ΗΜΜΥ ΕΜΠ

6ο Εξάμηνο

Ψηφιακές Επικοινωνίες Ι

Lab 4

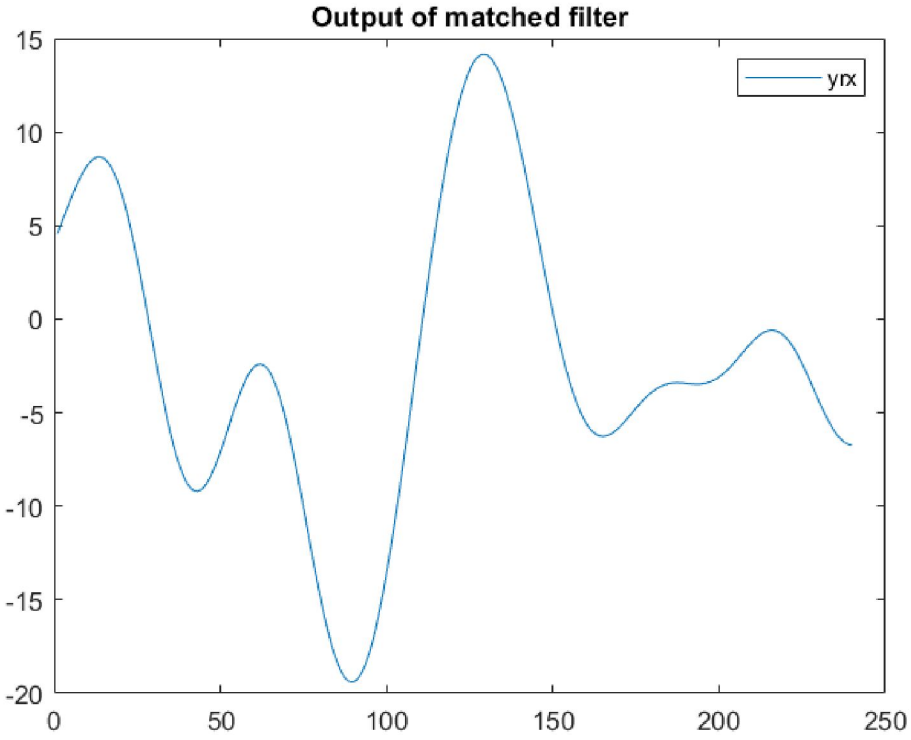
1. Παραγωγή σήματος με φίλτρα Nyquist – Διαγράμματα χρόνου και συχνότητας

Εξήγηση του mapping:

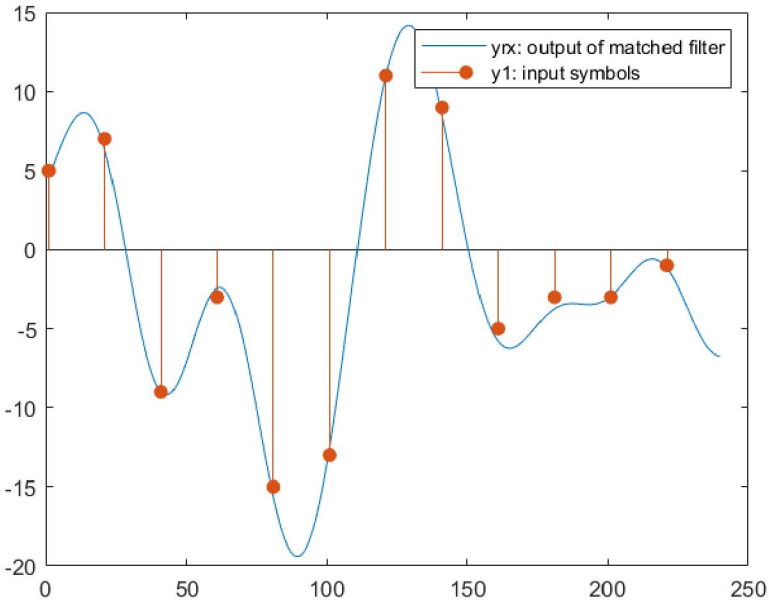
Το διάνυσμα x είναι μια ακολουθία από bits 0 και 1, τα οποία πρέπει ανά k=4 να αντιστοιχήσουμε σε ένα σύμβολο (πλάτος). Για το λόγο αυτό φτιάχνουμε το mapping, το οποίο είναι στην ουσία ένα lookup table με τις αντιστοιχίες αυτές σε κωδικοποίηση gray. Το mapping έχει L=16 τιμές, και το φτιάχνουμε με επαναληπτικό αλγόριθμο διπλασιάζοντας τις τιμές σε κάθε βήμα. Σε κώδικα gray 2 διαδοχικές τιμές διαφέρουν κατά 1 μόνο bit, οπότε το mapping αναδιατάσσει τα πλάτη –(L-1), -(L-1)+2, …, (L-1)-2, L-1 στη σειρά του κώδικα gray.

Για να κωδικοποιήσουμε το σήμα πρώτα φτιάχνουμε το xsym, το οποίο για κάθε k=4 bits του x έχει την αντίστοιχη δεκαδική τιμή. Στη συνέχεια μέσω του mapping φτιάχνουμε το y1, που αποθηκεύει την κωδικοποιημένη τιμή που αντιστοιχεί στην κάθε τιμή του xsym.

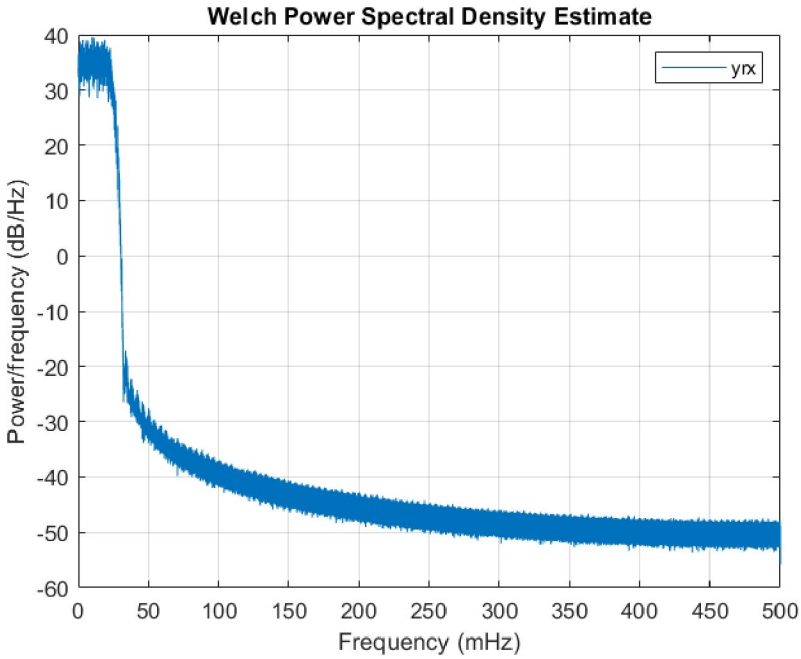
a.



b. Υπάρχουν μικρές αποκλίσεις, λόγω θορύβου:



c.



Παρατηρούμε ότι το εύρος ζώνης είναι περίπου 20 mHz. Αυτό είναι αναμενόμενο κοιτώντας την απόκριση συχνότητας του φίλτρο rNyquist, αφού μπλοκάρει τις συχνότητες πάνω από περίπου 0.05 (κανονικοποιημένη). Το φίλτρο Nyquist προσεγγίζει το ιδανικό βαθυπερατό φίλτρο όσο μικρότερο είναι το α (εδώ 0.25), καθώς το εύρος ζώνης του είναι $(1 + a)/T$.

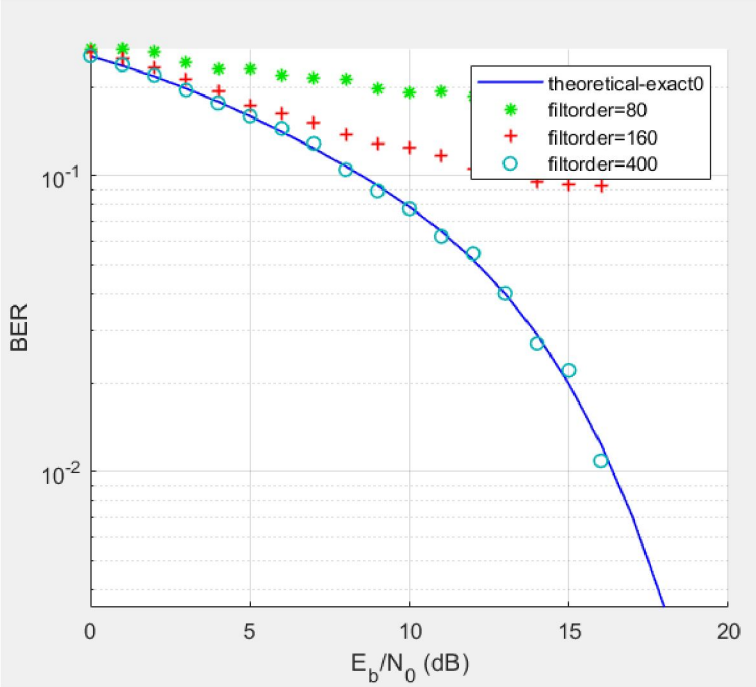
Κώδικας για την άσκηση 1: αρχείο *ask4_1.m*

```
clear all; close all;
L=16; step=2; k=log2(L); nsamp=20; Nsymb=10000;
EbNo=14;

% Διάνυσμα τυχαίων bits
x=randi([0,1],1,Nsymb*k);

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

%% Ορισμός παραμέτρων φίλτρου
delay = 6; % Group delay (# of input symbols)
filtorder = delay*nsamp*2; % τάξη φίλτρου
rolloff = 0.25; % Συντελεστής πτώσης -- rolloff factor
% κρουστική απόκριση φίλτρου τετρ. ρίζας ανυψ. συνημιτόνου
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);

% ----------------------
%% ΕΚΠΕΜΠΟΜΕΝΟ ΣΗΜΑ
% Υπερδειγμάτιση και εφαρμογή φίλτρου rNyquist
y=upsample(y1,nsamp);
ytx = conv(y,rNyquist);
% Διάγραμμα οφθαλμού για μέρος του φιλτραρισμένου σήματος
% eyediagram(ytx(1:2000),nsamp*2);

SNR=EbNo-10*log10(nsamp/2/k); % SNR ανά δείγμα σήματος
Py=10*log10(ytx*ytx'/length(ytx));
Pn=Py-SNR;
n=sqrt(10^(Pn/10))*randn(1,length(ytx));
ynoisy=ytx+n;
%ynoisy=awgn(ytx,SNR,'measured'); % θορυβώδες σήμα
% ----------------------
%% ΛΑΜΒΑΝΟΜΕΝΟ ΣΗΜΑ
% Φιλτράρισμα σήματος με φίλτρο τετρ. ρίζας ανυψ. συνημιτόνου
yrx=conv(ynoisy,rNyquist);
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay); % περικοπή, λόγω
καθυστέρησης
yr = downsample(yrx,nsamp); % Υποδειγμάτιση

% Γραφικές παραστάσεις
```

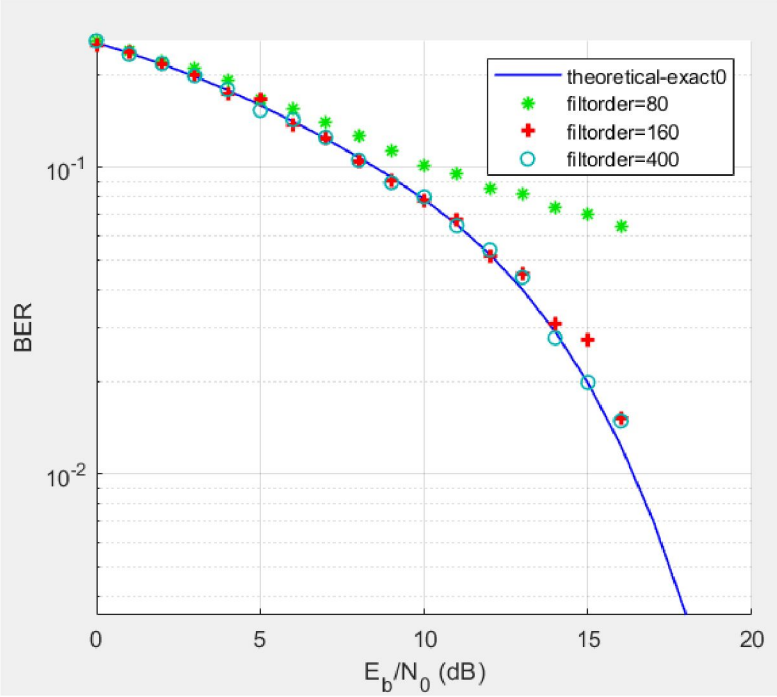
```
figure; plot(yrx(1:12*nsamp)); title('Output of matched filter');
legend('yrx');
figure; plot(yrx(1:12*nsamp)); hold on;
stem([1:nsamp:nsamp*12],y1(1:12),'filled'); hold off
legend('yrx: output of matched filter','y1: input symbols');
figure; pwelch(yrx,[],[],[],1);
legend('yrx');
```

2. Υπολογισμός επίδοσης BER vs Eb/No

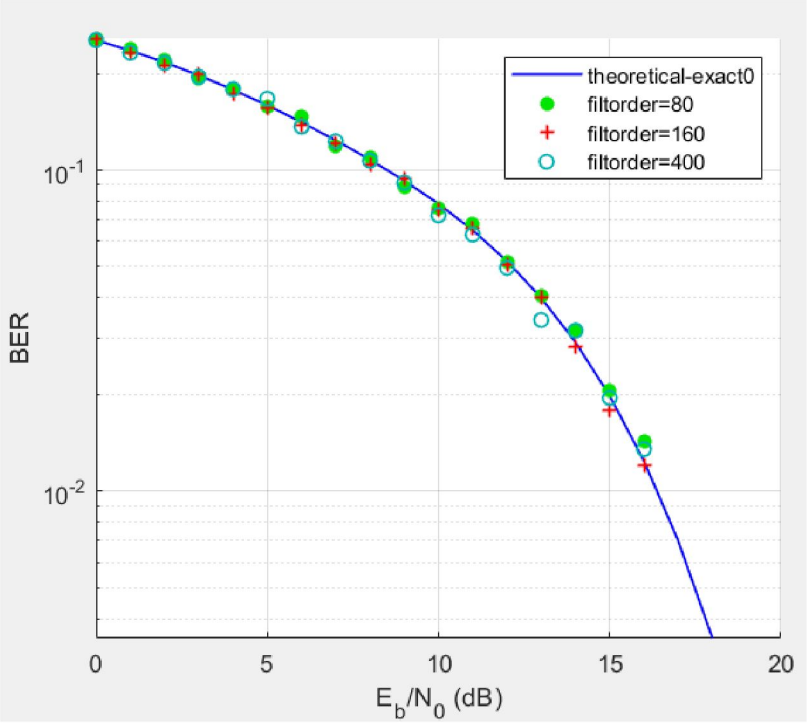
I. rolloff=0.1



II. rolloff=0.35



III. rolloff=0.9



Όσο μικρότερο είναι το roll-off τόσο περισσότερο το φίλτρο μοιάζει με ορθογωνικό, με αποτέλεσμα οι πλευρικοί λωβοί του φίλτρου να είναι ψηλότεροι. Έτσι η ισχύς θορύβου είναι μεγαλύτερη, γεγονός που οδηγεί σε περισσότερα λάθη όπως παρατηρούμε στο πρώτο σχήμα για r=0.1. Αυξάνοντας όμως την τάξη φίλτρου (δηλαδή το μήκος του, και την καθυστέρηση που εισάγει) τα λάθη μειώνονται, οπότε τελικά για group delay 10T το πειραματικό BER ταυτίζεται με τη θεωρητική καμπύλη ακόμα και για r=0.1.

Κώδικας για άσκηση 2:

Καλούμε το bertool, δίνοντας το αρχείο *ask_ber_func_nyq.m*

Στην αρχή του *ask_ber_func_nyq.m* ρυθμίζουμε τις παραμέτρους `rolloff, delay`.

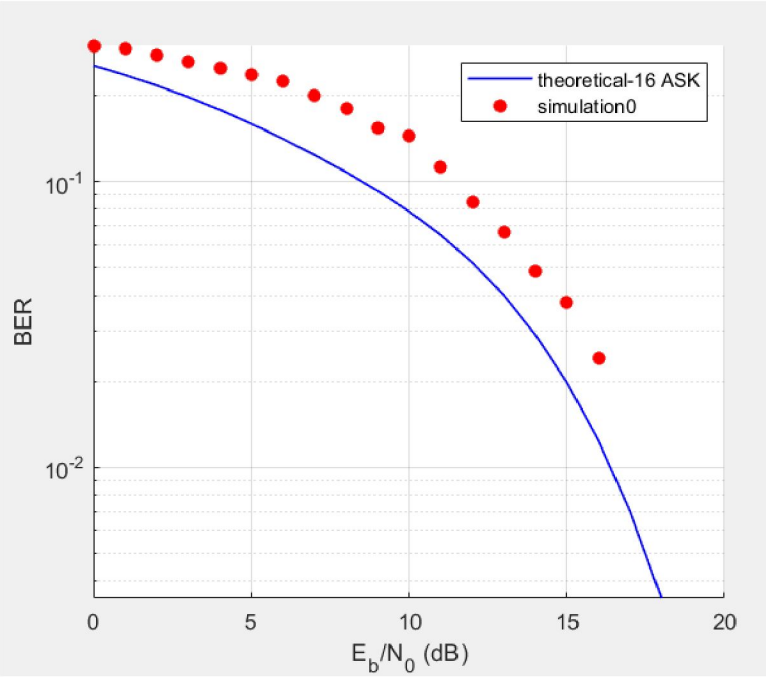
Το αρχείο αυτό καλεί τη συνάρτηση *ask_nyq_filter_2.m* με τις παραμέτρους αυτές.

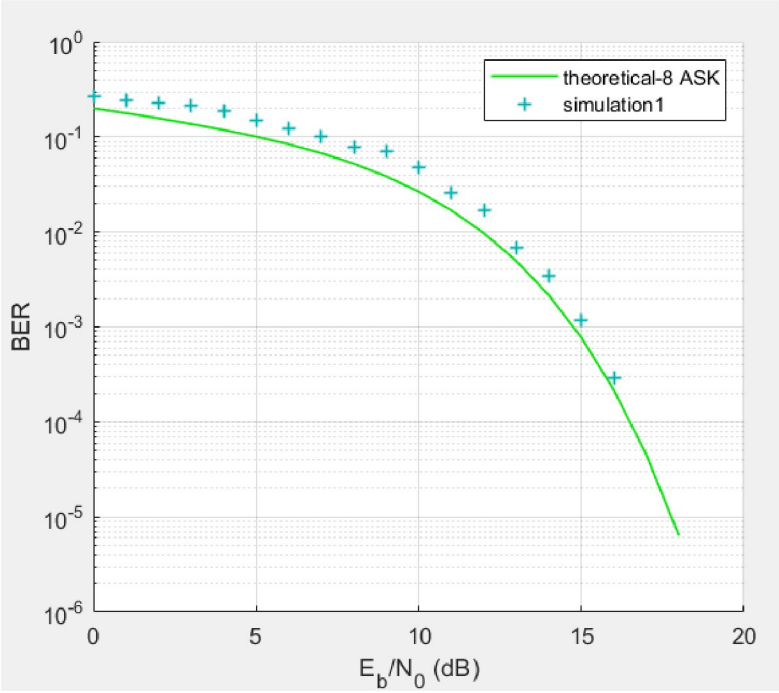
Συνάρτηση *ask_nyq_filter_2.m*:

```
function errors=ask_Nyq_filter_2(k,Nsymb,nsamp,EbNo,rolloff,delay)
L=2^k; step=2;
% Διάνυσμα τυχαίων bits
x=randi([0,1],1,Nsymb*k);

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

%% Ορισμός παραμέτρων φίλτρου
filtorder = delay*nsamp*2; % τάξη φίλτρου
% κρουστική απόκριση φίλτρου τετρ. ρίζας ανυψ. συνημιτόνου
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
% ----------------------
%% ΕΚΠΕΜΠΟΜΕΝΟ ΣΗΜΑ
% Υπερδειγμάτιση και εφαρμογή φίλτρου rNyquist
y=upsample(y1,nsamp);
ytx = conv(y,rNyquist);
% Διάγραμμα οφθαλμού για μέρος του φιλτραρισμένου σήματος
% eyediagram(ytx(1:2000),nsamp*2);

SNR=EbNo-10*log10(nsamp/2/k); % SNR ανά δείγμα σήματος
Py=10*log10(ytx*ytx'/length(ytx));
Pn=Py-SNR;
n=sqrt(10^(Pn/10))*randn(1,length(ytx));
ynoisy=ytx+n;
%ynoisy=awgn(ytx,SNR,'measured'); % θορυβώδες σήμα
% ----------------------
%% ΛΑΜΒΑΝΟΜΕΝΟ ΣΗΜΑ
% Φιλτράρισμα σήματος με φίλτρο τετρ. ρίζας ανυψ. συνημιτόνου
yrx=conv(ynoisy,rNyquist);
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay); % περικοπή, λόγω
καθυστέρησης
yr = downsample(yrx,nsamp); % Υποδειγμάτιση

% Ανιχνευτής ελάχιστης απόστασης L πλατών
xr=[];
for i=1:length(yr)
    [m,j]=min(abs(mapping-yr(i)));
    xr=[xr de2bi(j-1,k,'left-msb')];
end
errors=sum(not(x==xr));
```

3. Επίδραση του τρόπου κωδικοποίησης: Gray ή άλλη

Για delay=6 και rolloff=0.35 και `mapping=-(L-1):step:(L-1)`





Παρατηρούμε ότι τα λάθη είναι περισσότερα από την κωδικοποίηση gray, περισσότερο για μεγαλύτερες τιμές του k. Αυτό ισχύει επειδή αν ο δέκτης αποφασίσει λάθος σύμβολο, τις περισσότερες φορές επιλέγει κάποιο γειτονικό σύμβολο, που στην περίπτωση της κωδικοποίησης gray διαφέρουν κατά 1 μόνο bit. Με άλλη κωδικοποίηση τα γειτονικά σύμβολα διαφέρουν κατά περισσότερα bits, με αποτέλεσμα μια λάθος απόφαση συμβόλου να οδηγεί σε περισσότερα λανθασμένα bits κατά μέσο όρο.

Κώδικας για άσκηση 2:

Καλούμε το bertool, δίνοντας το αρχείο *ask_ber_func_nogray.m*

Ρυθμίζουμε την παράμετρο `k` στην επιθυμητή τιμή (3 και 4).

Το αρχείο αυτό καλεί τη συνάρτηση *ask_nyq_filter_nogray.m*.

4. Υπολογισμός παραμέτρων συστήματος

Έχουμε:

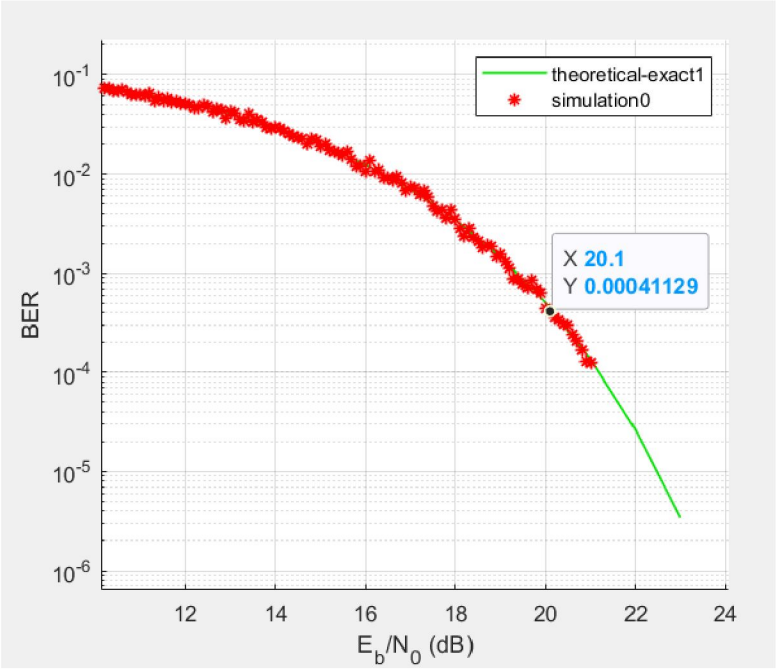
$$\frac{R}{\log_2 L} = \frac{1}{T} \quad \Rightarrow \quad \frac{10 \cdot 10^6}{\log_2 16} = \frac{1}{T} \quad \Rightarrow \quad \frac{1}{T} = \frac{10}{4} \cdot 10^6 = 2.5M\ symbols/sec\ \ Baud\ Rate$$

$$W = \frac{1}{2T}(1+a) \quad \Rightarrow \quad 2 \cdot 10^6 = \frac{10 \cdot 10^6}{2 \cdot 4}(1+a) \quad \Rightarrow \quad 1 + a = \frac{16}{10} \quad \Rightarrow$$

$$\Rightarrow \quad a = 0.6\ rolloff\ factor\ ,\ \text{δεκτό αφού } 0<\alpha<1$$

$$P_{bit\_error} = \frac{BER}{R} \leq \frac{4 \cdot 10^3}{10 \cdot 10^6} = 0.0004$$

Τρέχουμε στο bertool προσομοιώσεις της *ask_ber_func_nyq.m* για rolloff=0.6 και παρατηρούμε ότι με Eb/N0 περίπου 20.1 παίρνουμε την τιμή αυτή:

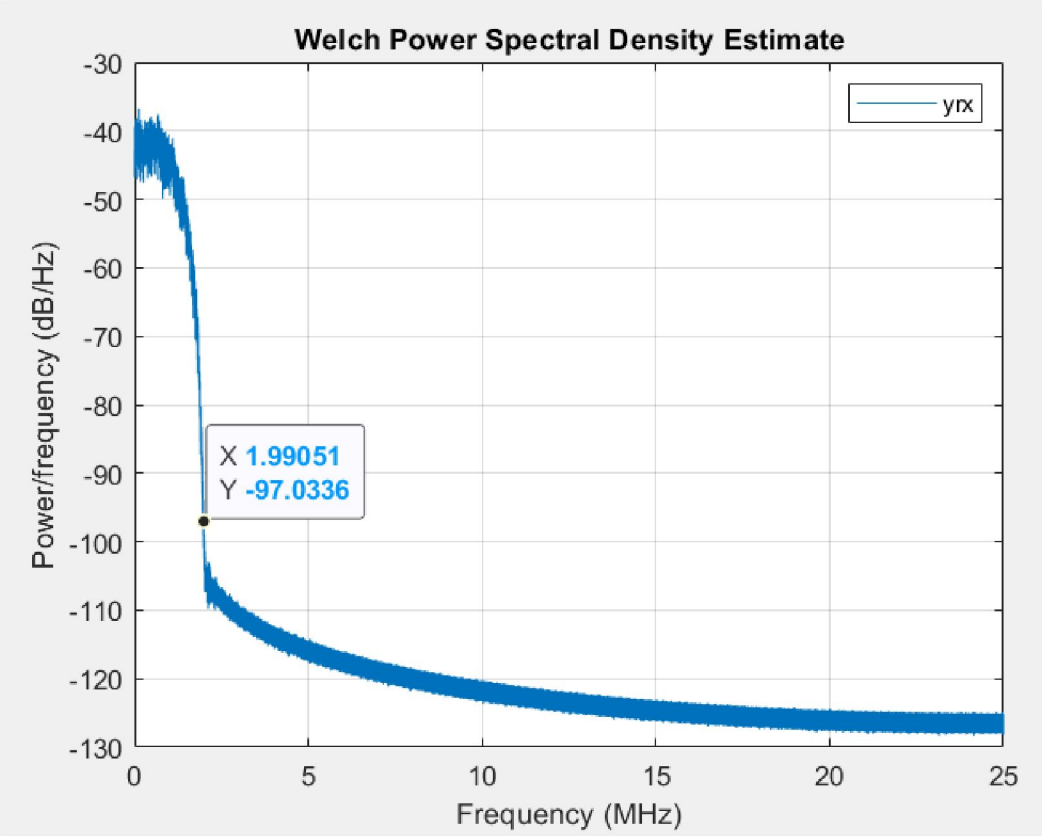


Αλλιώς θα μπορούσαμε μέσω της θεωρητικής καμπύλης να βρούμε το EbNo στο οποίο είναι Pe=0.0004 και μετά να υπολογίσουμε το Pe πειραματικά με το EbNo αυτό. Δοκιμάζοντάς το πολλές φορές, με EbNo=20.1, προέκυπταν τιμές μεταξύ 0.0003 και 0.0005.

Με Eb/N0 τουλάχιστον 20.1 για να έχουμε BER εντός των προδιαγραφών, απαιτείται ενέργεια bit:

$$\frac{E_b}{N_0} = 10^{\frac{20.1}{10}} \quad \Rightarrow \quad E_b = 10^{\frac{20.1}{10}} \cdot 80 \cdot 10^{-9} = 8.186 \cdot 10^{-6}\, Joule$$

Τέλος, για να ελέγξουμε το εύρος ζώνης, τρέχουμε το αρχείο *ask4_4.m* (ίδιο με την πρώτη άσκηση, αλλά με νέες τιμές των παραμέτρων `rollof=0.6` και `Fs=R/k*nsamp`, και επιπλέον υπολογισμό του Perror)



Κώδικας για την άσκηση 4: αρχείο *ask4_4.m*

```
clear all; close all;
L=16; step=2; k=log2(L); nsamp=20; Nsymb=10000;
R=10000000;
EbNo=20.1;

x=randi([0,1],1,Nsymb*k);

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 = 6;
filtorder = delay*nsamp*2;
rolloff = 0.6;
rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
Fs=R/k*nsamp;
% ----------------------
y=upsample(y1,nsamp);
ytx = conv(y,rNyquist);

SNR=EbNo-10*log10(nsamp/2/k);
Py=10*log10(ytx*ytx'/length(ytx));
Pn=Py-SNR;
n=sqrt(10^(Pn/10))*randn(1,length(ytx));
ynoisy=ytx+n;
% ----------------------
yrx=conv(ynoisy,rNyquist);
yrx = yrx(2*nsamp*delay+1:end-2*nsamp*delay);
yr = downsample(yrx,nsamp);
figure; pwelch(yrx,[],[],[],Fs);
legend('yrx');

xr=[];
for i=1:length(yr)
    [m,j]=min(abs(mapping-yr(i)));
    xr=[xr de2bi(j-1,k,'left-msb')];
end
errors=sum(not(x==xr));
Perror=errors/Nsymb/k
```