# ΤΗΛ 301 ΑΣΚΗΣΗ 3

Μιχαήλ Κρατημένος ΑΜ: 2018030104
Ιωάννης Λαμπρινίδης ΑΜ: 2018030075

23 Δεκεμβρίου 2020

## 1,2

Αφού δημιουργήθηκε η δυαδική ακολουθία bit seq με στοιχεία $3N$ ισοπίθανα bits, όπου $N = 100$ ως εξής:

```
%% Question 1
N=100;
bit_seq=(sign(randn(3*N, 1)) + 1)/2;
```

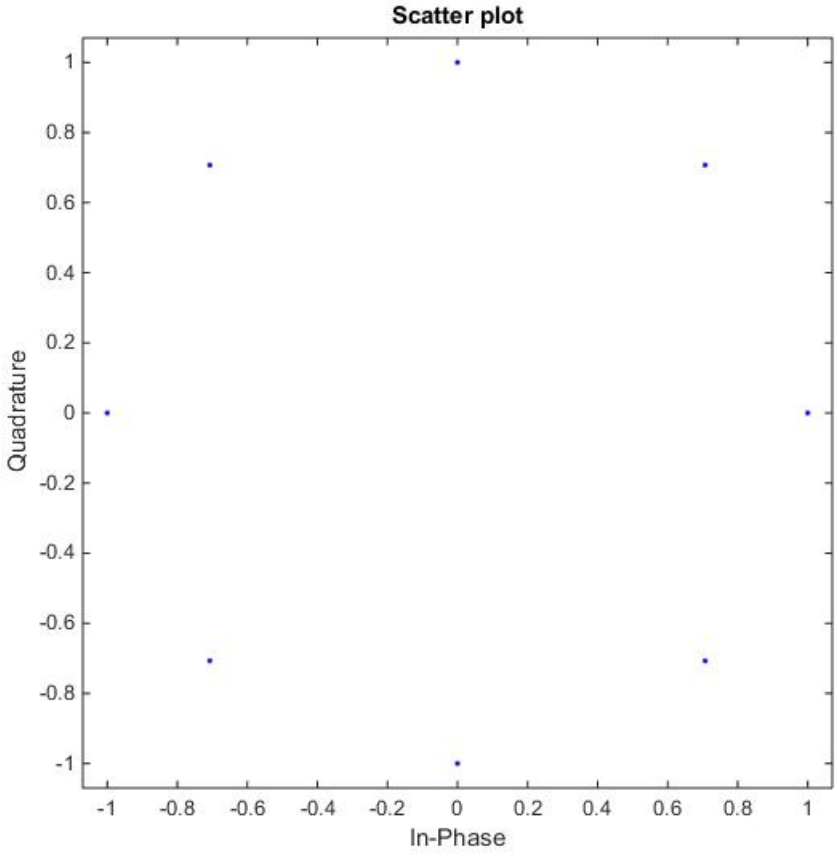
ύστερα γράφτηκε η συνάρτηση bits to PSK 8(bit seq) με χρήση κωδικοποίησης gray ως εξής:

```
function X=bits_to_PSK_8(bit_seq)

N=length(bit_seq)/3;
X=zeros(N,2);

for k = 0:3:length(bit_seq)-1
    i=k/3+1;
    if((bit_seq(k+1)==0) && (bit_seq(k+2)==0) && bit_seq(k+3)==0)
    X(i,1)=cos(0);
    X(i,2)=sin(0);
    elseif((bit_seq(k+1)==0) && (bit_seq(k+2)==0) && ...
        bit_seq(k+3)==1)
    X(i,1)=cos(2*pi*1/8);
    X(i,2)=sin(2*pi*1/8);
     elseif((bit_seq(k+1)==0) && (bit_seq(k+2)==1) && ...
         bit_seq(k+3)==1)
    X(i,1)=cos(2*pi*2/8);
    X(i,2)=sin(2*pi*2/8);
     elseif((bit_seq(k+1)==0) && (bit_seq(k+2)==1) && ...
         bit_seq(k+3)==0)
    X(i,1)=cos(2*pi*3/8);
    X(i,2)=sin(2*pi*3/8);
     elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==1) && ...
         bit_seq(k+3)==0)
    X(i,1)=cos(2*pi*4/8);
```

```
22      X(i,2)=sin(2*pi*4/8);
23       elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==1) && ...
             bit_seq(k+3)==1)
24      X(i,1)=cos(2*pi*5/8);
25      X(i,2)=sin(2*pi*5/8);
26       elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==0) && ...
             bit_seq(k+3)==1)
27      X(i,1)=cos(2*pi*6/8);
28      X(i,2)=sin(2*pi*6/8) ;
29       elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==0) && ...
             bit_seq(k+3)==0)
30      X(i,1)=cos(2*pi*7/8);
31      X(i,2)=sin(2*pi*7/8) ;
32      end
33  end
34
35  end
```

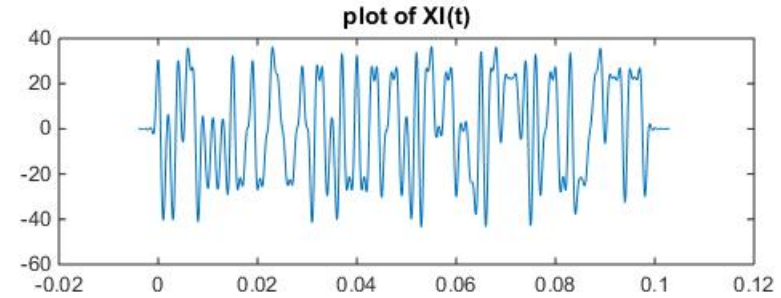
ενώ η απεικονίση της δυαδικής ακολουθίας εισόδου bit seq σε ακολουθία 8-PSK συμβόλων $X$, μήκους $N$, με στοιχεία τα διδιάστατα διανύσματα που παίρνουν τιμές από το αλφάβητο 8-PSK ως εξής:

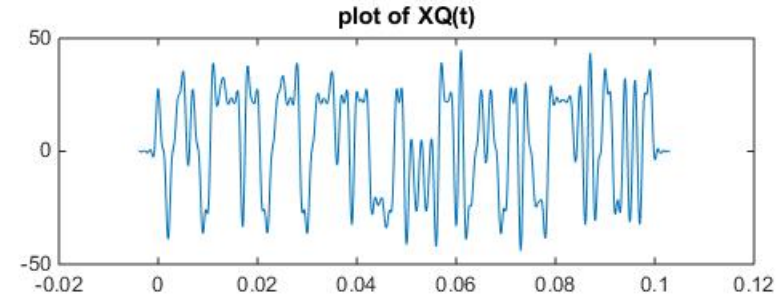
```
1  %% Question 2
2  X=bits_to_PSK_8(bit_seq);
3  scatterplot(X(:,1)+1i*X(:,2));
```

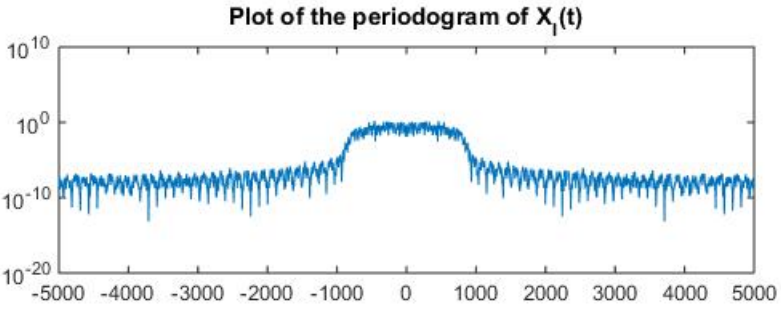


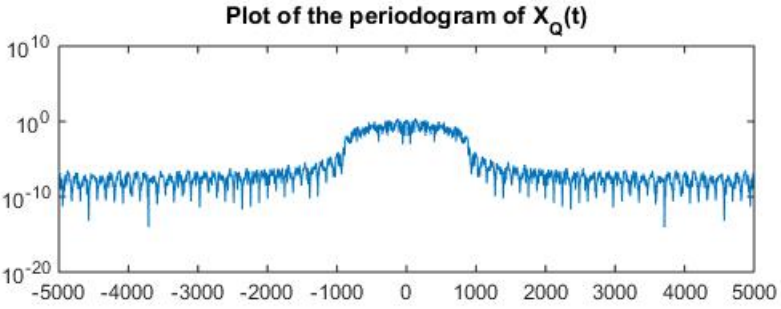

# 3

Αφού περάστηκαν οι ακολουθίες $Xi, n$ και $XQ, n$ από τα SRRC φίλτρα μορφοποίησης και υποθέτοντας, ενδεικτικά, περίοδο συμβόλου $T = 0.001$ sec, $over = 10$, $Ts = \frac{T}{over}$ σχεδιάστηκαν οι κυματομορφές εξόδου, καθώς και τα περιοδογράμματά τους.


plot of XI(t)
40
20
0
-20
-40
-60
-0.02
0
0.02
0.04
0.06
0.08
0.1
0.12


plot of XQ(t)
50
0
-50
-0.02
0
0.02
0.04
0.06
0.08
0.1
0.12

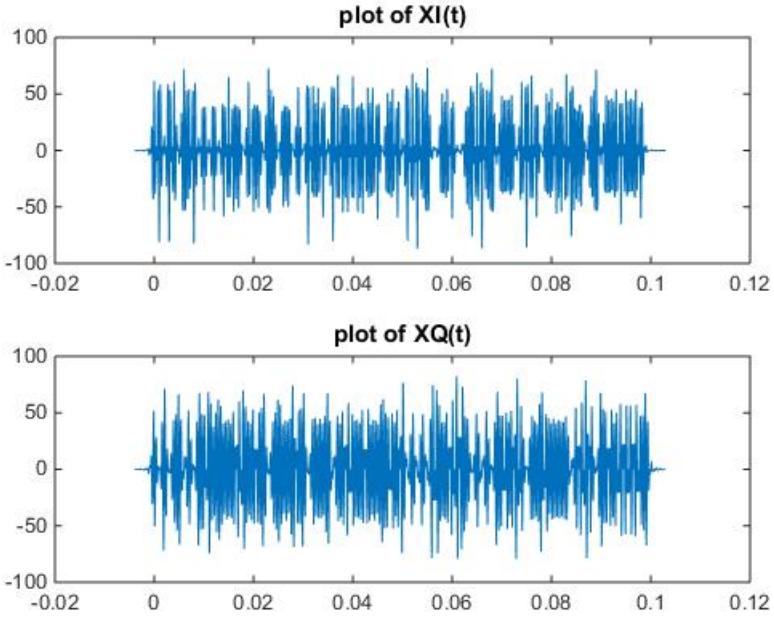

Plot of the periodogram of X_I(t)
10^10
10^0
10^-10
10^-20
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

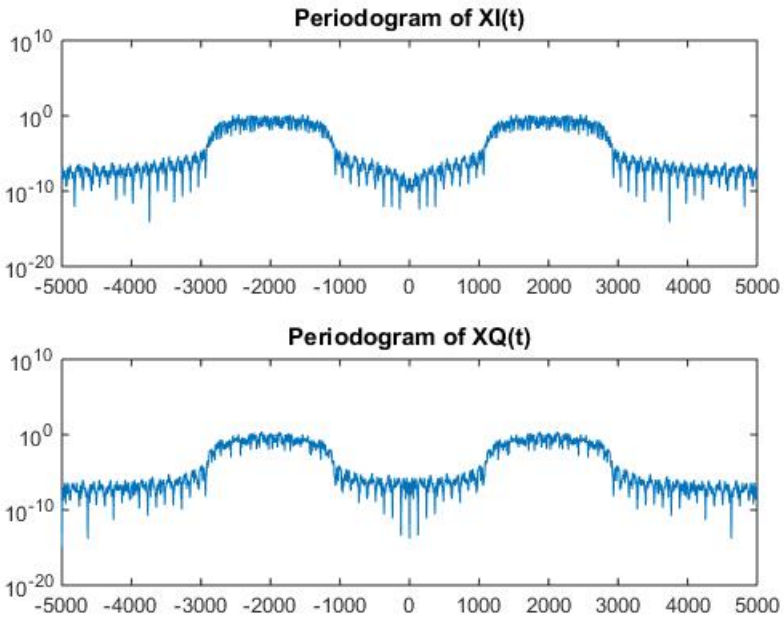

Plot of the periodogram of X_Q(t)
10^10
10^0
10^-10
10^-20
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

```
%% Question 3
T=0.001;
A=4;
over=10;
a=0.8;
over=10;
Ts=T/over;
Nf=2048;
Fs=1/Ts;
Faxis=-Fs/2:Fs/Nf:Fs/2-Fs/Nf;
[f,t]=srrc_pulse(T,over,A,a);
t2=t;
f_all_temp=zeros(length(X),length(t)+(length(X)-1)*over);
 %creating the time moved signals
 for i=0:length(X)-1
     for j=1:length(t)
         f_all_temp(i+1,j+i*over)=X(i+1,1).*f(j);
     end
 end
  %calculating the time of the signal
t1=t(end)+Ts:Ts:t(end)+T*(length(X)-1);
 t=[t t1];
 X_I=sum(f_all_temp, 1);
figure()
subplot(2,1,1)
plot(t,X_I)
title('plot of XI(t)')

hold on;


f_all_temp=zeros(length(X),length(t2)+(length(X)-1)*over);
 %creating the time moved signals
 for i=0:length(X)-1
     for j=1:length(t2)
         f_all_temp(i+1,j+i*over)=X(i+1,2)*f(j);
     end
 end
  %calculating the time of the signal
X_Q=sum(f_all_temp, 1);
subplot(2,1,2)
plot(t,X_Q)
title('plot of XQ(t)')


figure()
Px=fftshift(abs(fft(X_I,Nf)).^2)*Ts./length(t);
subplot(2,1,1)
semilogy(Faxis,Px)
title('Plot of the periodogram of X_I(t)')
subplot(2,1,2)
Px=fftshift(abs(fft(X_Q,Nf)).^2)*Ts./length(t);
semilogy(Faxis,Px)
title('Plot of the periodogram of X_Q(t)')
```

# 4

Ύστερα πολλαπλασιάζονται τα σήματα με συνημιτονοειδείς συναρτήσεις με $F = 2kHz$ και σχεδιάστηκαν οι κυματομορφές των $X_I$, $X_Q$ καθώς και τα περιοδογράμματά τους ως εξής:



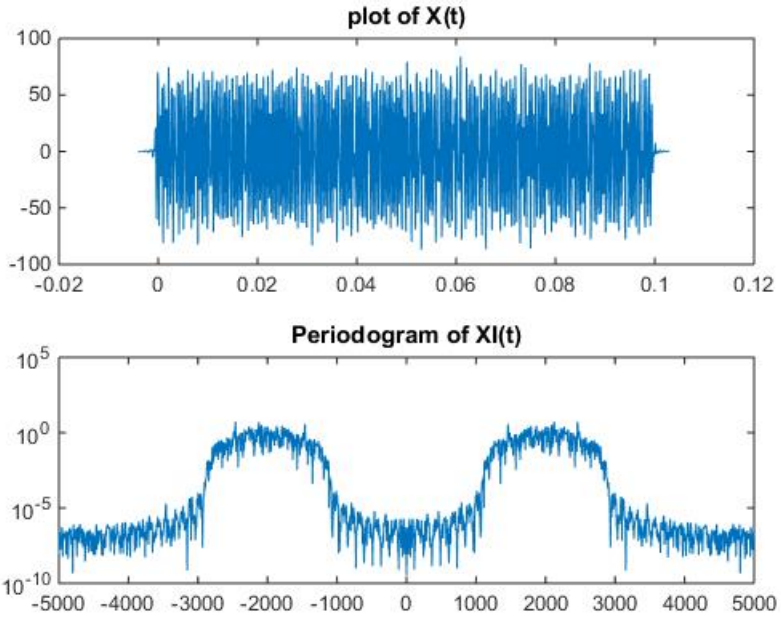



Παρατηρείται ότι λόγω του πολλαπλασιασμού με τις συναρτήσεις αυτές, τα σήματα μεταφέρονται στη ζώνη διέλευσης, για να μεταδοθούν επιτυχώς μέσω του καναλιού που πρόκειται να εισαχθούν.
Ακολουθεί και το αντίστοιχο νήμα κώδικα:

```
%% Question 4
F0=2000;

X_I_new=(X_I.*2.*cos(2.*pi.*F0.*t));
X_Q_new=(X_Q.*(-2*sin(2.*pi.*F0.*t)));
figure()
subplot(2,1,1)
plot(t,X_I_new)
title('plot of XI(t)')

subplot(2,1,2)
plot(t,X_Q_new)
title('plot of XQ(t)')

figure()
Px=fftshift(abs(fft(X_I_new,Nf)).^2)*Ts./length(t);
subplot(2,1,1)
semilogy(Faxis,Px)
title('Periodogram of XI(t)')

subplot(2,1,2)
Px=fftshift(abs(fft(X_Q_new,Nf)).^2)*Ts./length(t);
semilogy(Faxis,Px)
title('Periodogram of XQ(t)')
```

# 5

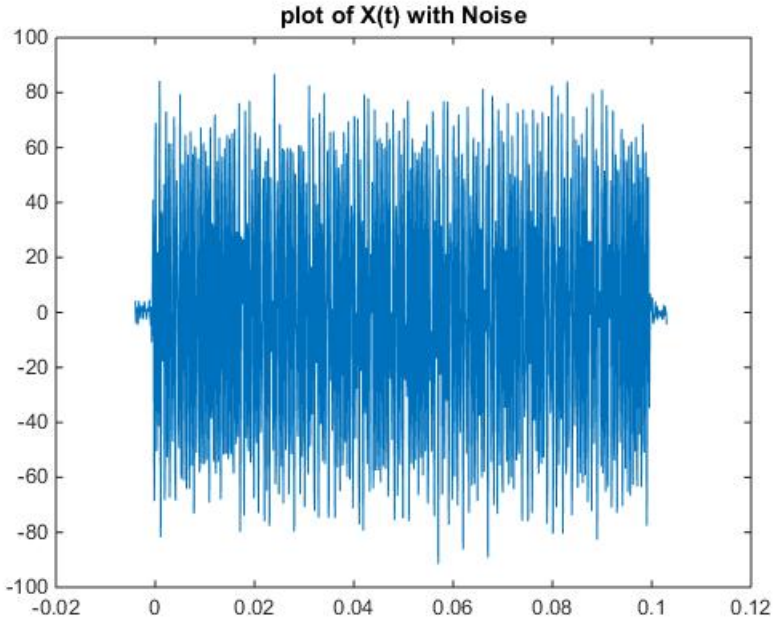
Αφού γίνει η πρόσθεση, λαμβάνεται το σήμα που είναι η είσοδος του καναλιού. Παρατηρείται ότι στο πεδίο της συχνότητας το σήμα θα μεταδοθεί γύρω από συχνότητα καναλιού 2 kHz




```
%% Question 5

Xt=X_I_new+X_Q_new;
figure()
subplot(2,1,1)
plot(t, Xt)
title('plot of X(t)')

Px=fftshift(abs(fft(Xt,Nf)).^2)*Ts./length(t);
subplot(2,1,2)
semilogy(Faxis,Px)
title('Periodogram of XI(t)')
```

# 6-7

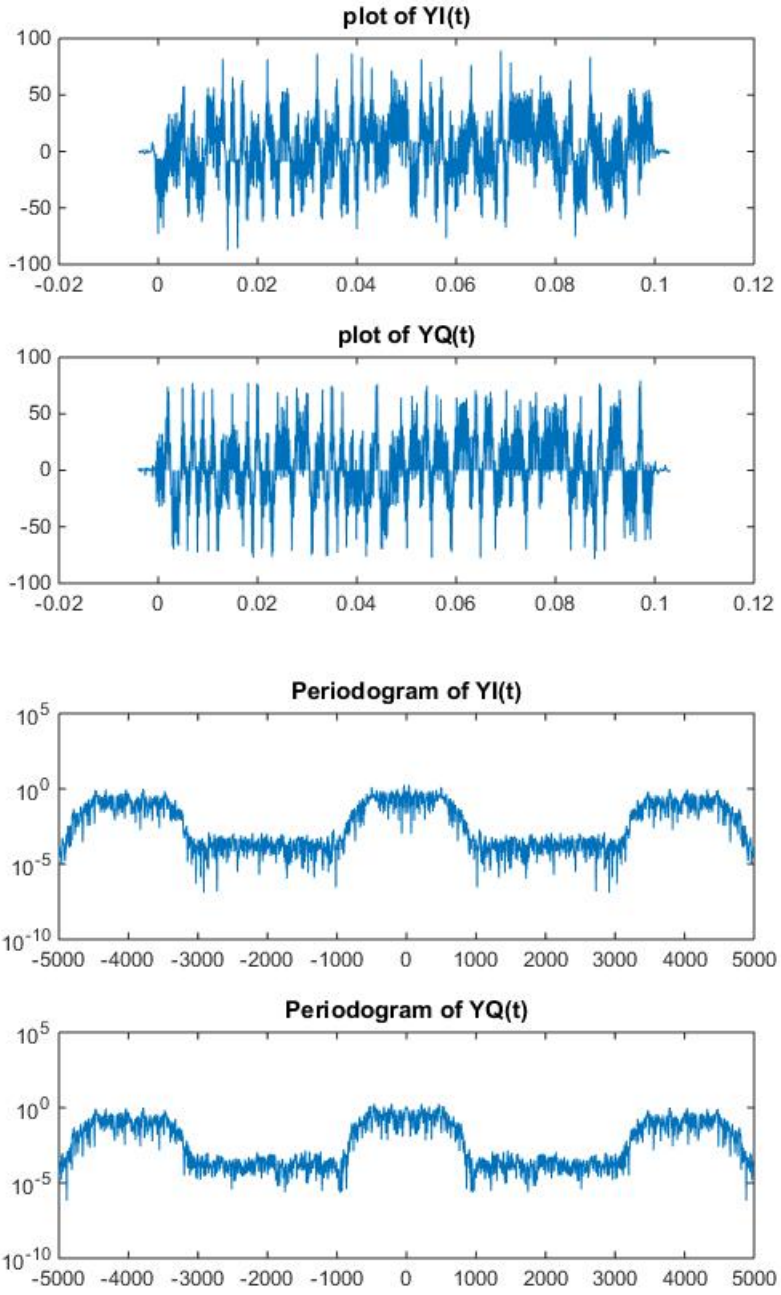
Αφού θεωρηθεί το κανάλι ιδανικό, γίνεται πρόσθεση λευκού Gaussian θορύβου $W(t)$ στο $X(t)$, όπου ακολουθεί κανονική κατανομή με διασπορά εξαρτόμενη από το $SNR_{dB}$, όπως φαίνεται από τον τύπο της. Παρακάτω φαίνεται το σχήμα που προκύπτει καθώς και το αντίστοιχο νήμα κώδικα:




```
%% Question 7

SNRdb=10;
varw=(1/Ts.*10^(SNRdb/10));
varN=Ts*varw/2;
Gaussian_Noise=sqrt(varN).*randn(1,length(Xt));

Yt=Xt+Gaussian_Noise;
figure()
plot(t,Yt)
title('plot of X(t) with Noise')
```

# 8

Αφού πολλαπλασιάστηκε η ενθόρυβη κυματομορφή $Y(t)$ στο δέκτη με τους κατάλληλους φορείς, σχεδιάστηκαν οι κυματομορφές που προκύπτουν και τα περιοδογράμματά τους ως εξής:

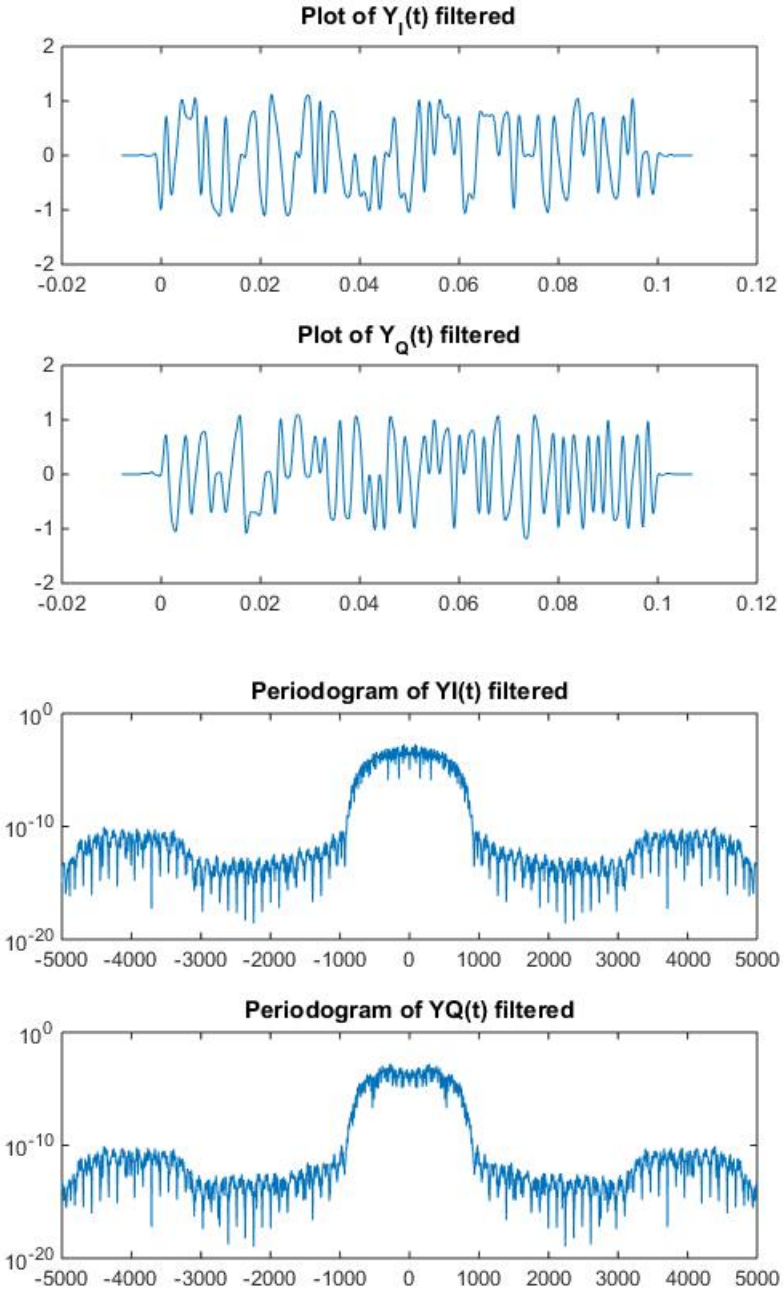



Παρατηρείται ότι έχει ανακτηθεί η πληροφορία, αλλά έχουν μείνει κάποια όροι έξω από το πεδίο των συχνοτήτων των διαμορφωτών (για $f < 2kHz$ ή $f > 2kHz$).

```
%% Question 8

Y_I_rec = Yt.*cos(2*pi*F0*t);
Y_Q_rec = -Yt.*sin(2*pi*F0*t);

figure()
subplot(2,1,1)
plot(t, Y_I_rec)
title('plot of YI(t)')

subplot(2,1,2)
plot(t,Y_Q_rec)
title('plot of YQ(t)')


Px=fftshift(abs(fft(Y_I_rec,Nf)).^2)*Ts./length(t);
figure()
subplot(2,1,1)
semilogy(Faxis, Px)
title('Periodogram of YI(t)')

Px=fftshift(abs(fft(Y_Q_rec,Nf)).^2)*Ts./length(t);
subplot(2,1,2)
semilogy(Faxis, Px)
title('Periodogram of YQ(t)')
```

# 9

Αφού περάστηκαν οι προηγούμενες κυματομορφές από τα προσαρμοσμένα φίλτρα, σχεδιάστηκαν οι κυματομορφές που προέκυψαν καθώς και τα περιοδογράμματά τους ως εξής:

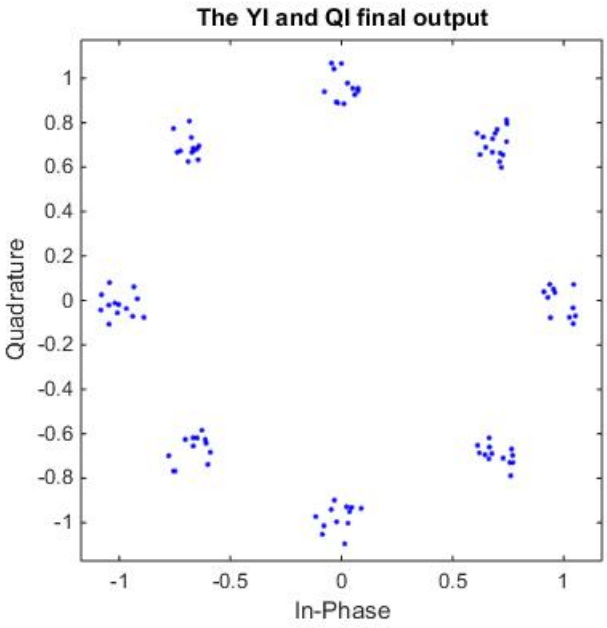



Παρατηρείται ότι με την επίδραση των προσαρμοσμένων φίλτρων τα όροι έξω από το πεδίο των συχνοτήτων των διαμορφωτών περιορίστηκαν σημαντικά έτσι ώστε να μην λαμβάνονται υπόψιν.

```
%% Question 9

Y_I_filtered=conv(f,Y_I_rec)*Ts;
Y_Q_filtered=conv(f,Y_Q_rec)*Ts;

t_conv=t(1)+t2(1):Ts:t(end)+t2(end);
figure()
subplot(2,1,1)
plot(t_conv,Y_I_filtered)
title('Plot of Y_I(t) filtered')

subplot(2,1,2)
plot(t_conv,Y_Q_filtered)
title('Plot of Y_Q(t) filtered')

Px=fftshift(abs(fft(Y_I_filtered,Nf)).^2)*Ts./length(t);
figure()
subplot(2,1,1)
semilogy(Faxis, Px)
title('Periodogram of YI(t) filtered')

Px=fftshift(abs(fft(Y_Q_filtered,Nf)).^2)*Ts./length(t);
subplot(2,1,2)
semilogy(Faxis, Px)
title('Periodogram of YQ(t) filtered')
```

# 10

Αφού δειγματοληπτήθηκε η έξοδος των προσαρμοσμένων φίλτρων στις κατάλληλες χρονικές στιγμές, σχεδιάστηκε η ακολουθία εξόδου $Y$ χρησιμοποιώντας την εντολή scatterplot όπου παρατηρείται και η παρουσία του λευκού Gaussian θορύβου:




```
%% Question 10

j=0;
while t_conv(j+1)<0
      j=j+1;
end
 t_temp=zeros(1,N);
 Y_I_final=zeros(1,N);
 Y_Q_final=zeros(1,N);

for i=1:N
    t_temp(i)=t_conv(j+(i-1)*over);
    Y_I_final(i)=Y_I_filtered(j+(i-1)*over);
    Y_Q_final(i)=Y_Q_filtered(j+(i-1)*over);
end

scatterplot(Y_I_final+1i*Y_Q_final);
title('The YI and QI final output')
```

# 11

Αποκωδικοποιήθηκε η ακολουθία με τον κανόνα του εγγύτερου γείτονα, δηλαδή όποια τιμή ήταν κοντά σε τιμή συμβόλου που στείλαμε και ταυτόχρονα χρησιμοποιήθηκε η αντίστροφη απεικόνιση Gray, για να υπολογιστεί η εκτιμώμενη δυαδική ακολουθία εισόδου

```
function [est_X,est_bit_seq]=detect_PSK_8(Y)

est_X=zeros(length(Y),1);
est_bit_seq=zeros(3*length(Y),1);
for k=1:length(Y)

    if(Y(k,1)>(cosd(337.5)))%0 degrees
        est_X(k,1)=1;
        est_X(k,2)=0;
        est_bit_seq(k*3-2)=0;
        est_bit_seq(k*3-1)=0;
        est_bit_seq(k*3)=0;

    elseif(cosd(67.5)<Y(k,1)&&Y(k,1)<cosd(22.5)&&Y(k,2)>0)%45 ...
        degrees
        est_X(k,1)=1/sqrt(2);
        est_X(k,2)=1/sqrt(2);
        est_bit_seq(k*3-2)=0;
        est_bit_seq(k*3-1)=0;
        est_bit_seq(k*3)=1;

    elseif(Y(k,2)>sind(67.5))%90 degrees
        est_X(k,1)=0;
        est_X(k,2)=1;
        est_bit_seq(k*3-2)=0;
        est_bit_seq(k*3-1)=1;
        est_bit_seq(k*3)=1;



    elseif(cosd(112.5)>Y(k,1)&& Y(k,1)>cosd(157.5)&&Y(k,2)>0)%135 ...
        degrees
        est_X(k,1)=-1/sqrt(2);
        est_X(k,2)=1/sqrt(2);
        est_bit_seq(k*3-2)=0;
        est_bit_seq(k*3-1)=1;
        est_bit_seq(k*3)=0;



    elseif(Y(k,1)<cosd(157.5))%180 degrees
        est_X(k,1)=-1;
        est_X(k,2)=0;
        est_bit_seq(k*3-2)=1;
        est_bit_seq(k*3-1)=1;
        est_bit_seq(k*3)=0;


    elseif(cosd(202.5)<Y(k,1)&& Y(k,1)<cosd(247.5) && ...
        Y(k,2)<0)%225 degrees
        est_X(k,1)=-1/sqrt(2);
        est_X(k,2)=-1/sqrt(2);
        est_bit_seq(k*3-2)=1;
        est_bit_seq(k*3-1)=1;
        est_bit_seq(k*3)=1;

```

```
55
56      elseif(Y(k,2)<sind(247.5))%270 degrees
57          est_X(k,1)=0;
58          est_X(k,2)=-1;
59          est_bit_seq(k*3-2)=1;
60          est_bit_seq(k*3-1)=0;
61          est_bit_seq(k*3)=1;
62
63
64
65          %est_X(k,1)>cos(292.5)&&est_X(k,1)<cos(337,5)&&est_X(k,2)<0
66      elseif(Y(k,1)>cosd(292.5) && Y(k,1)<cosd(337.5) && Y(k,2)<0)
67          est_X(k,1)=1/sqrt(2);
68          est_X(k,2)=-1/sqrt(2);
69          est_bit_seq(k*3-2)=1;
70          est_bit_seq(k*3-1)=0;
71          est_bit_seq(k*3)=0;
72      end
73  end
```

```
1  %% Question 11
2
3  Y_Final(:,1)=Y_I_final(1:N);
4  Y_Final(:,2)=Y_Q_final(1:N);
5  [est_X,est_bit_seq]=detect_PSK_8(Y_Final);
```

# 12

Αφού συγκρίθηκαν τα 2 διανύσματα (εισόδου και εξόδου) υπολογίστηκε το πλήθος των σφαλμάτων εκτίμησης συμβόλου

```
1  function num_of_symbol_errors=symbol_errors(est_X,X)
2  num_of_symbol_errors=0;
3  for i=1:length(X)
4      if(abs(X(i,1)-est_X(i,1))>0.001)
5          num_of_symbol_errors=num_of_symbol_errors+1;
6      end
7  end
8  for i=1:length(X)
9      if(abs(X(i,2)-est_X(i,2))>0.001)
10         num_of_symbol_errors=num_of_symbol_errors+1;
11     end
12 end
13
14 num_of_symbol_errors=ceil(num_of_symbol_errors/2);
15
16
17 end
```

## 13

Αφού συγκρίθηκαν τα 2 διανύσματα (εισόδου και εξόδου) υπολογίστηκε το πλήθος των σφαλμάτων εκτίμησης bit.

```
function num_of_bit_errors = bit_errors(est_bit_seq,b)
num_of_bit_errors=0;
for i=1:length(b)
    if(abs(b(i)-est_bit_seq(i))>0.001)
        num_of_bit_errors=num_of_bit_errors+1;
    end
end

end
```

# Β Μέρος

Δεν έβγαινε η εκτιμώμενη πιθανότητα σφάλματος συμβόλου σωστά παρόλο που η μέθοδος monte carlo φαινομενικά έχει εφαρμοστεί ορθά και δεν διακρίθηκε κάποιο λάθος από πλευράς μας.

```
%% B meros

SNR=[-2:2:16];

k=200;
Esymbol=zeros(length(SNR),1);
Ebit=zeros(length(SNR),1);


for p=1:10
       sum1=0;
       sum2=0;
   for  u=1:k
    b=(sign(randn(3*N, 1)) + 1)/2;

    X=bits_to_PSK_8(b);
    [f,t]=srrc_pulse(T,over,A,a);
    t2=t;
    f_all_temp=zeros(length(X),length(t)+(length(X)-1)*over);
    %creating the time moved signals
     for i=0:length(X)-1
         for j=1:length(t)
             f_all_temp(i+1,j+i*over)=X(i+1,1).*f(j);
         end
     end
    %calculating the time of the signal
    t1=t(end)+Ts:Ts:t(end)+T*(length(X)-1);
    t=[t t1];
    X_I=sum(f_all_temp, 1);

    f_all_temp=zeros(length(X),length(t2)+(length(X)-1)*over);
```

```
32      %creating the time moved signals
33       for i=0:length(X)-1
34            for j=1:length(t2)
35               f_all_temp(i+1,j+i*over)=X(i+1,2)*f(j);
36            end
37       end
38           %calculating the time of the signal
39      X_Q=sum(f_all_temp, 1);
40
41
42      X_I_new=(X_I.*2.*cos(2.*pi.*F0.*t));
43      X_Q_new=(X_Q.*(-2*sin(2.*pi.*F0.*t)));
44
45      Xt=X_I_new+X_Q_new;
46
47      SNRdb=SNR(p);
48
49
50      varw=(1/Ts.*10^(SNRdb/10));
51      varN=Ts*varw/2;
52      Gaussian_Noise=sqrt(varN).*randn(1,length(Xt));
53      Yt=Xt+Gaussian_Noise;
54
55      Y_I_rec = Yt.*cos(2*pi*F0*t);
56      Y_Q_rec = -Yt.*sin(2*pi*F0*t);
57
58      Y_I_filtered=conv(f,Y_I_rec)*Ts;
59      Y_Q_filtered=conv(f,Y_Q_rec)*Ts;
60
61      t_conv=t(1)+t2(1):Ts:t(end)+t2(end);
62
63      j=0;
64      while t_conv(j+1)<0
65               j=j+1;
66      end
67      t_temp=zeros(1,N);
68      Y_I_final=zeros(1,N);
69      Y_Q_final=zeros(1,N);
70
71
72      for i=1:N
73       t_temp(i)=t_conv(j+(i-1)*over);
74       Y_I_final(i)=Y_I_filtered(j+(i-1)*over);
75       Y_Q_final(i)=Y_Q_filtered(j+(i-1)*over);
76      end
77      Y_Final(:,1)=Y_I_final(1:N);
78      Y_Final(:,2)=Y_Q_final(1:N);
79      [est_X,est_bit_seq]=detect_PSK_8(Y_Final);
80
81      sum1=sum1+symbol_errors(est_X,X);
82
83      sum2=sum2+bit_errors(est_bit_seq,b);
84
85
86     end
87      Esymbol(p)=sum1/(N*k);
88      Ebit(p)=sum2/(N*3*k);
```

```
89  end
90
91  temp=Q(SNR);
92  figure()
93  semilogy(SNR, Esymbol);
94  hold on
95  semilogy(SNR, temp);
```

# Νήματα κώδικα που χρησιμοποιήθηκαν

## TEL301 3.m

```
1   clear all;
2   close all;
3   %% Question 1
4   N=100;
5   T=0.001;
6   A=4;
7   over=10;
8   a=0.8;
9   over=10;
10  Ts=T/over;
11  Nf=2048;
12  Fs=1/Ts;
13  Faxis=-Fs/2:Fs/Nf:Fs/2-Fs/Nf;
14
15  b=(sign(randn(3*N, 1)) + 1)/2;
16  % T=0.001;
17
18  %% Question 2
19  X=bits_to_PSK_8(b);
20  scatterplot(X(:,1)+1i*X(:,2));
21  title('scatterPlot of the PSK')
22  length(X)
23  %% Question 3
24
25  [f,t]=srrc_pulse(T,over,A,a);
26  t2=t;
27  f_all_temp=zeros(length(X),length(t)+(length(X)-1)*over);
28   %creating the time moved signals
29   for i=0:length(X)-1
30       for j=1:length(t)
31           f_all_temp(i+1,j+i*over)=X(i+1,1).*f(j);
32       end
33   end
34    %calculating the time of the signal
35  t1=t(end)+Ts:Ts:t(end)+T*(length(X)-1);
36   t=[t t1];
37   X_I=sum(f_all_temp, 1);
38  figure()
39  subplot(2,1,1)
40  plot(t,X_I)
41  title('plot of XI(t)')
```

```
42
43 hold on;
44
45
46 f_all_temp=zeros(length(X),length(t2)+(length(X)-1)*over);
47  %creating the time moved signals
48  for i=0:length(X)-1
49      for j=1:length(t2)
50          f_all_temp(i+1,j+i*over)=X(i+1,2)*f(j);
51      end
52  end
53   %calculating the time of the signal
54 X_Q=sum(f_all_temp, 1);
55 subplot(2,1,2)
56 plot(t,X_Q)
57 title('plot of XQ(t)')
58
59
60 figure()
61 Px=fftshift(abs(fft(X_I,Nf)).^2)*Ts./length(t);
62 subplot(2,1,1)
63 semilogy(Faxis,Px)
64 title('Plot of the periodogram of X_I(t)')
65 subplot(2,1,2)
66 Px=fftshift(abs(fft(X_Q,Nf)).^2)*Ts./length(t);
67 semilogy(Faxis,Px)
68 title('Plot of the periodogram of X_Q(t)')
69
70
71 %% Question 4
72 F0=2000;
73
74 X_I_new=(X_I.*2.*cos(2.*pi.*F0.*t));
75 X_Q_new=(X_Q.*(-2*sin(2.*pi.*F0.*t)));
76 figure()
77 subplot(2,1,1)
78 plot(t,X_I_new)
79 title('plot of XI(t)')
80
81 subplot(2,1,2)
82 plot(t,X_Q_new)
83 title('plot of XQ(t)')
84
85 figure()
86 Px=fftshift(abs(fft(X_I_new,Nf)).^2)*Ts./length(t);
87 subplot(2,1,1)
88 semilogy(Faxis,Px)
89 title('Periodogram of XI(t)')
90
91 subplot(2,1,2)
92 Px=fftshift(abs(fft(X_Q_new,Nf)).^2)*Ts./length(t);
93 semilogy(Faxis,Px)
94 title('Periodogram of XQ(t)')
95
96 %% Question 5
97
98 Xt=X_I_new+X_Q_new;
```

```
figure()
subplot(2,1,1)
plot(t, Xt)
title('plot of X(t)')

Px=fftshift(abs(fft(Xt,Nf)).^2)*Ts./length(t);
subplot(2,1,2)
semilogy(Faxis,Px)
title('Periodogram of XI(t)')

%% Question 7

SNRdb=10;
varw=(1/Ts.*10^(SNRdb/10));
varN=Ts*varw/2;
Gaussian_Noise=sqrt(varN).*randn(1,length(Xt));

Yt=Xt+Gaussian_Noise;
figure()
plot(t,Yt)
title('plot of X(t) with Noise')

%% Question 8

Y_I_rec = Yt.*cos(2*pi*F0*t);
Y_Q_rec = -Yt.*sin(2*pi*F0*t);

figure()
subplot(2,1,1)
plot(t, Y_I_rec)
title('plot of YI(t)')

subplot(2,1,2)
plot(t,Y_Q_rec)
title('plot of YQ(t)')


Px=fftshift(abs(fft(Y_I_rec,Nf)).^2)*Ts./length(t);
figure()
subplot(2,1,1)
semilogy(Faxis, Px)
title('Periodogram of YI(t)')

Px=fftshift(abs(fft(Y_Q_rec,Nf)).^2)*Ts./length(t);
subplot(2,1,2)
semilogy(Faxis, Px)
title('Periodogram of YQ(t)')

%% Question 9

Y_I_filtered=conv(f,Y_I_rec)*Ts;
Y_Q_filtered=conv(f,Y_Q_rec)*Ts;

t_conv=t(1)+t2(1):Ts:t(end)+t2(end);
figure()
subplot(2,1,1)
plot(t_conv,Y_I_filtered)
```

```
title('Plot of Y_I(t) filtered')

subplot(2,1,2)
plot(t_conv,Y_Q_filtered)
title('Plot of Y_Q(t) filtered')

Px=fftshift(abs(fft(Y_I_filtered,Nf)).^2)*Ts./length(t);
figure()
subplot(2,1,1)
semilogy(Faxis, Px)
title('Periodogram of YI(t) filtered')

Px=fftshift(abs(fft(Y_Q_filtered,Nf)).^2)*Ts./length(t);
subplot(2,1,2)
semilogy(Faxis, Px)
title('Periodogram of YQ(t) filtered')

%% Question 10


j=0;
while t_conv(j+1)<0
     j=j+1;
end
 t_temp=zeros(1,N);
 Y_I_final=zeros(1,N);
 Y_Q_final=zeros(1,N);


for i=1:N
    t_temp(i)=t_conv(j+(i-1)*over);
    Y_I_final(i)=Y_I_filtered(j+(i-1)*over);
    Y_Q_final(i)=Y_Q_filtered(j+(i-1)*over);
end


scatterplot(Y_I_final+1i*Y_Q_final);
title('The YI and QI final output')

%% Question 11

Y_Final(:,1)=Y_I_final(1:N);
Y_Final(:,2)=Y_Q_final(1:N);
[est_X,est_bit_seq]=detect_PSK_8(Y_Final);
%% Question 12

symbol_errors(est_X,X)

%% Question 13

bit_errors(est_bit_seq,b)

%% B meros

SNR=[-2:2:16];

k=200;
```

```
Esymbol=zeros(length(SNR),1);
Ebit=zeros(length(SNR),1);


for p=1:10
       sum1=0;
       sum2=0;
   for  u=1:k
    b=(sign(randn(3*N, 1)) + 1)/2;

    X=bits_to_PSK_8(b);
    [f,t]=srrc_pulse(T,over,A,a);
    t2=t;
    f_all_temp=zeros(length(X),length(t)+(length(X)-1)*over);
    %creating the time moved signals
     for i=0:length(X)-1
         for j=1:length(t)
             f_all_temp(i+1,j+i*over)=X(i+1,1).*f(j);
         end
     end
    %calculating the time of the signal
    t1=t(end)+Ts:Ts:t(end)+T*(length(X)-1);
    t=[t t1];
    X_I=sum(f_all_temp, 1);

    f_all_temp=zeros(length(X),length(t2)+(length(X)-1)*over);
    %creating the time moved signals
     for i=0:length(X)-1
          for j=1:length(t2)
             f_all_temp(i+1,j+i*over)=X(i+1,2)*f(j);
          end
     end
         %calculating the time of the signal
    X_Q=sum(f_all_temp, 1);


    X_I_new=(X_I.*2.*cos(2.*pi.*F0.*t));
    X_Q_new=(X_Q.*(-2*sin(2.*pi.*F0.*t)));

    Xt=X_I_new+X_Q_new;

    SNRdb=SNR(p);


    varw=(1/Ts.*10^(SNRdb/10));
    varN=Ts*varw/2;
    Gaussian_Noise=sqrt(varN).*randn(1,length(Xt));
    Yt=Xt+Gaussian_Noise;

    Y_I_rec = Yt.*cos(2*pi*F0*t);
    Y_Q_rec = -Yt.*sin(2*pi*F0*t);

    Y_I_filtered=conv(f,Y_I_rec)*Ts;
    Y_Q_filtered=conv(f,Y_Q_rec)*Ts;

    t_conv=t(1)+t2(1):Ts:t(end)+t2(end);

```

```
270      j=0;
271      while t_conv(j+1)<0
272               j=j+1;
273      end
274      t_temp=zeros(1,N);
275      Y_I_final=zeros(1,N);
276      Y_Q_final=zeros(1,N);
277
278
279      for i=1:N
280       t_temp(i)=t_conv(j+(i-1)*over);
281       Y_I_final(i)=Y_I_filtered(j+(i-1)*over);
282       Y_Q_final(i)=Y_Q_filtered(j+(i-1)*over);
283      end
284      Y_Final(:,1)=Y_I_final(1:N);
285      Y_Final(:,2)=Y_Q_final(1:N);
286      [est_X,est_bit_seq]=detect_PSK_8(Y_Final);
287
288      sum1=sum1+symbol_errors(est_X,X);
289
290      sum2=sum2+bit_errors(est_bit_seq,b);
291
292
293     end
294      Esymbol(p)=sum1/(N*k);
295      Ebit(p)=sum2/(N*3*k);
296  end
297
298  temp=Q(SNR);
299  figure()
300  semilogy(SNR, Esymbol)
301  hold on
302  semilogy(SNR, temp)
```

## detect PSK 8.m

```
1   function [est_X,est_bit_seq]=detect_PSK_8(Y)
2
3   est_X=zeros(length(Y),1);
4   est_bit_seq=zeros(3*length(Y),1);
5   for k=1:length(Y)
6
7       if(Y(k,1)>(cosd(337.5)))%0 degrees
8           est_X(k,1)=1;
9           est_X(k,2)=0;
10          est_bit_seq(k*3-2)=0;
11          est_bit_seq(k*3-1)=0;
12          est_bit_seq(k*3)=0;
13
14      elseif(cosd(67.5)<Y(k,1)&&Y(k,1)<cosd(22.5)&&Y(k,2)>0)%45 ...
            degrees
15          est_X(k,1)=1/sqrt(2);
16          est_X(k,2)=1/sqrt(2);
17          est_bit_seq(k*3-2)=0;
```

```
18          est_bit_seq(k*3-1)=0;
19          est_bit_seq(k*3)=1;
20
21      elseif(Y(k,2)>sind(67.5))%90 degrees
22          est_X(k,1)=0;
23          est_X(k,2)=1;
24          est_bit_seq(k*3-2)=0;
25          est_bit_seq(k*3-1)=1;
26          est_bit_seq(k*3)=1;
27
28
29
30      elseif(cosd(112.5)>Y(k,1)&& Y(k,1)>cosd(157.5)&&Y(k,2)>0)%135 ...
            degrees
31          est_X(k,1)=-1/sqrt(2);
32          est_X(k,2)=1/sqrt(2);
33          est_bit_seq(k*3-2)=0;
34          est_bit_seq(k*3-1)=1;
35          est_bit_seq(k*3)=0;
36
37
38
39
40      elseif(Y(k,1)<cosd(157.5))%180 degrees
41          est_X(k,1)=-1;
42          est_X(k,2)=0;
43          est_bit_seq(k*3-2)=1;
44          est_bit_seq(k*3-1)=1;
45          est_bit_seq(k*3)=0;
46
47
48      elseif(cosd(202.5)<Y(k,1)&& Y(k,1)<cosd(247.5) && ...
            Y(k,2)<0)%225 degrees
49          est_X(k,1)=-1/sqrt(2);
50          est_X(k,2)=-1/sqrt(2);
51          est_bit_seq(k*3-2)=1;
52          est_bit_seq(k*3-1)=1;
53          est_bit_seq(k*3)=1;
54
55
56      elseif(Y(k,2)<sind(247.5))%270 degrees
57          est_X(k,1)=0;
58          est_X(k,2)=-1;
59          est_bit_seq(k*3-2)=1;
60          est_bit_seq(k*3-1)=0;
61          est_bit_seq(k*3)=1;
62
63
64
65          %est_X(k,1)>cos(292.5)&&est_X(k,1)<cos(337,5)&&est_X(k,2)<0
66      elseif(Y(k,1)>cosd(292.5) && Y(k,1)<cosd(337.5) && Y(k,2)<0)
67          est_X(k,1)=1/sqrt(2);
68          est_X(k,2)=-1/sqrt(2);
69          est_bit_seq(k*3-2)=1;
70          est_bit_seq(k*3-1)=0;
71          est_bit_seq(k*3)=0;
72      end
```

```
73 end
```

## bits to PSK 8.m

```
1  function X=bits_to_PSK_8(bit_seq)
2
3  N=length(bit_seq)/3;
4  X=zeros(N,2);
5
6  for k = 0:3:length(bit_seq)-1
7      i=k/3+1;
8      if((bit_seq(k+1)==0) && (bit_seq(k+2)==0) && bit_seq(k+3)==0)
9      X(i,1)=cos(0);
10     X(i,2)=sin(0);
11     elseif((bit_seq(k+1)==0) && (bit_seq(k+2)==0) && ...
           bit_seq(k+3)==1)
12     X(i,1)=cos(2*pi*1/8);
13     X(i,2)=sin(2*pi*1/8);
14      elseif((bit_seq(k+1)==0) && (bit_seq(k+2)==1) && ...
            bit_seq(k+3)==1)
15     X(i,1)=cos(2*pi*2/8);
16     X(i,2)=sin(2*pi*2/8);
17      elseif((bit_seq(k+1)==0) && (bit_seq(k+2)==1) && ...
            bit_seq(k+3)==0)
18     X(i,1)=cos(2*pi*3/8);
19     X(i,2)=sin(2*pi*3/8);
20      elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==1) && ...
            bit_seq(k+3)==0)
21     X(i,1)=cos(2*pi*4/8);
22     X(i,2)=sin(2*pi*4/8);
23      elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==1) && ...
            bit_seq(k+3)==1)
24     X(i,1)=cos(2*pi*5/8);
25     X(i,2)=sin(2*pi*5/8);
26      elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==0) && ...
            bit_seq(k+3)==1)
27     X(i,1)=cos(2*pi*6/8);
28     X(i,2)=sin(2*pi*6/8) ;
29      elseif((bit_seq(k+1)==1) && (bit_seq(k+2)==0) && ...
            bit_seq(k+3)==0)
30     X(i,1)=cos(2*pi*7/8);
31     X(i,2)=sin(2*pi*7/8) ;
32     end
33 end
34
35 end
```

## symbol errors.m

```
1  function num_of_symbol_errors=symbol_errors(est_X,X)
```

```
num_of_symbol_errors=0;
for i=1:length(X)
    if(abs(X(i,1)-est_X(i,1))>0.001)
        num_of_symbol_errors=num_of_symbol_errors+1;
    end
end
for i=1:length(X)
    if(abs(X(i,2)-est_X(i,2))>0.001)
        num_of_symbol_errors=num_of_symbol_errors+1;
    end
end

num_of_symbol_errors=ceil(num_of_symbol_errors/2);

end
```

## bit errors.m

```
function num_of_bit_errors = bit_errors(est_bit_seq,b)
num_of_bit_errors=0;
for i=1:length(b)
    if(abs(b(i)-est_bit_seq(i))>0.001)
        num_of_bit_errors=num_of_bit_errors+1;
    end
end

end
```

## Q

```
function y = Q(x)
% y=Q(x)

y = 0.5  * erfc( x /sqrt(2) );
```

## srrcpulse

```
function [phi, t] = srrc_pulse(T, over, A, a)

Ts=T/over;

% Create time axis
t = [-A*T:Ts:A*T] + 10^(-8); % in order to avoid division by zero …
    problems at t=0.

if (a>0 && a≤1)
   num = cos((1+a)*pi*t/T) + sin((1-a)*pi*t/T) ./ (4*a*t/T);
```

```
10      denom = 1-(4*a*t./T).^2;
11      phi = 4*a/(pi*sqrt(T)) * num ./ denom;
12  elseif (a==0)
13      phi = 1/(sqrt(T)) * sin(pi*t/T)./(pi*t/T);
14  else
15       phi = zeros(length(t),1);
16       disp('Illegal value of roll-off factor')
17       return
18  end
```