ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εισαγωγή στις Τηλεπικοινωνίες

Εργαστηριακή Άσκηση, Ακαδ. Έτος 2020-21

Βικέντιος Βιτάλης el18803

Εργαστηριακή Άσκησης στο μάθημα Εισαγωγή στις Τηλεπικοινωνίες

Προθεσμία Υποβολής 15/1/2021

Όνομα ομάδας: A Telecom 24

Ερώτημα 1

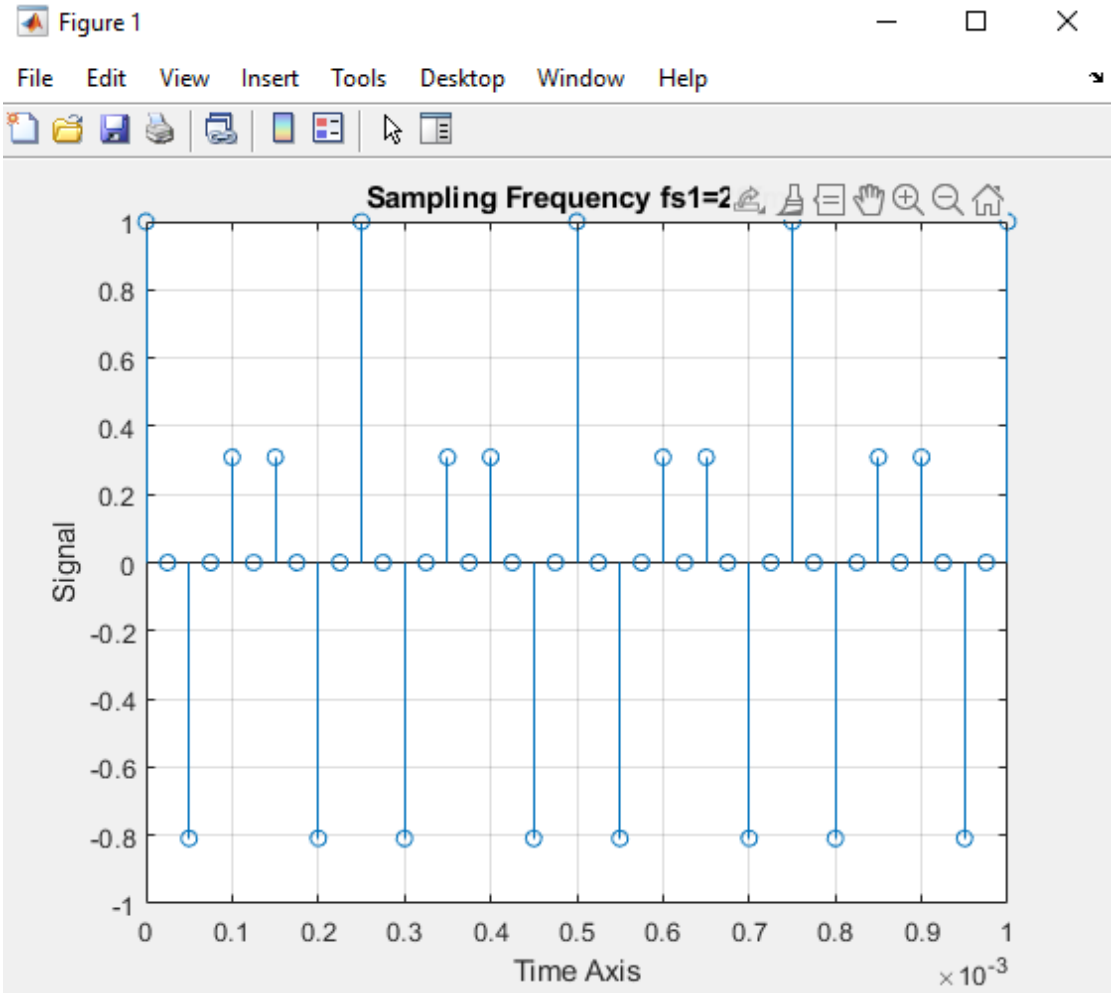
{Questions a(i),a(ii),a(iii)}

Ακολουθεί ο κώδικας ο οποίος παράγει τα σήματα, πραγματοποιεί την δειγματοληψία και τα αντίστοιχα διαγράμματα σύμφωνα με τα δεδομένα της εκφώνησης: A = 1V, am = 3, fm = 2KHz:

```
% Vikentios Vitalis el18803
% fm = 8 + 3 = 11 = 1 + 1 = 2
% am = 3
fm=2000;
am = 3;
fs1=20*fm;
fs2=100*fm;
fs3=5*fm;
Ts1=1/fs1;
Ts2=1/fs2;
Ts3=1/fs3;
dur=1;
Tm=1/fm;
N=dur/Tm;

% Question a(i) - Sample with fs1 = 20fm

for i=1:N
   t_samp1(i)=(i-1)*Ts1;
   x_samp1(i)=cos(2*pi*fm*t_samp1(i))*cos(2*pi*(am+2)*fm*t_samp1(i));
end

% Question a(ii) - Sample with fs2 = 100fm

for i=1:N
   t_samp2(i)=(i-1)*Ts2;
   x_samp2(i)=cos(2*pi*fm*t_samp2(i))*cos(2*pi*(am+2)*fm*t_samp2(i));
end

figure(1)
stem(t_samp1(1:41),x_samp1(1:41));
grid;
xlabel('Time Axis');
ylabel('Signal');
title('Sampling Frequency fs1=20fm');

figure(2)
stem(t_samp2(1:201),x_samp2(1:201));
grid;
xlabel('Time Axis');
ylabel('Signal');
```

```
title('Sampling Frequency fs2=100fm');

figure(3)
stem(t_samp2(1:201),x_samp2(1:201),'g');
hold on;
stem(t_samp1(1:41),x_samp1(1:41),'k');
grid;
xlabel(' Time axis(sec) ');
ylabel(' Signal ');
title('Common Graph Of fs1 and fs2');
legend('fs1=20*fm','fs2=100*fm');
hold off
```

Ο κώδικας παράγει και απεικονίζει κοινό διάγραμμα χρόνου για τα ζητούμενα διακριτά σήματα με συχνότητες δειγματοληψίας fs1=20*fm και fs2=100*fm.



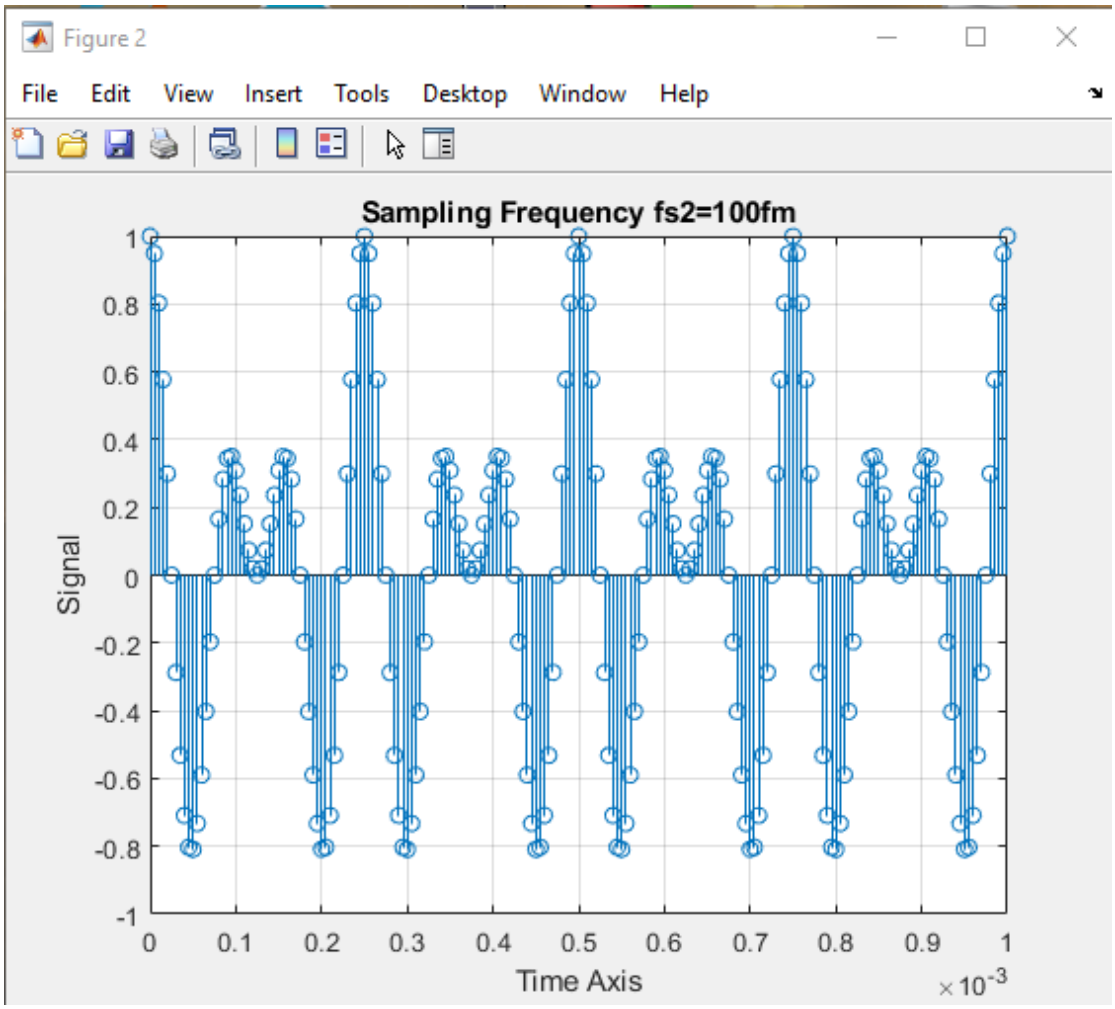
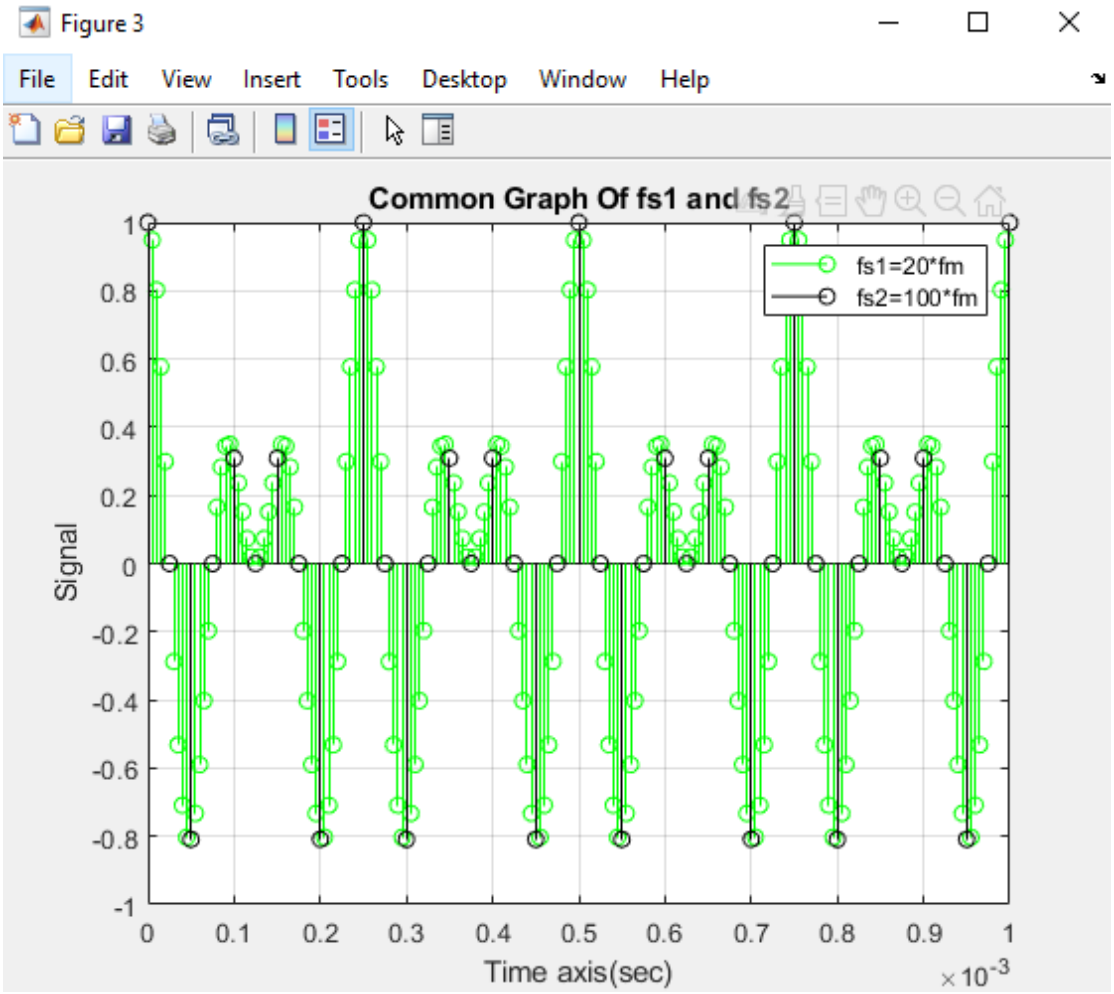


Figure 2
File Edit View Insert Tools Desktop Window Help
Sampling Frequency fs2=100fm
Signal
Time Axis
×10^-3
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1



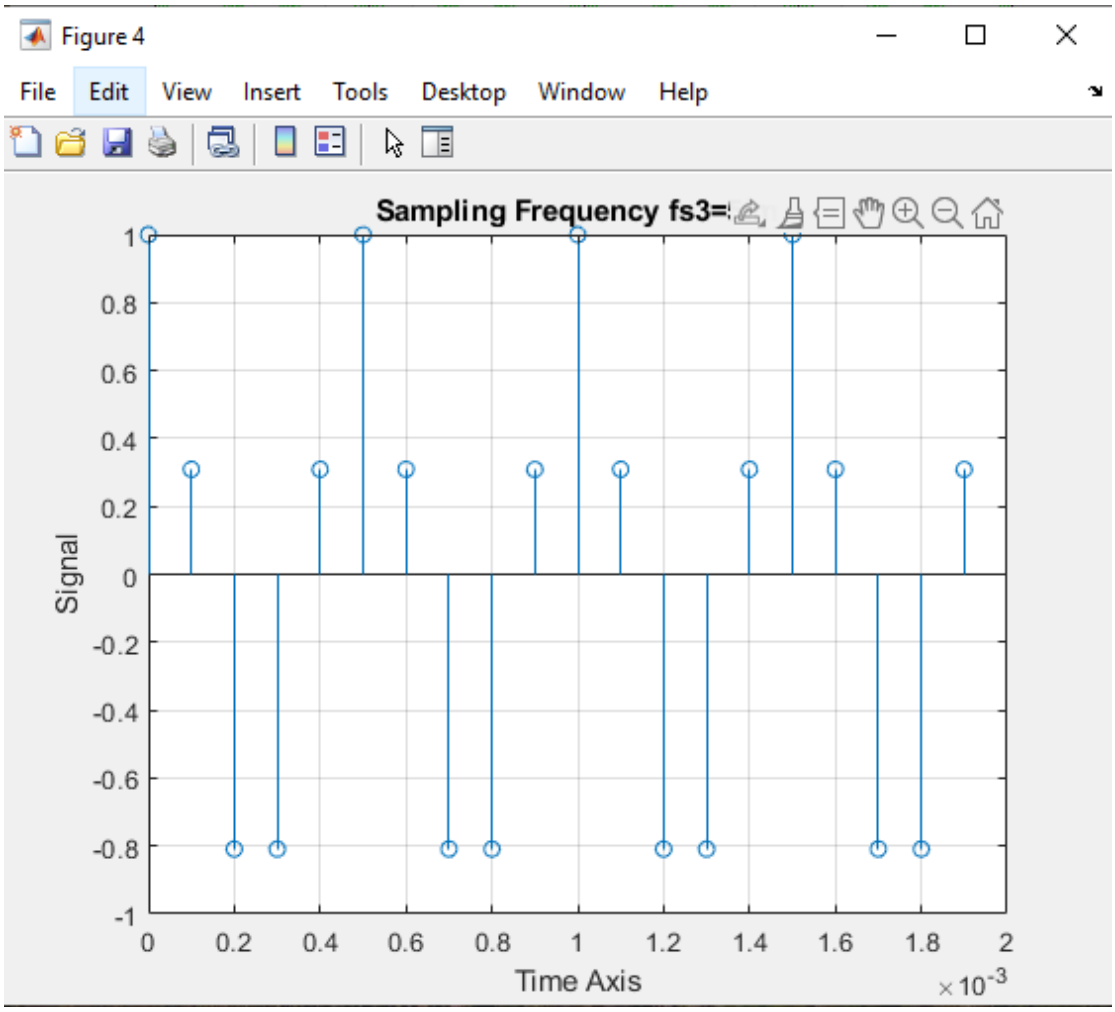
Παρατηρούμε ότι το σήμα με την υψηλότερη συχνότητα δειγματοληψίας κάνει πιο ακριβή απεικόνιση του διακριτού σήματος, γεγονός που ήταν αναμενόμενο. θα πρέπει επίσης, να ικανοποιείται το θεώρημα Shannon για την αποφυγή αναδίπλωσης και την επιτυχή ανακατασκευή του σήματος.

{Question b}

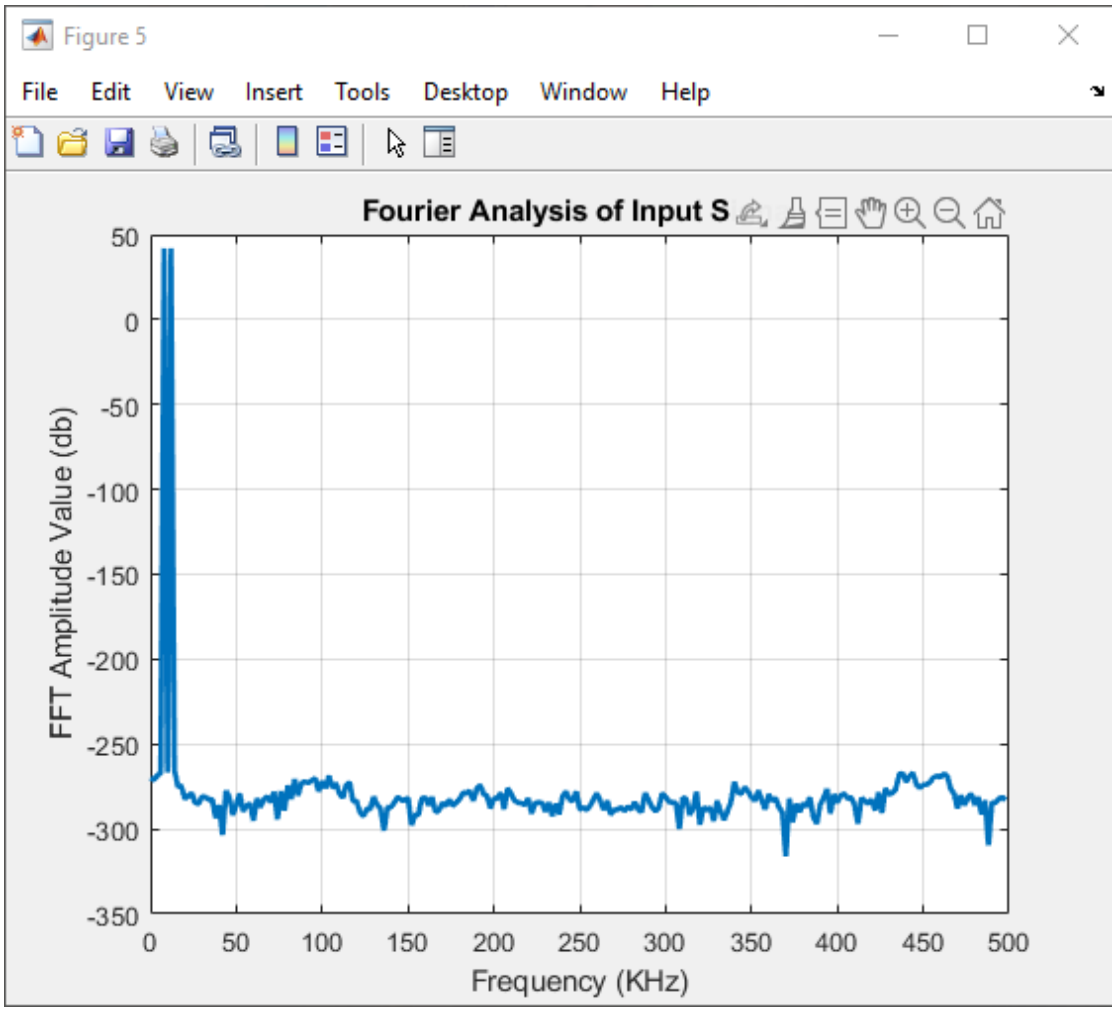
```
% Question b with Fs3=5fm

for i=1:N
   t_samp3(i)=(i-1)*Ts3;
   x_samp3(i)=cos(2*pi*fm*t_samp3(i))*cos(2*pi*(am+2)*fm*t_samp3(i));
end

hf_fs=500*fm;
hf_Ts=1/hf_fs;
hf_N=floor(Tm/hf_Ts);
for i=1:1:hf_N
    t_samp_hf(i)=(i-1)*hf_Ts;
```

```
x_samp_hf(i)=cos(2*pi*fm*t_samp_hf(i))*cos(2*pi*(am+2)*fm*t_samp_hf(i
));
end
x_fft_meas=abs(fft(x_samp_hf));
x_fft_meas_db=20*log10(abs(fft(x_samp_hf)));

% Graphs

figure(4)
stem(t_samp3(1:N/100),x_samp3(1:N/100));
grid;
xlabel('Time Axis');
ylabel('Signal');
title('Sampling Frequency fs3=5fm');

figure(5)
i=1:1:hf_N/2;
plot((i-1)*hf_fs/hf_N/1000,x_fft_meas_db(i),'LineWidth',2);
grid;
xlabel('Frequency (KHz)');
ylabel('FFT Amplitude Value (db)');
title('Fourier Analysis of Input Signal');
```

Πραγματοποιείται δειγματοληψία με συχνότητα fs3 = 5*fm. Εφόσον η συχνότητα είναι αρκετά πιο χαμηλή σε σχέση με πριν, αναμένουμε σημαντική αλλοίωση του σήματος στη διαδικασία αναπαράστασης των δειγμάτων του.



Παρατηρούμε ότι το δειγματοληπτημένο σήμα εμφανίζει σοβαρές αλλοιώσεις ως προς το πλάτος και τις κορυφές του, γεγονός φυσιολογικό λόγω της χαμηλής τιμής της συχνότητας δειγματοληψίας που χρησιμοποιήθηκε. Προκειμένου να γίνει ακριβής εκτίμηση της μέγιστης συχνότητας που το αρχικό αναλογικό σήμα φέρει, έτσι ώστε να χρησιμοποιηθεί το ελάχιστο όριο Nyquist (2*fmax) για τη δειγματοληψία. Το μέτρο του φάσματος για το σήμα έγινε με χρήση πολύ υψηλής συχνότητας δειγματοληψίας (500*fm).



Για το φάσμα έγινε χρήση της συνάρτησης fft(), η οποία υπολογίζει το μετασχηματισμό Fourier διακριτού σήματος. Από τη γραφική παράσταση βλέπουμε ότι αυτό έχει σημαντική πληροφορία μέχρι και συχνότητα f = 13 KHz όπου έχει πέσει σε τιμή περίπου 0 db. Αυτό το όριο μπορεί να θεωρηθεί και ως η μέγιστη συχνότητα του σήματος με βάση την οποία το όριο Nyquist θα ήταν fs = 2*13 KHz = 26 KHz. Με χρήση του θεωρήματος του Nyquist για τη δειγματοληψία συμπεραίνουμε ότι απαιτείται συχνότητα 26KHz όπως φαίνεται και στο παρακάτω διάγραμμα.

Θεωρητικά έχουμε

$$y(t) = \cos(4\pi t)\cos(20\pi t) = \frac{1}{2} * \cos(24\pi t) + \frac{1}{2} * \cos(16\pi t)$$

Σειρά Fourier:

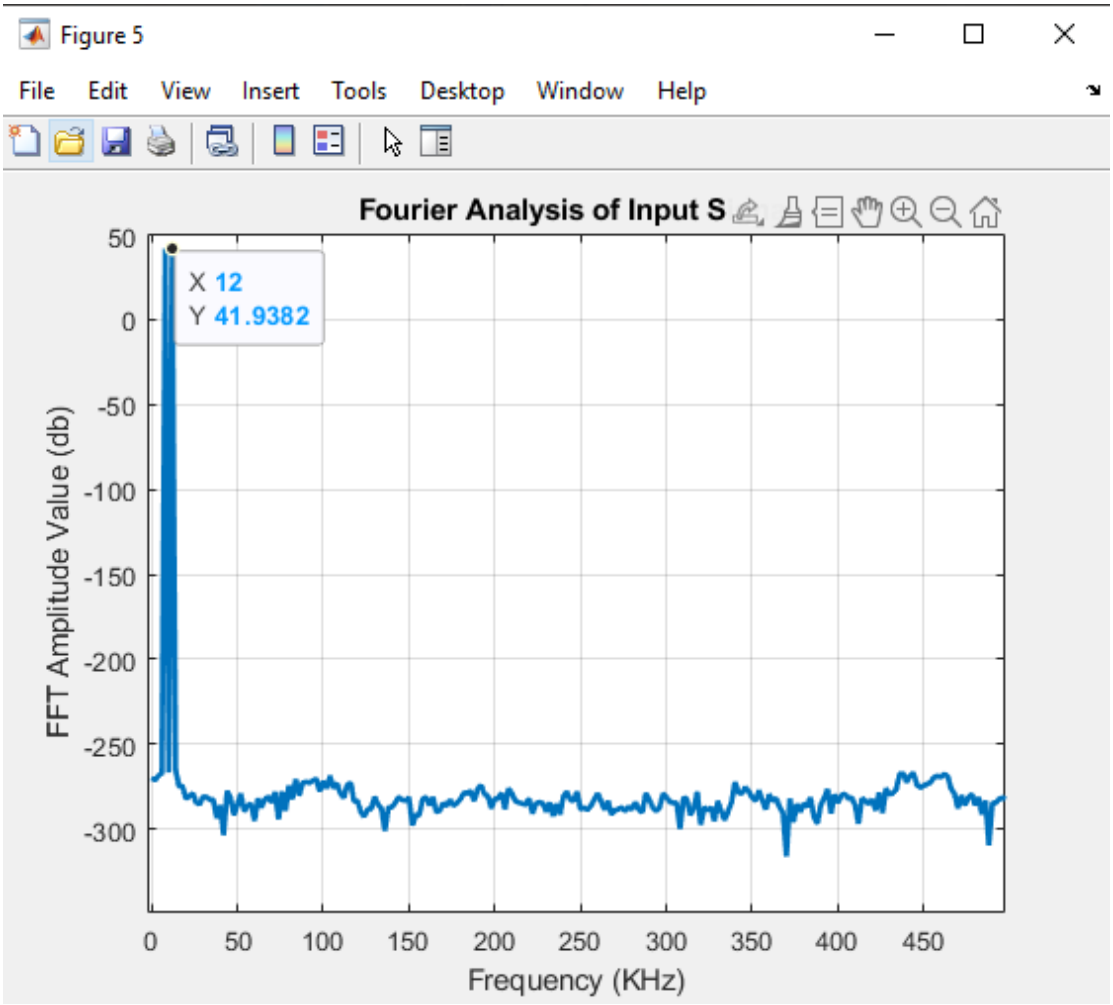
$y(t) = \frac{a_0}{2} + \sum_{k=0}^{n} an * \cos(n\omega t) + bn * sin(n\omega t)$, bn=0, άρτια συμμετρία

Ταυτόχρονα έχουμε fm=2KHz

Άρα: y(t) = $\frac{a0}{2} + \sum_{k=0}^{n} \quad an * sin(n\omega t)$

$$an = \frac{1}{2\pi n} * \int_{0}^{2} \frac{\cos(n\omega t) * \cos(4\pi t) * \cos(20\pi t)}{2} dt$$

Για n = 6 και μετά από πράξεις έχουμε fmax = 2 * 6 =12KHz και με χρήση του θεωρήματος Nyquist για την δειγματοληψία συμπεραίνουμε πως απαιτείται συχνότητα 24KHz, αρκετά κοντινή στον προσεγγιστικό υπολογισμό μέσω του διαγράμματος.



Ερώτημα 2

{Question a}

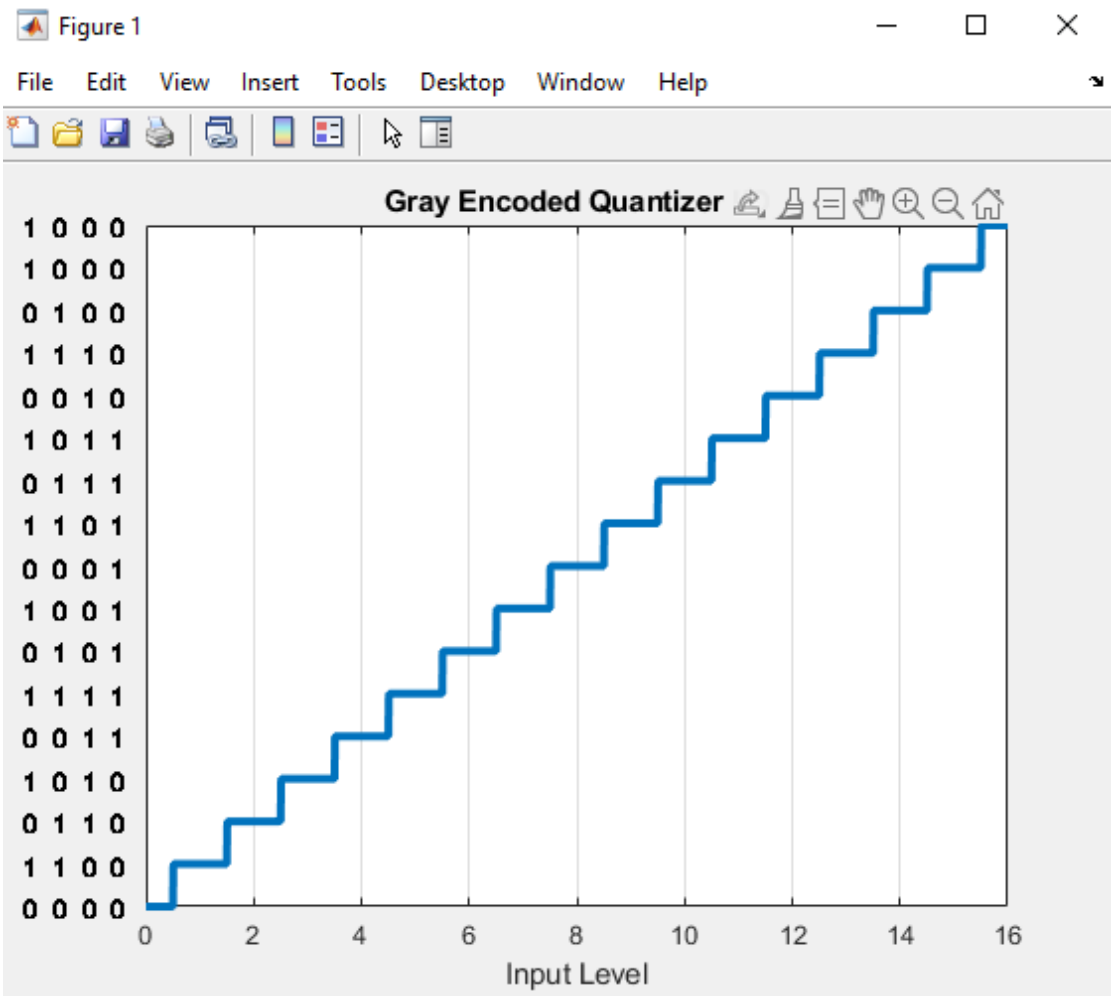
Κάνουμε χρήση του σήματος του προηγούμενου ερωτήματος, το σήμα δειγματοληπτήθηκε με συχνότητα fs=20*fm. Η συχνότητα fm=2 KHz και κβαντιστής των 4 bits, σύφμωνα με την εκφώνηση, λόγω αρτιότητας της συχνότητας.

```
% Vikentios Vitalis el18803
% fm = 8 + 3 = 11 = 1 + 1 = 2
% am = 3
fm=2000;
am=3;
fs1=20*fm;
Ts1=1/fs1;
duration=1;
Tm=1/fm;
N=floor(duration/Tm);

Amin=0;
Amax=16; % 2 ^ n = 2 ^ 4 = 16
n=4; % Logw artiohtas ths syxnothtas

% Question a

for i=1:N
    t_samp1(i)=(i-1)*Ts1;

x_samp1(i)=cos(2*pi*fm*t_samp1(i))*cos(2*pi*(am+2)*fm*t_samp1(i));
end

% Quantizer

delta=(Amax-Amin)/2^n;

partition(1) = Amin+delta/2;
for i=1:1:2^n-1
   partition(i+1) = partition(i)+delta;
end;

x_ind = quantiz(x_samp1,partition);

for i=1:1:N
   if (x_ind(i)+1>2^n)
      x_qnd(i)=partition(2^n)+delta/2;
   else
      x_qnd(i) = partition(x_ind(i)+1)-delta/2;
   end;
end;

k=0;
for i=Amin:0.01:Amax
   k=k+1;
   x_in(k)=i;
   y_ind(k) = quantiz(i,partition);
   if (y_ind(k)+1>2^n)
      y_out(k)=partition(2^n)+delta/2;
```

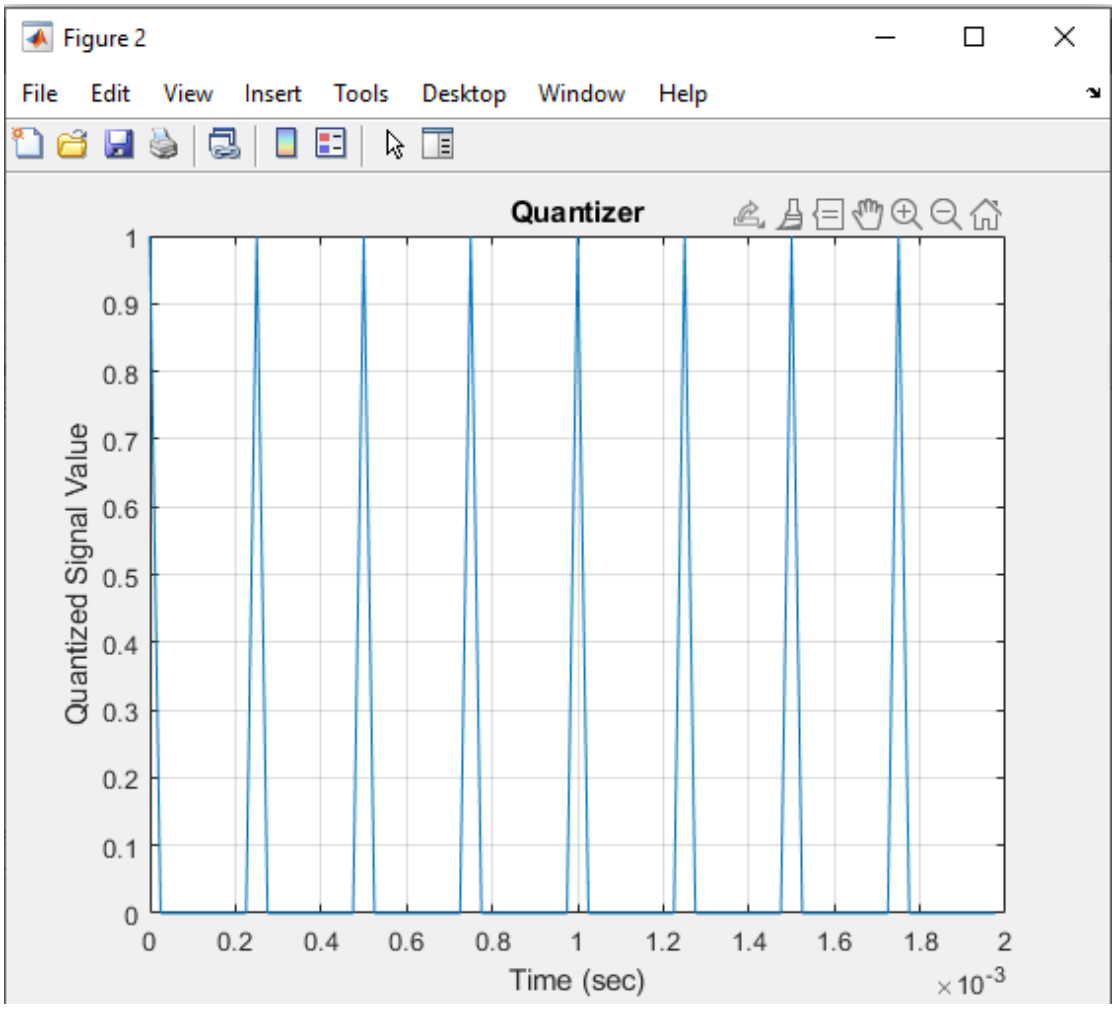
```
    else
        y_out(k)=partition(y_ind(k)+1)-delta/2;
    end;
end;

for i=1:1:k
    if (y_ind(i)<2^n)
        t1=de2bi(y_ind(i),n);
    else
        t1=de2bi(y_ind(i)-1,n);
    end;
    for j=1:1:n
        bin_y_qnd(i,j)=t1(j);
    end;

    t2=bin2gray(t1);
    for j=1:1:n
        gray_y_qnd(i,j)=t2(j);
    end;
end;

x_zeros=zeros(1,k)-2.3;
figure(1)
plot(x_in,y_out,'LineWidth',3);
set(gca,'YTick',[]);
text(x_zeros,y_out,num2str(gray_y_qnd),'FontSize',10);
grid;
xlabel(' Input Level ');
title(' Gray Encoded Quantizer I/O ');

figure(2)
plot(t_samp1(1:1:80), x_qnd(1:1:80));
grid;
xlabel('Time (sec)');
ylabel('Quantized Signal Value');
title('Quantizer');

figure(3)
plot(t_samp1(1:1:80),x_samp1(1:1:80),t_samp1(1:1:80),x_qnd(1:1:80));
grid;
xlabel('Time (sec)');
ylabel('Signal Value');
title('Original Signal - Quantized Signal');
legend('original','quantized');

for i=1:1:N
    if (x_ind(i)<2^n)
        t1=de2bi(x_ind(i),n);
    else
        t1=de2bi(x_ind(i)-1,n);
    end;
    for j=1:1:n
        bin_x_qnd(i,j)=t1(j);
    end;

    t2=bin2gray(t1);
    for j=1:1:n
        gray_x_qnd(i,j)=t2(j);
    end;
```

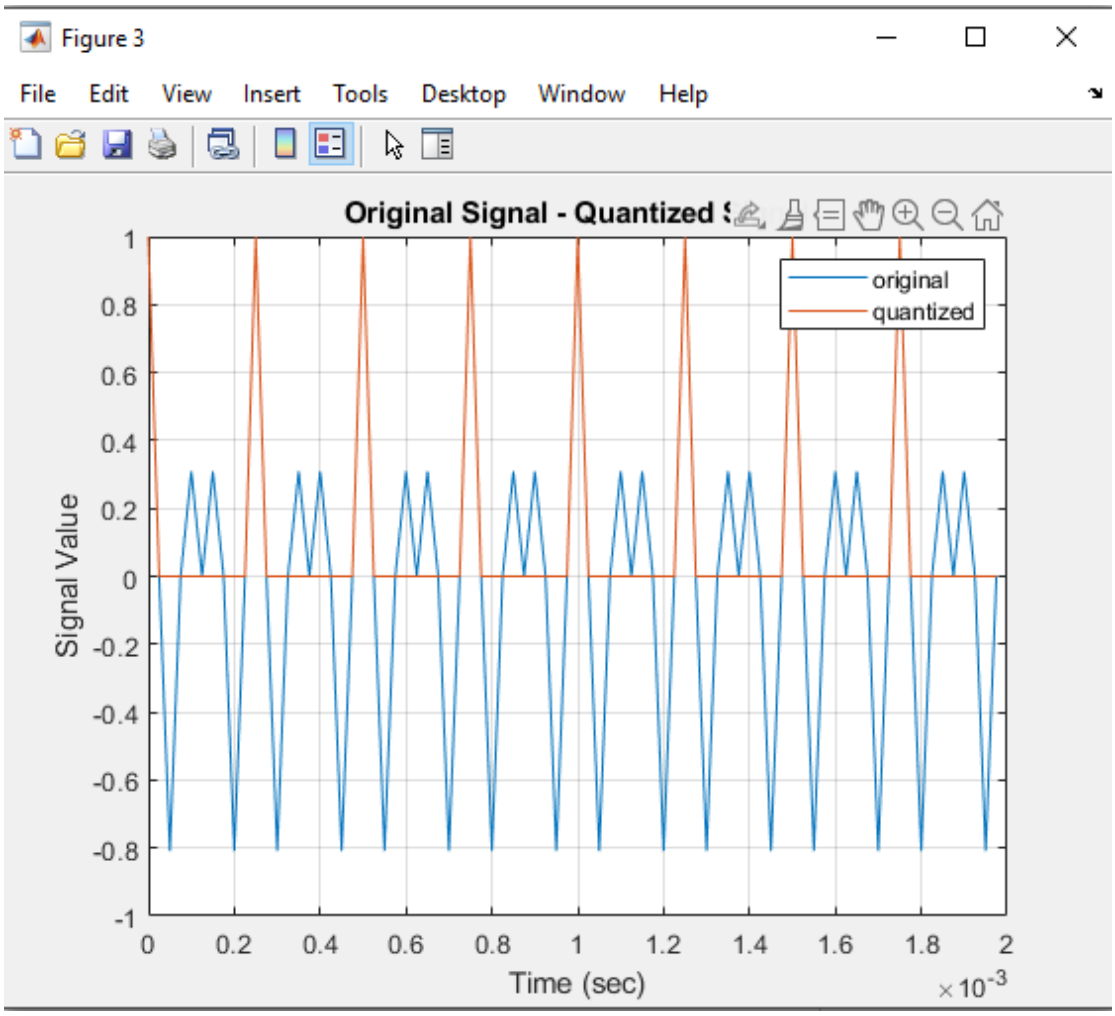
```
end;

figure(4)
stem(t_samp1(1:1:40),x_qnd(1:1:40));
text(t_samp1(1:1:40),x_qnd(1:1:40), num2str(gray_x_qnd(1:40,:)),
'FontSize',5);
grid;
xlabel('Time (sec)');
ylabel('Signal Value');
title('Gray Encoded Signal');
```

Πλάτος κβάντισης δ=(Amax-Amin)/2n, με Amax = 16 και Amin = 0. Υλοποίηση του κβαντιστή έγινε με τη χρήση της συνάρτησης quantiz().Για την υλοποίηση της κωδικοποίησης Gray χρησιμοποιήθηκε η συνάρτηση bin2gray η οποία έχει επισυναπτεί στο .zip file της υποβολής. Η χαρακτηριστική εισόδου – εξόδου του κβαντιστή με 16 στάθμες φαίνεται παρακάτω:

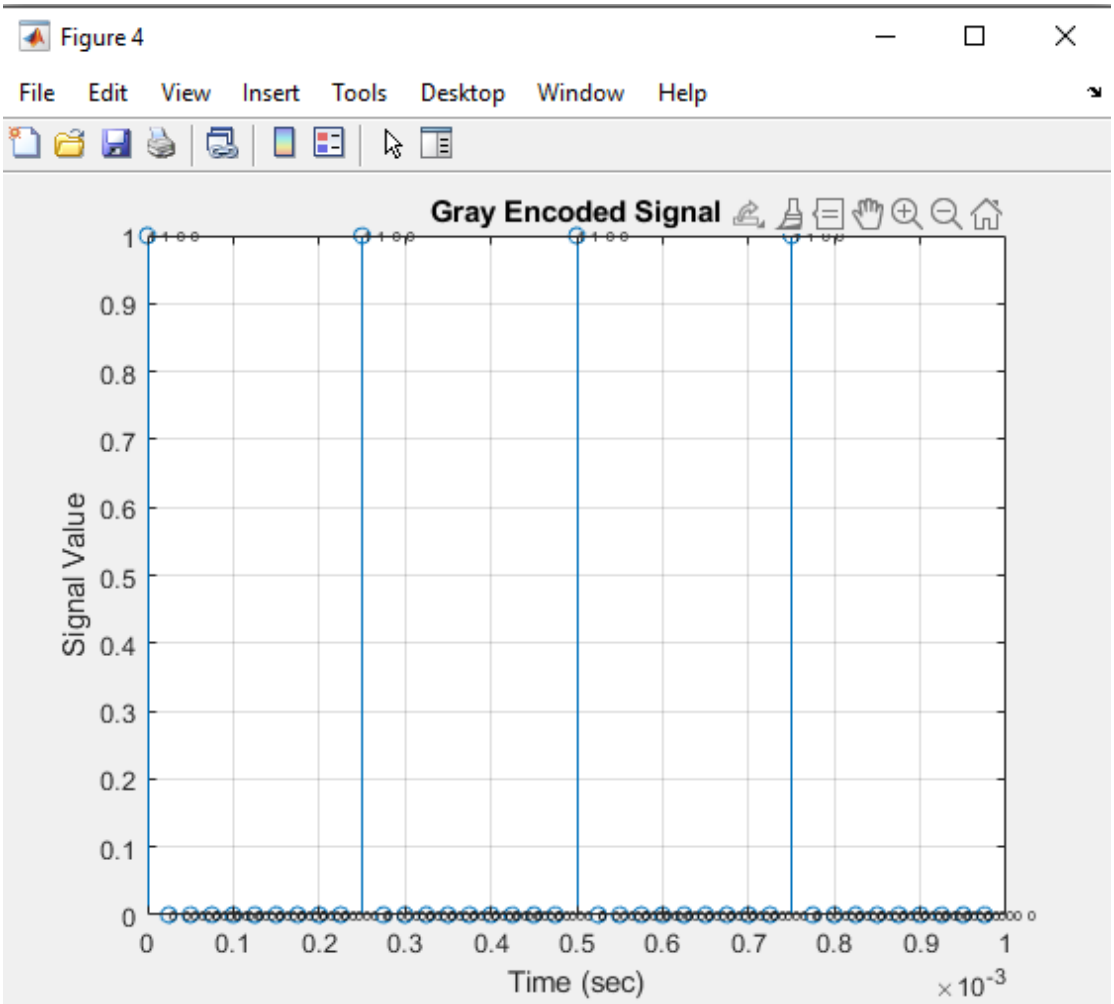


Στα σχήματα που ακολουθούν βλέπουμε τα παραγόμενα κβαντισμένα σήματα και τις τιμές τους για το τετράγωνο του δοθέντος ημιτονικού σήματος.





Με βάση το πλήθος κατωφλιών κβάντισης αλλοιώνεται το πλάτος του σήματος. Το παραγόμενο σφάλμα κβαντισης μειώνεται όσο αυξάνονται τα κατώφλια που ο κβαντιστής έχει για την αναπαράσταση του κβαντοποιημένου σήματος.
Παρουσιάζεται το κβαντισμένο σήμα με χρήση του κώδικα Gray.



Για την τυπική απόκλιση του σφάλματος κβάντισης και για το σηματοθορυβικό λόγο έχουμε:

{Question b}

```
% Question b

q_err=x_samp1-x_qnd;

std_10_samps=std(q_err(1:10));
power_10_samps=sum(x_samp1(1:10).^2)/10;
mse_10_samps=sum(q_err(1:10).^2)/10;
snr_10_samps=power_10_samps/(mse_10_samps/12);
snr_10_samps_db=10*log10(snr_10_samps);

std_20_samps=std(q_err(1:20));
power_20_samps=sum(x_samp1(1:20).^2)/20;
mse_20_samps=sum(q_err(1:20).^2)/20;
snr_20_samps=power_20_samps/(mse_20_samps/12);
snr_20_samps_db=10*log10(snr_20_samps);

disp(' Standard Deviation for 10 samples ');
disp(std_10_samps);
```

```
disp(' Standard Deviation for 20 samples ');
disp(std_20_samps);

disp(' SNR for 10 samples ');
disp(snr_10_samps);

disp(' SNR for 20 samples ');
disp(snr_20_samps);

disp(' SNR for 10 samples (db)');
disp(snr_10_samps_db);

disp(' SNR for 20 samples (db)');
disp(snr_20_samps_db);
```

Για τον προσδιορισμό των ζητούμενων δειγμάτων έγινε χρήση των συναρτήσεων std(), mean() οι οποίες υπολογίζουν τυπική απόκλιση και μέση τιμή από χρονοσειρές.

```
Standard Deviation for 10 samples
    0.3944

 Standard Deviation for 20 samples
    0.3839

 SNR for 10 samples
    20

 SNR for 20 samples
   20.0000

 SNR for 10 samples (db)
   13.0103

 SNR for 20 samples (db)
   13.0103
```

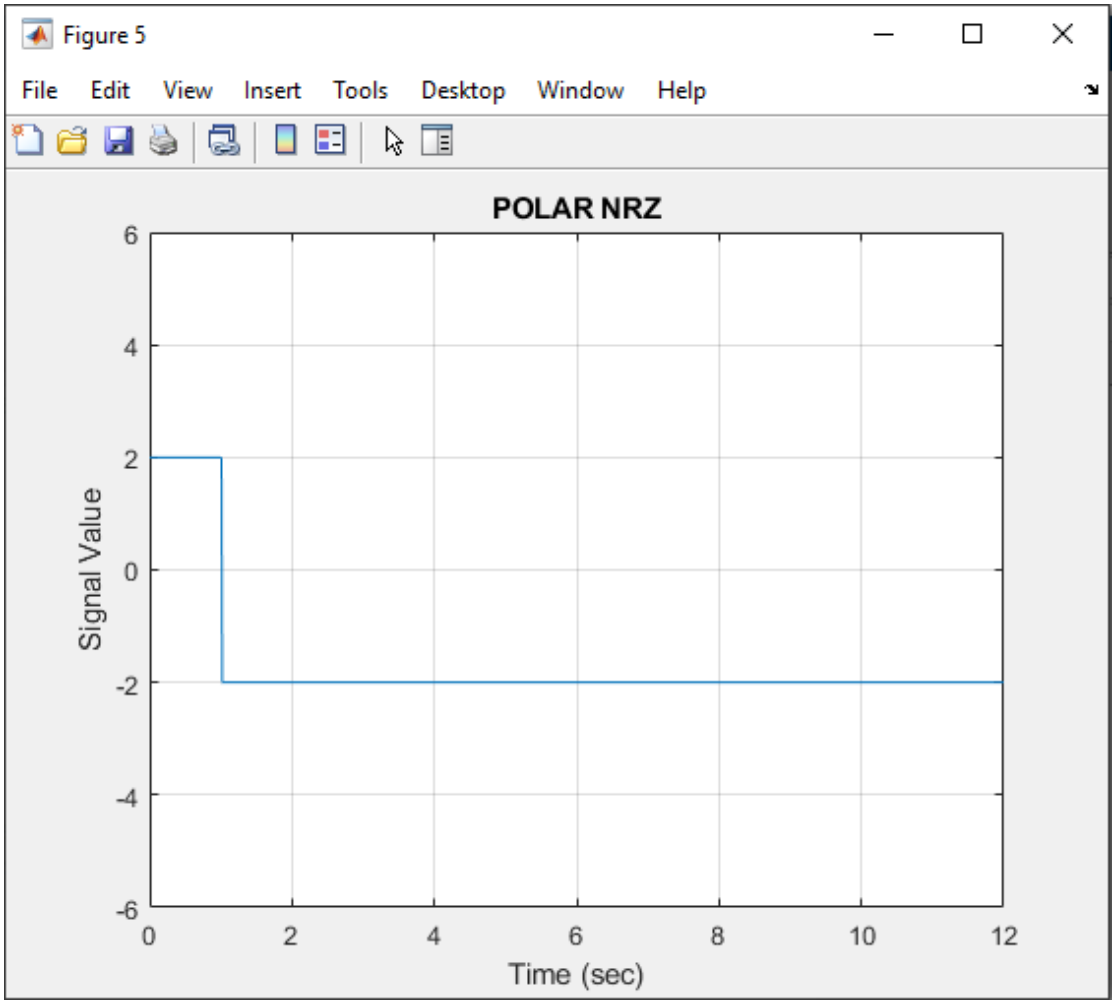
Η Τυπική απόκλιση παραμένει περίπου σταθερή και στις δύο περιπτώσεις των 10 και 20 δειγμάτων. Ο σηματοθορυβικός λόγος παραμένει σταθερός και στα δύο ενδεχόμενα δειγμάτων. Αυτό συμβαίνει επειδή τα 10 δείγματα ήδη επαρκούν για την εξαγωγή συμπερασμάτων για την δειγματοληψία του σήματος. Αυτό, επιβεβαιώνεται από την διατήρηση σταθερής τιμής και στα 20 δείγματα.

{Question c}

Για την υλοποίηση κωδικοποίησης γραμμής POLAR NRZ έχουμε:

```
% Question c


k=0;
for i=1:1:3
   for j=1:1:n
      k=k+1;
      bits(k)=bin_x_qnd(i,j);
   end;
end;

bitrate=n*1000;
Vp=fm/1000;

% Mapping
for i=1:length(bits)
  if (bits(i)==1)
    NRZ_out(i) = Vp;
  else
    NRZ_out(i) = -Vp;
  end;
end;

% Pulse Shaping

i=1;
t=0:0.01:length(bits);
for j=1:length(t)
  if (t(j)<=i)
    y(j)=NRZ_out(i);
  else
    y(j)=NRZ_out(i);
    i=i+1;
  end;
end;

% Plotting
figure(5)
plot(t,y)
axis([0  length(bits) -Vp-4 Vp+4])
grid;
xlabel('Time (sec)');
ylabel('Signal Value');
title('POLAR NRZ');
```

Η γραφική παράσταση του POLAR NRZ φαίνεται στο παρακάτω σχήμα:



Ερώτημα 3

Αρχικά για το ερώτημα παράχθηκε τυχαία ακολουθία μήκους n = 46 bits στην οποία τα ενδεχόμενα εμφάνισης 0 και 1 είναι ισοπίθανα. Η ακολουθία δημιουργήθηκε με χρήση της κανονικής κατανομής από την συνάρτηση randn().

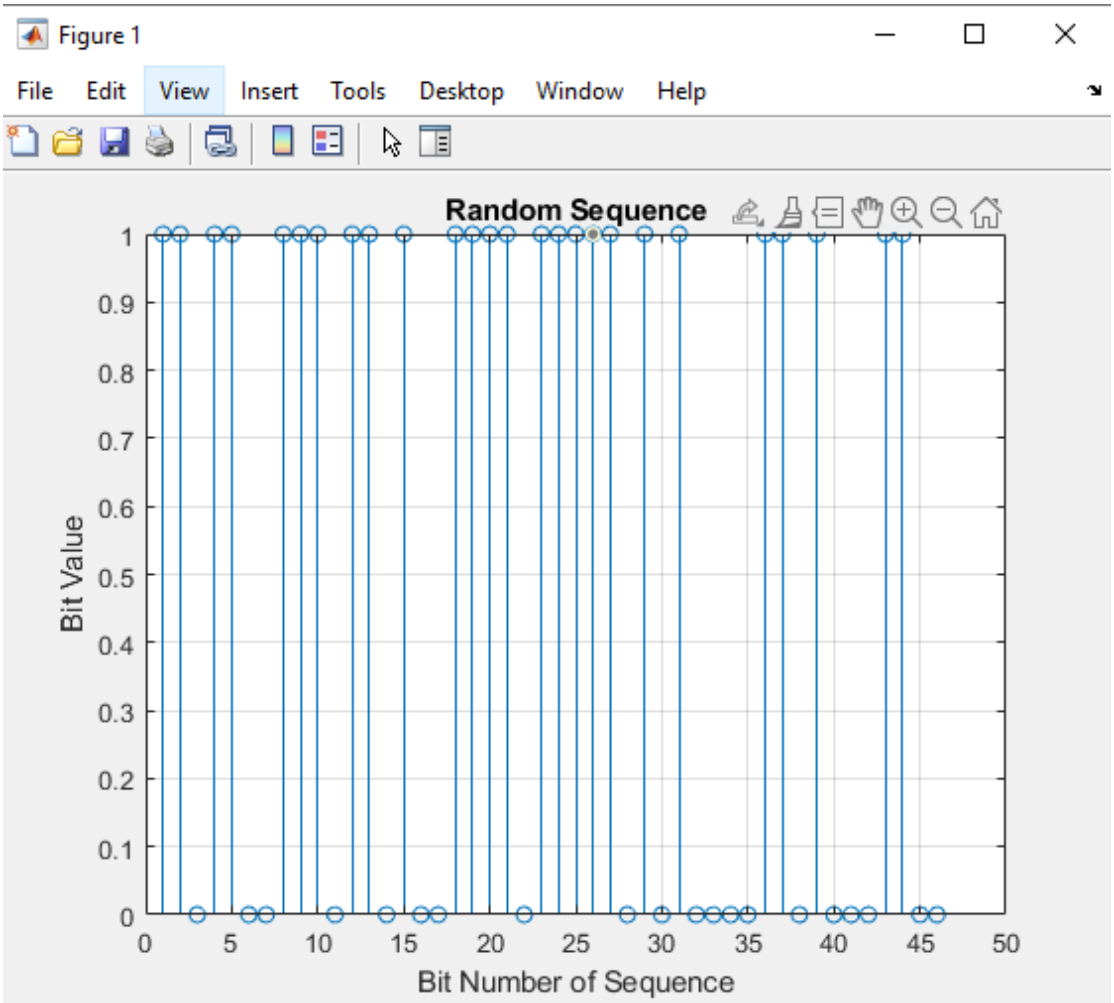
{Question a}

Για τη διαμόρφωση κατά B-PAM του σήματος, χρησιμοποιήθηκε πλάτος A = 2. Χρησιμοποιώντας την binary PAM προκύπτουν σύμβολα τα οποία

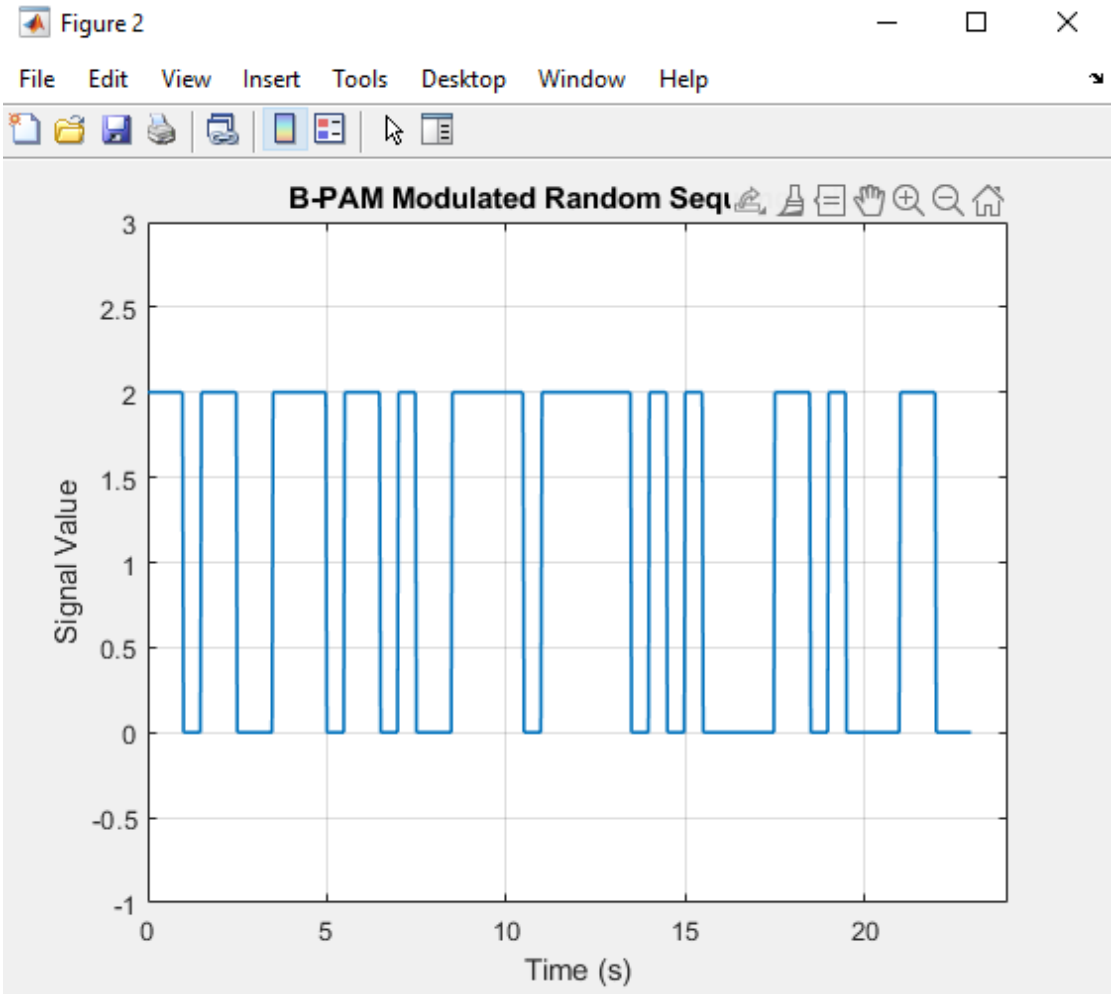
έχουν τιμές 0 όταν το σήμα προς μετάδοση είναι το λογικό 0, ενώ παράγεται ορθογωνικός παλμός διάρκειας Tb = 0.5 sec σε τιμή 1 για όλη τη διάρκεια μετάδοσης, όταν το αποσταλμένο σήμα «βρίσκεται» σε λογικό 1.

```
% Question a

   nbits=46;
   Tbit=0.5;

   randomseq=(1+sign(randn(nbits,1)))/2;

   amplitude=2;
   sampperbit=16;

   v=0;
   for i=1:1:nbits
      for j=1:1:sampperbit
         v=v+1;
         time_pt(v)=(Tbit/sampperbit)*(v-1);
         if (randomseq(i)==0)
            x_sign(v)=0;
         else
            x_sign(v)=amplitude;
         end;
      end;
   end;

 % Graphs

   figure(1)
   stem(randomseq,'o');
   grid;
   xlabel('Bit Number of Sequence');
   ylabel('Bit Value');
   title('Random Sequence');

   figure(2)
   plot(time_pt,x_sign,'LineWidth',1.2);
   grid;
   axis([0 nbits*Tbit+1 -1 amplitude+1]);
   xlabel('Time (s)');
   ylabel('Signal Value');
   title('B-PAM Modulated Random Sequence bits');
```

Γίνεται διαμόρφωση B-PAM και δίνει τα σχήματα των σημάτων για το χρονικό διάστημα μετάδοσης, τόσο για την αρχική ακολουθία των n = 46 bits όσο και για το χρονικά οριζόμενο σήμα μετά τη διαδικασία της διαμόρφωσης.



Τυχαία ακολουθία παραγωγής bits (n = 46)



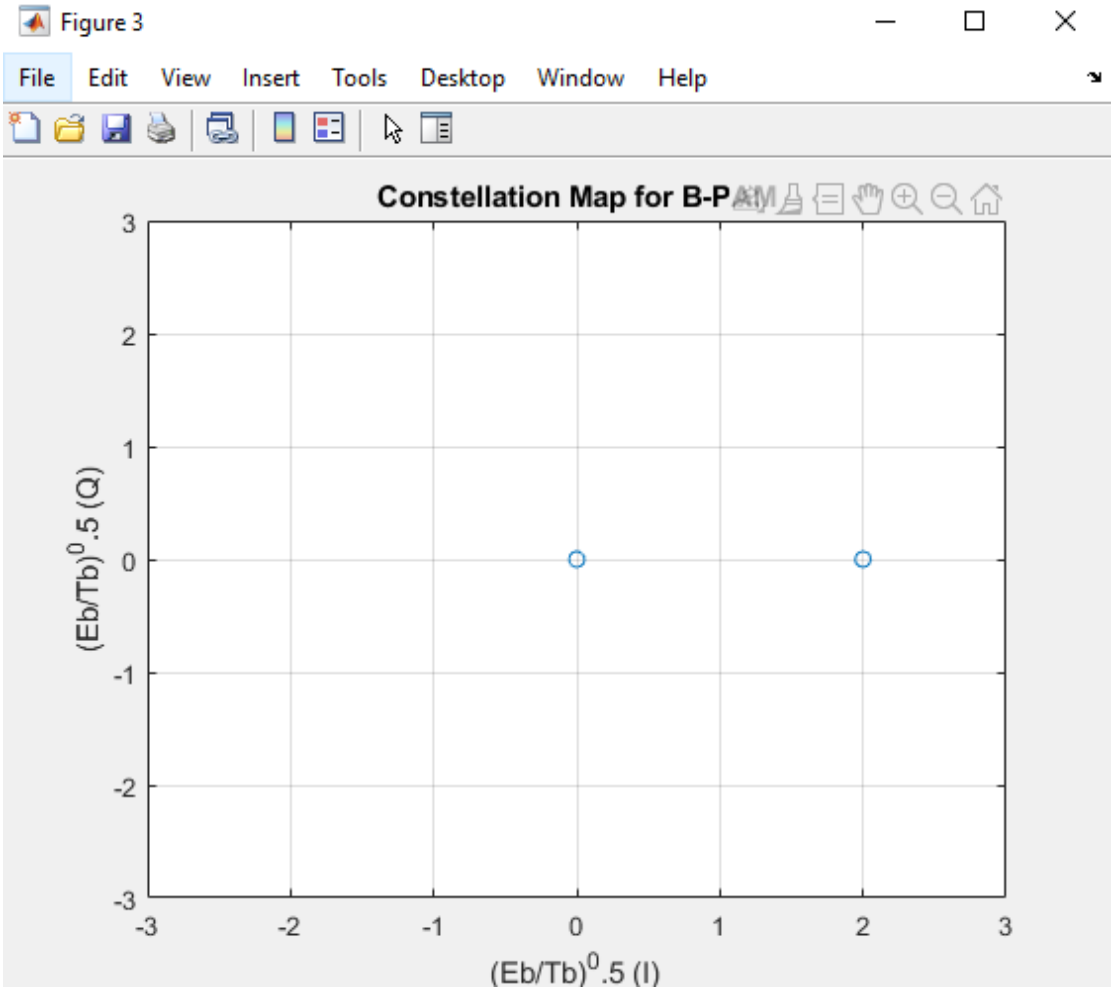
B-PAM διαμορφωμένο σήμα για την «τυχαία» ακολουθία

{Question b}

Για την παραγωγή του διαγράμματος αστερισμού της B-PAM, το ένα σύμβολο βρίσκεται στη θέση (0,0) και δίνει σήμα μηδενικής ενέργειας όταν θα πρέπει να μεταδοθεί το λογικό 0. Το άλλο σύμβολο βρίσκεται στη θέση (A,0), δίνοντας σήμα με ένεργεια ανά bit A.

```
% Question b

    PAMst=2;
    symamp(1)=0;
    symamp(2)=amplitude;

    for i=1:1:PAMst
       xstar_nonoise(i)=(i-1)*(Tbit*symamp(i)^2/Tbit)^0.5;
       ystar_nonoise(i)=0;
    end;

    figure(3)
```

```
    plot(xstar_nonoise,ystar_nonoise,'o');
    grid;
    axis([-max(xstar_nonoise)-1 max(xstar_nonoise)+1 -
max(xstar_nonoise)-1 max(xstar_nonoise)+1]);
    xlabel('(Eb/Tb)^0.5 (I)');
    ylabel('(Eb/Tb)^0.5 (Q)');
    title('Constellation Map for B-PAM');
```



Διάγραμμα αστερισμού για τη διαμόρφωση B-PAM

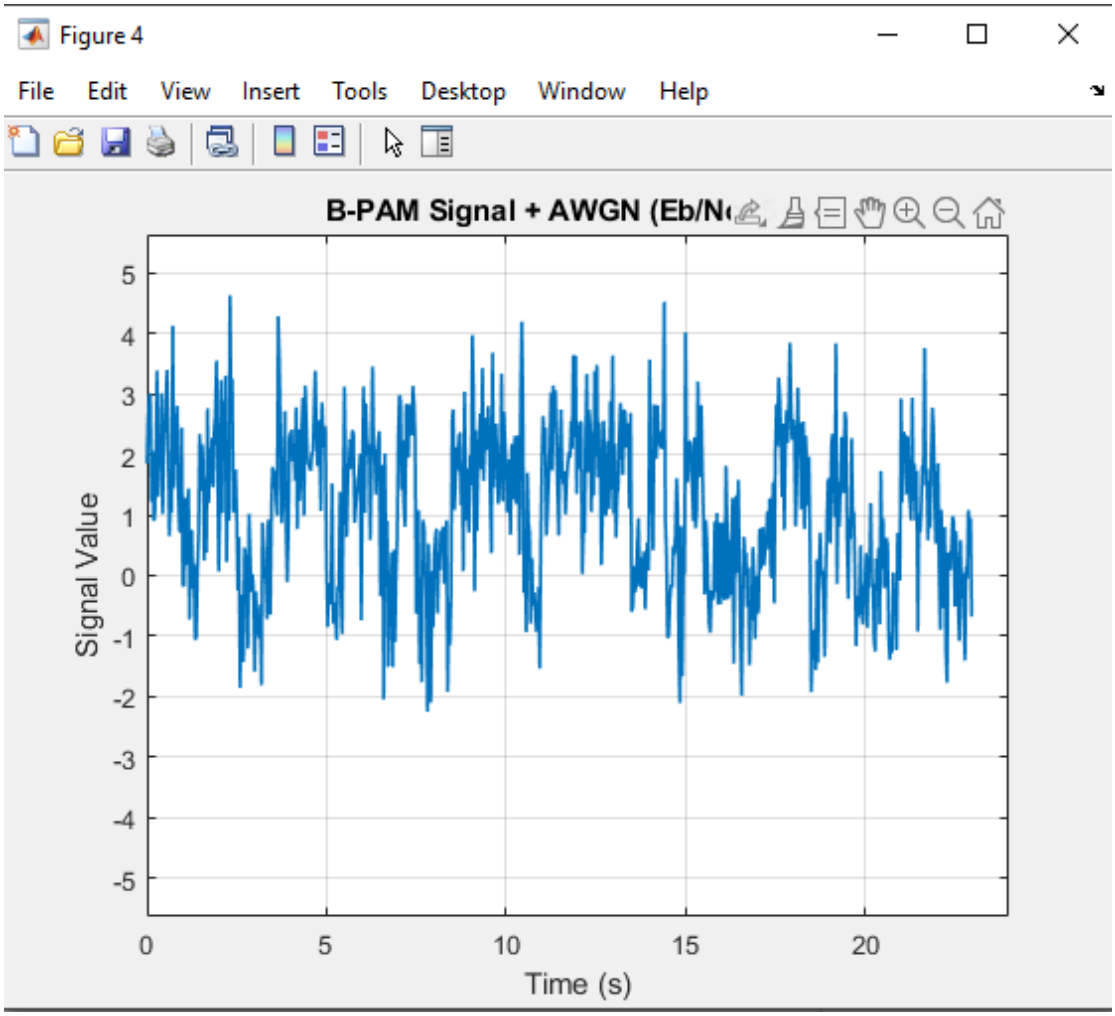
{Question c}

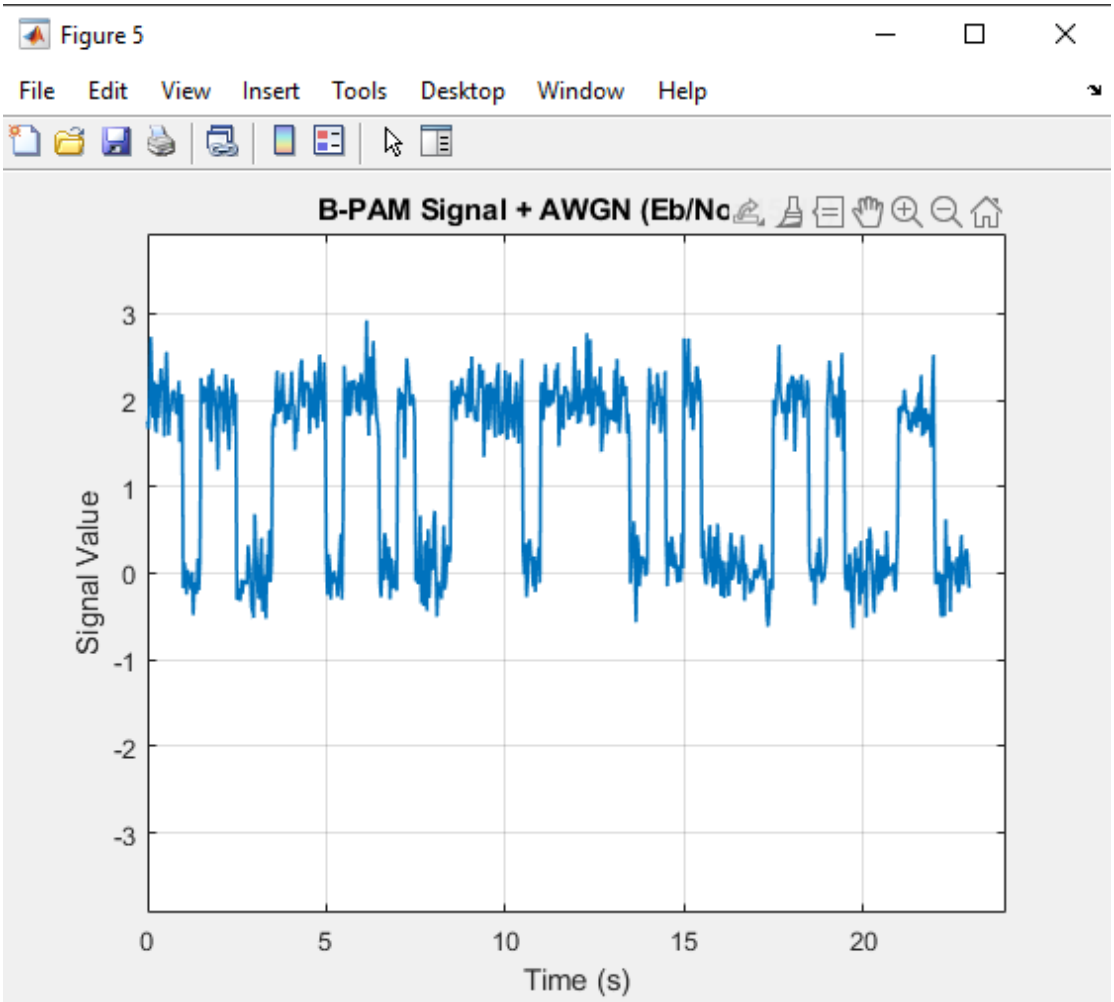
Για την παραγωγή AWGN, αρχικά υπολογίστηκε η ενέργεια του σήματος προς μετάδοση. Στη συνέχεια με βάση το SNR βρέθηκε η ενέργεια που θα χρειαστεί για το ίδιο μήκος μετάδοσης να αντιστοιχηθεί στο θόρυβο. Το σήμα θορύβου δγμιουργήθηκε με χρήση της συνάρτησης normrandn(), η οποία παράγει «τυχαίες» ακολουθίες αριθμών με μηδενική μέση τιμή και καθορισμένη διασπορά. Η διασπορά του θορύβου για τη γεννήτρια υπολογίστηκε από την ενέργεια του θορύβου

προς το πλήθος των bits μετάδοσης και διαθέτει μόνο πραγματικό μέρος.

```
% Question c

   SNR=5;

   x_sign_energy=sum(abs(x_sign).^2);

   noise_sign_energy=x_sign_energy/10^(SNR/10);
   noise_length=length(x_sign);
   sigma_n=(noise_sign_energy/noise_length)^0.5;
   noise_sigr=normrnd(0,sigma_n,[1,length(x_sign)]);
   noise_sign=noise_sigr;
   noise_sign_energy_calc=sum(abs(noise_sign).^2);

   SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
   SNRresult=SNR-SNRcalculation;
   noise_sign=10^(-SNRresult/10)*noise_sign;

   x_sign_n=x_sign+noise_sign;

   figure(4)
   plot(time_pt,x_sign_n,'LineWidth',1.2);
   grid;
   axis([0 nbits*Tbit+1 -1-max(x_sign_n) 1+max(x_sign_n)]);
   xlabel('Time (s)');
   ylabel('Signal Value');
   title('B-PAM Signal + AWGN (Eb/No=5 db)');

   SNR=15;

   x_sign_energy=sum(abs(x_sign).^2);

   noise_sign_energy=x_sign_energy/10^(SNR/10);
   noise_length=length(x_sign);
   sigma_n=(noise_sign_energy/noise_length)^0.5;
   noise_sigr=normrnd(0,sigma_n,[1,length(x_sign)]);
   noise_sign=noise_sigr;
   noise_sign_energy_calc=sum(abs(noise_sign).^2);

   SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
   SNRresult=SNR-SNRcalculation;
   noise_sign=10^(-SNRresult/10)*noise_sign;

   x_sign_n=x_sign+noise_sign;

   figure(5)
   plot(time_pt,x_sign_n,'LineWidth',1.2);
   grid;
   axis([0 nbits*Tbit+1 -1-max(x_sign_n) 1+max(x_sign_n)]);
   xlabel('Time (s)');
   ylabel('Signal Value');
   title('B-PAM Signal + AWGN (Eb/No=15 db)');
```

Ο κώδικας μετά την παραγωγή του θορύβου, επαναϋπολογίζει το SNR ώστε να εξαφανίσει τυχόν διαφορές, επανακαθορίζοντας το πλάτος του θορύβου. Γραφήματα μετά την επικάθιση του σήματος του προσθετικού θορύβου.





Διαμορφωμένο κατά B-PAM σήμα και AWGN με SNR=5 και 15 db αντίστοιχα.

Παρατηρούμε ότι για χαμηλές τιμές του SNR το σήμα παραμορφώνεται έντονα από την επίδραση του θορύβου. Όσο το SNR αυξάνεται, το σήμα βαίνει να αποκατασταθεί στην ιδανική μορφή του.

{Question d}

Είναι απαραίτητο να βρεθεί η ενέργεια των μεταδιδόμενων ψηφίων ανά περίοδο μετάδοσης και τα σημεία αυτά να εντοπιστούν στο διάγραμμα παράστασης αστερισμού του ιδανικού B-PAM. Στην περίπτωσή μας παράγεται μιγαδικός θόρυβος χρησιμοποιώντας την γεννήτρια «τυχαίων» αριθμών κανονικής κατανομής. Τα θορυβικά σήματα για το πραγματικό και το φανταστικό μέρος είναι ανεξαρτήτως παραγόμενα.

```
 % Question d

   SNR=5;

   x_sign_energy=sum(abs(x_sign).^2);

   noise_sign_energy=x_sign_energy/10^(SNR/10);
   noise_length=length(x_sign);
   sigma_n=(noise_sign_energy/noise_length)^0.5;
   noise_sigr=normrnd(0,sigma_n,[1,length(x_sign)]);
   noise_sigi=normrnd(0,sigma_n,[1,length(x_sign)]);
   noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
   noise_sign_energy_calc=sum(abs(noise_sign).^2);

   SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
   SNRresult=SNR-SNRcalculation;
   noise_sign=10^(-SNRresult/10)*noise_sign;

   x_sign_n=x_sign+noise_sign;

   for i=1:1:nbits
      bit_energyr(i)=sum(Tbit/sampperbit*real(x_sign_n(1+(i-
1)*sampperbit:sampperbit+(i-1)*sampperbit)).^2);
      bit_energyi(i)=sum(Tbit/sampperbit*imag(x_sign_n(1+(i-
1)*sampperbit:sampperbit+(i-1)*sampperbit)).^2);
   end;

   for i=1:1:nbits
      xstar(i)=(bit_energyr(i)/Tbit)^0.5;
      ystar(i)=(bit_energyi(i)/Tbit)^0.5;
   end;

   figure(6)
   plot(xstar,ystar,'x');
   hold on
   plot(xstar_nonoise,ystar_nonoise,'o');
   hold off
   grid;
   axis([-max(xstar)-1 max(xstar)+1 -max(xstar)-1 max(xstar)+1]);
   xlabel('(Eb/Tb)^0^.^5 (I)');
   ylabel('(Eb/Tb)^0^.^5 (Q)');
   title('Constellation Map for B-PAM (Eb/No=5 db)');
   legend('x=Stars for signal with noise','o=Stars for noiseless
signal');

   SNR=15;

   x_sign_energy=sum(abs(x_sign).^2);

   noise_sign_energy=x_sign_energy/10^(SNR/10);
   noise_length=length(x_sign);
   sigma_n=(noise_sign_energy/noise_length)^0.5;
   noise_sigr=normrnd(0,sigma_n,[1,length(x_sign)]);
   noise_sigi=normrnd(0,sigma_n,[1,length(x_sign)]);
   noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
   noise_sign_energy_calc=sum(abs(noise_sign).^2);

   SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
```

```
   SNRresult=SNR-SNRcalculation;
   noise_sign=10^(-SNRresult/10)*noise_sign;

   x_sign_n=x_sign+noise_sign;

   for i=1:1:nbits
      bit_energyr(i)=sum(Tbit/sampperbit*real(x_sign_n(1+(i-
1)*sampperbit:sampperbit+(i-1)*sampperbit)).^2);
      bit_energyi(i)=sum(Tbit/sampperbit*imag(x_sign_n(1+(i-
1)*sampperbit:sampperbit+(i-1)*sampperbit)).^2);
   end;

   for i=1:1:nbits
      xstar(i)=(bit_energyr(i)/Tbit)^0.5;
      ystar(i)=(bit_energyi(i)/Tbit)^0.5;
   end;

   figure(7)
   plot(xstar,ystar,'x');
   hold on
   plot(xstar_nonoise,ystar_nonoise,'o');
   hold off
   grid;
   axis([-max(xstar)-1 max(xstar)+1 -max(xstar)-1 max(xstar)+1]);
   xlabel('(Eb/Tb)^0^.^5 (I)');
   ylabel('(Eb/Tb)^0^.^5 (Q)');
   title('Constellation Map for B-PAM (Eb/No=15 db)');
   legend('x=Stars for signal with noise','o=Stars for noiseless
signal');
```

Παράγεται σύμφωνα με την ενέργεια του αρχικού σήματος, η ενέργεια που χρειάζεται για το θόρυβο. Αυτό έχει ως αποτέλεσμα τα προκύπτοντα σύμβολα να γίνονται σημεία με συντεταγμένες (x,y) οι οποίες λόγω του θορύβου δεν έχουν τεταγμένη y = 0 αλλά τυχαίες θέσεις.

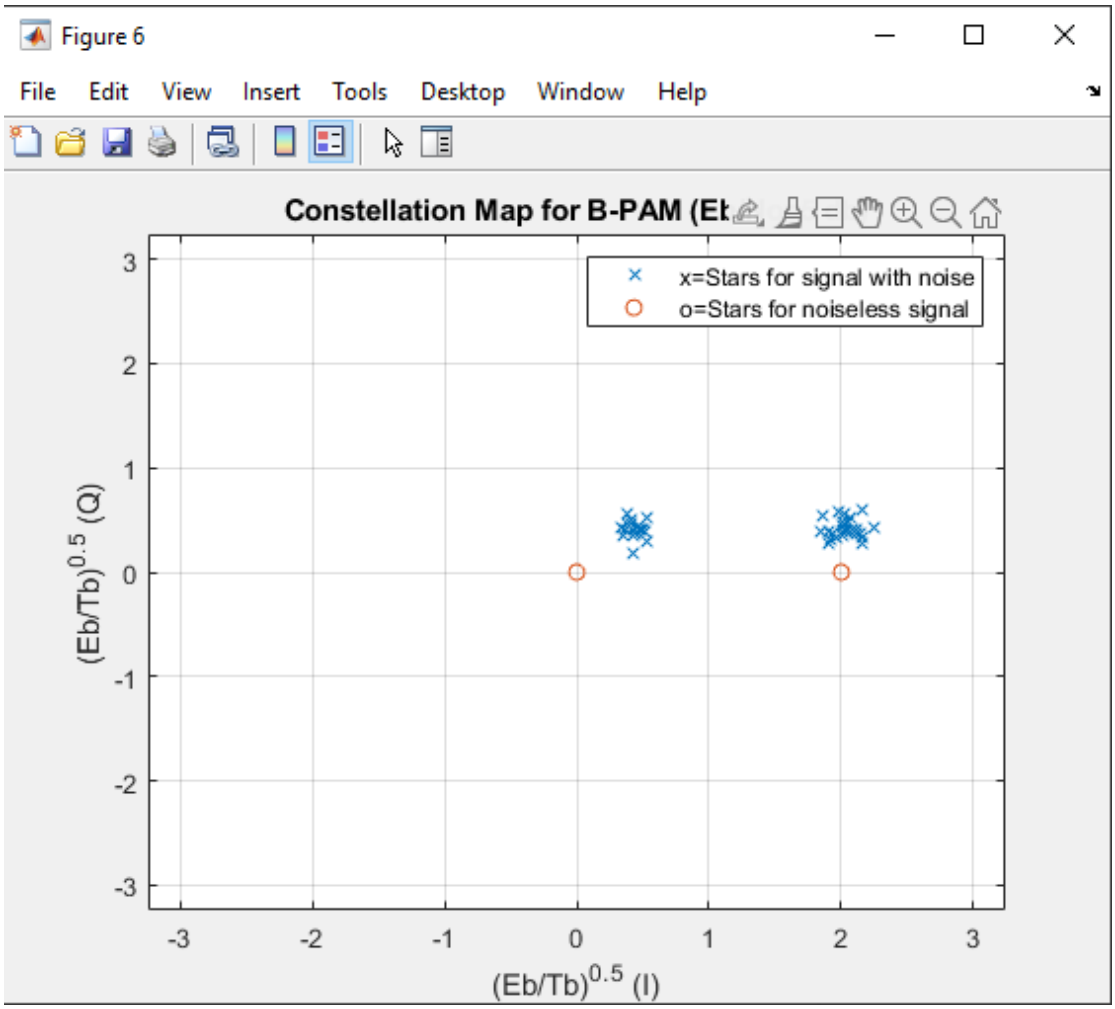
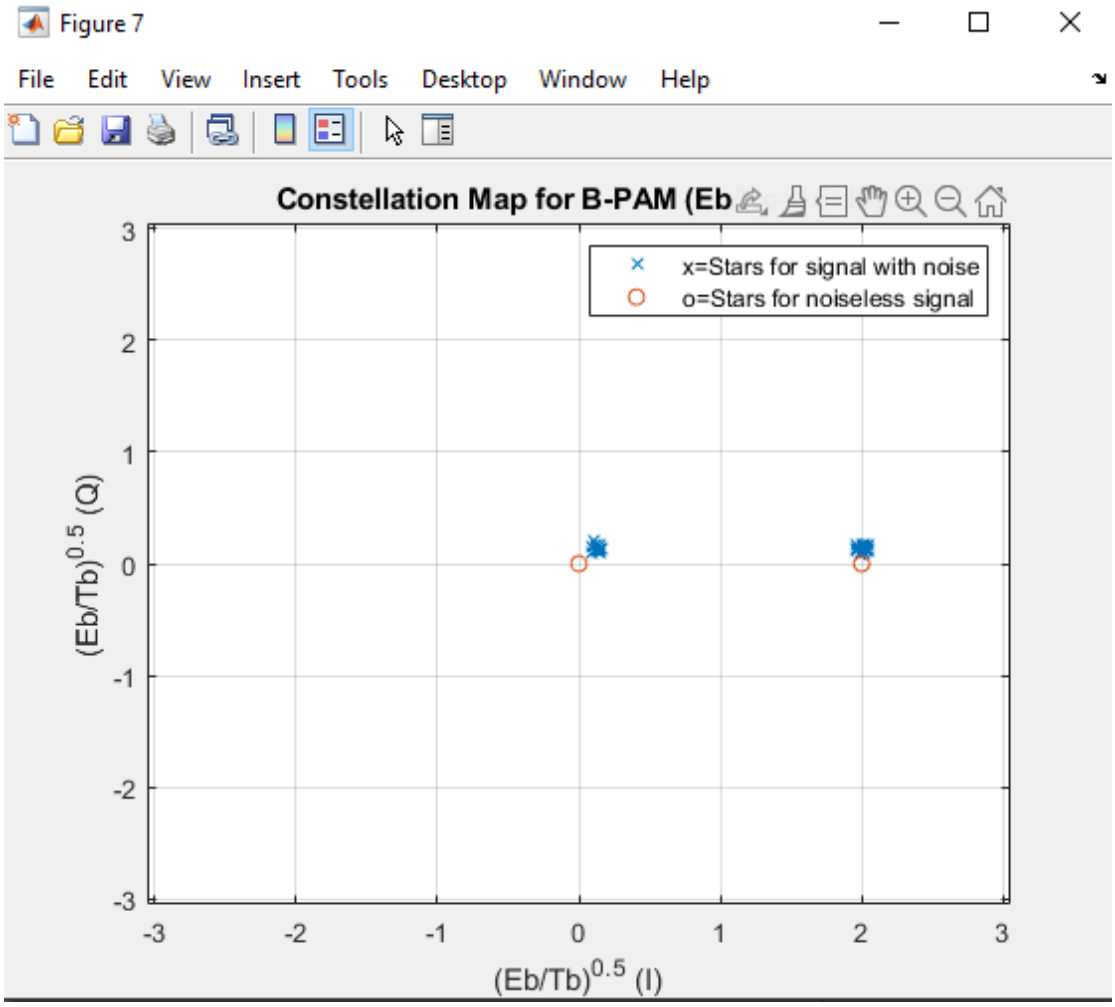


Figure 6
File Edit View Insert Tools Desktop Window Help
Constellation Map for B-PAM (Eb
x=Stars for signal with noise
o=Stars for noiseless signal
(Eb/Tb)^0.5 (Q)
(Eb/Tb)^0.5 (I)
3
2
1
0
-1
-2
-3



Διαγράμματα αστερισμού για τη μεταδιδόμενη κυματομορφή με SNR=5 και 15 db αντίστοιχα.

Από τα διαγράμματα αστερισμού του μεταδιδόμενου σήματος, παρατηρούμε ότι όσο αυξάνεται ο σηματοθορυβικός λόγος μετάδοσης, τόσο πιο κοντά προς τις ιδανικές θέσεις του αστερισμού πηγαίνουν τα σύμβολα μετάδοσης. Ακόμα, ο μιγαδικός θόρυβος εκτρέπει τα σύμβολα του αστερισμού από την τετμημένη y = 0.

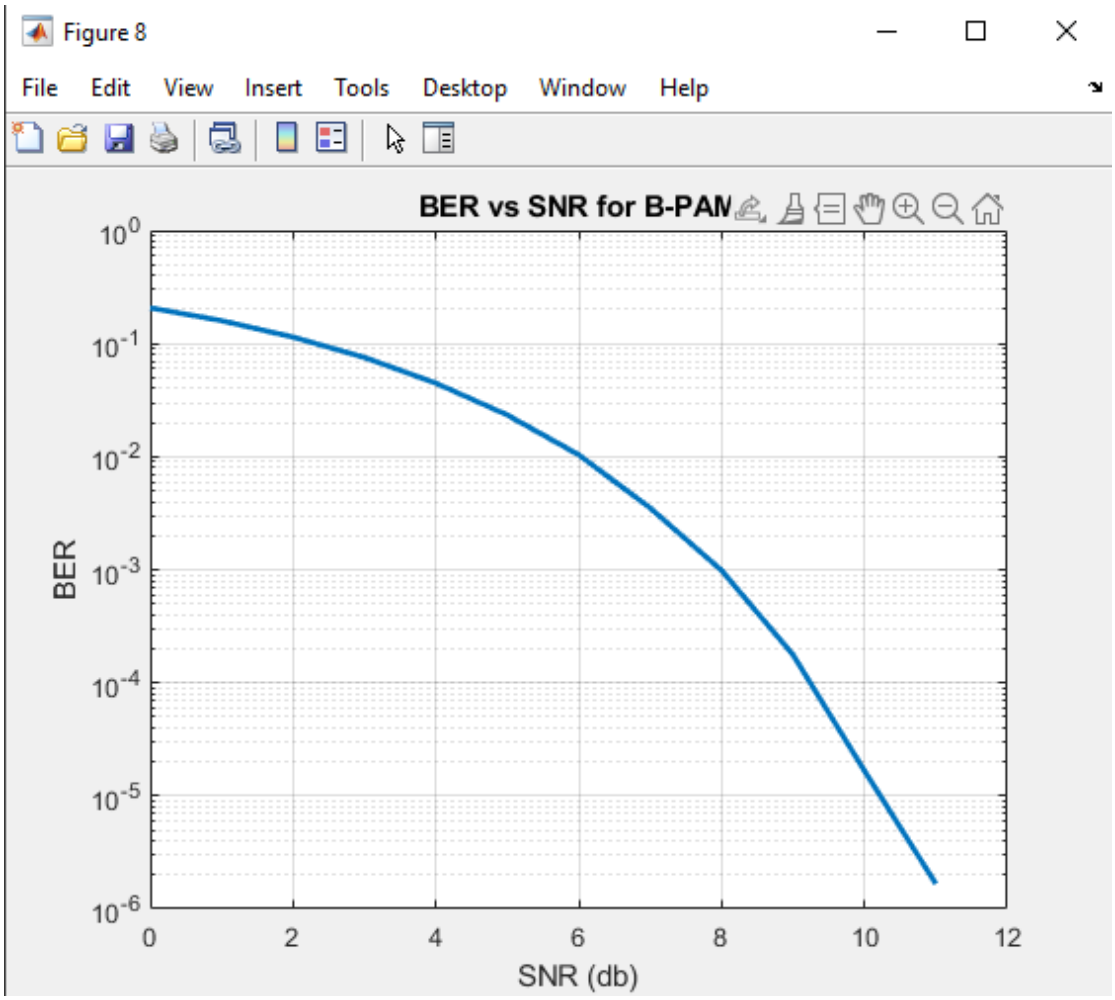
{Question e}

Για το ερώτημα αυτό θα χρειαστεί να παράξουμε θόρυβο με SNR για τη ζώνη 0 – 15 db με βήμα 1 db. Στόχος είναι η μέτρηση του πλήθους εσφαλμένων bits για τις δεδομένες μεταδόσεις που αντιστοιχούν στους συγκεκριμένους SNR και η κατασκευή της καμπύλης BER εν συναρτήσει SNR σε λογαριθμική κλίμακα.

```
% Question e

    format long;

    SNR_start=0;
    SNR_stop=15;
    SNR_step=1;
    SNR_iter=floor((SNR_stop-SNR_start)/SNR_step)+1;

    nbits=600000;
    Tbit=0.5;

    randomseq=(1+sign(randn(nbits,1)))/2;

    amplitude=9;
    sampperbit=1;

    v=0;
    for i=1:1:nbits
       for j=1:1:sampperbit
          v=v+1;
          time_pt(v)=(Tbit/sampperbit)*(v-1);
          if (randomseq(i)==0)
             x_sign(v)=0;
          else
             x_sign(v)=amplitude;
          end;
       end;
    end;

    x_sign_energy=sum(abs(x_sign).^2);

    for s=1:1:SNR_iter

       SNR_st(s)=SNR_start+(s-1)*SNR_step;
       SNR=SNR_st(s);

       noise_sign_energy=x_sign_energy/10^(SNR/10);
       noise_length=length(x_sign);
       sigma_n=(noise_sign_energy/noise_length)^0.5;
       noise_sigr=normrnd(0,sigma_n,[1,length(x_sign)]);
       noise_sigi=normrnd(0,sigma_n,[1,length(x_sign)]);
       noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
       noise_sign_energy_calc=sum(abs(noise_sign).^2);

       SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
       SNRresult=SNR-SNRcalculation;
       noise_sign=10^(-SNRresult/10)*noise_sign;

       x_sign_n=x_sign+noise_sign;

       for i=1:1:nbits
          if (abs(x_sign_n(i))<(amplitude/2))
             rand_seq_rx(i)=0;
          else
             rand_seq_rx(i)=1;
          end;
```

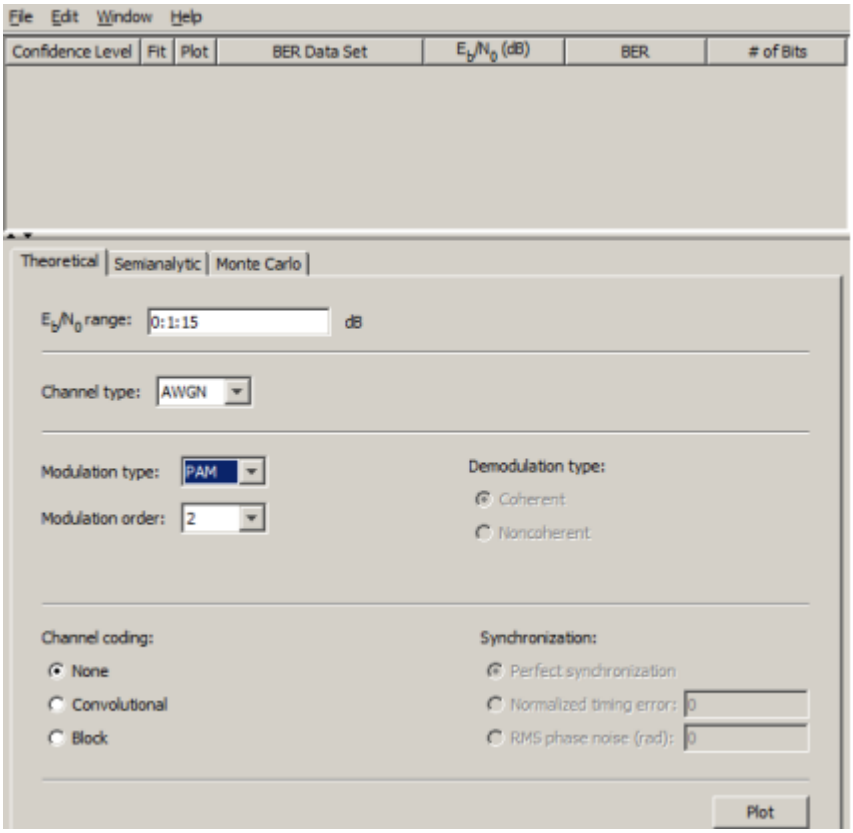
```
        end;

        BER(s)=0;
        for i=1:1:nbits
            if (randomseq(i)~=rand_seq_rx(i))
                BER(s)=BER(s)+1;
            end;
        end;
        BER_st(s)=BER(s)/nbits;

    end;

    figure(8)
    semilogy(SNR_st,BER_st,'LineWidth',2);
    grid;
    xlabel('SNR (db)');
    ylabel('BER');
    title('BER vs SNR for B-PAM');
```

Υλοποιείται ένας βρόχος επανάληψης με βάση το βήμα για το SNR και τις αρχικές και τελικές επιθυμητές τιμές. Χρησιμοποιείται μήκος δειγμάτων για κάθε μετάδοση N = 600.000 για την αξιοπιστία των μετρήσεων. Για τη λήψη απόφασης, χρησιμοποιείται ο κανόνας απόφασης με βάση την απόσταση από τα δύο σύμβολα της PAM. Η ζώνη απόφασης είναι διαιρεμένη δία δύο, άρα A/2 και αποδίδει όλα τα εισερχόμενα σε τιμή μηδέν αν η απόσταση τους είναι μικρότερη του A/2, και σε λογικό 1 στην αντίθετη περίπτωση. Για τη μέτρηση του BER γίνεται σύγκριση bit προς bit της ακολουθίας μετάδοσης και λήψης βρίσκοντας έτσι το πλήθος των σφαλμάτων.

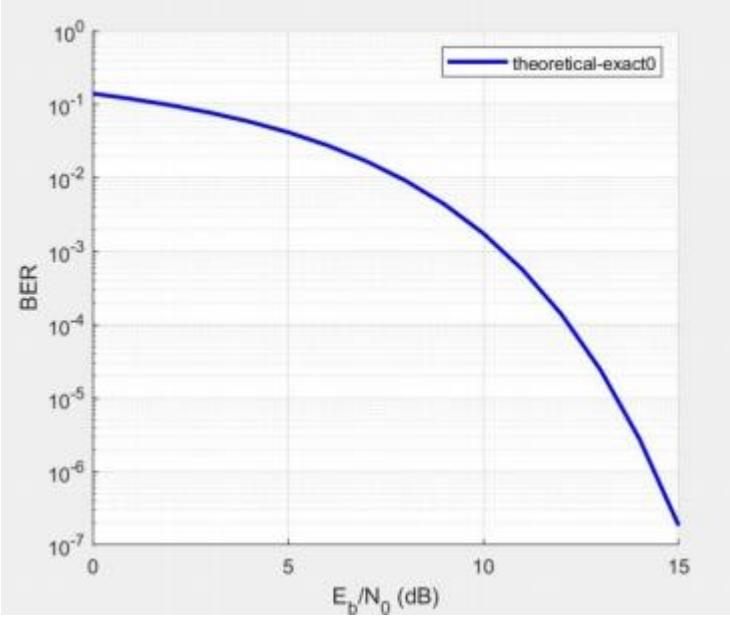


Παρατηρούμε ότι η αύξηση του SNR προκαλεί τη μείωση του BER όπως αναμενόταν και θεωρητικά. Επίσης παρατηρούμε ότι για SNR > 15 db τα σφάλματα ήταν μηδενικά και για αυτό το λόγο οι τιμές τους δεν αποτυπώνονται στο λογαριθμικό διάγραμμα.

Για τη θεωρητική σύγκριση χρησιμοποιήθηκε το bertool() και επιλέχθηκε PAM με 2 σύμβολα, δηλαδή ο καθορισμός παραμέτρων σύμφωνα με τα ζητούμενα του ερωτήματος.



Τα αποτελέσματα από τη θεωρητική ανάλυση με χρήση του bertool() παρουσιάζονται στο ακόλουθο σχήμα:



Θεωρητικό Διάγραμμα BER συναρτήσει του SNR για τη διοαμόρφωση B-PAM με χρήση του bertool()**.**

Παρατηρούμε ότι τα αποτελέσματα βρίσκονται μεταξύ θεωρητικών και πειραματικών μετρήσεων.

Ερώτημα 4

Για την υλοποίηση του ερωτήματος θεωρήσαμε την παραγόμενη QPSK με πλάτος φέροντος A=2V και διάρκεια bit = 1 / fc = 0.5 sec.

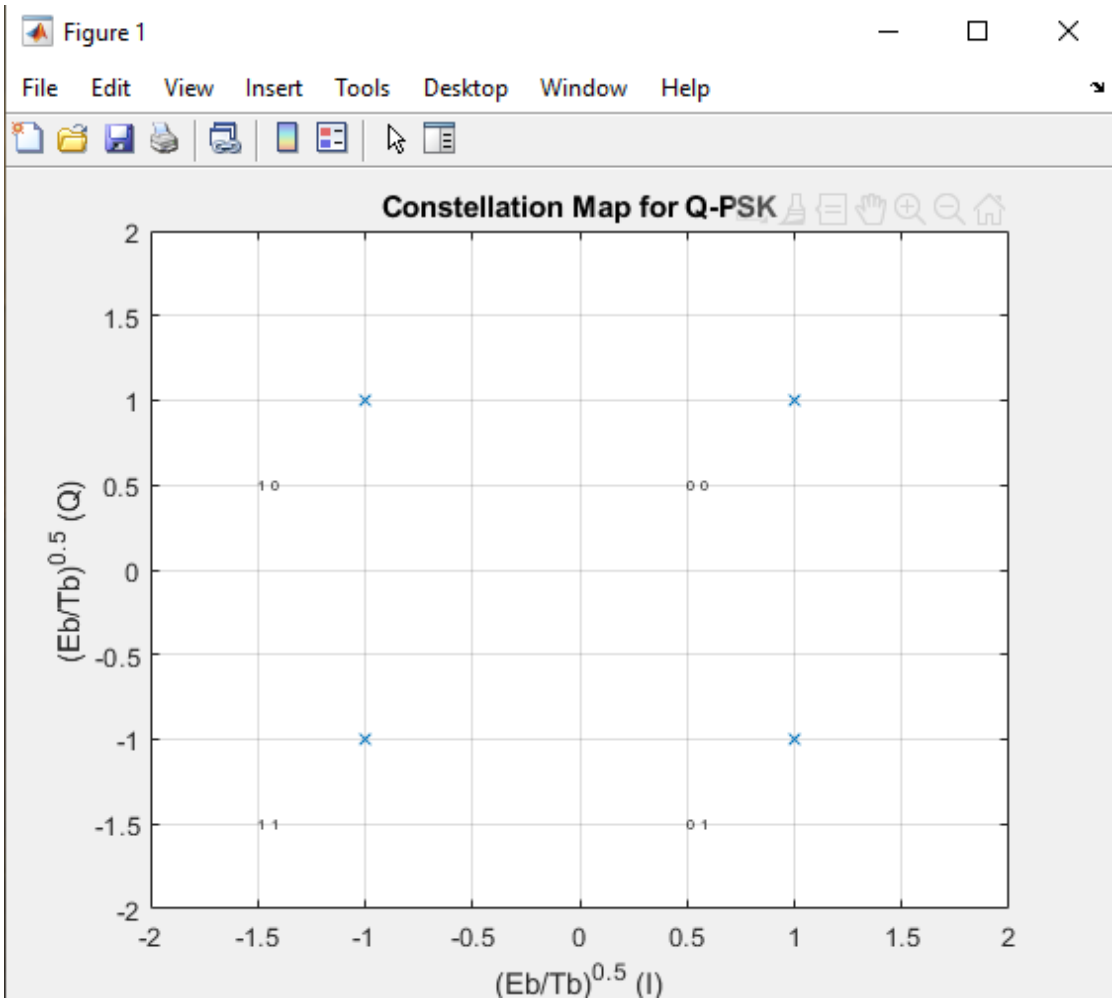
{Question a}

```
% Vikentios Vitalis el18803
% 8 + 3 = 11 = 1 + 1 = 2
% shannon_even.txt logw artiothtas
clc;
clear;

% Question a
   nbits = 46;
   randomseq = (1+sign(randn(1,nbits)))/2;
   amplitude = 2;
   samps_per_bit = 16;
   Tbit = 0.5;
   fc = 1/Tbit;
```

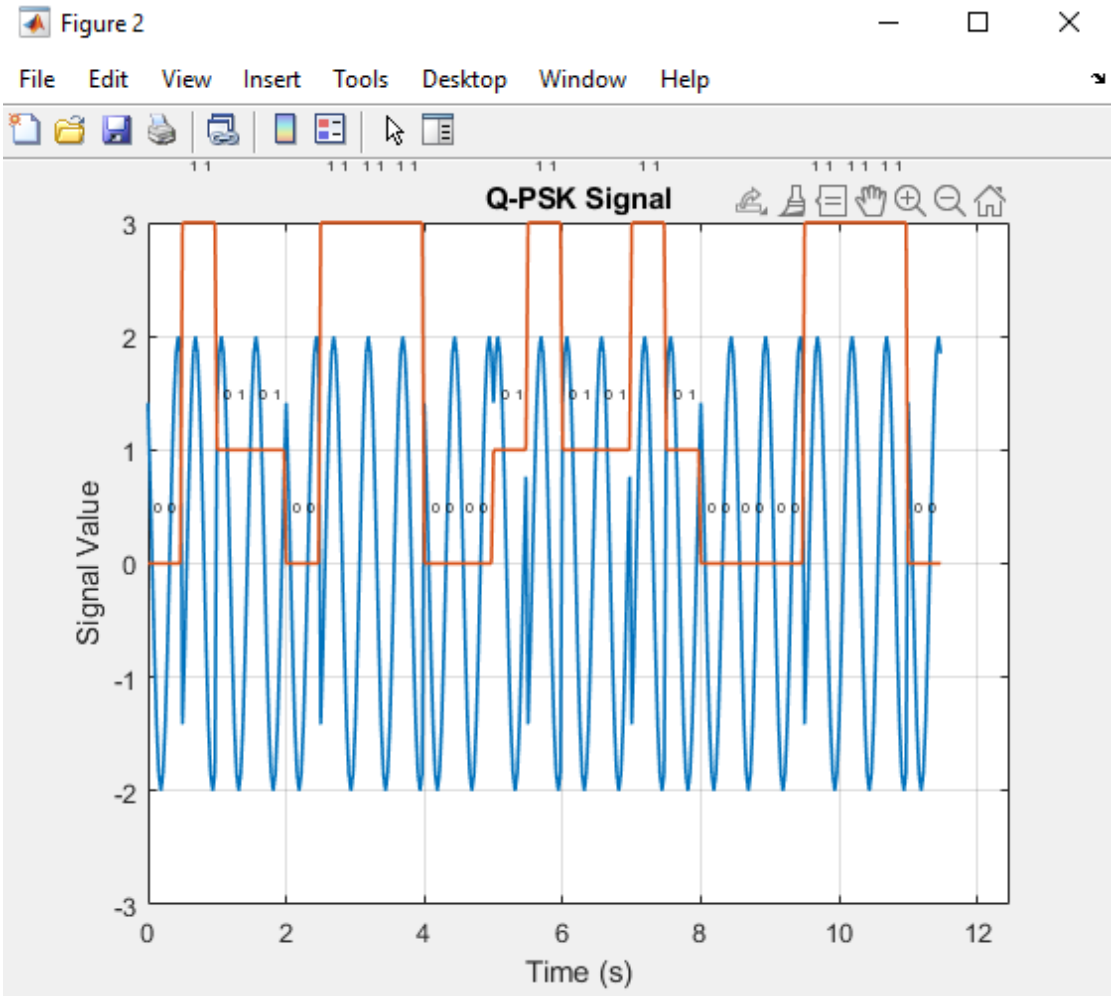
```
%   QPSK

   v=0;
   for i=1:1:nbits/2
      for j=1:1:samps_per_bit
         v=v+1;
         timepoint(v)=(Tbit/samps_per_bit)*(v-1);
         if (randomseq(2*(i-1)+1:2*(i-1)+2)==00)
            x_sign(v)=amplitude*cos(2*pi*fc*timepoint(v)+pi/4);
            d_sign(v)=0;
         elseif (randomseq(2*(i-1)+1:2*(i-1)+2)==01)

x_sign(v)=amplitude*cos(2*pi*fc*timepoint(v)+pi/4+3*pi/2);
            d_sign(v)=1;
         elseif (randomseq(2*(i-1)+1:2*(i-1)+2)==10)
            x_sign(v)=amplitude*cos(2*pi*fc*timepoint(v)+pi/4+pi/2);
            d_sign(v)=2;
         else
            x_sign(v)=amplitude*cos(2*pi*fc*timepoint(v)+pi/4+pi);
            d_sign(v)=3;
         end;
      end;
   end;

   L=4;
   % Mapping vector for M - PSK Gray encoding
   k=log2(L);           % Number of bits per point

   ph1 = [pi/4];
   theta1 = [ph1; -ph1; pi-ph1; -pi+ph1];
   map = exp(1j*theta1);
   if(k>2)
      for j=3:k
         theta1=theta1/2;
         map=exp(1j*theta1);
         map=[map; -conj(map)];
         theta1=real(log(map)/1j);
      end;
   end;
   xstar_nonoise=(amplitude^2/2)^0.5*real(map);
   ystar_nonoise=(amplitude^2/2)^0.5*imag(map);

   figure(1)
   plot(xstar_nonoise,ystar_nonoise,'x');
   for i=1:1:L
      text(xstar_nonoise(i)-0.5,ystar_nonoise(i)-0.5,num2str(de2bi(i-
1,log2(L),'left-msb')), 'FontSize', 5);
   end;
   grid;
   axis([-max(xstar_nonoise)-1 max(xstar_nonoise)+1 -
max(xstar_nonoise)-1 max(xstar_nonoise)+1]);
   xlabel('(Eb/Tb)^0^.^5 (I)');
   ylabel('(Eb/Tb)^0^.^5 (Q)');
   title('Constellation Map for Q-PSK');

   figure(2)
   plot(timepoint,x_sign,'LineWidth',1.2);
```

```
    hold on;
    plot(timepoint,d_sign,'LineWidth',1.2);
    hold off;
    for i=1:samps_per_bit:length(d_sign)

text(timepoint(i)+0.1,d_sign(i)+0.5,num2str(de2bi(d_sign(i),log2(L),'
left-msb')), 'FontSize', 5);
    end;
    grid;
    axis([0 max(timepoint)+1 -1-max(x_sign) 1+max(x_sign)]);
    xlabel('Time (s)');
    ylabel('Signal Value');
    title('Q-PSK Signal');
```

Παράγεται μέσω της παραπάνω υλοποίησης, η τοποθέτηση των τεσσάρων συμβόλων στο χώρο αστερισμού λαμβάνοντας υπόψη την κωδικοποίηση Gray. Έτσι παράγεται το διάνυσμα mapping με βάση την τοποθέτηση των ζητούμενων φάσεων για τα σύμβολα.



Από το διάγραμμα παρατηρούμε ότι δίπλα από κάθε σύμβολο αναγράφεται η ακολουθία των 2 bits που του αντιστοιχεί.



Απεικονίζεται η QPSK μορφή του σήματος, τα δεδομένα εισόδου χωρίζονται σε δύο ακολουθίες, οι οποίες εμφανίζουν ρυθμό δεδομένων το μισό του αρχικού ρυθμού.

{Question b}

Για την παραγωγή του θορύβου AWGN έγινε χρήση της συνάρτησης normrnd() ώστε να δημιουργηθεί ακολουθία μηδενικής μέσης τιμής με ελεγχόμενη διασπορά, η οποία προκύπτει από το ζητούμενο SNR.

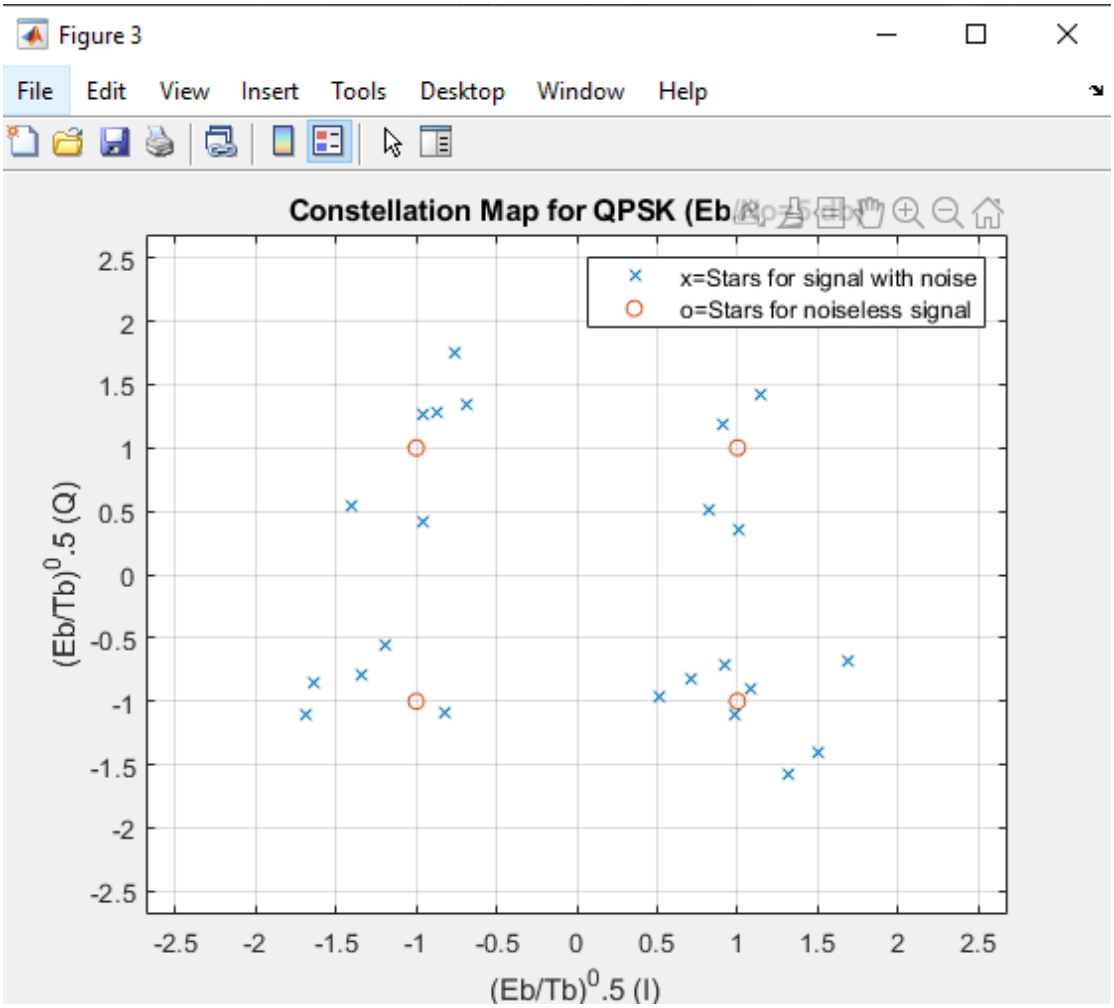
```
% Question b

    SNR=5;


xsymb=bi2de(reshape(randomseq,log2(L),length(randomseq)/log2(L)).','l
eft-msb');
    x_map=[];
    for k=1:length(xsymb)
        x_val(k)=real(map(xsymb(k)+1));
```

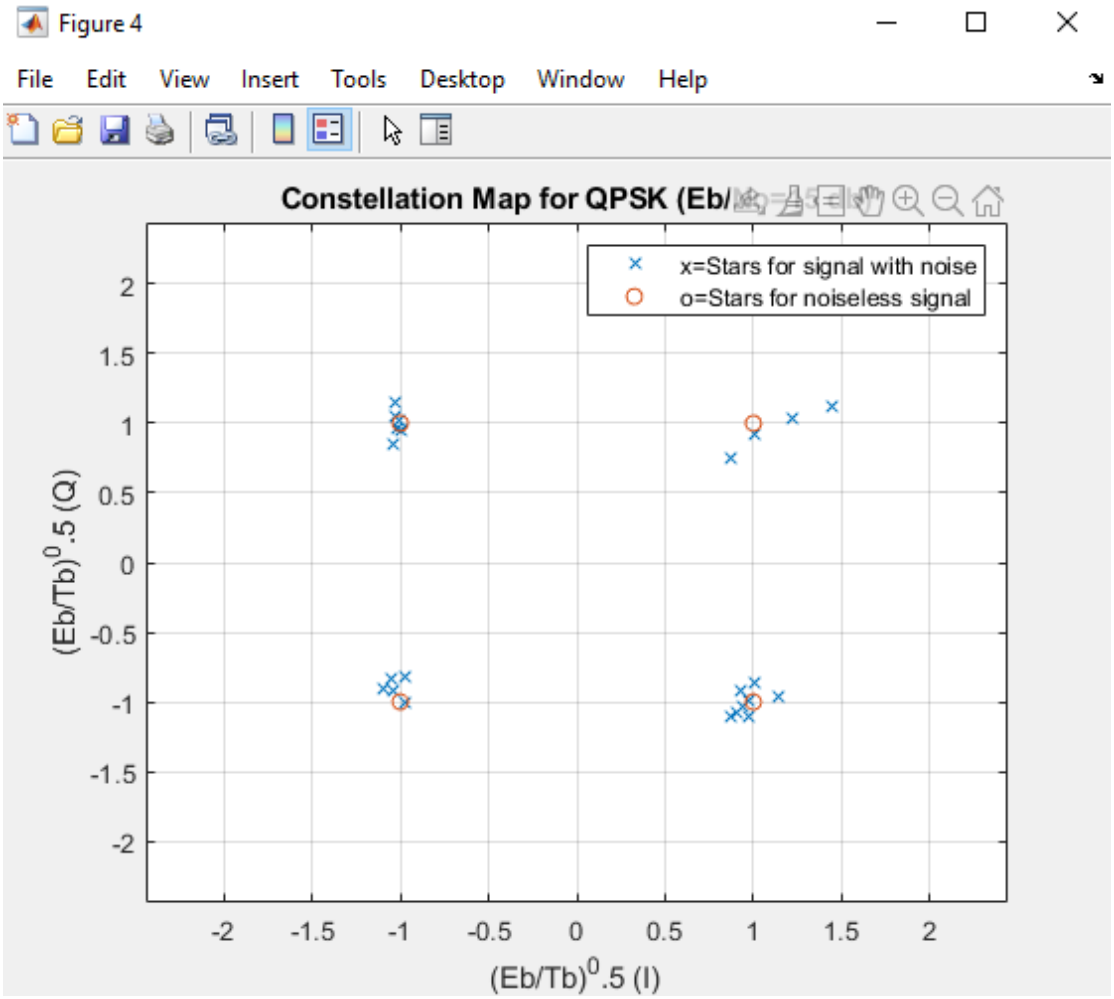
```
      y_val(k)=imag(map(xsymb(k)+1));
      x_map(k)=x_val(k)+exp(pi/2i)*y_val(k);
   end;
   x_map=(amplitude^2/2)^0.5*x_map;

   x_sign_energy=sum(abs(x_map).^2);

   noise_sign_energy=x_sign_energy/10^(SNR/10);
   noise_length=length(x_map);
   sigma_n=(noise_sign_energy/noise_length)^0.5;
   noise_sigr=normrnd(0,sigma_n,[1,length(x_map)]);
   noise_sigi=normrnd(0,sigma_n,[1,length(x_map)]);
   noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
   noise_sign_energy_calc=sum(abs(noise_sign).^2);

   SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
   SNRresult=SNR-SNRcalculation;
   noise_sign=10^(-SNRresult/10)*noise_sign;

   x_sign_n=x_map+noise_sign;

   xstar=real(x_sign_n);
   ystar=imag(x_sign_n);

   figure(3)
   plot(xstar,ystar,'x');
   hold on
   plot(xstar_nonoise,ystar_nonoise,'o');
   hold off
   grid;
   axis([-max(xstar)-1 max(xstar)+1 -max(xstar)-1 max(xstar)+1]);
   xlabel('(Eb/Tb)^0.5 (I)');
   ylabel('(Eb/Tb)^0.5 (Q)');
   title('Constellation Map for QPSK (Eb/No=5 db)');
   legend('x=Stars for signal with noise','o=Stars for noiseless
signal');



   SNR=15;


xsymb=bi2de(reshape(randomseq,log2(L),length(randomseq)/log2(L)).','l
eft-msb');
   x_map=[];
   for k=1:length(xsymb)
      x_val(k)=real(map(xsymb(k)+1));
      y_val(k)=imag(map(xsymb(k)+1));
      x_map(k)=x_val(k)+exp(pi/2i)*y_val(k);
   end;
   x_map=(amplitude^2/2)^0.5*x_map;

   x_sign_energy=sum(abs(x_map).^2);

   noise_sign_energy=x_sign_energy/10^(SNR/10);
   noise_length=length(x_map);
   sigma_n=(noise_sign_energy/noise_length)^0.5;
   noise_sigr=normrnd(0,sigma_n,[1,length(x_map)]);
```

```
    noise_sigi=normrnd(0,sigma_n,[1,length(x_map)]);
    noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
    noise_sign_energy_calc=sum(abs(noise_sign).^2);

    SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
    SNRresult=SNR-SNRcalculation;
    noise_sign=10^(-SNRresult/10)*noise_sign;

    x_sign_n=x_map+noise_sign;

    xstar=real(x_sign_n);
    ystar=imag(x_sign_n);

    figure(4)
    plot(xstar,ystar,'x');
    hold on
    plot(xstar_nonoise,ystar_nonoise,'o');
    hold off
    grid;
    axis([-max(xstar)-1 max(xstar)+1 -max(xstar)-1 max(xstar)+1]);
    xlabel('(Eb/Tb)^0.5 (I)');
    ylabel('(Eb/Tb)^0.5 (Q)');
    title('Constellation Map for QPSK (Eb/No=15 db)');
    legend('x=Stars for signal with noise','o=Stars for noiseless
signal');
```

Ο θόρυβος είναι μιγαδική συνάρτηση και προστίθεται στις τιμές μετάδοσης βάσει της QPSK διαμόρφωσης. Οπότε, τα αρχικά σύμβολα της διαμόρφωσης χωρίς θόρυβο μετατίθενται βάσει των πλατών του θορύβου κατά x και κατά y.

Για τη μετάδοση της ακολουθίας των 46 bits έχουμε τα παρακάτω διαγράμματα:





Διαγράμματα αστερσιμού για την QPSK με χρήση κωδικοποίησης Gray για SNR=5 και 15 db αντίστοιχα.

Τα μεταδιδόμενα σύμβολα είναι σχετικά κοντά με τα αντίστοιχα αθόρυβα. Με άλλα λόγια, μικρές τιμές του σηματοθορυβικού λόγου, αντιστοιχούν σε μεγαλύτερες απομακρύνσεις των συμβόλων από τις δεδομένες θέσεις του αστερισμού.

{Question c}

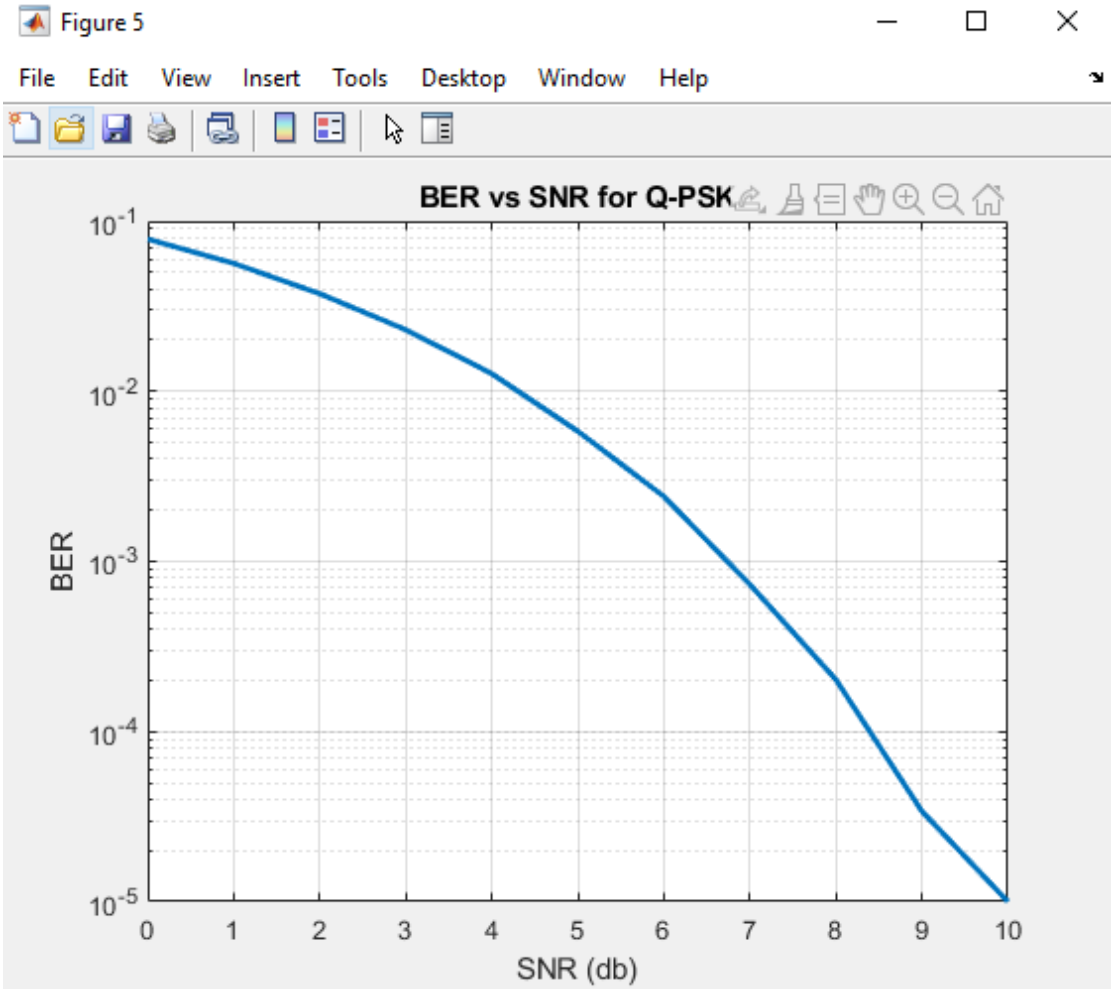
Για το ερώτημα χρειάστηκε επαναληπτικός βρόχος παραγωγής των ζητούμενων SNR για τη ζώνη 0 – 15 db με βήμα 1 db. Επίσης, χρειάστηκε μήκος «τυχαίας» ακολουθίας δειγμάτων ύψους 500.000 για να αυξήσει την ακρίβεια των αποτελεσμάτων.

```
% Question c - BER vs SNR

    format long;

    SNR_start=0;
```

```
    SNR_stop=15;
    SNR_step=1;
    SNR_iter=floor((SNR_stop-SNR_start)/SNR_step)+1;

    nbits=500000;
    Tbit=0.5;

    randomseq=(1+sign(randn(nbits,1)))/2;

    amplitude=9;
    samps_per_bit=1;


xsymb=bi2de(reshape(randomseq,log2(L),length(randomseq)/log2(L)).','l
eft-msb');
    x_map=[];
    for k=1:length(xsymb)
       x_val(k)=real(map(xsymb(k)+1));
       y_val(k)=imag(map(xsymb(k)+1));
       x_map(k)=x_val(k)+exp(pi/2i)*y_val(k);
    end;
    x_map=(amplitude^2/2)^0.5*x_map;

    x_sign_energy=sum(abs(x_map).^2);

    for s=1:1:SNR_iter

       SNR_st(s)=SNR_start+(s-1)*SNR_step;
       SNR=SNR_st(s);

       noise_sign_energy=x_sign_energy/10^(SNR/10);
       noise_length=length(x_map);
       sigma_n=(noise_sign_energy/noise_length)^0.5;
       noise_sigr=normrnd(0,sigma_n,[1,length(x_map)]);
       noise_sigi=normrnd(0,sigma_n,[1,length(x_map)]);
       noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
       noise_sign_energy_calc=sum(abs(noise_sign).^2);

       SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
       SNRresult=SNR-SNRcalculation;
       noise_sign=10^(-SNRresult/10)*noise_sign;

       x_sign_n=x_map+noise_sign;

       for i=1:1:nbits/2
          dist0=abs(x_sign_n(i)-amplitude*(1+exp(pi/2i)));
          dist1=abs(x_sign_n(i)-amplitude*(1-exp(pi/2i)));
          dist2=abs(x_sign_n(i)-amplitude*(-1+exp(pi/2i)));
          dist3=abs(x_sign_n(i)-amplitude*(-1-exp(pi/2i)));
          if (min(min(min(dist0,dist1),dist2),dist3)==dist0)
             randomseq_rx(1+2*(i-1))=0;
             randomseq_rx(2+2*(i-1))=0;
          elseif (min(min(min(dist0,dist1),dist2),dist3)==dist1)
             randomseq_rx(1+2*(i-1))=0;
             randomseq_rx(2+2*(i-1))=1;
          elseif (min(min(min(dist0,dist1),dist2),dist3)==dist2)
             randomseq_rx(1+2*(i-1))=1;
             randomseq_rx(2+2*(i-1))=0;
```

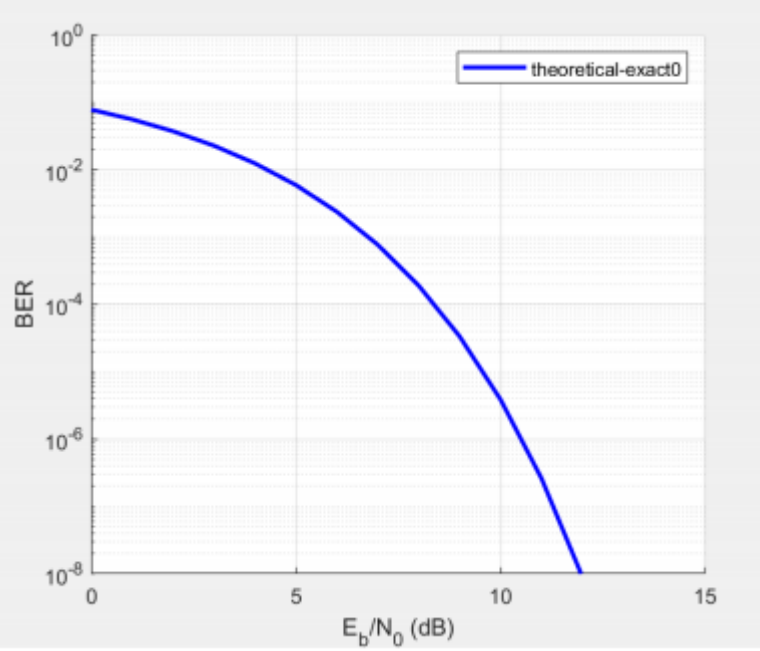
```
            else
                randomseq_rx(1+2*(i-1))=1;
                randomseq_rx(2+2*(i-1))=1;
            end;
        end;

        BER(s)=0;
        for i=1:1:nbits
            if (randomseq(i)~=randomseq_rx(i))
                BER(s)=BER(s)+1;
            end;
        end;
        BER_st(s)=BER(s)/nbits;

    end;

    figure(5)
    semilogy(SNR_st,BER_st,'LineWidth',2);
    grid;
    xlabel('SNR (db)');
    ylabel('BER');
    title('BER vs SNR for Q-PSK');
```

Ο κώδικας ελέγχει κάθε bit κατά τη λήψη της ακολουθίας αφού έχει αναγνωρίσει όλα τα σύμβολα βάσει του σχήματος αποκωδικοποίησης Gray. Η αναγνώριση του συμβόλου που λαμβάνουμε γίνεται χρησιμοποιώντας τις ευκλείδειες αποστάσεις του συμβόλου από το πιο κοντινό ιδανικό σύμβολο της περιοχής. Το αποτέλεσμα γίνεται ορατό σε λογαριθμικό διάγραμμα.



Σύμφωνα με το διάγραμμα, η αύξηση του SNR ελαττώνει το παρατηρούμενο επίπεδο σφαλμάτων (BER).

Για τη σύγκριση του παραγόμενου αποτελέσματος θεωρητικά έγινε χρήση της συνάρτησης bertool().



Από τη σύγκριση των δύο διαγραμμάτων, βλέπουμε ότι είναι παρόμοια. Η σύγκριση των επιδόσεων της QPSK με τη BPSK έγινε και πάλι μέσω της bertool().

{Question d}

Για το ερώτημα αυτό χρησιμοποιήθηκε το αρχείο shannon _even.txt λόγω της αρτιότητας του αθροίσματος των τρειών τελευταίων ψηφίων του αριθμού μητρώου.

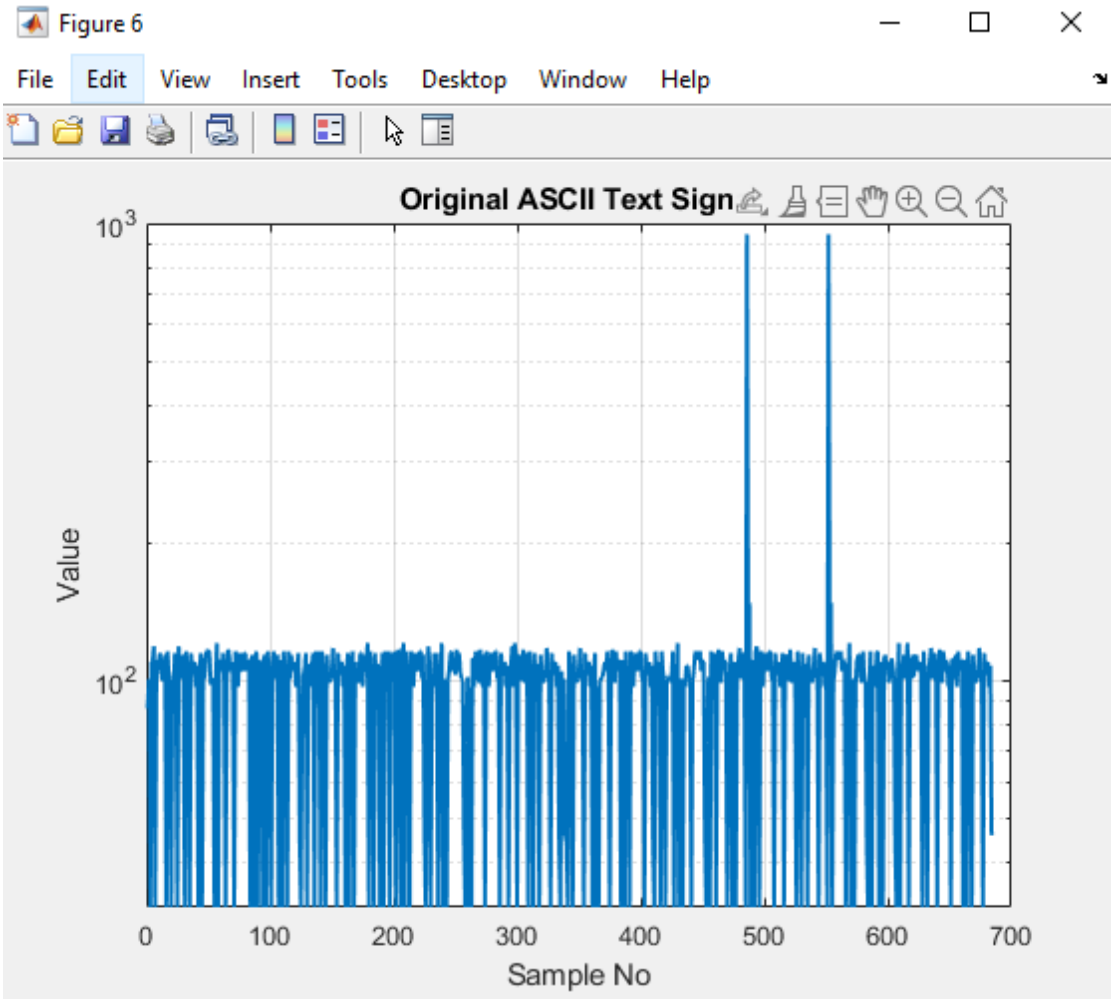
Για τη μετάδοση του αρχείου χρησιμοποιήθηκε η QPSK διαμόρφωση που υλοποιήθηκε πριν με πλάτος A = 2V και χρήση κωδικοποίησης Gray.

{i} Για την ανάγνωση του αρχείου και τη μετατροπή των ASCII χαρακτήρων σε δυαδική μορφή έχουμε:

```
% Question d

%(i)
    filename='shannon_even.txt';
    fid=fopen(filename);
    st_struct_str=fscanf(fid,'%c');
    ld=fclose(fid);
```

```
disp('--------------------------------------');
disp(' Send ASCII ');
disp('--------------------------------------');
disp(st_struct_str)

st_struct=double(st_struct_str);

sym_bits=8;
for i=1:1:length(st_struct)
   binfile(i,:)=de2bi(st_struct(i),sym_bits,'left-msb');
end;

v=0;
for i=1:1:length(st_struct)
   for j=1:1:sym_bits
      v=v+1;
      bin_stream(v)=binfile(i,j);
   end;
end;

figure(6)
i=1:1:length(st_struct);
semilogy(i,st_struct,'LineWidth',1.5);
grid;
xlabel('Sample No');
ylabel('Value');
title('Original ASCII Text Signal');
```

Ο κώδικας διαβάζει το αρχείο μέσω της συνάρτησης fscanf(). Μετά, παρέχει τις τιμές ανάγνωσης ως characters στο διάνυσμα st_struct(). Αλλάζει το vector χαρακτήρων στους αριθμούς που αντιστοιχούν μέσω της εντολής double(), και κατόπιν μετατρέπει σε binary level με τη βοήθεια της συνάρτησης de2bi() η οποία μετατρέπει μία δεκαδική τιμή σε δυαδική με κατάλληλη τοποθέτηση LSB και MSB.



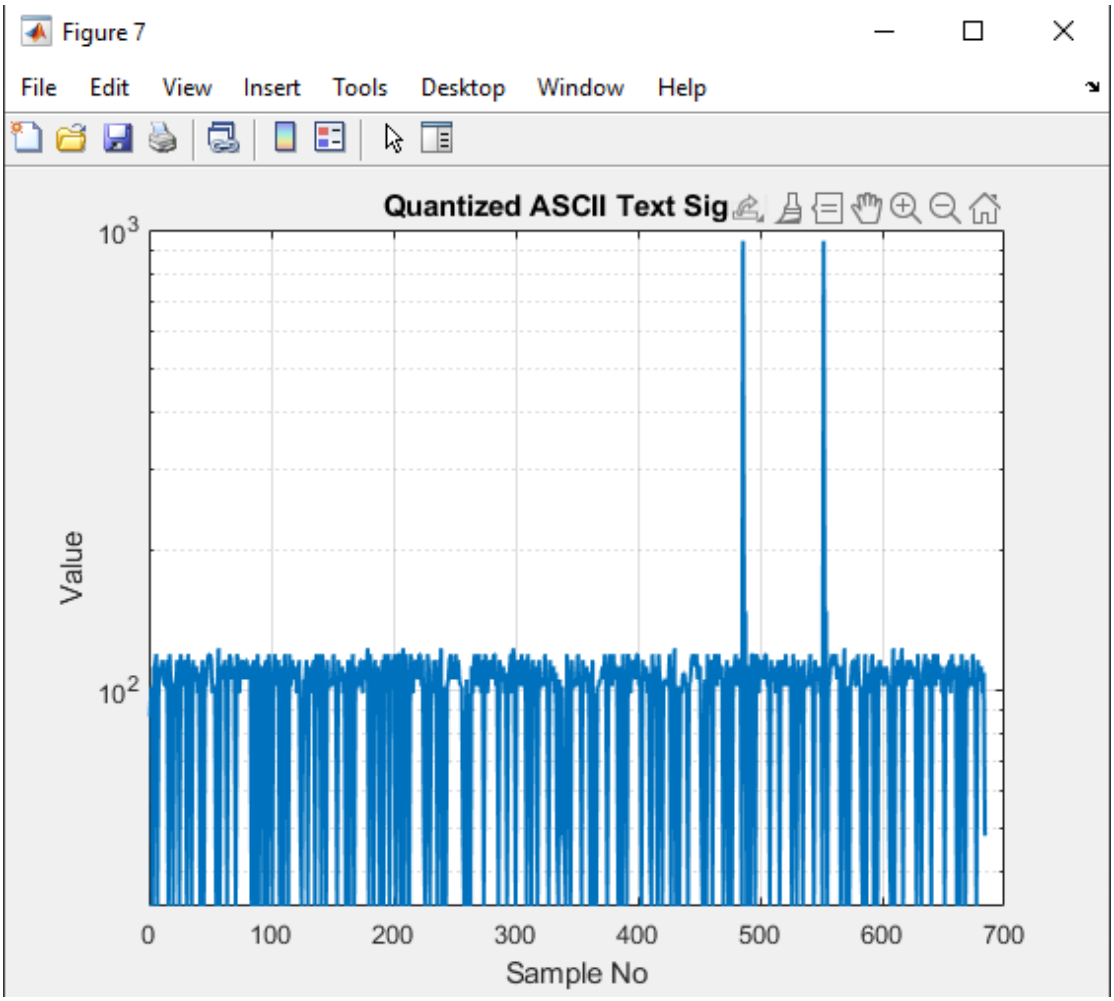
Χαρακτήρες ASCII κειμένου μετά τη μετατροπή τους σε δεκαδικούς αριθμούς

{ii} Όμοια υλοποιημένος ο κβαντιστής σύμφωνα με τα παραπάνω:

```
%(ii)
    Amax=max(st_struct);
    Amin=min(st_struct);
    n=8;
    N=length(st_struct);

    delta=(Amax-Amin)/2^n;

    partition(1) = Amin+delta/2;
    for i=1:1:2^n-1
       partition(i+1)=partition(i)+delta;
    end;

    x_ind = quantiz(st_struct,partition);

    for i=1:1:N
       if (x_ind(i)+1>2^n)
          x_qnd(i)=partition(2^n)+delta/2;
```

```
    else
        x_qnd(i)=partition(x_ind(i)+1);
    end;
end;

x_bin=de2bi(floor(x_qnd),sym_bits,'left-msb');

v=0;
for i=1:1:length(st_struct)
    for j=1:1:sym_bits
        v=v+1;
        x_bin_str(v)=x_bin(i,j);
    end;
end;

figure(7)
i=1:1:length(x_qnd);
semilogy(i,x_qnd,'LineWidth',1.5);
grid;
xlabel('Sample No');
ylabel('Value');
title('Quantized ASCII Text Signal');
```

Ο κβαντιστής ορίζεται από το διάνυσμα partition() που δημιουργείται από το πλήθος των 256 σταθμών με βήμα delta.



Χαρακτήρες ASCII κειμένου μετά τη μετατροπή τους σε δεκαδικούς αριθμούς και χρήση του κβαντιστή.

Από τη σύγκριση των δύο διαγραμμάτων δεν παρατηρούνται μεγάλες διαφορές, λόγω των πολλών σταθμών με μικρό βήμα.

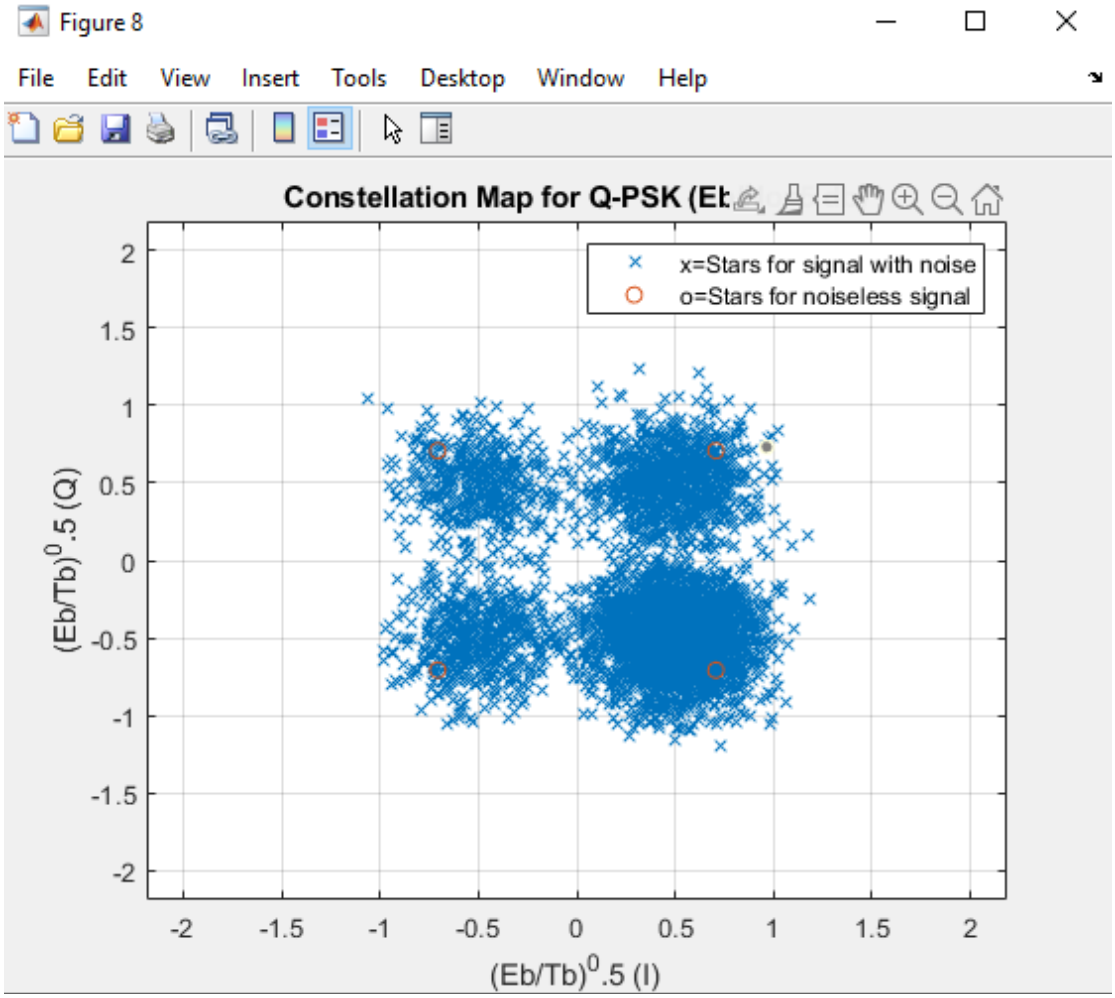
{iii-iv} Για την υλοποίηση της QPSK και την προσθήκη ελεγχόμενου θορύβου AWGN έχουμε:

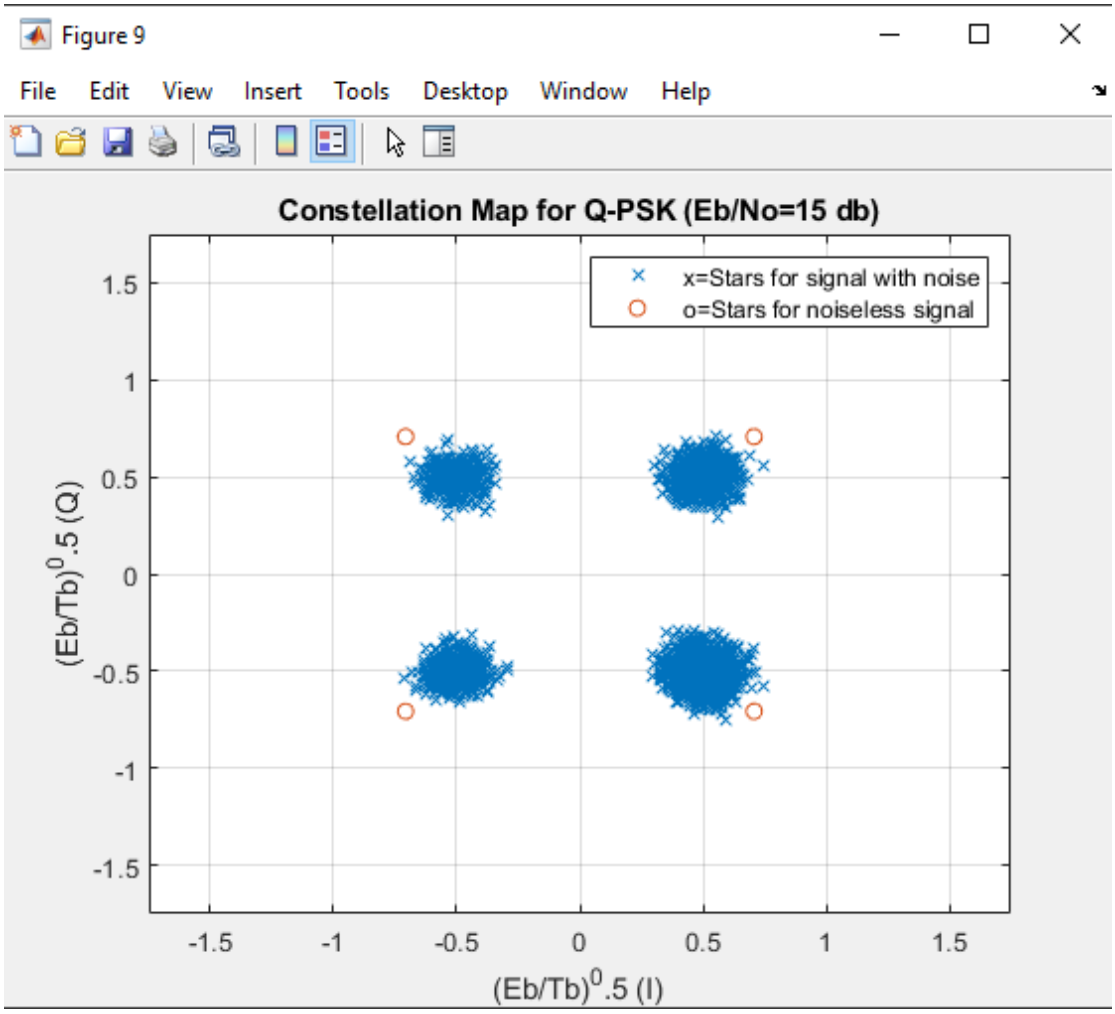
```
%(iii-iv)

    SNR=5;
    amplitude=1;


xsymb=bi2de(reshape(x_bin_str,log2(L),length(x_bin_str)/log2(L)).','l
eft-msb');
    x_map=[];
```

```
    for k=1:length(xsymb)
       x_val(k)=real(map(xsymb(k)+1));
       y_val(k)=imag(map(xsymb(k)+1));
       x_map(k)=x_val(k)+exp(pi/2i)*y_val(k);
    end;
    x_map=(amplitude^2/2)^0.5*x_map;

    x_sign_energy=sum(abs(x_map).^2);

    noise_sign_energy=x_sign_energy/10^(SNR/10);
    noise_length=length(x_map);
    sigma_n=(noise_sign_energy/noise_length)^0.5;
    noise_sigr=normrnd(0,sigma_n,[1,length(x_map)]);
    noise_sigi=normrnd(0,sigma_n,[1,length(x_map)]);
    noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
    noise_sign_energy_calc=sum(abs(noise_sign).^2);

    SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
    SNRresult=SNR-SNRcalculation;
    noise_sign=10^(-SNRresult/10)*noise_sign;

    x_sign_n_5=x_map+noise_sign;

    xstar=real(x_sign_n_5);
    ystar=imag(x_sign_n_5);
    xstar_nonoise=real(map);
    ystar_nonoise=imag(map);

    figure(8)
    plot(xstar,ystar,'x');
    hold on
    plot(xstar_nonoise,ystar_nonoise,'o');
    hold off
    grid;
    axis([-max(xstar)-1 max(xstar)+1 -max(xstar)-1 max(xstar)+1]);
    xlabel('(Eb/Tb)^0.5 (I)');
    ylabel('(Eb/Tb)^0.5 (Q)');
    title('Constellation Map for Q-PSK (Eb/No=5 db)');
    legend('x=Stars for signal with noise','o=Stars for noiseless
signal');

    SNR = 15;

    noise_sign_energy=x_sign_energy/10^(SNR/10);
    noise_length=length(x_map);
    sigma_n=(noise_sign_energy/noise_length)^0.5;
    noise_sigr=normrnd(0,sigma_n,[1,length(x_map)]);
    noise_sigi=normrnd(0,sigma_n,[1,length(x_map)]);
    noise_sign=noise_sigr+noise_sigi*exp(pi/2i);
    noise_sign_energy_calc=sum(abs(noise_sign).^2);

    SNRcalculation=10*log10(x_sign_energy/noise_sign_energy_calc);
    SNRresult=SNR-SNRcalculation;
    noise_sign=10^(-SNRresult/10)*noise_sign;

    x_sign_n_15=x_map+noise_sign;

    xstar=real(x_sign_n_15);
```

```
    ystar=imag(x_sign_n_15);
    xstar_nonoise=real(map);
    ystar_nonoise=imag(map);

    figure(9)
    plot(xstar,ystar,'x');
    hold on
    plot(xstar_nonoise,ystar_nonoise,'o');
    hold off
    grid;
    axis([-max(xstar)-1 max(xstar)+1 -max(xstar)-1 max(xstar)+1]);
    xlabel('(Eb/Tb)^0.5 (I)');
    ylabel('(Eb/Tb)^0.5 (Q)');
    title('Constellation Map for Q-PSK (Eb/No=15 db)');
    legend('x=Stars for signal with noise','o=Stars for noiseless
signal');
```

Ο κώδικας κάνει την χαρτογράφηση των τιμών του αρχείου προς μετάδοση στα σύμβολα της QPSK. Εν συνεχεία, προσθέτει AWGN θόρυβο με ελεγχόμενο SNR (5 – 15 db). Ο θόρυβος που προστέθηκε έτσι, τα παραγόμενα σύμβολα στο χώρο αστερισμού θα έχουν μετακινηθεί τόσο στον άξονα x όσο και στον άξονα y από τις αρχικές θέσεις.





Τα σύμβολα είναι πιο κοντά στις ιδανικές θέσεις για το ισχυρότερο SNR, ενώ είναι πιο απομακρυσμένα για χαμηλότερο SNR.

{v-vi} Για την εύρεση του BER μετά από τη διαδικασία της αναγνώρισης των συμβόλων, συγκρίθηκε κάθε bit της ακολουθίας που λάβαμε και για τις δύο εκδοχές μετάδοσης του σηματοθορυβικού λόγου.

```
% (v-vi)

    SNR = 5;

    for i=1:1:length(x_sign_n_5)
       dist0=abs(x_sign_n_5(i)-amplitude*(1+exp(pi/2i)));
       dist1=abs(x_sign_n_5(i)-amplitude*(1-exp(pi/2i)));
       dist2=abs(x_sign_n_5(i)-amplitude*(-1+exp(pi/2i)));
       dist3=abs(x_sign_n_5(i)-amplitude*(-1-exp(pi/2i)));
       if (min(min(min(dist0,dist1),dist2),dist3)==dist0)
          xbinstream_rx(1+2*(i-1))=0;
          xbinstream_rx(2+2*(i-1))=0;
       elseif (min(min(min(dist0,dist1),dist2),dist3)==dist1)
```

```
            xbinstream_rx(1+2*(i-1))=0;
            xbinstream_rx(2+2*(i-1))=1;
        elseif (min(min(min(dist0,dist1),dist2),dist3)==dist2)
            xbinstream_rx(1+2*(i-1))=1;
            xbinstream_rx(2+2*(i-1))=0;
        else
            xbinstream_rx(1+2*(i-1))=1;
            xbinstream_rx(2+2*(i-1))=1;
        end;
    end;

    BER_file=0;
    for i=1:1:length(x_bin_str)
        if (x_bin_str(i)~=xbinstream_rx(i))
            BER_file=BER_file+1;
        end;
    end;
    BER_file=BER_file/length(x_bin_str);

    for i=1:1:N
        ststruct_rx(i)=bi2de(xbinstream_rx(1+(i-
1)*sym_bits:sym_bits+(i-1)*sym_bits),'left-msb');
    end;

    disp('----------------------------------------');
    disp(' Used SNR for file Tx ');
    disp(SNR);
    disp(' Achieved BER (%) ');
    disp(100*BER_file);

    SNR = 15;

        for i=1:1:length(x_sign_n_15)
        dist0=abs(x_sign_n_15(i)-amplitude*(1+exp(pi/2i)));
        dist1=abs(x_sign_n_15(i)-amplitude*(1-exp(pi/2i)));
        dist2=abs(x_sign_n_15(i)-amplitude*(-1+exp(pi/2i)));
        dist3=abs(x_sign_n_15(i)-amplitude*(-1-exp(pi/2i)));
        if (min(min(min(dist0,dist1),dist2),dist3)==dist0)
            xbinstream_rx_15(1+2*(i-1))=0;
            xbinstream_rx_15(2+2*(i-1))=0;
        elseif (min(min(min(dist0,dist1),dist2),dist3)==dist1)
            xbinstream_rx_15(1+2*(i-1))=0;
            xbinstream_rx_15(2+2*(i-1))=1;
        elseif (min(min(min(dist0,dist1),dist2),dist3)==dist2)
            xbinstream_rx_15(1+2*(i-1))=1;
            xbinstream_rx_15(2+2*(i-1))=0;
        else
            xbinstream_rx_15(1+2*(i-1))=1;
            xbinstream_rx_15(2+2*(i-1))=1;
        end;
    end;

    BER_file=0;
    for i=1:1:length(x_bin_str)
        if (x_bin_str(i)~=xbinstream_rx_15(i))
            BER_file=BER_file+1;
        end;
    end;
    BER_file=BER_file/length(x_bin_str);
```

```
   for i=1:1:N
      ststruct_rx_15(i)=bi2de(xbinstream_rx_15(1+(i-
1)*sym_bits:sym_bits+(i-1)*sym_bits),'left-msb');
   end;

   disp('---------------------------------------');
   disp(' Used SNR for file Tx ');
   disp(SNR);
   disp(' Achieved BER (%) ');
   disp(100*BER_file);
```

Ο κώδικας μετά την ολοκλήρωση της διαδικασίας δίνει:

```
Used SNR for file Tx
     5

 Achieved BER (%)
   0.603070175438596

---------------------------------------
 Used SNR for file Tx
    15

 Achieved BER (%)
     0
```

Για SNR = 5 db το BER αναμενόταν θεωρητικά 0.50118 %. Πρακτικά, παρατηρούμε απόκλιση στην περίπτωση SNR = 5db και πλήρη ταύτιση θεωρητικού και πρακτικού υπολογισμού στην περίπτωση SNR = 15db.

{vii} Για την αποθήκευση των δεδομένων σε αρχείο .txt χρησιμοποιήθηκε η συνάρτηση native2unicode() η οποία κάνει τη μετατροπή από δεκαδική μορφή σε χαρακτήρες ASCII.

```
%(vii)

   out_string=native2unicode(ststruct_rx,'ASCII');

   filename_out='shannon_even_rx_5db.txt';
   fid=fopen(filename_out,'w');
   outf=fprintf(fid,'%c',out_string);
   ld=fclose(fid);

   disp('---------------------------------------');
   disp(' Recieved ASCII ');
   disp('---------------------------------------');
   disp(out_string)

   out_string=native2unicode(ststruct_rx_15,'ASCII');

   filename_out='shannon_even_rx_15db.txt';
```

```
fid=fopen(filename_out,'w');
outf=fprintf(fid,'%c',out_string);
ld=fclose(fid);

disp('---------------------------------------');
disp(' Recieved ASCII ');
disp('---------------------------------------');
disp(out_string)
```

Για την εγγραφή σε αρχείο .txt χρησιμοποιήθηκε η συνάρτηση fprintf().

Send ASCII

We now consider the case where the signal is perturbed by noise during transmission or at one or the other of the terminals. This means that the received signal is not necessarily the same as that sent out by the transmitter. Two cases may be distinguished. If a particular transmitted signal always produces the same received signal, i.e., the received signal is a definite function of the transmitted signal, then the effect may be called distortion. If this function has an inverse - no two transmitted signals producing the same received signal - distortion may be corrected, at least in principle, by merely performing the inverse functional operation on the received signal.

Used SNR for file Tx

5

Achieved BER (%)

0.698529411764706

Used SNR for file Tx

15

Achieved BER (%)

0

Recieved ASCII

We now consider the case w`ere the sig.al is perturb�d by noise`du�ing transmission or ad one or the other of txe terminaLs. This me`ns that the�received siglal is not necessarily the same as that seft out by the |rensmitter. Two cases oay be distinguished. If a particular pransmitted signal always produces the same received signal, i.e., the reGeived signal is a defin)te fungtion of the t�ansmitted signal, then the effect may b� called distortion. If th�s functiol hqs an inverse - no two transmitted signals pr/d5cing the same received signal - distortion m�y be corrected, at least in prmnciple, by(merely xerforming the inverse f}nctiona| opereti�n on the receiwed sygnal.

Recieved ASCII

We now consider the case where the signal is perturbed by noise during transmission or at one or the other of the terminals. This means that the received signal is not necessarily the same as that sent out by the transmitter. Two cases may be distinguished. If a particular transmitted signal always produces the same received signal, i.e., the received signal is a definite function of the transmitted signal, then the effect may be called distortion. If this function has an inverse - no two transmitted signals producing the same received signal - distortion may be corrected, at least in principle, by merely performing the inverse functional operation on the received signal.

Στα 15db η αναπαραγωγή του σήματος είναι ικανοποιητική ενώ στην περίπτωση των 5db δεν συμβαίνει κάτι τέτοιο.

Ερώτημα 5

{Question a}

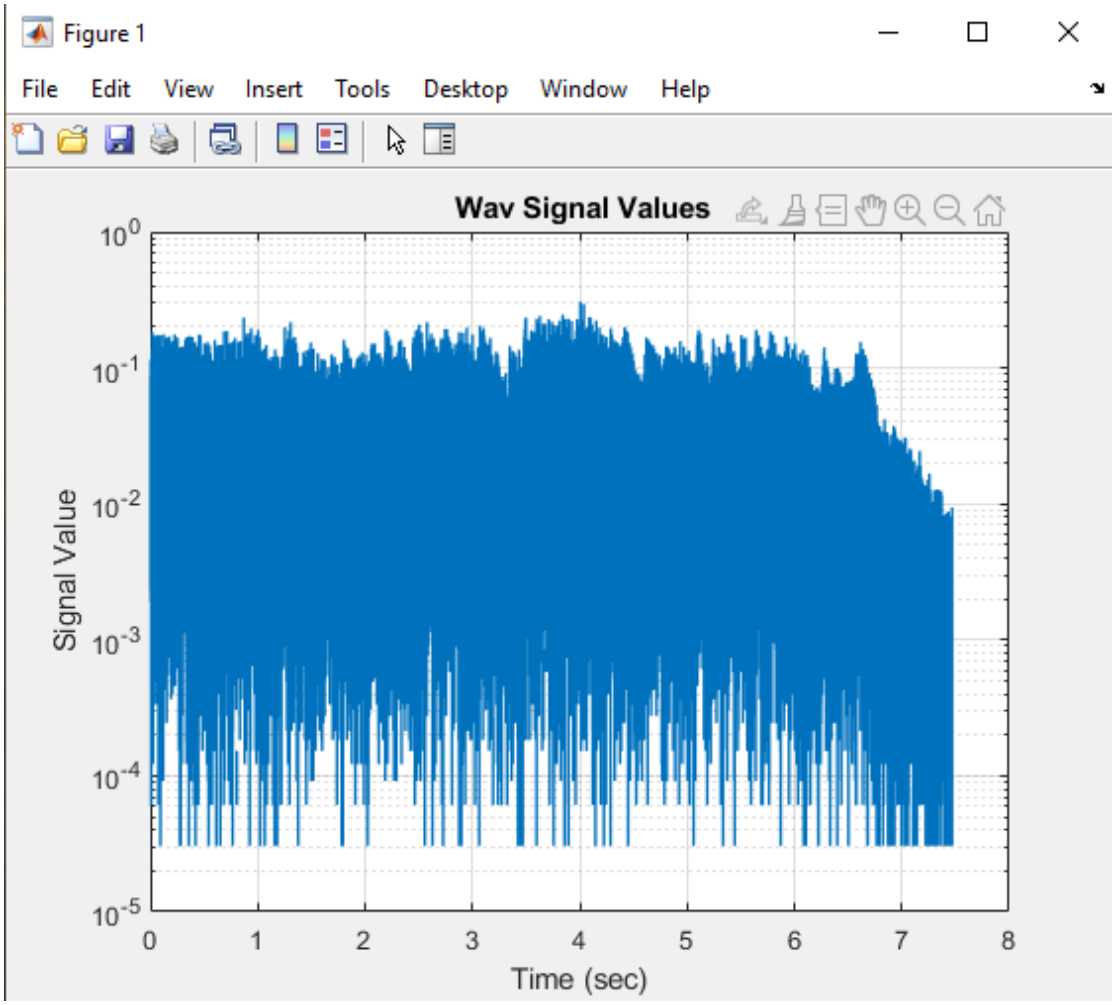
Για το διάβασμα του αρχείου ήχου σε format wav χρησιμοποιήθηκε η συνάρτηση audioread().

```
% Vikentios Vitalis el18803
% 8 + 3 = 11 = 1 + 1 = 2

clc;
clear;
format short;

%   Question a - Wav Read


   AM_sum=8+0+3;

   if (mod(AM_sum,2)==0)
      filename='soundfile2_lab2.wav';
   else
      filename='soundfile1_lab2.wav';
   end;

   [st_struct,fs]=audioread(filename);

   Amax=max(st_struct);
   Amin=min(st_struct);

   for i=1:1:length(st_struct)
      st_struct_norm(i)=floor(255*(st_struct(i)-Amin)/(Amax-Amin));
   end;

   sym_bits=8;
   for i=1:1:length(st_struct_norm)
      binfile(i,:)=de2bi(st_struct_norm(i),sym_bits,'left-msb');
      if (mod(i,1000)==0)
         clc;
         disp(' Binary Conversion Progress (%)');
         disp(100*i/length(st_struct_norm));
      end;
   end;
   clc;

   v=0;
   for i=1:1:length(st_struct)
```

```
    for j=1:1:sym_bits
        v=v+1;
        bin_stream(v)=binfile(i,j);
    end;
end;

figure(1)
i=1:1:length(st_struct);
semilogy((i-1)/fs,st_struct,'LineWidth',1);
grid;
xlabel(' Time (sec) ');
ylabel(' Signal Value ');
title(' Wav Signal Values');

sound(st_struct,fs);
```

Ο κώδικας διαβάζει το αρχείο ήχου μέσω της audioread() και αποθηκεύει τις τιμές του σήματος στο διάνυσμα st_struct(). Οι τιμές του αρχείου του ήχου ανήκουν μέσα στο διάστημα [-1,1].

Διάγραμμα τιμών του σήματος ήχου πριν τη διαδικασία κανονικοποίησης.



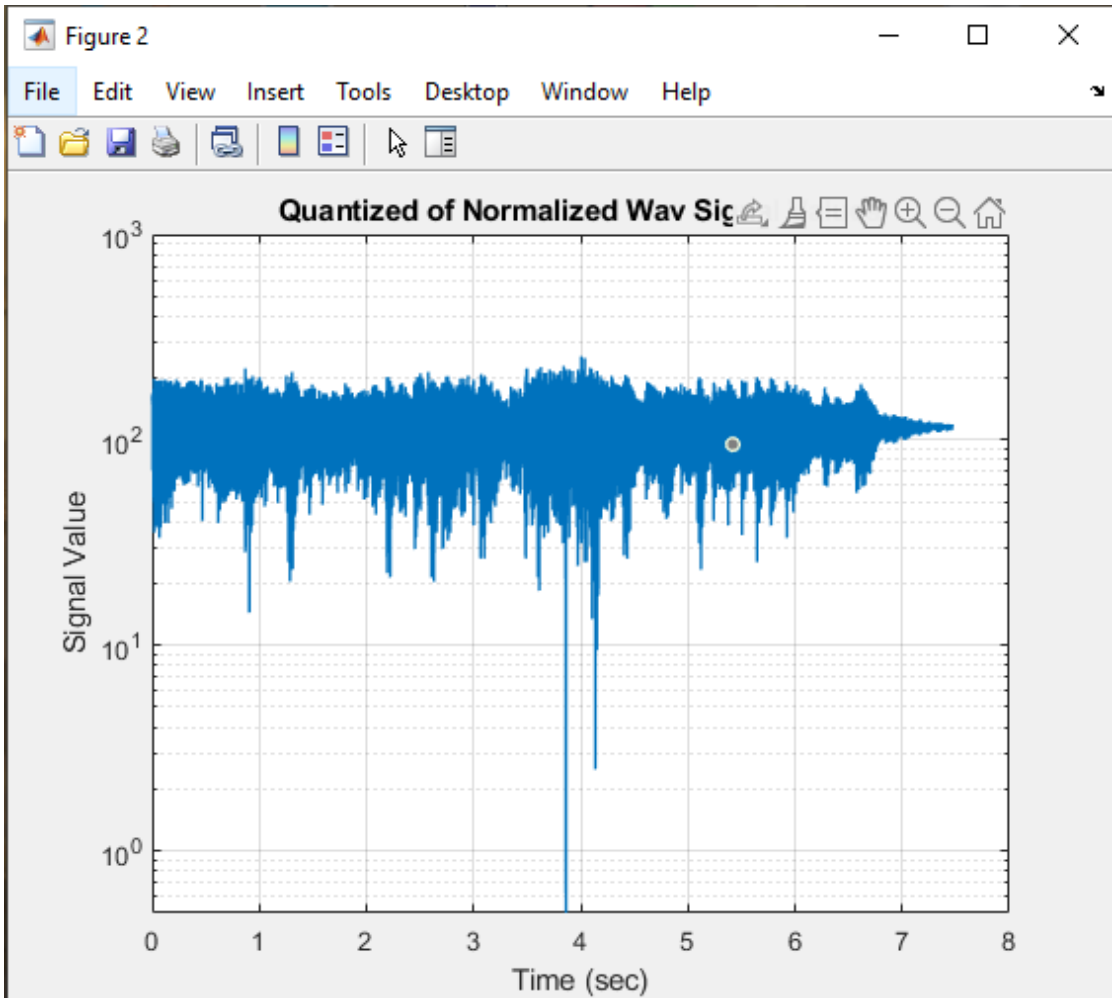
Η χρονική διάρκεια του αρχείου ανακατασκευάστηκε με χρήση τη συχνότητας δειγματοληψίας η οποία προέκυψε από την audioread() σε τιμή 44.100 Hz.

{Question b}

Επειδή ο κβαντιστής που χρησιμοποιούμε θα δώσει τα παραγόμενα δείγματα για μετάδοση, κάναμε την κανονικοποίηση προκειμένου να είναι απρόσημοι αριθμοί.

```
% Question b - Quantizer


Amax_norm=max(st_struct_norm);
Amin_norm=min(st_struct_norm);
n=8;
N=length(st_struct_norm);

delta=(Amax_norm-Amin_norm)/2^n;

partition(1) = Amin_norm+delta/2;
for i=1:1:2^n-1
   partition(i+1)=partition(i)+delta;
end;

x_ind=quantiz(st_struct_norm,partition);

for i=1:1:N
   if (x_ind(i)+1>2^n)
      x_qnd(i)=partition(2^n)+delta/2;
   else
      x_qnd(i)=partition(x_ind(i)+1);
   end;
end;

sym_bits=8;
x_bin=de2bi(floor(x_qnd),sym_bits,'left-msb');

v=0;
for i=1:1:length(st_struct)
   for j=1:1:sym_bits
      v=v+1;
      x_bin_stream(v)=x_bin(i,j);
   end;
end;

figure(2)
i=1:1:length(x_qnd);
semilogy((i-1)/fs,x_qnd,'LineWidth',1);
grid;
xlabel(' Time (sec) ');
ylabel(' Signal Value ');
title('Quantized of Normalized Wav Signal Values');
```

Τα παραγόμενα κβαντισμένα δέιγματα του σήματος μετά τη διαδικασία κανονικοποίησης:



Κανονικοποιήμενο σήμα στο χρόνο μετά τον κβαντιστή.

{Questions c,d}

Για την υλοποίηση του QPSK διαμορφωτή και την προσθήκη AWGN έχουμε:

```
%   Questions c-d - Q-PSK


   SNR=14;
   Amp=1;

   L=4;
   % Mapping vector for M - PSK Gray encoding
   k=log2(L);           % Number of bits per point

   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=real(log(mapping)/1j);
        end;
    end;
    xstar_noiseless=(Amp^2/2)^0.5*real(mapping);
    ystar_noiseless=(Amp^2/2)^0.5*imag(mapping);


xsym=bi2de(reshape(x_bin_stream,log2(L),length(x_bin_stream)/log2(L))
.','left-msb');
    x_map=[];
    for k=1:length(xsym)
        x_val(k)=real(mapping(xsym(k)+1));
        y_val(k)=imag(mapping(xsym(k)+1));
        x_map(k)=x_val(k)+exp(pi/2i)*y_val(k);
    end;
    x_map=(Amp^2/2)^0.5*x_map;

    x_sig_energy=sum(abs(x_map).^2);

    noise_sig_energy=x_sig_energy/10^(SNR/10);
    noise_length=length(x_map);
    sigma_n=(noise_sig_energy/noise_length)^0.5;
    noise_sigr=normrnd(0,sigma_n,[1,length(x_map)]);
    noise_sigi=normrnd(0,sigma_n,[1,length(x_map)]);
    noise_sig=noise_sigr+noise_sigi*exp(pi/2i);
    noise_sig_energy_calc=sum(abs(noise_sig).^2);

    SNR_calc=10*log10(x_sig_energy/noise_sig_energy_calc);
    SNR_res=SNR-SNR_calc;
    noise_sig=10^(-SNR_res/10)*noise_sig;

    x_sig_n=x_map+noise_sig;

    xstar=real(x_sig_n);
    ystar=imag(x_sig_n);
    xstar_noiseless=real(mapping);
    ystar_noiseless=imag(mapping);

    figure(3)
    plot(xstar,ystar,'x');
    hold on
    plot(xstar_noiseless,ystar_noiseless,'o');
    hold off
    grid;
    axis([-max(xstar)-1 max(xstar)+1 -max(xstar)-1 max(xstar)+1]);
    xlabel(' (Eb/Tb)^0.5 (I) ');
    ylabel(' (Eb/Tb)^0.5 (Q) ');
    title('Constellation Map for Q-PSK (Eb/No=14 db)');
    legend('x=Stars for signal with noise','o=Stars for noiseless
signal');
```

Γίνεται χαρτογράφηση του διανύσματος για την QPSK διαμόρφωση και προσθέτει μιγαδικό θόρυβο με καθορισμένη τιμή διασποράς.

{Question e}

Τα διαγράμματα αστερισμών προέκυψαν από την προσθήκη του θορύβου επί των ιδανικών συμβόλων μετάδοσης της QPSK. Η μιγαδική υφή του θορύβου αναμένεται να μετατοπίσει τα σύμβολα κατά χ και y από την ιδανική τους θέση.

Αποδιαμόρφωση για την εξαγωγή των συμβόλων:

```
%   Question e - Demodulation of Q-PSK


   for i=1:1:length(x_sig_n)
      dist0=abs(x_sig_n(i)-Amp*(1+exp(pi/2i)));
      dist1=abs(x_sig_n(i)-Amp*(1-exp(pi/2i)));
      dist2=abs(x_sig_n(i)-Amp*(-1+exp(pi/2i)));
      dist3=abs(x_sig_n(i)-Amp*(-1-exp(pi/2i)));
      if (min(min(min(dist0,dist1),dist2),dist3)==dist0)
         x_bin_stream_rx(1+2*(i-1))=0;
         x_bin_stream_rx(2+2*(i-1))=0;
      elseif (min(min(min(dist0,dist1),dist2),dist3)==dist1)
         x_bin_stream_rx(1+2*(i-1))=0;
         x_bin_stream_rx(2+2*(i-1))=1;
      elseif (min(min(min(dist0,dist1),dist2),dist3)==dist2)
         x_bin_stream_rx(1+2*(i-1))=1;
         x_bin_stream_rx(2+2*(i-1))=0;
      else
         x_bin_stream_rx(1+2*(i-1))=1;
         x_bin_stream_rx(2+2*(i-1))=1;
      end;
   end;
```

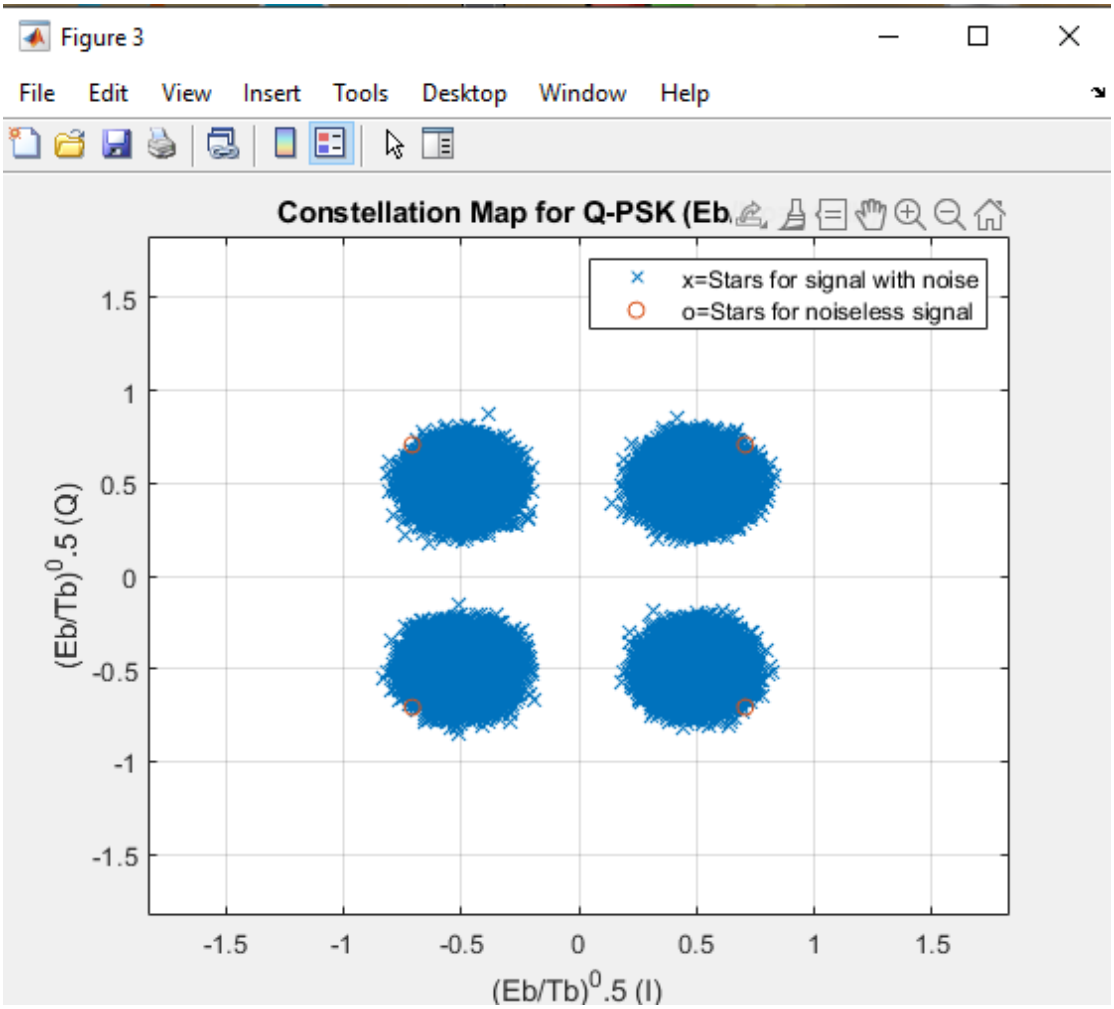
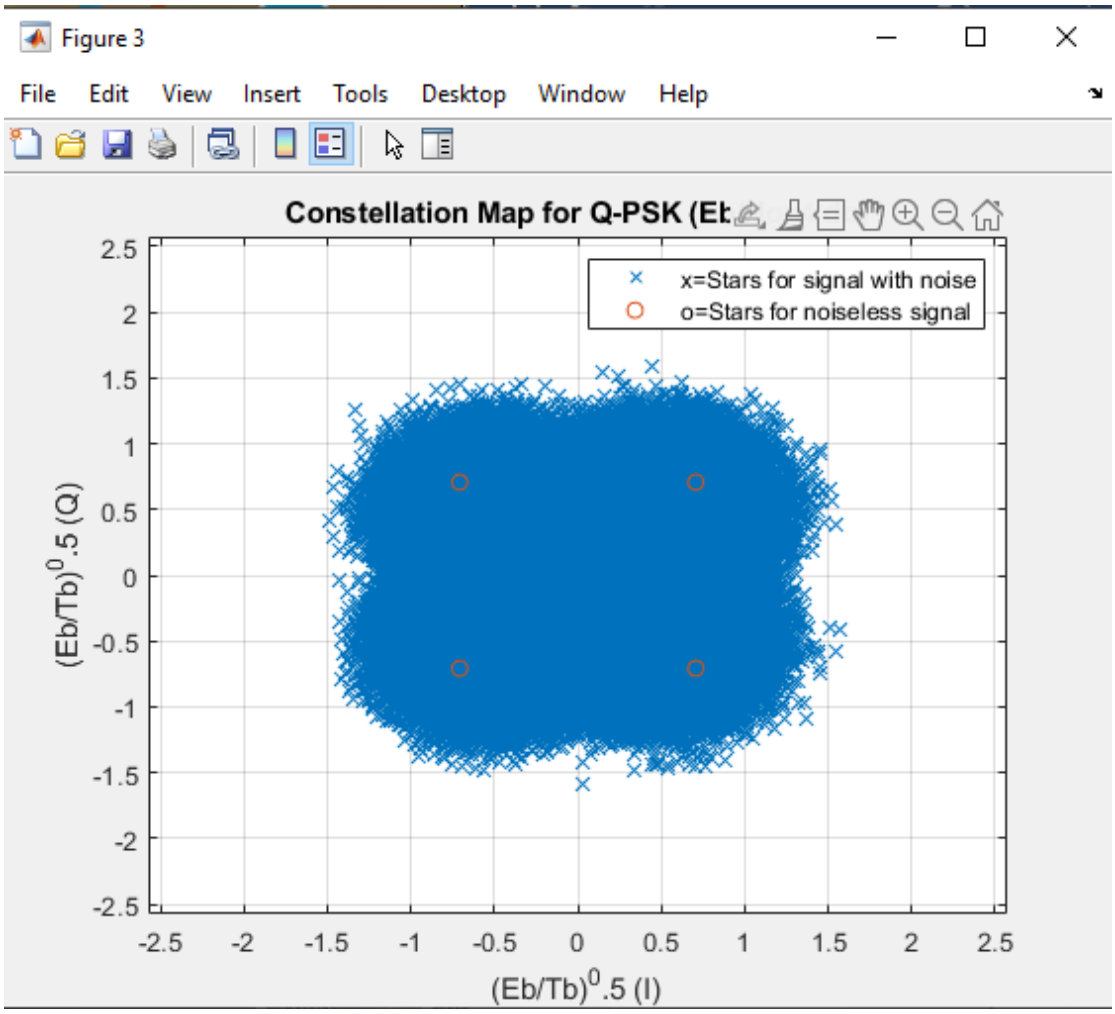


Figure 3
File Edit View Insert Tools Desktop Window Help
Constellation Map for Q-PSK (Eb
x=Stars for signal with noise
o=Stars for noiseless signal
(Eb/Tb)^0.5 (Q)
(Eb/Tb)^0.5 (I)
1.5
1
0.5
0
-0.5
-1
-1.5



Διαγράμματα αστερισμού του QPSK διαμορφωμένου σήματος με AWGN για SNR=14 και 4 db αντίστοιχα.

Η απόφαση της θέσης λαμβανόμενου συμβόλου γίνεται με χρήση της ευκλείδειας νόρμας απόστασης από τις ιδανικές τιμές των συμβόλων του αστερισμού.

{Question st}

Μετά τη διαδικασία της αποδιαμόρφωσης το πλήθος σφαλμάτων της μετάδοσης προκύπτει από έλεγχο κάθε bit για το μεταδιδόμενο και το λαμβανόμενο σήμα.

```
%   Question st - BER Calculation

   BER_file=0;
```

```
    for i=1:1:length(x_bin_stream)
        if (x_bin_stream(i)~=x_bin_stream_rx(i))
            BER_file=BER_file+1;
        end;
    end;
    BER_file=BER_file/length(x_bin_stream);

    for i=1:1:N
        st_struct_rx(i)=bi2de(x_bin_stream_rx(1+(i-
1)*sym_bits:sym_bits+(i-1)*sym_bits),'left-msb');
    end;

    disp('--------------------------------------');
    disp(' Used SNR for file Tx ');
    disp(SNR);
    disp(' Achieved BER (%) ');
    disp(100*BER_file);
```

Τα παραγόμενα BER για τις δύο συνθήκες μετάδοσης:

```
--------------------------------------
 Used SNR for file Tx
    14

 Achieved BER (%)
    0
--------------------------------------
 Used SNR for file Tx
    4

 Achieved BER (%)
    1.2562
```

Οπότε προκύπτει ότι το σύστημα μετάδοσης επιτυγχάνει λίγο χειρότερα από το θεωρητικά αναμενόμενα. Αυτό οφείλεται εν μέρει στη διαδικασία κβαντισμού. Για SNR=14db υπάρχει μηδενικός και απόλυτη θεωρητική και πειραματική συμφωνία.

{Question z}

Για την ανακατασκευή του αρχείου ήχου λήφθηκε υπόψη η απαίτηση για unsigned και 8 bit παράσταση κάθε δείγματος. Ο λαμβανόμενος ήχος από-κανονικοποιήθηκε. Το παραγόμενο αποτέλεσμα θετικοποιεί τα παραγόμενα δείγματα οπότε είναι απρόσημα, δεδομένου ότι δεν προσθέτει το αρνητικό offset Amin. Για την αποθήκευση σε αρχείο χρησιμοποιήθηκε η συνάρτηση audiowrite().

```
% Question z - Write Rx Wav to file

   out_values=(Amax-Amin)*st_struct_rx/255;

   if (mod(AM_sum,2)==0)
      filename_out='soundfile2_lab2_rx_14db.wav';
   else
      filename_out='soundfile1_lab2_rx_14db.wav';
   end;

   figure(4)
   i=1:1:length(out_values);
   semilogy((i-1)/fs,out_values,'LineWidth',1);
   grid;
   xlabel(' Time (sec) ');
   ylabel(' Signal Value ');
   title(' Rx Wav Signal Values');

   disp('---------------------------------------');
   disp(' Original Wav file Info ');
   disp('---------------------------------------');
   audioinfo(filename)

   audiowrite(filename_out,out_values,fs,'BitsPerSample',8);
   disp('---------------------------------------');
   disp(' Rx Wav file Store Info ');
   disp('---------------------------------------');
   audioinfo(filename_out)

   sound(out_values,fs);
```

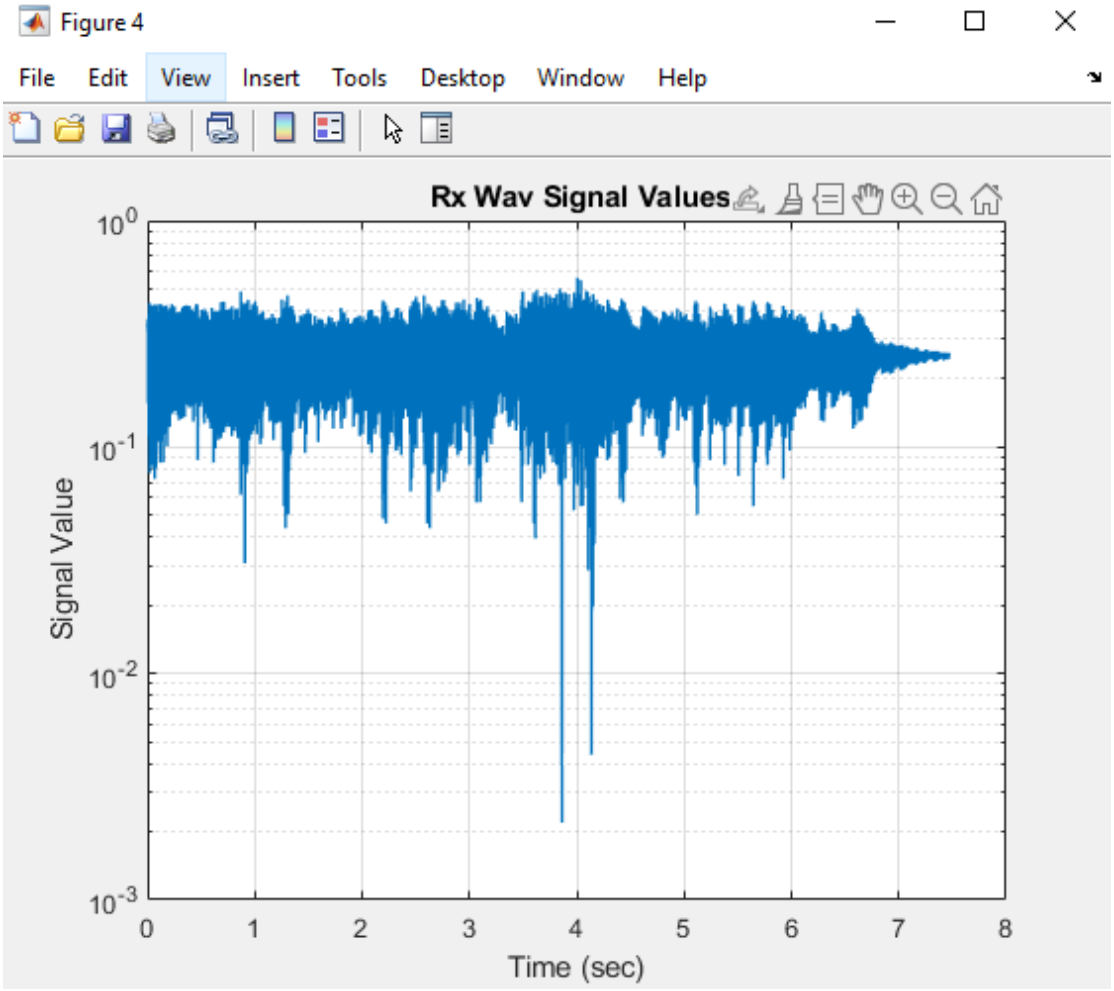
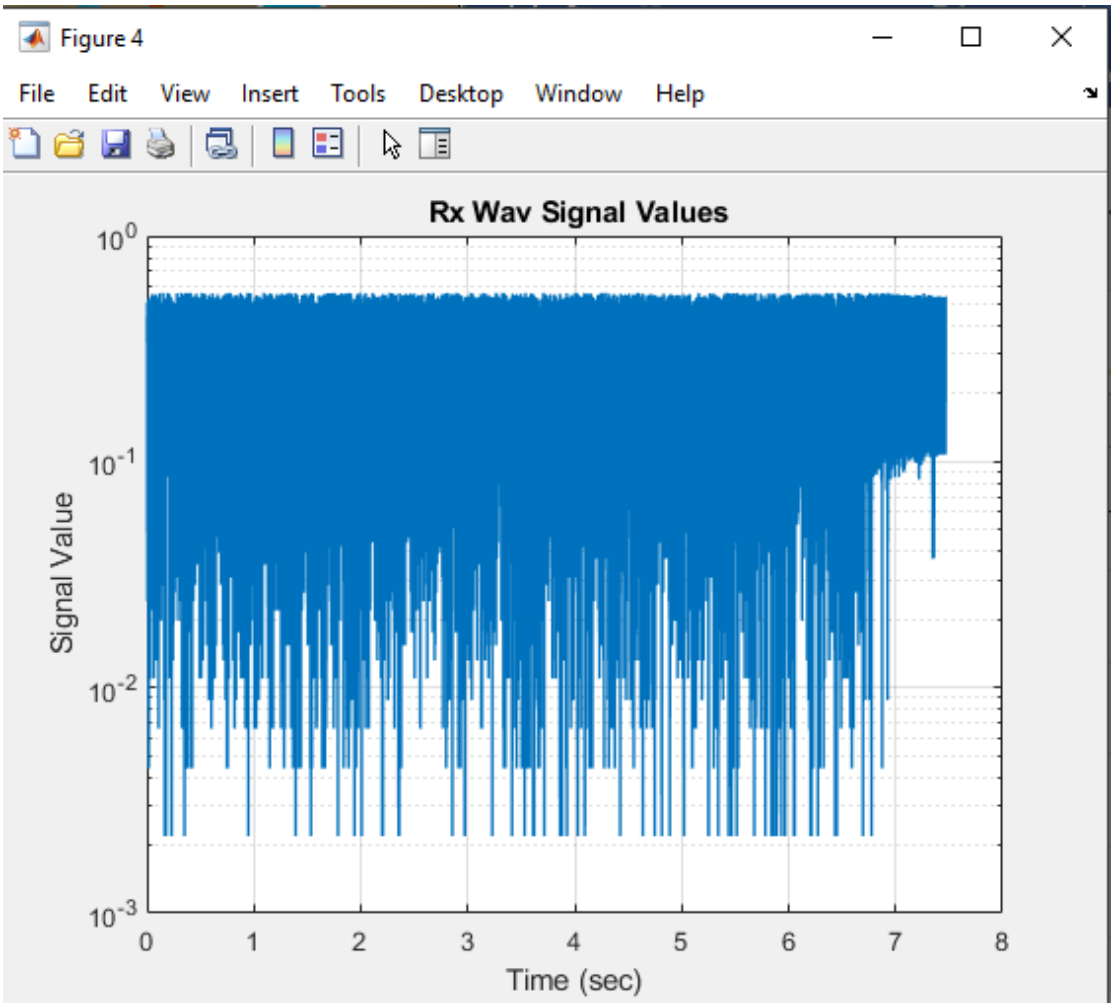


Figure 4
File Edit View Insert Tools Desktop Window Help
Rx Wav Signal Values
Signal Value
Time (sec)
10^0
10^-1
10^-2
10^-3
0 1 2 3 4 5 6 7 8



Rx Wav Τιμές Σήματος για SNR=14 και 4 db.

Έγινε χρήση της παραμέτρου ‘BitsPerSample’ η οποία τέθηκε σε τιμή 8 για την παράσταση των δειγμάτων με χρήση 8 bits.

Rx Wav file Store Info

---

ans =

struct with fields:

```
            Filename:
'C:\Users\viken\OneDrive\Desktop\soundfile1_lab2_rx_14db.wav'
    CompressionMethod: 'Uncompressed'
         NumChannels: 1
          SampleRate: 44100
        TotalSamples: 329413
            Duration: 7.4697
               Title: []
             Comment: []
              Artist: []
        BitsPerSample: 8
```

Αναπαράγεται το δοθέν αρχείο μέσω της συνάρτησης sound(), το αρχείο που αναλογεί στο ισχυρό SNR είναι πιο ακριβές σε σχέση με το χαμηλότερο SNR. Η απώλεια πληροφορίας οφείλεται στο κυρίως θόρυβο και στην διαδικασία κβαντοποίησης.

Βιβλιογραφία

[1] Σημειώσεις Μαθήματος Κωττής, Παναγόπουλος

[2] Matlab Mathworks Tutorial © 2020

[3] Gray Encoding Wikipedia © 2020