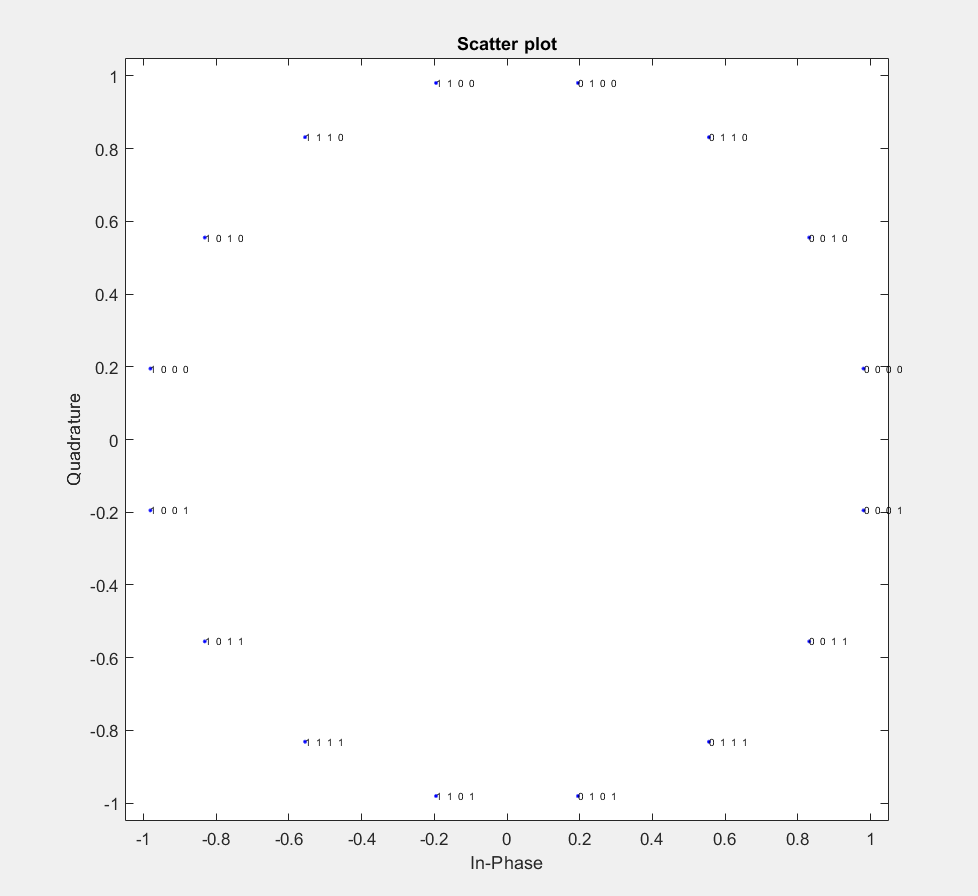
**Ψηφιακές Επικοινωνίες Ι – Αναφορά 5ης Εργ. Άσκησης**

Ζευγολατάκος Παναγιώτης

ΑΜ: 03117804

Μέρος 1ο:



Παρατίθεται ο κώδικας Matlab που χρησιμοποιήθηκε:

|  |
| --- |
| clear all;  close all;  clc;    % k is the number of bits per symbol  % mapping is the vector of psk points, in the gray-coding order  % i.e. mapping(1)<->00...00, mapping(2)<->00...01,  % mapping(3)<->00...10, ..  % For 16-PSK, set k=4;    k=4;  ph1=pi/4;  theta=[ph1; -ph1; pi-ph1; -pi+ph1];  mapping=exp(1j\*theta); % τετριμμένη κωδικοποίηση, M=4  if(k>2)  for j=3:k  theta=theta/2; % υποδιπλασιασμός των γωνιών  temp\_theta=pi-theta;  for i=1:2^(j-1)  if (temp\_theta(i)>pi) % αν η γωνία είναι μεγαλύτερη από π  temp\_theta(i)=temp\_theta(i)-2\*pi;  % -2π, ώστε η abs να είναι < π  end  end  theta=[theta; temp\_theta]; % συμμετρικές γωνίες  mapping=exp(1j\*theta);  end  end    scatterplot(mapping);  hold on;    for i=1:2^k  text(cos(theta(i)),sin(theta(i)),num2str(de2bi(i-1,k,'left-msb')),'FontSize', 6);  end |

Μέρος 2ο:

Έχουμε πως:

R = 12Mbps

W = 9 – 6 = 3MHz 🡪 fc = 7.5MHz

k = log2M ≥ (1+α) = 4(1+α)

Εφόσον 0 < α ≤ 1 ισχύει πως:

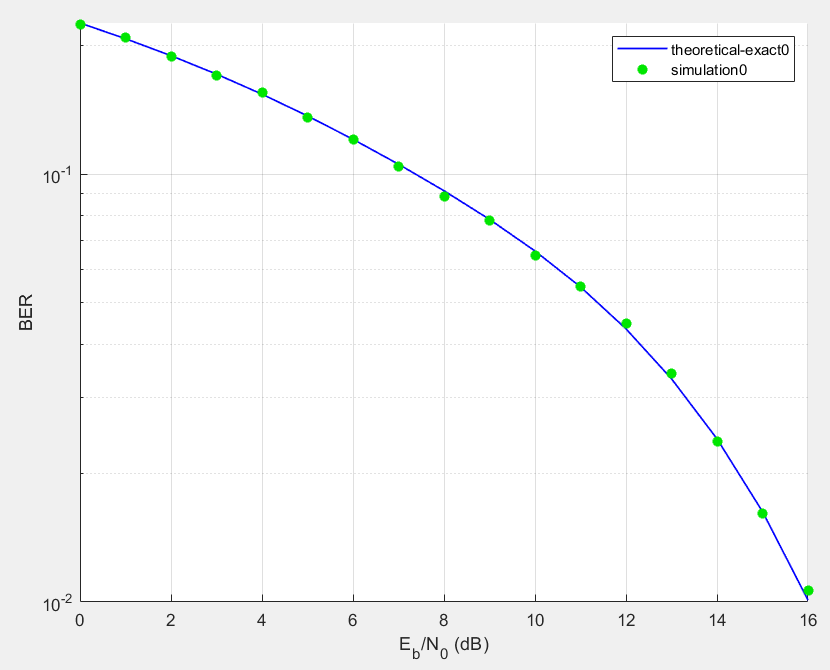
kmin = 5, Mmin = 32, = = 2.4·106

Από τον τύπο: W = (1+α) παίρνουμε πως α = 0.25

Για τη συχνότητα δειγματοληψίας έχουμε πως Fs =

Επίσης ισχύει πως: Fs ≥ 2R = 19.2Mbps 🡪 nsamp ≥ 8

Χρησιμοποιώντας τα παραπάνω, παίρνουμε την παρακάτω γραφική παράσταση (θεωρητική και πειραματική σχεδίαση της καμπύλης Pb 🡨🡪Eb/No):



Παρατίθεται ο κώδικας Matlab που χρησιμοποιήθηκε:

|  |
| --- |
| function [ber,numBits] = lab5\_ber\_func(EbNo, maxNumErrs, maxNumBits)  % 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    % ?. --- Set up parameters. ---  % --- INSERT YOUR CODE HERE.  k=4; % number of bits per symbol  Nsymb=10000; % number of symbols in each run  nsamp=8; % oversampling,i.e. number of samples per T    ph1=pi/4;  theta=[ph1; -ph1; pi-ph1; -pi+ph1];  mapping=exp(1j\*theta); % τετριμμένη κωδικοποίηση, M=4  if(k>2)  for j=3:k  theta=theta/2; % υποδιπλασιασμός των γωνιών  temp\_theta=pi-theta;  for i=1:2^(j-1)  if (temp\_theta(i)>pi) % αν η γωνία είναι μεγαλύτερη από π  temp\_theta(i)=temp\_theta(i)-2\*pi;  % -2π, ώστε η abs να είναι < π  end  end  theta=[theta; temp\_theta]; % συμμετρικές γωνίες  mapping=exp(1j\*theta);  end  end  % 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 (BERTool.getSimulationStop)  break;  end  % ?. --- INSERT YOUR CODE HERE.  errors=lab5\_2\_17804(k,mapping,Nsymb,nsamp,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; |
| function errors=lab5\_2\_17804(k,mapping,Nsymb,nsamp,EbNo)    %EbNo=15;  fc=7.5\*10^6;  W=3\*10^6;  rolloff=0.25;  T=(1+rolloff)/W;  Fs=nsamp/T;  group\_delay=8;  filtorder=2\*group\_delay\*nsamp;  L=2^k;  SNR=EbNo-10\*log10(nsamp/k/2);    x=floor(2\*rand(k\*Nsymb,1)); % τυχαία δυαδική ακολουθία    xsym=bi2de(reshape(x,k,length(x)/k).','left-msb');  y1=[];  for i=1:length(xsym)  y1=[y1; mapping(xsym(i)+1)];  end    % Πομπός  rNyquist= rcosine(1,nsamp,'fir/sqrt',rolloff,group\_delay);  y=upsample(y1,nsamp); % υπερδειγμάτιση  ytx = conv(y,rNyquist); % εφαρμογή φίλτρου Nyquist  m=(1:length(ytx))';  s=real(ytx.\*exp(1j\*2\*pi\*fc\*m/Fs));  %pwelch(s,[],[],[],Fs);    snoisy=awgn(s,SNR,'measured'); % εφαρμογή θορύβου    % Δέκτης  yrx=2\*snoisy.\*exp(-1j\*2\*pi\*fc\*m/Fs);  yrx=conv(yrx,rNyquist); % εφαρμογή φίλτρου Nyquist  yrx = downsample(yrx,nsamp); % υποδειγμάτιση  yrx = yrx(filtorder/nsamp+1:end-filtorder/nsamp);      xr=[];  for i=1:length(yrx)  [m,j]=min(abs(mapping-yrx(i)));  xr=[xr; de2bi(j-1,k,'left-msb')'];  end  errors=sum(not(x==xr));  end |

Μέρος 3ο:

(Ο κώδικας που χρησιμοποιείται εδώ είναι ο παραπάνω κώδικας, μόνο για τη συγκεκριμένη περίπτωση EbNo=15, με τη μόνη αλλαγή 🡪 k=4)

Χρησιμοποιούμε PSK μικρότερης τάξης, άρα k=4 και παίρνουμε τα αποτελέσματα:



Με:



Από τα 40000 σύμβολα που στέλνουμε, τα 18 παρουσιάζουν σφάλμα, επομένως η πιθανότητα εσφαλμένου ψηφίου είναι ίση με = 0.00045 < 0.001 (ζητούμενο).

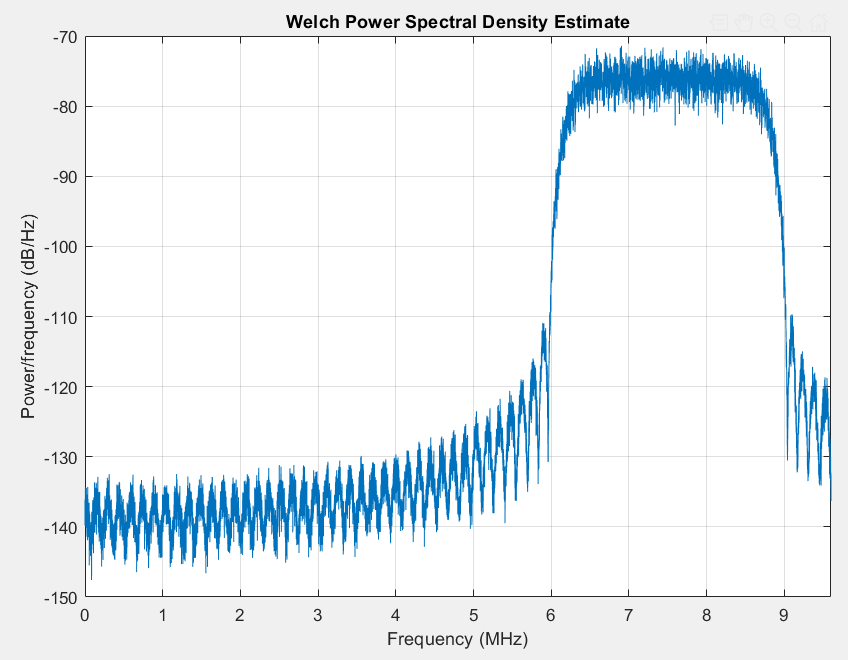
Ισχύει πως: R ≤ = 9.6MHz

Άρα ο μέγιστος ρυθμός μετάδοσης είναι R=9.6MHz.

Χρησιμοποιούμε την εντολή:

pwelch(s,[],[],[],Fs);

Προκειμένου να δούμε την ισχύ του σήματος s, πριν προστεθεί ο AWGN. Παρατηρούμε πως δεν υπάρχουν διαφοροποιήσεις σε σχέση με το προηγούμενο ερώτημα που είχαμε Μ = 32· το εύρος ζώνης παραμένει 6-9MHz με κεντρική συχνότητα fc = 7.5MHz:



Μέρος 4ο:

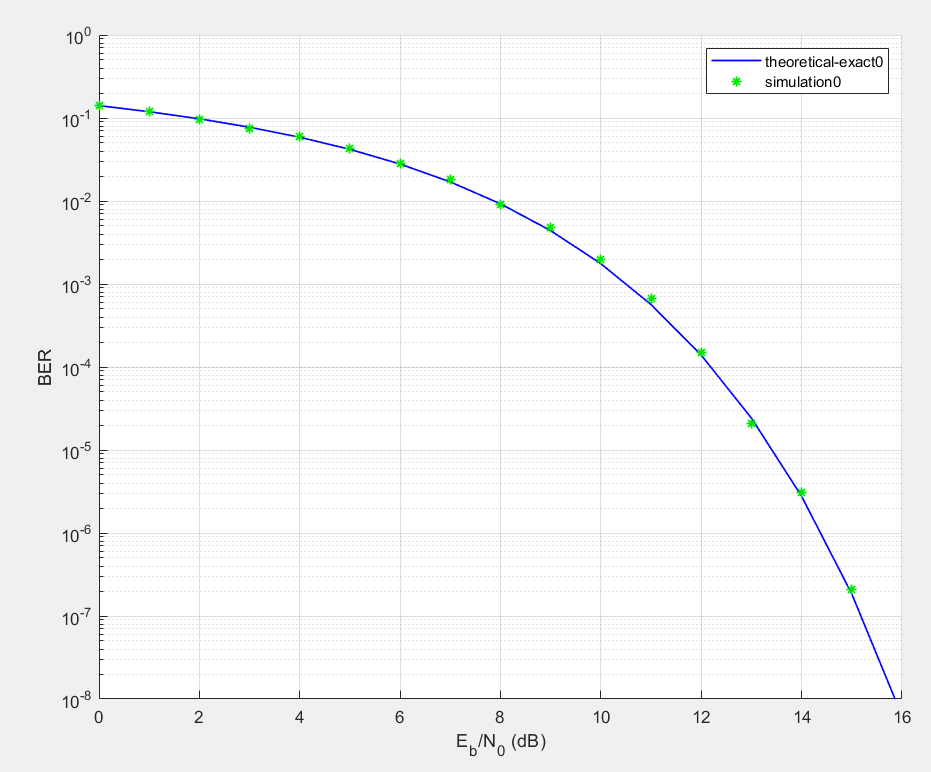
Ισχύει πως:

R ≤ = = = 10.286Mbps

Επομένως, ο ρυθμός μετάδοσης μπορεί να αυξηθεί κατά 0.686MHz για roll-off μειωμένο κατά .

Μέρος 5ο:

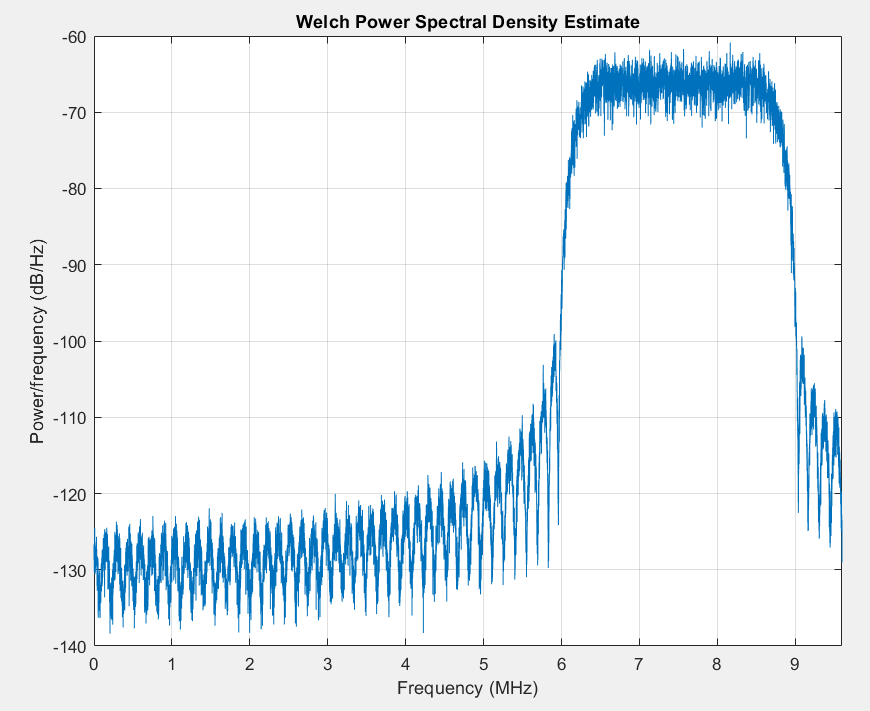
Εξομοιώνουμε 16-QAM (θεωρητική και πειραματική καμπύλη):



Παρατηρούμε πως για ίδιες τιμές σηματοθορυβικού λόγου, η πιθανότητα εσφαλμένου ψηφίου είναι μικρότερη από αυτή του συστήματος 16-PSK (μπορούμε αν επιθυμούμε να ελέγξουμε/συγκρίνουμε τις τιμές μέσα από το bertool).

Παρατηρούμε στο παρακάτω διάγραμμα το φάσμα που δημιουργείται με τη χρήση της εντολής:

pwelch(s,[],[],[],Fs);



Το οποίο είναι ίδιο με το εύρος ζώνης του συστήματος 16-PSK, εφόσον το δημιουργήσαμε με τις ίδιες μεταβλητές (κώδικας του αρχείου lab5\_2\_17804 στο ερώτημα 2). Η μόνη διαφορά με το ερώτημα 2 (και κατ’ επέκταση το ερώτημα 3), ουσιαστικά, ήταν το mapping στη συνάρτηση που καλούμε με το bertool. Παρατίθεται ο κώδικας Matlab που χρησιμοποιήθηκε για τη συνάρτηση αυτή:

|  |
| --- |
| function [ber,numBits] = lab5\_5\_ber\_func(EbNo, maxNumErrs, maxNumBits)  % 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    % ?. --- Set up parameters. ---  % --- INSERT YOUR CODE HERE.  k=4; % number of bits per symbol  Nsymb=10000; % number of symbols in each run  nsamp=8; % oversampling,i.e. number of samples per T    l=2;  core=[1+1i;1-1i;-1+1i;-1-1i];  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  % 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 (BERTool.getSimulationStop)  break;  end  % ?. --- INSERT YOUR CODE HERE.  errors=lab5\_2\_17804(k,mapping,Nsymb,nsamp,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; |