Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Πανεπιστήμιο Πατρών
πολυτεχνική σχολή

Τομέας Υλικού και Αρχιτεκτονικής των Υπολογιστών
Διδάσκων: Κωνσταντίνος Μπερμπερίδης
Ακαδημαϊκό Έτος: 2024 – 2025
Ημ/νία Παράδοσης: 16/02/2025

Ψηφιακές Τηλεπικοινωνίες

2ο σετ Εργαστηριακών Ασκήσεων

Στοιχεία Φοιτητή:

Ονοματεπώνυμο:	Σωκράτης Μαντές
Α.Μ.:	1093421
E – mail:	up1093421@ac.upatras.gr
Εξάμηνο:	7

ΜΕΡΟΣ Α: Περιγραφή Τηλεπικοινωνιακού Συστήματος

1. Φίλτρα Πομπού-Δέκτη

Σε αυτό το βήμα θα πρέπει να δημιουργήσουμε ένα συμμετρικό SRRC φίλτρο το οποίο θα μπορεί να χρησιμοποιηθεί τόσο για διαμόρφωση όσο και για αποδιαμόρφωση ενός ψηφιακού σήματος. Αυτό το φίλτρο είναι πολύ σημαντικό στην ψηφιακή επικοινωνία, καθώς διαμορφώνει παλμούς με βέλτιστο τρόπο ώστε να μην παρεμβάλλονται μεταξύ τους, μειώνοντας την Διασυμβολική Παρεμβολή, δηλαδή όταν τα σύμβολα επικαλύπτονται μεταξύ τους, προκαλώντας δυσκολίες στην ορθή αποκωδικοποίηση του λαμβανόμενου σήματος. Συγκεκριμένα, ομαλοποιεί τις μεταβάσεις μεταξύ συμβόλων εξαλείφοντας τις απότομες αλλαγές, περιορίζει το εύρος ζώνης (το σήμα καταλαμβάνει λιγότερο bandwidth, άρα είναι πιο αποδοτικό.) κα διατηρεί την ιδιότητα Nyquist, άρα η παρεμβολή μεταξύ συμβόλων μηδενίζεται στα σημεία δειγματοληψίας.
Το φίλτρο που υλοποιείται θα κανονικοποιηθεί ώστε να έχει συνολική ενέργεια ίση με 1, ώστε να διατηρείται σταθερή η ισχύς του σήματος. Για αρχή, ορίζουμε τις παραμέτρους του φίλτρου. Πρώτα, τον παράγοντα roll-off (β) ίσο με 0.3, ο οποίος καθορίζει το εύρος ζώνης του φίλτρου. Μικρότερο roll_off σημαίνει πιο απότομο φίλτρο. Μετά, ορίζουμε τον αριθμό των περιόδων συμβόλων (symbol periods) που καλύπτει το φίλτρο αλλά και τον αριθμό των δειγμάτων ανά σύμβολο για την υπερδειγματοληψία.

```
clc; clear; close all;

%% Βήμα 1: Δημιουργία Φίλτρων Πομπού-Δέκτη (SRRC Filter)
roll_off = 0.3;       % Παράγοντας αναδίπλωσης (Roll-off factor)
span = 6;             % Αριθμός περιόδων σηματοδοσίας (Ts)
sps = 4;              % Υπερδειγματοληψία (samples per symbol)
```

Έπειτα, δημιουργείται ένας διακριτός χρονικός άξονας t για το φίλτρο SRRC, ο οποίος εκτείνεται από -span/2 έως span/2 σε βήματα 1/sps και στην συνέχεια το υπολογίζουμε λαμβάνοντας υπόψην πως είναι μία τροποποιημένη έκδοση της συνάρτησης sinc χρησιμοποιώντας τον τύπο:

$$h(t) = \frac{\sin(\pi t)}{\pi t} \times \frac{\cos(\pi \beta t)}{1-(2\beta t)^2} \quad (1)$$

Ο πρώτος όρος είναι η συνάρτηση sinc, που είναι το βασικό ιδανικό φίλτρο και ο δεύτερος όρος είναι ένας συντελεστής διόρθωσης που εισάγει την κυματομορφή cos, προσαρμόζοντας το εύρος ζώνης. Υπάρχουν όμως κάποια σημεία όπου ο παρονομαστής γίνεται 0 (διαιρέσεις με το μηδέν), άρα πρέπει να ορίσουμε ειδικές τιμές για το σημειο για t=0 και t=1/(2*roll_off), t=-1/(2*roll_off). Αντικαθιστούμε τη συνάρτηση σε t = 0 με την τιμή h(0)=1 - β+ 4β/π. Αυτό εξασφαλίζει ότι το φίλτρο είναι ομαλό και δεν έχει ανωμαλίες στο t = 0. Ακόμη, αντικαθιστούμε την τιμή στο σημείο ειδικής ασυνέχειας t =1/2β, t =-1/2β. Αυτό γίνεται γιατί εκεί ο παρονομαστής μηδενίζεται.

```
% Χρονικός άξονας για το φίλτρο
t = (-span/2:1/sps:span/2)';

% Υπολογισμός SRRC φίλτρου
h_srrc = (sin(pi*t)./(pi*t)) .* (cos(pi*roll_off*t)./(1 -
(2*roll_off*t).^2));
h_srrc(t == 0) = 1 - roll_off + (4 * roll_off / pi); % Ειδική τιμή στο 0
h_srrc(abs(t) == 1/(2*roll_off)) = (roll_off/sqrt(2)) * ((1 + 2/pi) *
sin(pi/(2*roll_off)) + (1 - 2/pi) * cos(pi/(2*roll_off))); % Ειδική τιμή
σε ±T/(2β)
```

Τέλος, το φίλτρο κανονικοποιείται ώστε η συνολική ενέργεια να είναι ίση με 1, καθώς έτσι εξασφαλίζεται ότι το φίλτρο δεν αλλοιώνει την ισχύ του σήματος. Ορίζουμε το ίδιο φίλτρο για τον πομπό (tx_filter) και τον δέκτη (rx_filter). Με την διπλή αυτή εφαρμογή επιτυγχάνεται η ελαχιστοποίηση της διασυμβολικής παρεμβολής (ISI).

```
h_srrc = h_srrc / sqrt(sum(h_srrc.^2));

tx_filter = h_srrc; % Φίλτρο πομπού
rx_filter = h_srrc; % Φίλτρο δέκτη
```

2. Υπερδειγματοληψία Ακολουθιών Και Καναλιού

Δημιουργούμε για αρχή 10.000 σύμβολα προς μετάδοση και καθορίζουμε ότι η διαμόρφωση είναι 4-PSK, δηλαδή κάθε σύμβολο μπορεί να πάρει μία από 4 πιθανές διακριτές φάσεις (45°, 135°, 225°, 315°) και κάθε σύμβολο μεταφέρει 2 bits πληροφορίας. Δημιουργείται ένας τυχαίος πίνακας από τιμές 0, 1, 2, 3, που είναι τα πιθανά σύμβολα στη 4-PSK, που έχει μέγεθος num_symbols × 1 (δηλαδή στήλη με 10.000 τυχαίες τιμές). Μετατρέπουμε τα σύμβολα σε σημεία του αστερισμού της 4-PSK μέσω του τύπου:

$$S = e^{j(\frac{\pi}{4} + \frac{2\pi * symbols}{M})} \quad (2)$$

Αυτό αντιστοιχεί στα παρακάτω σύμβολα:

Σύμβολο	Φάση	Θέση στο επίπεδο
0	$\frac{\pi}{4}$	$e^{j\frac{\pi}{4}}$
1	$\frac{3\pi}{4}$	$e^{j\frac{3\pi}{4}}$
2	$\frac{5\pi}{4}$	$e^{j\frac{5\pi}{4}}$
3	$\frac{7\pi}{4}$	$e^{j\frac{7\pi}{4}}$

```
num_symbols = 10000; % Αριθμός συμβόλων
M = 4; % Διαμόρφωση 4-PSK
symbols = randi([0 M-1], num_symbols, 1); % Τυχαία συμβολική ακολουθία

% Χαρτογράφηση συμβόλων σε αστερισμό 4-PSK
S = exp(1j * (pi/4 + (2 * pi * symbols) / M));
```

Στη συνέχεια, δημιουργούμε έναν μηδενικό πίνακα μήκους num_symbols * sps, έχοντας ορίσει παραπάνω το sps να είναι 4, δηλαδή κάθε σύμβολο θα αναπαρίσταται από 4 δείγματα. Στον πίνακα αυτό εισαγάγουμε τα σύμβολα κάθε 4 θέσεις, αφήνοντας 3 μηδενικά ενδιάμεσα. Επίσης, ορίζουμε το μη ιδανικό κανάλι γεμίζοντας τον πίνακα h που μας δίνεται προσομοιώνοντας ένα πραγματικό κανάλι. Για την υπερδειγματοληψία του καναλιού δημιουργούμε έναν μηδενικό πίνακα h_up με sps φορές μεγαλύτερο μήκος και εισαγάγουμε τα στοιχεία του h κάθε sps θέσεις, αφήνοντας μηδενικά ενδιάμεσα. Αυτό το κάνουμε επειδή όπως και στα σύμβολα, η υπερδειγματοληψία του καναλιού πρέπει να μας επιτρέπει να προσομοιώσουμε την επίδραση του καναλιού στη σωστή κλίμακα χρόνου. Ορίζουμε και το ιδανικό κανάλι, το οποίο δεν παραμορφώνει το σήμα, είναι απλά μία μοναδιαία κρουστική απόκριση. Οπότε, δημιουργούμε έναν μηδενικό πίνακα h_ideal_up μήκους sps και ορίζουμε το πρώτο στοιχείο ως 1, προκειμένου να διατηρεί το σήμα αναλλοίωτο στη συνέλιξη.

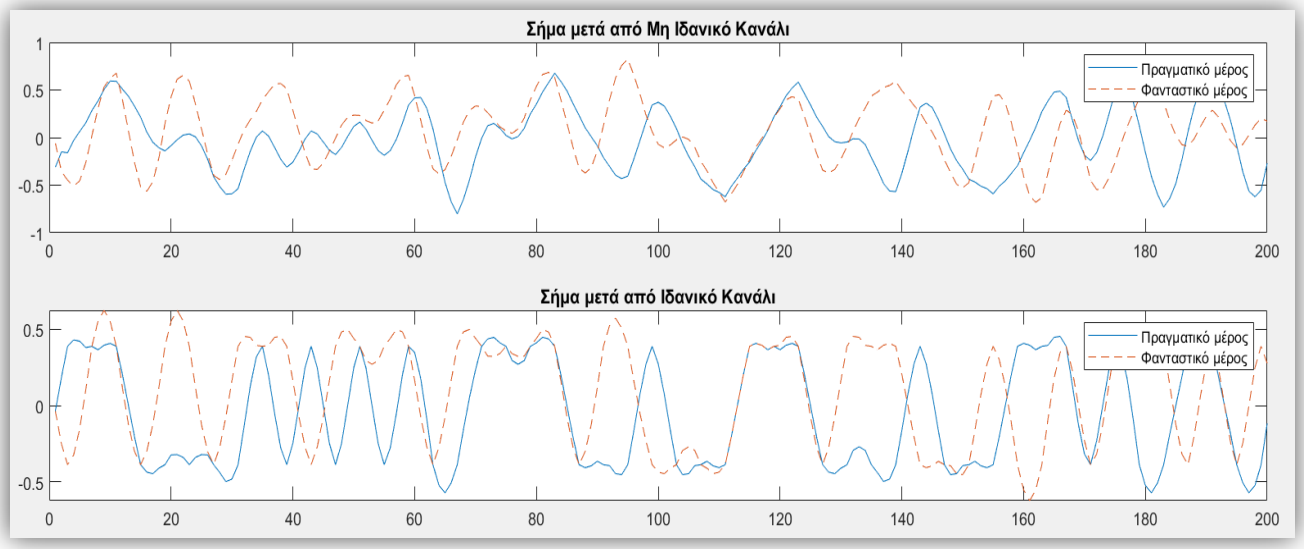
```
S_up = zeros(num_symbols * sps, 1);
S_up(1:sps:end) = S; % Τοποθέτηση συμβόλων κάθε 4 δείγματα (3 μηδενικά ενδιάμεσα)

% Κρουστική απόκριση μη ιδανικού καναλιού
h = [0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0 0.21 0.03 0.07];

% Υπερδειγματοληψία καναλιού (εισαγωγή 3 μηδενικών μεταξύ κάθε συντελεστή)
h_up = zeros(length(h) * sps, 1);
h_up(1:sps:end) = h;

% Ιδανικό κανάλι (χωρίς παραμόρφωση)
h_ideal = [1];
h_ideal_up = zeros(sps, 1);
h_ideal_up(1) = 1;
```

3. Κανάλια

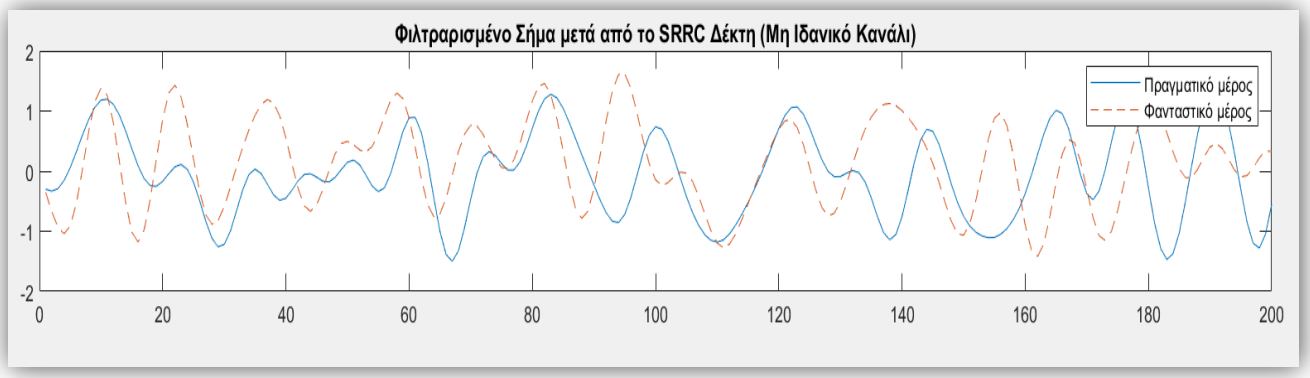
Εδώ ασχολούμαστε με την διέλευση του σήματος από τα κανάλια. Ξεκινάμε εκτελώντας τη συνέλιξη του υπερδειγματοληπτημένου σήματος εισόδου (S_up) με το φίλτρο SRRC (tx_filter), διατηρώντας το ίδιο μήκος στο αποτέλεσμα ('same'). Η συνέλιξη είναι η πράξη που φιλτράρει το σήμα εισόδου μέσω του φίλτρου SRRC, κάνοντας το πιο ομαλό και προσαρμόζοντάς το για μετάδοση και εκτελείται μέσω της conv. Μετά, για την διέλευση του σήματος από το μη ιδανικό κανάλι εκτελώντας τη συνέλιξη του μεταδιδόμενου σήματος (tx_signal) με την κρουστική απόκριση του καναλιού (h_up), δημιουργώντας το σήμα rx_signal, δηλαδή το σήμα που λαμβάνει ο δέκτης μετά την διέλευσή του από το μη ιδανικό κανάλι. Διατηρούμε πάλι το ίδιο μήκος στο αποτέλεσμα, γιατί αν δεν το χρησιμοποιούσαμε, η συνέλιξη θα είχε μεγαλύτερο μήκος, λόγω των άκρων του φίλτρου. Ουσιαστικά αυτό το κανάλι μπορεί να εξασθενεί ή να ενισχύει ορισμένα μέρη του σήματος. Ακολουθούμε ακριβώς την ίδια διαδικασία και για την διέλευση από το ιδανικό κανάλι, αλλάζοντας μόνο την μη ιδανική κρουστική απόκριση με την μοναδιαία(ιδανική). Το αποτέλεσμα rx_signal_ideal θα είναι το ίδιο το σήμα χωρίς παραμορφώσεις. Για να ολοκληρώσουμε την διαδικασία, θα υλοποιήσουμε και την διέλευση από το φίλτρο δέκτη ώστε να προετοιμαστεί το σήμα για αποδιαμόρφωση. Όμοια θα εφαρμόσουμε συνέλιξη μεταξύ του λαμβανόμενου σήματος (rx_signal ή rx_signal_ideal) και του φίλτρου SRRC (rx_filter) στον δέκτη. Το αποτέλεσμα rx_filtered θα είναι καθαρότερο και λιγότερο επηρεασμένο από ISI, ενώ το rx_filtered_ideal θα πρέπει να είναι σχεδόν ίδιο με το αρχικό σήμα πριν τη μετάδοση.

```
% Διέλευση από το φίλτρο πομπού (SRRC)
tx_signal = conv(S_up, tx_filter, 'same');

% Διέλευση από μη ιδανικό κανάλι
rx_signal = conv(tx_signal, h_up, 'same');

% Διέλευση από ιδανικό κανάλι
rx_signal_ideal = conv(tx_signal, h_ideal_up, 'same');

% Διέλευση από το φίλτρο δέκτη (SRRC)
rx_filtered = conv(rx_signal, rx_filter, 'same');
rx_filtered_ideal = conv(rx_signal_ideal, rx_filter, 'same');
```

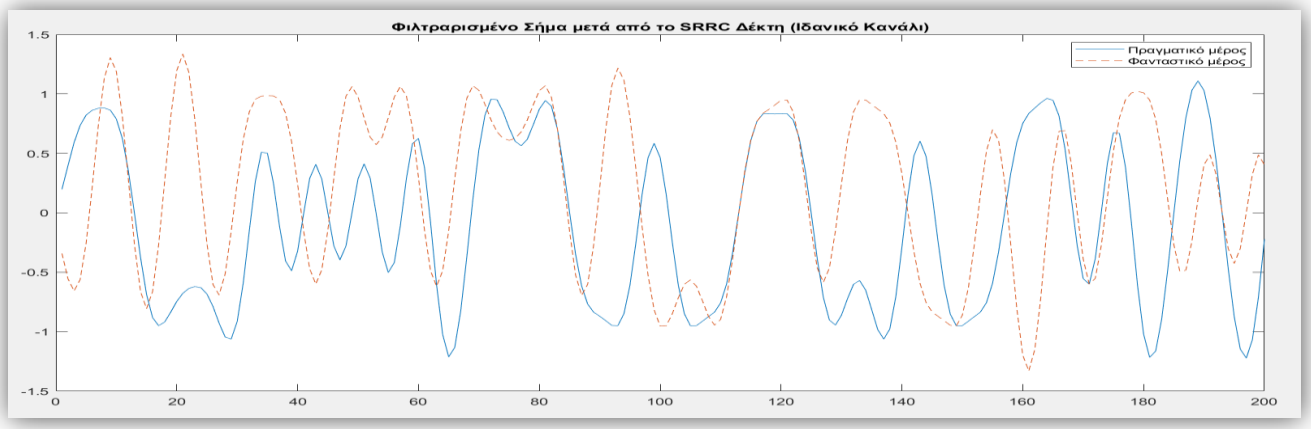
Πρέπει να εφαρμόσουμε SRRC φίλτρο και στον δέκτη, ώστε το συνολικό σύστημα (πομπός και δέκτης) να έχει Raised Cosine απόκριση. Το SRRC φίλτρο του πομπού από μόνο του δεν αρκεί, καθώς το σήμα δεν θα ήταν χρονικά ευθυγραμμισμένο και ο δέκτης δεν θα μπορούσε να αναγνωρίσει σωστά τα σύμβολα. Επίσης, το φάσμα του σήματος θα είχε υψηλές συχνότητες, κάτι που θα δημιουργεί παρεμβολές.



Στο μη ιδανικό κανάλι, το σήμα είναι παραμορφωμένο. Συγκεκριμένα, υπάρχουν ασυμμετρίες, και η κυματομορφή φαίνεται αλλοιωμένη λόγω του ISI και των multipath effects.
Στο ιδανικό κανάλι, το σήμα είναι πολύ καθαρότερο, αφού δεν υπήρξαν αλλοιώσεις κατά τη μετάδοση. Οι κυματομορφές είναι συμμετρικές και κοντά στο αρχικό σήμα.
Άρα επιβεβαιώνεται ότι το κανάλι επικοινωνίας προκαλεί παραμόρφωση στο σήμα λόγω ανακλάσεων και καθυστερήσεων, και ότι η χρήση φίλτρου SRRC στον δέκτη είναι απαραίτητη για να διορθώσουμε αυτές τις παραμορφώσεις.



Εδώ βλέπουμε το σήμα μετά από τη διέλευση από το SRRC φίλτρο στον δέκτη για την περίπτωση του μη ιδανικού καναλιού. Παρατηρούμε ότι η παραμόρφωση του σήματος έχει μειωθεί σημαντικά και ότι το σήμα είναι πιο καθαρό σε σχέση με πριν.Το πραγματικό και φανταστικό μέρος φαίνονται πιο ομαλά και με λιγότερες αιχμές από ότι στο μη φιλτραρισμένο σήμα. Βλέπουμε ότι το φίλτρο SRRC βοήθησε σημαντικά στην αποκατάσταση του σήματος, αλλά δεν έλυσε πλήρως το πρόβλημα του ISI που προστίθεται, καθώς υπάρχουν ακόμα κάποια υπολείμματα παρεμβολής μεταξύ συμβόλων.



Εδώ βλέπουμε το φιλτραρισμένο σήμα μετά από SRRC στον δέκτη, αλλά αυτή τη φορά για την περίπτωση του ιδανικού καναλιού. Το σήμα είναι πολύ καθαρό και χωρίς παραμορφώσεις. Το πραγματικό και το φανταστικό μέρος είναι ομαλά και ευθυγραμμισμένα, χωρίς διακυμάνσεις ή ISI.Η κυματομορφή διατηρεί τη σωστή χρονική ευθυγράμμιση, που σημαίνει ότι η αποδιαμόρφωση θα είναι πολύ εύκολη. Άρα, όταν το κανάλι είναι ιδανικό, το φίλτρο SRRC είναι αρκετό για τέλεια ανάκτηση του σήματος. Δεν υπάρχουν εμφανή προβλήματα ISI, καθώς το κανάλι δεν προσθέτει καθυστερήσεις ή παραμορφώσεις και το τελικό αποτέλεσμα μοιάζει σχεδόν με το αρχικό διαμορφωμένο σήμα.

4. Θόρυβος Συστήματος

Ξεκινάμε ορίζοντας την τιμή SNR (Signal-to-Noise Ratio) σε dB. Αυτό καθορίζει την αναλογία ισχύος του σήματος προς την ισχύ του θορύβου. Το mean(abs(y).^2) θα μας δώσει τη μέση ενέργεια του σήματος. Υπολογίζουμε την μέση ισχύ του σήματος μετά τη διέλευση από το μη ιδανικό κανάλι χρησιμοποιώντας τον τύπο :

$$Ps = \frac{1}{N} \sum_{n=1}^{N} |x[n]|^2 \quad (3)$$

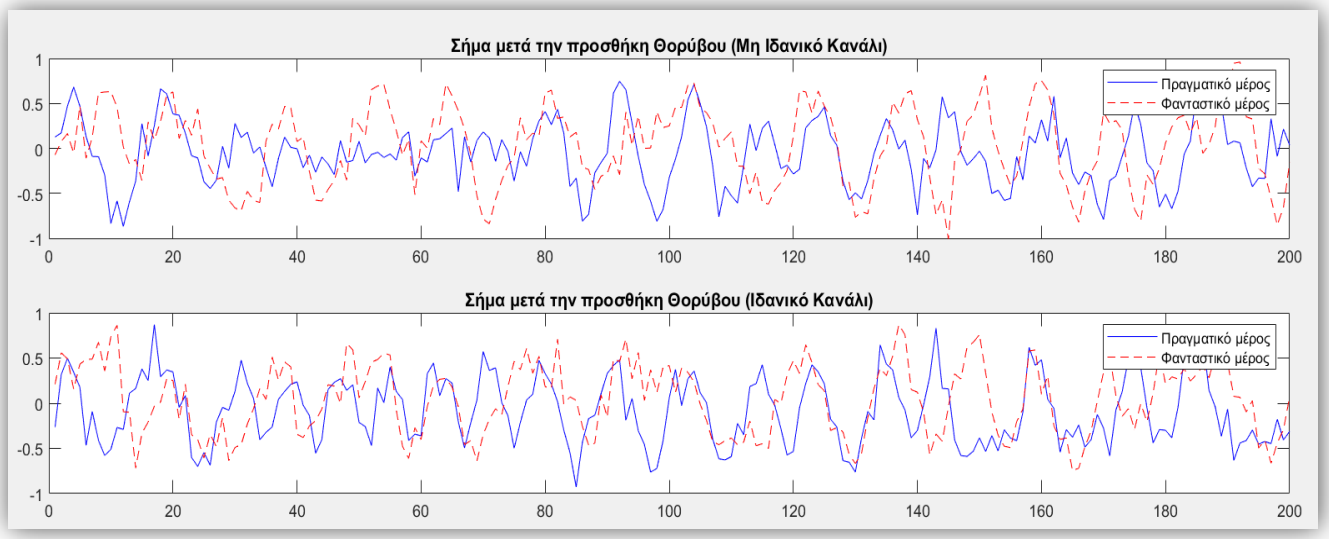
όπου x[n] είναι το σήμα, |x[n]|^2 είναι το τετράγωνο του μέτρου του κάθε δείγματος και N είναι το σύνολο των δειγμάτων. Το abs(x).^2 υπολογίζει την ισχύ του κάθε δείγματος |x[n]|^2 αφού προηγουμένως το abs(x) υπολογίσει το μέτρο του μιγαδικού σήματος |x[n]|. Το mean(...) θα υπολογίσει το μέσο όρο όλων των τιμών. Χρησιμοποιώντας τις σχέσεις:

$$w[n] = \sqrt{P_N} * (Xr[n] + j*Xi[n]) \quad (4)$$

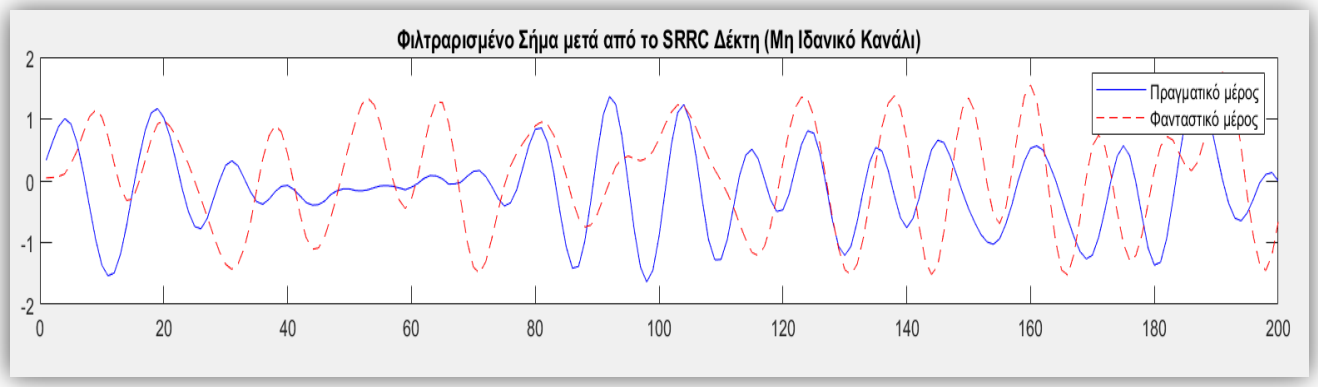
$$SNR(dB) = 10\log_{10}\left(\frac{Ps}{PN}\right) \leftrightarrow P_N = \frac{Ps}{10^{\left(\frac{SNR}{10}\right)}} \quad (5)$$

όπου P_N είναι η ισχύς του θορύβου, Xr[n] και Xi[n] είναι πραγματικοί αριθμοί από κανονική κατανομή με μέση τιμή 0 και διασπορά 1 (Gaussian Noise) και j είναι η φανταστική μονάδα.
Η randn(size(rx_signal)) παράγει την κανονική κατανομή (Gaussian noise) με μηδενική μέση τιμή και διασπορά 1. Αυτοί είναι οι πραγματικοί αριθμοί του θορύβου (Xr[n]).Το 1j*randn(...) προσθέτει το φανταστικό μέρος του θορύβου (Xi[n]), γιατί το σήμα είναι συμπλεγματικό. Το άθροισμά αυτών πολλαπλασιάζεται με sqrt(Pn) ώστε ο τελικός θόρυβος να έχει σωστή ισχύ P_N. Μετά, προσθέτουμε τον θόρυβο στο σήμα μετά τη διέλευση από το μη ιδανικό κανάλι με το rx_signal_noisy που προκύπτει να είναι το θορυβώδες σήμα που εισέρχεται στο φίλτρο του δέκτη. Παρομοίως υπολογίζουμε την ισχύ του σήματος μετά τη διέλευση από το ιδανικό κανάλι καθώς και την ισχύ του θορύβου για το ιδανικό κανάλι, χρησιμοποιώντας την ίδια σχέση όπως πριν. Ο λευκός Gaussian θόρυβος για το ιδανικό κανάλι θα έχει ξανά τα ίδια χαρακτηριστικά και τελικά θα πραγματοποιηθεί πάλι προσθήκη του θορύβου στο σήμα που πέρασε από το ιδανικό κανάλι, με το rx_signal_ideal_noisy να είναι το θορυβώδες σήμα που εισέρχεται στο φίλτρο του δέκτη.

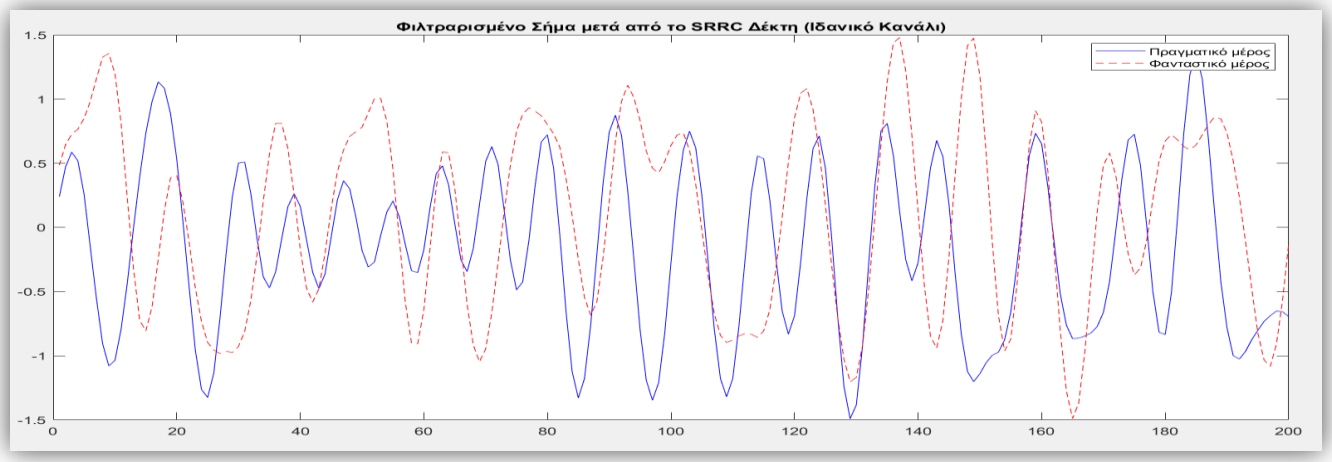
```
SNR_dB = 10; % Επιθυμητό SNR σε dB

% Υπολογισμός ισχύος του σήματος μετά τη διέλευση από το μη ιδανικό κανάλι
Ps = mean(abs(rx_signal).^2);
Pn = Ps / (10^(SNR_dB/10));
noise = sqrt(Pn) * (randn(size(rx_signal)) + 1j*randn(size(rx_signal)));
rx_signal_noisy = rx_signal + noise;

% Υπολογισμός ισχύος του σήματος μετά τη διέλευση από το ιδανικό κανάλι
Ps_ideal = mean(abs(rx_signal_ideal).^2);
Pn_ideal = Ps_ideal / (10^(SNR_dB/10));
noise_ideal = sqrt(Pn_ideal) * (randn(size(rx_signal_ideal)) +
1j*randn(size(rx_signal_ideal)));
rx_signal_ideal_noisy = rx_signal_ideal + noise_ideal;

%% Διέλευση από το Φίλτρο Δέκτη (SRRC)
rx_filtered = conv(rx_signal_noisy, rx_filter, 'same');
rx_filtered_ideal = conv(rx_signal_ideal_noisy, rx_filter, 'same');
```



Το μη ιδανικό κανάλι έχει ήδη προκαλέσει παρεμβολή μεταξύ των συμβόλων(ISI) και μετά την προσθήκη του Gaussian θορύβου, το σήμα γίνεται ακόμα πιο θορυβώδες και παρουσιάζει μεγαλύτερη διακύμανση. Οι κορυφές των παλμών αλλοιώνονται και τα σημεία του σήματος δεν είναι τόσο ευδιάκριτα.
Το ιδανικό κανάλι είχε ήδη καθαρότερη κυματομορφή πριν την προσθήκη του θορύβου. Ωστόσο, μετά την προσθήκη του Gaussian θορύβου, βλέπουμε πως έχει μικρότερη παραμόρφωση σε σχέση με το μη ιδανικό κανάλι. Ο θόρυβος είναι εμφανής, αλλά δεν έχει επιπλέον παραμορφώσεις λόγω ISI.
Κατανοούμε ότι ο θόρυβος επηρεάζει και τα δύο σήματα, αλλά το μη ιδανικό κανάλι είναι πιο ευαίσθητο λόγω ISI. Επίσης, το ιδανικό κανάλι διατηρεί καλύτερη δομή, καθώς δεν έχει ISI, μόνο θόρυβο. Αν το SNR ήταν χαμηλότερο, η παραμόρφωση θα ήταν ακόμα πιο έντονη.



Εδώ αντιλαμβανόμαστε πως το SRRC φίλτρο μείωσε σημαντικά την επίδραση του θορύβου. Οι κυματομορφές είναι πιο λείες και πιο ευδιάκριτες, πράγμα που σημαίνει ότι το ISI και ο θόρυβος έχουν περιοριστεί σε κάποιο βαθμό. Το πραγματικό και φανταστικό μέρος εξακολουθούν να έχουν παραμορφώσεις, αλλά πλέον μπορούμε να διακρίνουμε καλύτερα τη δομή του σήματος. Οι αιχμές του σήματος είναι πιο ομαλές, κάτι που σημαίνει ότι το φίλτρο SRRC λειτούργησε σωστά. Οπότε, και σε αυτή την περίπτωση το φίλτρο SRRC βελτίωσε τη δομή του σήματος μειώνοντας το ISI και τον θόρυβο. Αν πάλι υπήρχε χαμηλότερο SNR, το φίλτρο δεν θα μπορούσε να απομακρύνει πλήρως τον θόρυβο, διότι όταν το SNR μειώνεται, το σήμα γίνεται αδύναμο σε σχέση με τον θόρυβο. Αυτό σημαίνει ότι το φίλτρο SRRC δεν μπορεί να διαχωρίσει αποτελεσματικά το σήμα από τον θόρυβο.



Εδώ το φίλτρο SRRC καθάρισε ακόμα περισσότερο το σήμα, αφού εδώ δεν υπήρχε ISI, αλλά μόνο Gaussian θόρυβος. Οι κυματομορφές είναι πολύ πιο λείες και συμμετρικές σε σύγκριση με το μη ιδανικό κανάλι. Το φίλτρο εξάλειψε μεγάλο μέρος του θορύβου, αφήνοντας το σήμα σχεδόν στην αρχική του μορφή. Άρα, μπορούμε πλέον να ανακτήσουμε με μεγάλη ακρίβεια τα σύμβολα που στάλθηκαν. Το φίλτρο SRRC είχε εξαιρετική απόδοση στο ιδανικό κανάλι, γιατί το μόνο πρόβλημα ήταν μόνο ο θόρυβος. Το τελικό σήμα είναι πολύ καθαρό, οπότε η αποδιαμόρφωση θα είναι πολύ πιο αξιόπιστη.

5. Διάταξη Απόφασης

Κάνουμε υπερδειγματοληψία στα κατάλληλα διαστήματα. Αυτό σημαίνει ότι παίρνουμε ένα δείγμα ανά σύμβολο από τα φιλτραρισμένα σήματα rx_filtered και rx_filtered_ideal και χρησιμοποιείται το 1:sps:end για να πάρουμε μόνο τα δείγματα στις κατάλληλες χρονικές στιγμές. Αυτό το κάνουμε γιατίέχουμε περισσότερα δείγματα για τα σήματα από όσα χρειαζόμαστε. Πρέπει να πάρουμε ένα δείγμα ανά σύμβολο, δηλαδή στα σημεία που αντιστοιχούν στα αρχικά σύμβολα. Έπειτα, δημιουργούμε την QPSK συμβολική αναπαράσταση πάλι ως μονοδιάστατο πίνακα (1x4), ώστε να συγκρίνουμε το ληφθέν σήμα με αυτές και να πάρουμε απόφαση.

```
% Υποδειγματοληψία στα κατάλληλα διαστήματα
rx_sampled = rx_filtered(1:sps:end);
rx_sampled_ideal = rx_filtered_ideal(1:sps:end);

% Ορισμός αστερισμού συμβόλων για 4-PSK (QPSK)
symbol_constellation = exp(1j * (pi/4 + (2 * pi * (0:M-1)) / M));
```

Αντιγράφουμε τα δείγματα (rx_sampled) σε πολλές στήλες ώστε να έχουν συμβατές διαστάσεις με τον πίνακα symbol_constellation. Αυτό είναι απαραίτητο για να υπολογίσουμε σωστά την Ευκλείδεια απόσταση από όλα τα σύμβολα του αστερισμού. Το κάνουμε με την βοήθεια της συνάρτησης repmatη οποία αντιγράφει τον πίνακα rx_sampled μια φορά στις γραμμές και τέσσερις φορές στις στήλες. Αν δεν αλλάξουμε τον πίνακα rx_sampled και προσπαθήσουμε να αφαιρέσουμε τον πίνακα 1x4 symbol_constellation από τον πίνακα num_symbols x 1 (rx_sampled), θα προκύψει ασυμβατότητα διαστάσεων. Οι νέες διαστάσεις θα είναι num_symbols x M, όπου κάθε στήλη θα περιέχει το ίδιο rx_sampled. Μόνο με έναν πίνακα πολλαπλών στηλών μπορούμε να υπολογίσουμε ταυτόχρονα την απόσταση από όλα τα σύμβολα.

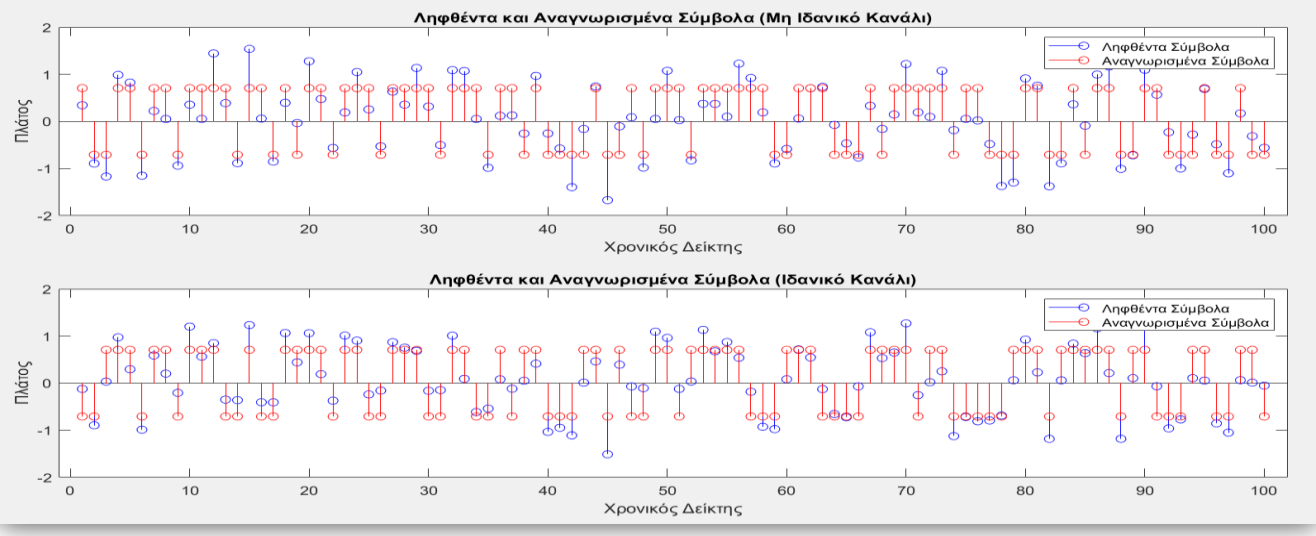
```
% Προσαρμογή διαστάσεων για σωστό υπολογισμό απόστασης
rx_sampled_matrix = repmat(rx_sampled, 1, M);
rx_sampled_matrix_ideal = repmat(rx_sampled_ideal, 1, M);
```

Συνεχίζουμε υπολογίζοντας την Ευκλείδεια απόσταση μεταξύ κάθε ληφθέντος δείγματος και όλων των δυνατών συμβόλων του αστερισμού, και επιλέγουμε το πιο κοντινό σύμβολο. Η Ευκλείδεια απόσταση μεταξύ ενός ληφθέντος συμβόλου r και ενός εκπεμπόμενου συμβόλου Sm είναι:

$$D(r,s_m) = |r - s_m| \quad (6)$$

όπου r είναι το ληφθέν δείγμα (από rx_sampled_matrix) και sm είναι το ιδανικό σύμβολο από το symbol_constellation. Αντί να το κάνουμε για ένα σύμβολο τη φορά, η MATLAB υπολογίζει τις αποστάσεις για όλα τα σύμβολα ταυτόχρονα. Πιο συγκεκριμένα, εφαρμόζει broadcasting και αφαιρεί το κάθε σύμβολο του αστερισμού από κάθε ληφθέν σύμβολο, δημιουργώντας έναν νέο πίνακα με όλες τις αποστάσεις. Χρησιμοποιούμε την συνάρτηση abs(), η οποία μετατρέπει τα μιγαδικά μεγέθη σε πραγματικές αποστάσεις και την min(..., [], 2),που βρίσκει τη στήλη με τη μικρότερη απόσταση, δηλαδή το πιο κοντινό σύμβολο. Το detected_symbols αποθηκεύει τους δείκτες των συμβόλων του αστερισμού που βρέθηκαν πιο κοντά στα ληφθέντα δείγματα. Εφαρμόζουμε αυτόν τον κανόνα ξεχωριστά στο μη ιδανικό και στο ιδανικό κανάλι. Τέλος, για μεγαλύτερη ακρίβεια υπολογίζουμε το ποσοστό των λαθών στην αναγνώριση των συμβόλων. Το detected_symbols ~= symbols δημιουργεί ένα λογικό διάνυσμα 1 αν το σύμβολο είναι λάθος, και 0 αν είναι σωστό και το sum() μετράει το συνολικό πλήθος των λαθών.Διαιρούμε με num_symbols και παίρνουμε το ποσοστό των σφαλμάτων.

```
[~, detected_symbols] = min(abs(rx_sampled_matrix - symbol_constellation), [], 2);
[~, detected_symbols_ideal] = min(abs(rx_sampled_matrix_ideal -
symbol_constellation), [], 2);

% Υπολογισμός του Bit Error Rate (BER)
BER = sum(detected_symbols ~= symbols) / num_symbols;
BER_ideal = sum(detected_symbols_ideal ~= symbols) / num_symbols;
```



Bit Error Rate (Μη Ιδανικό Κανάλι): 0.90500
Bit Error Rate (Ιδανικό Κανάλι): 0.85210

Στο Μη Ιδανικό Κανάλι υπάρχουν αρκετά λάθη στην αναγνώριση των συμβόλων, αφού τα αναγνωρισμένα σύμβολα (κόκκινα) αποκλίνουν από τα ληφθέντα (μπλε). Οι κόκκινες γραμμές είναι μεγάλες, κάτι που σημαίνει μεγάλη διαφορά μεταξύ του ληφθέντος και του εκτιμώμενου συμβόλου. Αυτό φείλεται σε διαλείψεις (fading) και παρεμβολές από το μη ιδανικό κανάλι.
Στο Ιδανικό Κανάλι τα ληφθέντα και αναγνωρισμένα σύμβολα είναι πιο κοντά μεταξύ τους, δείχνοντας ότι η απόφαση ML λειτουργεί καλύτερα. Υπάρχουν λιγότερα λάθη σε σχέση με το μη ιδανικό κανάλι, καθώς το φίλτρο και το κανάλι δεν αλλοιώνουν τα σήματα. Ο θόρυβος εξακολουθεί να υπάρχει, αλλά το σύστημα είναι πιο αξιόπιστο.
Άρα, το μη ιδανικό κανάλι προκαλεί περισσότερα λάθη, αυξάνοντας το BER (Bit Error Rate), ενώ το ιδανικό κανάλι έχει πολύ καλύτερη ανίχνευση, αφού οι αποκλίσεις είναι μικρότερες.

ΜΕΡΟΣ Β: Διαμόρφωση M-PSK Βασικής Ζώνης

1. Αρχικά, δημιουργήστε μια αρκούντως μεγάλη ψευδοτυχαία δυαδική ακολουθία.

Για την ψευδοτυχαία δυαδική ακολουθία δημιουργούμε ένα διάνυσμα μήκους num_bits γεμάτο με τυχαία 0 και 1, όπου κάθε στοιχείο αντιστοιχεί σε ένα bit δεδομένων. Ομαδοποιούμε τα bits σε ομάδες των bits_per_symbol bits και μέσω της reshape αντιστρέφουμε τη δομή ώστε κάθε σειρά να είναι ένα σύμβολο, δηλαδή μετατρέπουμε το διάνυσμα σε έναν πίνακα όπου κάθε γραμμή αντιστοιχεί σε ένα σύμβολο, αλλάζοντας παράλληλα και τις γραμμές σε στήλες για σωστή μορφή στον πίνακα.
Μετά, μετατρέπουμε κάθε δυαδική ομάδα bits σε έναν αντίστοιχο δεκαδικό αριθμό με την χρήση της εντολής 2.^(bits_per_symbol-1:-1:0). Για παράδειγμα, για την περίπτωση του M=4, τα bits_per_symbol είναι 2, οπότε δημιουργείται το διάνυσμα των δυαδικών συντελεστών: 2.^(1:-1:0) = [2^1 2^0] = [2 1] και το bit_groups .* [2 1] εφαρμόζει στοιχειώδη πολλαπλασιασμό:

[1 0] .* [2 1] = [2 0] → sum([2 0]) = 2
[1 1] .* [2 1] = [2 1] → sum([2 1]) = 3
[0 0] .* [2 1] = [0 0] → sum([0 0]) = 0
[1 0] .* [2 1] = [2 0] → sum([2 0]) = 2
Τελικά, το αποτέλεσμα θα είναι symbol_indices = [2; 3; 0; 2].

```
% Δημιουργία ψευδοτυχαίας δυαδικής ακολουθίας
bit_sequence = randi([0 1], num_bits, 1);

% Μετατροπή των bits σε σύμβολα (ομαδοποίηση ανά bits_per_symbol bits)
bit_groups = reshape(bit_sequence, bits_per_symbol, []).';
symbol_indices = sum(bit_groups .* 2.^(bits_per_symbol-1:-1:0), 2); % Δυαδικό σε
δεκαδικό
```

2. Υλοποιήστε τις διαμορφώσεις 4-PSK και 8-PSK και δώστε το σχήμα του αστερισμού που υλοποιήσατε σε καθεμία από τις δύο περιπτώσεις.

Θα ορίσουμε τις βασικές παραμέτρους του συστήματος, δηλαδή τον αριθμό bits που θα διαμορφώσουμε, την ενέργεια συμβόλου (το πλάτος των σημείων στον αστερισμό θα είναι σταθερό) και την τιμή M για τα δύο σχήματα διαμόρφωσης 4-PSK, 8-PSK. Στην 4-PSK θα διαμορφώνονται 4 σύμβολα (κάθε σύμβολο έχει 2 bits) και στην 8-PSK θα διαμορφώνονται 8 σύμβολα (κάθε σύμβολο έχει 3 bits).

```
%% ====== Ορισμός Παραμέτρων ======
num_bits = 10000; % Συνολικός αριθμός δυαδικών bit
Es = 1; % Ενέργεια συμβόλου
M_values = [4, 8]; % Τάξεις διαμόρφωσης (4-PSK & 8-PSK)
```

Συνεχίζουμε υλοποιώντας τον βρόχο for που εκτελείται δύο φορές, η πρώτη επανάληψη (i=1) διαμορφώνει την 4-PSK και η δεύτερη επανάληψη (i=2) διαμορφώνει την 8-PSK. Η M = M_values(i); επιλέγει το αντίστοιχο M (πρώτα 4, μετά 8) και η bits_per_symbol = log2(M); υπολογίζει πόσα bits αντιστοιχούν σε κάθε σύμβολο. Επίσης, πρέπει να στρογγυλοποιήσουμε τον συνολικό αριθμό των bits που ορίσαμε ώστε να είναι πολλαπλάσιο του bits_per_symbol. Αυτό επιτυγχάνεται πολλαπλασιάζοντας τα bits_per_symbol με το κάτω φράγμα του αριθμού των πλήρων συμβόλων που μπορούμε να δημιουργήσουμε, διασφαλίζοντας ότι το adjusted_num_bits είναι έγκυρο για τη διαμόρφωση. Διαμορφώνουμε την ψευδοτυχαία ακολουθία και κάνουμε τις κατάλληλες μετατροπές των bits με τον τρόπο που δείξαμε στο ζητούμενο 1 και τελικά υπολογίζουμε τα διαμορφωμένα σύμβολα M-PSK και τα τοποθετούμε στον μοναδιαίο κύκλο στο μιγαδικό επίπεδο.

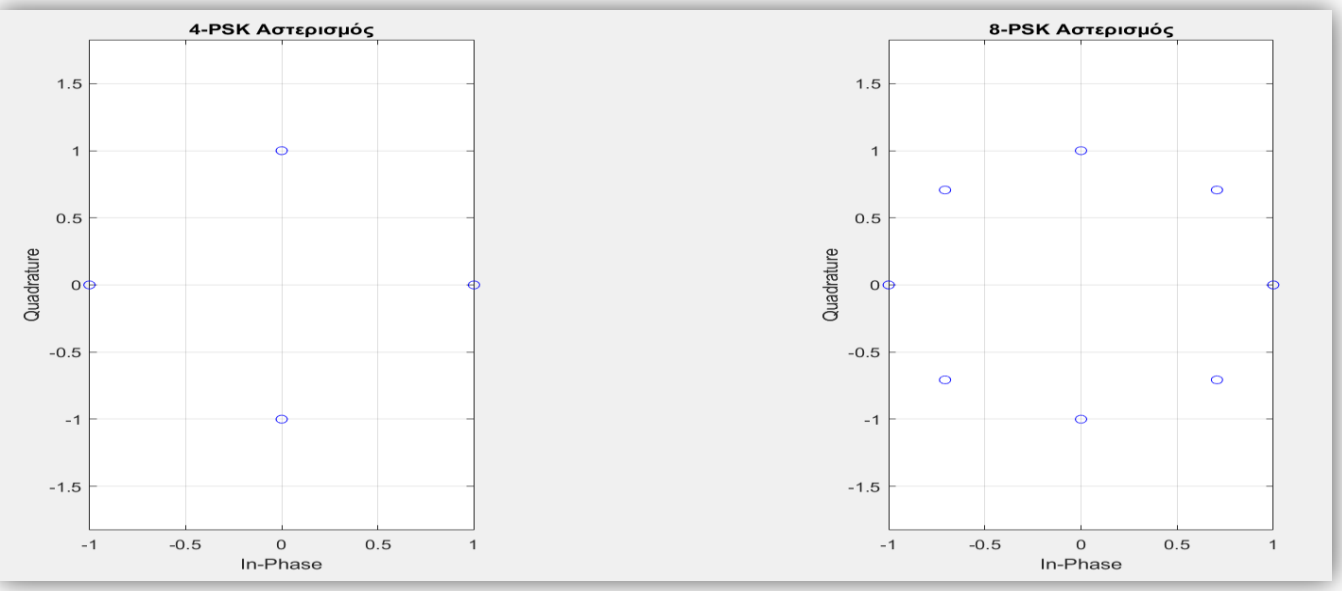
Αναλυτικότερα, μετατρέπουμε τους ακέραιους δείκτες symbol_indices (π.χ. 0,1,2,3 για 4-PSK) σε φάσεις μεταξύ 0◦ και 360 ◦ για M-PSK διαμόρφωση και στη συνέχεια δημιουργούμε τα σύμβολα ως σημεία στον μοναδιαίο κύκλο στο μιγαδικό επίπεδο. Μετατρέπουμε τους δείκτες σε γωνίες (φάσεις) στον κύκλο με την 2 * pi * symbol_indices / M,υπολογίζουμε το αντίστοιχο μιγαδικό σημείο e^jθ στον μοναδιαίο κύκλο με την exp(1j *) και ρυθμίζουμε την ενέργεια του συμβόλου, ώστε όλα να έχουν σταθερό πλάτος με την sqrt(Es).

```
figure;
for i = 1:length(M_values)
    M = M_values(i); % Επιλογή M-PSK
    bits_per_symbol = log2(M); % Bits ανά σύμβολο

    % Προσαρμογή num_bits ώστε να είναι πολλαπλάσιο του bits_per_symbol
    adjusted_num_bits = bits_per_symbol * floor(num_bits / bits_per_symbol);

    % Δημιουργία ψευδοτυχαίας δυαδικής ακολουθίας
    bit_sequence = randi([0 1], adjusted_num_bits, 1);

    % Μετατροπή των bits σε σύμβολα
    bit_groups = reshape(bit_sequence, bits_per_symbol, []).'; % Μετατροπή σε
ομάδες bits_per_symbol
    symbol_indices = sum(bit_groups .* 2.^(bits_per_symbol-1:-1:0), 2); % Δυαδικό
σε δεκαδικό

    % Διαμόρφωση M-PSK
    S = sqrt(Es) * exp(1j * (2 * pi * symbol_indices / M));
```

Σύμβολο m	Γωνία θ= 2πm/M (σε μοίρες για 4-PSK)	Γωνία θ= 2πm/M (σε μοίρες για 8-PSK)
0	0◦	0◦
1	90◦	45◦
2	180◦	90◦
3	270◦	135◦
4		180◦
5		225◦
6		270◦
7		315◦

Για 4-PSK, τα σημεία στον αστερισμό είναι S = [1, j, -1, -j] ή αλλιώς S = [e^j*0, e^j*π/2, e^j*π, e^j*3π/2] που αντιστοιχούν στις φάσεις 0°, 90°, 180°, 270°.

Για 8-PSK, τα σημεία στον αστερισμό είναι S = [1, 0.707+0.707j, j, -0.707+0.707j, -1, -0.707-0.707j, -j, 0.707-0.707j] ή αλλιώς S = [e^j*0, e^j*π/4, e^j*π/2, e^j*3π/4, e^j*π, e^j*5π/4, e^j*3π/2, e^j*7π/4] που αντιστοιχούν στις φάσεις 0°, 45◦, 90◦, 135◦, 180◦, 225◦, 270◦, 315◦.



Ο άξονας x (In-Phase) αντιστοιχεί στο πραγματικό μέρος του σήματος.
Ο άξονας y (Quadrature) αντιστοιχεί στο φανταστικό μέρος του σήματος.

Στη θέση του αστερισμού για την διαμόρφωση 4-PSK παρατηρούμε 4 σημεία, τοποθετημένα στις γωνίες του μοναδιαίου κύκλου. Οι θέσεις των συμβόλων αντιστοιχούν στις γωνίες 0◦, 90◦,180◦, 270◦ στο μιγαδικό επίπεδο. Αυτή η μορφή είναι συμμετρική και σταθερή, αφού το 4-PSK είναι μια πολύ αποδοτική διαμόρφωση με μεγάλη αντοχή στο θόρυβο. Τα σημεία βρίσκονται πάνω στον μοναδιαίο κύκλο με σταθερό πλάτος και η απόσταση μεταξύ των συμβόλων είναι μέγιστη, κάτι που βοηθά στη σωστή αποκωδικοποίηση.

Στη θέση του αστερισμού για την διαμόρφωση 8-PSK παρατηρούμε 8 σημεία, τοποθετημένα ομοιόμορφα γύρω από τον μοναδιαίο κύκλο. Οι θέσεις των συμβόλων αντιστοιχούν στις γωνίες 0◦, 45◦, 90◦, ..., 315◦ στο μιγαδικό επίπεδο. Επειδή έχει έχει διπλάσια σύμβολα από το 4-PSK, αυξάνει την αποδοτικότητα του φάσματος και χρησιμοποιείται για καλύτερη εκμετάλλευση του εύρους ζώνης, αλλά με μεγαλύτερη πιθανότητα σφάλματος σε χαμηλό SNR. Από την άλλη, η απόσταση μεταξύ των σημείων είναι μικρότερη σε σχέση με το 4-PSK, κάτι που σημαίνει ότι το 8-PSK είναι πιο ευαίσθητο στο θόρυβο.

3. Για καθένα από τα δύο συστήματα και για κάθε ένα από τα δύο κανάλια μετρήστε την πιθανότητα σφάλματος συμβόλου και σχεδιάστε την καμπύλη SER για τιμές του SNR=[0:2:30] dB.

Προσθέτουμε στις βασικές παραμέτρους που ορίζουμε το εύρος των τιμών του SNR σε dB από 0 έως 30 με βήμα 2, καθώς και δύο διαφορετικές διαμορφώσεις, 4-PSK και 8-PSK που χρησιμοποιούμε. Χρησιμοποιούμε τις τιμές της κρουστικής απόκρισης για το μη ιδανικό κανάλι, που ορίσαμε κατά την υλοποίηση του βήματος 2. Υπολογίζουμε το φίλτρο SRRC όπως ακριβώς έχουμε περιγράψει και στη συνέχεια μπαίνουμε στον βρόχο για 4-PSK και 8-PSK υλοποιώντας τον ακριβώς όπως στο προηγούμενο ζητούμενο προσθέτοντας 2 ακόμη αρχικοποιημένους με μηδενικά πίνακες μεγέθους SNR_dB_range, ώστε να αποθηκεύσουμε το SER για το ιδανικό και το μη ιδανικό κανάλι αντίστοιχα.

```
M_values = [4, 8]; % Διαμόρφωση 4-PSK & 8-PSK
SNR_dB_range = 0:2:30; % Τιμές SNR σε dB
num_symbols = 10000; % Αριθμός συμβόλων
sps = 4; % Δείγματα ανά σύμβολο
roll_off = 0.3; % Roll-off factor
span = 6; % Αριθμός περιόδων SRRC
h = [0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0 0.21 0.03 0.07]; % Μη Ιδανικό Κανάλι

% Δημιουργία φίλτρου SRRC
t = (-span/2:1/sps:span/2)';
h_srrc = (sin(pi*t)./(pi*t)) .* (cos(pi*roll_off*t)./(1 - (2*roll_off*t).^2));
h_srrc(t == 0) = 1 - roll_off + (4 * roll_off / pi);
h_srrc(abs(t) == 1/(2*roll_off)) = (roll_off/sqrt(2)) * ((1 + 2/pi) *
sin(pi/(2*roll_off)) + (1 - 2/pi) * cos(pi/(2*roll_off)));
h_srrc = h_srrc / sqrt(sum(h_srrc.^2));

figure;
for m_idx = 1:length(M_values)
    M = M_values(m_idx); % Επιλογή M-PSK
    bits_per_symbol = log2(M); % Bits ανά σύμβολο
    SER_nonideal = zeros(size(SNR_dB_range)); % Αποθήκευση SER για μη ιδανικό κανάλι
    SER_ideal = zeros(size(SNR_dB_range));
```

Μέσα στον βρόχο εκτελείται και ο δεύτερος βρόχος for για κάθε τιμή του SNR που έχουμε ορίσει στο SNR_dB_range. Δηλαδή, το SNR παίρνει τις τιμές [0, 2, 4, ..., 30] dB. Ο βρόχος τρέχει όσες φορές είναι οι τιμές του SNR, δηλαδή 16 επαναλήψεις, αφού το 0:2:30 έχει 16 στοιχεία. Στο τέλος κάθε επανάληψης αναθέτουμε την τρέχουσα τιμή SNR σε dB στη μεταβλητή SNR_dB. Σε αυτό το εύρος επαναλήψεων θα δημιουργήσουμε την τυχαία ακολουθία bits, θα μετατρέψουμε τα bits σε σύμβολα και θα α δημιουργήσουμε τον αστερισμό των συμβόλων στο μιγαδικό επίπεδο, μετατρέποντας τους ακέραιους δείκτες symbol_indices σε μιγαδικά σύμβολα χρησιμοποιώντας τον πίνακα S_ref ξεκινώντας από το 1 και όχι από το 0, λόγω λειτουργίας των πινάκων στην MATLAB. Επίσης, κανονικοποιούμε τον πίνακα με τα μιγαδικά σύμβολα ώστε Es = 1.

```
for snr_idx = 1:length(SNR_dB_range)
        SNR_dB = SNR_dB_range(snr_idx);

        % Δημιουργία ψευδοτυχαίας ακολουθίας
        bit_sequence = randi([0 1], num_symbols * bits_per_symbol, 1);
        bit_groups = reshape(bit_sequence, bits_per_symbol, []).';
        symbol_indices = sum(bit_groups .* 2.^(bits_per_symbol-1:-1:0), 2);

        % Διαμόρφωση M-PSK
        S_ref = exp(1j * (2 * pi * (0:M-1) / M));
        S = S_ref(symbol_indices + 1);
        S = S / sqrt(mean(abs(S).^2)); % Κανονικοποίηση ώστε Es = 1
```

Για την υπερδειγματοληψία συμβόλων δημιουργούμε κατά τα γνωστά έναν μεγαλύτερο πίνακα S_up με μηδενικά για να υπερδειγματοληπτήσουμε τα σύμβολα, τοποθετούμε τα σύμβολα στα πολλαπλάσια του sps και προσθέτουμε sps-1 μηδενικά ανάμεσα. Έτσι, κάθε σύμβολο επαναλαμβάνεται ανά sps δείγματα. Στη συνέχεια, το υπερδειγματοληπτημένο σήμα περνάει από φίλτρο πομπού SRRC για το ιδανικό κανάλι, το οποίο φίλτρο μειώνει το εύρος ζώνης του σήματος. Πάντα κρατάμε το μέγεθος του σήματος σταθερό (δεν επεκτείνεται λόγω της συνέλιξης). Έπειτα, για την διέλευση από το μη ιδανικό κανάλι υπερδειγματοληπτούμε την κρουστική απόκριση του καναλιού, ώστε να ταιριάζει με το υπερδειγματοληπτημένο σήμα S_up. Αν δεν εφαρμόσουμε υπερδειγματοληψία στο κανάλι, δεν θα μπορούμε να κάνουμε συνέλιξη σωστά. Οπότε, δημιουργούμε έναν νέο πίνακα μήκους length(h) * sps αρχικά με μηδενικά, που θα περιέχει την υπερδειγματοληπτούμενη μορφή του h. Μετά, κάθε στοιχείο του h τοποθετείται ανά sps = 4 δείγματα, ενώ τα υπόλοιπα ενδιάμεσα δείγματα είναι μηδενικά. Τελικά, εκτελούμε πάλι πράξη συνέλιξης μεταξύ του σήματος που λαμβάνεται στον δέκτη αφού περάσει από το φίλτρο πομπού (SRRC) χωρίς αλλοιώσεις από το κανάλι, και του h_up μετά την υπερδειγματοληψία.

```
% Υπερδειγματοληψία συμβόλων
        S_up = zeros(num_symbols * sps, 1);
        S_up(1:sps:end) = S;

        % Διέλευση από το ιδανικό κανάλι
        y_ideal = conv(S_up, h_srrc, 'same');

        % Διέλευση από το μη ιδανικό κανάλι
        h_up = zeros(length(h) * sps, 1);
        h_up(1:sps:end) = h;
        y_nonideal = conv(y_ideal, h_up, 'same');
```

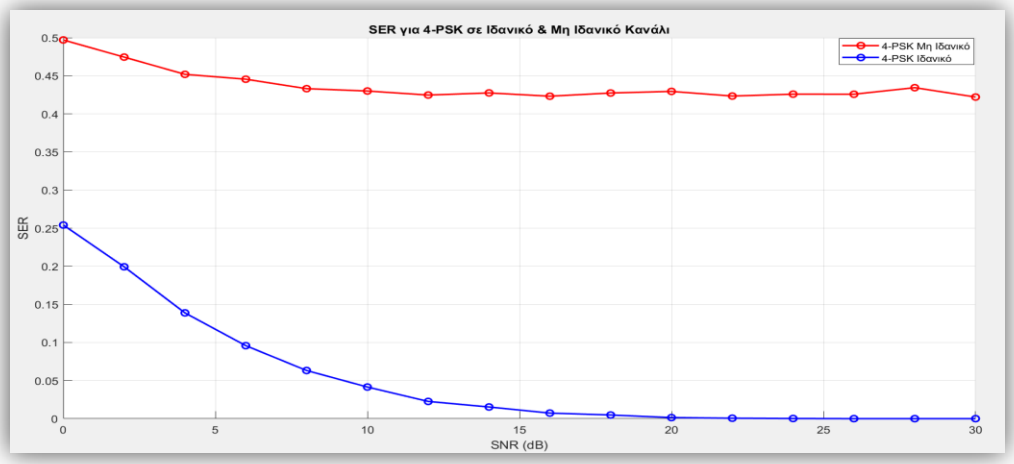
Πριν την προσθήκη θορύβου στο σήμα θα πρέπει πρώτα να υπολογίσουμε την ισχύ του ιδανικού σήματος, δηλαδή του σήματος χωρίς παραμορφώσεις και μετά την ισχύ του σήματος που έχει περάσει από το κανάλι με παραμορφώσεις. Ανατρέχουμε στον τύπο *(3).
Σειρά έχει ο υπολογισμός της ισχύος θορύβου με βάση το SNR με χρήση του τύπου *(4),(5). Προσθέτουμε Gaussian θόρυβο στο σήμα και προκύπτει το σήμα σε ιδανικό κανάλι με θόρυβο και το σήμα μετά από παραμορφώσεις και θόρυβο. Το θορυβώδες αυτό σήμα θα περάσει στον δέκτη. Για να αποκαταστήσουμε τη σωστή μορφή του σήματος, το φιλτράρουμε με το ίδιο SRRC φίλτρο που χρησιμοποιήθηκε στον πομπό. Η συνέλιξη με το φίλτρο SRRC μειώνει τον ISI και εξομαλύνει το σήμα. Ακόμη, πρέπει να επιλέξουμε τα σωστά δείγματα, δηλαδή ένα δείγμα ανά σύμβολο. Το κατάλληλο σημείο υποδειγματοληψίας είναι κάθε sps δείγματα. Απορρίπτουμε τα ενδιάμεσα δείγματα που προστέθηκαν κατά την υπερδειγματοληψία. Άν δεν επιλέξουμε σωστά τα δείγματα, ο δέκτης θα κάνει περισσότερα λάθη. Δημιουργούμε όλα τα πιθανά σύμβολα που μπορεί να λάβει ο δέκτης για 4-PSK και για 8-PSK. Αυτά τα σύμβολα θα είναι τα αναγνωρισμένα σύμβολα που συγκρίνουμε με το λαμβανόμενο σήμα.

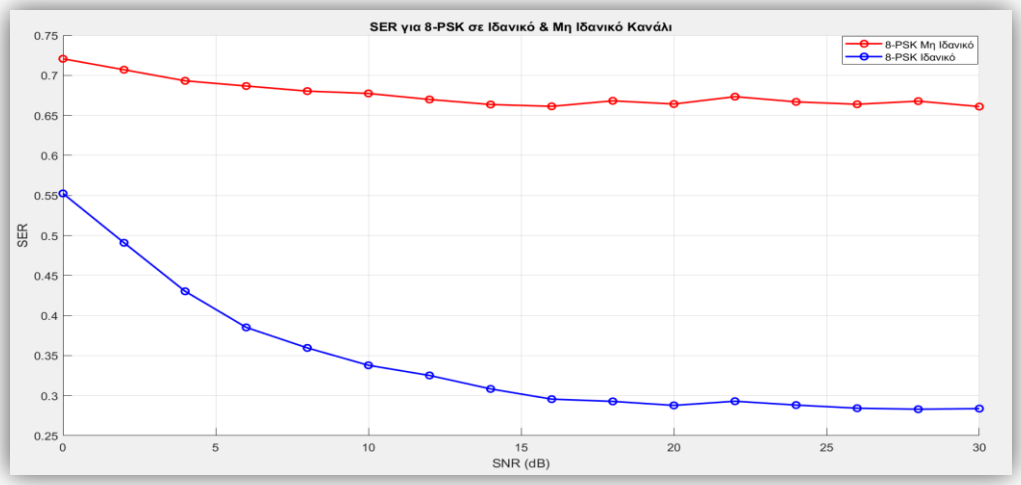
```
% Υπολογισμός θορύβου
        P_N_nonideal = mean(abs(y_nonideal).^2) / (10^(SNR_dB/10));
        noise_nonideal = sqrt(P_N_nonideal) * (randn(size(y_nonideal)) + 1j *
randn(size(y_nonideal)));
        y_nonideal_noisy = y_nonideal + noise_nonideal;

      P_N_ideal = mean(abs(y_ideal).^2) / (10^(SNR_dB/10));
      noise_ideal = sqrt(P_N_ideal) * (randn(size(y_ideal)) + 1j *
randn(size(y_ideal)));
        y_ideal_noisy = y_ideal + noise_ideal;

 % Φιλτράρισμα με φίλτρο δέκτη SRRC
      y_nonideal_filtered = conv(y_nonideal_noisy, h_srrc, 'same';
      y_ideal_filtered = conv(y_ideal_noisy, h_srrc, 'same');

        % Υποδειγματοληψία
        y_nonideal_sampled = y_nonideal_filtered(sps/2:sps:end);
        y_ideal_sampled = y_ideal_filtered(sps/2:sps:end);
```

Τέλος, χρησιμοποιώντας τον τύπο *(6) υπολογίζουμε την ευκλείδεια απόσταση του κάθε δείγματος από όλα τα δυνατά σύμβολα και βρίσκουμε ποιο σύμβολο είναι πιο κοντά στο λαμβανόμενο σήμα. Το ελάχιστο λάθος σημαίνει ότι έχουμε βρει το σωστό σύμβολο. Ουσιαστικά υπολογίζουμε τις αποστάσεις όλων των δειγμάτων που λήφθηκαν στον δέκτη μετά την υποδειγματοληψία, από όλα τα πιθανά σύμβολα του M-PSK αστερισμού. Η abs(y_ideal_sampled - S_ref) υπολογίζει το απόλυτο μέτρο της απόστασης και η min(..., [], 2) επιστρέφει το ελάχιστο κατά γραμμή (axis=2), δηλαδή για κάθε δείγμα ξεχωριστά. Η ίδια διαδικασία εκτελείται για το μη ιδανικό κανάλι. Αφού βρεθεί το κοντινότερο σύμβολο, επιστρέφεται ο δείκτης του στη μεταβλητή idx_ideal ή idx_nonideal. Ο idx_ideal είναι ο δείκτης των σωστών συμβόλων για το ιδανικό κανάλι και ο idx_nonideal είναι ο δείκτης των σωστών συμβόλων για το μη ιδανικό κανάλι. Και οι δύο είναι ακέραιοι αριθμοί που δείχνουν ποιο σύμβολο του αστερισμού M-PSK έχει την ελάχιστη απόσταση από το λαμβανόμενο δείγμα. Τα αποτελέσματα είναι τα αναγνωρισμένα σύμβολα, τα οποία αποθηκεύονται στις μεταβλητές decided_symbols_ideal και decided_symbols_nonideal.
Για να υπολογίσουμε το SER (Symbol Error Rate), το οποίο είναι ο ρυθμός σφαλμάτων σε επίπεδο συμβόλων, πρέπει να υπολογίσουμε το ποσοστό των λανθασμένων συμβόλων στο συνολικό αριθμό των μεταδοθέντων

συμβόλων. Η decided_symbols_ideal ~= S είναι η λογική πράξη που συγκρίνει τα ληφθέντα σύμβολα με τα αρχικά και επιστρέφει 1 όπου υπάρχει σφάλμα, δηλαδή όταν ένα σύμβολο δεν ταιριάζει, και 0 όπου η αναγνώριση είναι σωστή, δηλαδή το σύμβολο ταιριάζει. Προκύπτει ένας λογικός πίνακας με 1 για σφάλματα και 0 για σωστή αναγνώριση. Μετράμε πόσα 1 υπάρχουν με τη συνάρτηση sum(...), δηλαδή τον συνολικό αριθμό των σφαλμάτων. Για να βρούμε το τελικό ποσοστό λανθασμένης αναγνώρισης διαιρούμε τον αριθμό των λανθασμένων συμβόλων με το συνολικό αριθμό των συμβόλων (num_symbols). Αν το SER γίνει μηδέν σε πολύ υψηλές τιμές του SNR, βάζουμε το 1e-10 ως ελάχιστο όριο για να αποφύγουμε τα προβλήματα αριθμητικής υπολογιστικής ακρίβειας κατά τη σχεδίαση των γραφημάτων. Έτσι, προκύπτει το SER για το ιδανικό κανάλι σε μια συγκεκριμένη τιμή του SNR και αποθηκεύεται στη μεταβλητή SER_ideal. Η ίδια διαδικασία εκτελείται και για το μη ιδανικό κανάλι και το SER για το μη ιδανικό κανάλι σε μια συγκεκριμένη τιμήτου SNR αποθηκεύεται στη μεταβλητή SER_nonideal.

```
% ML Detection - Απόφαση
        [~, idx_nonideal] = min(abs(y_nonideal_sampled - S_ref), [], 2);
        decided_symbols_nonideal = S_ref(idx_nonideal);

        [~, idx_ideal] = min(abs(y_ideal_sampled - S_ref), [], 2);
        decided_symbols_ideal = S_ref(idx_ideal);

% Υπολογισμός SER
        SER_nonideal(snr_idx) = max(sum(decided_symbols_nonideal ~= S) /
num_symbols, 1e-10);
        SER_ideal(snr_idx) = max(sum(decided_symbols_ideal ~= S) / num_symbols,
1e-10);
```





4-PSK σε Ιδανικό & Μη Ιδανικό Κανάλι :

Ιδανικό Κανάλι :
Το SER μειώνεται σημαντικά καθώς το SNR αυξάνεται, όπως αναμέναμε. Μάλιστα, για SNR > 15 dB, το SER είναι σχεδόν μηδενικό, που σημαίνει ότι το σύστημα λειτουργεί άψογα, καθώς δεν υπήρξαν καθόλου λανθασμένα σύμβολα κατά τη μετάδοση. Αυτό πρακτικά σημαίνει ότι όλα τα λαμβανόμενα σύμβολα αποκωδικοποιήθηκαν σωστά και δεν υπήρξε απώλεια ή παραμόρφωση στα σύμβολα που μεταδόθηκαν. Το κανάλι δηλαδή είναι τόσο καθαρό που τα σφάλματα είναι μηδενικά. Άρα, η θεωρητική συμπεριφορά του 4-PSK συστήματος σε ιδανικό κανάλι φαίνεται να επιβεβαιώνεται.

Μη Ιδανικό Κανάλι :
Το SER μειώνεται αρκετά λιγότερο, καθώς από 0.5 φτάνει σε 0.42 ακόμα και για τις πολύ υψηλές τιμές SNR. Αυτό δείχνει ότι το κανάλι προκαλεί σημαντική παραμόρφωση, με αποτέλεσμα το θόρυβο να μην είναι ο κύριος παράγοντας που προκαλεί τα σφάλματα. Φαίνεται ότι το μη ιδανικό κανάλι δεν μπορεί να επωφεληθεί από την αύξηση του SNR, κάτι που πιθανώς υποδηλώνει σοβαρή ISI επικάλυψη μεταξύ των συμβόλων, καθιστώντας δύσκολη την ανίχνευση του σωστού συμβόλου. Ακόμη, στις διαμορφώσεις PSK, καθώς αυξάνεται το M, η απόσταση μεταξύ των συμβόλων μειώνεται. Αυτό σημαίνει ότι η διαμόρφωση είναι πιο ευαίσθητη σε λάθη, ειδικά αν υπάρχει μη γραμμικότητα πομπού-δέκτη ή ISI.

8-PSK σε Ιδανικό & Μη Ιδανικό Κανάλι :

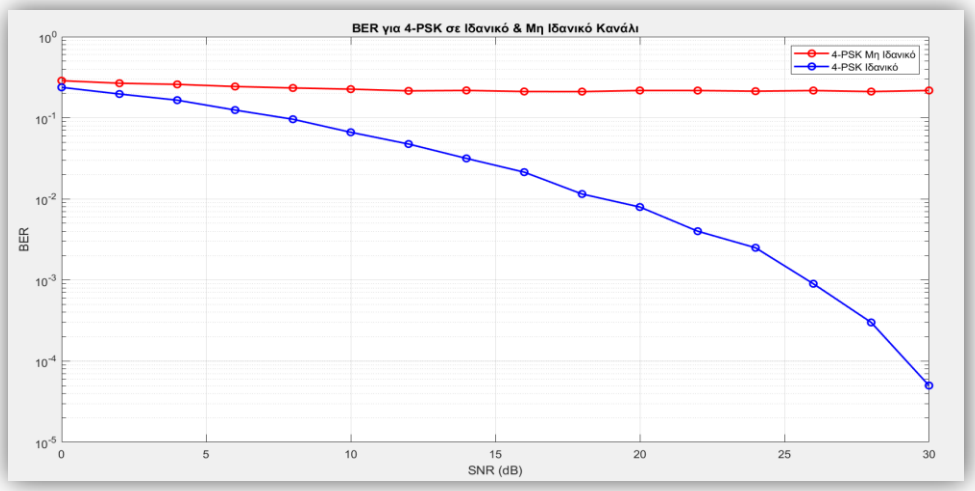
Ιδανικό Κανάλι :
Το SER είναι μεγαλύτερο από το 4-PSK για χαμηλά SNR, καθώς το 8-PSK έχει μικρότερη απόσταση μεταξύ των συμβόλων στον αστερισμό. Η βελτίωση του SER με την αύξηση του SNR είναι σαφής, αλλά η απόδοση είναι χαμηλότερη από το 4-PSK. Το SER μειώνεται, αλλά πιο αργά σε σχέση με το 4-PSK, κάτι που είναι αναμενόμενο.

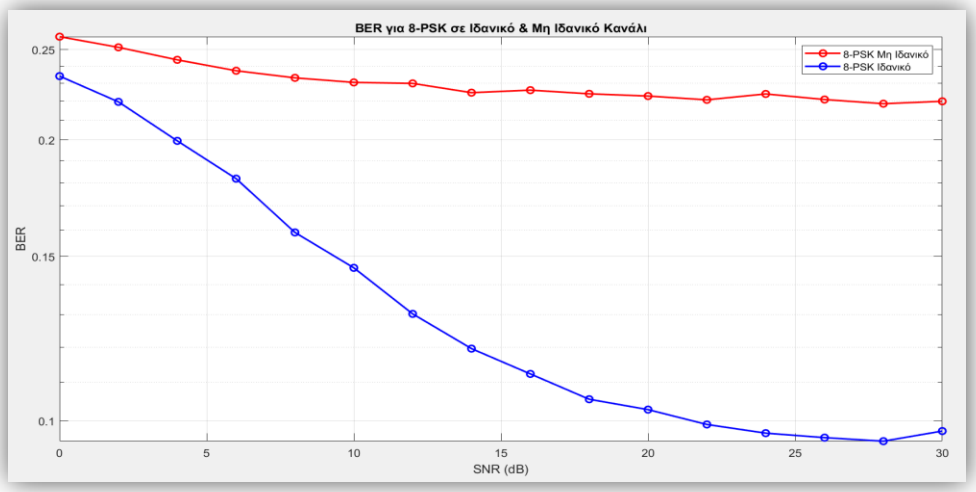
Μη Ιδανικό Κανάλι :
Παρατηρείται ξανά σταθερό μικρή ελάττωση του SER (0.72 μέχρι 0.65), όπως και στο 4-PSK.
Η παραμόρφωση του καναλιού κυριαρχεί και δεν επιτρέπει στο σύστημα να εκμεταλλευτεί την αύξηση του SNR. Η επίδοση είναι χειρότερη από το 4-PSK, αφού το 8-PSK είναι πιο ευαίσθητο σε παραμορφώσεις και παρεμβολές.

4. Μετρήστε την πιθανότητα σφάλματος bit και σχεδιάστε τις καμπύλες Bit Error Rate (BER) για τιμές του SNR=[0:2:30]dB. Σχολιάστε τα αποτελέσματα. Ποιο σύστημα είναι καλύτερο ως προς την πιθανότητα σφάλματος για το ίδιο SNR; Συγκρίνετε τις καμπύλες BER με τις καμπύλες SER που υπολογίσατε προηγουμένως;

Το BER (Bit Error Rate) είναι η πιθανότητα λάθους σε επίπεδο bit και εκφράζει το ποσοστό των λανθασμένων bits σε σχέση με το σύνολο των μεταδιδόμενων bits. Η μόνη αλλαγή που χρειάζεται να κάνουμε είναι να δημιουργήσουμε δύο νέους πίνακες BER_nonideal και BER_ideal που θα αποθηκεύουν τις τιμές του BER για κάθε σημείο του SNR. Για να υπολογίσουμε το BER από το SER αρκεί να διαιρέσουμε το SER με τον αριθμό των bits ανά σύμβολο. Το BER θα είναι πάντα χαμηλότερο από το SER επειδή κάθε σύμβολο περιέχει πολλά bits.

```
        BER_nonideal = zeros(size(SNR_dB_range));
        BER_ideal = zeros(size(SNR_dB_range));
% Υπολογισμός BER
        BER_nonideal(snr_idx) = max(sum(decided_symbols_nonideal ~= S) /
(num_symbols * bits_per_symbol), 1e-10);
        BER_ideal(snr_idx) = max(sum(decided_symbols_ideal ~= S) / (num_symbols *
bits_per_symbol), 1e-10);
```





Εδώ βλέπουμε ότι το ιδανικό κανάλι μειώνει το BER εκθετικά με την αύξηση του SNR, ενώ το μη ιδανικό παρουσιάζει σταθερά λάθη λόγω ISI. Παρατηρούμε ότι το 4-PSK έχει χαμηλότερο BER από το 8-PSK για όλες τις τιμές SNR. Το 8-PSK έχει μεγαλύτερη πιθανότητα λάθους, επειδή τα σύμβολα βρίσκονται πιο κοντά στον αστερισμό, καθιστώντας το πιο ευαίσθητο στον θόρυβο. Οπότε, για το ίδιο SNR, το 4-PSK είναι πιο αξιόπιστο και έχει χαμηλότερο BER. Αυτό συμβαίνει επειδή το 8-PSK χρησιμοποιεί μικρότερες αποστάσεις μεταξύ των συμβόλων, αυξάνοντας τον κίνδυνο λανθασμένης ανίχνευσης.

Οι καμπύλες BER έχουν την ίδια τάση με τις καμπύλες SER, αλλά οι τιμές τους είναι χαμηλότερες. Αυτό συμβαίνει γιατί κάθε σύμβολο έχει πολλά bits και ένα λάθος σύμβολο δεν σημαίνει πάντα λάθος σε όλα τα bits. Συμπεραίνουμε ότι το BER δίνει μια πιο ακριβή εκτίμηση της συνολικής ποιότητας του συστήματος, καθώς λαμβάνει υπόψη το ποσοστό των εσφαλμένων bits και όχι απλώς των συμβόλων.

Για βελτίωση του BER στο μη ιδανικό κανάλι, είναι αρκετά χρήσιμο να χρησιμοποιήσουμε ισοστάθμιση (equalization) για μείωση του ISI. Ο ισοσταθμιστής προσαρμόζεται δυναμικά στις αλλαγές του καναλιού, και εκτελεί τα εξής:

Υπολογίζει τις παραμορφώσεις που έχει προκαλέσει το κανάλι, αναστρέφει αυτές τις επιδράσεις (όσο το δυνατόν περισσότερο) και βελτιώνει την ανίχνευση των συμβόλων, ώστε το SER να μειωθεί.

Όλοι οι κώδικες που χρησιμοποιήθηκαν σε αυτή την εργασία βρίσκονται στο παρακάτω Παράρτημα :

ΜΕΡΟΣ Α

```
clc; clear; close all;

%% Δημιουργία Φίλτρων Πομπού-Δέκτη (SRRC Filter)
roll_off = 0.3;       % Παράγοντας αναδίπλωσης
span = 6;             % Αριθμός περιόδων σηματοδοσίας (Ts)
sps = 4;              % Υπερδειγματοληψία (samples per symbol)

% Χρονικός άξονας για το φίλτρο
t = (-span/2:1/sps:span/2)';

% Υπολογισμός SRRC φίλτρου
h_srrc = (sin(pi*t)./(pi*t)) .* (cos(pi*roll_off*t)./(1 - (2*roll_off*t).^2));
h_srrc(t == 0) = 1 - roll_off + (4 * roll_off / pi); % Ειδική τιμή στο 0
h_srrc(abs(t) == 1/(2*roll_off)) = (roll_off/sqrt(2)) * ((1 + 2/pi) * sin(pi/(2*roll_off)) + (1 - 2/pi)
* cos(pi/(2*roll_off))); % Ειδική τιμή σε ±T/(2β)

% Κανονικοποίηση ώστε να έχει συνολική ενέργεια 1
h_srrc = h_srrc / sqrt(sum(h_srrc.^2));

tx_filter = h_srrc; % Φίλτρο πομπού
rx_filter = h_srrc; % Φίλτρο δέκτη

%% Υπερδειγματοληψία Ακολουθιών και Καναλιού
num_symbols = 10000; % Αριθμός συμβόλων
M = 4; % Διαμόρφωση 4-PSK
symbols = randi([0 M-1], num_symbols, 1); % Τυχαία συμβολική ακολουθία

% Χαρτογράφηση συμβόλων σε αστερισμό 4-PSK
S = exp(1j * (pi/4 + (2 * pi * symbols) / M));

% Υπερδειγματοληψία συμβόλων
S_up = zeros(num_symbols * sps, 1);
S_up(1:sps:end) = S; % Τοποθέτηση συμβόλων κάθε 4 δείγματα

% Κρουστική απόκριση μη ιδανικού καναλιού
h = [0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0 0.21 0.03 0.07];

% Υπερδειγματοληψία καναλιού
h_up = zeros(length(h) * sps, 1);
h_up(1:sps:end) = h;

% Ιδανικό κανάλι (χωρίς παραμόρφωση)
h_ideal = [1];
h_ideal_up = zeros(sps, 1);
h_ideal_up(1) = 1;

%% Διέλευση του Σήματος
% Διέλευση από το φίλτρο πομπού (SRRC)
tx_signal = conv(S_up, tx_filter, 'same');

% Διέλευση από μη ιδανικό κανάλι
rx_signal = conv(tx_signal, h_up, 'same');

% Διέλευση από ιδανικό κανάλι
rx_signal_ideal = conv(tx_signal, h_ideal_up, 'same');
```

```
%% Προσθήκη Λευκού Gaussian Θορύβου
SNR_dB = 10; % Επιθυμητό SNR σε dB

% Υπολογισμός ισχύος του σήματος μετά τη διέλευση από το μη ιδανικό κανάλι
Ps = mean(abs(rx_signal).^2);
Pn = Ps / (10^(SNR_dB/10));
noise = sqrt(Pn) * (randn(size(rx_signal)) + 1j*randn(size(rx_signal)));
rx_signal_noisy = rx_signal + noise;

% Υπολογισμός ισχύος του σήματος μετά τη διέλευση από το ιδανικό κανάλι
Ps_ideal = mean(abs(rx_signal_ideal).^2);
Pn_ideal = Ps_ideal / (10^(SNR_dB/10));
noise_ideal = sqrt(Pn_ideal) * (randn(size(rx_signal_ideal)) + 1j*randn(size(rx_signal_ideal)));
rx_signal_ideal_noisy = rx_signal_ideal + noise_ideal;

%% Διέλευση από το Φίλτρο Δέκτη (SRRC)
rx_filtered = conv(rx_signal_noisy, rx_filter, 'same');
rx_filtered_ideal = conv(rx_signal_ideal_noisy, rx_filter, 'same');

% Υποδειγματοληψία στα κατάλληλα διαστήματα
rx_sampled = rx_filtered(1:sps:end);
rx_sampled_ideal = rx_filtered_ideal(1:sps:end);

% Ορισμός αστερισμού συμβόλων για 4-PSK (QPSK)
symbol_constellation = exp(1j * (pi/4 + (2 * pi * (0:M-1)) / M));

% Προσαρμογή διαστάσεων για σωστό υπολογισμό απόστασης
rx_sampled_matrix = repmat(rx_sampled, 1, M);
rx_sampled_matrix_ideal = repmat(rx_sampled_ideal, 1, M);

% Απόφαση ML (Minimum Euclidean Distance)
[~, detected_symbols] = min(abs(rx_sampled_matrix - symbol_constellation), [], 2);
[~, detected_symbols_ideal] = min(abs(rx_sampled_matrix_ideal - symbol_constellation), [], 2);

% Υπολογισμός του Bit Error Rate (BER)
BER = sum(detected_symbols ~= symbols) / num_symbols;
BER_ideal = sum(detected_symbols_ideal ~= symbols) / num_symbols;

%% Οπτικοποίηση

figure;
subplot(3,1,1);
plot(real(rx_signal(1:200)));
hold on;
plot(imag(rx_signal(1:200)), '--');
title('Σήμα μετά από Μη Ιδανικό Κανάλι');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');

subplot(3,1,2);
plot(real(rx_signal_ideal(1:200)));
hold on;
plot(imag(rx_signal_ideal(1:200)), '--');
title('Σήμα μετά από Ιδανικό Κανάλι');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');

subplot(3,1,3);
plot(real(rx_filtered(1:200)));
hold on;
plot(imag(rx_filtered(1:200)), '--');
title('Φιλτραρισμένο Σήμα μετά από το SRRC Δέκτη (Μη Ιδανικό Κανάλι)');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');
```

```
figure;
plot(real(rx_filtered_ideal(1:200)));
hold on;
plot(imag(rx_filtered_ideal(1:200)), '--');
title('Φιλτραρισμένο Σήμα μετά από το SRRC Δέκτη (Ιδανικό Κανάλι)');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');

figure;
subplot(3,1,1);
plot(real(rx_signal_noisy(1:200)), 'b'); hold on;
plot(imag(rx_signal_noisy(1:200)), 'r--');
title('Σήμα μετά την προσθήκη Θορύβου (Μη Ιδανικό Κανάλι)');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');

subplot(3,1,2);
plot(real(rx_signal_ideal_noisy(1:200)), 'b'); hold on;
plot(imag(rx_signal_ideal_noisy(1:200)), 'r--');
title('Σήμα μετά την προσθήκη Θορύβου (Ιδανικό Κανάλι)');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');

subplot(3,1,3);
plot(real(rx_filtered(1:200)), 'b'); hold on;
plot(imag(rx_filtered(1:200)), 'r--');
title('Φιλτραρισμένο Σήμα μετά από το SRRC Δέκτη (Μη Ιδανικό Κανάλι)');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');

figure;
plot(real(rx_filtered_ideal(1:200)), 'b'); hold on;
plot(imag(rx_filtered_ideal(1:200)), 'r--');
title('Φιλτραρισμένο Σήμα μετά από το SRRC Δέκτη (Ιδανικό Κανάλι)');
legend('Πραγματικό μέρος', 'Φανταστικό μέρος');

figure;
subplot(2,1,1);
stem(real(rx_sampled(1:100)), 'b'); hold on;
stem(real(symbol_constellation(detected_symbols(1:100))), 'ro');
title('Ληφθέντα και Αναγνωρισμένα Σύμβολα (Μη Ιδανικό Κανάλι)');
legend('Ληφθέντα Σύμβολα', 'Αναγνωρισμένα Σύμβολα');
xlabel('Χρονικός Δείκτης');
ylabel('Πλάτος');

subplot(2,1,2);
stem(real(rx_sampled_ideal(1:100)), 'b'); hold on;
stem(real(symbol_constellation(detected_symbols_ideal(1:100))), 'ro');
title('Ληφθέντα και Αναγνωρισμένα Σύμβολα (Ιδανικό Κανάλι)');
legend('Ληφθέντα Σύμβολα', 'Αναγνωρισμένα Σύμβολα');
xlabel('Χρονικός Δείκτης');
ylabel('Πλάτος');
```

ΜΕΡΟΣ Β

Ζητούμενο 1

```
clc; clear; close all;

%% ====== Ορισμός Παραμέτρων ======
M = 4; % M-PSK διαμόρφωση (4-PSK, 8-PSK, 16-PSK)
num_bits = 10000; % Συνολικός αριθμός δυαδικών bit
bits_per_symbol = log2(M); % Bits ανά σύμβολο
num_symbols = num_bits / bits_per_symbol; % Αριθμός συμβόλων

% Δημιουργία ψευδοτυχαίας δυαδικής ακολουθίας
bit_sequence = randi([0 1], num_bits, 1);

% Μετατροπή των bits σε σύμβολα (ομαδοποίηση ανά bits_per_symbol bits)
bit_groups = reshape(bit_sequence, bits_per_symbol, []).';
symbol_indices = sum(bit_groups .* 2.^(bits_per_symbol-1:-1:0), 2); % Δυαδικό σε δεκαδικό

% Διαμόρφωση M-PSK
Es = 1; % Ενέργεια συμβόλου
S = sqrt(Es) * exp(1j * (2 * pi * symbol_indices / M));

%% ====== Οπτικοποίηση του Αστερισμού ======
figure;
plot(real(S), imag(S), 'bo');
grid on; axis equal;
title(['Αστερισμός ', num2str(M), '-PSK']);
xlabel('In-Phase'); ylabel('Quadrature');

disp('Η ψευδοτυχαία δυαδική ακολουθία και η διαμόρφωση M-PSK ολοκληρώθηκαν.');
```

Ζητούμενο 2

```
clc; clear; close all;

%% ====== Ορισμός Παραμέτρων ======
num_bits = 10000; % Συνολικός αριθμός δυαδικών bit
Es = 1; % Ενέργεια συμβόλου
M_values = [4, 8]; % Τάξεις διαμόρφωσης (4-PSK & 8-PSK)

figure;
for i = 1:length(M_values)
    M = M_values(i); % Επιλογή M-PSK
    bits_per_symbol = log2(M); % Bits ανά σύμβολο

    % Προσαρμογή num_bits ώστε να είναι πολλαπλάσιο του bits_per_symbol
    adjusted_num_bits = bits_per_symbol * floor(num_bits / bits_per_symbol);

    % Δημιουργία ψευδοτυχαίας δυαδικής ακολουθίας
    bit_sequence = randi([0 1], adjusted_num_bits, 1);

    % Μετατροπή των bits σε σύμβολα
    bit_groups = reshape(bit_sequence, bits_per_symbol, []).'; % Μετατροπή σε ομάδες bits_per_symbol
    symbol_indices = sum(bit_groups .* 2.^(bits_per_symbol-1:-1:0), 2); % Δυαδικό σε δεκαδικό

    % Διαμόρφωση M-PSK
    S = sqrt(Es) * exp(1j * (2 * pi * symbol_indices / M));
```

```
 % Σχεδίαση αστερισμού
    subplot(1,2,i);
    plot(real(S), imag(S), 'bo');
    grid on; axis equal;
    title([num2str(M), '-PSK Αστερισμός']);
    xlabel('In-Phase'); ylabel('Quadrature');
end

disp('Η διαμόρφωση 4-PSK και 8-PSK ολοκληρώθηκε.');
```

Ζητούμενο 3

```
clc; clear; close all;

%% ====== Ορισμός Παραμέτρων ======
M_values = [4, 8]; % Διαμόρφωση 4-PSK & 8-PSK
SNR_dB_range = 0:2:30; % Τιμές SNR σε dB
num_symbols = 10000; % Αριθμός συμβόλων
sps = 4; % Δείγματα ανά σύμβολο
roll_off = 0.3; % Roll-off factor
span = 6; % Αριθμός περιόδων SRRC

h = [0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0 0.21 0.03 0.07]; % Μη Ιδανικό Κανάλι

%% ====== Δημιουργία SRRC Φίλτρου ======
t = (-span/2:1/sps:span/2)';
h_srrc = (sin(pi*t)./(pi*t)) .* (cos(pi*roll_off*t)./(1 - (2*roll_off*t).^2));
h_srrc(t == 0) = 1 - roll_off + (4 * roll_off / pi);
h_srrc(abs(t) == 1/(2*roll_off)) = (roll_off/sqrt(2)) * ((1 + 2/pi) * sin(pi/(2*roll_off)) + (1 - 2/pi)
* cos(pi/(2*roll_off)));
h_srrc = h_srrc / sqrt(sum(h_srrc.^2)); % Κανονικοποίηση

for m_idx = 1:length(M_values)
    M = M_values(m_idx);
    bits_per_symbol = log2(M);
    SER_nonideal = zeros(size(SNR_dB_range));
    SER_ideal = zeros(size(SNR_dB_range));

    for snr_idx = 1:length(SNR_dB_range)
        SNR_dB = SNR_dB_range(snr_idx);

        % Δημιουργία ψευδοτυχαίας ακολουθίας
        bit_sequence = randi([0 1], num_symbols * bits_per_symbol, 1);
        bit_groups = reshape(bit_sequence, bits_per_symbol, []).';
        symbol_indices = sum(bit_groups .* 2.^(bits_per_symbol-1:-1:0), 2);

        % Διαμόρφωση M-PSK
        S_ref = exp(1j * (2 * pi * (0:M-1) / M));
        S = S_ref(symbol_indices + 1);
        S = S / sqrt(mean(abs(S).^2)); % Κανονικοποίηση ώστε Es = 1


        % Υπερδειγματοληψία συμβόλων
        S_up = zeros(num_symbols * sps, 1);
        S_up(1:sps:end) = S;

        % Διέλευση από το ιδανικό κανάλι
        y_ideal = conv(S_up, h_srrc, 'same');
```

```
% Διέλευση από το μη ιδανικό κανάλι
        h_up = zeros(length(h) * sps, 1);
        h_up(1:sps:end) = h;
        y_nonideal = conv(y_ideal, h_up, 'same');

        % Υπολογισμός θορύβου
        P_N_nonideal = mean(abs(y_nonideal).^2) / (10^(SNR_dB/10));
        noise_nonideal = sqrt(P_N_nonideal) * (randn(size(y_nonideal)) + 1j * randn(size(y_nonideal)));
        y_nonideal_noisy = y_nonideal + noise_nonideal;

        P_N_ideal = mean(abs(y_ideal).^2) / (10^(SNR_dB/10));
        noise_ideal = sqrt(P_N_ideal) * (randn(size(y_ideal)) + 1j * randn(size(y_ideal)));
        y_ideal_noisy = y_ideal + noise_ideal;

        % Φιλτράρισμα με φίλτρο δέκτη SRRC
        y_nonideal_filtered = conv(y_nonideal_noisy, h_srrc, 'same');
        y_ideal_filtered = conv(y_ideal_noisy, h_srrc, 'same');

        % Υποδειγματοληψία
        y_nonideal_sampled = y_nonideal_filtered(sps/2:sps:end);
        y_ideal_sampled = y_ideal_filtered(sps/2:sps:end);

        % ML Detection - Απόφαση
        [~, idx_nonideal] = min(abs(y_nonideal_sampled - S_ref), [], 2);
        decided_symbols_nonideal = S_ref(idx_nonideal);
        [~, idx_ideal] = min(abs(y_ideal_sampled - S_ref), [], 2);


        decided_symbols_ideal = S_ref(idx_ideal);

        % Υπολογισμός SER
        SER_nonideal(snr_idx) = max(sum(decided_symbols_nonideal ~= S) / num_symbols, 1e-10);
        SER_ideal(snr_idx) = max(sum(decided_symbols_ideal ~= S) / num_symbols, 1e-10);
    end


% Δημιουργία νέου γραφήματος για κάθε M-PSK
    figure; hold on;
    semilogy(SNR_dB_range, SER_nonideal, '-ro', 'LineWidth', 1.5, 'DisplayName', [num2str(M), '-PSK Μη
Ιδανικό']);
    semilogy(SNR_dB_range, SER_ideal, '-bo', 'LineWidth', 1.5, 'DisplayName', [num2str(M), '-PSK
Ιδανικό']);

    grid on;
    title(['SER για ', num2str(M), '-PSK σε Ιδανικό & Μη Ιδανικό Κανάλι']);
    xlabel('SNR (dB)');
    ylabel('SER');
    legend;
    disp(['Υπολογισμός & σχεδίαση SER ολοκληρώθηκε για ', num2str(M), '-PSK']);
end
```

Ζητούμενο 4

```
clc; clear; close all;

%% ====== Ορισμός Παραμέτρων ======
M_values = [4, 8]; % Διαμόρφωση 4-PSK & 8-PSK
SNR_dB_range = 0:2:30; % Τιμές SNR σε dB
```

```
num_symbols = 10000; % Αριθμός συμβόλων
sps = 4; % Δείγματα ανά σύμβολο
roll_off = 0.3; % Roll-off factor
span = 6; % Αριθμός περιόδων SRRC
h = [0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0 0.21 0.03 0.07]; % Μη Ιδανικό Κανάλι

%% ====== Δημιουργία SRRC Φίλτρου ======
t = (-span/2:1/sps:span/2)';
h_srrc = (sin(pi*t)./(pi*t)) .* (cos(pi*roll_off*t)./(1 - (2*roll_off*t).^2));
h_srrc(t == 0) = 1 - roll_off + (4 * roll_off / pi);
h_srrc(abs(t) == 1/(2*roll_off)) = (roll_off/sqrt(2)) * ((1 + 2/pi) * sin(pi/(2*roll_off)) + (1 - 2/pi)
* cos(pi/(2*roll_off)));
h_srrc = h_srrc / sqrt(sum(h_srrc.^2)); % Κανονικοποίηση

for m_idx = 1:length(M_values)
    M = M_values(m_idx);
    bits_per_symbol = log2(M);
    BER_nonideal = zeros(size(SNR_dB_range));
    BER_ideal = zeros(size(SNR_dB_range));

    for snr_idx = 1:length(SNR_dB_range)
        SNR_dB = SNR_dB_range(snr_idx);

        % Δημιουργία ψευδοτυχαίας ακολουθίας
        bit_sequence = randi([0 1], num_symbols * bits_per_symbol, 1);
        bit_groups = reshape(bit_sequence, bits_per_symbol, []).';
        symbol_indices = sum(bit_groups .* 2.^(bits_per_symbol-1:-1:0), 2);

        % Διαμόρφωση M-PSK
        S_ref = exp(1j * (2 * pi * (0:M-1) / M));
        S = S_ref(symbol_indices + 1);
        S = S / sqrt(mean(abs(S).^2)); % Κανονικοποίηση ώστε Es = 1

        % Υπερδειγματοληψία συμβόλων
        S_up = zeros(num_symbols * sps, 1);
        S_up(1:sps:end) = S;


 % Διέλευση από το ιδανικό κανάλι
        y_ideal = conv(S_up, h_srrc, 'same');

        % Διέλευση από το μη ιδανικό κανάλι
        h_up = zeros(length(h) * sps, 1);
        h_up(1:sps:end) = h;
        y_nonideal = conv(y_ideal, h_up, 'same');

        % Υπολογισμός θορύβου
        P_N_nonideal = 1 / (10^(SNR_dB/10));
        noise_nonideal = sqrt(P_N_nonideal) * (randn(size(y_nonideal)) + 1j * randn(size(y_nonideal)));
        y_nonideal_noisy = y_nonideal + noise_nonideal;

        P_N_ideal = 1 / (10^(SNR_dB/10));
        noise_ideal = sqrt(P_N_ideal) * (randn(size(y_ideal)) + 1j * randn(size(y_ideal)));
        y_ideal_noisy = y_ideal + noise_ideal;

        % Φιλτράρισμα με φίλτρο δέκτη SRRC
        y_nonideal_filtered = conv(y_nonideal_noisy, h_srrc, 'same');
        y_ideal_filtered = conv(y_ideal_noisy, h_srrc, 'same');

        % Υποδειγματοληψία
        y_nonideal_sampled = y_nonideal_filtered(sps/2:sps:end);
        y_ideal_sampled = y_ideal_filtered(sps/2:sps:end);
```

```
        % ML Detection - Απόφαση
        [~, idx_nonideal] = min(abs(y_nonideal_sampled - S_ref), [], 2);
        decided_symbols_nonideal = S_ref(idx_nonideal);
        [~, idx_ideal] = min(abs(y_ideal_sampled - S_ref), [], 2);
        decided_symbols_ideal = S_ref(idx_ideal);

        % Υπολογισμός BER
        BER_nonideal(snr_idx) = max(sum(decided_symbols_nonideal ~= S) / (num_symbols *
bits_per_symbol), 1e-10);
        BER_ideal(snr_idx) = max(sum(decided_symbols_ideal ~= S) / (num_symbols * bits_per_symbol), 1e-
10);
    end

    figure;
    semilogy(SNR_dB_range, BER_nonideal, '-ro', 'LineWidth', 1.5, 'DisplayName', [num2str(M), '-PSK Μη
Ιδανικό']);
    hold on;
    semilogy(SNR_dB_range, BER_ideal, '-bo', 'LineWidth', 1.5, 'DisplayName', [num2str(M), '-PSK
Ιδανικό']);
    grid on;
    title(['BER για ', num2str(M), '-PSK σε Ιδανικό & Μη Ιδανικό Κανάλι']);
    xlabel('SNR (dB)');
    ylabel('BER');
    legend;
end
disp('Υπολογισμός & σχεδίαση BER ολοκληρώθηκε.');
```