Technical University of Crete

Τηλεπικοινωνιακά Συστήματα I

Άσκηση 3

Author:
Σπυριδάκης Χρήστος

AM: 2014030022

January 9, 2019

Εισαγωγή

Για την διευκόλυνση της υλοποίησης είναι καλό να σημειωθεί ότι το κάθε μέρος υλοποιήθηκε σε διαφορετικό script, αυτό είχε τα θετικά του, στο να είναι περισσότερο διακριτός ο κώδικας για ανάπτυξη και αποσφαλμάτωση, αλλά χρειάστηκε να ξανά δημιουργηθούν σήματα που είχαν δημιουργηθεί σε προηγούμενα ερωτήματα. Δεν επηρεάζονται κάπως τα αποτελέσματα απλά είναι μία διευκρίνηση σχετικά με την δομή που δόθηκε στον κώδικα.
Επίσης παρατηρήθηκε ότι κατά την διάρκεια της άσκησης είναι πολλά πράγματα τα οποία μοιάζουν μεταξύ των ζητουμένων. Ακολουθήθηκε λοιπόν μία τακτική δομημένου προγραμματισμού με χρήση συναρτήσεων, προκειμένου ενέργειες που ζητούνται από περισσότερο του ενός σημεία, να μην επαναλαμβάνονται από το μηδέν. Για αυτό το λόγο υπάρχουν πολλαπλά βοηθητικά αρχεία τα οποία δημιουργήθηκαν. Σε κάθε ενότητα όπως χρειάζεται θα εισαγάγεται η λειτουργικότητα της κάθε συνάρτησης που περιέχεται σε ένα αρχείο και στην συνέχεια θα δείχνεται μόνο ο τρόπος κλήσης της. Τα αρχεία τα οποία υπάρχουν συνολικά για την ολοκληρωμένη εκτέλεση της άσκησης είναι τα εξής: ***srrc_pulses.m*** , ***bits_to_4_PAM.m*** , ***detect_4_PAM.m*** , ***PAM_4_to_bits.m*** , ***fourier_transform.m*** , ***display_waveform_periodogram.m*** , ***periodogram.m*** , ***part_a.m*** , ***part_b.m***.
Να αναφερθεί ότι στους ενδιάμεσους κώδικες (για το κάθε ερώτημα) εμφανίζεται MONO το κομμάτι υπολογισμού του ερωτήματος, δεν εμφανίζονται δηλαδή κατά κύριο λόγω κομμάτια κώδικα σχετικά με την δημιουργία των figure ή τις έξτρα πληροφορίες για αυτά - αν δεν είναι σημαντικό - όπως επίσης και κάποια από τα σχόλια για εξοικονόμηση χώρου. Γενικά έχει ελαφρώς αλλαχθεί ο κώδικας που παρουσιάζεται σε κάθε ερώτηση ώστε να κρατηθούν μόνο τα σημαντικά σημεία. Στο τέλος της αναφοράς υπάρχει ολόκληρος ο κώδικας για έλεγχο και αυτών των σημείων.
Τέλος, αν λόγω της εκτύπωσης σε χαρτί δεν είναι εμφανές σε ικανοποιητικό βαθμό κάποιο από τα figures μπορούν να βρεθούν όλα τα μέρη του project στο παρακάτω repository όπου υπάρχουν και screenshot αυτών, που φαίνονται με καλύτερη ανάλυση: `https://github.com/CSpyridakis/CommSys`

Ερώτημα Α

A.1 Create random bits

Πρώτο ζητούμενο της συγκεκριμένης άσκησης είναι να δημιουργηθεί μία δυαδική ακολουθία με 4N ισοπίθανα bits. Για να γίνει αυτό, χρησιμοποιήσαμε τον κώδικα που εμφανίζεται στο listing 1 και είναι όπως έχει πραγματοποιηθεί και στις προηγούμενες ασκήσεις για αυτό δεν γίνεται περαιτέρω ανάλυση. Το μόνο που πρέπει να σημειωθεί είναι ότι επιλέχθηκε ως $N = 200$. Συνεπώς θα δημιουργούνταν 800 τυχαία bits.

Listing 1: A.1 Create random bits

```
% A.1
N = 200;                                  % Random bits E.g.
bit_seq = (sign(randn(4*N, 1)) + 1)/2;  % 0 1 1 0 0 1 1 1. . .
```

A.2 Bits to 4-PAM

Αφού κάναμε το A1 έπρεπε να συντάξουμε συνάρτηση ***bits_to_4_PAM(bit_seq, A)*** παρόμοια με αυτή που κάναμε στο δεύτερο project με την διαφορά ότι θα έπρεπε σε αυτή να χρησιμοποιήσουμε κώδικα Gray στην αναπαράσταση bit σε σύμβολα 4-PAM. Αυτό που μας βοηθάει η κωδικοποίηση Gray είναι ότι στην περίπτωση όπου έχουμε σφάλμα στην μετάδοση ενός συμβόλου και τυγχάνει να πάρουμε ένα γειτονικό σύμβολο αντί αυτού (κάτι που κατά πάσα πιθανότητα αυτό θα συμβεί όταν έχουμε σωστό σχεδιασμό), επειδή το ένα σύμβολο από το άλλο διαφέρουν μόνο κατά 1 bit έτσι και το σφάλμα σε επίπεδο bit θα είναι σε αυτήν την περίπτωση μόνο ένα.

Listing 2: A.2 `bits_to_4_PAM(bit_seq, A)`

```
function [ X ] = bits_to_4_PAM(bit_seq, A)
    k=1;
    X=zeros(1,length(bit_seq)/2);
    for i=1:2:length(bit_seq)
        if(bit_seq(i)==0 && bit_seq(i+1)==0)        % 00 -> +3
            X(k) = 3*A;
        elseif(bit_seq(i)==0 && bit_seq(i+1)==1)    % 01 -> +1
            X(k) = 1*A;
        elseif(bit_seq(i)==1 && bit_seq(i+1)==1)    % 11 -> -1
            X(k) = -1*A;
        elseif(bit_seq(i)==1 && bit_seq(i+1)==0)    % 10 -> -3
            X(k) = -3*A;
        end
        k=k+1;
    end
end
```

Αφού είχαμε δημιουργήσει την συνάρτηση αυτή (η οποία έγινε με χρήση απλών συνθηκών ελέγχου), το μόνο που χρειαζόταν ήταν στο κύριο πρόγραμμα να την καλέσουμε για $A = 1$ ώστε να μετατρέψουμε τα τυχαία bits που δημιουργήσαμε σε σύμβολα. Αφού έχουμε επιλέξει $N = 200$ συνεπώς έχουμε στο bit_seq μέγεθος 800bits και επειδή δύο bits μας δίνουν ένα σύμβολο 4-PAM, θα έχουμε 400 σύμβολα στο Xn.

Listing 3: A.2 `Convert bits to symbols`

```
% A.2
A = 1;                          % Bits to 4 Pam E.g
Xn = bits_to_4_PAM(bit_seq, A);  % +1 -3 -1 -1 +3 .
```

A.3 $\{X_{I,n}\}$ and $\{X_{Q,n}\}$

Έχοντας κάνει τα παραπάνω 'σπάσαμε' την ακολουθία συμβόλων σε δύο ξεχωριστές. Η πρώτη είναι η $X_{I,n}$ για $n = 1, ..., N$ για τα 2N πρώτα bits. Ενώ η δεύτερη $X_{Q,n}$ για $n = 1, ..., N$ για τα 2N τελευταία bits. Συνεπώς στην κάθε μία υπάρχουν 200bits.

Listing 4: A.3 `Split Symbols`

```
% A.3
XI_n = Xn(1:N);                  % In Phase Symbols
XQ_n = Xn(N+1:2*N);              % Quadrature Symbols
```

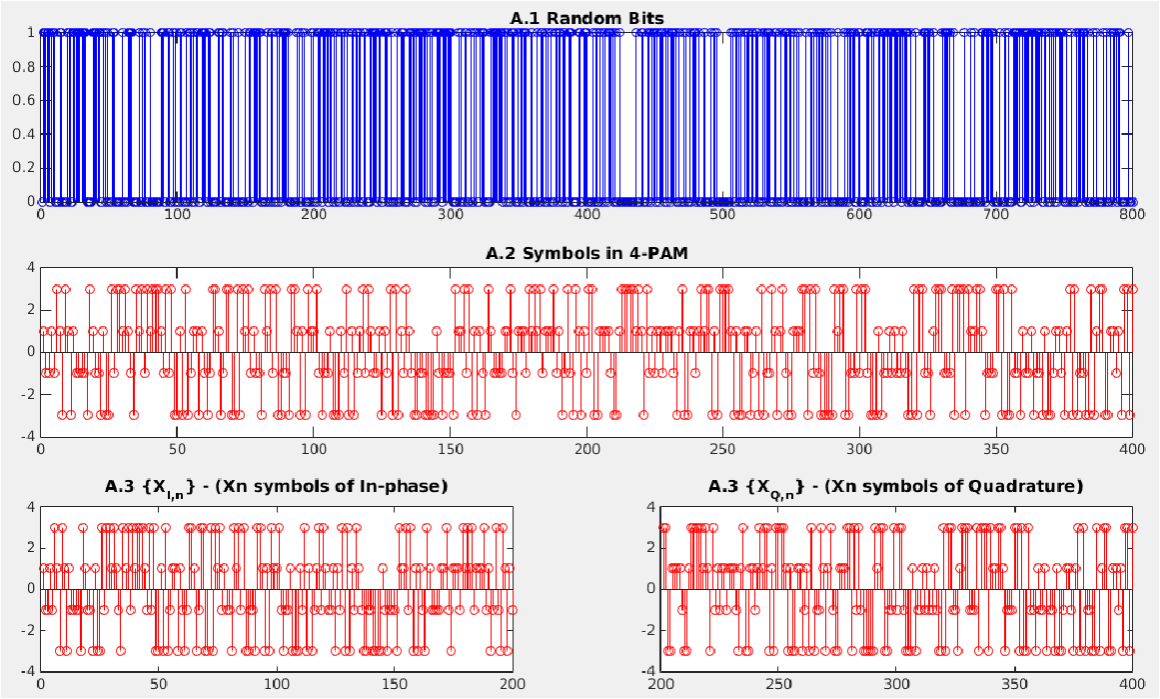

Figure 1: Random Bits, 4-PAM Symbols and Inphase/Quadrature Symbols

A.4 $X_I(t)$ and $X_Q(t)$

Σε αυτό το σημείο αφού είχαμε τις δύο ακολουθίες από bits - $X_{I,n}$ και $X_{Q,n}$ - χρειάστηκε να τις περάσουμε από τα SRRC φίλτρα μορφοποίησης. Για τα φίλτρα χρησιμοποιήθηκε $T = 0.01$ sec, $over = 10$, $T_s = \frac{T}{over}$ ενώ για το A και α δόθηκαν οι τιμές που δοκιμάστηκαν στο δεύτερο project δηλαδή $A = 4$ και $a = 0.5$. Στο listing 5 εμφανίζεται ο τρόπος με τον οποίο περνάμε τις δύο συμβολοσειρές από τα φίλτρα. Ουσιαστικά είναι με τον ίδιο τρόπο που ήδη έχει παρουσιαστεί στα προηγούμενα project, συνεπώς δεν ξανά αναλύεται. Επίσης να αναφερθεί ότι το $Nf = 2048$ κατά όλη την διάρκεια της άσκησης.

Listing 5: A.4 Calculate $X_I(t)$ and $X_Q(t)$

```
% A.4
T = 0.01 ; over = 10 ; Ts = T/over ; A_s = 4 ; a = 0.5;
Nf = 2048 ; Fs = 1/Ts ; F = [-Fs/2 : Fs/Nf : Fs/2-Fs/Nf]; % Frequency vector

% Phi
[phi_t t_phi] = srrc_pulse(T, Ts, A_s, a);

% Create upsampled X_delta signals and using it calculate conv
XI_d = 1/Ts * upsample(XI_n, over) ; XI_t = conv(XI_d, phi_t).*Ts ;
XQ_d = 1/Ts * upsample(XQ_n, over) ; XQ_t = conv(XQ_d, phi_t).*Ts ;
td = [0 : Ts : (N*over-1)*Ts ]; t_Xt = [td(1) + t_phi(1) : Ts : td(end) + t_phi(end)];

display_waveform_periodogram('A.4', 'X_i(t)', XI_t, 'X_q(t)', XQ_t, t_Xt,t_Xt, Ts, Nf)
```

Στο listing 5 παρουσιάζεται η συνάρτηση display_waveform_periodogram(). Ουσιαστικά ο ρόλος της συγκεκριμένης συνάρτησης είναι να εμφανίζει την κυμματοφορφή/κυμματομορφές που της δίνονται ως είσοδο, να υπολογίζει το/τα περιοδόγραμμα τους και να εμφανίζει και αυτά. Δεν δίνεται περισσότερη ανάλυση εδώ καθώς είναι κυρίως εντολές σχετικά με εμφάνιση (figure, plot, etc...) και η χρήση της custom συνάρτησης periodogram() που δημιουργήθηκε και αναλύθηκε στο δεύτερο project και σκοπό έχει να υπολογίζει το περιοδόγραμμα ενός σήματος. Παρόλα αυτά δίνεται στο τέλος η υλοποίηση της.

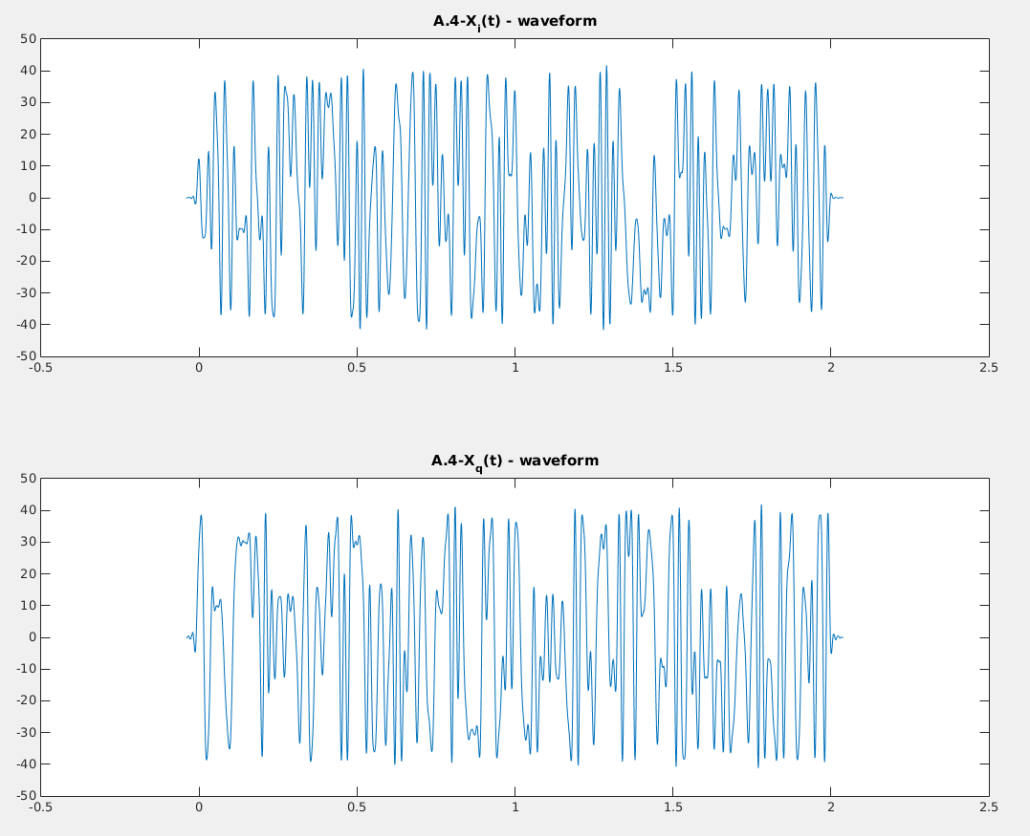

Figure 2: $X_I(t)$ `and` $X_Q(t)$ `waveforms`

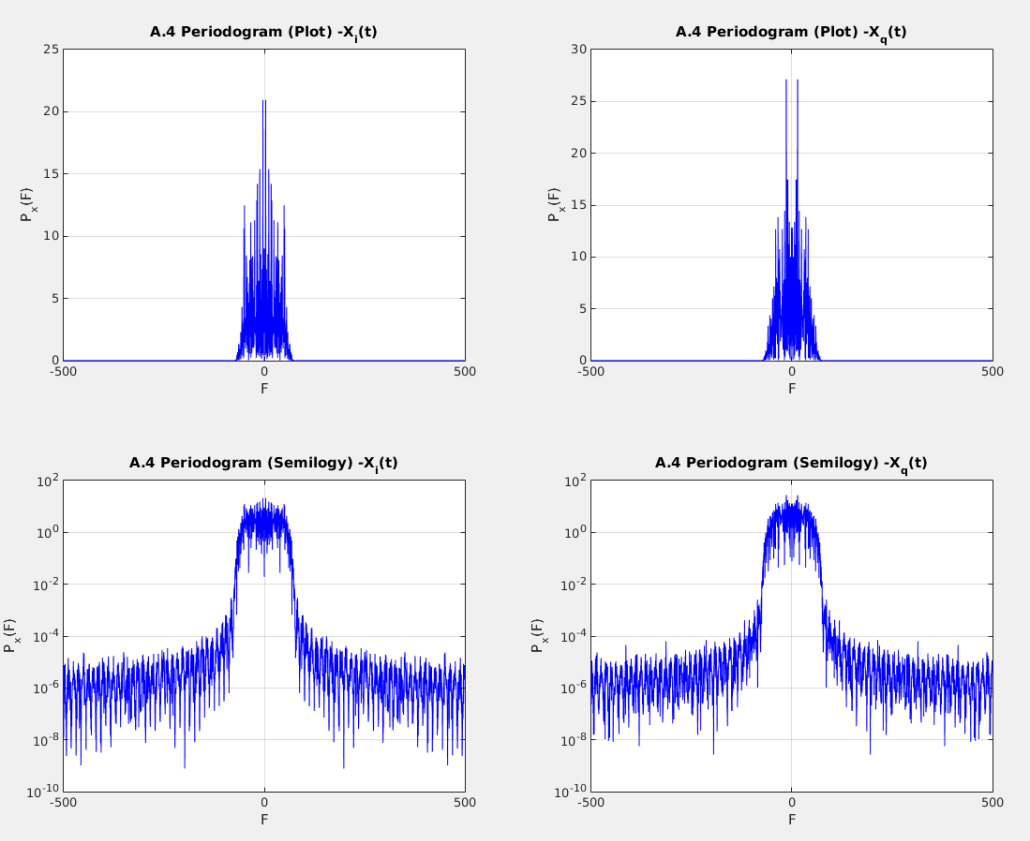

Figure 3: $X_I(t)$ `and` $X_Q(t)$ `periodograms`

A.5 $X_I^{mod}(t)$ and $X_Q^{mod}(t)$

Έπειτα, αυτό που ζητήθηκε ήταν να πολλαπλασιάσουμε τις κυμματομορφές $X_I(t)$ και $X_Q(t)$ με τους αντίστοιχους φορείς για $F_o = 200Hz$ (όπως διευκρινίστηκε στην διόρθωση μέσω e-mail) ώστε να δημιουργήσουμε τις κυμματομορφές $X_I^{mod}(t)$ και $X_Q^{mod}(t)$. Ουσιαστικά έπρεπε να ακολουθήσουμε τους μαθηματικούς τύπους της εκφώνησης, για την δημιουργία των δύο συνιστωσών της εισόδου του καναλιού.

Listing 6: A.5 `Calculate` $X_I^{mod}(t)$ `and` $X_Q^{mod}(t)$

```
% A.5
Fo = 200;
XI_mod =  2 * XI_t .* cos(2*pi*Fo*t_Xt);
XQ_mod = -2 * XQ_t .* sin(2*pi*Fo*t_Xt);
display_waveform_periodogram('A.5', 'X_i^{mod}', XI_mod, 'X_q^{mod}', XQ_mod, t_Xt,
    t_Xt, Ts, Nf)
```

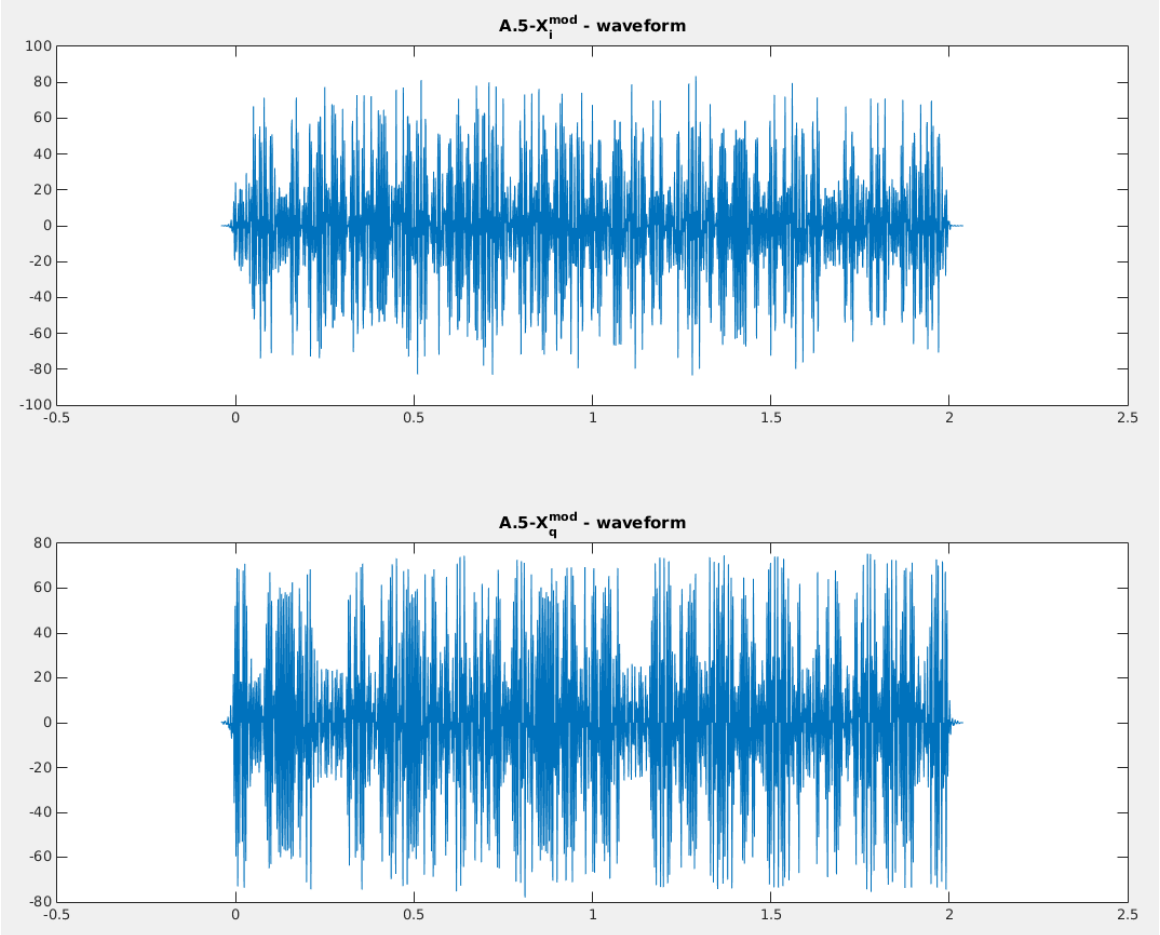

Figure 4: $X_I^{mod}(t)$ `and` $X_Q^{mod}(t)$ `waveforms`

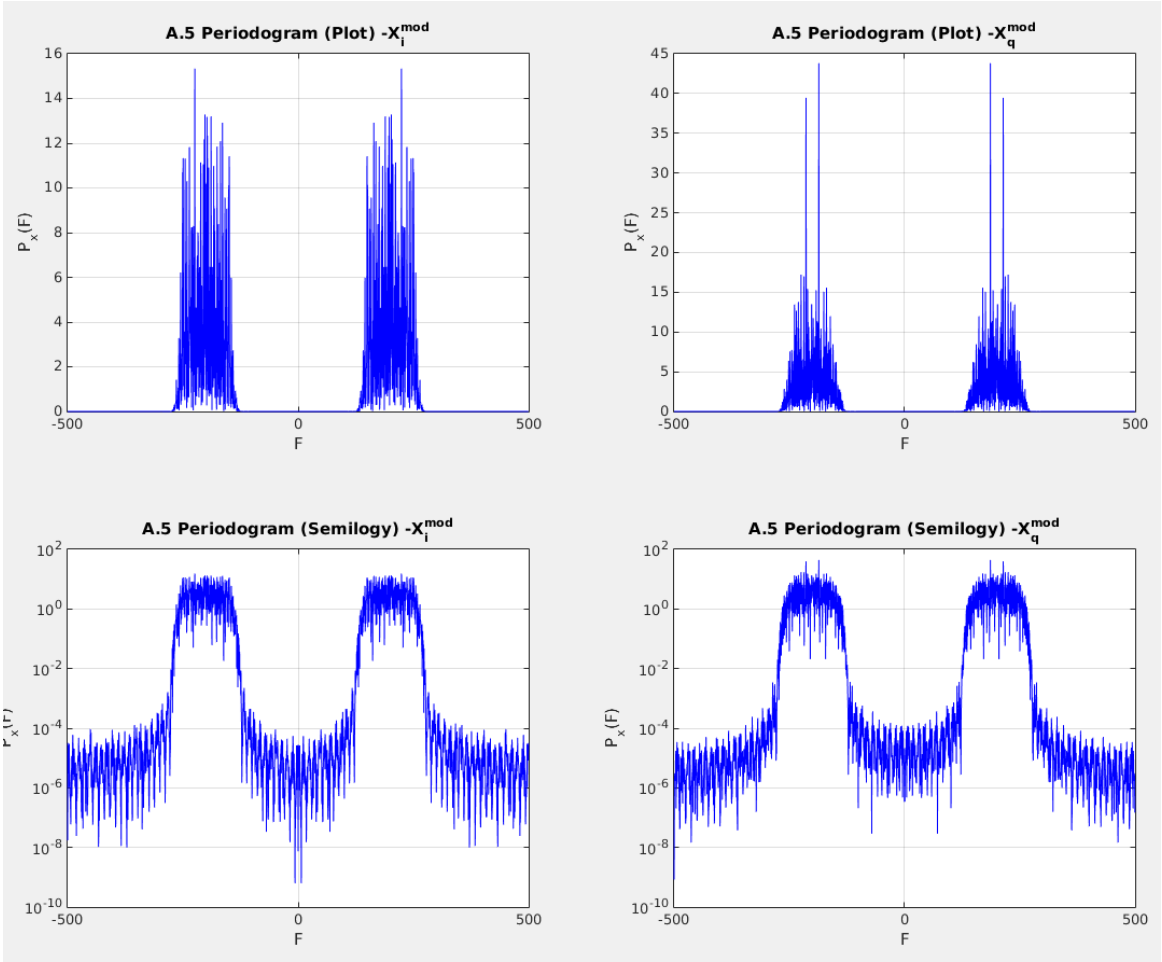

Figure 5: $X_I^{mod}(t)$ and $X_Q^{mod}(t)$ periodograms

Αυτό που μπορούμε να παρατηρήσουμε είναι ότι οι κυμματομορφές έχουν μεγαλύτερη συχνότητα, πιο συγκεκριμένα βλέπουμε τα περιοδογράμματα ότι έχουν μετακινηθεί γύρω από την συχνότητα μετάδοσης (Fo=200Hz).

A.6 $X^{mod}(t)$

Στην συνέχεια έπρεπε απλά να συνδιάσουμε τις δύο παραπάνω συνιστώσες για να δημιουργήσουμε την είσοδο του καναλιού, δηλαδή το σήμα $X^{mod}(t)$ όπως το ονομάζουμε. Αυτό το καταφέρνουμε με το να αθροίσουμε το Inphase και το Quadrature, ενώ παρακάτω έχουμε τα ζητούμενα figures.

Listing 7: A.6 Calculate $X^{mod}(t)$

```
% A.6
t_X_mod = t_Xt;
X_mod = XI_mod + XQ_mod;
display_waveform_periodogram('A.6', ' X_{mod} = X_i^{mod} + X_q^{mod} - Plot', X_mod,
    '', [], t_X_mod, [], Ts, Nf)
```

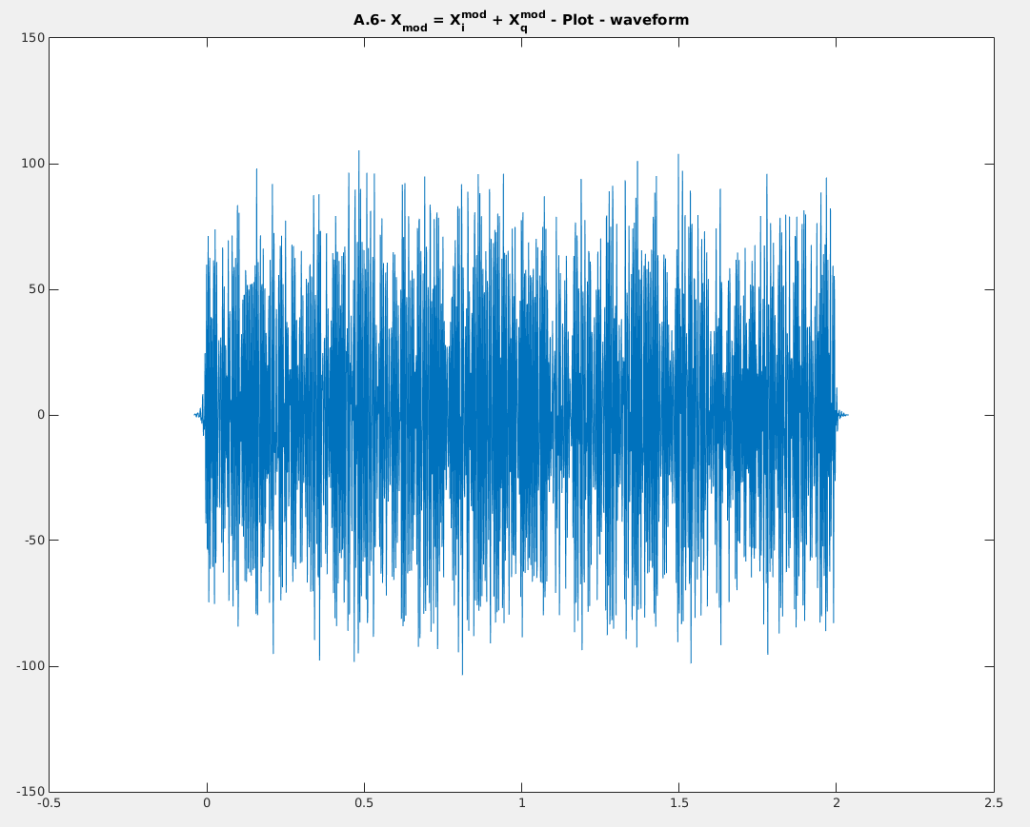

Figure 6: $X^{mod}(t)$ `waveform`

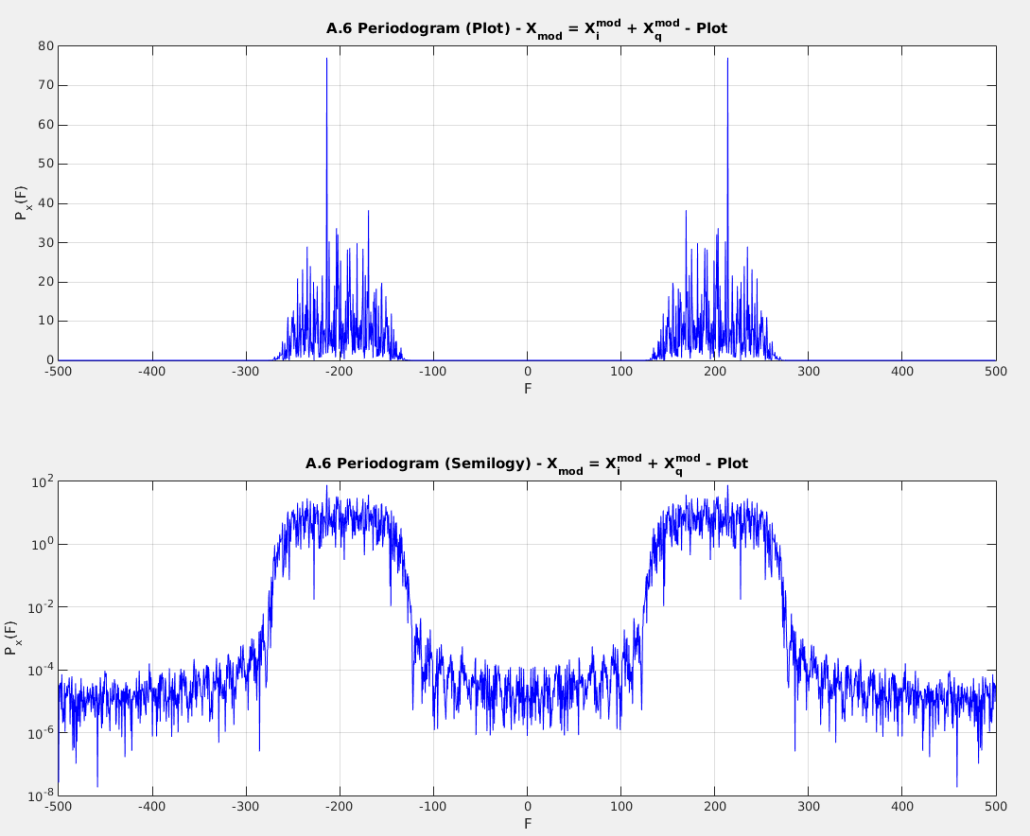

Figure 7: $X^{mod}(t)$ `periodogram`

Αυτό που μπορούμε να παρατηρήσουμε σε αυτήν την περίπτωση είναι ότι το φάσμα συνεχίζει να είναι γύρω από την συχνότητα μετάδοσης, με μόνη διαφορά ότι έχει αυξηθεί το πλάτος εξαιτίας της άθροισης.

A.7 Ιδανικό κανάλι

Αν είχαμε ιδανικό κανάλι, αυτό θα σήμαινε ότι δεν θα υπήρχε καμία αλλοίωσή στο σήμα κατά την μεταφορά. Δηλαδή δεν θα υπήρχε κάποια μεταβολή στο πλάτος ή χρονική μετατόπιση σε αυτό.

A.8 Gaussian White Noise

Αντίθετα με το ιδανικό κανάλι που σε ελάχιστες περιπτώσεις στην πραγματικότητα μπορούμε να το προσεγγίσουμε, πιο συχνό είναι να υπάρχει θόρυβος σε αυτό. Συνεπώς σε αυτό το ερώτημα δημιουργήσαμε White Gaussian Noise (WGN) με την διασπορά που δίνεται στην εκφώνηση της άσκησης με $SNR_{dB} = 10$ και το προσθέσαμε στην έξοδο του καναλιού.

Listing 8: A.8

```
% A.8
SNR = 10;
var_n = (10*A^2)/(Ts*(10^(SNR/10)));
WGN = sqrt(var_n)*randn(1, length(X_mod));
ch_sig = X_mod + WGN;
```

A.9 Received signals

Πλέον στην είσοδο του δέκτη είχαμε μία ενθόρυβη κυμματοφορφή, μέσα στην οποία περιεχόταν η πληροφορία που θέλαμε να μεταδώσουμε αρχικά. Πρώτο βήμα λοιπόν για την ανάκτηση της ήταν να την διακλαδώσουμε και να την πολλαπλασιάσουμε με τους αντίστοιχους φορείς $cos(2\pi Fot)$ και $-sin(2\pi Fot)$. Αυτό που θέλουμε να επιτύχουμε ουσιαστικά είναι να αποδιαμορφώσουμε το σήμα που έχουμε εκλάβει.

Listing 9: A.9

```
% A.9
ch_sig_I = ch_sig.*cos(2*pi*Fo*t_X_mod);
ch_sig_Q = ch_sig.*(-1*sin(2*pi*Fo*t_X_mod));
display_waveform_periodogram('A.9', 'Received I', ch_sig_I, 'Received Q', ch_sig_Q,
    t_X_mod, t_X_mod, Ts, Nf)
```

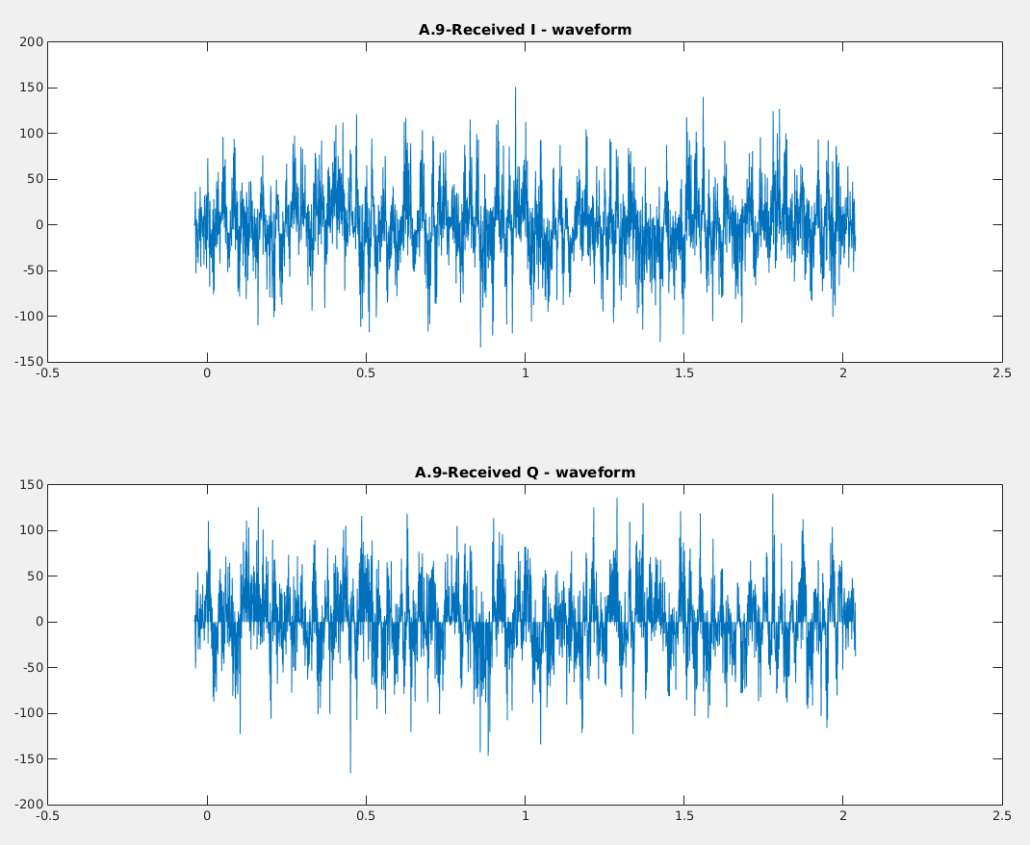

Figure 8: `Received signals waveforms`

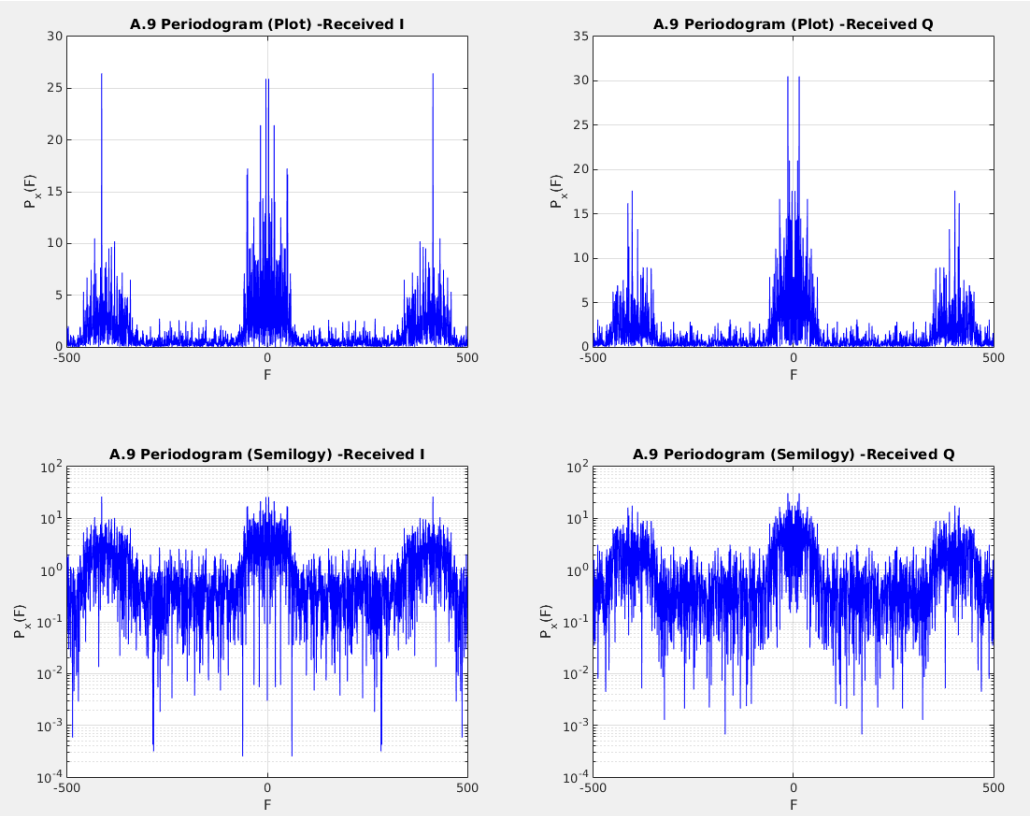

Figure 9: `Received signals periodograms`

Σημαντικό σε αυτό το σημείο είναι να παρατηρούμε ότι παρά την αποδιαμόρφωση, συνεχίζουν να υπάρχουν πλευρικοί λοβοί γύρω από την $2Fo$ άρα 400Hz. Θα πρέπει όμως μόλις περάσουν από τα προσαρμοσμένα φίλτρα να εξαφανιστούν.

A.10 $Y_I(t)$ and $Y_Q(t)$

Αφού είχαμε αποδιαμορφώσει και διακλαδώσει το σήμα εισόδου περάσαμε τα σήματα αυτά από τα προσαρμοσμένα φίλτρα SRRC, ουσιαστικά δηλαδή η συνέλιξη με αυτά. Ενώ φτιάξαμε κατάλληλα τον άξονα χρόνου όπως έχουμε ήδη αναφέρει στα προηγούμενα project.

Listing 10: A.10 Calculate $Y_I(t)$ and $Y_Q(t)$

```
% A.10
YI = conv(ch_sig_I,phi_t).*Ts;
YQ = conv(ch_sig_Q,phi_t).*Ts;
t_Xt_Rec = [t_X_mod(1) + t_phi(1) : Ts : t_X_mod(end) + t_phi(end)];
display_waveform_periodogram('A.10', 'Filtered I (Conv)', YI, 'Filtered Q (Conv)', YQ,
    t_Xt_Rec, t_Xt_Rec, Ts, Nf)
```

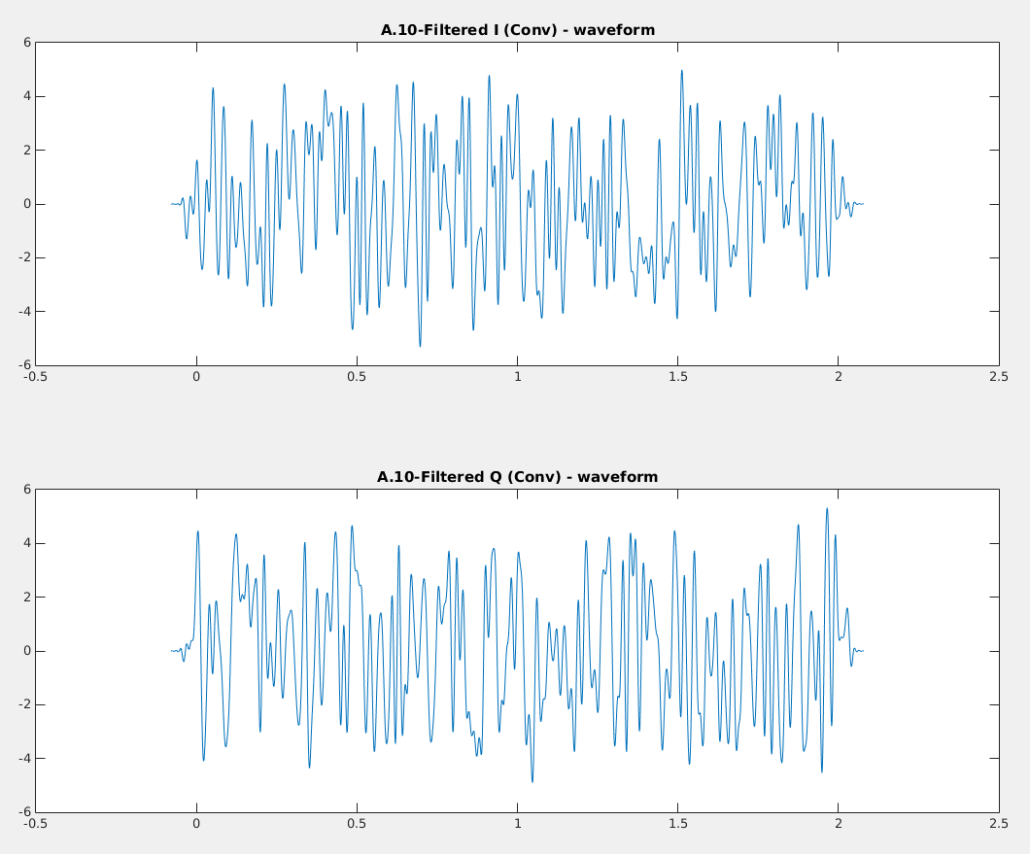

Figure 10: $Y_I(t)$ and $Y_Q(t)$ waveforms

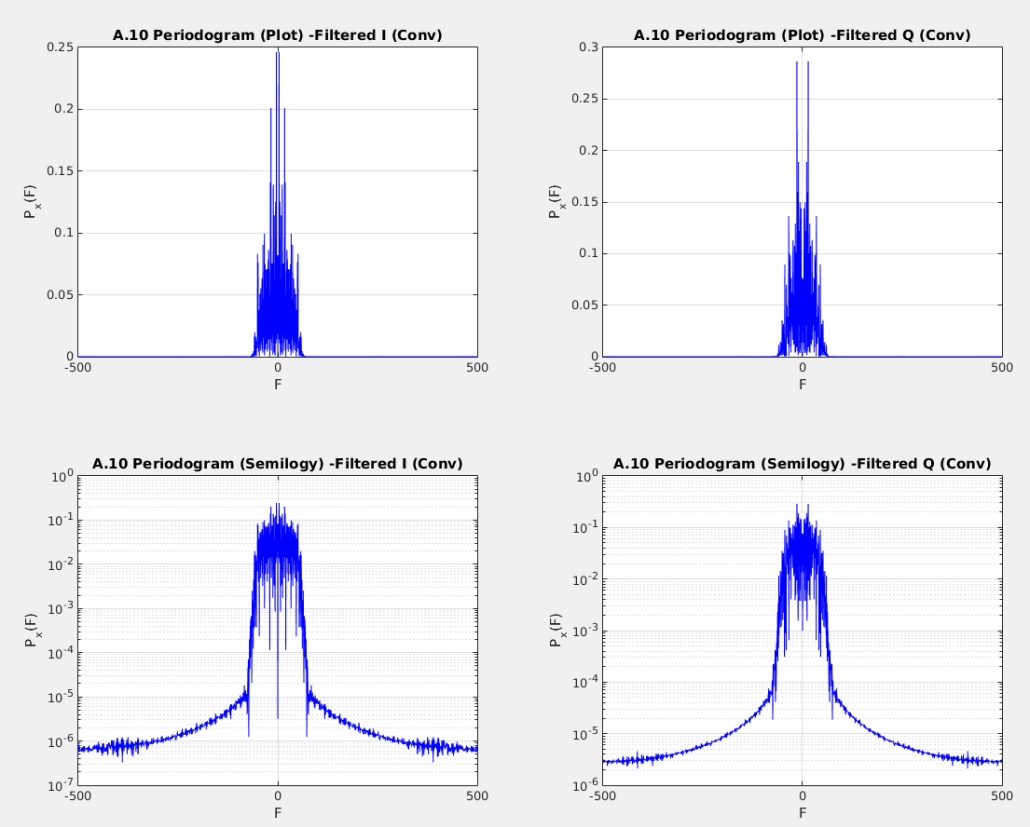

Figure 11: $Y_I(t)$ and $Y_Q(t)$ periodograms

Μπορούμε εύκολα να παρατηρήσουμε σε αυτό το σημείο, ότι έχουν αποκοπεί οι πλευρικοί λοβοί όπως ακριβώς περιμέναμε.

A.11 $\{Y_{I,k}\}$ and $\{Y_{Q,k}\}$

Έπειτα αυτό που χρειάστηκε να κάνουμε ήταν να δειγματοληπτήσουμε την έξοδο από τα προσαρμοσμένα φίλτρα στις κατάλληλες χρονικές στιγμές και να σχεδιάσουμε την ακολουθία εξόδου χρησιμοποιώντας την εντολή scatterplot. Για την δειγματοληψία έχει ακολουθηθεί ο τρόπος που περιγράφηκε στο φροντιστήριο του μαθήματος και αυτό που ουσιαστικά κάνουμε είναι να αποκόψουμε τα πλευρικά σημεία πριν πάρουμε αυτά που μας ενδιαφέρουν.

Listing 11: A.11 Calculate $\{Y_{I,k}\}$ and $\{Y_{Q,k}\}$

```
% A.11
YI_sampled = YI(2*A_s*over+1:over:2*A_s*over+1+N*over);
YQ_sampled = YQ(2*A_s*over+1:over:2*A_s*over+1+N*over);
figure() ; scatter(YI_sampled, YQ_sampled); grid on ; title('A.11 Sampled');
```

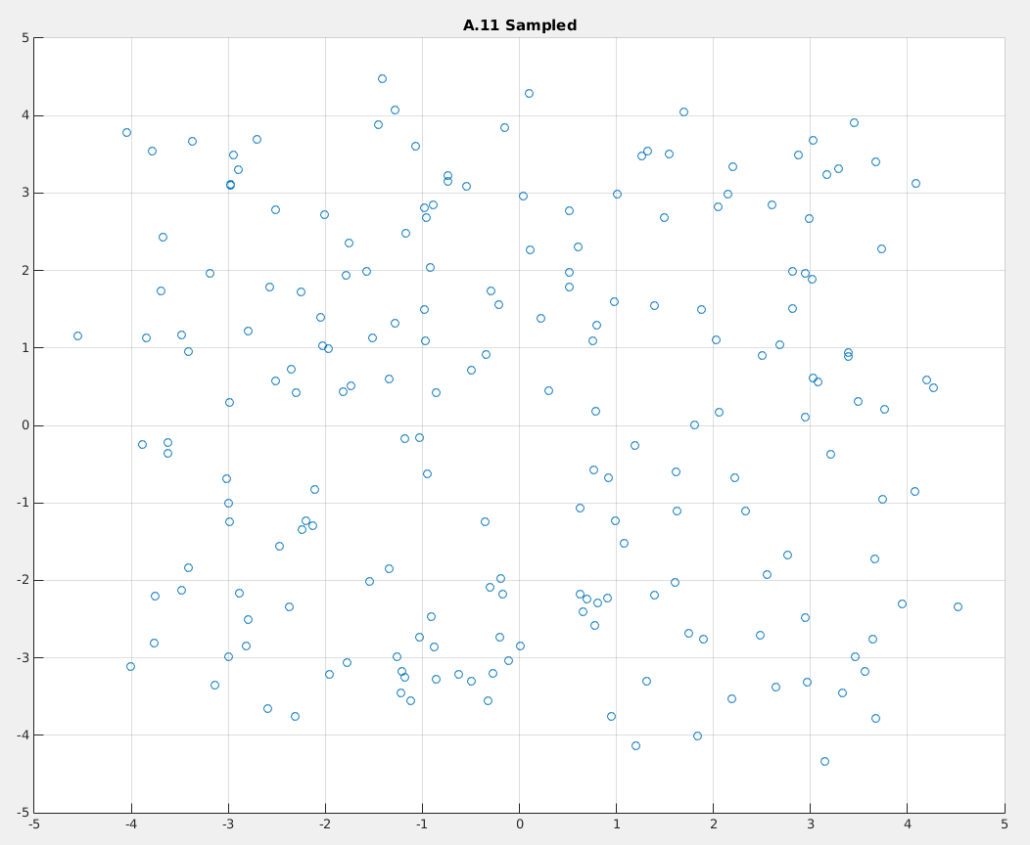

Figure 12: `Scatterplot {`$Y_{I,k}$`} and {`$Y_{Q,k}$`} for` $SNR_{dB} = 10$

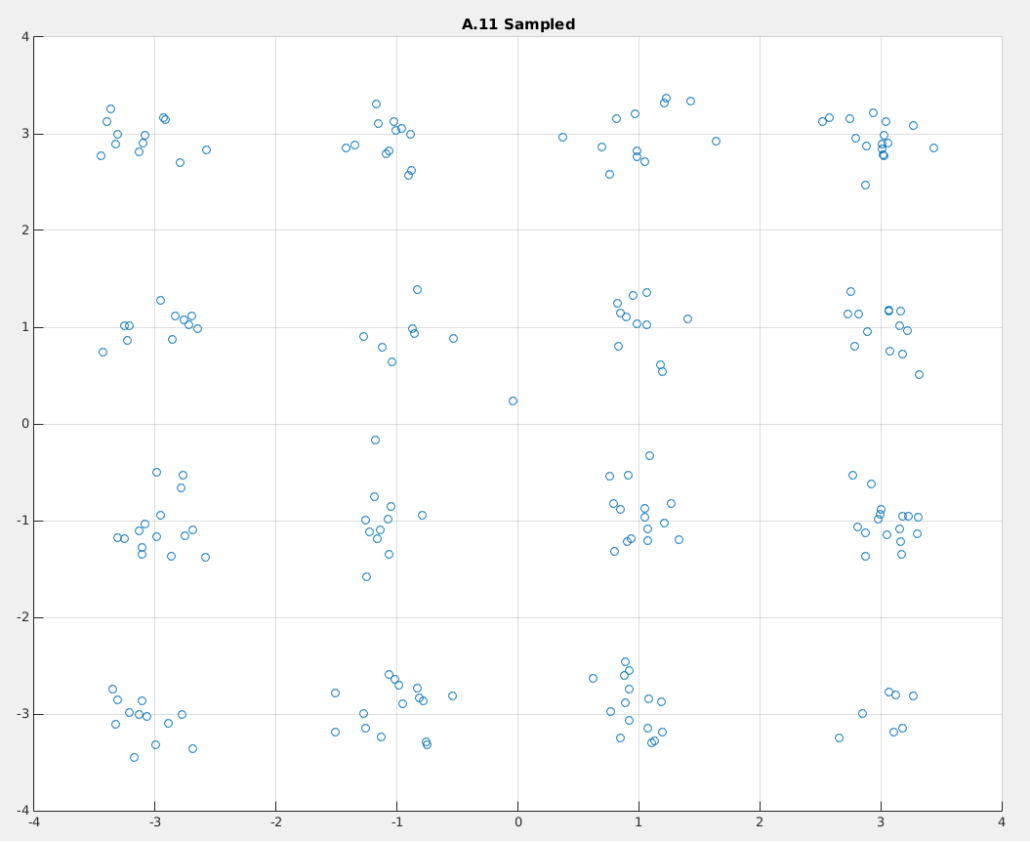

Figure 13: `Scatterplot {`$Y_{I,k}$`} and {`$Y_{Q,k}$`} for` $SNR_{dB} = 20$

Παραπάνω παρουσιάζεται το scatterplot και για $SNR_{dB} = 20$ καθώς είναι πιο κοντά σε αυτό που θα περιμέναμε να δούμε, όμως κρατήθηκε η υλοποίηση για $SNR_{dB} = 10$ ώστε να έχουμε περισσότερα σφάλματα και να μπορούμε να δούμε αποτελέσματα και για τα επόμενα ερωτήματα.

A.12 Estimated symbols

Πλέον έπρεπε να δημιουργήσουμε συνάρτηση η οποία θα προέβλεπε πιο σύμβολο έχει σταλθεί χρησιμοποιώντας τον κανόνα εγγύτερου γείτονα (λόγω των ισοπίθανων συμβόλων) για να αποφασίσει για την ακολουθία εισόδου 4-PAM σύμβολο-προς-σύμβολο, και να την εφαρμόσουμε στα δείγματα του inphase και του quadrature. Ουσιαστικά πραγματοποιήθηκαν και σε αυτό το σημείο απλές συνθήκες ελέγχου για να επιτευχθεί αυτό.

Listing 12: `detect_4_PAM()`

```
function [est_X] = detect_4_PAM(Y, A)
    est_X=zeros(1,length(Y));
    for i=1:length(Y)
        if(Y(i) > 3*A || (Y(i) > 2*A && Y(i) < 3*A))
            est_X(i) = 3*A;                              % -3 | -1 | +1 -> +3
        elseif((Y(i) > 0 && Y(i) < 1*A) || (Y(i) > 1*A && Y(i) < 2*A))
            est_X(i) = 1*A;                              % -3 | -1 -> +1 <- +3
        elseif((Y(i) < 0 && Y(i) > -1*A) || (Y(i) < -1*A && Y(i) > -2*A))
            est_X(i) = -1*A;                             % -3 -> -1 <- +1 | +3
        elseif(Y(i) < -3*A || (Y(i) < -2*A && Y(i) > -3*A))
            est_X(i) = -3*A;                             % -3 <- -1 | +1 | +3
        end
    end
end
```

Ενώ για να την εφαρμόσουμε στα δείγματα που inphase και του quadrature παρουσιάζεται παρακάτω τι έπρεπε να κάνουμε.

Listing 13: A.12 `Setect Symbols`

```
% A.12
YI_est = detect_4_PAM(YI_sampled, A);
YQ_est = detect_4_PAM(YQ_sampled, A);
figure() ; scatter(YI_est, YQ_est) ; grid on; title('A.12 Estimations');
```

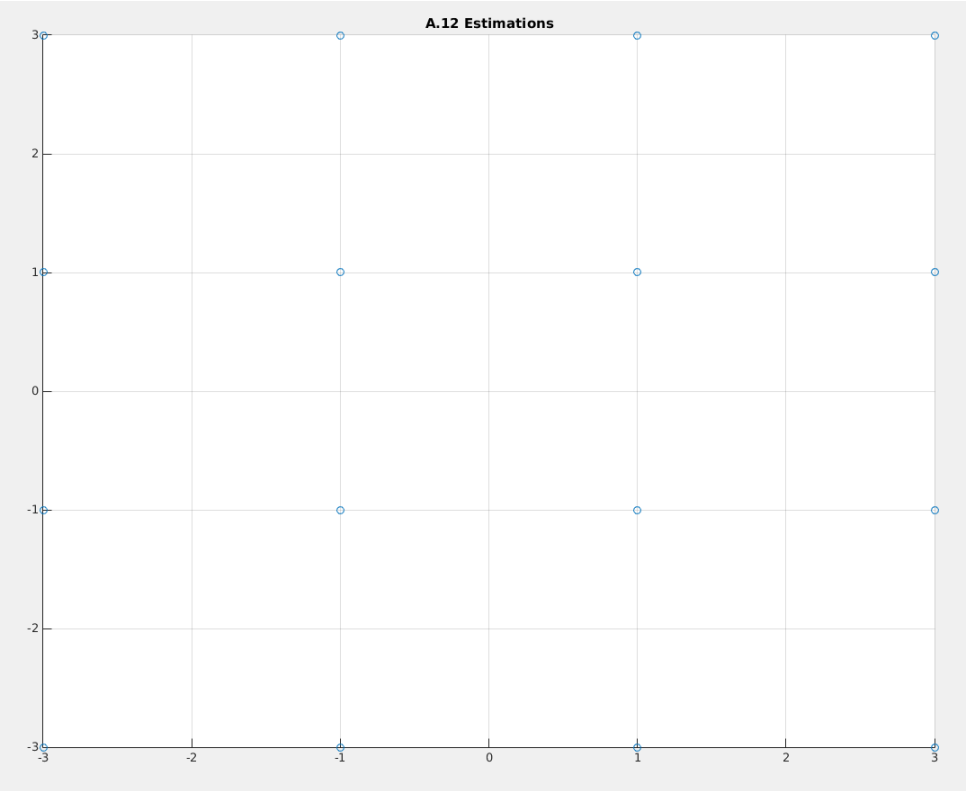

Figure 14: `Estimations (Without Noise)`

A.13 Symbols errors

Στην συνέχεια χρειάστηκε χρησιμοποιώντας τις ακολουθίες εισόδου και τις αποφάσεις να υπολογιστεί ο αριθμός σφαλμάτων απόφασης συμβόλου για τον αστερισμό 16-QAM. Και αυτό έγινε με απλές συνθήκες ελέγχου για ένα προς ένα τα σύμβολα του inphase και του quadrature, με αντίστοιχους μετρητές όταν διαφέρει η είσοδος του καθενός με την ανάλογη απόφαση. Ενώ στο τέλος έγινε άθροιση των δύο αυτών μετρητών για να υπολογιστεί το σύνολο των λαθών. Όμως προκειμένου να γίνει και οπτική απεικόνιση δημιουργήθηκαν τα arrays Err_I και Err_Q τα οποία σε κάθε σημείο που δεν υπάρχει κάποιο λάθος είναι κενό ενώ μόνο στα λάθος σημεία κρατείται η τιμή της λάθος απόφασης (**εσωτερική στοίχιση επειδή δεν ήταν κάτι που ζητήθηκε**).

Listing 14: A.13 `Symbols errors`

```
% A.13
YI_est = YI_est(1:end-1); YQ_est = YQ_est(1:end-1);
I_err = 0;                Err_I=zeros(1,length(YI_est)); Err_I(Err_I==0)=nan;
Q_err = 0;                Err_Q=zeros(1,length(YQ_est)); Err_Q(Err_Q==0)=nan;
for i=1:N
    if(YI_est(i) ~= XI_n(i))
        I_err = I_err + 1;
                          Err_I(i) = YI_est(i);
    end
    if(YQ_est(i) ~= XQ_n(i))
        Q_err = Q_err + 1;
                         Err_Q(i) = YQ_est(i);
    end
end
IQ_err = Q_err + I_err;
disp(['A.13: SER: ', num2str(IQ_err), '/', num2str(N*2)]);
```

A.14 4-PAM to bits

Το μόνο που έμενε για την ανάκτηση της πληροφορίες ήταν να δημιουργήσουμε συνάρτηση η οποία θα χρησιμοποιούσε την αντίστροφη απεικόνιση Gray, ώστε να μετατρέψει τα σύμβολα που είχαμε πάρει από τον κανόνα απόφασης σε δυαδική ακολουθία από bits. Για ακόμα μία φορά αυτό επιτεύχθηκε με χρήση απλών συνθηκών, έπρεπε όμως να προσέξουμε ο κανόνας Gray και είναι αντίστοιχος με αυτόν που είχαμε χρησιμοποιήσει στην bits_to_4_PAM().

Listing 15: `PAM_4_to_bits()`

```
function [est_bit] = PAM_4_to_bits(X, A)
    k=1;
    est_bit=zeros(1,2*length(X));
    for i=1:length(X)
        if(X(i) == 3*A)         % +3 -> 00
            est_bit(k) = 0;
            est_bit(k+1) = 0;
        elseif(X(i) == 1*A)     % +1 -> 01
            est_bit(k) = 0;
            est_bit(k+1) =  1;
        elseif(X(i) == -1*A)    % -1 -> 11
            est_bit(k) = 1;
            est_bit(k+1) = 1;
        elseif(X(i) == -3*A)    % -3 -> 10
            est_bit(k) = 1;
            est_bit(k+1) = 0;
        end
        k=k+2;
    end
end
```

Ενώ απλά χρειάστηκε να την καλέσουμε με τον εξής τρόπο.

Listing 16: A.14 `4-PAM to bits`

```
% A.14
est_bit_XI = PAM_4_to_bits(YI_est, A);
est_bit_XQ = PAM_4_to_bits(YQ_est, A);
est_bit_X = [est_bit_XI est_bit_XQ];
```

A.15 Bits Errors

Αφού είχαμε μεταφέρει και ανακτήσει στον δέκτη την πληροφορία που στείλαμε, χρειάστηκε τέλος να υπολογίσουμε τον αριθμό σφαλμάτων σε bit. Ο τρόπος με τον οποίο το κάναμε αυτό είναι σε πλήρη αναλογία με αυτόν για τα σύμβολα, ενώ και σε αυτήν την περίπτωση για την οπτική απεικόνιση που υπάρχει στην συνέχεια υπάρχει σε εσωτερική στοίχιση ο πίνακας Err_b για τον ίδιο λόγω με τους πίνακες που αναφέρθηκαν νωρίτερα.

Listing 17: A.15

```
% A.15
ber = 0;                Err_b=zeros(1,length(bit_seq)); Err_b(Err_b==0)=nan;
for i=1:length(bit_seq)
    if(bit_seq(i) ~= est_bit_X(i))
        ber = ber + 1;
                        Err_b(i) = bit_seq(i);
    end
end
disp(['A.15: BER: ', num2str(ber), '/', num2str(N*4)]);
```

```
A.13: SER: 53/400
A.15: BER: 53/800
```

Figure 15: `Matlab output sample for` $SNR_{dB} = 10$

Στο παρακάτω figure μπορούμε να δούμε (με την χρήση των arrays που αναφέρθηκαν νωρίτερα) σε ποια ακριβώς σύμβολα και bit έχουν υπάρξει σφάλματα κατά την μεταφορά, ώστε να επαληθεύσουμε την ορθότητα των ερωτημάτων.

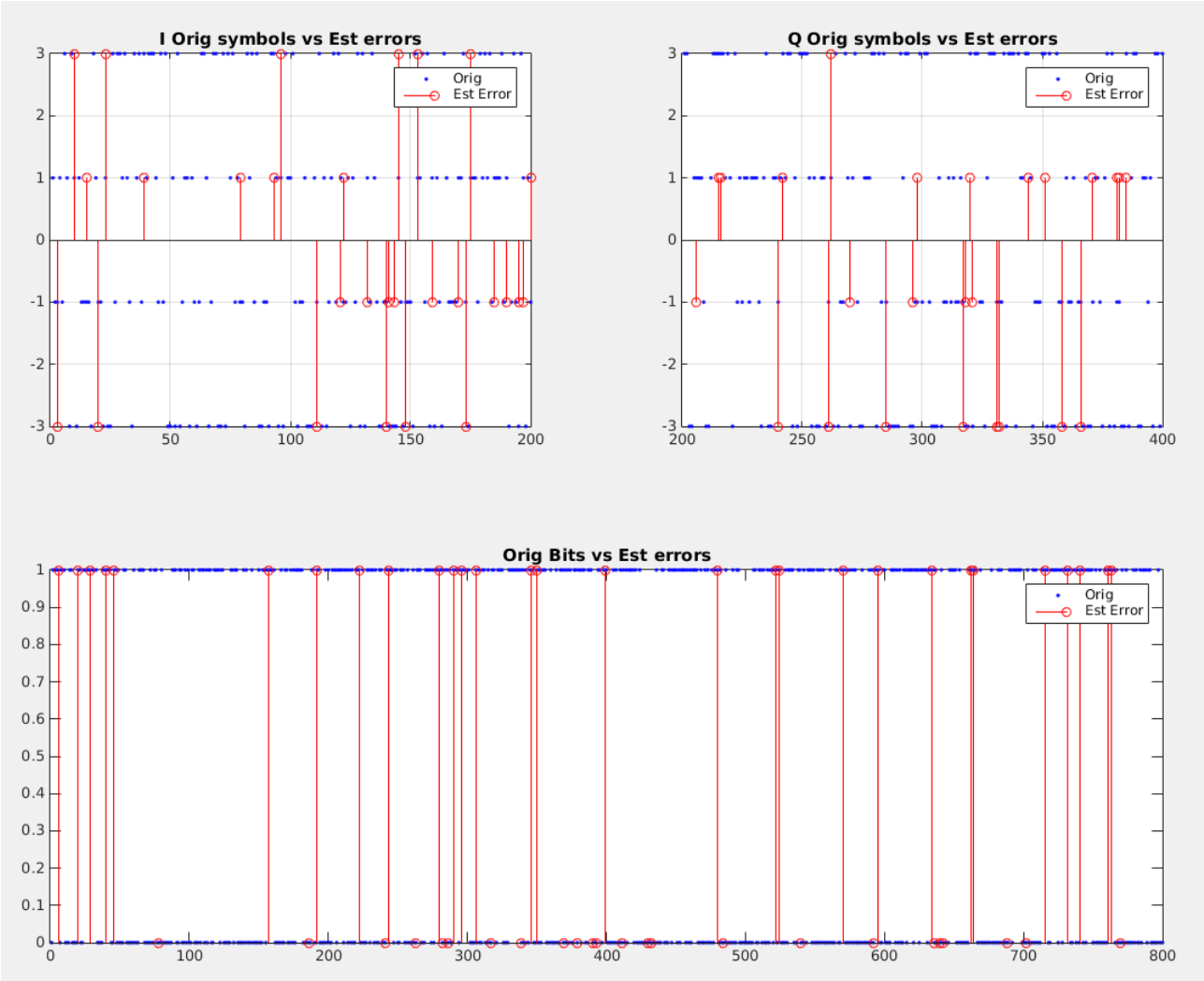

Figure 16: `Error Symbols/Bits Vs Estimations`

Code:

Listing 18: `bits_to_4_PAM.m`

```
function [ X ] = bits_to_4_PAM(bit_seq, A)
    k=1;
    X=zeros(1,length(bit_seq)/2);
    for i=1:2:length(bit_seq)
        if(bit_seq(i)==0 && bit_seq(i+1)==0)        % 00 -> +3
            X(k) = 3*A;
        elseif(bit_seq(i)==0 && bit_seq(i+1)==1)    % 01 -> +1
            X(k) = 1*A;
        elseif(bit_seq(i)==1 && bit_seq(i+1)==1)    % 11 -> -1
            X(k) = -1*A;
        elseif(bit_seq(i)==1 && bit_seq(i+1)==0)    % 10 -> -3
            X(k) = -3*A;
        end
        k=k+1;
    end
end
```

Listing 19: `detect_4_PAM.m`

```
function [est_X] = detect_4_PAM(Y, A)
    est_X=zeros(1,length(Y));
    for i=1:length(Y)
        if(Y(i) > 3*A || (Y(i) > 2*A && Y(i) < 3*A))                        % -3 | -1
            | +1 -> +3
            est_X(i) = 3*A;
        elseif((Y(i) > 0 && Y(i) < 1*A) || (Y(i) > 1*A && Y(i) < 2*A))      % -3 | -1
           -> +1 <- +3
            est_X(i) = 1*A;
        elseif((Y(i) < 0 && Y(i) > -1*A) || (Y(i) < -1*A && Y(i) > -2*A))   % -3 -> -1
             <- +1 | +3
            est_X(i) = -1*A;
        elseif(Y(i) < -3*A || (Y(i) < -2*A && Y(i) > -3*A))                 % -3 <- -1
             | +1 | +3
            est_X(i) = -3*A;
        end
    end
end
```

Listing 20: `fourier_transform.m`

```
function [X_F, F_X] = fourier_transform(Xt, Ts, Nf)
    Fs = 1/Ts;
    X_F = fftshift(fft(Xt,Nf)*Ts);
    F_X = [-Fs/2 : Fs/Nf : Fs/2-Fs/Nf];
end
```

Listing 21: `periodogram.m`

```
function [Px_F, F_Px] = periodogram(Xt, t_Xt, Ts, Nf)
```

```
    Ttotal = length(t_Xt)*Ts;
    [X_F, F_Px] = fourier_transform(Xt, Ts, Nf);
    Px_F = (abs(X_F).^2)./Ttotal;
end
```

Listing 22: PAM_4_to_bits.m

```
function [est_bit] = PAM_4_to_bits(X, A)
    k=1;
    est_bit=zeros(1,2*length(X));
    for i=1:length(X)
        if(X(i) == 3*A)          % +3 -> 00
            est_bit(k) = 0;
            est_bit(k+1) = 0;
        elseif(X(i) == 1*A)      % +1 -> 01
            est_bit(k) = 0;
            est_bit(k+1) =  1;
        elseif(X(i) == -1*A)     % -1 -> 11
            est_bit(k) = 1;
            est_bit(k+1) = 1;
        elseif(X(i) == -3*A)     % -3 -> 10
            est_bit(k) = 1;
            est_bit(k+1) = 0;
        end
        k=k+2;
    end
end
```

Listing 23: display_waveform_periodogram.m

```
function [] = display_waveform_periodogram(part, I_text, I_wav, Q_text, Q_wav, t_Xt_I,
     t_Xt_Q, Ts, Nf)
    % Plot waveforms
    if (~isempty(Q_text));
        figure()
        subplot(2,1,1) ; plot(t_Xt_I, I_wav); title(strcat(part, '-', I_text,' -
            waveform'));
        subplot(2,1,2) ; plot(t_Xt_Q, Q_wav); title(strcat(part, '-', Q_text,' -
            waveform'));
    else
        figure() ; plot(t_Xt_I, I_wav); title(strcat(part, '-', I_text,' - waveform'))
            ;
    end;

    %Periodogram
    [Px_F_I, F_I] = periodogram(I_wav, t_Xt_I, Ts, Nf);
    [Px_F_Q, F_Q]= periodogram(Q_wav, t_Xt_Q, Ts, Nf);

    % Plot Periodograms
    if (~isempty(Q_text));
        figure()
        subplot(2,2,1); plot(F_I, Px_F_I, 'b') ; grid on; title(strcat(part, '
            Periodogram (Plot) -', I_text)) ; xlabel('F') ; ylabel('P_x(F)');
```

```
        subplot(2,2,2); plot(F_Q, Px_F_Q, 'b') ; grid on; title(strcat(part, '
            Periodogram (Plot) -', Q_text)) ; xlabel('F') ; ylabel('P_x(F)');
        subplot(2,2,3); semilogy(F_I, Px_F_I, 'b') ; grid on; title(strcat(part, '
            Periodogram (Semilogy) -', I_text)) ; xlabel('F') ; ylabel('P_x(F)');
        subplot(2,2,4); semilogy(F_Q, Px_F_Q, 'b') ; grid on; title(strcat(part, '
            Periodogram (Semilogy) -', Q_text)) ; xlabel('F') ; ylabel('P_x(F)');
    else
        figure()
        subplot(2,1,1); plot(F_I, Px_F_I, 'b') ; grid on; title(strcat(part, '
            Periodogram (Plot) -', I_text)) ; xlabel('F') ; ylabel('P_x(F)');
        subplot(2,1,2); semilogy(F_I, Px_F_I, 'b') ; grid on; title(strcat(part, '
            Periodogram (Semilogy) -', I_text)) ; xlabel('F') ; ylabel('P_x(F)');
    end;
end
```

Listing 24: `part_a.m`

```
% ------------------------------------------------------------------------------
%   Exercise 3, part A
%
%   Authors : Spyridakis Christos
%   Created Date : 15/12/2019
%   Last Updated : 19/12/2019
%
%   Description:
%               Code created for Exercises of Communication Systems Course
%               in Tecnhical University of Crete
% ------------------------------------------------------------------------------

clear all ; close all ; clc ;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.1
N = 200;                          % Random bits E.g.
bit_seq = (sign(randn(4*N, 1)) + 1)/2; % 0 1 1 0 . . .

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.2
A = 1;                           % Bits to 4 Pam E.g
Xn = bits_to_4_PAM(bit_seq, A);  % +1 -3 -1 -1 +3 .

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.3
XI_n = Xn(1:N);                     % In Phase Symbols
XQ_n = Xn(N+1:2*N);                 % Quadrature Symbols

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.4
T = 0.01 ; over = 10 ; Ts = T/over ; A_s = 4 ; a = 0.5;
Nf = 2048 ; Fs = 1/Ts ; F = [-Fs/2 : Fs/Nf : Fs/2-Fs/Nf]; % Frequency vector

% Phi
[phi_t t_phi] = srrc_pulse(T, Ts, A_s, a);

```

```matlab
% Create upsampled X_delta signals and using it calculate conv
XI_d = 1/Ts * upsample(XI_n, over) ; XI_t = conv(XI_d, phi_t).*Ts ;
XQ_d = 1/Ts * upsample(XQ_n, over) ; XQ_t = conv(XQ_d, phi_t).*Ts ;
td = [ 0 : Ts : (N*over-1)*Ts ] ; t_Xt = [td(1) + t_phi(1) : Ts : td(end) + t_phi(end) ];

% Plot waveforms
figure()
subplot(4,2,1:2) ; stem([1:N*4], bit_seq, 'b') ; title('A.1 Random Bits');
subplot(4,2,3:4) ; stem([1:N*2], Xn, 'r') ; title('A.2 Symbols in 4-PAM');
subplot(4,2,5) ; stem([1:N], XI_n, 'r'); title('A.3 \{X_{I,n}\} - (Xn symbols of In-phase)'); subplot(4,2,6) ; stem([N+1:N*2], XQ_n, 'r'); title('A.3 \{X_{Q,n}\} - (Xn symbols of Quadrature)');
subplot(4,2,7) ; plot(t_Xt, XI_t); xlim([-0.1 2.1]) ; ylim([-50 50]) ; title('A.4 X_I (t)'); subplot(4,2,8) ; plot(t_Xt, XQ_t) ; xlim([-0.1 2.1]) ; ylim([-50 50]) ; title('A.4 X_Q (t)');

display_waveform_periodogram('A.4', 'X_i(t)', XI_t, 'X_q(t)', XQ_t, t_Xt, t_Xt, Ts, Nf)

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.5
Fo = 200;
XI_mod =  2 * XI_t .* cos(2*pi*Fo*t_Xt);
XQ_mod = -2 * XQ_t .* sin(2*pi*Fo*t_Xt);
display_waveform_periodogram('A.5', 'X_i^{mod}', XI_mod, 'X_q^{mod}', XQ_mod, t_Xt, t_Xt, Ts, Nf)

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.6
t_X_mod = t_Xt;
X_mod = XI_mod + XQ_mod;
display_waveform_periodogram('A.6', ' X_{mod} = X_i^{mod} + X_q^{mod} - Plot', X_mod, '', [], t_X_mod, [], Ts, Nf)

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.7
% On report

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.8
SNR = 10;
var_n = (10*A^2)/(Ts*(10^(SNR/10)));
WGN = sqrt(var_n)*randn(1, length(X_mod));
ch_sig = X_mod + WGN;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.9
ch_sig_I = ch_sig.*cos(2*pi*Fo*t_X_mod);
ch_sig_Q = ch_sig.*(-1*sin(2*pi*Fo*t_X_mod));
display_waveform_periodogram('A.9', 'Received I', ch_sig_I, 'Received Q', ch_sig_Q, t_X_mod, t_X_mod, Ts, Nf)
```

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.10
YI = conv(ch_sig_I,phi_t).*Ts;
YQ = conv(ch_sig_Q,phi_t).*Ts;
t_Xt_Rec = [t_X_mod(1) + t_phi(1) : Ts : t_X_mod(end) + t_phi(end)];
display_waveform_periodogram('A.10', 'Filtered I (Conv)', YI, 'Filtered Q (Conv)', YQ,
     t_Xt_Rec, t_Xt_Rec, Ts, Nf)

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.11
YI_sampled = YI(2*A_s*over+1:over:2*A_s*over+1+N*over);
YQ_sampled = YQ(2*A_s*over+1:over:2*A_s*over+1+N*over);
figure() ; scatter(YI_sampled, YQ_sampled); grid on ; title('A.11 Sampled');

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.12
YI_est = detect_4_PAM(YI_sampled, A);
YQ_est = detect_4_PAM(YQ_sampled, A);
figure() ; scatter(YI_est, YQ_est) ; grid on; title('A.12 Estimations');

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.13
YI_est = YI_est(1:end-1); YQ_est = YQ_est(1:end-1);
I_err = 0; Err_I=zeros(1,length(YI_est)); Err_I(Err_I==0)=nan;
Q_err = 0; Err_Q=zeros(1,length(YQ_est)); Err_Q(Err_Q==0)=nan;
for i=1:N
    if(YI_est(i) ~= XI_n(i))
        I_err = I_err + 1;
        Err_I(i) = YI_est(i);
    end
    if(YQ_est(i) ~= XQ_n(i))
        Q_err = Q_err + 1;
        Err_Q(i) = YQ_est(i);
    end
end
IQ_err = Q_err + I_err;
disp(['A.13: SER: ', num2str(IQ_err), '/', num2str(N*2)]);

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.14
est_bit_XI = PAM_4_to_bits(YI_est, A);
est_bit_XQ = PAM_4_to_bits(YQ_est, A);
est_bit_X = [est_bit_XI est_bit_XQ];

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A.15
ber = 0; Err_b=zeros(1,length(bit_seq)); Err_b(Err_b==0)=nan;
for i=1:length(bit_seq)
    if(bit_seq(i) ~= est_bit_X(i))
        ber = ber + 1;
        Err_b(i) = bit_seq(i);
    end
end
```

```
134 disp(['A.15: BER: ', num2str(ber), '/', num2str(N*4)]);
```