run
ns=256; nsamp=256; % 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=fsk_errors_non_coherent(bps,Nsymb,ns,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 το φάσμα του ζωνοπερατού σήματος του ερωτήματος 2, για τις εξής περιπτώσεις:

Δημιουργούμε το φάσμα της ασύμφωνης 16-FSK, όπως φαίνεται παρακάτω:



Μεγεθύνουμε την παραπάνω γραφική παράσταση και σημειώνουμε τις συντεταγμένες δύο διαδοχικών συνιστωσών και 2 ακόμη σημεία, ώστε να επαληθεύσουμε ότι παρεμβάλλονται 16 συνιστώσες εντός του εύρους ζώνης:



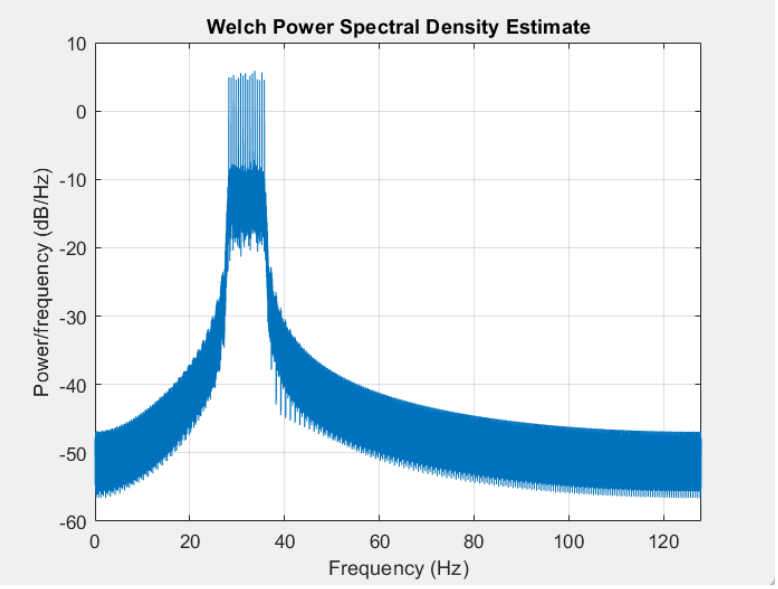
Επαληθεύουμε ότι οι διαδοχικές κρουστικές-συνιστώσες μεταξύ των συχνοτήτων, απέχουν μεταξύ τους BR = 1/T= 1 Hz και το συνολικό εύρος ζώνης είναι περίπου 16 Hz.

Μπορούμε να επαληθεύσουμε τα παραπάνω, αν τυπώσουμε τις διαδοχικές τιμές του διανύσματος f, όπως φαίνεται παρακάτω:

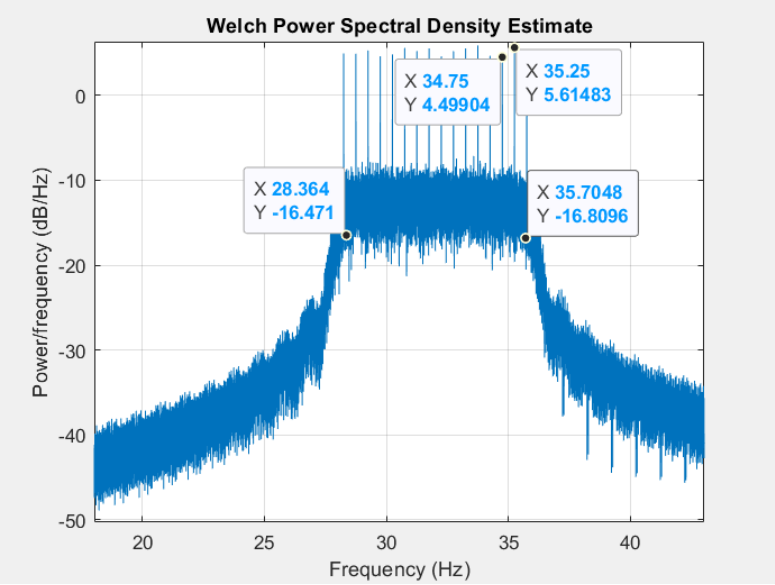
```
f =

   24.5000   25.5000   26.5000   27.5000   28.5000   29.5000   30.5000   31.5000   32.5000   33.5000   34.5000   35.5000   36.5000   37.5000   38.5000   39.5000
```

Δημιουργούμε το φάσμα της σύμφωνης 16-FSK, όπως φαίνεται παρακάτω:



Μεγεθύνουμε την παραπάνω γραφική παράσταση και σημειώνουμε τις συντεταγμένες δύο διαδοχικών συνιστωσών και 2 ακόμη σημεία, ώστε να επαληθεύσουμε ότι παρεμβάλλονται 16 κρουστικές-συνιστώσες εντός του εύρους ζώνης:



Επαληθεύουμε ότι οι διαδοχικές συνιστώσες-κρουστικές, απέχουν μεταξύ τους BR/2 = 1/2T= 0.5 Hz και το συνολικό εύρος ζώνης είναι περίπου 8 Hz.

Μπορούμε να επαληθεύσουμε τα παραπάνω, αν τυπώσουμε τις διαδοχικές τιμές του διανύσματος f, όπως φαίνεται παρακάτω:

```
f =

   28.2500   28.7500   29.2500   29.7500   30.2500   30.7500   31.2500   31.7500   32.2500   32.7500   33.2500   33.7500   34.2500   34.7500   35.2500   35.7500
```

Ο κώδικας που χρησιμοποιήθηκε για την απεικόνιση φάσματος ασύμφωνης 16-FSK είναι ο παρακάτω:

```
%Eirini Donti 03119839
function errors=fsk_errors_non_coherent(bps,Nsymb,ns,EbNo)
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %% 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
 bps=4; Nsymb=10000; ns=256; EbNo=10;
 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 "non-coherent" distance (BR)
 f=fc+BR*((1:M)-(M+1)/2);
 % 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
 figure; pwelch(s,[],[],[],ns);
 % add noise to the FSK (passband) signal
 s=awgn(s,SNR, 'measured');
 %% FSK receiver
 % non coherent demodulation
 %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
 th=rand()*pi; %
 si=sin(2*pi*(f(i)*tk + th)); %
 sq=sin(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
```

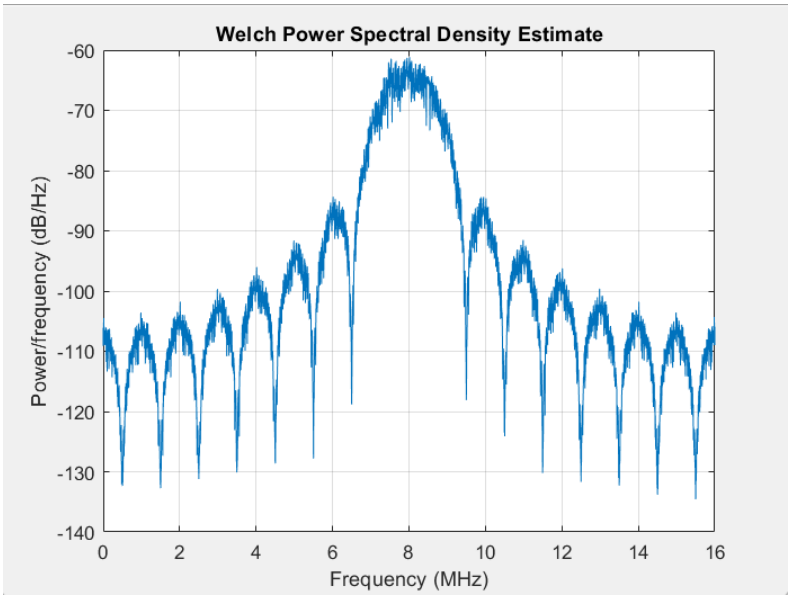
Ο κώδικας που χρησιμοποιήθηκε για την απεικόνιση φάσματος σύμφωνης 16-FSK είναι ο παρακάτω:

```
%Eirini Donti 03119839
function errors=fsk_errors_coherent(bps,Nsymb,ns,EbNo)
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %% 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
 bps=4; Nsymb=10000; ns=256; EbNo=10;
 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 "non-coherent" distance (BR)
f=fc+(BR/2)*((1:M)-(M+1)/2);
% 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
figure; pwelch(s,[],[],[],ns);
% add noise to the FSK (passband) signal
s=awgn(s,SNR, 'measured');
%% FSK receiver
% coherent demodulation
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
% count errors
err=not(x==xr);
errors=sum(sum(err));
End
```

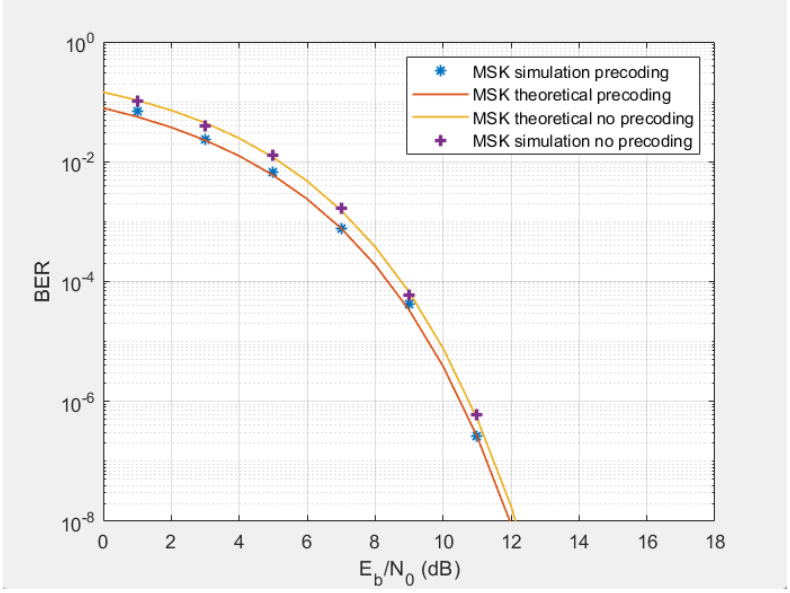
4. Με βάση τον Κώδικα 6.2 των σημειώσεων, εξομοιώνουμε σύστημα MSK μετάδοσης σε ζωνοπερατό δίαυλο με κεντρική συχνότητα 8 MHz και ρυθμό μετάδοσης 2 Mbps.

   Σχεδιάζουμε το φάσμα του ζωνοπερατού σήματος MSK, όπως απεικονίζεται παρακάτω:

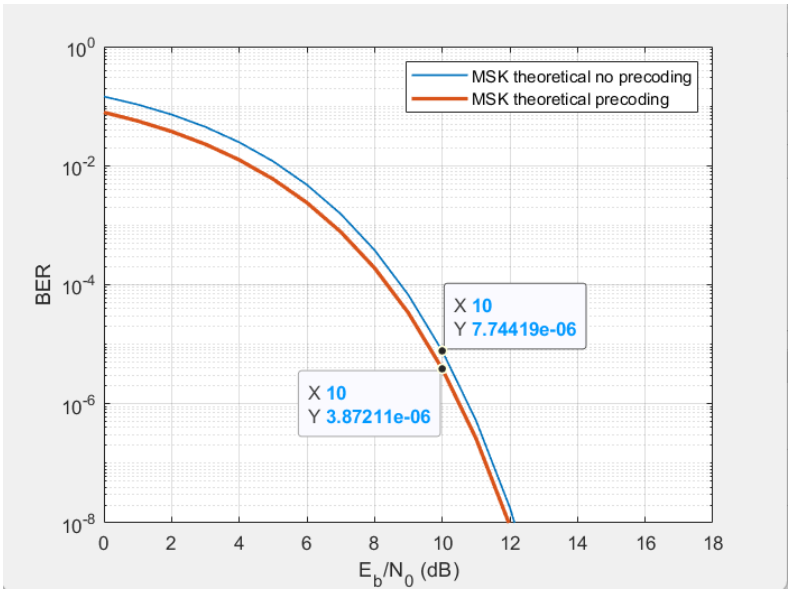


Δημιουργούμε, μέσω του εξομοιωτή BERTool, το διάγραμμα BER-Eb/No και υπολογίζουμε το BER όταν Eb/No=10 dB.

Εκτελούμε την εξομοίωση για σύστημα MSK χωρίς προκωδικοποίηση (no precoding) και με προκωδικοποίηση αντίστοιχα, όπως φαίνεται παρακάτω:



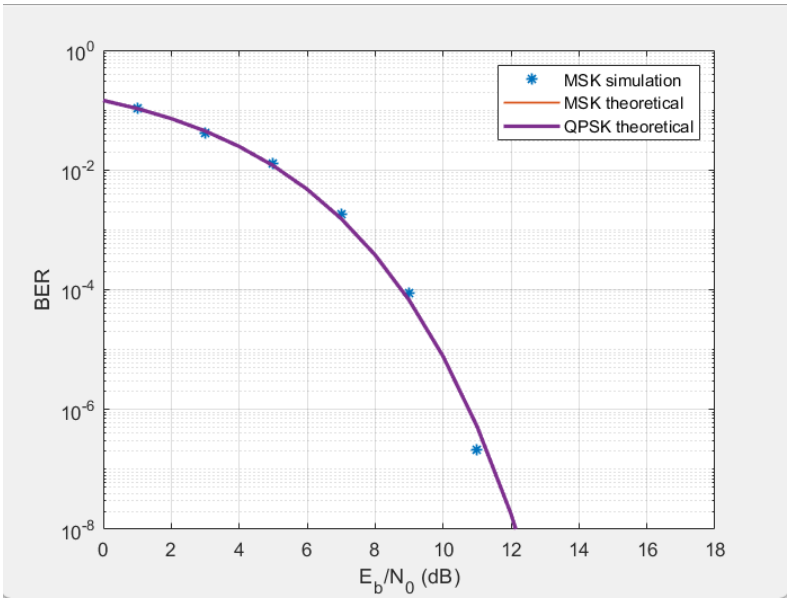
Υπολογίζουμε το BER όταν Eb/No=10 dB, από την παρακάτω θεωρητική γραφική παράσταση:



Όταν η τιμή του σηματοθορυβικού λόγου είναι Eb/No=10 dB, τότε η τιμή BER είναι $7.744*10^{-6}$ στην περίπτωση εξομοίωσης της MSK χωρίς προκωδικοποίηση.

Όταν η τιμή του σηματοθορυβικού λόγου είναι Eb/No=10 dB, τότε η τιμή BER είναι $3.872*10^{-6}$ στην περίπτωση εξομοίωσης της MSK με προκωδικοποίηση.

5. Βρίσκουμε τις τιμές των παραμέτρων ισοδύναμου συστήματος QPSK. Συγκρίνουμε τα δύο συστήματα ως προς το BER και το εύρος ζώνης.



Παρατηρούμε ότι τα συστήματα QPSK και MSK με προκωδικοποίηση ταυτίζονται ως προς το BER, όπως φαίνεται στην παραπάνω.

Για Eb/No = 10 dB, η πιθανότητα λάθους είναι BER = $3.872*10^{-6}$.

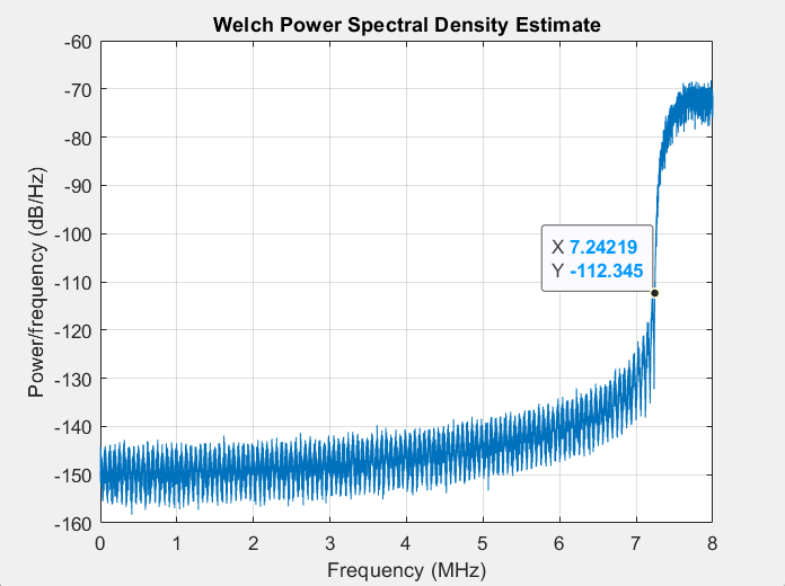
Το Baud rate για την QPSK βρίσκεται από τον τύπο $R=\frac{log_2 M}{T}$ ή $\frac{1}{T}$ = 1 MHz για R=2 Mbps.

Παρατηρούμε, από το ερώτημα 4, ότι το σύστημα MSK έχει εύρος ζώνης σχεδόν $\frac{3}{T}$.

Θεωρούμε σύστημα QPSK με φίλτρο Nyquist a=0.5 και συνεπώς το εύρος ζώνης είναι:

$W=\frac{(1+\alpha)*R}{(log_2 M)}$ = 1.5 MHz

Εκτελούμε το πρόγραμμα για το σύστημα QPSK, οπότε λαμβάνουμε την παρακάτω γραφική παράσταση για το φάσμα του συστήματος QPSK:



Παρατηρούμε ότι επαληθεύεται ο παραπάνω υπολογισμός για το εύρος ζώνης του συστήματος QPSK, καθώς το εύρος ζώνης από το παραπάνω διάγραμμα είναι περίπου 2*0.75 = 1.5 MHz.

Επίσης, παρατηρούμε, από τα παραπάνω, ότι το QPSK έχει μικρότερες απαιτήσεις σε εύρος ζώνης από το MSK.

Ο κώδικας που χρησιμοποιήθηκε για το σύστημα QPSK είναι ο παρακάτω:

```
%Eirini Donti 03119839
function errors=ask_Nyq_psk_6_5(k,Nsymb,nsamp,EbNo,rolloff)
k=2; Nsymb=10000; nsamp=16; EbNo=10;
rolloff=0.5;
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=2000000; % 2 Mbps % R=4575000
Fs=R/k*nsamp;
fc=8;  % 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));
```