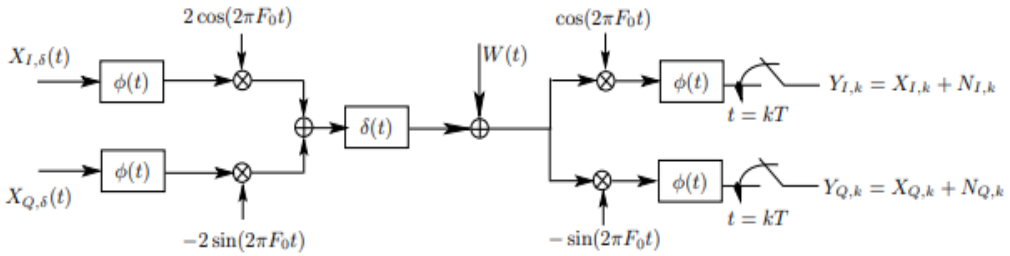
ΤΗΛ 301 Διδάσκων Α. Λιάβας
Αναφορά 3ης Εργασίας

Πρινίτης Πολύδωρος 2018030098
Λεοντής Παναγιώτης 2018030099

Μέρος Α



Σε αυτή την άσκηση, θα προσομοιώσουμε το παραπάνω τηλεπικοινωνιακό σύστημα υποθέτοντας ότι χρησιμοποιείται διαμόρφωση 8-PSK, και θα μελετήσουμε την απόδοσή του.

A.1)

```
%Task 1
b = (sign (randn(N,3)) + 1)/2;
```

<u>Σχολιασμός κώδικα</u>

Αρχικά δημιουργήσαμε την δυαδική ακολουθία b με στοιχεία 3N ισοπίθανα bits. Ενδεικτικά χρησιμοποιήσαμε N=100.

A.2)

```
%Task 2
Xn=bits_to_PSK_8(b);
XI=Xn(:,1);
XQ=Xn(:,2);
```

```
function [ x ] = bits_to_PSK_8( bit_seq )
%bits_to_PSK_8 converts a bit sequence into 8-PSK coding
%Picked Gray Code:
%              000->001->011->010->110->100->101->111
   for i=1:length(bit_seq)
       if bit_seq(i,1)==0 && bit_seq(i,2)==0 && bit_seq(i,3)==0
          x(i,1)=cos(2*pi*0/8);
          x(i,2)=sin(2*pi*0/8);
       elseif bit_seq(i,1)==0 && bit_seq(i,2)==0 && bit_seq(i,3)==1
          x(i,1)=cos(2*pi*1/8);
          x(i,2)=sin(2*pi*1/8);
        elseif bit_seq(i,1)==0 && bit_seq(i,2)==1 && bit_seq(i,3)==1
          x(i,1)=cos(2*pi*2/8);
          x(i,2)=sin(2*pi*2/8);
        elseif bit_seq(i,1)==0 && bit_seq(i,2)==1 && bit_seq(i,3)==0
          x(i,1)=cos(2*pi*3/8);
          x(i,2)=sin(2*pi*3/8);
        elseif bit_seq(i,1)==1 && bit_seq(i,2)==1 && bit_seq(i,3)==0
          x(i,1)=cos(2*pi*4/8);
          x(i,2)=sin(2*pi*4/8);
        elseif bit_seq(i,1)==1 && bit_seq(i,2)==0 && bit_seq(i,3)==0
          x(i,1)=cos(2*pi*5/8);
          x(i,2)=sin(2*pi*5/8);
        elseif bit_seq(i,1)==1 && bit_seq(i,2)==0 && bit_seq(i,3)==1
          x(i,1)=cos(2*pi*6/8);
          x(i,2)=sin(2*pi*6/8);
       elseif bit_seq(i,1)==1 && bit_seq(i,2)==1 && bit_seq(i,3)==1
          x(i,1)=cos(2*pi*7/8);
          x(i,2)=sin(2*pi*7/8);
       end
    end
end
```

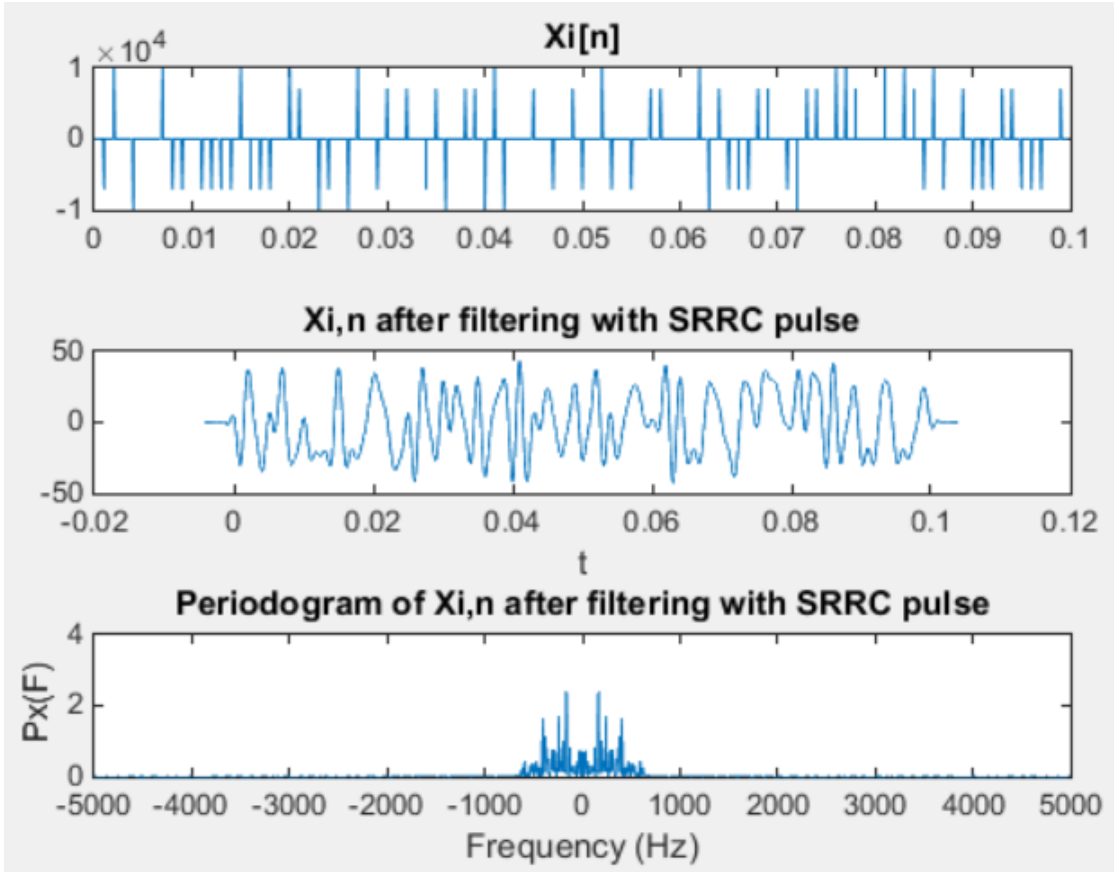
<u>Σχολιασμός Κώδικα</u>

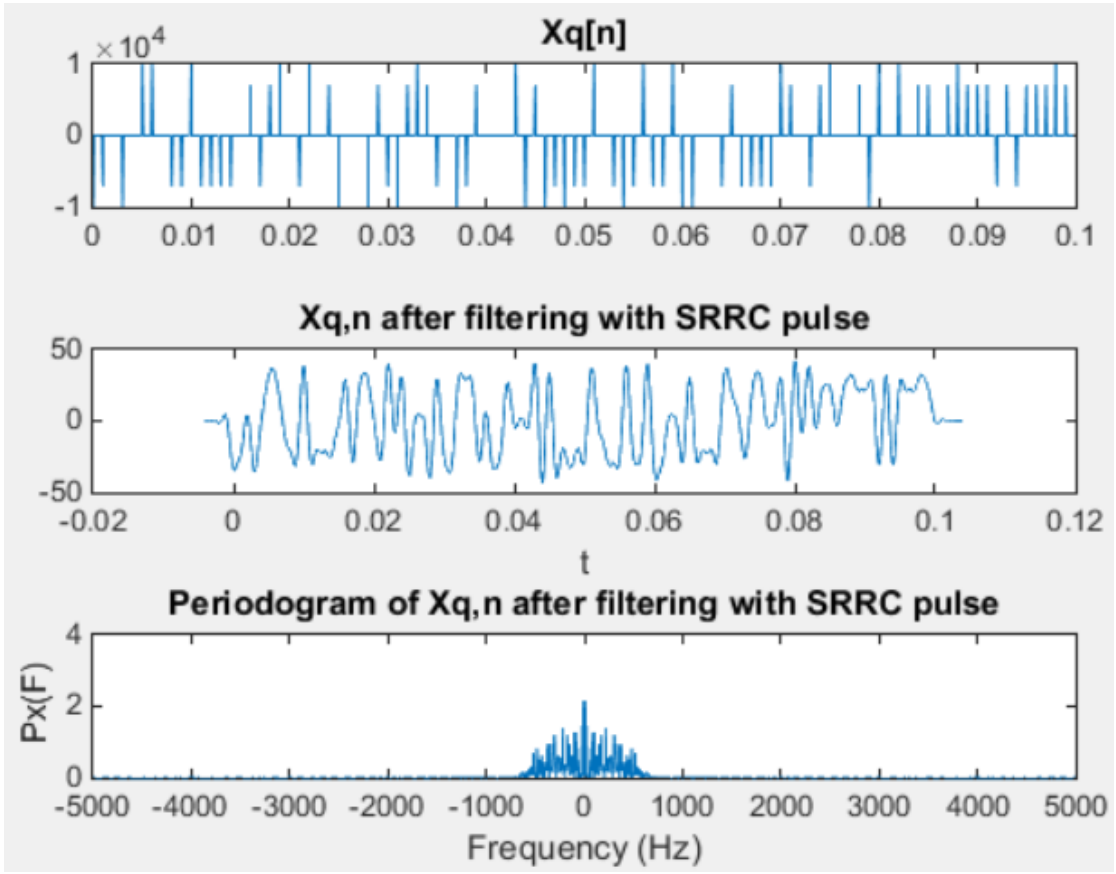
Υλοποιήσαμε την συνάρτηση bits_to_PSK_8 η οποία χρησιμοποιεί τον κώδικα Gray:

$$000 \rightarrow 001 \rightarrow 011 \rightarrow 010 \rightarrow 110 \rightarrow 100 \rightarrow 101 \rightarrow 111$$

Έτσι μπορέσαμε να κάνουμε την κατάλληλη απεικόνιση της ακολουθίας bits (b) σε ακολουθία 8-PSK συμβόλων. Ο πίνακας x που επιστρέφεται έχει N γραμμές και 2 στήλες. Από την στήλη 1 ορίζουμε το διάνυσμα XI και από την στήλη 2 το διάνυσμα XQ όπως φαίνεται παραπάνω.

A.3)


Xi[n]
×10^4
1
0
-1
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1
Xi,n after filtering with SRRC pulse
50
0
-50
-0.02 0 0.02 0.04 0.06 0.08 0.1 0.12
t
Periodogram of Xi,n after filtering with SRRC pulse
Px(F)
4
2
0
-5000 -4000 -3000 -2000 -1000 0 1000 2000 3000 4000 5000
Frequency (Hz)


Xq[n]
×10^4
1
0
-1
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1
Xq,n after filtering with SRRC pulse
50
0
-50
-0.02 0 0.02 0.04 0.06 0.08 0.1 0.12
t
Periodogram of Xq,n after filtering with SRRC pulse
Px(F)
4
2
0
-5000 -4000 -3000 -2000 -1000 0 1000 2000 3000 4000 5000
Frequency (Hz)

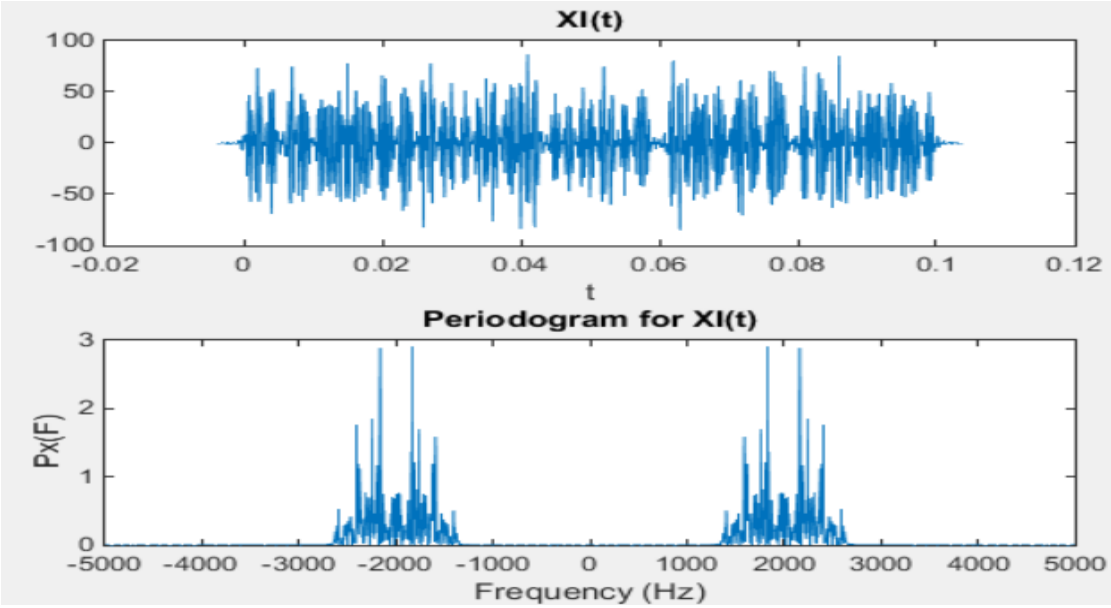
```
%% Task 3
%Create SRRC pulse
[ph,t] = srrc_pulse(T,over,A,a);
F = -Fs/2:Fs/Nf:Fs/2-Fs/Nf;
%Create Xd signals for usage in Convolution
Xdi = Fs*upsample(XI,over);
t_Xdi =(0:Ts:N/(1/T)-Ts);
Xdq = Fs*upsample(XQ,over);
t_Xdq = (0:Ts:N/(1/T)-Ts);
%Compute Convolution and create time axis. Also find length of Conv.
t_Xti = (t(1)+t_Xdi(1):Ts:t(end)+t_Xdi(end));
t_Xtq = (t(1)+t_Xdq(1):Ts:t(end)+t_Xdq(end));
Xti = conv(ph,Xdi)*Ts;
Xtq = conv(ph,Xdq)*Ts;
Xti_total=length(t_Xti)*Ts;
Xtq_total = length(t_Xtq)*Ts;
%Fourier Transform and compute Periodogram for both signals
Fxi=fftshift(fft(Xti,Nf)*Ts);
Pxi=(abs(Fxi).^2)/Xti_total;
Fxq=fftshift(fft(Xtq,Nf)*Ts);
Pxq=(abs(Fxq).^2)/Xtq_total;
%Show results for each signal
figure();
subplot(3,1,1);
plot(t_Xdi,Xdi);
title('Xi[n]');
subplot(3,1,2);
plot(t_Xti,Xti);
title('Xi,n after filtering with SRRC pulse');
xlabel('t');
subplot(3,1,3);
plot(F,Pxi);
title('Periodogram of Xi,n after filtering with SRRC pulse');
ylabel('Px(F)');
xlabel('Frequency (Hz)');

figure();
subplot(3,1,1);
plot(t_Xdq,Xdq);
title('Xq[n]');
subplot(3,1,2);
plot(t_Xtq,Xtq);
title('Xq,n after filtering with SRRC pulse');
xlabel('t');
subplot(3,1,3);
plot(F,Pxq);
title('Periodogram of Xq,n after filtering with SRRC pulse');
ylabel('Px(F)');
xlabel('Frequency (Hz)');
```

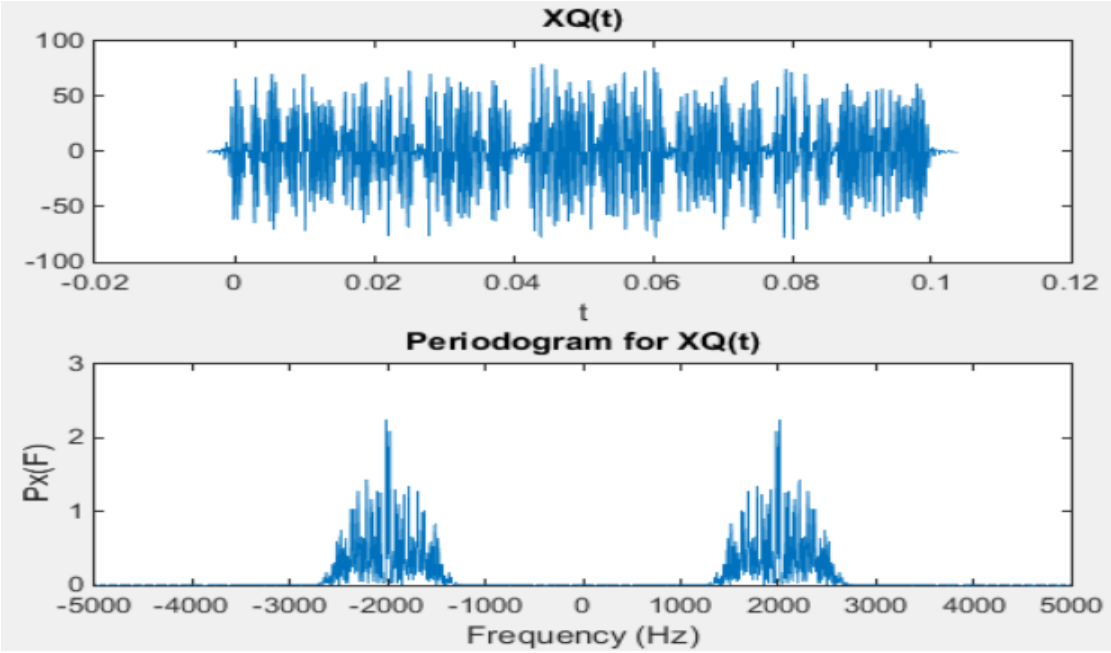
Σχολιασμός Κώδικα

Επιλέξαμε A=4 και δημιουργήσαμε τον παλμό SRRC από την συνάρτηση η οποία μας δίνεται στα έγγραφα του μαθήματος, με τις σταθερές που αναφέρονται στην

εκφώνηση. Έπειτα ορίσαμε το διάστημα της συχνότητας ώστε να χρησιμοποιηθεί στον σχεδιασμό των περιοδογραμμάτων. Χρησιμοποιήσαμε την upsample() ώστε να δημιουργήσουμε σήματα κατάλληλα να χρησιμοποιηθούν στην συνέλιξη. Ουσιαστικά φτιάξαμε τα σήματα $XI_{\delta}(t)$ και $XQ_{\delta}(t)$. Παρακάτω κάναμε την συνέλιξη του κάθε σήματος με τον παλμό(φιλτράρισμα) και δημιουργήσαμε τον κατάλληλο άξονα χρόνου. Έγινε ο μετασχηματισμός Fourier για τη κάθε συνέλιξη και υπολογίσαμε το περιοδόγραμμά του. Τέλος εμφανίζουμε τα φιλτραρισμένα σήματα ,τα περιοδογράμματά τους και τις ακολουθίες XI,n και XQ,n στους κατάλληλους άξονες με plot. (Εναλλακτικά χρησιμοποιούμε semilogy).

A.4)





```
%% TASK 4

%Create XI(t) and XQ(t) signals
XTi = 2*(Xti).*(cos(2*pi*Fo*transpose(t_Xti)));
Xti_total = length(t_Xti)*Ts;

XTq = -2*(Xtq).*(sin(2*pi*Fo*transpose(t_Xtq)));
Xtq_total = length(t_Xtq)*Ts;
%Fourier Transform and compute Periodogram for both signals
Fxi = fftshift(fft(XTi,Nf)*Ts);
Pxi = (abs(Fxi).^2)/Xti_total;

Fxq = fftshift(fft(XTq,Nf)*Ts);
Pxq = (abs(Fxq).^2)/Xtq_total;

%Show results for each signal
figure();
subplot(2,1,1);
plot(t_Xti,XTi);
title('XI(t)');
xlabel('t');
subplot(2,1,2);
plot(F,Pxi);
title('Periodogram for XI(t)');
ylabel('Px(F)');
xlabel('Frequency (Hz)');

figure();
subplot(2,1,1);
plot(t_Xtq,XTq);
title('XQ(t)')
xlabel('t');
subplot(2,1,2);
plot(F,Pxq);
title('Periodogram for XQ(t)');
ylabel('Px(F)');
xlabel('Frequency (Hz)');
```

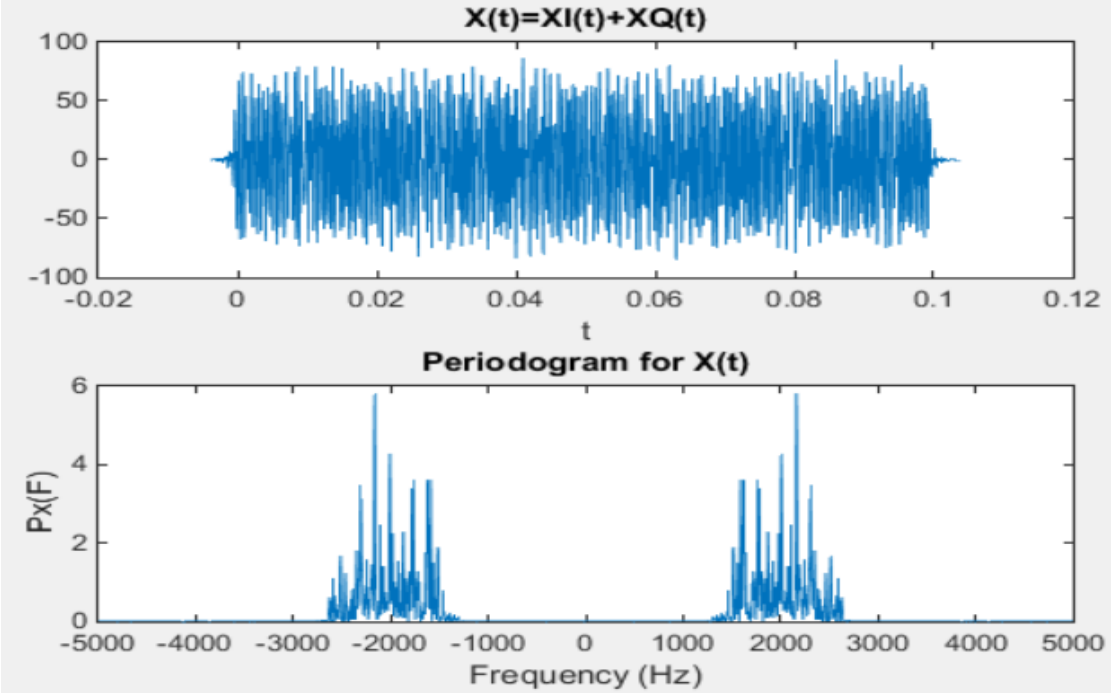
<u>Σχολιασμός Κώδικα</u>

Όπως αναφέρεται στην εκφώνηση της άσκησης, έχουμε Fo=2000 Hz. Στην συνέχεια πολλαπλασιάσαμε τα φιλτραρισμένα σήματα με τους αντίστοιχους φορείς συχνότητας Fo όπως φαίνεται και στην απεικόνιση του συστήματος. Έτσι δημιουργήσαμε τα σήματα $X_I(t)$ και $X_Q(t)$ όπως αυτά περιγράφονται στην εκφώνηση. Υπολογίσαμε τον μετασχηματισμό Fourier για κάθε σήμα και στην συνέχεια το περιοδόγραμμά του. Οι άξονες του χρόνου και της συχνότητας δεν αλλάζουν, επομένως εμφανίζουμε τα αποτελέσματα με plot. (Εναλλακτικά χρησιμοποιούμε semilogy).

Παρατηρήσεις

Από την στιγμή που πολλαπλασιάσαμε με τους φορείς, το σύστημα δεν καλείται πλέον βασικής ζώνης, αφού το φάσμα του αρχικού σήματος μετακινείται γύρω από τη συχνότητα του φορέα. Το παραπάνω μπορούμε να το συμπεράνουμε παρατηρώντας τα δύο περιοδογράμματα. Παρατηρούμε όντως ότι το φάσμα έχει μετακινηθεί γύρω από τη συχνότητα Fo(2000 Hz). Επίσης διακρίνουμε πως αυξήθηκε το πλάτος και η συχνότητα του κάθε σήματος αντίστοιχα, όπως ήταν αναμενόμενο καθώς πολλαπλασιάστηκαν με το φορέα.

A.5)




```
%% TASK 5
%Create channel input signal
Xt = XTi+XTq;
T_total = length(t_Xtq)*Ts;
%Fourier Transform and compute Periodogram
Fxt = fftshift(fft(Xt,Nf)*Ts);
Pxt = (abs(Fxt).^2)/T_total;

figure();
subplot(2,1,1);
plot(t_Xtq,Xt);
title('X(t)=XI(t)+XQ(t)');
xlabel('t');
subplot(2,1,2);
plot(F,Pxt);
title('Periodogram for X(t)');
ylabel('Px(F)');
xlabel('Frequency (Hz)');
```

Σχολιασμός Κώδικα

Προσθέσαμε τα δύο σήματα $X_I(t)$ και $X_Q(t)$ ώστε να σχηματίσουμε την είσοδο του καναλιού X(t). Υπολογίσαμε τον μετασχηματισμό Fourier και στην συνέχεια το περιοδόγραμμά του σήματος εισόδου. Τέλος σχεδιάσαμε στους κατάλληλους άξονες με plot. (Εναλλακτικά χρησιμοποιούμε semilogy).

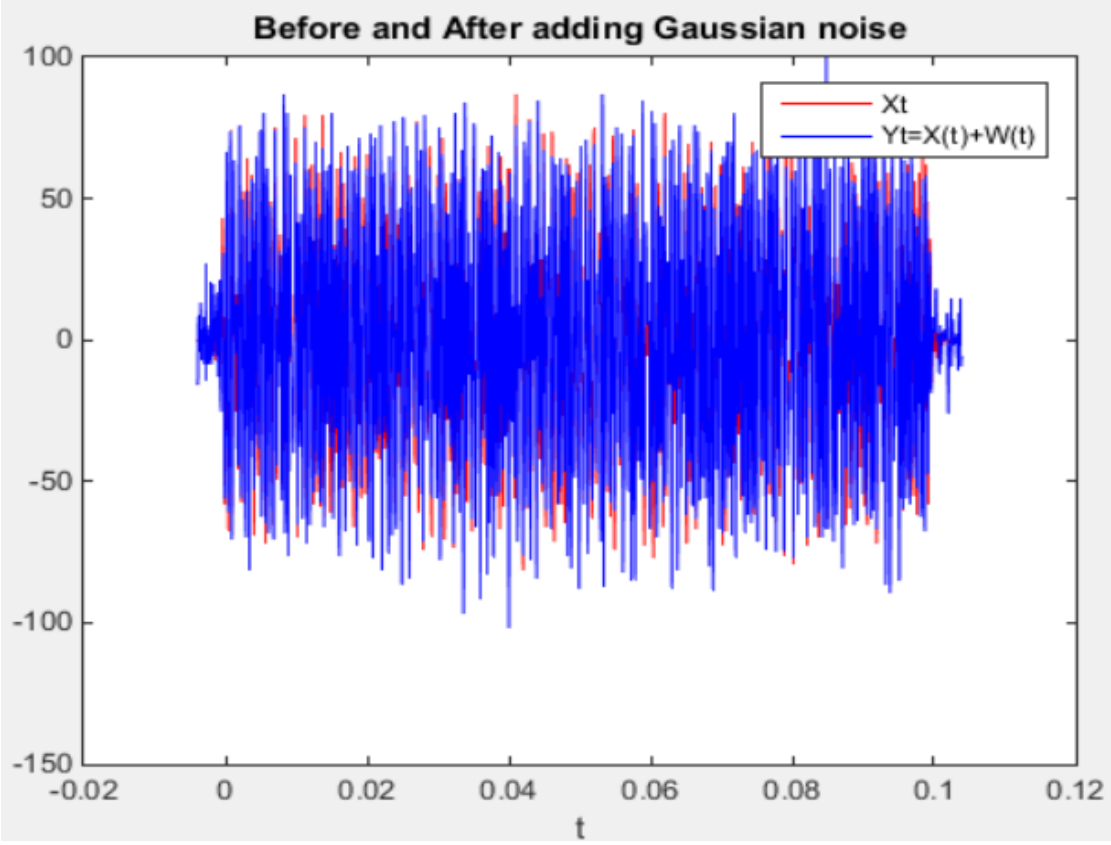
Παρατηρήσεις

Λαμβάνοντας το σήμα που τελικά θα είναι η είσοδος του καναλιού διακρίνουμε ότι το πλάτος έχει αυξηθεί. Αναμενόμενο λόγω της πρόσθεσης των δύο συνιστωσών. Ακόμη παρατηρούμε ξανά την επίδραση του φορέα στο αρχικό μας σήμα καθώς το κέντρο έχει μεταφερθεί στις συχνότητες Fo και –Fo.

A.6)

Υποθέτουμε πως το κανάλι είναι ιδανικό. Επομένως το σύστημα έχει κρουστική απόκριση δ(t), άρα η έξοδός του θα είναι το σήμα εισόδου αναλλοίωτο.

A.7)

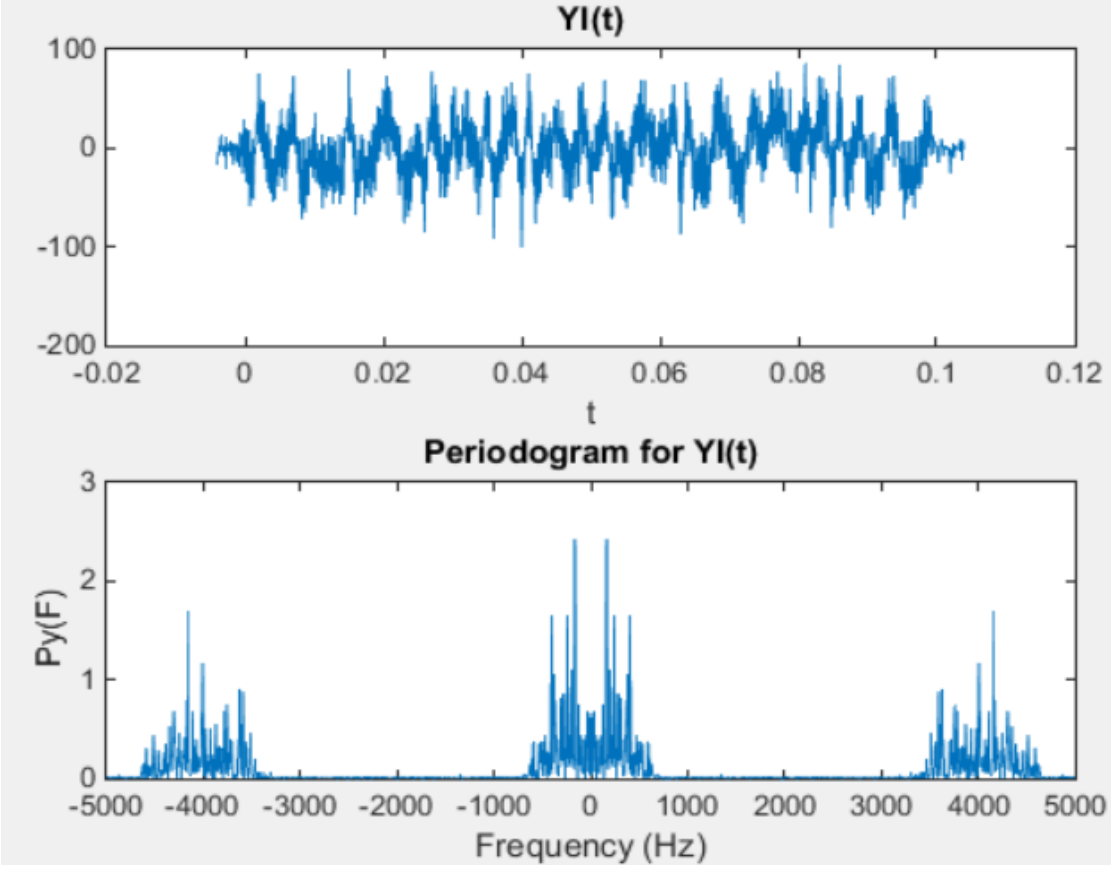


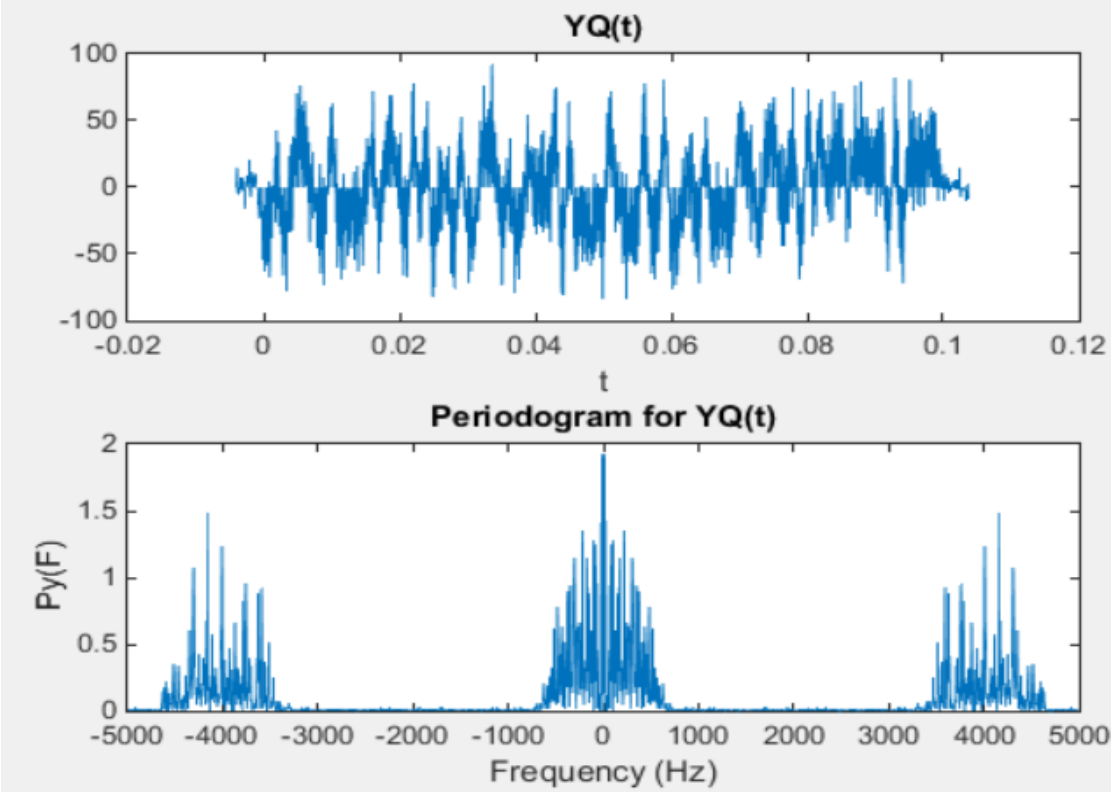
```
%% TASK 7
%Compute variance and create Gaussian Noise
SNR = 20;
s2w = 1/(Ts*(10^(SNR/10)));
Wt = sqrt(s2w).*randn(length(Xt),1);
%Create Y(t) signal
YT = Xt+Wt;
%Show results
figure();
plot(t_Xtq,Xt,'red');
hold on;
plot(t_Xtq,YT,'blue');
hold off;
title('Before and After adding Gaussian noise');
xlabel('t');
legend('Xt','Yt=X(t)+W(t)');
```

Σχολιασμός Κώδικα

Αρχικά επιλέξαμε SRN=20. Στην συνέχεια ορίσαμε την διασπορά του θορύβου και έπειτα δημιουργήσαμε το σήμα W(t) (Gaussian Noise). Προσθέσαμε την είσοδο του καναλιού X(t) και τον θόρυβο και δημιουργήσαμε την ενθόρυβη κυματομορφή Y(t). Ενδεικτικά σχεδιάζουμε το σήμα X(t) πριν και μετά την προσθήκη θορύβου.

A.8)






```
%% TASK 8
%Create YI(t) and YQ(t) signals
Yi_1 = YT.*(cos(2*pi*Fo*transpose(t_Xtq)));
Yq_1 = YT.*(-sin(2*pi*Fo*transpose(t_Xtq)));

Tyi1_total = length(t_Xtq)*Ts;
Tyq1_total = length(t_Xtq)*Ts;
%Fourier Transform and compute Periodogram for both signals
Fyi1_x = fftshift(fft(Yi_1,Nf)*Ts);
Pyi1_x = (abs(Fyi1_x).^2)/Tyi1_total;

Fyq1_x = fftshift(fft(Yq_1,Nf)*Ts);
Pyq1_x = (abs(Fyq1_x).^2)/Tyq1_total;
%Show results for each signal
figure();
subplot(2,1,1);
plot(t_Xtq,Yi_1);
title('YI(t)');
xlabel('t');
subplot(2,1,2);
plot(F,Pyi1_x);
title('Periodogram for YI(t)');
ylabel('Py(F)');
xlabel('Frequency (Hz)');

figure();
subplot(2,1,1);
plot(t_Xtq,Yq_1);
title('YQ(t)')
xlabel('t');
subplot(2,1,2);
```

```
plot(F,Pyq1_x);
title('Periodogram for YQ(t)');
ylabel('Py(F)');
xlabel('Frequency (Hz)');
```

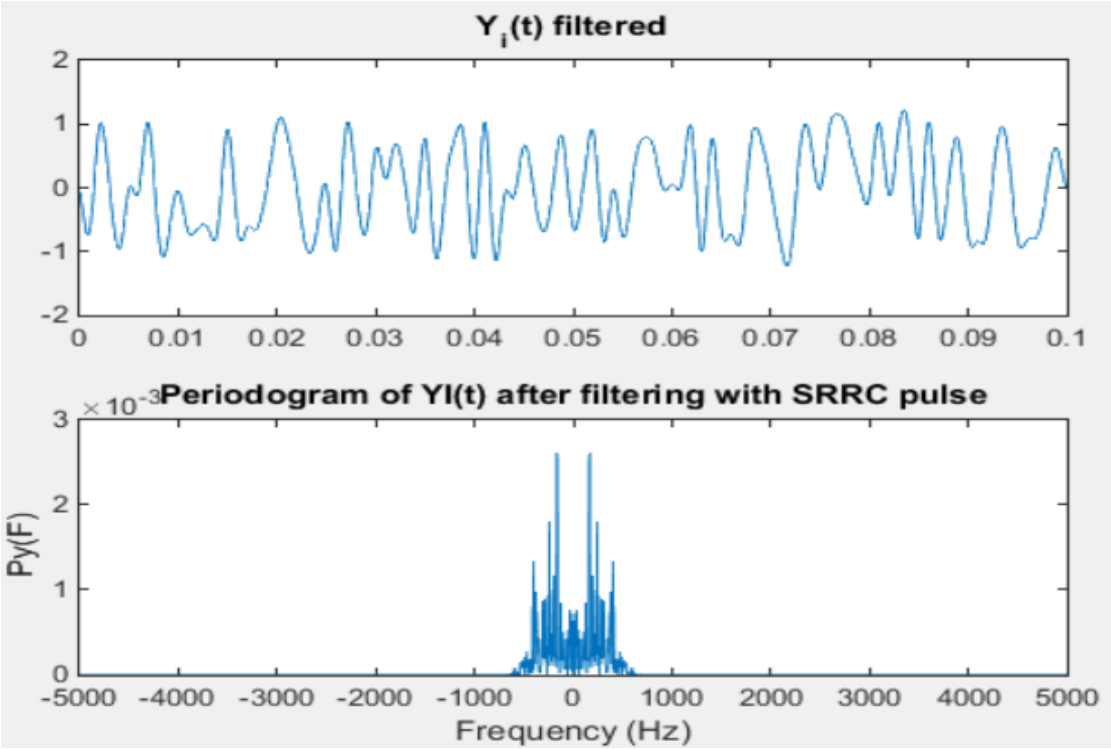
Σχολιασμός Κώδικα

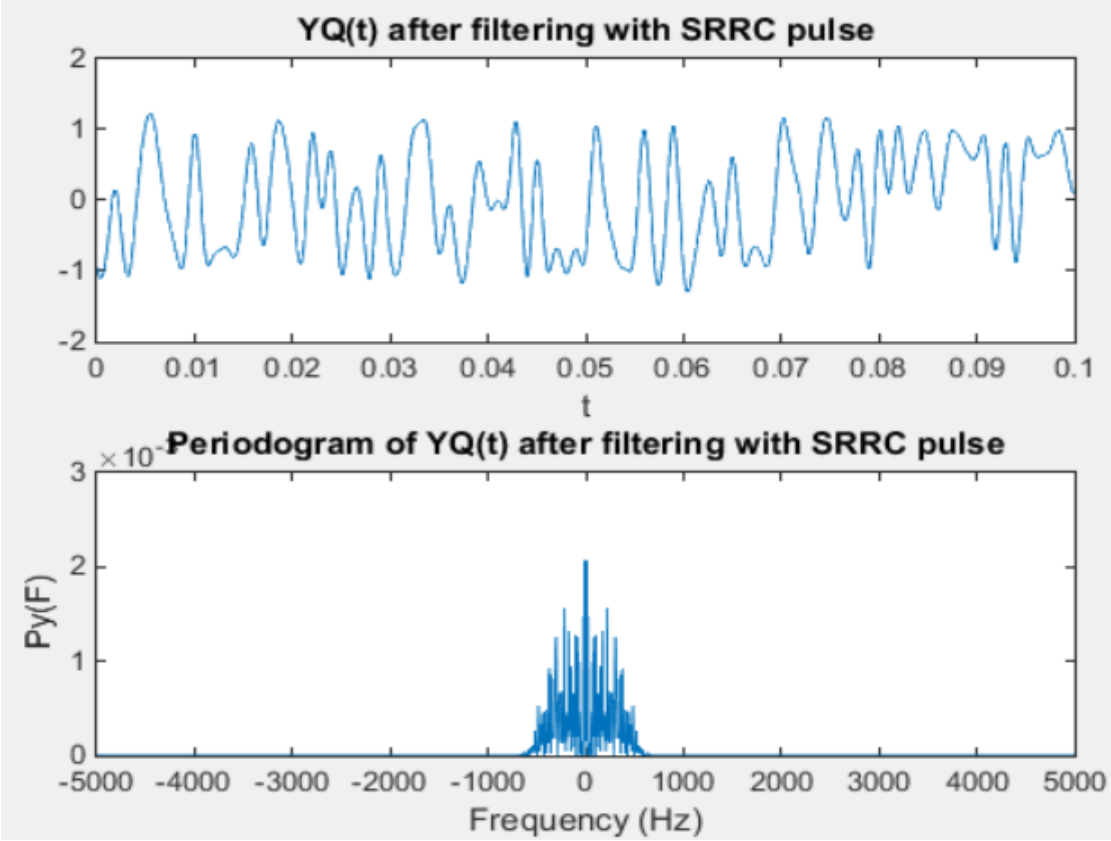
Αρχικά πολλαπλασιάζουμε την κάθε συνιστώσα με τον αντίστοιχο φορέα συχνότητας Fo όπως φαίνεται και στην απεικόνιση του συστήματος. Στη συνέχεια κάναμε τον μετασχηματισμό Fourier και υπολογίσαμε το περιοδόγραμμα για τα δύο σήματα που προκύπτουν ξεχωριστά. Τέλος σχεδιάζουμε τα σήματα και τα περιοδογράμματά τους με plot. (Εναλλακτικά χρησιμοποιούμε semilogy).

Παρατηρήσεις

Αρχικά πρέπει να επαναφέρουμε το σήμα στην βασική ζώνη ώστε να μελετηθεί περαιτέρω. Αυτός είναι και ο λόγος που πολλαπλασιάζουμε με τους αντίστοιχους φορείς(αποδιαμόρφωση). Στα περιοδογράμματα παρατηρούμε πως εμφανίστηκε το φάσμα στη βασική ζώνη, ενώ οι τιμές που αντιστοιχούν στις υψηλές συχνότητες που έχουν καθορίσει οι διαμορφωτές, εξακολουθούν να υπάρχουν. Γι αυτό τον λόγο και στο επόμενο βήμα θα χρησιμοποιήσουμε τα αντίστοιχα φίλτρα ώστε να περιορίσουμε το πλάτος των λοβών αυτών.

A.9)






```
%%% TASK 9
%Using SRRC pulse (ph) from Task 3
%Compute Convolution and create time axis.
Yi_2 = conv(ph,Yi_1)*Ts;
t_Yi_2 = (t(1)+t_Xtq(1):Ts:t(end)+t_Xtq(end));
Yq_2 = conv(ph,Yq_1)*Ts;
t_Yq_2 = (t(1)+t_Xtq(1):Ts:t(end)+t_Xtq(end));

%Tail cutting
counter = 0;
for n = 1:length(t_Yi_2)
    if t_Yi_2(n)<0
       counter = counter+1;
    end
end
Yi_cut = Yi_2(counter+1:(length(t_Yi_2)-(counter)));
t_Yi_cut = t_Yi_2(counter+1:(length(t_Yi_2)-(counter)));

Yq_cut = Yq_2(counter+1:(length(t_Yi_2)-(counter)));
t_Yq_cut = t_Yq_2(counter+1:(length(t_Yi_2)-(counter)));
%Fourier Transform and compute Periodogram for both cutted signals
Tyi_2_total = length(t_Yi_cut)*Ts;
Fyi_2 = fftshift(fft(Yi_cut,Nf)*Ts);
Pyi_2 = (abs(Fyi_2).^2)/Tyi_2_total;

Tyq_2_total = length(t_Yq_cut)*Ts;
Fyq_2 = fftshift(fft(Yq_cut,Nf)*Ts);
Pyq_2 = (abs(Fyq_2).^2)/Tyq_2_total;
%Show results for each signal
```

```
figure();
subplot(2,1,1);
plot(t_Yi_cut,Yi_cut);
title('Y_i(t) filtered');
subplot(2,1,2);
plot(F,Pyi_2);
title('Periodogram of YI(t) after filtering with SRRC pulse');
ylabel('Py(F)');
xlabel('Frequency (Hz)');

figure();
subplot(2,1,1);
plot(t_Yq_cut,Yq_cut);
title('YQ(t) after filtering with SRRC pulse')
xlabel('t');
subplot(2,1,2);
plot(F,Pyq_2);
title('Periodogram of YQ(t) after filtering with SRRC pulse');
ylabel('Py(F)');
xlabel('Frequency (Hz)');
```

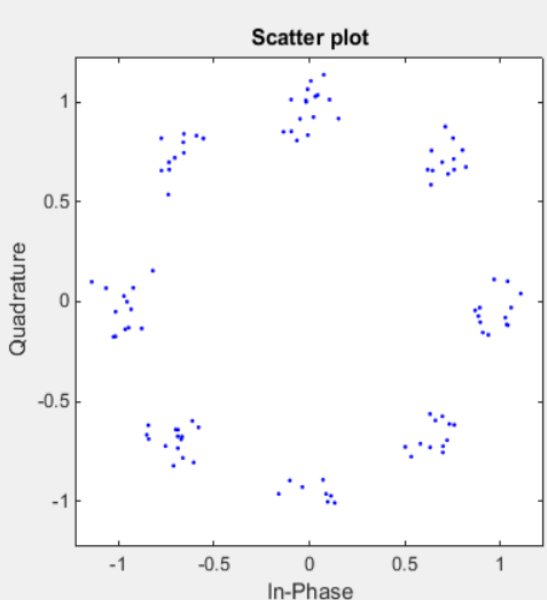
Σχολιασμός Κώδικα

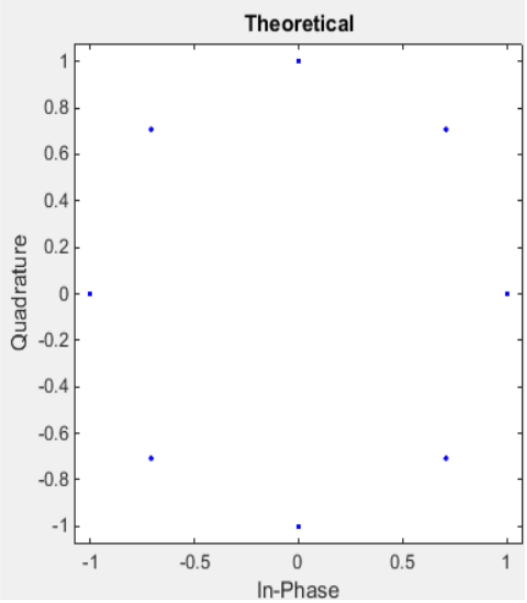
Αρχικά κάναμε την συνέλιξη μεταξύ του κάθε σήματος και του SRRC παλμού (ph) που δημιουργήθηκε στο ερώτημα Α.3. Στην συνέχεια ορίσαμε τον νέο άξονα του χρόνου ώστε να μπορέσουμε να σχεδιάσουμε τη συνέλιξη και μετέπειτα υπολογίσαμε την διάρκεια της συνέλιξης ώστε να χρησιμοποιηθεί στο περιοδόγραμμα. Στην συνέχεια με μια επαναληπτική δομή for εργαστήκαμε κατάλληλα ώστε να κόψουμε τις «ουρές» που εμφανίζονται στις δυο συνελίξεις και να δημιουργήσουμε δύο σήματα χωρίς αυτές ώστε να χρησιμοποιηθούν στην δειγματοληψία του επόμενου ερωτήματος. Υπολογίσαμε τον μετασχηματισμό Fourier και το περιοδόγραμμα της κάθε συνέλιξης(με αποκομμένη την ουρά). Τέλος σχεδιάσαμε τα φιλτραρισμένα σήματα (συνέλιξη) και το αντίστοιχο περιοδόγραμμά τους με plot.

Παρατηρήσεις

Παρατηρώντας τα περιοδογράμματα επαληθεύουμε ότι οι λοβοί στις υψηλές συχνότητες δεν λαμβάνονται υπόψην λόγω του φιλτραρίσματος. Επιπλέον διακρίνουμε μια ομοιότητα της κάθε συνιστώσας με την αντίστοιχη που είχε προκύψει στο ερώτημα Α.3.

Α.10)






```
%% TASK 10
%Downsampling and creating Yi,n and Yq,n
YI = downsample(Yi_filtered,over);
YQ = downsample(Yq_filtered,over);
%Creating Yn sequence
Yn=[YI YQ];
%Use Scatterplot for showing results
scatterplot(Yn);
```

<u>Σχολιασμός Κώδικα</u>

Αρχικά χρησιμοποιήσαμε το downsample() ώστε να γίνει η δειγματοληψία στις δυο συνελίξεις με αποκομμένες τις «ουρές». Έπειτα για κάθε σύμβολο δημιουργούμε τον πίνακα Yn με N (επιλέξαμε 100) γραμμές και 2 στήλες.

<u>Παρατηρήσεις</u>

Όπως ήταν αναμενόμενο παρατηρούμε ότι ο θόρυβος που προστέθηκε έχει δημιουργήσει αρκετές αποκλίσεις οι οποίες είναι ανάλογες του SNR. Έτσι προκύπτει και το νέφος γύρω από τα 8 σημεία που θα αναμέναμε να παρουσιάζονται εάν δεν υπήρχε θόρυβος. Για λόγους σύγκρισης παραθέτουμε την θεωρητική απεικόνιση της 8-PSK διαμόρφωσης.

A.11)

```
%% Task 11
[est_X,est_bit_seq]=detect_PSK_8(Yn);
```

```
function [est_X,exit_bit_seq ] = detect_PSK_8( Y )
%Compare distances and decide according to nearest neighbor rule
J=sqrt(2)/2;
```

```
for i=1:length(Y)
        dx0 = norm([1 0]-Y(i,:));
        dx1 = norm([J J]-Y(i,:));
        dx2 = norm([0 1]-Y(i,:));
        dx3 = norm([-J J]-Y(i,:));
        dx4 = norm([-1 0]-Y(i,:));
        dx5 = norm([-J -J]-Y(i,:));
        dx6 = norm([0 -1]-Y(i,:));
        dx7 = norm([J -J]-Y(i,:));

        dmin = min([dx0,dx1,dx2,dx3,dx4,dx5,dx6,dx7]);
% Decision of symbol m=0,1,..7
        if (dmin == dx0)
            est_X(i,:) = [1 0];
        elseif (dmin == dx1)
            est_X(i,:) = [J J];
        elseif (dmin == dx2)
            est_X(i,:) = [0 1];
        elseif (dmin == dx3)
            est_X(i,:) = [-J J];
        elseif (dmin == dx4)
            est_X(i,:) = [-1 0];
        elseif (dmin == dx5)
            est_X(i,:) = [-J -J];
        elseif (dmin == dx6)
            est_X(i,:) = [0 -1];
        elseif (dmin == dx7)
            est_X(i,:) = [J -J];
        end
end

%Decoding
%Gray code picked for bits_to_PSK_8 :
%000->001->011->010->110->100->101->111
for i=1:length(Y)
    if est_X(i,1)==1  && est_X(i,2)==0                          %m=0
        exit_bit_seq(i,:)=[0 0 0];
    elseif est_X(i,1)==J  && est_X(i,2)==J                      %m=1
        exit_bit_seq(i,:)=[0 0 1];
    elseif est_X(i,1)==0  && est_X(i,2)==1                      %m=2
        exit_bit_seq(i,:)=[0 1 1 ];
    elseif est_X(i,1)==-J  && est_X(i,2)==J                     %m=3
        exit_bit_seq(i,:)= [ 0 1 0];
    elseif est_X(i,1)==-1  && est_X(i,2)==0                     %m=4
        exit_bit_seq(i,:)=[1 1 0];
    elseif est_X(i,1)==-J && est_X(i,2)==-J                     %m=5
        exit_bit_seq(i,:)=[1 0 0];
    elseif est_X(i,1)==0  && est_X(i,2)==-1                     %m=6
        exit_bit_seq(i,:)=[1 0 1];
    elseif est_X(i,1)==J  && est_X(i,2)==-J                     %m=7
        exit_bit_seq(i,:)=[1 1 1];
    end
end
end
```

Σχολιασμός Κώδικα

Αρχικά για να εκτιμήσουμε την ακολουθία συμβόλων εισόδου χρησιμοποιήσαμε μια επαναληπτική δομή for για κάθε σύμβολο του σήματος που δίνεται σαν όρισμα. Έπειτα για κάθε δυάδα συμβόλων υπολογίσαμε την απόσταση που έχει από τις υπόλοιπες δυάδες με την χρήση της συνάρτησης norm. Στην συνέχεια χρησιμοποιήσαμε την συνάρτηση min ώστε να βρούμε την ελάχιστη απόσταση και να δώσουμε την αντίστοιχη τιμή στην εκτιμώμενη ακολουθία συμβόλων.

Έπειτα για να υπολογίσουμε την εκτιμώμενη δυαδική ακολουθία εισόδου εργαζόμαστε αντίστροφα από τον τρόπο υλοποίησης της bits_to_PSK_8(). Μέσα σε μια for ελέγχουμε κάθε φορά τις δύο συνιστώσες. Συγκρίνουμε την τιμή των δύο συνιστωσών με το $\cos\left(\frac{2\pi m}{8}\right), \sin\left(\frac{2\pi m}{8}\right)$ $m = 0, \dots 7$ αντίστοιχα και σύμφωνα με τον κώδικα Gray με τον οποίο είχαμε κωδικοποιήσει δημιουργούμε την αντίστοιχη τριάδα που αντιστοιχεί στο m. Η τριάδα αποθηκεύεται μέσα στον πίνακα N γραμμών και 3 στηλών est_bit_seq.

A.12)

```
%% Task 12
Num_Of_Symbol_Errors=num_of_symbol_errors(est_X,Xn)
```

```
function [ num_of_symbol_errors] = num_of_symbol_errors(est_X,X )
%Compares the symbols at the input(X) with the estimation of them at the
%output(est_X)
%       est_X:          Output Symbol Sequence Estimation
%       X:              Input Symbol Sequence
%Returns total number of mismatches found
num_of_symbol_errors=0;
N=length(X);
%Posible values [-1, -sqrt(2)/2 , 0 , sqrt(2)/2 , 1]
%Multiply by 2 and make values integer so we can compare
est_X_toInt=round(2.*est_X);
X_toInt=round(2.*X);
%Now possible values are [-2 ,-1, 0, 1, 2]
for i=1:N
    %There is a mismatch.Increase total errors
    if est_X_toInt(i,1)~=X_toInt(i,1) || est_X_toInt(i,2)~=X_toInt(i,2)
        num_of_symbol_errors=num_of_symbol_errors+1;
    end
end
end
```

Σχολιασμός Κώδικα

Σε αυτή τη συνάρτηση συγκρίνουμε τις ακολουθίες συμβόλων που στάλθηκαν με αυτές που εκτιμήθηκαν. Αυτές που εκτιμήθηκαν είναι αυτές που προέκυψαν από την detect_PSK_8. Να επισημάνουμε πως εμφανίστηκε ένα πρόβλημα καθώς σε

μερικά σύμβολα η τιμή ήταν σε Integer ενώ σε κάποια άλλα σε Double και δεν μπορούσε να γίνει σύγκριση (πχ 0 και 0.00000). Έτσι πολλαπλασιάσαμε με το 2 και χρησιμοποιήσαμε το round ώστε να στρογγυλοποιήσουμε όλα τα σύμβολα στον κοντινότερο ακέραιο και να γίνει σωστά η σύγκριση. Εάν δεν διπλασιάζαμε τότε οι τιμές $\pm\frac{\sqrt{2}}{2}$ θα στρογγυλοποιούνταν στην μονάδα και θα συμπίπτανε με την στρογγυλοποίηση του πιθανού συμβόλου τιμής ± 1 αντίστοιχα. Ελέγχουμε τις δύο συνιστώσες της κάθε ακολουθίας με της άλλης και αν δεν ταιριάζουν και οι δύο συνιστώσες έχουμε λάθος και αυξάνουμε τον συνολικό αριθμό λαθών.

A.13)

```
%% Task 13
 Num_Of_Bits_Errors=num_of_bit_errors(est_bit_seq,b)
```

```
function [ num_of_bit_errors ] = num_of_bit_errors(est_bit_seq,b )
%Compares the bit sequence of input signal and the estimated bit sequence of
output
%       est_bit_seq:        Output Binary 3-bit Sequence
%       b:                  Input Binary 3-bit Sequence
%Returns total number of mismatches found

%Number or triads
N=length(b);
num_of_bit_errors=0;
%Repeat for all triads
for i=1:N
    %There is a mismatch. Increase total errors
    if b(i,1)~=est_bit_seq(i,1) ||b(i,2)~=est_bit_seq(i,2) ||b(i,3)~=est_bit_seq(i,3)
        num_of_bit_errors=num_of_bit_errors+1;
    end
end
```

<u>Σχολιασμός Κώδικα</u>

Σε αυτή τη συνάρτηση εργαστήκαμε με παρόμοιο τρόπο με την προηγούμενη. Πήραμε την εκτίμηση της δυαδικής ακολουθίας που προέκυψε από την detect_PSK_8 και την συγκρίναμε με την δυαδική ακολουθία της εισόδου. Σύμφωνα με την αρχή του κώδικα Gray έχουμε αριθμούς των 3-bit οι οποίοι διαφέρουν μόνο κατά ένα bit από τον προηγούμενο και τον επόμενό τους. Επομένως ένα μόνο λάθος bit μπορεί να υπάρχει στην εκτιμώμενη ακολουθία από bit .

<u>Παρατηρήσεις</u>

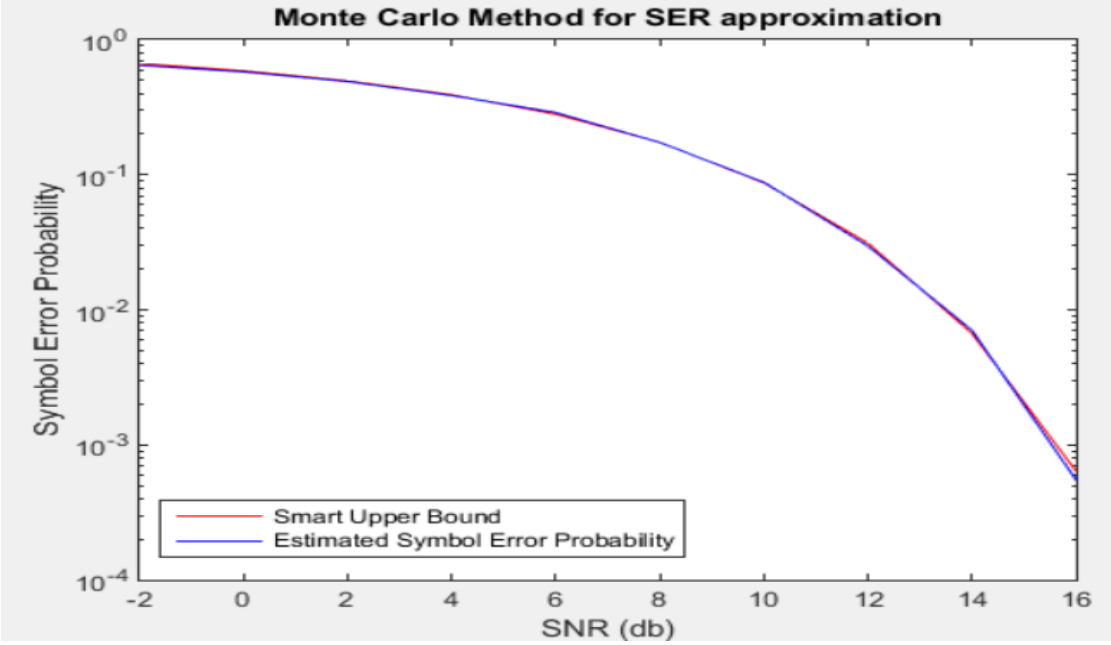
Τρέχουμε τις δύο συναρτήσεις και παρατηρούμε πως δεν έχουμε λάθος στην μετάδοση των συμβόλων αλλά ούτε και των bits. Ενδεχομένως εάν χρησιμοποιούσαμε ισχυρότερο θόρυβο, να είχαμε κάποιο σφάλμα μετάδοσης.

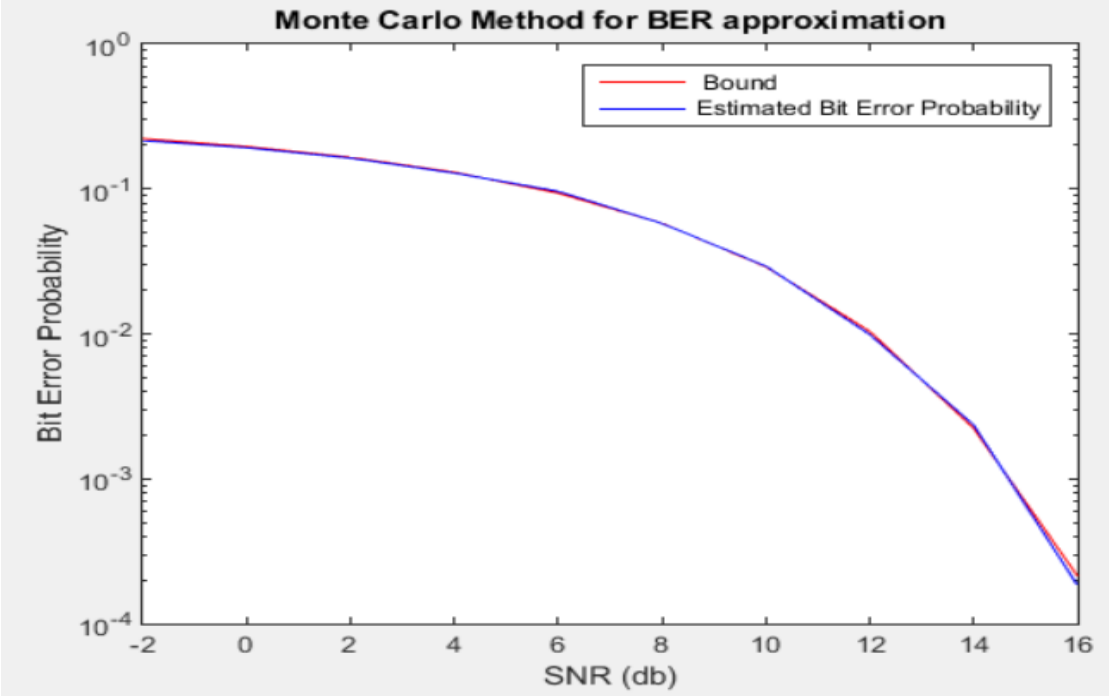
```
Num_Of_Symbol_Errors =

     0
```

```
Num_Of_Bits_Errors =

     0
```

Για περαιτέρω επιβεβαίωση μπορούμε και να εκτυπώσουμε τις μεταβλητές Xn, est_X και τις μεταβλητές b, est_bit_seq για να δούμε ότι δεν διαφέρουν σε κανένα σημείο.(Έχουμε επιλέξει SNR_{db}=20)

Μέρος Β





```
N=100;
SNR=[-2:2:16];
T=10^(-3);
over=10;
Ts=T/over;
Fs=1/Ts;
A=4;
a=1/2;
Fo=2000;
K=200;
%Create SRRC pulse
[ph,t] = srrc_pulse(T,over,A,a);

%For different values of SNR repeat experiment and calculate
%Symbol Error Probability and Bit Error Probability
for test=1:length(SNR)
    %Initalize for each repetition
    Num_Of_Symbol_Errors=0;
    Num_Of_Bits_Errors=0;

    for samples=1:K
        b = (sign (randn(N,3)) + 1)/2;

        Xn=bits_to_PSK_8(b);
        XI=Xn(:,1);
        XQ=Xn(:,2);
        %Create Xd signals for usage in Convolution
        Xdi = Fs*upsample(XI,over);
        t_Xdi =(0:Ts:N/(1/T)-Ts);
        Xdq = Fs*upsample(XQ,over);
        t_Xdq = (0:Ts:N/(1/T)-Ts);
        %Compute Convolution and create time axis. Also find length of Conv.
        t_Xti = (t(1)+t_Xdi(1):Ts:t(end)+t_Xdi(end));
        t_Xtq = (t(1)+t_Xdq(1):Ts:t(end)+t_Xdq(end));
        Xti = conv(ph,Xdi)*Ts;
        Xtq = conv(ph,Xdq)*Ts;

        %Create XI(t) and XQ(t) signals (modulation)
        XTi = 2*(Xti).*(cos(2*pi*Fo*transpose(t_Xti)));

        XTq = -2*(Xtq).*(sin(2*pi*Fo*transpose(t_Xtq)));

       %Create channel input signal
        Xt = XTi+XTq;
        %Compute variance and create Gaussian Noise
        s2w = 1/(Ts*(10^(SNR(test)/10)));
        s2n=Ts*s2w/2;
        Wt = sqrt(s2w).*randn(length(Xt),1);
        %Create Y(t) signal
        Yt = Xt+Wt;

        %Create YI() and YQ(t) signals( de-modulation)
        Yi_1 = Yt.*(cos(2*pi*Fo*transpose(t_Xtq)));
        Yq_1 = Yt.*(-sin(2*pi*Fo*transpose(t_Xtq)));

        %Compute Convolution and create time axis.
        Yi_2 = conv(ph,Yi_1)*Ts;
        t_Yi_2 = (t(1)+t_Xtq(1):Ts:t(end)+t_Xtq(end));
        Yq_2 = conv(ph,Yq_1)*Ts;

        %Tail cutting
```

```
        counter = 0;
        for n = 1:length(t_Yi_2)
          if t_Yi_2(n)<0
            counter = counter+1;
          end
        end
        Yi_cut = Yi_2(counter+1:(length(t_Yi_2)-(counter)));
        Yq_cut = Yq_2(counter+1:(length(t_Yi_2)-(counter)));

       %Downsampling and creating Yi,n and Yq,n
        YI = downsample(Yi_cut,over);
        YQ = downsample(Yq_cut,over);
        %Creating Yn sequence
        Yn=[YI YQ];

        %Detect 8-PSK Sequence and calculate symbol errors and bit errors
        [est_X,est_bit_seq]=detect_PSK_8(Yn);

        Num_Of_Symbol_Errors=Num_Of_Symbol_Errors+num_of_symbol_errors(est_X,Xn);
        Num_Of_Bits_Errors= Num_Of_Bits_Errors+num_of_bit_errors(est_bit_seq,b);
    end
    %Store probability of error on each sequence into matrix for every repetition
    Symbol_Errors(1,test)=Num_Of_Symbol_Errors/(N*K);
    Bits_Errors(1,test)=Num_Of_Bits_Errors/(N*K*3);

    symbol_bound(test)=2*Q(1./(sqrt(s2n))*sin(pi/8));
    %Log2(8)=3
    bit_bound(test)=symbol_bound(test)/3;
end

%Show results with semilogy
figure()
semilogy(SNR,symbol_bound,'red');
hold on;
semilogy(SNR,Symbol_Errors,'blue');
hold off;
xlabel('SNR (db)');
ylabel('Symbol Error Probability');
legend('Smart Upper Bound','Estimated Symbol Error Probability');
title('Monte Carlo Method for SER approximation');

figure()
semilogy(SNR,bit_bound,'red');
hold on;
semilogy(SNR,Bits_Errors,'blue');
hold off;
xlabel('SNR (db)');
ylabel('Bit Error Probability');
legend(' Bound','Estimated Bit Error Probability');
title('Monte Carlo Method for BER approximation');
```

<u>Σχολιασμός Κώδικα</u>

Πρακτικά στο μέρος Β επαναλάβαμε ακριβώς τα ζητούμενα του μέρους Α μέσα σε δύο επαναλήψεις for. Μια επανάληψη είναι για τις διάφορές τιμές του SNR και για κάθε τιμή επαναλαμβάνουμε το μέρος Α για ενδεικτικά k=200 επαναλήψεις ώστε να εκτιμήσουμε την πιθανότητα σφάλματος συμβόλου και την πιθανότητα σφάλματος

bit. Συνοπτικά δημιουργούμε την τυχαία ακολουθία από bits (b) και έπειτα την απεικόνιση σε 8-PSK. Στην συνέχεια παίρνουμε τις δυο συνιστώσες που προκύπτουν και τις φιλτράρουμε με τον παλμό SRRC. Πολλαπλασιάζουμε τα δύο φιλτραρισμένα σήματα με τον αντίστοιχο φορέα συχνότητας Fo και δημιουργούμε το σήμα εισόδου X(t) του καναλιού. Αφού του προσθέσουμε τον θόρυβο ανακτάμε τις δύο συνιστώσες του ενθόρυβου σήματος και κάνουμε αποδιαμόρφωση. Στην συνέχεια τις φιλτράρουμε με τον SRRC παλμό και αποκόπτουμε τις «ουρές» που εμφανίζονται ώστε να γίνει σωστά η δειγματοληψία. Γίνεται η δειγματοληψία και σχηματίζουμε την ακολουθία εξόδου Yn. Έπειτα μέσω της detect_PSK_8 εξάγουμε την ακολουθία από σύμβολα και την ακολουθία από bits και στην συνέχεια υπολογίζουμε τα σφάλματα που προέκυψαν στην κάθε ακολουθία και τα προσθέτουμε στα αρχικά ώστε τελικά να έχουμε τον συνολικό τους αριθμό. Υπολογίζουμε την αντίστοιχη πιθανότητα σφάλματος σύμφωνα με τους τύπους που γνωρίζουμε από την θεωρία:

$$P\left(E_{symbol}\right) = \frac{συνολικό\ πλήθος\ σφαλμάτων\ απόφασης\ συμβόλου}{συνολικό\ πλήθος\ απεσταλμένων\ συμβόλων}$$

$$P(E_{bit}) = \frac{συνολικό\ πλήθος\ σφαλμάτων\ απόφασης\ bit}{συνολικό\ πλήθος\ απεσταλμένων\ bit}$$

Τέλος για κάθε τιμή του SNR υπολογίζουμε το έξυπνο άνω φράγμα για την πιθανότητα σφάλματος συμβόλου και το άνω φράγμα για την πιθανότητα σφάλματος bit. Για τον υπολογισμό χρειάστηκε η συνάρτηση Q() η οποία μας δόθηκε υλοποιημένη. Σχεδιάζουμε σε κοινό semilogy τα αντίστοιχα πειραματικά και θεωρητικά δεδομένα.

<u>Παρατηρήσεις</u>

Αρχικά παρατηρούμε ελάχιστη απόκλιση και στα δυο διαγράμματα μεταξύ της αντίστοιχης πειραματικής και θεωρητικής καμπύλης. Εάν επιθυμείται καλύτερη προσέγγιση μπορούμε να αυξήσουμε τον αριθμό των επαναλήψεων k. Επιπλέον επαληθεύουμε την επιρροή της τιμής του SNR_{db}. Όπως γνωρίζουμε από την θεωρία, όσο αυξάνεται το SNR θα περιμένουμε μικρότερη πιθανότητα σφάλματος. Τέλος παρατηρούμε πως οι γραφικές παραστάσεις αλλάζουν μορφή για τις διάφορες τιμές του A και του συντελεστή roll-off (a).