# Ψηφιακές Επικοινωνίες

## 4η Εργαστηριακή Άσκηση

*Άννα Κουτσώνη-03120019*

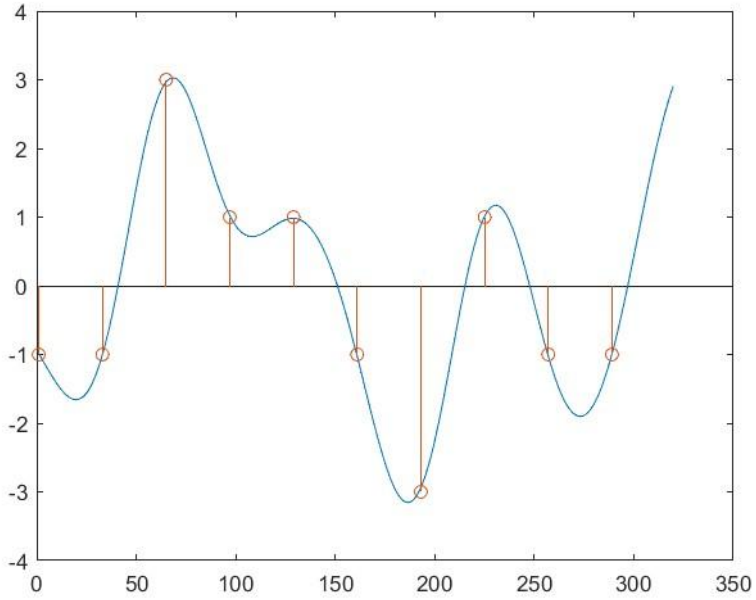
### Μέρος 1ο

Χρησιμοποιούμε τον εξής κώδικα:

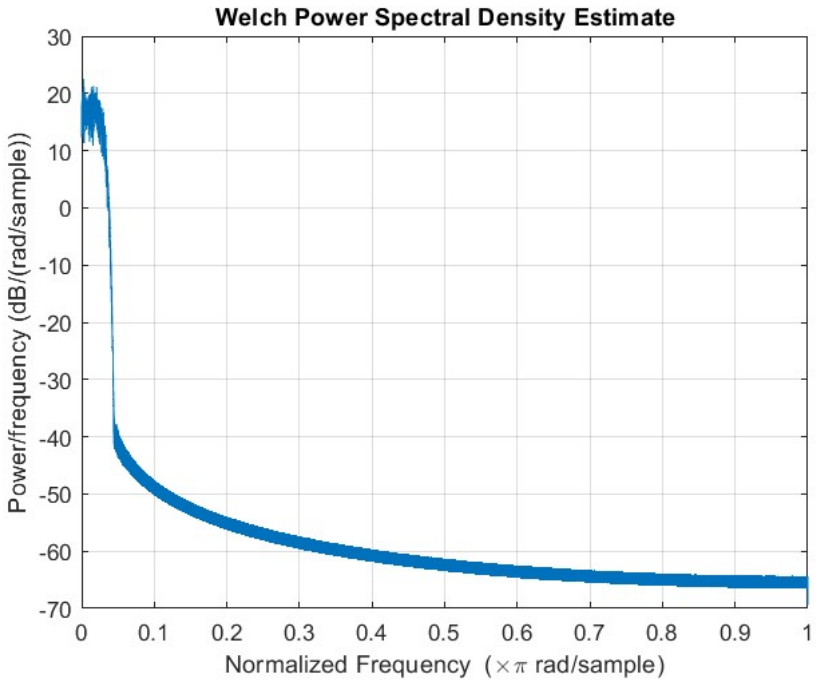
```
clear all;
close all;
clc;
L=8;
step=2; % ο αριθμός των πλατών και το βήμα μεταξύ τους
k=floor(log(L));
K=4;
Nsymb=2500;
x=round(rand(1,K*Nsymb));
mapping=[step/2; -step/2];
if(k>1)
 for j=2:k
 mapping=[mapping+2^(j-1)*step/2;-mapping-2^(j-1)*step/2];
 end
end
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb');
y1=[];
for i=1:length(xsym)
 y1=[y1 mapping(xsym(i)+1)];
end
```

```
nsamp=32;
rolloff=0.4;
delay=4;
filtorder=delay*nsamp*2;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
y2=upsample(y1,nsamp);
ytx=conv(y2,rNyquist);
yrx=conv(ytx,rNyquist);
yrx = yrx(2*delay*nsamp+1:end-2*delay*nsamp);
figure(1);
plot(yrx(1:10*nsamp));
hold;
stem([1:nsamp:nsamp*10],y1(1:10));
figure(3);
pwelch(yrx);
```

Γραφική παράσταση τμήματος του σήματος στην έξοδο προσαρμοσμένου φίλτρου στο δέκτη διάρκειας 10*T με υπερθεμένα στο τμήμα αυτό τα αντίστοιχα δείγματα του σήματος εισόδου στο πλέγμα περιόδου T:




Φάσμα σήματος στο δέκτη:




Παρατηρούμε ότι το φάσμα του σήματος εξόδου είναι σχεδόν ιδανικό και έχει τη μορφή ενός βαθυπερατού φίλτρου. Αυτό οφείλεται στην μικρή τιμή του rolloff(0.4) διότι όσο πιο κοντά είναι στο 0 τόσο πιο πολύ το φίλτρο προσεγγίζει το ιδανικό (αποκόπτει ιδανικά στη συχνότητα αποκοπής που επιθυμούμε). Το εύρος ζώνης του είναι περίπου 0.04 σε κανονικοποιημένη συχνότητα.

## Μέρος 2ο

Για την εξομοίωση με bertool χρησιμοποιούμε για τον υπολογισμό των λανθασμένων συμβόλων την ask_Nyq_filter την οποία καλούμε μέσα από την ask_ber_func_Nyq στο bertool.

ask_Nyq_filter:

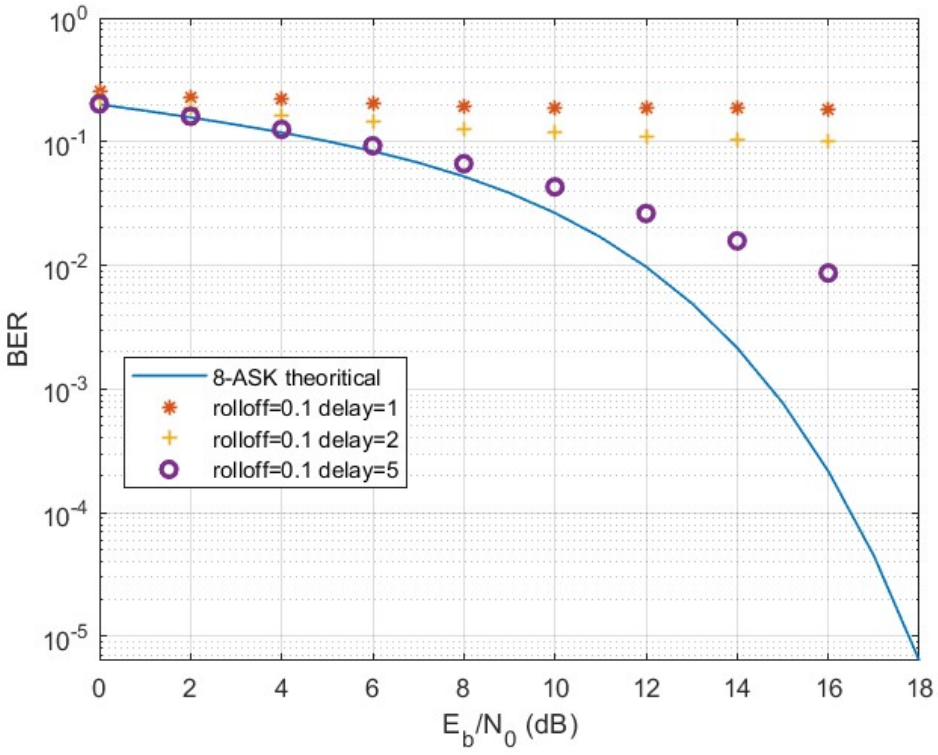
```
function errors=ask_Nyq_filter(k,Nsymb,nsamp,EbNo)
L=2^k;
step=2;
SNR=EbNo-10*log10(nsamp/2/k); % SNR ανά δείγμα σήματος
x=round(rand(1,k*Nsymb));
mapping=[step/2; -step/2];
if(k>1)
 for j=2:k
 mapping=[mapping+2^(j-1)*step/2;-mapping-2^(j-1)*step/2];
 end
end
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb');
y1=[];
for i=1:length(xsym)
 y1=[y1 mapping(xsym(i)+1)];
end
```

```
delay = 5; % Group delay (# of input symbols)
filtorder = delay*nsamp*2; % τάξη φίλτρου
rolloff = 0.4; % Συντελεστής πτώσης -- rolloff factor
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
y2=upsample(y1,nsamp);
ytx= conv(y2,rNyquist);
ynoisy=awgn(ytx,SNR,'measured'); % θορυβώδες σήμα
yrx=conv(ynoisy,rNyquist);
yrx = downsample(yrx,nsamp); % Υποδειγμάτιση
yrx = yrx(2*delay+1:end-2*delay); % περικοπή, λόγω καθυστέρησης
xr=[];
for i=1:length(yrx)
[m,f]=min(abs(mapping-yrx(i)));
xr=[xr de2bi(f-1,k,'left-msb')];
end
err=not(xr==x);
errors=sum(err);
symbols=length(err);
end
```

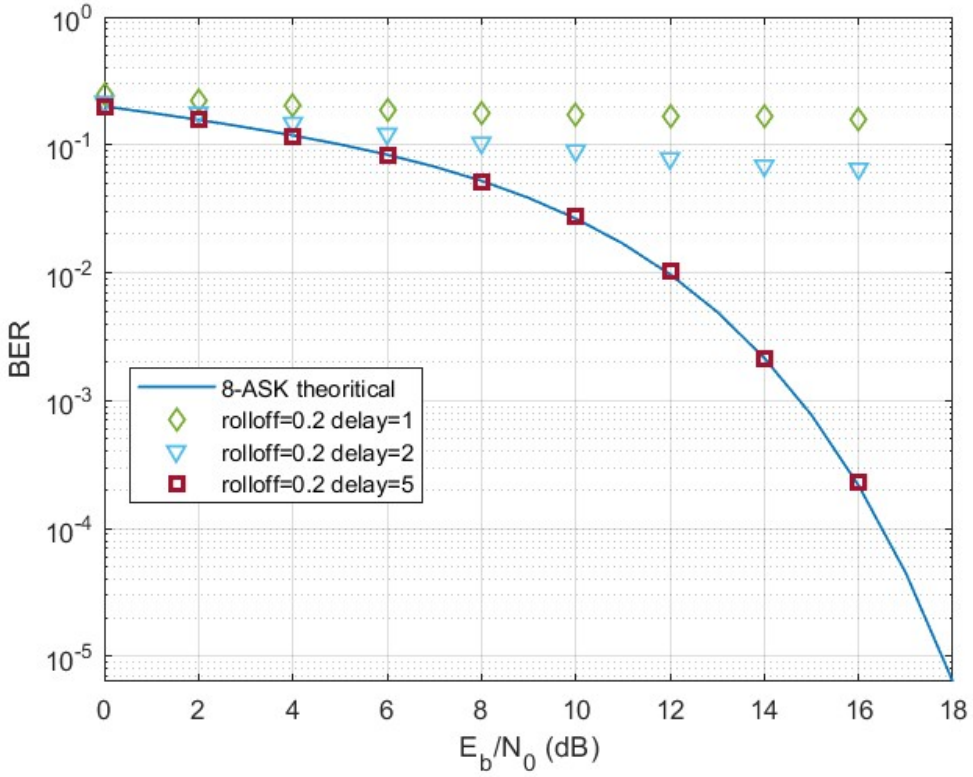
ask_ber_func_Nyq:

```
function [ber,numBits] = ask_ber_func_Nyq(EbNo, maxNumErrs, maxNumBits)
import com.mathworks.toolbox.comm.BERTool.*;
totErr = 0; % Number of errors observed
numBits = 0; % Number of bits processed
k=3; % number of bits per symbol
Nsymb=10000; % number of symbols in each run
nsamp=32; % oversampling,i.e. number of samples per T
while((totErr < maxNumErrs) && (numBits < maxNumBits))
 errors=ask_Nyq_filter(k,Nsymb,nsamp,EbNo);
 totErr=totErr+errors;
 numBits=numBits + k*Nsymb;
end % End of loop
ber = totErr/numBits;
end
```

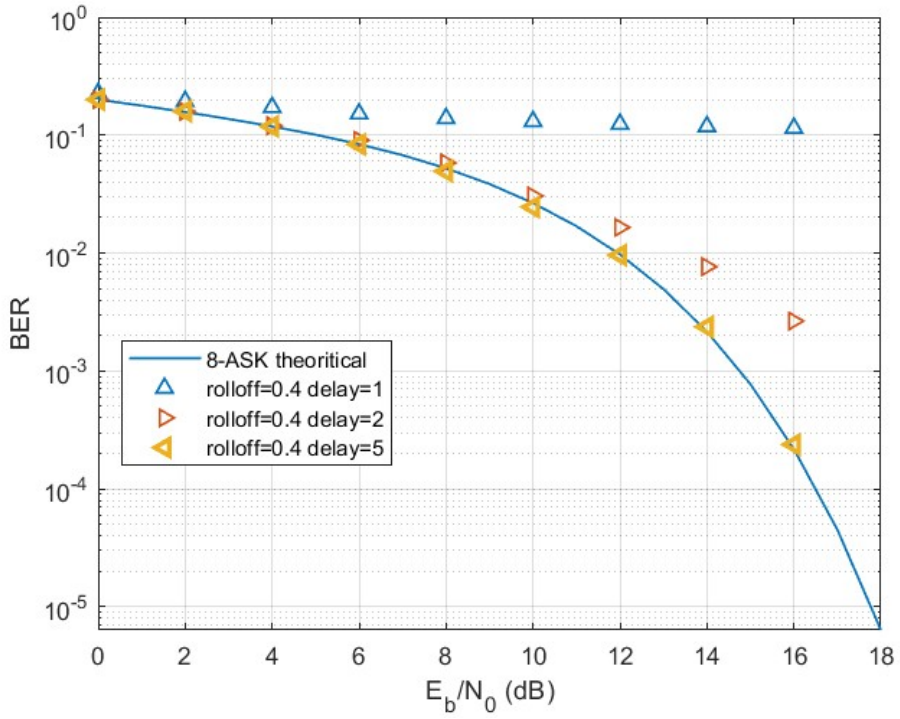
I. Για rolloff=0.1 και delay={1,2,5} για αντίστοιχα τάξη φίλτρου={64,128,320} προκύπτει το εξής γράφημα:




II. Για rolloff=0.2 και delay={1,2,5} για αντίστοιχα τάξη φίλτρου={64,128,320} προκύπτει το εξής γράφημα:



III. Για rolloff=0.4 και delay={1,2,5} για αντίστοιχα τάξη φίλτρου={64,128,320} προκύπτει το εξής γράφημα:



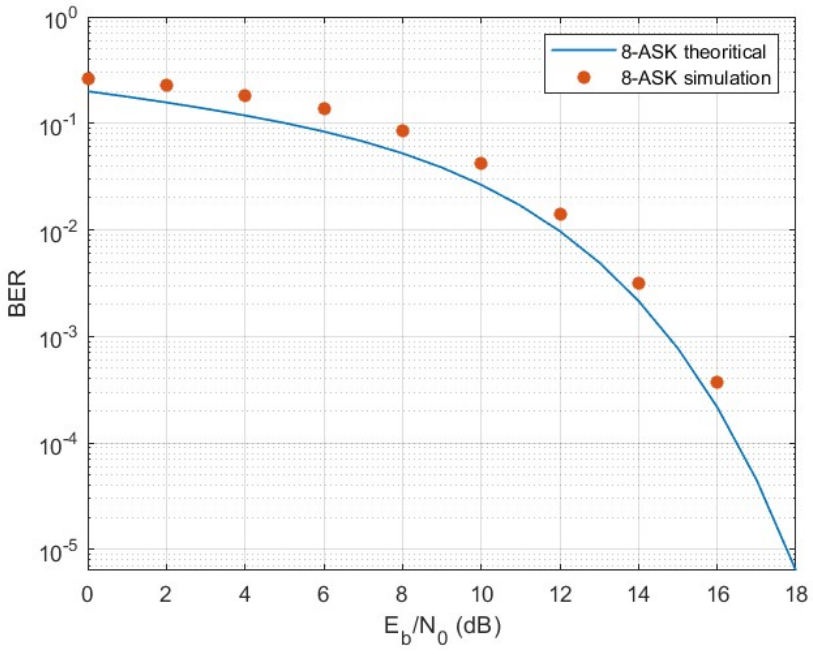

Παρατηρούμε ότι όσο αυξάνουμε την τάξη του φίλτρου, όσο δηλαδή αυξάνεται και η καθυστέρηση που εισάγει το φίλτρο, τόσο περισσότερο προσεγγίζει τη θεωρητική καμπύλη. Επίσης, αυξάνοντας τον συντελεστή πτώσης (rolloff) πετυχαίνουμε καλύτερη προσέγγιση με φίλτρο χαμηλότερης τάξης. Αντίθετα για μικρό rolloff χρειάζεται φίλτρο που να εισάγει μεγάλη καθυστέρηση και να είναι άρα και μεγαλύτερης τάξης. Όσο μικρότερο είναι το rolloff τόσο περισσότερο το φίλτρο παρουσιάζει την ιδανική συμπεριφορά ενός βαθυπερατού φίλτρου, όμως τόσο υψηλότεροι είναι οι πλευρικοί λοβοί του, οπότε δημιουργείται μια επικάλυψη μεταξύ των εκπεμπόμενων συμβόλων και αυξάνεται το BER.

**Μέρος 3ο**

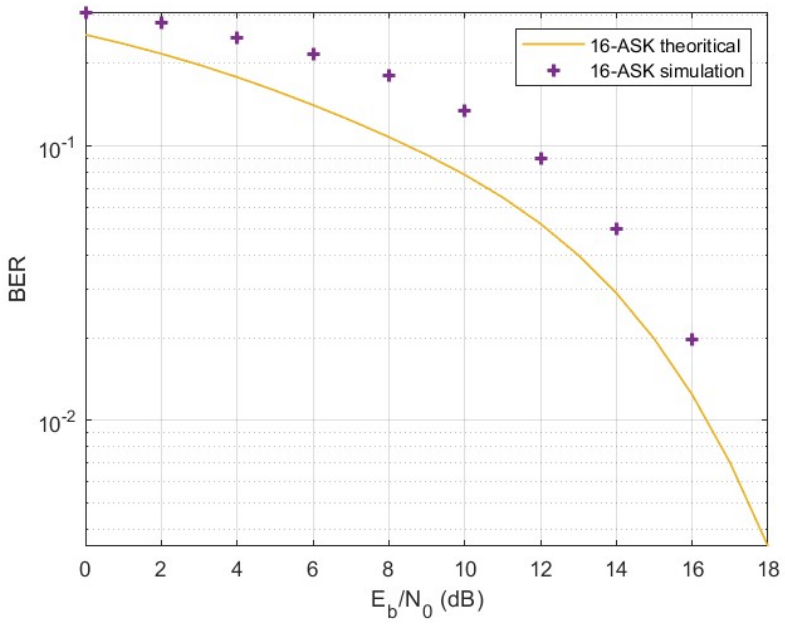
Αντικαθιστώντας το κομμάτι του κώδικα της κωδικοποίησης Gray με άλλη κωδικοποίηση στην οποία mapping=-(L-1):step:(L-1).
Χρησιμοποιούμε τον εξής τροποποιημένο κώδικα για την ask_Nyq_filter ενώ η ask_ber_func παραμένει ίδια με πριν:

```
function errors=ask_Nyq_filter(k,Nsymb,nsamp,EbNo)
L=2^k;
step=2;
SNR=EbNo-10*log10(nsamp/2/k); % SNR ανά δείγμα σήματος
x=round(rand(1,k*Nsymb));
mapping=-(L-1):step:(L-1);
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb');
y1=[];
for i=1:length(xsym)
 y1=[y1 mapping(xsym(i)+1)];
end
delay = 5; % Group delay (# of input symbols)
filtorder = delay*nsamp*2; % τάξη φίλτρου
rolloff = 0.4; % Συντελεστής πτώσης -- rolloff factor
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
y2=upsample(y1,nsamp);
ytx= conv(y2,rNyquist);
ynoisy=awgn(ytx,SNR,'measured'); % θορυβώδες σήμα
yrx=conv(ynoisy,rNyquist);
yrx = downsample(yrx,nsamp); % Υποδειγμάτιση
yrx = yrx(2*delay+1:end-2*delay); % περικοπή, λόγω καθυστέρησης
xr=[];
for i=1:length(yrx)
[m,f]=min(abs(mapping-yrx(i)));
xr=[xr de2bi(f-1,k,'left-msb')];
end
err=[];
errors=[];
err=not(xr==x);
errors=sum(err);
symbols=length(err);
end
```

Μέσω του bertool προκύπτουν τα εξής γραφήματα:
Για 8-ASK:



Για 16-ASK:




Παρατηρούμε ότι με διαφορετική κωδικοποίηση όσο αυξάνεται ο αριθμός των bits ανά σύμβολο, τόσο χειροτερεύει ο ρυθμός λανθασμένων ψηφίων BER.

**Μέρος 4ο**

Για k=3(L=2^k=8 →8-ASK), W=1 MHz , $N_o$=100 picowatt/Hz , R=4Mbps , BER=2Kbps θα έχουμε:

$$\frac{R}{\log_2 L} = \frac{1}{T} \Rightarrow T = \frac{\log_2 L}{R} = \frac{\log_2 8}{4 * 10^6} = \frac{3}{4 * 10^6} = 0{,}75\ \mu sec$$

$$W = \frac{1}{2T}(1 + a) \Rightarrow a = 2 * W * T - 1 = 2 * 10^6 * O{,}75 * 10^{-6} - 1 = 0{,}5$$

$$\acute{\alpha}\rho\alpha\ 0 < a < 1$$

άρα η τιμή είναι αποδεκτή και επαληθεύεται η προδιαγραφή για το εύρος ζώνης.

$$BER = \frac{2 * 10^3}{4 * 10^6} = 0{,}0005$$