Πολυτεχνείο Κρήτης Τμήμα ΗΜΜΥ

Διδάσκων: Αθανάσιος Λιάβας

Τηλεπικοινωνιακά Συστήματα Ι: Άσκηση 3η

Περίδης Γιάννης 2018030069

Σκλάβος Παναγίωτης 2018030170

Χειμερινό Εξάμηνο 2020-2021

Θέμα Α

Στο πρώτο κομμάτι της άκσησης το ζητούμενο είναι η προσομοίωση του τηλεπικοινωνιακού συστήματος που φαίνεται παρακάτω, πραγαματοποιώντας την υπόθεση ότι χρησιμοποιείται 8-PSK διαμόρφωση, όπως αυτή μελετήθηκε στο μάθημα. Σε δεύτερη φάση, εφόσον έχει παραχθεί το ζητούμενο σύστημα, γίνεται μελέτη της απόδοσής του.

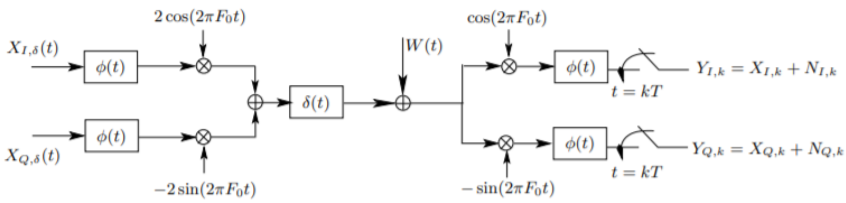


Figure 1: Illustration of the system that is being analyzed

A.1

Ξεκινώντας λοιπόν, θεωρείται, σύμφωνα με την εκφώνηση, $N = 100$ και δημιουργείται δυαδική ακολουθία bit_seq με $3N = 300$ ισοπίθανα $bits$ (ομοιόμορφα κατανεμημένα).

```
%A.1

%N number of bits
N=100;
%creating a'a random sequence of 3*N independent and  equally possible bits
%disp('a random sequence of 3*N independent and  equally possible bits')
b=(sign(randn(3*N,1))+1)/2;
```

A.2

Σε δεύτερη φάση δημιουργείται συνάρτηση $bits\_to\_PSK\_8$ η οποία, μετατρέπει την ακολουθία bit_seq των 3N $bits$ του προηγούμενου ερωτήματος, σε ακολουθία $8-PSK$ συμβόλων X. Η νέα ακολοθουθία συμβόλων έχει μήκος N, καθώς κωδικοποιούμε τριάδες $bits$ σε 1 σύμβολο. Συνεπώς, με κώδικα $Gray$ τριών $bits$ μπορούμε να περιγράψουμε πλήρως εως και 8 καταταστάσεις. Η κωδικοποίηση αυτή, όχι μόνο είναι επαρκής, αλλα και αποδοτική καθώς μειώνει της πιθανότητας σφάλματος αργότερα στην διαδικασία. Αυτο συμβαίνει διότι τα γειτονικά συμβολα έχουν απόσταση 1 bit, οπότε ένα σφάλμα συμβόλου, το οποίο κωδικοποιεί 3 $bits$, πιθανότατα θα αποφέρει σφάλμα ενός μονάχα βιτ. Η κώδικας $Gray$ λοιπόν που χρησιμοποιείται είναι ο ακόλουθως:

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

όπου κάθε διάνυσμα X_n, $n = 0, \ldots, N-1$ παίρνει τιμές από το αλφάβητο $8-PSK$ $x_0, \ldots, x_7$ με:

$$x_m = \begin{bmatrix} \cos(\frac{2\pi m}{8}) \\ \sin(\frac{2\pi m}{8}) \end{bmatrix}, n = 0, \ldots, N-1$$

Συνεπώς, στην θέση με γωνία 0 rad θα τοποθετούνται τα $bits$ 000, στην θέση με γωνία: $\frac{2\pi}{8}$ rad, θα τοποθετούνται τα $bits$ 001 κ.ο.κ

Ουσιαστικά λοιπον, τα σύμβολα αποτελούνται απο μία $in - phase$ και μια $quadrature$ συνιστώσα, οπότε καταλαμβάνουν θέσεις και στα δύο πεδία (πραγματικό, φανταστικό), με ομοιόρφο τρόπο πάνω στον κύκλο.

Ο κώδικας σε $Matlab$ που περιγράφει την παραπάνω συνάρτηση είναι ο εξής:

```
function [X] = bits_to_PSK_8(bit_seq)

% This function takes as input a sequence of bits, and converts it into
% a sequence of 8-PSK symbols by using Gray coding. Specifically the
% coding is: 000 -> 001 -> 011 -> 010 -> 110 -> 111 -> 101 -> 100
% position :  0  ->  1  ->  2  ->  3  ->  4  ->  5  ->  6  ->  7
% Input: The bit sequence that is coded.
% Output: The 8-PSK sequence

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

for k = 1: 3: size(bit_seq)
    % The index in the symbol sequence.
    index = (k-1)/3 +1;

    % Check bits by 3, and create symbols as the the Gray code implies.
    if(bit_seq(k) == 0 && bit_seq(k+1) == 0 && bit_seq(k+2) == 0)
        pos = 0;
    elseif(bit_seq(k) == 0 && bit_seq(k+1) == 0 && bit_seq(k+2) == 1)
        pos = 1;
    elseif(bit_seq(k) == 0 && bit_seq(k+1) == 1 && bit_seq(k+2) == 1)
        pos = 2;
    elseif(bit_seq(k) == 0 && bit_seq(k+1) == 1 && bit_seq(k+2) == 0)
        pos = 3;
    elseif(bit_seq(k) == 1 && bit_seq(k+1) == 1 && bit_seq(k+2) == 0)
        pos = 4;
    elseif(bit_seq(k) == 1 && bit_seq(k+1) == 1 && bit_seq(k+2) == 1)
        pos = 5;
    elseif(bit_seq(k) == 1 && bit_seq(k+1) == 0 && bit_seq(k+2) == 1)
        pos = 6;
    elseif(bit_seq(k) == 1 && bit_seq(k+1) == 0 && bit_seq(k+2) == 0)
```

```
        pos = 7;
    end
    X(index,1) = cos(2*pi*pos/8);
    X(index,2) = sin(2*pi*pos/8);
end
end
```

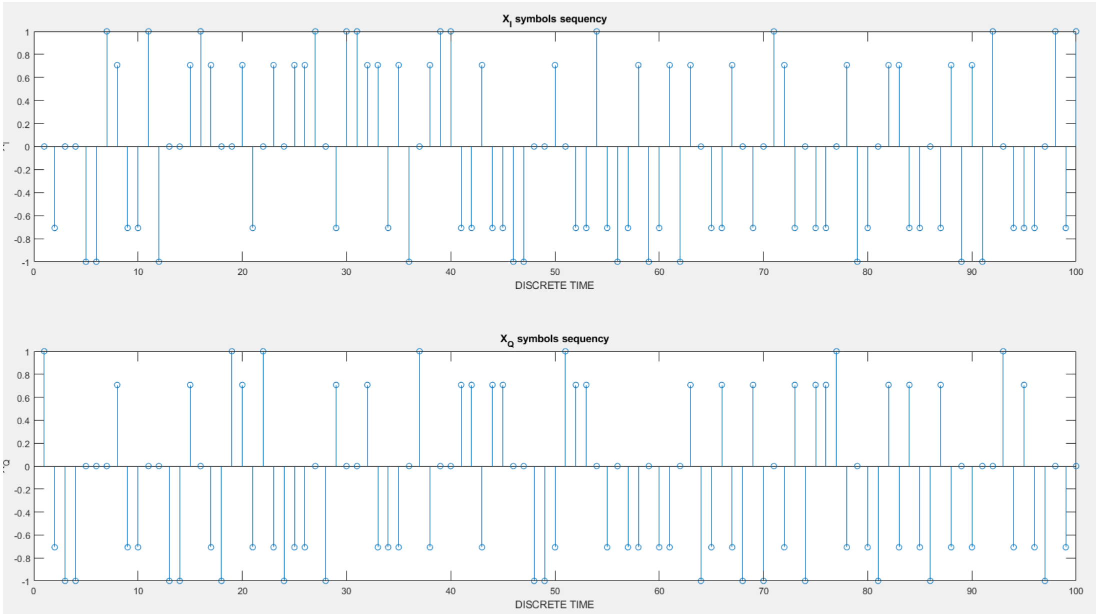


Figure 2: The generated sequences.

A.3

Στην συνέχεια, οι ακολουθίες συμβόλων που παρήχθησαν στο προηγούμενο ερώτημα ($X_{I,n}$ και $X_{Q,n}$), χρησιμοποιούνται για δημιουργία παλμοσειρών της μορφής:

$$X_{\delta}(t) = \sum_{n=0}^{N-1} X_n \, \delta(t - nT),$$

Έπειτα, οι παλμοσειρές αυτές περνούνται από τα $SRRC$ φίλτρα μορφοποίησης (με $h(t) = \phi(t)$). Τα φίλτρα αυτά χρησιμοποιούνται όπως έχουμε δεί και στις προηγούμενες δύο εργασίες του μαθήματος, οπότε δεν θα γίνει κάποια περεταίρω ανάλυση πάνω σε αυτό. Αρκεί να αναφερθούν τα ορίσματα που χρησιμοποιούνται,για την δημιουργία του αποκομμένου $SRRC$ παλμού, τα οποία είναι τα εξής: $T = 10^{-3}$ sec, $over = 10$, $T_s = \frac{T}{over} = 10^{-4}$, όπως δίνονται από εκφώνηση, ενώ το $A = 4$, και $\alpha = 0.5$, τα οποία επιλέχθηκαν εμπειρικώς.

ο κώδικας που πραγματοποιεί την παραπάνω διαδικασία φαίνεται παρακάτω:

```
%A.3

%intialization of symbol period T,oversampling factor over,
%parameter A(half duration of the pulse)  and roll—off factor
T=10^—3;
over=10;
A=4;
a=0.5;
%creating srrc pulses
[phi,t]=srrc_pulse(T,over,A,a);

%intialization of sampling period Ts,sampling frequency Fs,
%parameter Nf(number of equidistant points) and F axis
Ts=T/over;
Fs=1/Ts;
Nf=2048;
F=[—Fs/2:Fs/Nf:Fs/2—Fs/Nf];

%time axis of Xi and Xq
t1=(0:Ts:N*T—Ts);

%Creation of continious Xi and Xq:

%creation of X_delta_i and X_delta_q
X_delta_i=1/Ts*upsample(Xi,over);
X_delta_q=1/Ts*upsample(Xq,over);
%creation of Xic=sum(deltai*phi(t—n*T)) and Xiq=sum(deltaq*phi(t—n*T))
%convolution of phi and X_delta_i and X_delta_q
tconv=[t(1)+t1(1):Ts:t(end)+t1(end)];
Xic=conv(phi,X_delta_i)*Ts;
Xqc=conv(phi,X_delta_q)*Ts;

%creation of the graph of XIc and XQc symbols sequencies in continious time in
figure();
subplot(2,1,1)
plot(tconv,Xic)
```

```
title('X_Ic symbols sequency');
xlabel('CONTINIOUS TIME');
ylabel('X_I');
subplot(2,1,2)
plot(tconv,Xqc)
title('X_Qc symbols sequency');
xlabel('CONTINIOUS TIME');
ylabel('X_Q')
```

Ενώ στην έξοδο του φίλτρου λαμβάνουμε:

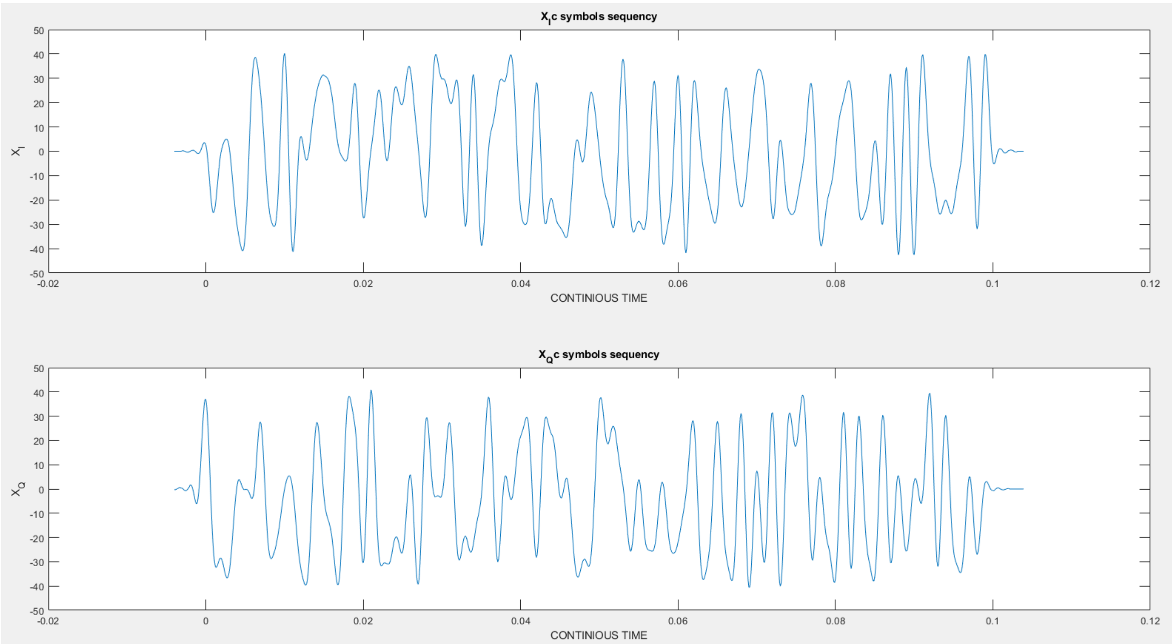


Figure 3: The waveform that represents the symbol train pulse after it passes through a filter with $h(t) = \phi(t) = SRRC$

Στην συνέχεια, υπολογίζονται τα περιοδόγραμματα των 2 δύο συνιστωσών χρησιμοποιώντας την προσεγγιστική μέθοδο αφού πρώτα υπολογιστεί ο μετασχηματισμός $Fourier$ τους:

$$P_X(F) = \frac{\left|X(F)\right|^2}{Ttotal},$$

Έτσι προκύπτουν τα πρακάτω:

```
%A3.2
%creating fast fourier transformation of Xic and Xqc
XIC=fftshift(fft(Xic,Nf)*Ts);
XQC=fftshift(fft(Xqc,Nf)*Ts);
%the total duration time
Ttotal=length(tconv)*Ts;

%creation of a periodogram of one of the implementations of XIC and XIQ
P_xic=abs(XIC).^2/Ttotal;
P_xqc=abs(XQC).^2/Ttotal;

%creation of periodogram graph of XQc and XIc in logarithmic y axis
figure();
subplot(2,1,1)
semilogy(F,P_xic)
title('Periodogram of X_Ic ');
xlabel('F=frequency in Hz');
ylabel('P(X_Ic)');
subplot(2,1,2)
semilogy(F,P_xqc)
title('Periodogram of X_Qc ');
xlabel('F=frequency in Hz');
ylabel('P(X_Qc)');
```

παράγοντας την ακόλουθη απεικόνιση:

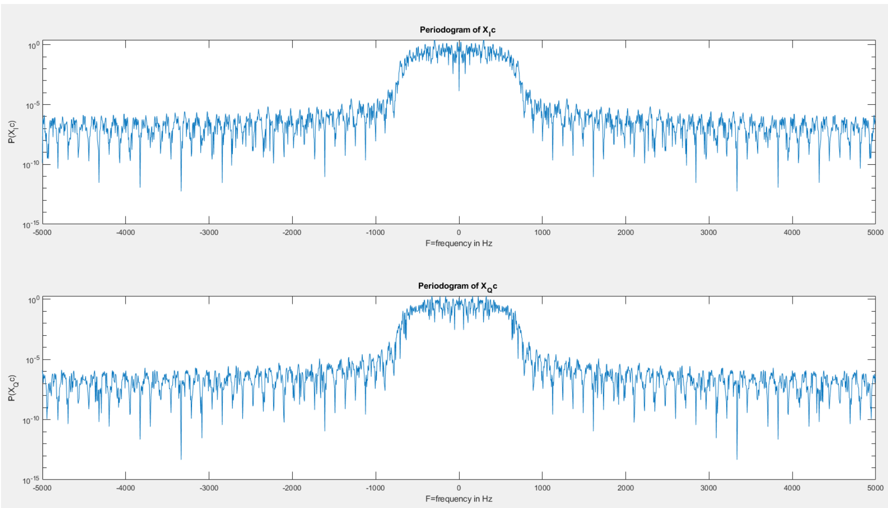


Figure 4: The periodogram of X_I and X_Q

A.4

Στο ερώτημα αυτό πολλαπλασιάζονται οι παραγώμενες κυματομορφές του προηγούμενου ερωτήματος με τους αντίστοιχους φορείς,$(2\cos(2\pi F_0 t)$ και $-2\sin(2\pi F_0 t)$ με αποτέλεσμα την μεταφορά τους στην ζώνη διέλευσης. Με βάση λοιπόν, τον μετασχηματισμό $Fourier$ των ημιτονοειδών φορέων, οι κυματομορφές που προκύπτουν για τα X_I και X_Q θα είναι μετατοπισμένες κατα F_0 , του οποίου η τιμή δίνεται: $F_0 = 2000$, ενώ το πλάτος του θα μειωθεί στο μισό. Εξού και το πλάτος των φορέων είναι: $A = 2$ ώστε, μετά την διαμόρφωση, να μήν έχουμε μείωση του πλάτους των κυματομορφών $X_{I,\delta}$ και $X_{Q,\delta}$.

ο κώδικας κατώθεν πραγματοποιεί όσα αναφέρθηκαν παραπάνω:

```
%A4

%initialization of carrier frequency Fo
Fo=2000;
%creation of the carrier
z1=2*cos(2*pi*Fo*tconv);
z2=-2*sin(2*pi*Fo*tconv);
%creation of modulated signals XIt and XQt
Xit=Xic.*z1;
Xqt=Xqc.*(z2);

%creation of the graphs of the modulated signals XIt and XQt
figure();
subplot(2,1,1)
plot(tconv, Xit)
title('XIc modulated by carrier cos');
xlabel('t=time in seconds');
ylabel('X_It');
subplot(2,1,2)
plot(tconv, Xqt)
title('XQc modulated by carrier -sin');
xlabel('t=time in seconds');
ylabel('X_Qt');

%creating fast fourier transformation of Xit and Xqt
XIT=fftshift(fft(Xit,Nf)*Ts);
XQT=fftshift(fft(Xqt,Nf)*Ts);
%the total duration time
```

```
Ttotal=length(tconv)*Ts;

%creation of a periodogram of one of the implementations of XIt and XQt
P_xit=abs(XIT).^2/Ttotal;
P_xqt=abs(XQT).^2/Ttotal;

%creation of periodogram graph of XQt and XIt in logarithmic y axis
figure();
subplot(2,1,1)
semilogy(F,P_xit)
title('periodogramm of X_It ');
xlabel('F=frequency in HZ');
ylabel('P(X_It)');
subplot(2,1,2)
semilogy(F,P_xqt)
title('periodogramm of X_Qt ');
xlabel('F=frequency in HZ');
ylabel('P(X_Qt)');
```

Ενώ η διαμορφωμένη κυματομορφή φαίνεται παρακάτω:

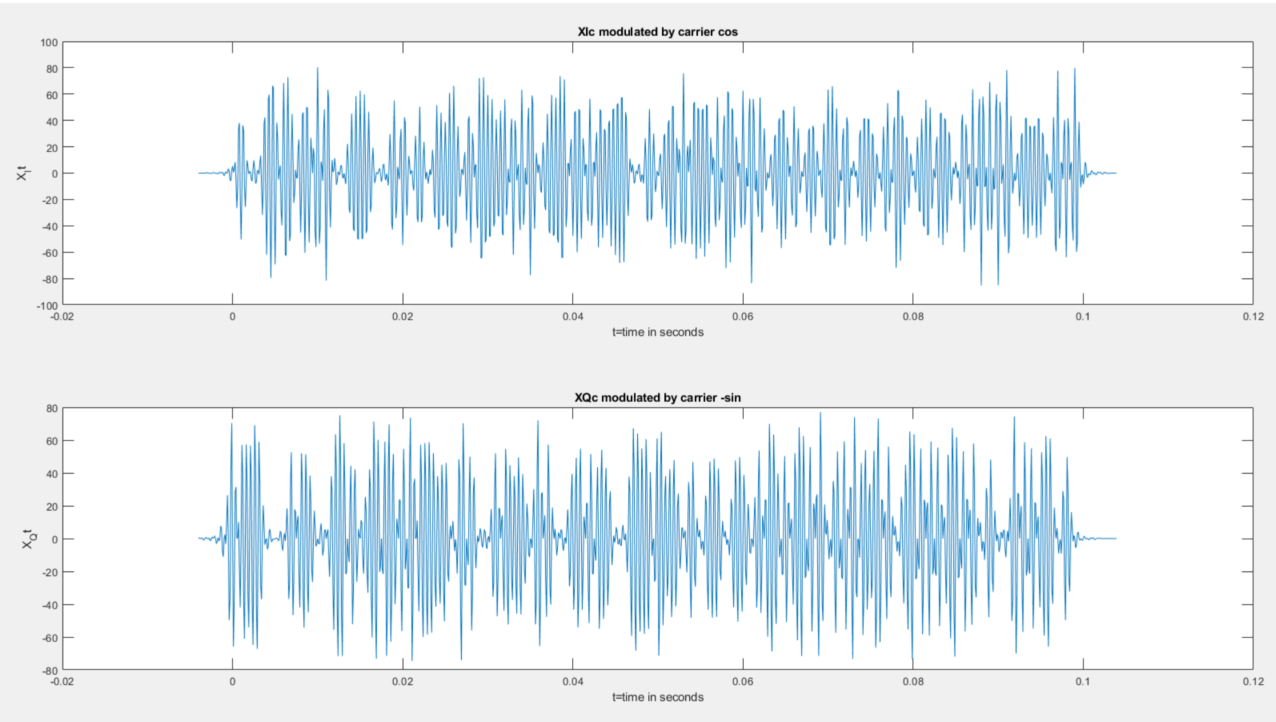


Figure 5: The modulated waveform.

Έπειτα υπολογίζονται τα περιοδογράμματα όπως και στο ερώτημα 3.

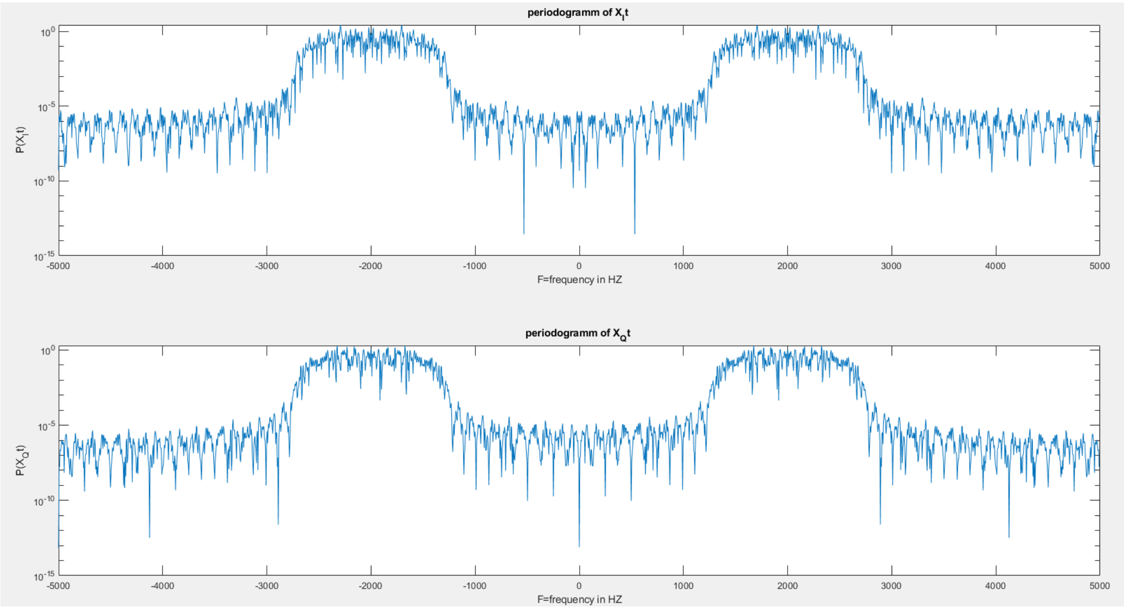


Figure 6: The modulated waveform.

Τα περιοδογράμματα γίνεται εύκολα αντιληπτό ότι πλέον, έχουν μεταφερθεί γύρω απο τη συχνότητα διαμόρφωσης.

A.5

Η είσοδος που θα επιβληθεί τελικά στο κανάλι μας θα είναι το άθροισμα των δυο συνιστοσών X_I και X_Q. το οποίο προκύπτει ώς εξής:

```
%A.5,6

%creation of the channel's input X(t)
X_t=Xit+Xqt;

%creation of the inputs graph
figure();
plot(tconv,X_t)
title('channels input WITHOUT NOISE FACTOR ');
```

```
xlabel('t=time in seconds');
ylabel('X(t)');

%creating fast fourier transformation of Xic and Xqc
XF=fftshift(fft(X_t,Nf)*Ts);
%the total duration time
Ttotal=length(tconv)*Ts;
%creation of a periodogram of one of the implementations of X(t)
P_x=abs(XF).^2/Ttotal;

%creation of periodogram graph of X(t) in logarithmic y axis
figure();
semilogy(F,P_x)
title('periodogramm of X(t) ');
xlabel('F=frequency in HZ');
ylabel('P(X)');
```

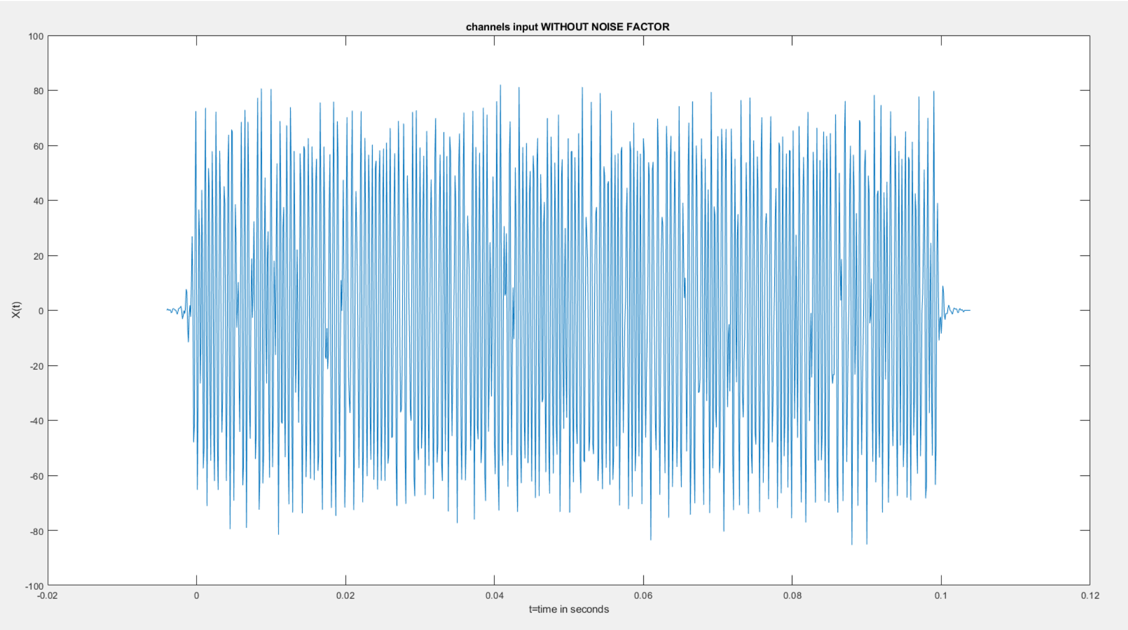


Figure 7: The modulated waveform.

Και το περιοδόγραμμα:

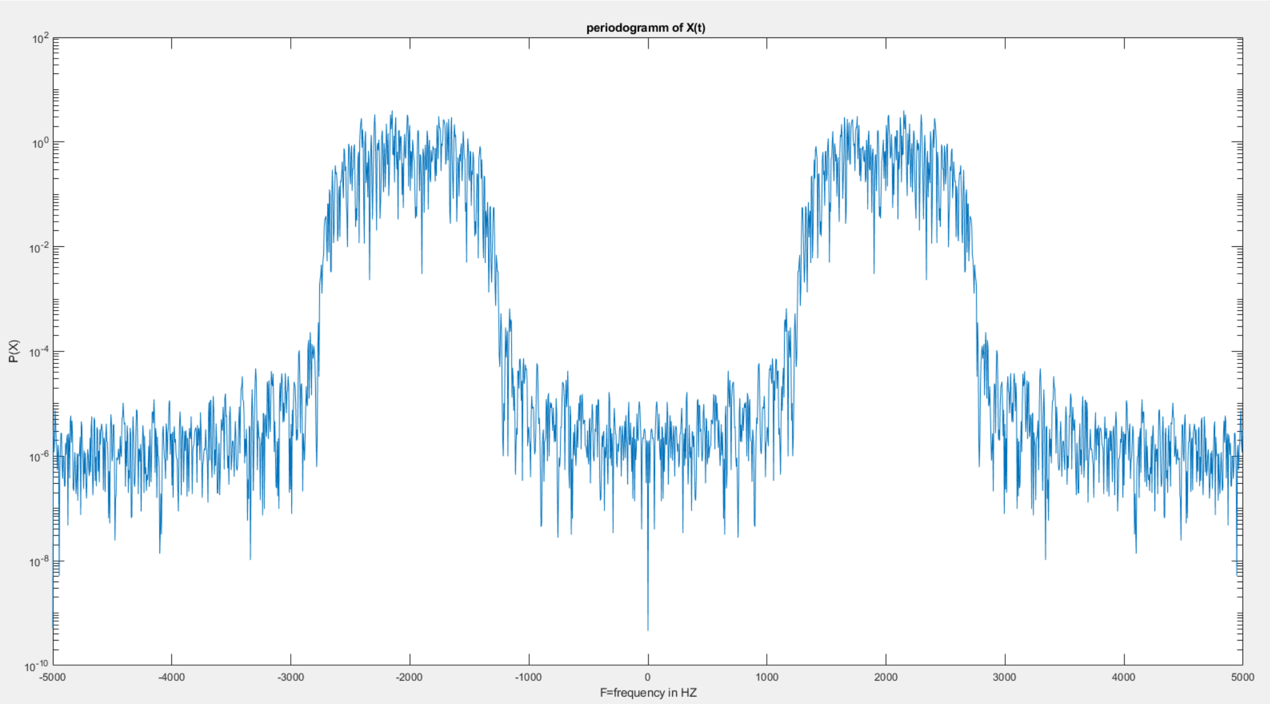


Figure 8: The modulated waveform.

Παρατηρώντας την παραπάνω απεικόνιση αντιλαμβανόμαστε ότι το πλάτος της εισόδου αυξάνεται σημαντικά,θα μπορούσαμε να πούμε ότι διπλασιάζεται, το οποίο είναι αναμενόμενο, καθώς αθροίσαμε δύο συνιστώσες ίσου πλάτους. Όσον αφορά την συχνότητα, εφόσον, τόσο η $In - phase$ συνιστώσα $X_{I,n}$ όσο και η $Quadrature - phase$ $X_{Q,n}$ ενεργούν με κέντρο τις συχνότητες $-F_0 = -2000$ και $F_0 = 2000$, το ίδιο ισχύει και για το άθροισμά τους.

A.6

Εφόσον δεν υπάρχει θόρυβος να επηρεάζει την αποδοτικότητα του συστήματος, οδηγούμαστε στο συμπέρασμα ότι το σήμα εξόδου της διαδικασίας, αφενός, δεν θα δεν θα έχει υποστεί καθυστέρηση, και αφετέρου ότι το πλάτος του δεν θα έχει ενισχυθεί. Συνεπώς η ανάκτηση του σήματος θα είναι εύκολη και η πιθανότητα σφάλματος ελάχιστη.

A.7

Στην σημείο αυτό, προσεγγίζουμε λίγο καλύτερα τις πραγματικές συνθήκες επικοινωνίας εφαρμόζοντας στο σήμα μια *Gaussian* συνιστώσα θορύβου: $W(t)$, από το κανάλι. Παρόλο που ο λευκός θόρυβος δεν συναντάται στην φύση, η μελέτη του συστήματος στις συνθήκες αυτές, μας δίνει σημαντική πληροφορία για την συμπεριφορά του. Στην προκειμένη περίπτωση η διασπορά του θορύβου θα δίνεται από την σχέση:

$$\sigma_W^2 = \frac{1}{T_s \cdot 10^{\frac{SNR_{dB}}{10}}}$$

Τελικά λοιπόν, το σήμα που θα βρίσκεται στην έξοδο του καναλιού θα είναι το ενθόρυβο:

$$Y(t) = X(t) + W(t)$$

```
%A.7

%insertion of white gaussian noise
SNR_DB=10;
s_w=1/(Ts*10^(SNR_DB/10));
s_n = Ts*s_w/2;
%Random numbers with specific variance at the length of transmitted signal X_t
w = sqrt(s_n).*randn(1, length(X_t));
Y_t = X_t + w;

%creation of X_t with noise factor
figure();
plot(tconv,Y_t)
title('channels input WITH NOISE FACTOR ');
xlabel('t=time in seconds');
ylabel('X(t) with noise');

%creation of the difference of X_t with and without noise
diff=Y_t-X_t;
%creation of the difference graph
figure();
plot(tconv,diff)
title('difference of the input signals with and without noise');
xlabel('t=time in seconds');
ylabel('X(t)with noise-X(t)without noise ');
```

Η ενθόρυβη κυματομορφή που προκύπτει θα είναι:

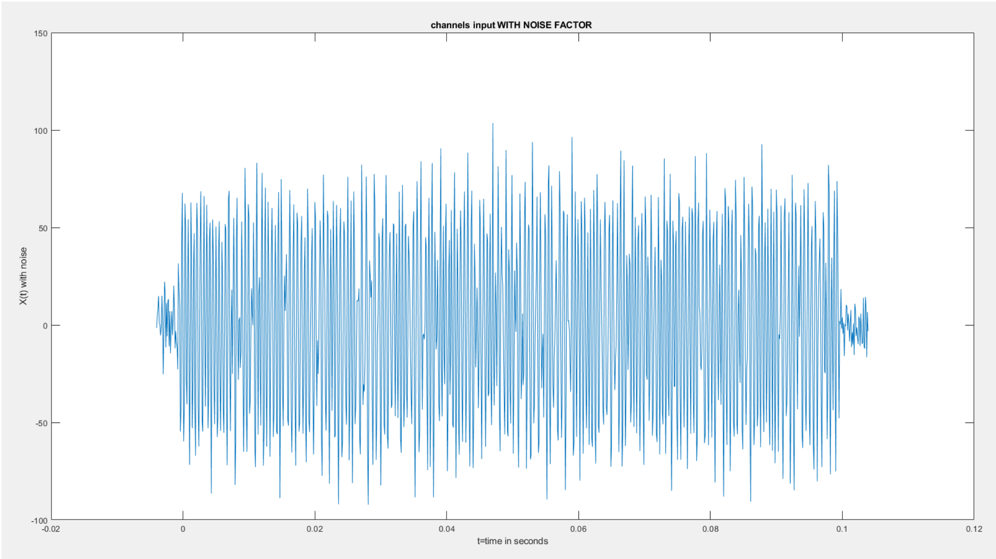


Figure 9: The waveform of the signal that arrives at the receiver

Για την σύγκριση των δύο κυματομορών, με και χωρίς θόρυβο, υπολογίστηκε η διαφορά τους, ώστε να φανούν καλύτερα οι διαφορές τους. Συγκεκριμένα:

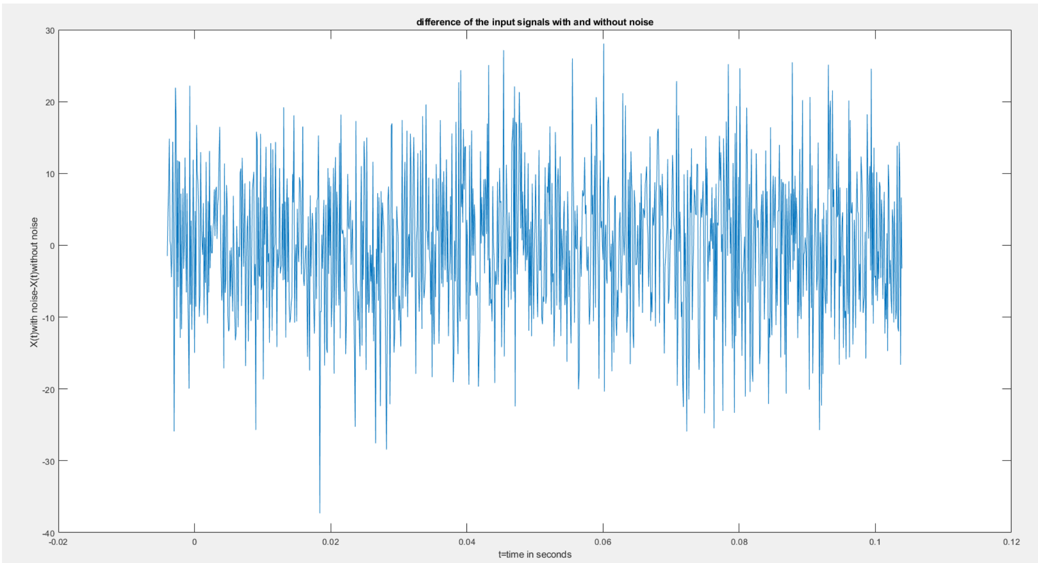


Figure 10: The difference between the waveform of the signal that arrives at the receiver with noise, and without noise.

A.8

Εφόσον το σήμα έχει αποχωρήσει από το πομπό, και σε αυτό έχει προστεθεί η συνιστώσα λευκού θορύβου, από το κανάλι μετάδοσης, καταλήγει τελικά στον δέκτη ο οποίος καλείται να αποκωδικοποιήση την πληροφορία. Για τον λόγο αυτό θα εφαρμοστεί η αντίστροφη διαδικασία αποδιαμόρφωσης. Συγκεκριμένα το σήμα που λαμβάνεται στο δέκτη πολλαπλασιάζεται με τους φορείς $\cos(2\pi F_0 t)$ και $-\sin(2\pi F_0 t)$. Αξίζει να σημειωθεί ότι το πλάτος του φορέα στην φάση αυτή θα είναι ίσο με την μονάδα, σε αντίθεση με τους φορείς, κατά την διαμόρφωση οι οποίοι είχα $A = 2$. Αυτό συμβαίνει καθώς επιστρέφοντας στην βασική ζώνη, τα ¨τμήματα' του σήματος που είχαν διασπαστεί στις συχνότητες $-F_0 = -2000Hz$ και $F_0 = 2000Hz$, θα αθροιστούν ξανά. Οπότε ζητούμενο είναι το πλάτος κάθε μίας από αυτές τις δύο συνιστώσες να είναι $\frac{A}{2}$ ώστε $A = 2\frac{A}{2}$.

Ο κώδικας που πραγματοποιεί την διαδικασία της αποδιαμόρδφωσης φαίνεται παρακάτω:

```
%A.8

%creation of remodulated signals Yz1 and Yz2
Yz1=Y_t.*cos(2*pi*Fo*tconv);
Yz2=Y_t.*(-sin(2*pi*Fo*tconv));

%creation of ther graph of Yz1 and Yz2
figure();
subplot(2,1,1)
plot(tconv,Yz1)
title('Y_t remodulated by carrier cos ');
xlabel('t=time in seconds');
ylabel('Yz1');
subplot(2,1,2)
plot(tconv,Yz2)
title('Y_t remodulated by carrier -sin');
xlabel('t=time in seconds');
ylabel('Yz2');

%creating fast fourier transformation of Yz1 and Yz2
YZ1=fftshift(fft(Yz1,Nf)*Ts);
YZ2=fftshift(fft(Yz2,Nf)*Ts);
%the total duration time
Ttotal=length(tconv)*Ts;
%creation of a periodogram of one of the implementations of Yz1 and Yz2
```

```
P_yz1=abs(YZ1).^2/Ttotal;
P_yz2=abs(YZ2).^2/Ttotal;

%creation of the graph of periodogramms of YZ1 and YZ2
figure();
subplot(2,1,1)
semilogy(F,P_yz1)
title('periodogramm of the  Y_t remodulated by carrier cos ');
xlabel('F=frequency in HZ');
ylabel('P(YZ1)');
subplot(2,1,2)
semilogy(F,P_yz2)
title('periodogramm of the Y_t remodulated by carrier -sin ');
xlabel('F=frequency in HZ');
ylabel('P(YZ2)');
```

τα αποτελέσματα είναι τα ακόλουθα:

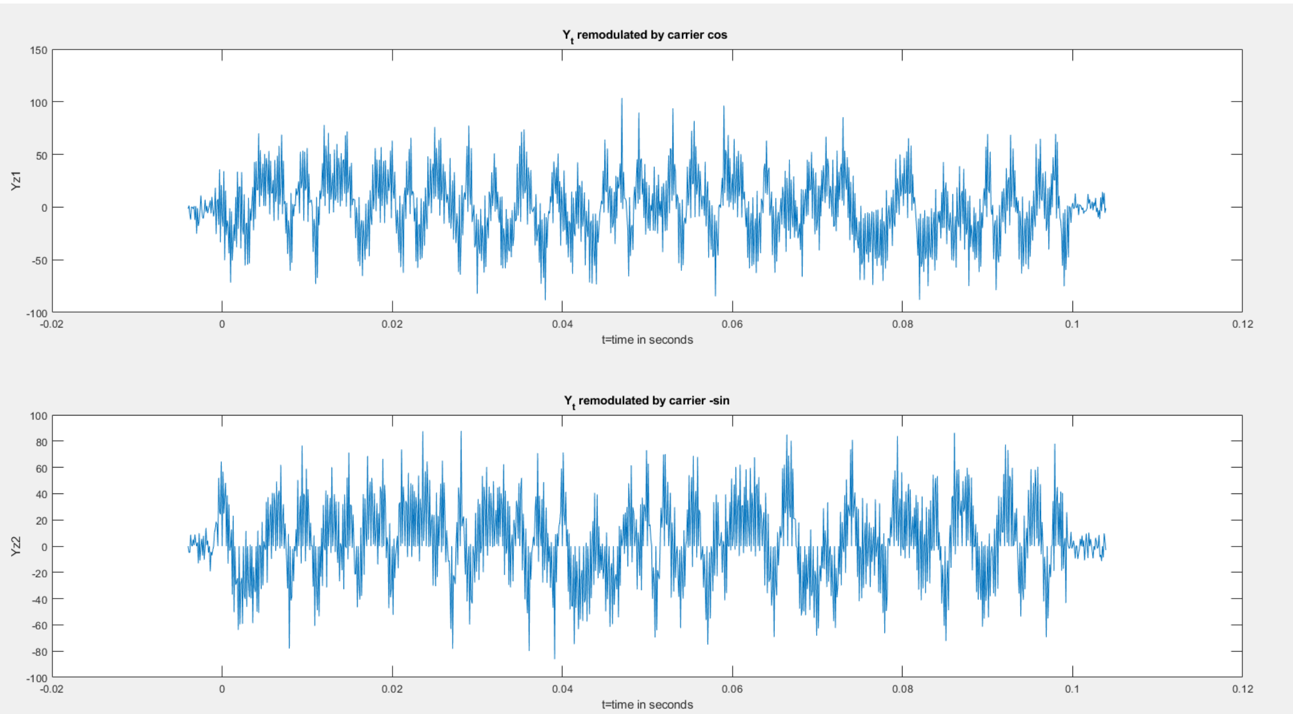


Figure 11: The remodulated Y_{it} and Y_{qt}

Αντίστοιχα τα περιοδογράμματα που προκύπτουν είναι τα κατώθεν:

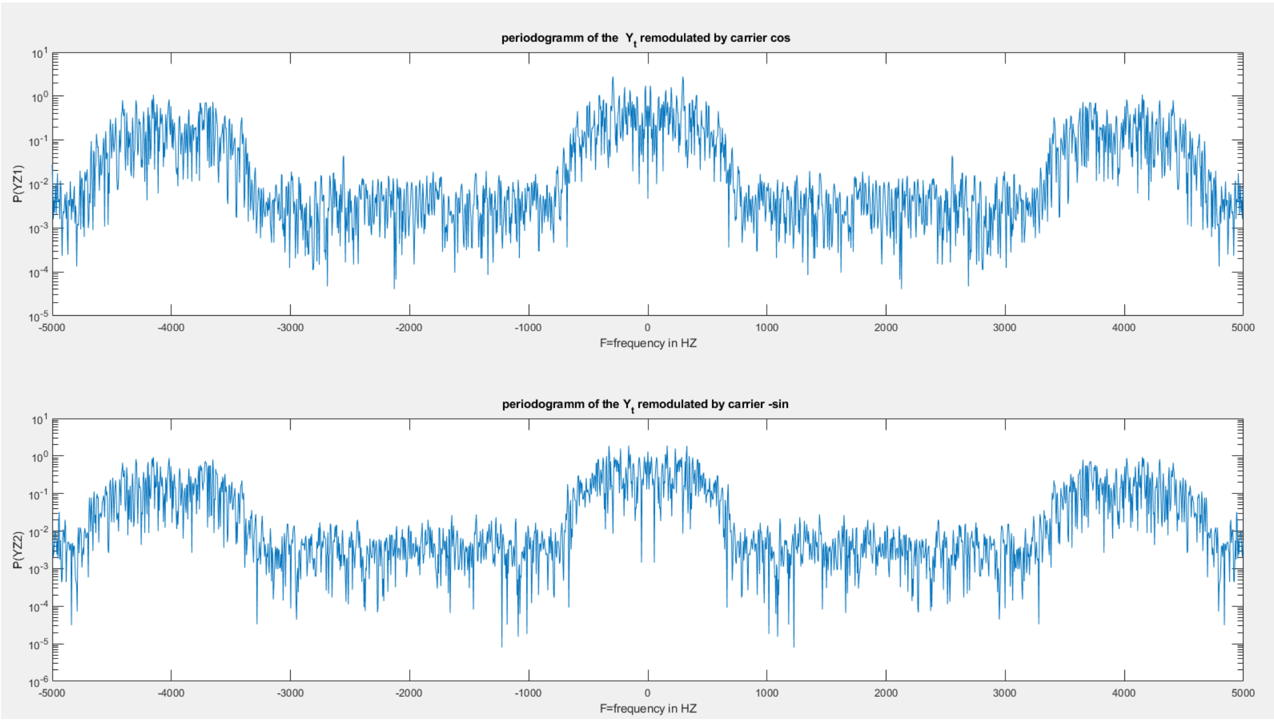


Figure 12: The periodograms of the remodulated Y_{it} and Y_{qt}

A.9

Αφού περαστούν οι ανώθεν κυματομορφές από τα προσαρμοσμένα φίλτρα με απόκριση $h(t) = phi(t)$, λόγω της ορθοκανικότητας της φ ώς προς τις μετατοπίσεις της κατα ακέραια πολλαπλάσια της περιόδου συμβόλων T, λαμβάνεται μια παλμοσειρά που εμπεριέχει τα σύμβολα μαζί με την συνσιστώσα του θορύβου. Οι διαδικασίες αυτές παρουσιάζονται παρακάτω:

```
%A9

%creation of the new time axis
new_tconv=tconv(1)+t(1):Ts:tconv(end)+t(end);

%Yz1new=the new Xit after the filter
Yz1new=conv(Yz1,phi)*Ts;
%Yz2new=the new Xqt after the filter
Yz2new=conv(Yz2,phi)*Ts;

%creation of the new waveforms Yz1new and Yz2new
figure();
subplot(2,1,1)
plot(new_tconv,Yz1new)
title('Yz1 after the filter= Xit*');
xlabel('t=time in seconds');
ylabel('Xit*');
subplot(2,1,2)
plot(new_tconv,Yz2new)
title('Yz2 after the filter= Xqt*');
xlabel('t=time in seconds');
ylabel('Xqt*');

%creating fast fourier transformation of Yz1new and Yz2new
YZ1NEW=fftshift(fft(Yz1new,Nf)*Ts);
YZ2NEW=fftshift(fft(Yz2new,Nf)*Ts);
%the total duration time
Ttotal=length(tconv)*Ts;
%creation of a periodogram of one of the implementations of Yz1 and Yz2
P_yz1new=abs(YZ1NEW).^2/Ttotal;
P_yz2new=abs(YZ2NEW).^2/Ttotal;

```

```
%creation of the graph of periodogramms of YZ1NEW and YZ2NEW
figure();
subplot(2,1,1)
semilogy(F,P_yz1new)
title('periodogramm of Xit* ');
xlabel('F=frequency in HZ');
ylabel('P(XIF*)');
subplot(2,1,2)
semilogy(F,P_yz2new)
title('periodogramm of Xqt* ');
xlabel('F=frequency in HZ');
ylabel('P(XQF*)');
```

προκύπτουν λοιπόν τα ακόλουθα:

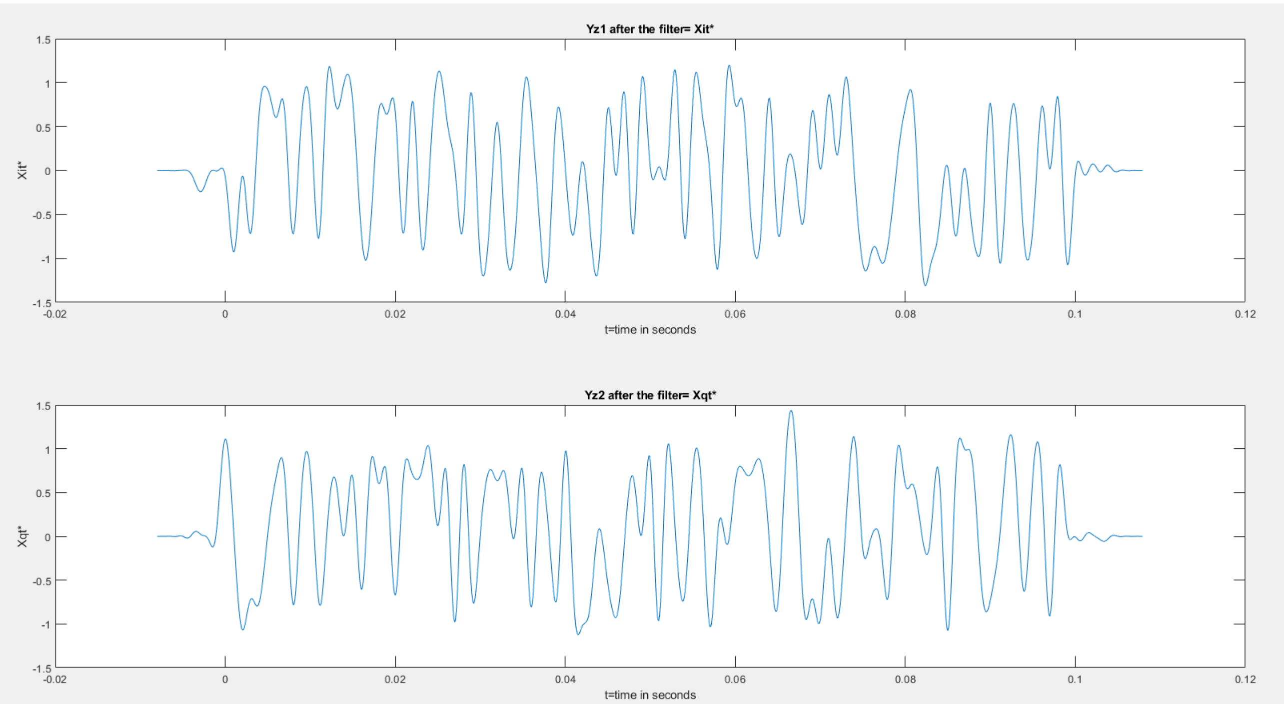


Figure 13: The waveform that comes out of the batched filter for Y_i and Y_q

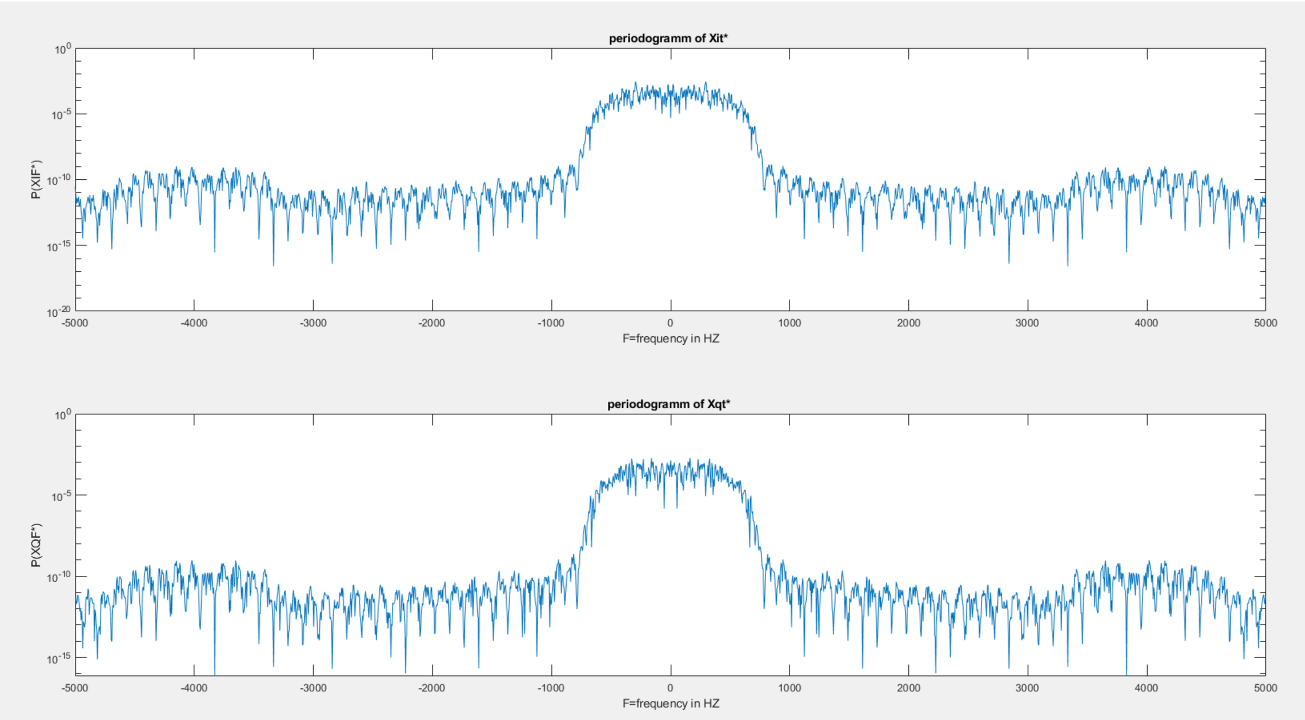


Figure 14: The periodogram of the signals that come out of the batched filter for Y_i and Y_q

Παρατηρώντας, πλέον την φιλτραρισμένη κυματομορφή, βλέπουμε ότι οι πλάγιοι λοβοί που απαντώνταν στην προηγούμενη κυματομορφή των περιοδογραμμάτων, περιορίστηκαν σημαντικά, με αποτέλεσμα πλέον οι παραμορφώσεις που προκαλούσαν να μπορούν πλέον να θεωρηθούν αμελητέες.

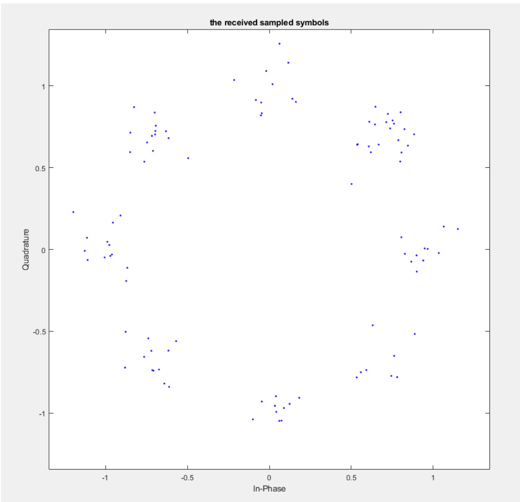
A.10

Σε τελική ανάλυση θα πρέπει να γίνει ανάκτηση των αρχικών συμβόλων, οπότε θα πραγματοποιηθεί δειγματοληψία, παίρνοντας όσα δείγματα όσα και τα σύμβολα μας. Συνεπώς θα λαμβάνουμε 1 ένα δείγμα κατα ακέραια πολλαπλάσια του T. Έτσι η ακολουθία συμβόλων που προκύπτει καταληκτικά είναι:

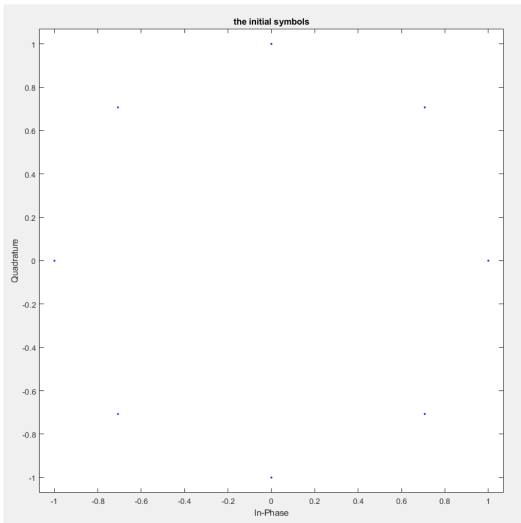
```
%A10

%sampling of the output of the filters , Xit* and Xqt*
n=0:N−1;
yi(n+1)=Yz1new(n*over + 80);
yq(n+1)=Yz2new(n*over + 80);

%creation of the new sampled signals with scatter graph
figure();
scatter(yi,yq)
title("the received sampled symbols")
```



(a) The output sequence Y in using scatterplot



(b) The initial sequence X in using scatterplot

A.11

Στο ερώτημα αυτό, δημιουργείται μια συνάρτηση $detect\_PSK\_8(Y)$ της οποίας η λειτουργία είναι διμερής. Συγκεκριμένα, σε πρώτη φάση η συνάρτηση αυτή θα πρέπει αξιοποιώντας τον κανόνα του εγγύτερου γείτονα θα αποφαίνεται για το ποίο θα θεωρείται ότι ήταν το σύμβολο της εισόδου(est_X). Η υλοποίηση της λειτουργικότητας αυτής θα μπορούσαμε να πούμε ότι είναι ιδιαίτερα απλή. Με άλλα λόγια, η μόνη λειτουργία που εκτελείται είναι η σύγκριση της απόστασης ενός στοιχείου του διανύμσατος εξόδου Y με όλες τις πιθανές του τοποθετήσεις πάνω στον μιγαδικό κύκλο. Η τοποθέτηση εκείνη η οποία παρουσιάζει την ελάχιστη απόσταση από το εξεταζόμενο στοιχείο, είναι εκείνη που θα θεωρήσουμε ότι στεγάζει το δίανυσμα εισόδου. Συγκεκριμένα, ο κώδικας που επιτελεί την λειτουργικότητα αυτή φαίνεται παρακάτω:

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

% This function takes the sequence that reaches the receiver, and tries to
% determine what the initial sequence of symbols was, and subsequently,
% what was the initial bit sequence.
% INPUT:
% Y: The sequence at the receiver
% OUTPUT:
% est_X: The estimated sequence of symbols the transmitter sent
% est_bit_seq: The estimated sequence of bits the transmitter sent

% Initializations
len_Y = length(Y);
est_X = zeros(len_Y,2);
est_bit_seq = zeros(3*len_Y,1);
len_Y = length(Y);


% All the possible positions on the complex circle:
zero = [cos(2*pi*0/8), sin(2*pi*0/8)];
one = [cos(2*pi*1/8), sin(2*pi*1/8)];
two = [cos(2*pi*2/8), sin(2*pi*2/8)];
three = [cos(2*pi*3/8), sin(2*pi*3/8)];
four = [cos(2*pi*4/8), sin(2*pi*4/8)];
five = [cos(2*pi*5/8), sin(2*pi*5/8)];
six = [cos(2*pi*6/8), sin(2*pi*6/8)];
seven = [cos(2*pi*7/8), sin(2*pi*7/8)];

positions = [zero; one; two; three; four; five; six; seven];
len_pos = length(positions);

for i=1: len_Y
    best_est = zero;
    min_dist = 100000;
    best_est_index = 0;

    for j=1: len_pos
```

```
        if(norm(Y(i,:)-positions(j,:),2) < min_dist)
            min_dist = norm(Y(i,:)-positions(j,:),2);
            best_est = positions(j,:);
            best_est_index = j;
        end
    end
    est_X(i,1) = best_est(1);
    est_X(i,2) = best_est(2);

    % Now the bit sequence for the specific symbol is generated
    % position:  0  -> 1  -> 2  -> 3  -> 4  -> 5  -> 6  -> 7
    % in index:  1  -> 2  -> 3  -> 4  -> 5  -> 6  -> 7  -> 8
    % bits    : 000 -> 001 -> 011 -> 010 -> 110 -> 111 -> 101 -> 100

    % The index in the symbol sequence.
    index = 3*(i-1)+1;
    if(best_est_index == 1)
       est_bit_seq(index) = 0;
       est_bit_seq(index+1) = 0;
       est_bit_seq(index+2) = 0;
    elseif(best_est_index == 2)
       est_bit_seq(index) = 0;
       est_bit_seq(index+1) = 0;
       est_bit_seq(index+2) = 1;
    elseif(best_est_index == 3)
       est_bit_seq(index) = 0;
       est_bit_seq(index+1) = 1;
       est_bit_seq(index+2) = 1;
    elseif(best_est_index == 4)
       est_bit_seq(index) = 0;
       est_bit_seq(index+1) = 1;
       est_bit_seq(index+2) = 0;
    elseif(best_est_index == 5)
       est_bit_seq(index) = 1;
       est_bit_seq(index+1) = 1;
       est_bit_seq(index+2) = 0;
    elseif(best_est_index == 6)
       est_bit_seq(index) = 1;
       est_bit_seq(index+1) = 1;
       est_bit_seq(index+2) = 1;
    elseif(best_est_index == 7)
       est_bit_seq(index) = 1;
       est_bit_seq(index+1) = 0;
```

```
            est_bit_seq(index+2) = 1;
        elseif(best_est_index == 8)
            est_bit_seq(index) = 1;
            est_bit_seq(index+1) = 0;
            est_bit_seq(index+2) = 0;
        end
    end
end
```

Η υλοποίηση του δεύτερου τμήματος της συνάρτησης είναι εξίσου απλή, καθώς, απλώς παράγει την αντίστροφη απεικόνιση του κώδικα $Gray$, δήλαδή $1symbol \rightarrow 3bits$ απεικόνιση, μεταφράζοντας τα σύμβολα που υπολογίστηκαν όπως φαίνεται παραπάνω.

A.12

Έπειτα παράγεται μια ακόμη συνάρτηση: $symbol_errors(est\_X, X)$, η οποία θα λαμβάνει ώς είσοδο την αρχική ακολουθία συμβόλων, και την εκτιμόμενη ακολουθία, όπως αυτή υπολογίστηκε στο προηγούμενο ερώτημα, και επιστρέφει τον αριθμό των σφαλμάτων. Αυτό γίνεται ελέγχοντας επαναληπτικά τα στοιχεία των ακολουθιών, και εφόσον αυτά διαφέρουν, αυξάνουμε τον μετρητή. Συγκεκριμένα:

```
function [num_of_symbol_errors] = symbol_errors(est_X,X)

% This function takes as input the sequence that reaches the receiver,
% and the one that the transmitter sent, and attempts to find the number
% of mistakes that have occured in the process.
% INPUT:
% est_X: The estimated sequence of symbols, the transmitter propably sent
% X    : The real sequence of symbols, sent by the transmitter.
% OUTPUT:
% num_of_symbol_errors: The number of errors that happened.

% Initializations
num_of_symbol_errors = 0;
len_X = length(X);

for i=1: len_X
  if(est_X(i) ~= X(i))
```

```
    num_of_symbol_errors = num_of_symbol_errors + 1;
  end
end
```

A.13

Η τελευταία συνάρτηση $bit\_errors(est_{b}it_{s}eq, b)$, λαμβάνει ώς είσοδο την αρχική ακολουθία $bits$, και την εκτιμόμενη ακολουθία, όπως αυτή υπολογίστηκε στο ερώτημα A.11, και επιστρέφει τον αριθμό των σφαλμάτων. Αυτό γίνεται ελέγχοντας επαναληπτικά τα στοιχεία των ακολουθιών ανα τριάδες, και εφόσον αυτά διαφέρουν, αυξάνουμε τον μετρητή. Συγκεκριμένα:

```
function [num_of_bit_errors] = bit_errors(est_bit_seq,bit_seq)

% ?his function takes as input the sequence of bits that reaches the receiver,
% and the one that the transmitter sent, and attempts to find the number
% of errors that have occured in the process.
% INPUT:
% est_bit_seq: The estimated sequence of bits, the transmitter propably sent
% bit_seq    : The real sequence of bits, sent by the transmitter.
% OUTPUT:
% num_of_bit_errors: The number of errors that happened.

% Initializations
num_of_bit_errors = 0;
len_bit_seq = length(bit_seq);

for i=1:len_bit_seq
  if(est_bit_seq(i) ~= bit_seq(i))
    num_of_bit_errors = num_of_bit_errors + 1;
  end
end
```

Θέμα Β

Β.1

Για το παρακάτω ερώτημα ,μας ζητήθηκε να εκτιμηθεί η πιθανότητα σφάλματος συμβόλου και *bit* μέσω πειράματος ανεξάρτητων επαναλήψεων , με χρήση της μεθόδου *Monte Carlo*. Πραγματοποιήθηκαν λοιπόν τα βήματα της προσομοίωσης του τηλεπικοινωνιακού σήματος που απεικονίζεται στο σχήμα , τα οποία αναλύθηκαν παραπάνω σε πλήθος 1000 ανεξάρτητων επαναλήψεων για κάθε SNR από [-2,16] με βήμα 2. Το παραπάνω πραγματοποιήθηκε με δύο εμφωλευμένους βρόγχους επανάληψης για όλο το πλήθος των επαναλήψεων και για όλες τις δυνατές τιμές του διανύσματος SNR. Ακόμη, υπολογίστηκε το άθροισμα των συνολικών σφαλμάτων συμβόλου και *bit* και μέσω της μεθόδου *Monte Carlo*, εκτιμήθηκε η πιθανότητα σφάλματος συμβόλου και *bit* εφαρμόζοντας τους παρακάτω τύπους:

$$P(E_{symbol}) = \frac{number\ of\ errors\ in\ symbols}{total\ symbols\ sent}$$

$$P(E_{bit}) = \frac{number\ of\ errors\ in\ bits}{total\ bits\ sent}$$

Ο κώδικας σε *Matlab* που πραγματοποιεί τα παραπάνω φαίνεται κατώθεν:

```
%B
%number of times the process is runed
K=1000;
%SNR in decibels
SNR_db= [-2:2:16];
%the monte carlo probabilities
%initializing the symbol error probability
P_Esymbol=zeros(1,length(SNR_db));
%initializing the bit error probability
P_Ebit=zeros(1,length(SNR_db));
%initializing the upper and lower bounds
upper_bound = zeros(1,length(SNR_db));
lower_bound = zeros(1,length(SNR_db));

for i=1:length(SNR_db)
```

```
%initializing to zero the number of symbol and bit errors
num_of_symbol_errors=0;
num_of_bit_errors=0;

for j=1:K

    %creation of bit sequence
    b=(sign(randn(3*N,1))+1)/2;

    %call of the created function that transfers bits to 8PSK
    X = bits_to_PSK_8(b);
    Xi = transpose(X(:,1));
    Xq = transpose(X(:,2));

    %creating the srrc pulse
    [phi,t]=srrc_pulse(T,over,A,a);

    %time axis of Xi and Xq
    t1=(0:Ts:N*T-Ts);
    %creation of X_delta_i and X_delta_q
    X_delta_i=1/Ts*upsample(Xi,over);
    X_delta_q=1/Ts*upsample(Xq,over);

    %convolution of phi and X_delta_i and X_delta_q
    tconv=[t(1)+t1(1):Ts:t(end)+t1(end)];
    Xic=conv(phi,X_delta_i)*Ts;
    Xqc=conv(phi,X_delta_q)*Ts;

    %creation of the carrier
    z1=2*cos(2*pi*Fo*tconv);
    z2=-2*sin(2*pi*Fo*tconv);
    %creation of modulated signals XIt and XQt
    Xit=Xic.*z1;
    Xqt=Xqc.*(z2);

    %creation of the channel's input X(t)
    X_t=Xit+Xqt;

    %insertion of white gaussian noise
    disper = sqrt(1/(Ts*10^(SNR_db(i)/10)));
    W_t = disper*randn(length(X_t),1);
    Y_t = zeros(1, length(X_t));
```

```
        for p=1 : length(X_t)
         Y_t(p) = X_t(p) + W_t(p);
        end

        %creation of remodulated signals Yz1 and Yz2
        Yz1=Y_t.*cos(2*pi*Fo*tconv);
        Yz2=Y_t.*(-sin(2*pi*Fo*tconv));

        %creation of the new time axis
        new_tconv=tconv(1)+t(1):Ts:tconv(end)+t(end);
        %Yz1new=the new Xit after the filter
        Yz1new=conv(Yz1,phi)*Ts;
        %Yz2new=the new Xqt after the filter
        Yz2new=conv(Yz2,phi)*Ts;

        %sampling of the output of the filters , Xit* and Xqt*
        n=0:N-1;
        yi(n+1)=Yz1new(n*over + 80);
        yq(n+1)=Yz2new(n*over + 80);

        %calling of the detect function
        Y=[transpose(yi),transpose(yq)];
        [est_X, est_bit_seq] = detect_PSK_8(Y);

        %calling of the symbol errors function and counting the total
        %number of symbol errors
        num_of_symbol_errors = num_of_symbol_errors + symbol_errors(est_X,X);

        %calling of the bit errors function and counting the total number
        %of bit errors
        num_of_bit_errors = num_of_bit_errors + bit_errors(est_bit_seq,b);
    end

    %calculating the probabilities of symbol and bit errors with the monte
    %carlo method
    P_Esymbol(1, i) = num_of_symbol_errors/((length(b))*K);
    P_Ebit(1, i) = num_of_bit_errors/(length(b)*K);
    %calcuting the the smart upper bound for every SNR, with the given Q function
    upper_bound(i) = 2*Q(sqrt(2*(10^(SNR_db(1,i)/10)))*sin(pi/8));
    %calculating the lower bound , for every wrong symbol there is at least
    %one wrong bit
    lower_bound(i) = upper_bound(i)/3;
```

```
end

%creation of the graph  of the smart upper bound and the probability of the
%symbol errors in the same plot with logarithmic y axis
figure();
semilogy(SNR_db,P_Esymbol);
hold on;
semilogy(SNR_db, upper_bound);
legend('P(Esymbols)','Smart Upper Bound');
xlabel('SNR_{db}');
ylabel('Probability of symbol error');
title('Graph with logarithmic y axis with the Theoritical and Expeimental Probability Of Symbol Error');
hold off;

%creation of the graph  of the lower bound and the probability of the
%biterrors in the same plot with logarithmic y axis
figure();
semilogy(SNR_db,P_Ebit);
hold on;
semilogy(SNR_db, lower_bound);
legend('P(Ebits)','Lower Bound');
xlabel('SNR_db');
ylabel('Probability of bit error');
title('Graph with logarithmic y axis with the Theoritical and Expeimental Probability Of Bit Error');
hold off;
```

B.2-B.3

Στην συνέχεια, μέσω της δεδομένης συνάρτησης X και χρησιμοποιώντας τον παρακάτω τύπο,

$$P(E) \leq 2Q(\sqrt{2 \cdot SNR} \cdot sin(\frac{\pi}{8}))$$

υπολογίστηκε το έξυπνο άνω φράγμα αντίστοιχα ξανά για κάθε τιμή του διανύσματος SNR.Έπειτα, σχεδιάστηκαν οι θεωρητικές και πειραματικές γραφικές παραστάσεις της πιθανότητας σφάλματος συμβόλου , σε ένα κοινό διάγραμμα με λογαριθμικό κατακόρυφο άξονα.

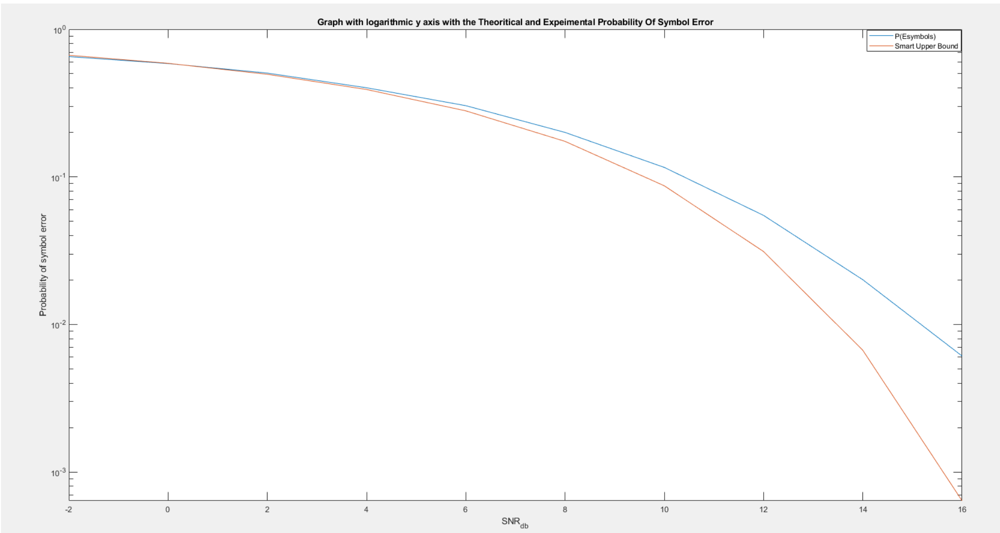


Figure 16: Graph with logarithmic y axis with the Theoretical and Experimental Probability Of Symbol Error

Τέλος, για την εύρεση του κάτω φράγματος της πιθανότητας σφάλματος bit, διαιρέθηκε το έξυπνο άνω φράγμα της πιθανότητας σφάλματος συμβόλου με το 3, καθώς και για κάθε λάθος σύμβολο ,απεικονιζόμενο με κωδικοποίηση 8-PSK, υπάρχει τουλάχιστον ένα λάθος bit.Παρακάτω φαίνεται το αντίστοιχο διάγραμμα των πειραματικών και θεωρητικών παραστάσεων της πιθανότητας σφάλματος bit , σε ένα κοινό διάγραμμα με λογαριθμικό κατακόρυφο άξονα.

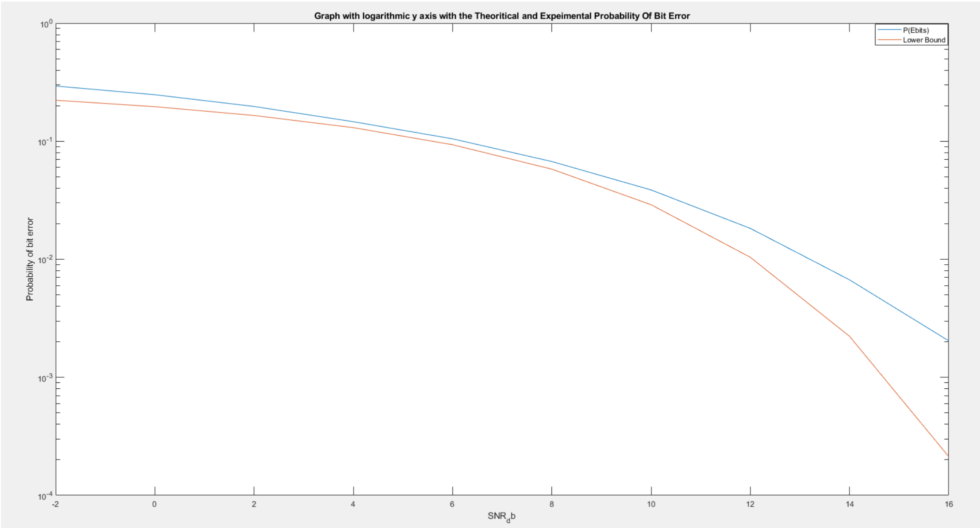


Figure 17: Graph with logarithmic y axis with the Theoretical and Experimental Probability Of Bit Error

Παρατηρείται πως οι πειραματικές τιμές της μεθόδου προσεγγίζουν με καλή ακρίβεια το έξυπνο άνω φράγμα, αλλά και το κάτω φράγμα(με ελάχιστη παραπάνω απόκλειση) .Παρόλα αυτά, συμβαίνει και κάτι το ασυνήθες , για τιμές του SNR‘6 υπάρχει τέλεια ταύτιση των δύο γραφικών στην περίπτωση της πιθανότητας σφαλμάτων συμβόλου και καλύτερη προσέγγιση τους στην περίπτωση της πιθανότητας σφάλματος bit, πράγμα που στέκεται αντίθετα στην θεωρία, καθώς και είναι γνωστό πως όσο το SNR μειώνεται ,τότε ο θόρυβος θα έπρεπε να αυξάνεται και όχι να μειώνεται .Προφανώς όσο λιγότερος θόρυβος υπάρχει τόσο καλύτερη γίνεται η προσέγγιση