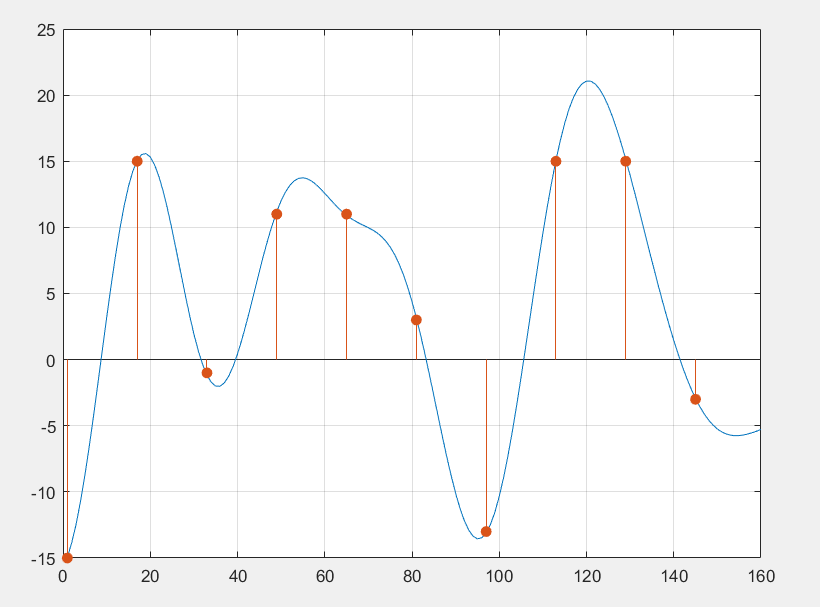
**Ψηφιακές Επικοινωνίες Ι – Αναφορά 4ης Εργ. Άσκησης**

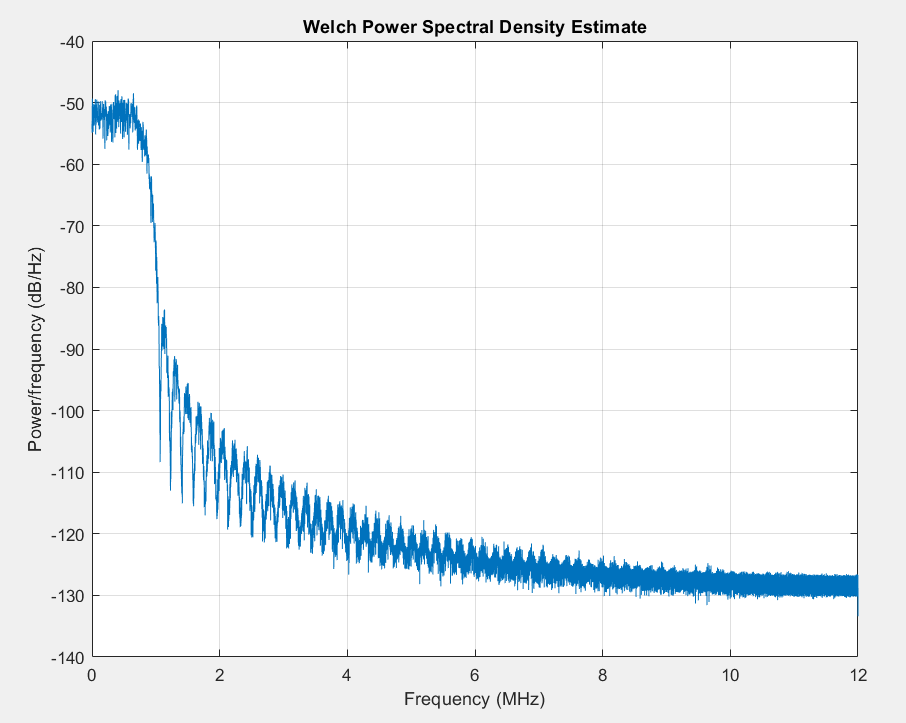
Ζευγολατάκος Παναγιώτης

ΑΜ: 03117804

Μέρος 1ο: Παραγωγή σήματος με φίλτρα Nyquist – Διαγράμματα χρόνου και συχνότητας

Παρατίθενται οι γραφικές παραστάσεις που προέκυψαν:





Στην πρώτη, βλέπουμε πως στην έξοδο του προσαρμοσμένου φίλτρου του δέκτη φαίνονται τα ολισθημένα επίπεδα πλατών που στέλνει ο πομπός, λόγω της καθυστέρησης που εισάγουν τα δύο φίλτρα.

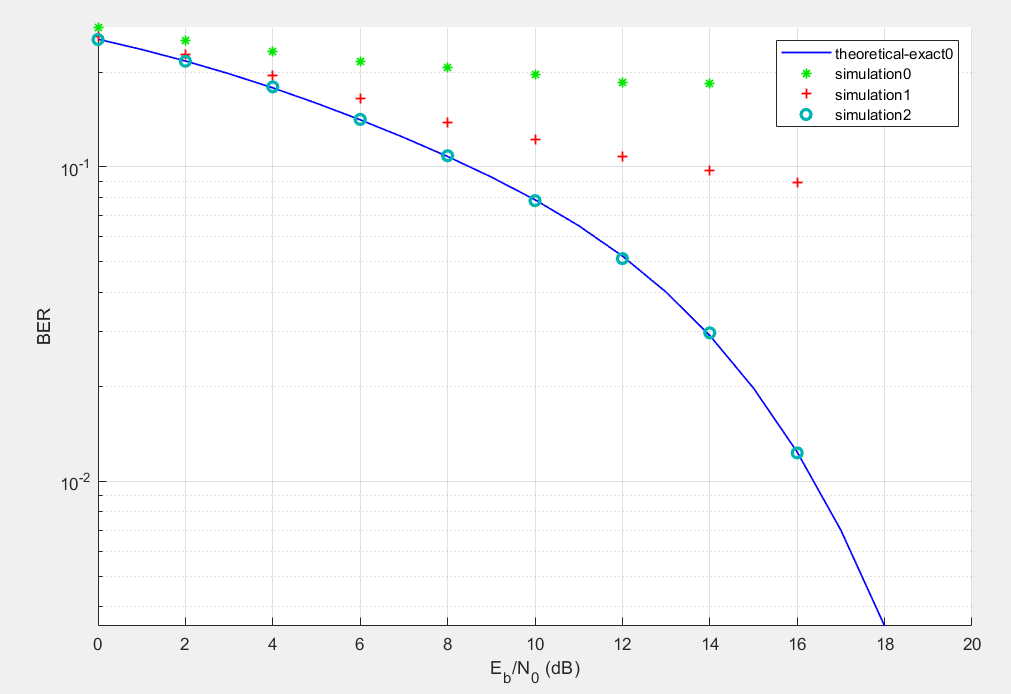
Στην δεύτερη, βλέπουμε τη φασματική πυκνότητα ισχύος στην είσοδο του δέκτη. Παρατηρούμε πως με βαθυπερατό φίλτρο στην έξοδο του πομπού αποκόπτονται οι υψηλές συχνότητες.

Παρατίθεται ο κώδικας Matlab που χρησιμοποιήθηκε:

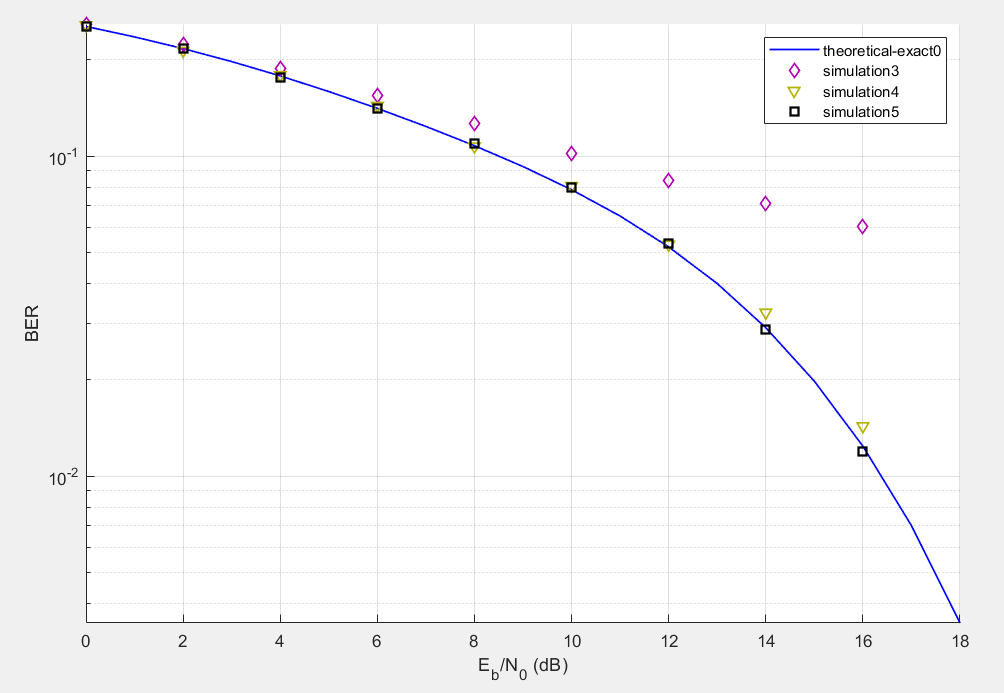
|  |
| --- |
| EbNo=18;  Nsymb=2500;  nsamp=16;  L=16;  k=log2(L);  x = randi([0, 1], 1, k\*Nsymb);  delay=4;  filtorder=128; % delay\*nsamp\*2  rolloff=0.35;  rNyquist = rcosine(1,nsamp,'fir/sqrt',rolloff,delay);    step=2; % ο αριθμός των πλατών και το βήμα μεταξύ τους  mapping=[step/2; -step/2];  if(k>1)  for j=2:k  mapping=[mapping+2^(j-1)\*step/2; ...  -mapping-2^(j-1)\*step/2];  end  end  temp=reshape(x,k,length(x)/k).';  xsym=bi2de(temp,'left-msb');  y=[];  for i=1:length(xsym)  y=[y mapping(xsym(i)+1)];  end  yt=upsample(y,nsamp);  ytx = conv(yt,rNyquist);  yrx=conv(ytx,rNyquist);    yrx=yrx((2\*delay\*nsamp)+1:end-(2\*delay\*nsamp));  plot(yrx(1:10\*nsamp)); hold; % 10Τ  stem((1:nsamp:10\*nsamp), y(1:10),'filled'); % ζητούμενη υπέρθεση  grid;  figure();  pwelch(ytx, [], [], [], 1); |

Μέρος 2ο: Υπολογισμός επίδοσης BER vs Eb/No

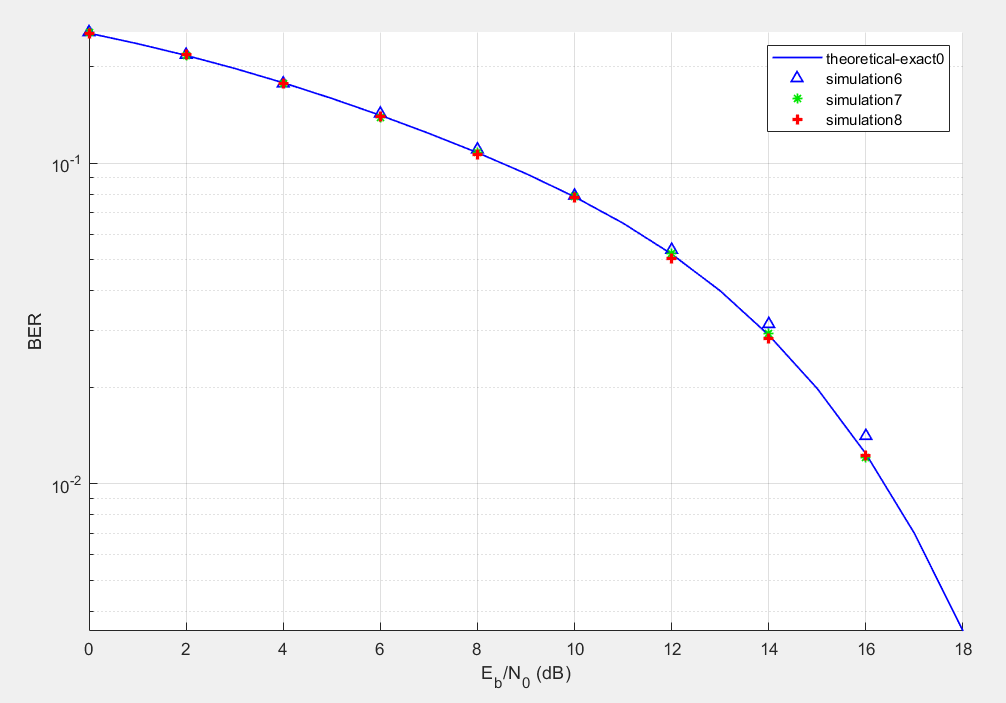
(Ι)



(ΙΙ)



(ΙΙΙ)



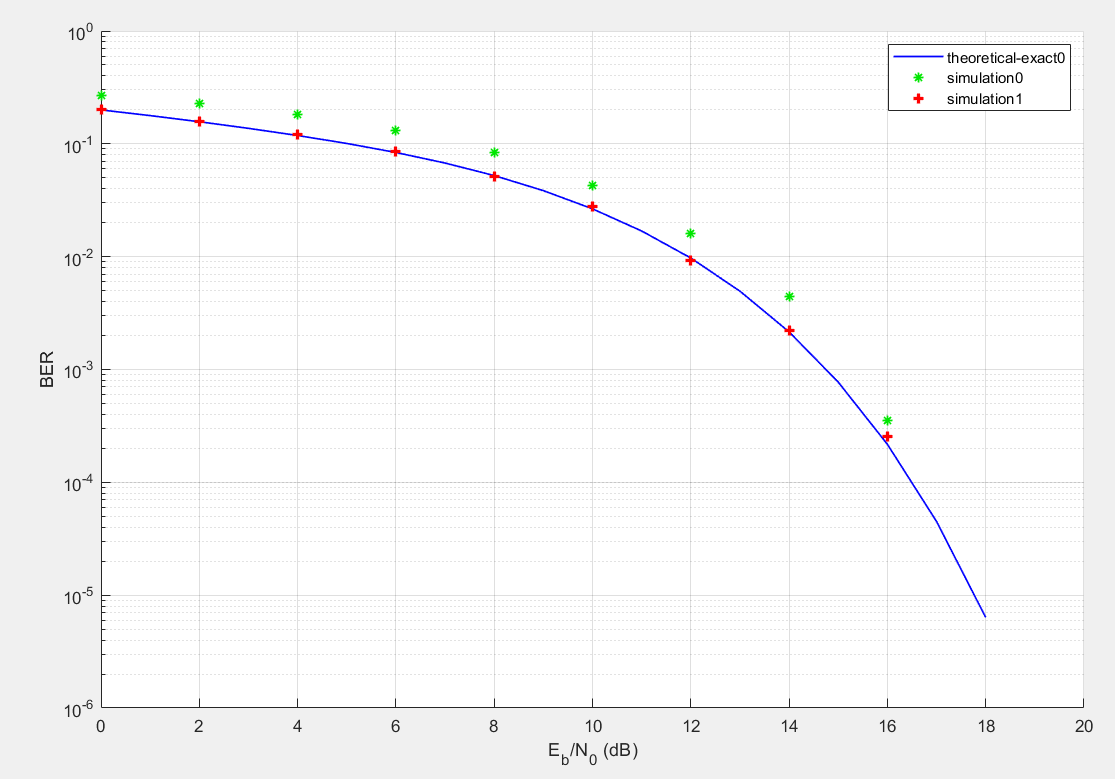
(στις παραπάνω γραφικές τα simulation 0, 3, 6 είναι για τάξη φίλτρου 64, τα simulation 1, 4, 7 για τάξη φίλτρου 128 και τα simulation 2, 5, 8 είναι για τάξη φίλτρου 512)

Παρατηρούμε πως όταν χρησιμοποιούμε μικρότερο εύρος ζώνης (roll-off) πρέπει να χρησιμοποιήσουμε και φίλτρο μεγαλύτερης τάξης για να πάρουμε αποτέλεσμα που να προσεγγίζει το θεωρητικό. Συνεπώς, χρειαζόμαστε μεγαλύτερο delay, εφόσον οι κυματώσεις της απόκρισης φίλτρου είναι μεγαλύτερες σε σχέση με το φίλτρο με μεγαλύτερο εύρος ζώνης (roll-off).

Παρατίθεται ο κώδικας Matlab που χρησιμοποιήθηκε:

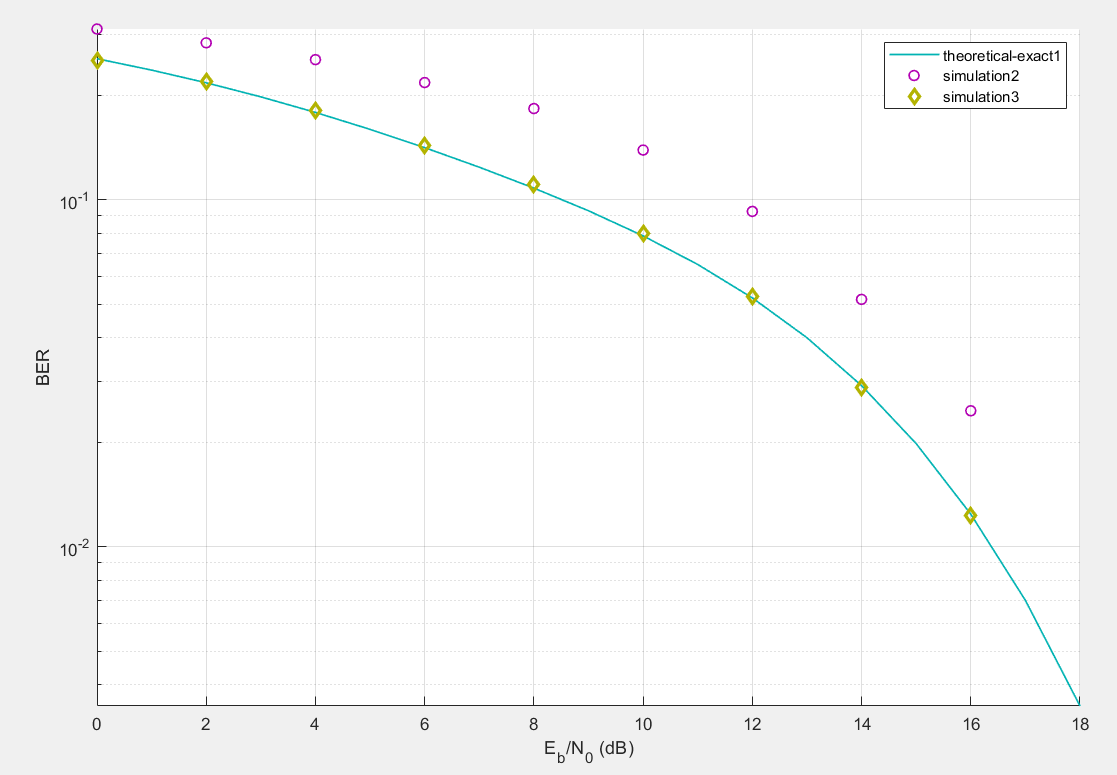
|  |
| --- |
| function [ber,numBits] = lab4\_2\_ber\_func(EbNo, maxNumErrs, maxNumBits)  % Import Java class for BERTool.  import com.mathworks.toolbox.comm.BERTool;  % Initialize variables related to exit criteria.  totErr = 0; % Number of errors observed  numBits = 0; % Number of bits processed  % Α. --- Set up parameters. ---  % --- INSERT YOUR CODE HERE.  k=4; % number of bits per symbol  Nsymb=2500; % number of symbols in each run  delay=16;  rolloff=0.9;  nsamp=16; % 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 (BERTool.getSimulationStop)  break;  end  % Β. --- INSERT YOUR CODE HERE.  errors=lab4\_ask\_errors(k,Nsymb,nsamp,delay,rolloff,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=lab4\_ask\_errors(k,Nsymb,nsamp,delay,rolloff,EbNo)  % Η συνάρτηση αυτή εξομοιώνει την παραγωγή και αποκωδικοποίηση  % θορυβώδους σήματος L-ASK και μετρά τον αριθμό των εσφαλμένων συμβόλων.  % Επιστρέφει τον αριθμό των εσφαλμένων συμβόλων (στη μεταβλητή errors).  % k είναι ο αριθμός των bits/σύμβολο, επομένως L=2^k -- ο αριθμός των  % διαφορετικών πλατών  % Nsymb είναι ο αριθμός των παραγόμενων συμβόλων (μήκος ακολουθίας L-ASK)  % nsamp ο αριθμός των δειγμάτων ανά σύμβολο (oversampling ratio)  % EbNo είναι ο ανηγμένος σηματοθορυβικός λόγος Eb/No, σε db    x=randi([0, 1], 1,k\*Nsymb); %random bit generation  L=2^k; % L=16  filtorder=delay\*nsamp\*2;  rNyquist = rcosine(1,nsamp,'fir/sqrt',rolloff,delay);    step=2; % ο αριθμός των πλατών και το βήμα μεταξύ τους  mapping=[step/2; -step/2];  SNR=EbNo-10\*log10(nsamp/2/k);  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');  y=[];  for i=1:length(xsym)  y=[y mapping(xsym(i)+1)];  end    yt=upsample(y,nsamp);  ytx = conv(yt,rNyquist); % στην έξοδο τον πομπό    ynoisy = awgn(ytx, SNR, 'measured'); % κατά την εκπομπή    yrx=conv(ynoisy,rNyquist); % στην είσοδο του δέκτη  yrx=yrx(filtorder+1:end-filtorder);  yrx = downsample(yrx,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: Επίδραση του τρόπου κωδικοποίησης: Gray ή άλλη



(Simulation 0: απλή κωδικοποίηση

Simulation 1: κωδικοποίηση Gray σε 8-ASK)



(Simulation 2: απλή κωδικοποίηση

Simulation 3: κωδικοποίηση Gray σε 16-ASK)

Παρατηρούμε από τις παραπάνω γραφικές παραστάσεις πως με κωδικοποίηση Gray έχουμε καλύτερη απόδοση από την απλή.

Παρατίθεται ο κώδικας Matlab που χρησιμοποιήθηκε:

|  |
| --- |
| function [ber,numBits] = lab4\_2\_ber\_func(EbNo, maxNumErrs, maxNumBits)  …  % --- INSERT YOUR CODE HERE.  k=3; % για 8-ASK και k=4; για 16-ASK  Nsymb=2500; % number of symbols in each run  delay=4;  rolloff=0.3;  nsamp=16; % oversampling,i.e. number of samples per T  …  (ο κώδικας για το bertool είναι ο ίδιος με του προηγούμενου ερωτήματος, με εξαίρεση τις παραπάνω αλλαγές) |
| function errors=lab4\_3\_17804(k,Nsymb,nsamp,delay,rolloff,EbNo)    x=randi([0, 1], 1,k\*Nsymb); %random bit generation  L=2^k;  filtorder=delay\*nsamp\*2;  rNyquist = rcosine(1,nsamp,'fir/sqrt',rolloff,delay);    step=2; % ο αριθμός των πλατών και το βήμα μεταξύ τους  mapping=-(L-1):step:(L-1);  SNR=EbNo-10\*log10(nsamp/2/k);    xsym=bi2de(reshape(x,k,length(x)/k).','left-msb');  y=[];  for i=1:length(xsym)  y=[y mapping(xsym(i)+1)];  end    yt=upsample(y,nsamp);  ytx = conv(yt,rNyquist); % στην έξοδο τον πομπό    ynoisy = awgn(ytx, SNR, 'measured'); % κατά την εκπομπή    yrx=conv(ynoisy,rNyquist); % στην είσοδο του δέκτη  yrx = downsample(yrx,nsamp);  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 4ο: Υπολογισμός παραμέτρων συστήματος

Ισχύουν οι τύποι:

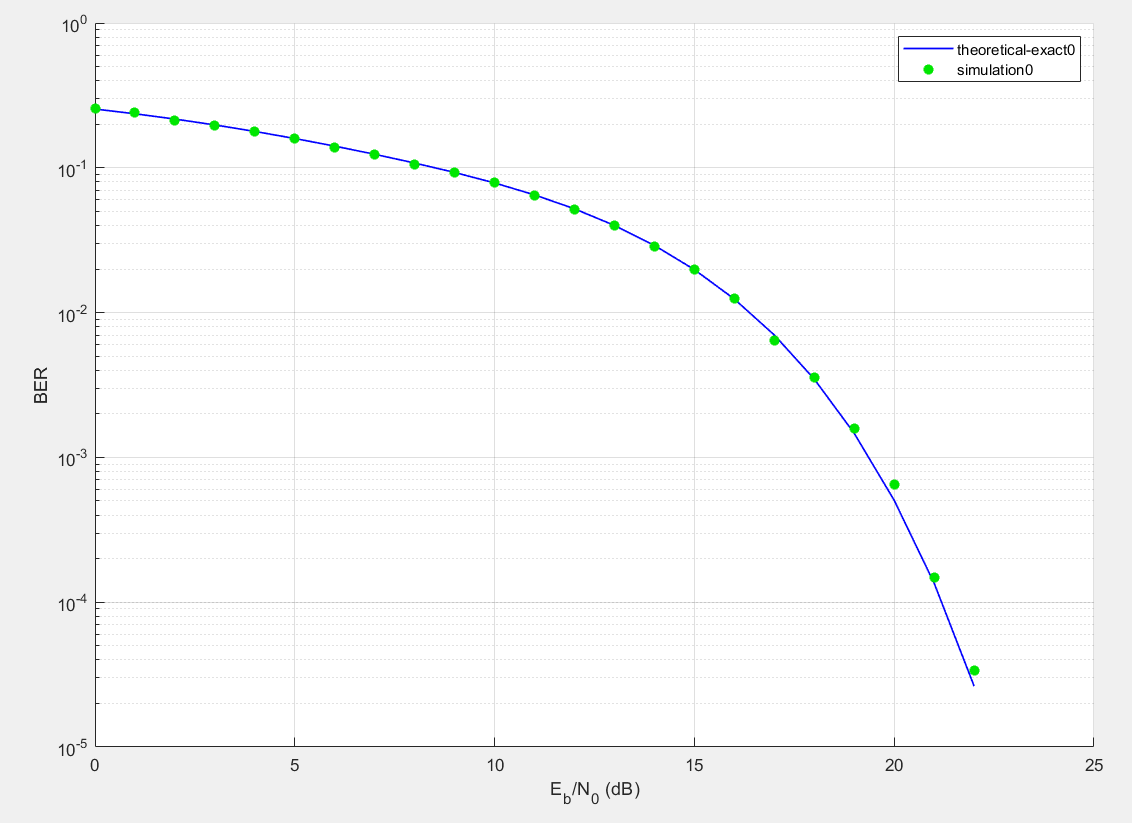
🡪 🡪 log2L =

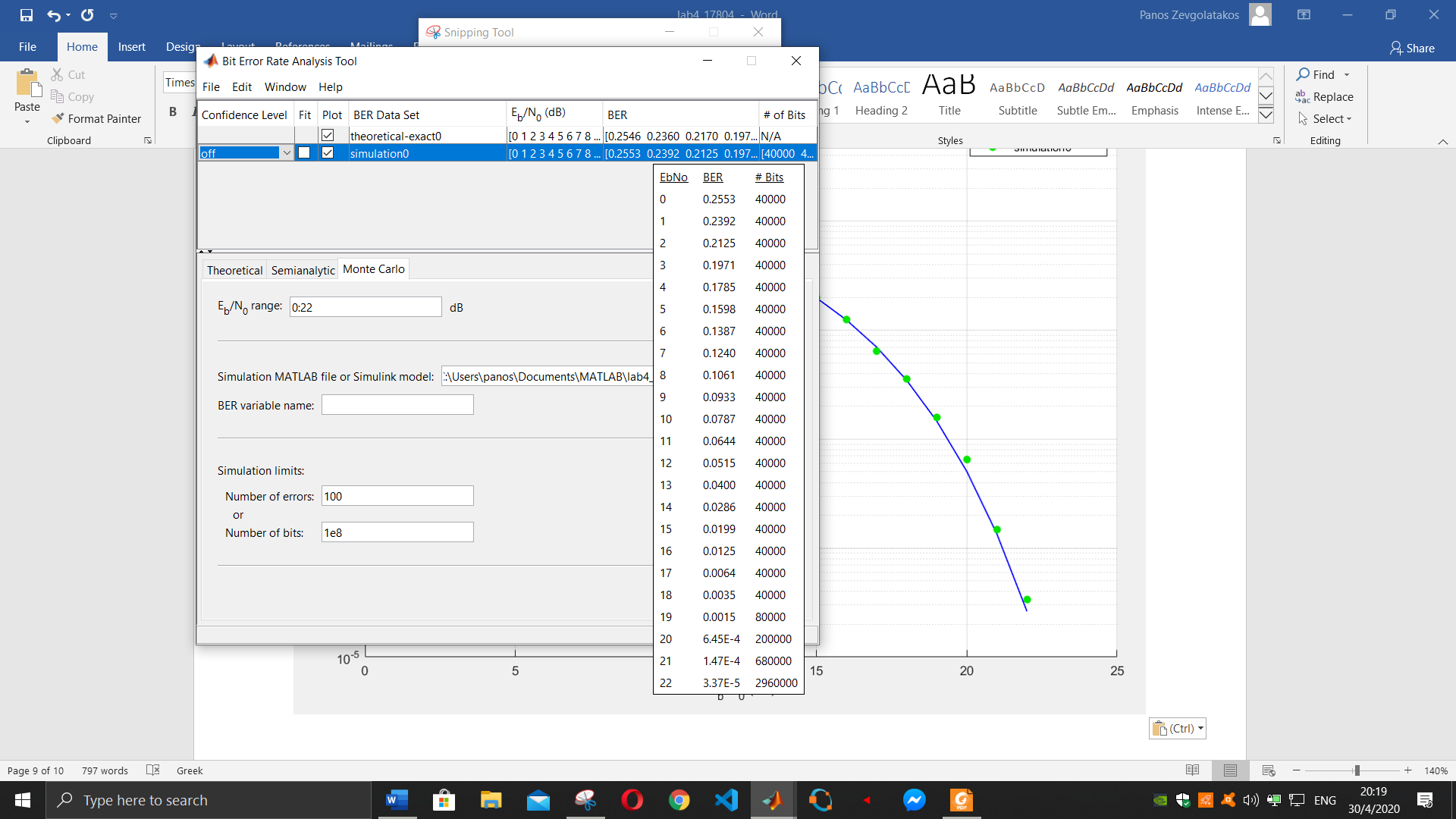
Ισχύει πως Baud Rate: Fb = = = = 1.5Mbaud

Για W=1, = 1.5 🡪 α = 0.333…

Και η πιθανότητα εσφαλμένου ψηφίου είναι: Perr = = 3.3·10-4

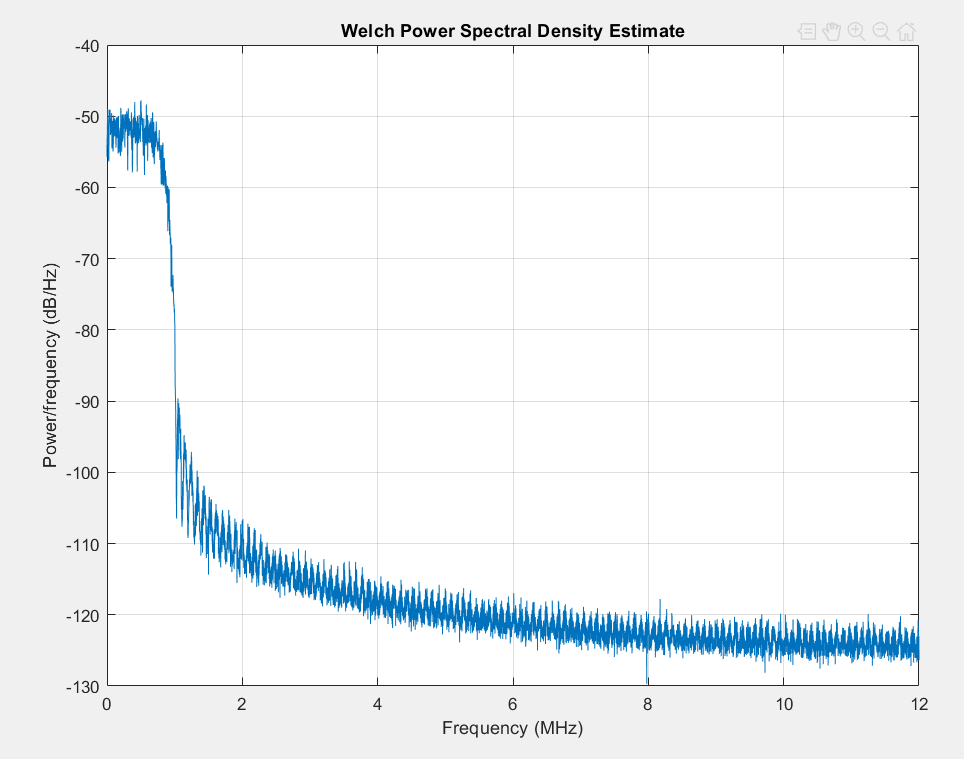
Χρησιμοποιώντας το bertool:





Βρίσκουμε πως η πιθανότητα αυτή ισχύει για Eb/N0>20, επομένως η τιμή SNR μετά του φίλτρο του δέκτη θα πρέπει να είναι SNR>20+10log() = 27.7 dB.

Χρησιμοποιώντας τη συνάρτηση lab4\_ask\_errors με EbNo = 21, επαληθεύουμε το εύρος ζώνης:



Θέτουμε τις παρακάτω τιμές:

k=4;

Nsymb=10000;

delay=8;

rolloff=0.333333;

nsamp=16;

EbNo=21;

Και παίρνουμε το παρακάτω αποτέλεσμα:



Συνεπώς, Perr = = = 3·10-4

Και επομένως BER = Perr·6Mbps = 1.8kbps < 2kbps.