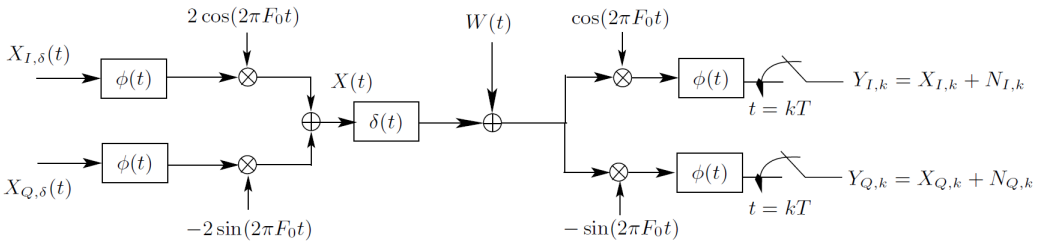
**Πολυτεχνείο Κρήτης**
**Σχολή ΗΜΜΥ**
**Τηλεπικοινωνιακά Συστήματα Ι**
Παράδοση 3ης εργασίας
Ημερομηνία Παράδοσης: 12 Ιουνίου 2023
Μονάδες 130/1000

# Ομάδα 55

| | Φοιτητής |
|---|---|
| Επώνυμο | **Πέτρου** |
| Όνομα | **Δημήτριος** |
| Α.Μ. | **2018030070** |
| Χρόνος Εκπόνησης | **22 ώρες** |

# Μέρος Α

Στο Α Μέρος της άσκησης στόχος ήταν να προσομοιωθεί το τηλεπικοινωνιακό σύστημα του παρακάτω σχήματος, χρησιμοποιώντας διαμόρφωση 16-PSK. Πάνω σε αυτή την υλοποιήση μελετήθηκε η απόδοση του προτύπου 16-PSK καθώς και τα σφάλματα με συμβαίνουν κατά τις μεταδόσεις εξ-αιτίας αυτού.




## A1.

Η δυαδική ακολουθία 4N ισοπίθανων bit `bit_seq` δημιουργήθηκε ως ένας πίνακας 100x4. Η επιλογή αυτή έγινε καθώς κάθε γραμμή αυτού του πίνακα θα μετατραπεί αργότερα σε ένα σύμβολο 16-PSK:

```
//partA.m
...
bit_seq = (sign(randn(N,4))+1)/2;
...
```

## A2.

Η μετατροπή της ακολουθίας bits σε σύμβολα γινεται μέσω της συνάρτησης `bits_to_PSK_16(bit_seq)` η οποία υλοποιήθηκε ως εξής:

```
//bits_to_PSK_16.m
function X = bits_to_PSK_16(bit_seq)
    N = size(bit_seq,1); % number of rows
    if size(bit_seq,2) ~= 4
        error('Each row of bit sequence must be 4 bits long for 16-PSK');
    end
    X = zeros(N,2);
    for n = 1:N
        bits = bit_seq(n,:);
        m = bi2de(bits,'left-msb'); % Convert bits to integer using Gray encoding
        X(n,:) = [cos(2*pi*m/16), sin(2*pi*m/16)];
    end
end
```

Η συνάρτηση αρχικά καταλαβαίνει πόσες γραμμές από 4-άδες διαθέτει η ακολουθία που δόθηκε ως παράμετρος και ελέγχει αν η κάθε γραμμή έχει 4 στοιχεία. Στη συνέχεια για κάθε γραμμή του πίνακα μετατρέπει την ακολουθία bits που αντιστοιχεί στην εκάστοτε γραμμή σε αριθμό με δεκαδική βάση που θα χρησιμοποιηθεί για τον υπολογισμό του διανύσματος των φορεών. Στην εκάστοτε γραμμή του διανύσματος συμβόλων που θα επιστραφεί με την ολοκλήρωση της συνάρτηση παρατίθενται οι δύο φορείς που ζητούνται.

## A3.

Χρησιμοποιώντας την συνάρτηση `srrc_pulse()` που είχε υλοποιηθεί από την 1η εργασία δημιουργήθηκε παλμός SRRC που θα χρησιμοποιηθεί για το φιλτράρισμα των ακολουθιών συμβόλων $X_{i,n}$ και $X_Q, n$.

```
//partA.m
...
T = 10^(-2);
over = 10;
Ts = T/over;
A = 4;
a = 0.5;
...
[ph,t] = srrc_pulse(T,Ts,A,a);
...
```

Το φιλτράρισμα των ακολουθιών συμβόλων με το παραγμένο παλμό επιτεύχθηκε με συνέλιξη μεταξύ τους. Για να γίνει εφικτό αυτό, παράχθηκαν τα σήμα delta και συνελίχθηκαν με τον SRRC παλμό. Υπολογίστηκε επίσης ο κατάλληλος άξονας χρόνου.

```
//partA.m
...
%Create the X_delta signals for XI and XQ, to be used for convolution.
XI_d = Fs * upsample(XI, over);
t_XI_d = (0:Ts:N/(1/T)-Ts);

XQ_d = Fs * upsample(XQ, over);
t_XQ_d = (0:Ts:N/(1/T)-Ts);

%Convolve the delta signals with the pulse and create the time axis.
t_XIt = (t(1)+t_XI_d(1):Ts:t(end)+t_XI_d(end));
t_XQt = (t(1)+t_XQ_d(1):Ts:t(end)+t_XQ_d(end));

XI_t = conv(ph, XI_d)*Ts;
XQ_t = conv(ph, XQ_d)*Ts;
t_total_I = length(t_XIt)*Ts;
t_total_Q = length(t_XQt)*Ts;
...
```

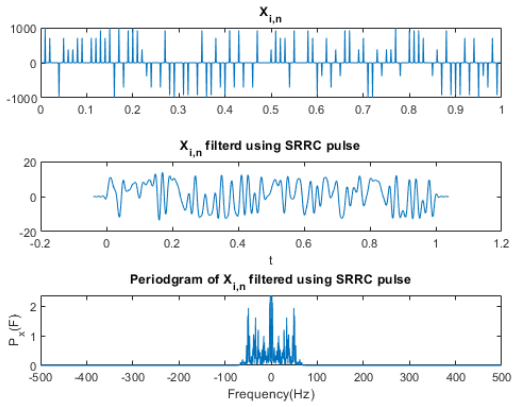
Στη συνέχεια για την απεικόνιση του περιοδογράμματος του κάθε φιλτραρισμένου σήματος, έγινε ο μετασχηματισμός Fourier του και το περιοδόγραμμα υπολογίστηκε σύμφωνα με τον τύπο:

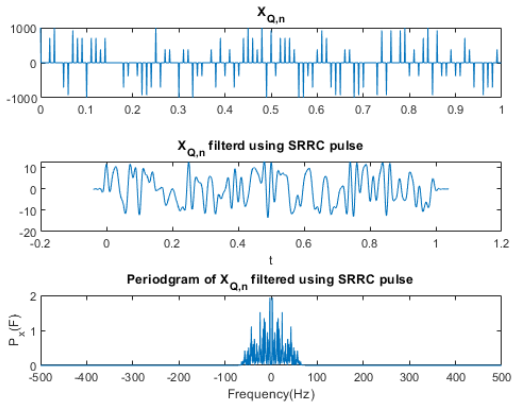
$$P_X(F) = \frac{|\mathcal{F}[X(t)]|^2}{T_{total}}$$

Ο κώδικας που πραγματοποιεί τον μετασχηματισμό και υπολογίζει το περιοδόγραμμα είναι:

```
//partA.m
...
%Perform Fourier Transform and calculate periodgram
FX_i = fftshift(fft(XI_t,Nf)*Ts);
PX_i = (abs(FX_i).^2)/t_total_I;

FX_q = fftshift(fft(XQ_t,Nf)*Ts);
PX_q = (abs(FX_q).^2)/t_total_Q;
...
```

Με χρήση της `plot()` απεικονίστηκαν τόσο τα φιλτραρισμένα σήματα όσο και τα περιοδογράμματά τους όπως αυτά αποτυπώνονται παρακάτω:




## A4.

Με σκοπό τον πολλαπλασιασμό των εξόδων των φίλτρων με τους κατάλληλους φορείς συγκεκριμένης συχνότητας, δημιουργήθηκαν οι παρακάτω πολλαπλασιασμοί με φορείς σύμφωνα με το σχηματικό του τηλεπικοινωνιακού συστήματος:

```
//partA.m
...
%Create XI(t) and XQ(t) by multiplying the filter outputs with appropriate
%phasors
XI = 2*(XI_t).*(cos(2*pi*F0*transpose(t_XIt)));
XI_total = length(t_XIt)*Ts;

XQ = -2*(XQ_t).*(sin(2*pi*F0*transpose(t_XQt)));
XQ_total = length(t_XQt)*Ts;
...
```

Για την απεικόνιση των περιοδογραμμάτων ήταν πρώτα απαραίτητα ο υπολογισμός τους μέσω του γνωστού τρόπου με μετασχηματισμό Fourier:

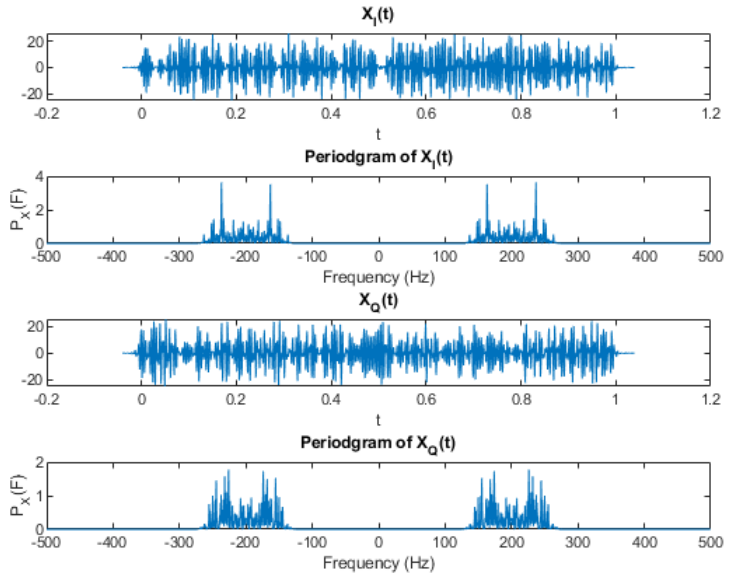
```
//partA.m
...
%Perform Fourier Transform and produce periodgrams
FXI = fftshift(fft(XI,Nf)*Ts);
PXI = (abs(FXI).^2)/XI_total;

FXQ = fftshift(fft(XQ,Nf)*Ts);
PXQ = (abs(FXQ).^2)/XQ_total;
...
```

Η τελική απεικόνιση των φερόμενων σημάτων και των περιοδογραμμάτων των δύο αρχικών σημάτων έγινε με χρήση της `plot()`:

```
//partA.m
...
%Plot the the XI,XQ and their periodgrams
figure('Name','A4')
subplot(4,1,1);
plot(t_XIt, XI);
title('X_I(t)');
xlabel('t');
subplot(4,1,2);
plot(F, PXI);
title('Periodgram of X_I(t)');
ylabel('P_X(F)');
xlabel('Frequency (Hz)');
subplot(4,1,3);
```

```
plot(t_XQt, XQ);
title('X_Q(t)');
xlabel('t');
subplot(4,1,4);
plot(F, PXQ);
title('Periodgram of X_Q(t)');
ylabel('P_X(F)');
xlabel('Frequency (Hz)');
...
```

Το αποτέλεσμα της απεικόνισης ήταν ως εξής:




Ο πολλαπλασιασμός των δύο φιλτραρισμένων εξόδων με τους αντίστοιχους φορείς καθιστά πλέον το σύστημα μη-βασικής ζώνης. Το φάσμα των αρχικών σημάτων καταλάμβανει πλέον την περιοχή γύρω από την συχνότητα του φορέα ($F_0 = \pm 200Hz$). Το πλάτος και των δύο εξόδων έχει αυξηθεί γεγονός που είναι δικαιολογημένο αφού ο φορέας με τον οποίο πολλαπλασιάστηκαν επέφερε ενίσχυση.

## A5.

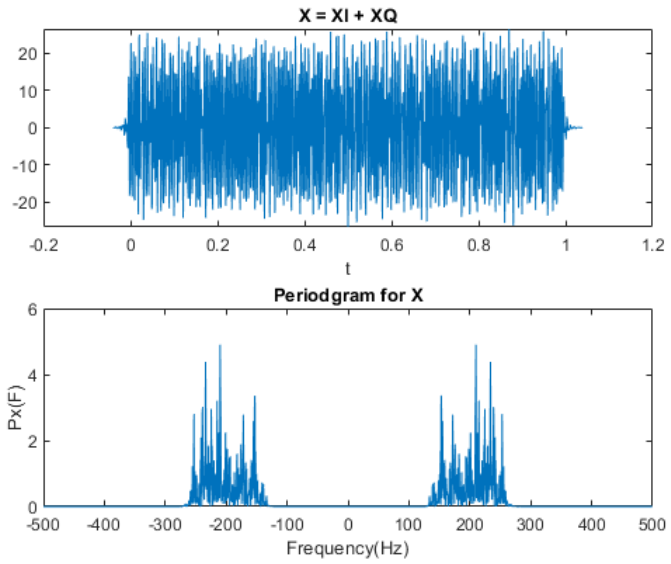
Η είσοδος X(t) του καναλιού είναι το άθροισμα των δύο επιμέρους εισόδων $X_I(t)$ και $X_Q(t)$:

$$X(t) = X_I(t) + X_Q(t)$$

Το X(t) υπολόγιστηκε όπως φαίνεται στον κώδικα παρακάτω ενώ το περιοδόγραμμα του παράχθηκε σύμφωνα με τον μετασχηματισμό Fourier $\mathcal{F}(X(t))$:

```
//partA.m
...
%Come up with the channel input X(t)
Xt = XI + XQ;
T_total = length(t_XQt)*Ts
%Perform Fourier Transform and compute Periodgram
Fxt = fftshift(fft(Xt,Nf)*Ts);
Pxt = (abs(Fxt).^2)/T_total;
...
```

Η απεικόνιση της εισόδου του καναλιού και του περιοδογράμματος της έγινε μέσω της `plot()`:




Το περιοδόγραμμα αποδεικνύει και πάλι ότι οι συχνότητες των φορέων που προσαρτήθηκαν στα σήματα εξόδου κυριαρχούν και πάλι καθώς το φάσμα του αθροίσματος (δηλ. της εισόδου του καναλιού) καταλαμβάνει τις περιοχές γύρω από τα $F_0 = \pm 200Hz$. Το πλάτος παρουσίαζεται αυξημένο γεγονός που οφείλεται στην άθροιση των πλατών των επικείμενων σημάτων.

## A6.

Δεδομένου ότι το κανάλι είναι ιδανικό, συμπεραίνεται πως το σύστημα έχει κρουστική απόκριση $\delta(t)$ και έτσι το σήμα εισόδου περνάει στην έξοδο χωρίς απώλειες ή αλλοιώσεις.

## A7.

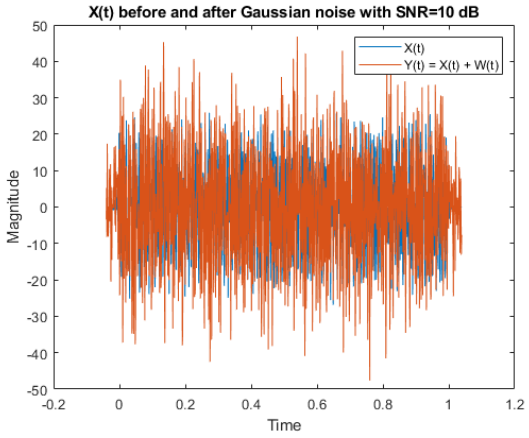
Ο ζητούμενος Gaussian White Noise δημιουργείται με διασπορά:

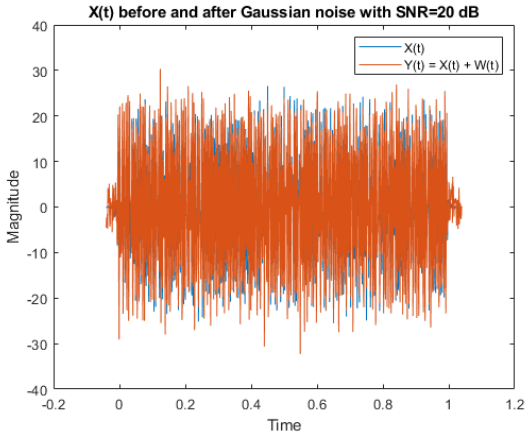
$$\sigma_W^2 = \frac{1}{T_s \cdot 10^{\frac{SNR_{dB}}{10}}}$$

για SNR = 10dB,20dB και προσαρτάται στην είσοδο του καναλιού με τον εξής κώδικα:

```
//partA.m
...
%Create the Gaussian noise W(t)
SNR = 10;
sw2 = 1/(Ts*(10^(SNR/10)));
W = sqrt(sw2).*randn(length(Xt),1);

Y = Xt + W;
...
```

Το σήμα $\Upsilon$ που πλέον φέρει απεικονίζεται με την χρήση `plot` και το αποτέλεσμα φαίνεται στην επόμενη σελίδα:



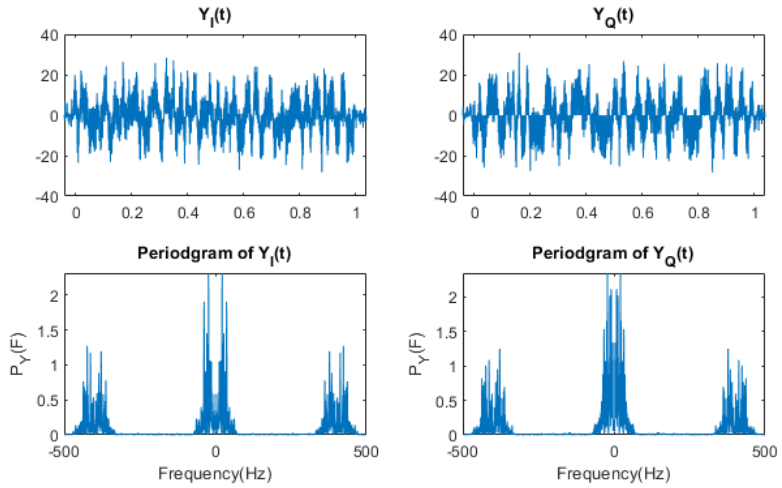




## A8.

Η ενθόρυβη κυματομορφή μπορεί πλέον να πολλαπλασιαστεί με τους κατάλληλους φορείς για να ξεκίνησει η διαδικασία ανάκτησης των αρχικών ακολουθιών συμβόλων. Επιλέχθηκε να γίνει η απεικόνιση για την τιμή SNR=20dB. Εκτελώντας τον παρακάτω κώδικα η ενθόρυβη κυματομορφή πολλαπλασιάζεται με φορείς όπως αυτοί ορίζονται απο το διάγραμμα του συστήματος, ενώ το περιοδόγραμμα της κάθε μιας προέρχεται από τον μετασχηματισμό Fourier:

```
1  //partA.m
2  ...
3  %Those are the YI,YQ signals coming from the multiplication of Yt with
4  %appropriate phasors
5  YI = Y.*(cos(2*pi*F0*transpose(t_XQt)));
6  YQ = Y.*(-sin(2*pi*F0*transpose(t_XQt)));
7
8  t_YI_total = length(t_XQt)*Ts;
9  t_YQ_total = length(t_XQt)*Ts;
10
11 %Perform Fourier Transform on them and produce periodgrams
12 FYI = fftshift(fft(YI,Nf)*Ts);
13 FYQ = fftshift(fft(YQ,Nf)*Ts);
14
15 PYI = (abs(FYI).^2)/t_YI_total;
16 PYQ = (abs(FYQ).^2)/t_YQ_total;
17 ...
```

Τα αποτελέσμα αποτύπωνονται παρακάτω όπως προέκυψαν με χρήση της `plot()`:



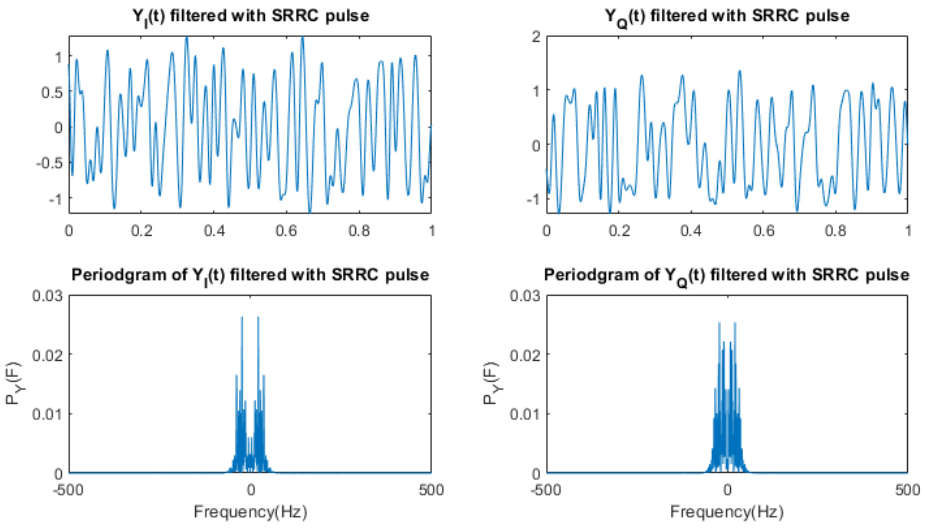
Η διαδικασία αποδιαμόρφωσης του σήματος, πολλαπλασιάζοντάς το με τους φορείς συχνότητας $F_0$ στοχεύει στην επαναφορά του στην βασική ζώνη. Πλέον, όπως αυτό αποτυπώνεται και στα δύο περιοδογράμματα, υπάρχει ορατό φάσμα στη βασική ζώνη (δηλ. γύρω από το 0) χωρίς όμως να υστερούν οι υψηλές συχνότητες. Προκειμένου να απαλειφθούν οι λοβοί στις υψηλές συχνότητες τα αποδιαμορφωμένα πλέον σήματα θα περάσουν από φίλτρα SRRC στο επόμενο ερώτημα.

## A9.

Όπως αναφέρεται και παραπάνω προκειμένου να αποδεσμευτούν πλήρως τα αποδιαμορφωμένα σήματα από τις υψηλές συχνότητες θα περάσουν μέσα από SRRC φίλτρα όμοια με αυτά που χρησιμοποιήθηκαν πριν την διαμόρφωση τους.

```
//partA.m
...
%Using the SRRC pulse from A3 we convolve the the YI,YQ signals with it.
YI_2 = conv(ph, YI)*Ts;
YQ_2 = conv(ph, YQ)*Ts;
%Specify the correct time axis
t_YI_2 = (t(1)+t_XQt(1):Ts:t(end)+t_XQt(end));
t_YQ_2 = (t(1)+t_XQt(1):Ts:t(end)+t_XQt(end));

%Tail cut the filtered signal by 80 indexes
YI_cut = YI_2(80+1:(length(t_YI_2)-80));
t_YI_cut = t_YI_2(80+1:(length(t_YI_2)-80));

YQ_cut = YQ_2(80+1:(length(t_YQ_2)-80));
t_YQ_cut = t_YQ_2(80+1:(length(t_YQ_2)-80));
...
```

Το τελικό φιλτραρισμένο σήμα αποτελείται από 160 στοιχεία παραπάνω για το λόγο αυτό ακολουθούμε τεχνική tail cutting προκειμένου να τα απορρίψουμε ώστε να μπορούν να χρησιμοποιηθούν στις δειγματοληψίες αργότερα. Τα περιοδογράμματα προέκυψαν μέσω μετασχηματισμού Fourier και η τελική απεικόνιση μέσω της `plot()` έδωσε τα παρακάτω αποτελέσματα;



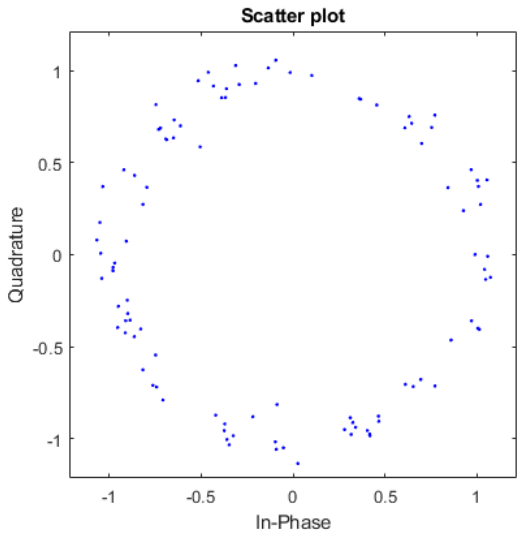

Οι υψήσυχνοι λοβοί δεν υπάρχουν πλέον και δύναται να ανακτηθεί πληροφορία για το αρχικό σήμα.

## A10.

Η έξοδος των προσαρμοσμένων φίλτρων δειγματοληπτείται και σχεδιάζεται η ακολουθία εξόδου $\Upsilon$ μέσω της `scatterplot()`. Μέσω της `downsample()` υποδειγματοοληπτείται το κάθε σύμβολο και ανακτάται το Yn ένας πίνακας συμβόλων 100x2. Σημειώνεται πως παρακάτω αποτυπώνεται η περίπτωση για SNR=20dB:

```
//partA.m
...
%Downsample and create the default Yi,n and Yq,n
YI_down = downsample(YI_cut, over);
YQ_down = downsample(YQ_cut, over);

%Create Yn
Yn=[YI_down, YQ_down];
%Scatterplot
scatterplot(Yn);
...
```

Απεικονίζοντας με την χρήση της `scatterplot()` προκύπτει:




Γύρω από τα 16 milestones του επιπέδου (εξαιτίας του 16-PSK) δημιουργείται ένα νέφος σημείων. Η απόκλιση του κάθε σημείου από την ακριβή του θέση οφείλεται στο θόρυβο που προσαρτήθηκε.

## A11.

Χρησιμοποιώντας τον κανόνα του εγγύτερου γείτονα με την ευκλείδια απόσταση, δημιουργήθηκε η συνάρτηση `detect_PSK_16(Yn)`. Παράλληλα μέσω της αντίστροφη κωδικοποίησης Gray υπολογίζεται μέσω των συμβόλων και η ακολουθία bits που έχει διαμορφωθεί με 16-PSK. Η συνάρτηση επιστρέφει δύο ακολουθίες, μια ακολουθία συμβόλων και μια ακολουθία από bits. Παρατίθεται ο κώδικας της:

```
//detect_PSK_16.m
% Function to detect 16-PSK symbols using the nearest neighbor rule
% and to compute the estimated bit sequence using the inverse Gray code
function [est_X, est_bit_seq] = detect_PSK_16(Y)
    % Get the number of symbols (rows in Y)
    N = size(Y,1);

    % Check that each symbol in Y is 2-dimensional
    if size(Y,2) ~= 2
```

```
        error('Each row of Y must be 2-dimensional for 16-PSK');
    end

    % Initialize the estimated symbol and bit sequence matrices
    est_X = zeros(N,2);
    est_bit_seq = zeros(N,4);

    % For each symbol in Y...
    for n = 1:N
        % Initialize the minimum distance to infinity
        min_dist = inf;

        % For each possible 16-PSK symbol...
        for m = 0:15
            % Compute the 16-PSK symbol for m
            X_m = [cos(2*pi*m/16), sin(2*pi*m/16)];

            % Compute the Euclidean distance from Y(n,:) to X_m
            dist = norm(Y(n,:) - X_m);

            % If this distance is less than the current minimum distance...
            if dist < min_dist
                % ... then update the minimum distance ...
                min_dist = dist;
                % ... and update the estimated symbol and bit sequence.
                est_X(n,:) = X_m;
                est_bit_seq(n,:) = de2bi(m, 4, 'left-msb'); % Convert integer m to 4-bit binary using Gray encoding
            end
        end
    end
end
```

## A12.

Αφού ανακτηθεί η ακολουθία συμβόλων μέσω της `detect_PKS_16(Y)` κρίνεται απαραίτητο να αποδειχθεί αν η διαμόρφωση και η μετάδοση της αρχικής στο τηλεπικοινωνιακό σύστημα ήταν ασφαλής και αναλλοίωτη. Ο έλεγχος αυτό επιτυγχάνεται μέσω της συνάρτησης `symbol_errors(est_X,X)`. Ο κώδικας της αποτυπώνεται παρακάτω:

```
//symbol_errors.m

function num_of_symbol_errors = symbol_errors(est_X, X)
    % Compare the first column of est_X and X element-wise, and generate a logical array.
    column1_errors = est_X(:, 1) ~= X(:, 1);

    % Compare the second column of est_X and X element-wise, and generate a logical array.
    column2_errors = est_X(:, 2) ~= X(:, 2);

    % Combine the logical arrays using the logical OR operator.
    % Elements with true values indicate symbol errors.
    total_errors = column1_errors | column2_errors;

    % Compute the sum of true values to get the number of symbol errors.
    num_of_symbol_errors = sum(total_errors);

    % Return the total number of symbol errors.
end
```

## A13.

Αντίστοιχα με το προηγούμενο ερώτηματα απαιτείται ένας τρόπος για ανίχνευση σφαλμάτων μετάδοσης σε επίπεδο bit. Για το σκοπό αυτό υλοποιήθηκε η συνάρτηση `symbol_errors(est_bit_seq,b)`:

```
//bit_errors.m

function num_of_bit_errors = bit_errors(est_bit_seq, b)
    % Check the dimensions of the input bit sequences
    if size(est_bit_seq) ~= size(b)
        error('Dimensions of the estimated bit sequence and the true bit sequence must match');
    end

    % Compute the bit errors through logical comparisons
    bit_errors = est_bit_seq ~= b;

    % Sum all the bit errors
    num_of_bit_errors = sum(bit_errors(:));
end
```

Συνήθως τα symbol και bit erros που προκύπτουν είναι 0 ή πολύ κοντα στο μηδέν (1-2) γεγονός που τα καθιστά αμελητέα, οπότε θεωρείται κι ότι η αρχική πληροφορία παραμένει αναλλοίωτη δεδομένης της ισχύς του θορύβου για SNR=20dB.

# Μέρος Β

Στο 2ο μέρος της εργασίας σκοπός είναι η εκτίμηση των πιθανοτήτων σφάλματος συμβόλου και bit με χρήση της μεθόδου Monte Carlo.

## B1 B2 B3.

Γενικότερα η πειραματική εκτίμηση των πιθανοτήτων σφάλματος γίνεται ως εξής:

- για σύμβολα

$$\hat{P}(E_{Symbol}) = \frac{\text{συνολικό πλήθος σφαλμάτων απόφασης συμβόλου}}{\text{συνολικό πλήθος απεσταλμένων συμβόλων}}$$

- για bit

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

Ζητείται η εκτίμηση των πιθανοτήτων για διάφορες τιμές του $SNR_{dB}$ στο εύρος [-2:2:24], άρα συνολικά 14 τιμές SNR. Για κάθε μία από αυτές τις τιμές θα εκτελεστούν 1000 πειράματα προσομοίωσης του αρχικού τηλεπικοινωνιακού συστήματος και θα καταγραφούν τα σφάλματα συμβόλων και bits που προέκυψαν σε καθε ένα από αυτά. Με την ολοκλήρωση της διαδικασίας θα υπάρξει μια ασφαλής εκτίμηση για τις δύο πιθανότητες οι οποίες και θα απεικονιστούν ως συνάρτηση του $SNR_{dB}$ με χρήση `semilogy()`. Στα διάγραμματα που θα προκύψουν σχεδιάζονται επίσης το έξυπνο άνω φραγμα για τα σφάλματα συμβόλου και το κάτω φράγμα για τα σφάλματα bit. Οι παλμοί SRRC που χρησιμοποιούνται ως φίλτρα στα πειράματα είναι ίδιων προδιαγραφών με αυτούς του Α'Μέρους. Η διαδικασία που ακολουθείται σε κάθε πείραμα ουσιαστικά αποτελεί τα βήματα του Α'Μέρους. Παρατίθεται ο κώδικας που υλοποιεί τα ζητούμενα:

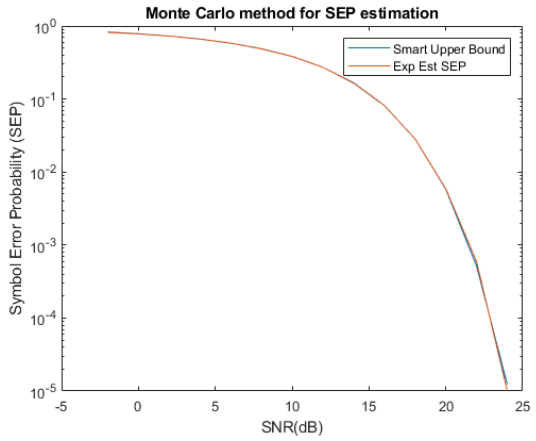
```
//partB.m
%% B1

N=100;
SNR=[-2:2:24];
```

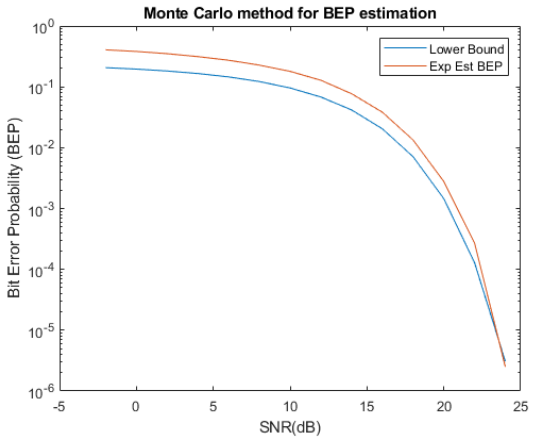
```
T=10^(-2);
over=10;
Ts=T/over;
Fs=1/Ts;
A=4;
a=0.5;
F0=200;
K=1000;

%Generate SRRC pulse, all cycles should be
%performed using the same SRRC filters.
[ph,t] = srrc_pulse(T,Ts,A,a);

disp('Please be patient...this might take a while');
%For all values of SNR set in the vector above, simulate
%the telecom system over and over again to calculate
%the Symbol Error Probability and Bit Error Probability
for cycle=1:length(SNR)

    symbolErrorsOfSamples = 0;
    bitErrorsOfSamples = 0;
    fprintf('Running %d experiments for cycle %d for SNR value=%ddB\n',K,cycle,SNR(cycle));

    for z=1:K
        bit_seq = (sign(randn(N,4))+1)/2;
        Xn = bits_to_PSK_16(bit_seq);
        Xi = Xn(:,1);
        Xq = Xn(:,2);

        %Calculate the deltas
        Xi_d = Fs*upsample(Xi,over);
        Xq_d = Fs*upsample(Xq,over);

        t_Xi_d = (0:Ts:N/(1/T)-Ts);
        t_Xq_d = (0:Ts:N/(1/T)-Ts);

        %Convolve with the SRRC filter
        Xi_t = conv(ph,Xi_d)*Ts;
        Xq_t = conv(ph,Xq_d)*Ts;

        t_Xi_t = (t(1)+t_Xi_d(1):Ts:t(end)+t_Xi_d(end));
        t_Xq_t = (t(1)+t_Xq_d(1):Ts:t(end)+t_Xq_d(end));

        %Attach the signals to phasors
        XI = 2 * (Xi_t) .* (cos(2*pi*F0*transpose(t_Xi_t)));
        XQ = -2 * (Xq_t) .* (sin(2*pi*F0*transpose(t_Xq_t)));

        %Create the channel input signal
        X = XI + XQ;

        %Generate Gaussian White Noise and attach to signal;
        sw2 = 1/(Ts*(10^(SNR(cycle)/10)));
        W = sqrt(sw2).*randn(length(X),1);
        %Calculate variance
        sigma2 = Ts*sw2/2;

        Y = X + W;

        %Generate Yi,Yq
        Yi_t = Y.*(cos(2*pi*F0*transpose(t_Xq_t)));
        Yq_t = Y.*(-sin(2*pi*F0*transpose(t_Xq_t)));

        %Convole the 2 signals with the SRRC filter
        YI_f = conv(ph, Yi_t)*Ts;
        YQ_f = conv(ph, Yq_t)*Ts;

        t_YI_f = (t(1)+t_Xq_t(1):Ts:t(end)+t_Xq_t(end));
        t_YQ_f = (t(1)+t_Xq_t(1):Ts:t(end)+t_Xq_t(end));

```

```
75          %Tailcut the two convolutions
76          Yi_cut = YI_f(80+1:(length(t_YI_f)-80));
77          Yq_cut = YQ_f(80+1:(length(t_YQ_f)-80));
78
79          %Downsample and produce Yi,n and Yq,n
80          YI = downsample(Yi_cut,over);
81          YQ = downsample(Yq_cut,over);
82
83          %Create the derived Yn sequence of symbols
84          Yn = [YI YQ];
85
86          %Get the estimated bit sequence
87          [est_X, est_bit_seq] = detect_PSK_16(Yn);
88
89          symbolErrorsOfSamples = symbolErrorsOfSamples + symbol_errors(est_X, Xn);
90          bitErrorsOfSamples = bitErrorsOfSamples + bit_errors(est_bit_seq, bit_seq);
91      end
92
93      %Update statistics
94      %Calculate probabilities of passed cycle
95      SEP(1,cycle) = symbolErrorsOfSamples/(N*K);
96      BEP(1,cycle) = bitErrorsOfSamples/(N*K*4);
97
98      %Calculate smart upper bound for SEP and bit lower bound for BEP
99      SUB(cycle) = 2*Q(1./sqrt(sigma2)*sin(pi/16));
100     BLB(cycle) = SUB(cycle)/4;
101 end
102
103 %%
104 %Plot the outcomes of the experiment
105 figure('Name','B2');
106 semilogy(SNR, SUB);
107 hold on;
108 semilogy(SNR, SEP);
109 hold off;
110 xlabel('SNR(dB)');
111 ylabel('Symbol Error Probability (SEP)');
112 legend('Smart Upper Bound','Exp Est SEP');
113 title('Monte Carlo method for SEP estimation');
114
115 figure('Name','B3');
116 semilogy(SNR, BLB);
117 hold on;
118 semilogy(SNR, BEP);
119 hold off;
120 xlabel('SNR(dB)');
121 ylabel('Bit Error Probability (BEP)');
122 legend('Lower Bound','Exp Est BEP');
123 title('Monte Carlo method for BEP estimation');
```

Όπως αποτυπώνεται στις γραμμές 89 και μετά, για κάθε τιμή SNR συλλέγεται πληροφορία για συνολικά σφάλματα που συνέβησαν στα 1000 πειράματα και αποθηκεύεται. Η πληροφορία αυτή χρησιμοποείται στις γραμμές 95,96 για το υπολογισμό των 2 ειδών πιθανοτήτων. Στην πρώτη περίπτωση το πλήθος συμβόλων που απεστάλθηκαν για τη δεδομένη τιμή SNR ηταν 1000xN ενώς το πλήθος bits ήταν 1000xNx4. To smart upper bound υπολογίστηκε χρησιμοποιώντας την δεδομένη συνάρτηση `Q.m`.

Η απεικόνιση με `semilogy()` δίνει τα παρακάτω αποτελέσματα:




Στα δύο διαγράμματα παρατηρείται ελάχιστη απόκλιση μεταξύ των φραγμάτων και των καμπύλων που προέκυψαν από τα πειράματα. Ο θόρυβος που ορίζεται από την παράμετρο SNR προκαλεί μεταβολή και των δύο πιθανοτήτων σφάλματος. Με την αύξηση του SNR αναμένεται μείωση των πιθανοτήτων σφάλματος δεδομένης της μεταβολής του θορύβου.