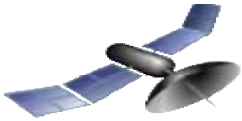
ΤΕΙ Δυτικής Ελλάδας

Σχολή Διοίκησης και Οικονομίας

Τμήμα Πληροφορικής & ΜΜΕ

(παράρτημα Πύργου)



Πτυχιακή Εργασία

Μελέτη συστημάτων λήψης δορυφορικών επικοινωνιών με χρήση αστερισμών APSK. Αλγόριθμοι πρώιμης εκτίμησης απόκλισης συχνότητας και συνθηκών καναλιού.

Δογάνης Θεόδωρος

Πιτσινής Ιωάννης

Επιβλέπων καθηγητής : Καρακίτσος Γεώργιος

2015

Περιεχόμενα

Εισαγωγή

Στην εποχή της παγκοσμιοποίησης όπου η χρήση των τεχνολογιών επικοινωνίας σε πλανητικό επίπεδο αποτελεί καθημερινή ανάγκη, τα δορυφορικά συστήματα έχουν εδραιωθεί ως απαραίτητο στοιχείο. Η πιο απλή μορφή δορυφορικού συστήματος περιλαμβάνει έναν επίγειο πομπό ψηφιακού σήματος, έναν δορυφορικό αναμεταδότη και έναν επίγειο δέκτη. Η ανάγκη προέκυψε από την εξέλιξη της παγκοσμιοποίησης, όπου κρίνονταν πλέον απαραίτητη η μετάδοση μηνυμάτων και δεδομένων ανάμεσα σε δύο σημεία, που λόγω της κλίσης της γης δεν είχαν οπτική επαφή μεταξύ τους . Η λειτουργία ενός δορυφόρου πλέον δεν περιορίζεται μόνο στην αναμετάδοση δεδομένων από επίγειους ψηφιακούς σταθμούς, αλλά είναι σε θέση να πραγματοποιήσει και αυτόνομη εκπομπή σήματος.

Η χρήση δορυφόρων στις επικοινωνίες ξεκίνησε από το 1960 από την Αμερική με την εκτόξευση των δορυφόρων echo1 και αργότερα τον echo2 αλλά η μικρή ισχύς των πυραύλων που τους συνόδευαν στην αρχική τους τοποθέτηση δεν επέτρεπε την τοποθέτησή τους σε ύψος μεγαλύτερο των 1500-1618 χιλιομέτρων. Αυτό είχε ως αποτέλεσμα η γωνιακή ταχύτητά τους να είναι μεγαλύτερη από την γωνιακή ταχύτητα της γης οπότε να απαιτείται κινητό σύστημα λήψης για την επιτυχή επικοινωνία. Οι echo δεν ήταν τίποτε περισσότερο από τεράστια μπαλόνια με συστήματα καθρεπτισμού του εισερχόμενου σήματος πίσω στην επιφάνεια της Γής οπότε και χαρακτηρίστηκαν παθητικοί δορυφόροι με αποτέλεσμα η NASA να εγκαταλείψει το project και να ασχοληθεί στη συνέχεια με ενεργούς δορυφόρους.

Με την εξέλιξη της τεχνολογίας και την δημιουργία ισχυρότερων πυραύλων Delta rocket B τέθηκε σε τροχιά τον Απρίλιο του 1963 από τη NASA ο πρώτος γεωστατικός δορυφόρος Syncom 2, από το ακρωτήριο Κανάβεραλ στην Florida των ΗΠΑ και λίγο αργότερα ο Syncom 3. Γεωστατικοί ονομάστηκαν οι δορυφόροι όπου λόγω του ύψους της τροχιάς τους στα 35.800 km Αποκτούσαν γωνιακή ταχύτητα ίδια με αυτήν της Γής οπότε από σταθερό επίγειο δέκτη φαίνονταν ακίνητοι. Τέτοιου τύπου είναι οι δορυφόροι που χρησιμοποιούνται έκτοτε για να καλύψουν τις επικοινωνιακές μας ανάγκες και με τους οποίους θα ασχοληθούμε στην παρούσα εργασία. Στο 1^{ο} κεφάλαιο μελετάμε τους αστερισμούς επικοινωνιών σύμφωνα με

το πρότυπο ETSI 2012 QPSK, 8APSK, 16APSK και 32APSK. Στο 2^{ο} Κεφάλαιο περιγράφουμε την εφαρμογή που δημιουργήσαμε για την μελέτη και αξιολόγηση των αλγορίθμων που περιλαμβάνονται στο διεθνές πρότυπο. Στο 3^{ο} Κεφάλαιο παρουσιάζουμε τους αλγόριθμους που υλοποιήσαμε για την ανίχνευση θορύβου σε πρώιμο στάδιο. Στο 4^{ο} Κεφάλαιο παρουσιάζουμε την εφαρμογή αξιολόγησης και τα αποτελέσματα της σύγκρισης των πειραμάτων μας με τις θεωρητικές τιμές και στο 5^{ο} Κεφάλαιο παρουσιάζουμε τα συμπεράσματά μας.

1 Μελέτη Δορυφορικών συστημάτων

1.1 Που χρησιμοποιούνται

Κάποιες από τις διευκολύνσεις που μας παρέχει η χρήση δορυφορικών συστημάτων επικεντρώνονται στους εξής τομείς

· Τηλεπικοινωνίες υπό τη μορφή διηπειρωτικών κλήσεων, τηλεοπτικών και ραδιοφωνικών μεταδόσεων τηλεφωνικών επικοινωνιών, επικοινωνία υπολογιστών κ.α.

· Εντοπισμός θέσης και ταχύτητας αντικειμένων (GPS) σε πραγματικό χρόνο και καταγραφή της τροχιάς τους.

· Δορυφορική τηλεόραση και ραδιόφωνο

· Δορυφορικό internet

· Μεταφορά πληροφοριών σε πραγματικό χρόνο - απομακρυσμένη εποπτεία χώρων, αντικατασκοπεία

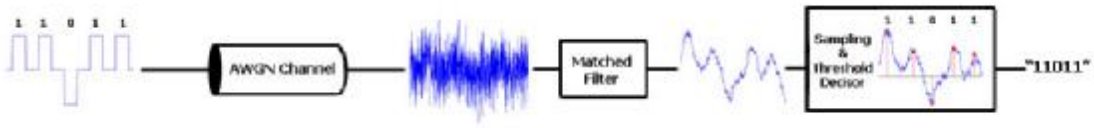
· Μετεωρολογία και πρόβλεψη καιρού

1.2 Επισκόπηση συστήματος

Παρατίθενται κάποια πολύ γενικά στοιχεία για την διαδικασία αποστολής και λήψης του δορυφορικού σήματος. Η διαδικασία είναι χωρισμένη σε τρία τμήματα που αποτελούν και τα φυσικά αντικείμενα του συστήματος – πομπός, μέσο, δέκτης.

Εικόνα 1 Πομπός - μέσο – Δέκτης

Στον πομπό δημιουργείται ένα τυχαίο σήμα το οποίο κωδικοποιείται βάσει των επιλογών του χρήστη. Οι αστερισμοί που χρησιμοποιούμε για τις ανάγκες της εργασίας είναι οι QPSK, 8APSK, 16APSK, 32APSK των οποίων τα χαρακτηριστικά αναλύονται σε επόμενο κεφάλαιο. Το σήμα δημιουργείται έχοντας γνώση της συνάρτησης απεικόνισης στον κατάλληλο αστερισμό οπότε βρίσκεται στην αντίστοιχη ψηφιακή μορφή (των 2, 3, 4 ή 5 bit αντίστοιχα). Στο φίλτρο του πομπού γίνεται υπερδειγματοληψία του σήματος (με προεπιλεγμένη τιμή 8) και το σήμα γίνεται ψευδοαναλογικό (τα επιπλέον σύμβολα παίρνουν τιμές ανάμεσα στην προηγούμενη και την επόμενη του σήματος ώστε να εξομαλύνουν την μετάβαση).



Το σήμα περνάει από λευκό προσθετικό θόρυβο (Gauss) ο οποίος αντιπροσωπεύει το κανάλι. Στο δέκτη υπάρχει ένα φίλτρο που εξομαλύνει τις αποκλίσεις από τις αρχικές θέσεις ισορροπίας των συμβόλων και πραγματοποιεί υποδειγματοληψία για να πάρουμε ένα σήμα μήκους ίσο με το αρχικό. Στη συνέχεια γίνεται αποκωδικοποίηση βάσει του προεπιλεγμένου χάρτη ανιχνεύοντας τις αποστάσεις των συμβόλων από τις επιμέρους θέσεις συμβόλων του χάρτη. Η τελική σειρά bit

συγκρίνεται με την αρχική για να πάρουμε το Bit Error Rate που μας δείχνει την αποτελεσματικότητα του συστήματος.

Για να είναι εφικτό να δουλέψει ένα τέτοιο σύστημα πρέπει να γνωρίζουμε από την αρχή ένα πλήθος λεπτομερειών για τον πομπό, το κανάλι και τον δέκτη που ονομάζονται παράμετροι και είναι αρκετές ώστε η παραμικρή αλλαγή να μας δίνει ένα τελείως διαφορετικό αποτέλεσμα. Με τις παραμέτρους εισόδου θα ασχοληθούμε σε επόμενο κεφάλαιο αξίζει ωστόσο να σημειώσουμε κάποια πολύ σημαντικά στοιχεία :

1. Εκείνο το οποίο «βλέπει» ο δέκτης και καλείται να αποκωδικοποιήσει είναι ένα «σύννεφο» σημείων γνωρίζοντας ωστόσο το μήκος της συμβολοσειράς και το χάρτη αστερισμού που πρέπει να χρησιμοποιήσει. Αυτό συντελεί στο ότι οι δέκτες σχεδιάζονται ειδικά για συγκεκριμένες περιπτώσεις όπως πχ μετάδοση δεδομένων στο σύστημα QPSK.
2. Το κανάλι που χρησιμοποιείται σ' αυτήν την εργασία είναι ένα εξιδανικευμένο κανάλι λευκού θορύβου του οποίου τις παραμέτρους ορίζουμε αυθαίρετα για πειραματικούς σκοπούς. Στην καθημερινή μετάδοση δορυφορικών σημάτων υπεισέρχονται επιπλέον θόρυβοι με τελείως διαφορετικά χαρακτηριστικά των οποίων την κατάσταση δεν μπορούμε να προσεγγίσουμε με ακρίβεια χρησιμοποιώντας το awgn κανάλι.
3. Σε ένα δορυφορικό σύστημα οι παράμετροι είναι πάρα πολλές για να ορίζονται σε κάθε εκτέλεση οπότε οι περισσότερες είναι προεπιλεγμένες (όπως πχ. Η παράμετρος υπερδειγματοληψίας του φίλτρου πομπού). Το σύστημα είναι πλήρως ανοικτό σε πειραματισμούς καθώς έχει προβλεφτεί η εξωτερική (χωρίς να χρειάζεται επεξεργασία) απόδοση τιμών των παραμέτρων.
4. Κατά την εκτέλεση ο χρήστης ερωτάται για την επιλογή αστερισμού που θα χρησιμοποιηθεί και ο πομπός δημιουργεί ένα τυχαίο σήμα, ανάλογο της επιλογής του, σε ψηφιακή μορφή. Υπάρχει ωστόσο η δυνατότητα επεκτασιμότητας του προγράμματος έτσι ώστε να μπορεί να επεξεργαστεί ένα υπάρχων σήμα.

1.3 Μελέτη αστερισμών

Η εργασία μας ασχολείται με την μελέτη της αρχιτεκτονικής και της μετάδοσης ενός δορυφορικού συστήματος, βήμα προς βήμα, από την κωδικοποίηση του σήματος αποστολής από τον πομπό, έως την τελική λήψη από τον δέκτη.

Αξίζει να σημειωθεί ότι εδώ έχουν παραληφθεί τα στάδια κβαντισμού και ψηφιοποίησης καθώς θεωρούμε ότι έχουμε έτοιμο το ψηφιακό σήμα προς εκπομπή ενώ θα επικεντρωθούμε στην κωδικοποίηση και την πορεία που ακολουθείται ως την τελική λήψη.

Στο πρώτο στάδιο το ψηφιακό σήμα μετατρέπεται σε αναλογικό βάσει του αντίστοιχου χάρτη αστερισμού.

Οι χάρτες αστερισμών που θα μελετήσουμε ανά κατηγορία και τύπο δεδομένων έχουν ως εξής :

i. Qpsk

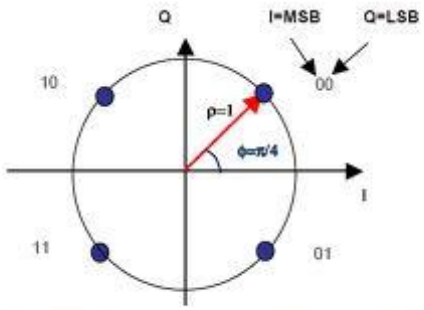
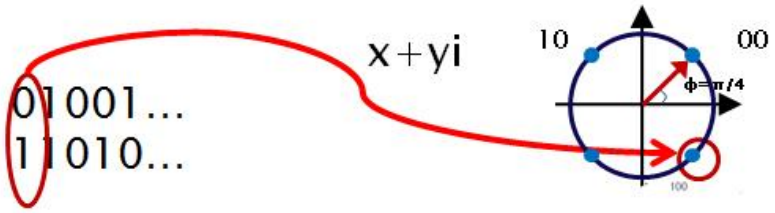


Figure 9: Bit mapping into QPSK constellation

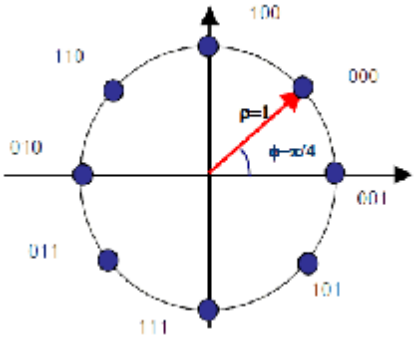
Κάθε σύμβολο περιγράφεται από 2 bit (binary digit). Η απεικόνιση πραγματοποιείται σειριακά. Υπάρχουν 4 ειδών σύμβολα 00, 01, 10, 11 και τέσσερα σημεία στο χάρτη QPSK. Η διαδικασία υλοποιείται ως εξής:



Κάθε σύμβολο αντιστοιχίζεται σε ένα μιγαδικό αριθμό x+yi που υποδηλώνει την θέση του σημείου στο χάρτη και καθορίζεται από την ακτίνα R του κύκλου

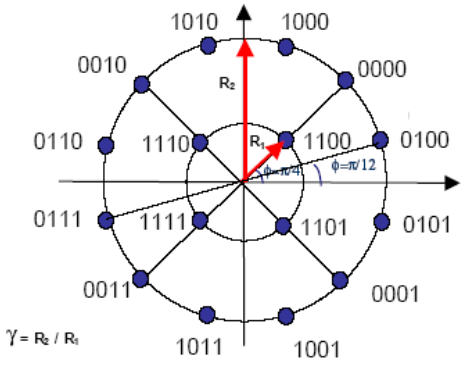
αστερισμού και την γωνία φάσης θ, R(cosθ+sinθ). Στον QPSK έχουμε μόνο έναν κύκλο οπότε η ακτίνα (συνήθως) έχει τιμή 1. Για το σύμβολο του παραδείγματος (01) ο μιγαδικός αριθμός στον οποίο απεικονίζεται είναι ο cos(-pi/4)+sin(-pi/4)*i. Η διαδικασία συνεχίζεται για κάθε σύμβολο μέχρι το τέλος του μηνύματος.

ii. 8apsk



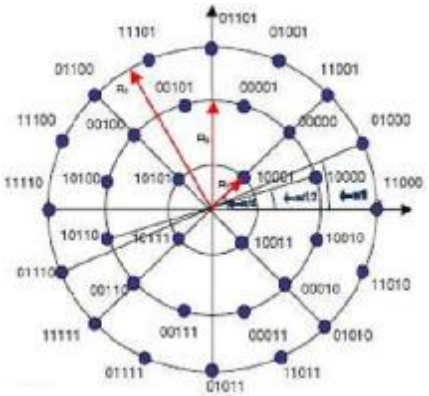
Κάθε σύμβολο αποτελείται από 3 bit. Αυτό σημαίνει ότι μπορώ να αναπαραστήσω 8 διαφορετικά σύμβολα 000, 001, 010, 011, 100, 101, 110, 111.

iii. 16apsk



Κάθε σύμβολο αποτελείται από 4 bit

iv. 32apsk



Κάθε σύμβολο αποτελείται από 5 bit

Ο κάθε χάρτης είναι πλήρως παραμετροποιήσιμος βάσει μέσης ισχύος συμβόλου (Ps) και ρυθμού κωδικοποίησης (coderate) σύμφωνα με τα πρότυπα ETSI . Αυτές οι δύο παράμετροι καθορίζουν τις ακτίνες των κύκλων πάνω στους οποίους απεικονίζονται τα σύμβολα στους χάρτες αστερισμών. Στον QPSK και στον 8APSK έχουμε έναν κύκλο οπότε αυτός ταυτίζεται και την τιμή της μέσης ισχύος. Οι τιμή που αντιστοιχίζεται ενδεικτικά είναι 1 οπότε οι κύκλοι είναι μοναδιαίοι. Όταν τελειώσει η διαδικασία απεικόνισης θα έχω έναν πίνακα από μιγαδικούς αριθμούς ανεξάρτητα από τον χάρτη αστερισμού που είχε επιλεχθεί. Αυτό ανεξαρτητοποιεί τα τμήματα του προγράμματος.

2 Παρουσίαση εφαρμογής

2.1 Μεταβλητές εισόδου

- Γενικές μεταβλητές

```
Ps=1;                   Μέση Ενέργεια ανα σύμβολο = Ισχύς
coderate =2/3;          Ρυθμός κωδικοποίησης 16APSK
coderate32=3/4;         Ρυθμός κωδικοποίησης 32APSK
Radius3=0;              Αρχικοποίηση μεταβλητών ακτίνας 32APSK
Radius2=0;              Αρχικοποίηση μεταβλητών ακτίνας 16APSK
```

Οι αλλαγές στις παραπάνω μεταβλητές επιφέρουν αλλαγές στους χάρτες απεικόνισης των δορυφορικών αστερισμών σύμφωνα με το πρότυπο ETSI EN 302 583

- Μεταβλητές φίλτρου

```
Nsym=6;                 Number of symbols.Αριθμός συμβόλων επιρροής
                        του φίλτρου SRRC
overSamp = 8;           Oversampling Rate. Ρυθμός υπερδειγματοληψίας
rollOff = .35;          Παράμετρος του φίλτρου SRRC (ETSI EN 302
                        583)
```

- Μεταβλητές ενίσχυσης σήματος στο δέκτη

```
gainmode=3;             Επιλογή μεθόδου ενίσχυσης σήματος στο δέκτη
gain=0.5;               Προεπιλεγμένη τιμή ενίσχυσης φίλτρου πομπού
gainR=0.5;              Προεπιλεγμένη τιμή ενίσχυσης φίλτρου δέκτη
```

- Μεταβλητές επιλογής μεγέθους δείγματος, αστερισμού, καναλιού και τμηματοποίησης

```
Txlength=100;           Μέγεθος δείγματος σε σύμβολα. Αν η τιμή
                        είναι πάνω από 200 απενεργοποιούνται
                        αυτόματα με κατάλληλους ελέγχους οι γραφικές
                        παραστάσεις καθώς είναι πολύ πυκνές οι τιμές
                        για να βγει κάποιο συμπέρασμα από αυτές
PSK=0;                  Επιλογή κωδικοποίησης 1.QPSK 2.8PSK 3.16APSK
                        4.32APSK

awgnmode=0;             Αναλόγως τις επιλογές μας ενεργοποιείται
                        αυτόματα η μέθοδος απόδοσης θορύβου με SNR ή
                        με Eb/No δίνοντας τιμές στις μεταβλητές
                        snrinput kai ebnoinput αντίστοιχα
                        Η μεταβλητή awgnmode είναι ο διακόπτης για
                        τον τύπο του καναλιού
                        Για awgnmode=1 το κανάλι δουλεύει με ebno
                        αλλιώς με snr
SNRG=5;                 Θόρυβος καναλιού
```

- Μεταβλητές ενεργοποίησης/απενεργοποίησης γραφικών παραστάσεων

```
overrideplot=0;
```

Επιλογή παραβίασης(override) των απαγορεύσεων σχεδίασης γραφικών παραστάσεων

```
TxSW=1;
ChSW=1;
RxSW=1;
```

Διακόπτες ενεργοποίησης πομπού, καναλιού, δέκτη. Αυτοί οι διακόπτες έχουν τοποθετηθεί για να απενεργοποιούμε όποιο απο τα τρία μέρη του αλγορίθμου (πομπός- κανάλι- Δέκτης) θέλουμε κατα την εκτέλεση. Αν κάποιο απο τα μέρη αυτά απενεργοποιηθεί με διακόπτη, απενεργοποιείται αυτόματα και η δυνατοτήτα αναπαράστασης γραφικών παραστάσεων των αποτελεσμάτων. Υπάρχει ένας τρόπος να Εμφανιστούν οι γραφικες παραστάσεις με τη χρήση του διακόπτη παραβίασης ovverideplot

2.2 Είσοδος – Έξοδος

Αν και οι μεταβλητές εισόδου στο σύστημα είναι αρκετές για την εισαγωγή σε κάθε εκτέλεση, μόνο μία από αυτές είναι απόλυτα αναγκαία, η επιλογή αστερισμού. Η εξωτερική μεταβλητή που αντιστοιχεί σε αυτήν είναι η PSK και αν δεν έχει ήδη οριστεί, μας παρουσιάζεται η οθόνη επιλογής κατά την εκτέλεση :

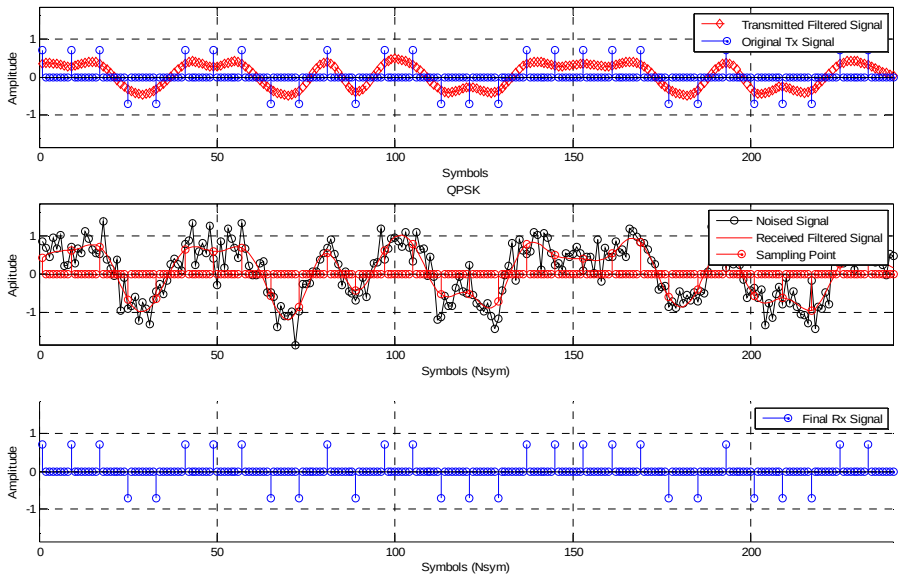
```
Please Select INPUT MODE :
1.    QPSK
2.    8APSK
3.    16APSK
4.    32APSK
selection :
```

Αν επιλέξουμε 1 (QPSK) με προεπιλεγμένες τιμές

```
Nsym=6; overSamp = 8;
rollOff = .35;

gainmode=3;
gain=0.5;
gainR=0.5;
Txlength=100;
```

Το αποτέλεσμα που θα πάρουμε δείχνει ως εξής :

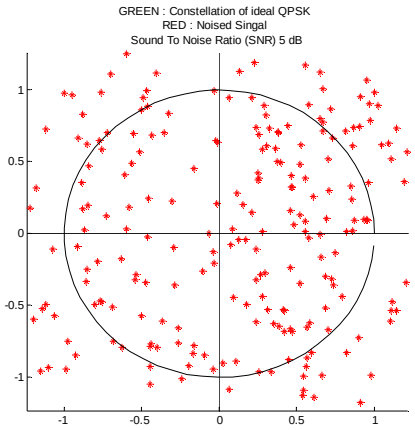


Η πρώτη γραφική παράσταση δείχνει την μία διάσταση (x) του σήματος πομπού μετά την απεικόνιση στο χάρτη αστερισμού με μπλε καθώς και το αποτέλεσμα από την εφαρμογή του φίλτρου πομπού σε ψευδοαναλογική μορφή.

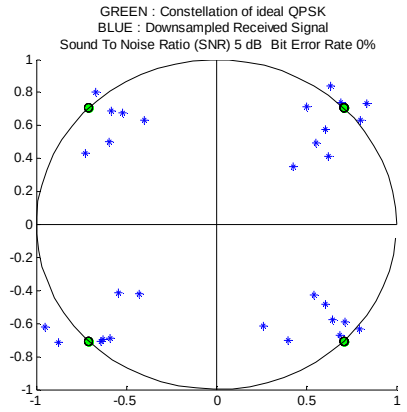
Η δεύτερη γραφική παράσταση δείχνει την παραμόρφωση του φιλτραρισμένου σήματος αφού περάσει μέσα από το κανάλι (με μαύρο). Η κόκκινη γραμμή αναπαριστά το αποτέλεσμα της εφαρμογής του φίλτρου του δέκτη στο θορυβημένο σήμα.

Η Τρίτη γραφική παράσταση δείχνει το αποκωδικοποιημένο σήμα μετά την εφαρμογή του φίλτρου του δέκτη και την υποδειγματοληψία.

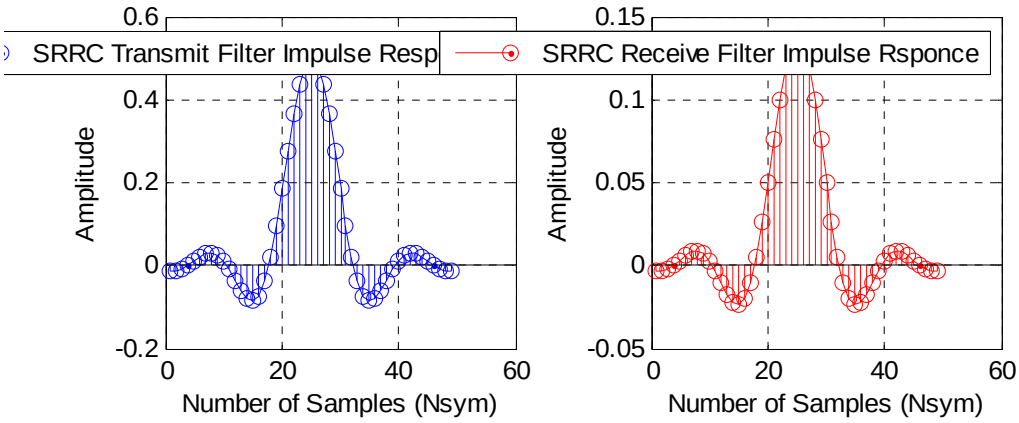
Η επόμενη παράσταση δείχνει το θορυβημένο σήμα που βλέπει ο δέκτης και που καλείται να αποκωδικοποιήσει σε ένα σύστημα QPSK όπου ο συντελεστής SignalToNoise Ratio έχει την τιμή 5.



Το σήμα μετά την εφαρμογή του φίλτρου και την υποδειγματοληψία παίρνει την ακόλουθη μορφή



Τα φίλτρα πομπού και δέκτη έχουν ως εξής :


SRRC Transmit Filter Impulse Resp
SRRC Receive Filter Impulse Rsponce
Amplitude
Number of Samples (Nsym)
0.6
0.4
0.2
0
-0.2
0.15
0.1
0.05
-0.05
0
20
40
60

2.3 Τμηματοποίηση

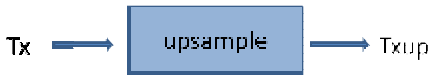
Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο το σύστημα αποτελείται από τρία διαφορετικά τμήματα. Πομπός-κανάλι και δέκτης.

- Πομπός

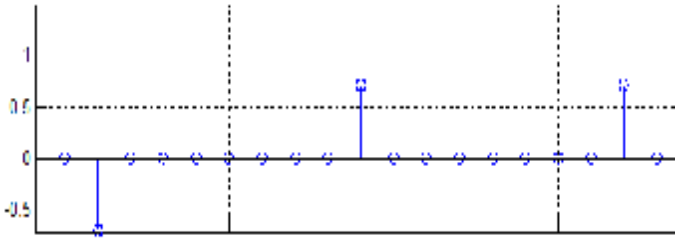
Ο πομπός σε πρώτη φάση δημιουργεί ένα σήμα δεδομένου του αριθμού των συμβόλων που θέλουμε να στείλουμε (Txlength) και της μορφής κωδικοποίησης (QPSK, 8PSK, 16APSK, 32APSK). Στην συνέχεια όπως έχει ήδη περιγραφεί το σήμα κωδικοποιείται ανά σύμβολο σε μιγαδική μορφή

x_1+y_1*i
x_2+y_2*i
x_3+y_3*i
x_4+y_4*i
x_5+y_5*i
x_6+y_6*i
x_7+y_7*i
.
.
. .

Σε δεύτερη φάση το σήμα του πομπού περνά από υπερδειγματοληψία με συγκεκριμένες παραμέτρους.



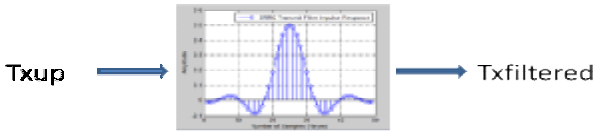
Το Tx είναι το μιγαδικό σήμα ενώ το Txup είναι το υπερδειγματοληφθέν.
Στην προκειμένη περίπτωση τοποθετούνται 7 σύμβολα ανάμεσα σε δύο διαδοχικά. Αυτό σημαίνει ότι το σήμα γίνεται 8 φορές μεγαλύτερο.



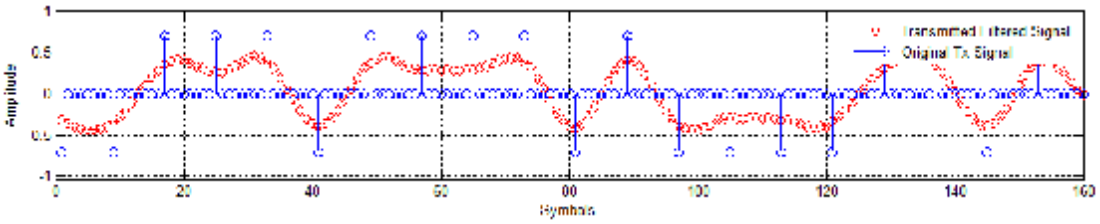
Εικόνα 2 Διάσταση x του σήματος πομπού

Τα μηδενικά αντιπροσωπεύουν τα επιπλέον σύμβολα που δημιουργούνται κατά την υπερδειγματοληψία.

Στο τρίτο και τελευταίο στάδιο του πομπού, το υπερδειγματοληφθέν σήμα (Txup) περνάει από το φίλτρο SRRC (Square Root Raised Cosine) ενισχυμένο κατά 0.5 και μας δίνει το φιλτραρισμένο σήμα (Txfiltered).



Το φίλτρο εξομαλύνει τις τιμές ανάμεσα στα σημεία και το αποτέλεσμα δείχνει ψευδοαναλογικό ωστόσο στην πραγματικότητα είναι ψηφιακό απλά 8 φορές μεγαλύτερο σε πλήθος συμβόλων.



Εικόνα 3 Εφαρμογή SRRC φίλτρου στον πομπό

Όπως θα δούμε στην πορεία, ένα από τα μειονεκτήματα της χρήσης του φίλτρου είναι ότι μειώνει το πλάτος της ταλάντωσης του ψευδοαναλογικού σήματος το οποίο αντιστοιχεί στην διανυσματική ακτίνα των μιγαδικών αριθμών του σήματος. Αυτό είναι ένα πρόβλημα που θα κληθούμε να αντιμετωπίσουμε στον δέκτη όπου θα χρειαστεί να αντιστοιχίσουμε τα σύμβολα στις θέσεις του αστερισμού.

- Κανάλι

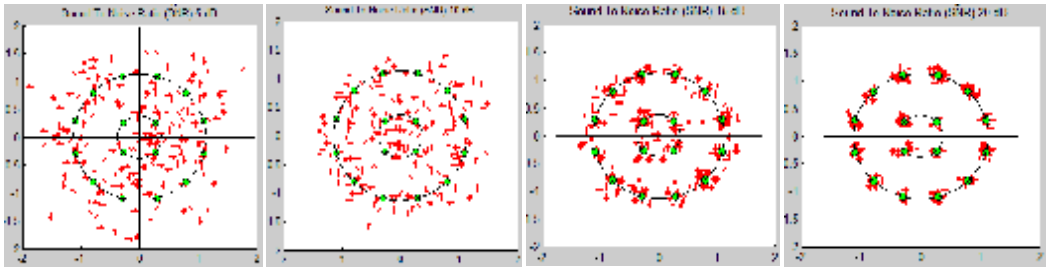
Όπως έχουμε προαναφέρει, εδώ θεωρούμε ότι το κανάλι παρεμβάλει λευκό προσθετικό θόρυβο (Gauss). Ο λόγος που γίνεται αυτό είναι γιατί ο λευκός «Γκαουσιανός» θόρυβος έχει συγκεκριμένα χαρακτηριστικά γνωρίσματα που μας

βοηθάνε ιδιαίτερα στην μελέτη του σήματος που περνά μέσα από ταυτόν.Τα πιο χαρακτηριστικά είναι :

- Ο Λευκός προσθετικός θόρυβος ακολουθεί κανονική κατανομή με αριθμητικό μέσο 0 και τυπική απόκλιση n $Z_i \sim N(0,n)$.Αυτό σημαίνει ότι το αλγεβρικό άθροισμα των αποκλίσεων του σήματος από την θέση ισορροπίας είναι μηδενικό.
- Ο θόρυβος είναι προσθετικός. Αυτό σημαίνει ότι δεν χρειάζεται να εφαρμόσω φίλτρο για να περάσω ένα σήμα μέσα από αυτόν. Αρκεί να δημιουργήσω το διάνυσμα του θορύβου και απλά να τον προσθέσω στο σήμα
- Το μέγεθος της τυπικής απόκλισης μπορεί να παραμετροποιηθεί ανάλογα με την ισχύ του σήματος θορύβου που θέλω να έχω στο κανάλι.

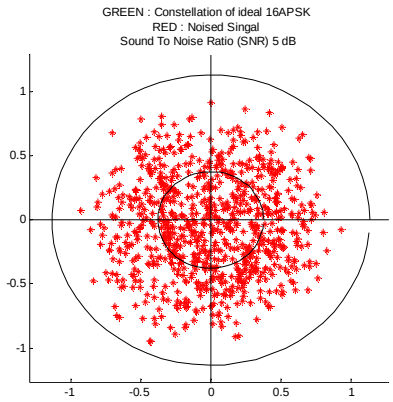
Στα πλεονεκτήματα της χρήσης ενός τέτοιου καναλιού οφείλουμε να προσθέσουμε και το γεγονός ότι υπάρχει έτοιμη εφαρμογή στην Matlab που φιλτράρει το σήμα μας με λευκό προσθετικό θόρυβο (awgn).Η παράμετρος εισόδου του καναλιού είναι το Signal To Noise Ratio (SNR)

Το φιλτραρισμένο σήμα Txfiltered περνάει μέσα από το κανάλι AWGN και μας δίνει το θορυβημένο σήμα Rxnoisy.Παρατίθενται κάποιες από τις καταστάσεις του σήματος μετά από την προσθήκη θορύβου για διάφορες τιμές του SNR.



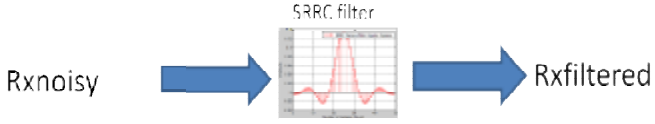
Με πράσινο φαίνονται οι θέσεις των συμβόλων του αστερισμού ενώ με κόκκινο το θορυβημένο σήμα. Όπως είναι προφανές, όσο μεγαλύτερη είναι η τιμή του SNR, τόσο μικρότερη είναι η απόκλιση των σημείων από τις θέσεις του αστερισμού ενώ όσο μικρότερο είναι το SNR τόσο πιο σκορπισμένα είναι τα σημεία στο μιγαδικό επίπεδο.

Υπάρχει μια δεύτερη υλοποίηση του καναλιού με λευκό προσθετικό θόρυβο που στηρίζεται στον καθορισμό της τιμής EbNo ως τιμής εισόδου για το κανάλι. Το EbNo είναι συντομογραφία στην αγγλική για Energy per Bit / Noise Spectral Density. Η ακριβής μετάφραση είναι Ενέργεια ανά Bit/ Φασματική πυκνότητα θορύβου και είναι εφάμιλλη (και ενδεχομένως σχεδόν ανάλογη) του SNR. Αποτελεί δε έναν από τους δημοφιλέστερους τρόπους καθορισμού του καναλιού ως προς το θόρυβο που εισαγάγει στο κανάλι. Το αποτέλεσμα δείχνει σαν σύννεφο σημείων στις δύο διαστάσεις του μιγαδικού επιπέδου και είναι εκείνο που λαμβάνει ο δέκτης



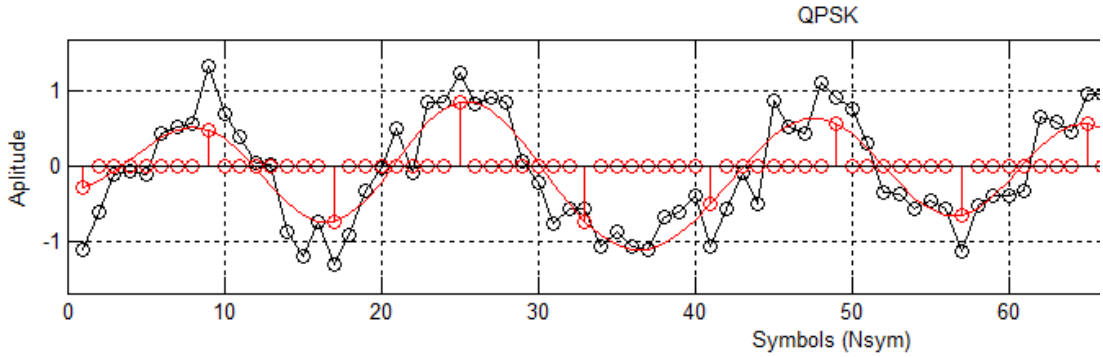
- Δέκτης

Το πρώτο στάδιο στο δέκτη είναι η εξομάλυνση του θορύβου η οποία πραγματοποιείται με ένα φίλτρο παρόμοιο με του πομπού.



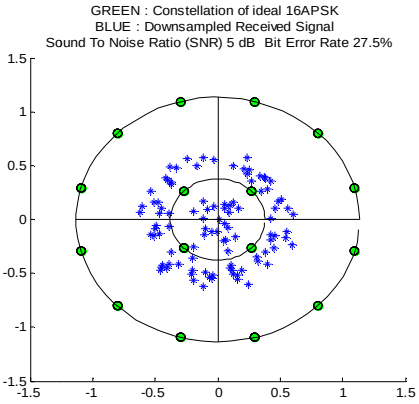
Το φίλτρο παίρνει το θορυβημένο σήμα Rxnoisy και μας δίνει ένα σήμα ίδιου πλήθους σε σύμβολα ψευδοαναλογικής και πάλι μορφής.ο τρόπος με τον οποίο το

καταφέρνει αυτό περιγράφεται στην επόμενη παράσταση της x διάστασης του θορυβημένου σήματος.



Η μαύρη γραμμή αναπαριστά το θορυβημένο σήμα ενώ η κόκκινη το φιλτραρισμένο στο δέκτη σήμα. Τα κόκκινα κυκλάκια στον άξονα δείχνουν τις θέσεις υπερδειγματοληψίας ενώ οι κόκκινες κάθετες τα σημεία στα οποία πρέπει στο επόμενο βήμα να γίνει υποδειγματοληψία για να πάρουμε το σήμα που θα αποκωδικοποιηθεί.

Το σήμα που έχουμε μετά την υποδειγματοληψία έχει την εξής μορφή



Εικόνα 4 Σήμα μετά την υποδειγματοληψία

Το πρόβλημα που είχαμε αναφέρει νωρίτερα σε αυτήν την ενότητα είναι ότι η διανυσματική ακτίνα των σημείων φαίνεται να είναι μικρότερη από εκείνη που έπρεπε να είναι. Αυτό σημαίνει ότι στην τρέχουσα μορφή δεν μπορούμε να

αντιστοιχίσουμε τα σύμβολα στα θεωρητικά σημεία του αστερισμού για να πραγματοποιήσουμε την αποκωδικοποίηση. Οι πιθανές λύσεις είναι οι εξής :

1. Να χρησιμοποιηθεί για το δέκτη ένα φίλτρο ίδιο με το φίλτρο του πομπού.
2. Να χρησιμοποιηθεί φίλτρο ενισχυμένο με διαφορετική ποσότητα από εκείνη του πομπού
3. Να πραγματοποιήσουμε προενίσχυση του σήματος (πριν την αποκωδικοποίηση) με μια ποσότητα ώστε να επαναφέρουμε την μέγιστη διανυσματική ακτίνα του σήματος στα αρχικά της επίπεδα.

Αν υποθέσουμε ότι η μέση διανυσματική απόσταση του αρχικού σήματος ισούται με την ισχύ του τότε η τελική (μετά το κανάλι) θα είναι η ισχύς του θορυβημένου σήματος :

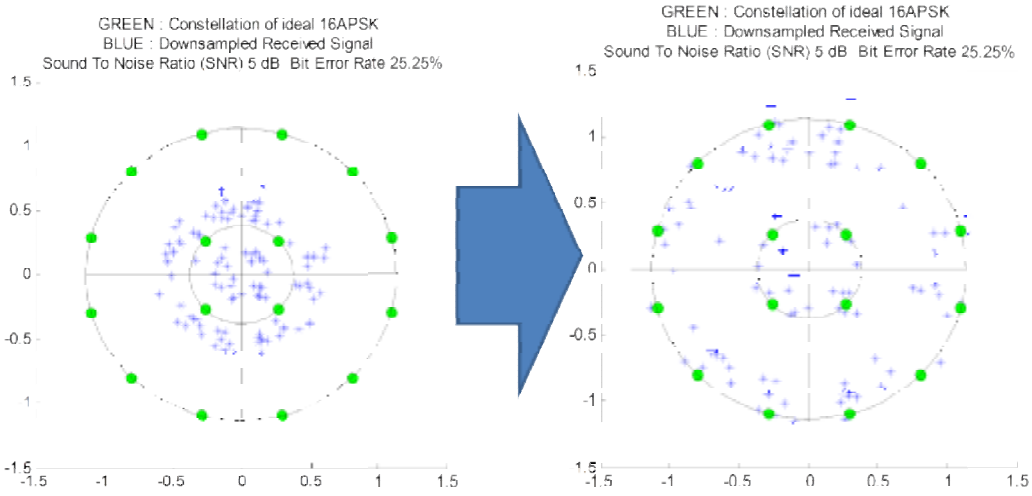
$$\frac{\sum |Rxnoisy|^2}{n}$$

Το οποίο στη matlab μεταφράζεται ως mean(abs(Rxnoisy)^2)

Οπότε για να επαναφέρουμε την διανυσματική ακτίνα στα αρχικά της επίπεδα, πρέπει να πολλαπλασιάσουμε το σήμα με

gain=Ps/ mean(abs(Rxnoisy)^2)

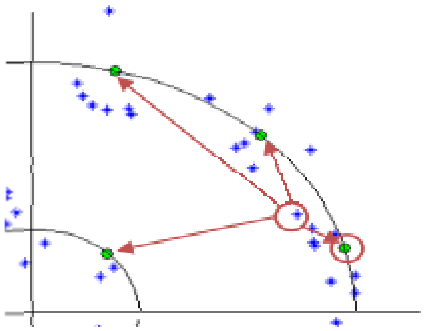
Η εφαρμογή στη matlab θα δώσει το εξής οπτικό αποτέλεσμα



Το τελευταίο βήμα είναι η αποκωδικοποίηση. Αυτή είναι μια σειριακή διαδικασία που πραγματοποιείται σε βήματα.

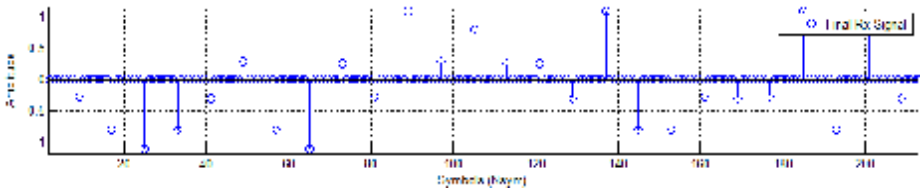
Βημα 1 : Αντιστοιχίζουμε το σύμβολο στο τεταρτημόριο που ανήκει

Βήμα 2 : Υπολογίζουμε την απόστασή του από όλα τα σημεία του τεταρτημορίου του ιδανικού αστερισμού

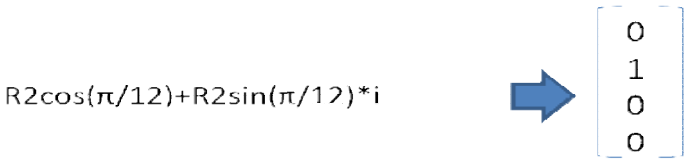


Βήμα 3 : Βρίσκουμε το κοντινότερο σημείο - εκείνο που έχει την μικρότερη απόσταση από το υπό εξέταση - και καταγράφουμε τον μιγαδικό αριθμό που του αντιστοιχεί.

Όταν τελειώσουμε με την μετατροπή όλων των σημείων. Η x διάσταση θα δείχνει όπως και στον πομπό πριν την εφαρμογή του φίλτρου :



Βήμα 4 : Μετατρέπουμε την μιγαδική μορφή σε δυαδική (Αυτό το βήμα μπορεί να πραγματοποιηθεί και μαζικά μετά την μιγαδική αντιστοίχιση όλων των συμβόλων)



Βήμα 5 : Συγκρίνουμε το σήμα που παράγεται σε δυαδική μορφή με εκείνο που είχε δημιουργηθεί στον πομπό και αθροίζουμε τις αποκλίσεις στα bit με την βοήθεια της XOR. Ο μέσος όρος των αποκλίσεων ονομάζεται Bit Error Rate και συμβολίζεται με BER. Αυτή η ποσότητα αποτελεί και τον αντικειμενικότερο συντελεστή για την αποτελεσματικότητα του συστήματος.

3 Ανίχνευση θορύβου σε πρώιμο στάδιο

3.1 Αλγόριθμος ανίχνευσης

Μπορέσουμε να υπολογίσουμε με σιγουριά τον δείκτη SNR (Signal to Noise Ratio) που μας δείχνει το θόρυβο που έχουμε στο κανάλι σε σχέση με το αρχικό σήμα, αρκεί να ξέρουμε τρεις ποσότητες

1. Το πλήθος των σημείων
2. Το θορυβημένο σήμα
3. Το φιλτραρισμένο σήμα του πομπού

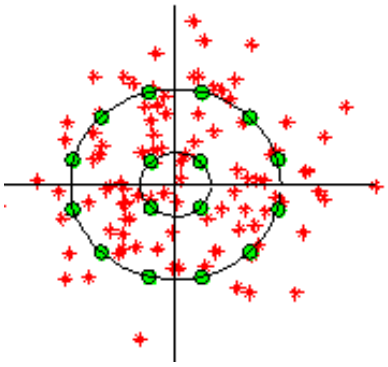
Αυτό προκύπτει από τις εξισώσεις υπολογισμού της ισχύος σήματος και θορύβου.

$$P(n) - \frac{\sum_{i=1}^{N} ||Rxnoisy(i) \quad Txfiltered(i)||^2}{N}$$

$$P(s) - \frac{\sum_{i-1}^{N} ||Txfiltered(i)||^2}{N}$$

$$SNR - 10Log_{10}\frac{Ps}{Pn}$$

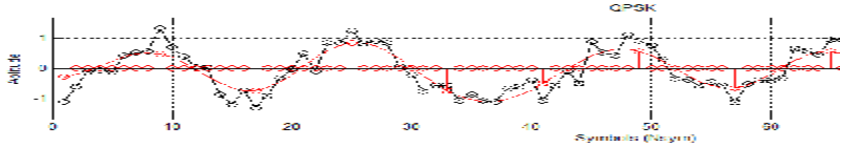
Αυτό που δέχεται ο δέκτης ως είσοδο ωστόσο είναι το εξής :



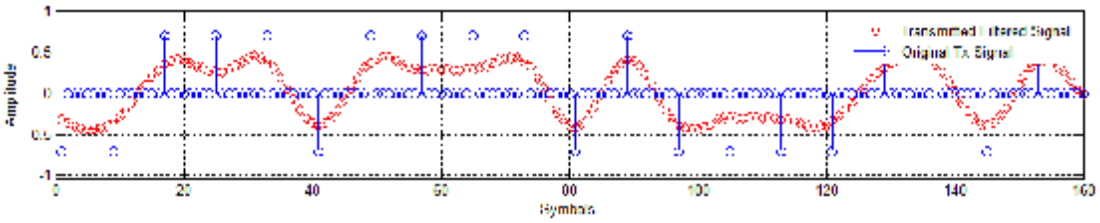
Από το οποίο μπορεί να εξάγει

1. Το σύνολο των συμβόλων
2. Το θορυβημένο σήμα

Δεν μπορεί όμως να έχει το φιλτραρισμένο σήμα του πομπού. Υπάρχει ωστόσο ένα φίλτρο στο δέκτη όπου το σήμα που παράγεται έχει τη μορφή (στην x διάσταση)



Αυτό δείχνει να έχει την ίδια μορφή με το φιλτραρισμένο σήμα του πομπού.



Με τη διαφορά ότι το φιλτραρισμένο σήμα του δέκτη είναι άμεσα διαθέσιμο στο δέκτη σε αντίθεση με το φιλτραρισμένο σήμα του πομπού που χρειαζόμαστε για να βρούμε με βεβαιότητα το SNR.

Για να πειραματιστούμε, αντικαθιστούμε στις εξισώσεις το φιλτραρισμένο σήμα του πομπού με το φιλτραρισμένο σήμα του δέκτη και πλέον γίνονται

$$P(n) = \frac{\sum_{i=1}^{N} ||Rxnoisy(i) - \text{Rxfiltered(i)}||^2}{N}$$

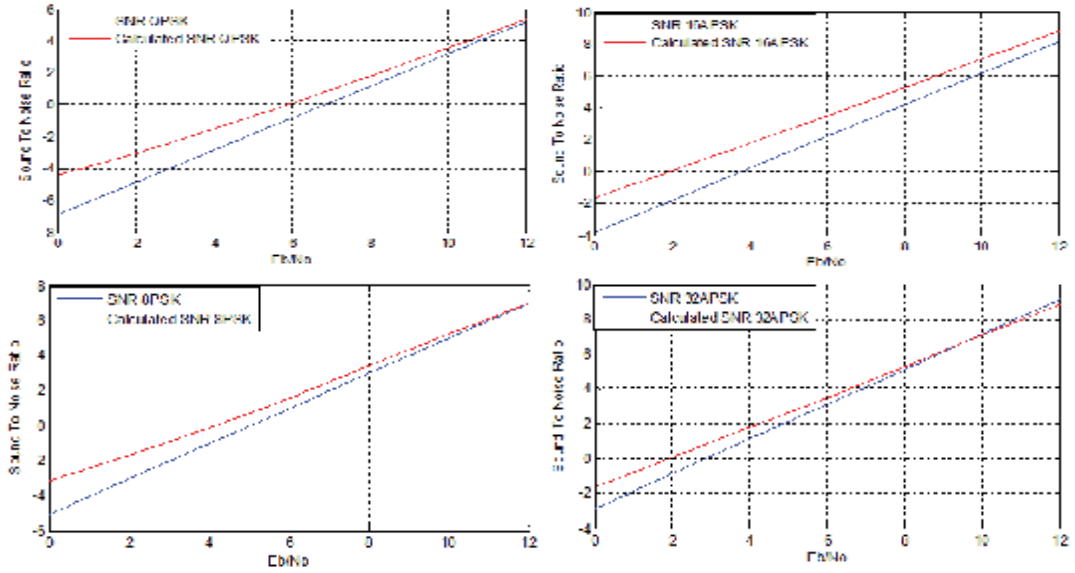
$$P(s) = \frac{\sum_{i=1}^{N} ||\ \text{Rxfiltered(i)}\ ||^2}{N}$$

$$SNR = 10Log_{10}\frac{Ps}{Pn}$$

Πλέον έχουμε όλα όσα χρειαζόμαστε για να υπολογίσουμε το νέο SNR. Το μόνο που μένει είναι να δούμε αν το μοντέλο συμπίπτει και πειραματικά.

3.2 Αποτελεσματικότητα αλγορίθμου

Για να μπορέσουμε να βγάλουμε ασφαλή συμπεράσματα, καθορίσαμε το κανάλι με την ποσότητα EbNo έτσι ώστε να έχουμε ένα σταθερό μέτρο σύγκρισης. Μετά από πειραματισμό προαναφερθείσες εξισώσεις, οι ευθείες που προκύπτουν στο διάγραμμα SNR/ EbNo έχουν ως εξής :



Με μπλε απεικονίζονται οι θεωρητικές τιμές ενώ με κόκκινο οι υπολογισμένες με την νέα σχέση. Είναι προφανές ότι το πείραμα στέφθηκε με επιτυχία καθώς οι δύο ευθείες έχουν πολύ καλή συσχέτιση. Είναι δε προφανές ότι η αποτελεσματικότητα της μεθόδου εξαρτάται σε μεγάλο βαθμό από το θόρυβο του καναλιού. Δηλαδή όσο μικρότερο θόρυβο έχω, τόσο καλύτερη είναι η προσέγγιση του SNR καθώς το φιλτραρισμένο σήμα του δέκτη τείνει να ταυτιστεί με το φιλτραρισμένο σήμα του πομπού.

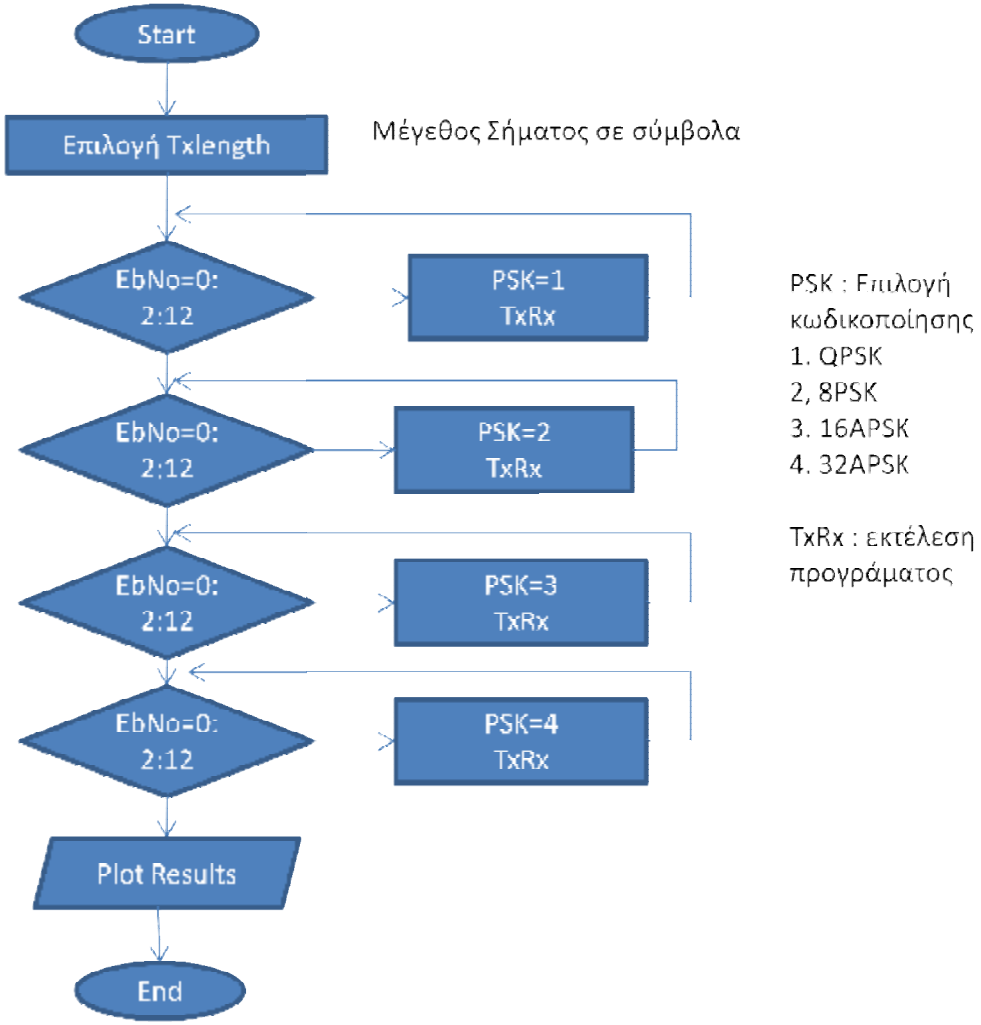
Συνοπτικά η μέθοδος αυτή παρουσιάζει τα εξής πλεονεκτήματα

- Γρήγορη υλοποίηση στο δέκτη
- Πολύ Καλή προσέγγιση

4. Αξιολόγηση βάσει προτύπων

4.2 Ανάλυση εφαρμογής αξιολόγησης

Για να μπορέσουμε να αξιολογήσουμε τα αποτελέσματα, σχεδιάσαμε ένα αλγόριθμο που καλεί επαναληπτικά την εφαρμογή Δημιουργίας, αποστολής και λήψης του δορυφορικού σήματος για βαθμωτές τιμές του EbNo. Το διάγραμμα ροής του έχει ως εξής



Η μόνη επιλογή εισόδου εδώ είναι το πλήθος των συμβόλων που θέλουμε να χρησιμοποιήσουμε. Οποιαδήποτε άλλη μεταβλητή μπορεί να παραμετροποιηθεί εντός του αρχείου TxRx που είναι και η αρχική εφαρμογή αποστολής-λήψης σήματος. Όπως φαίνεται στο διάγραμμα ροής έχουμε

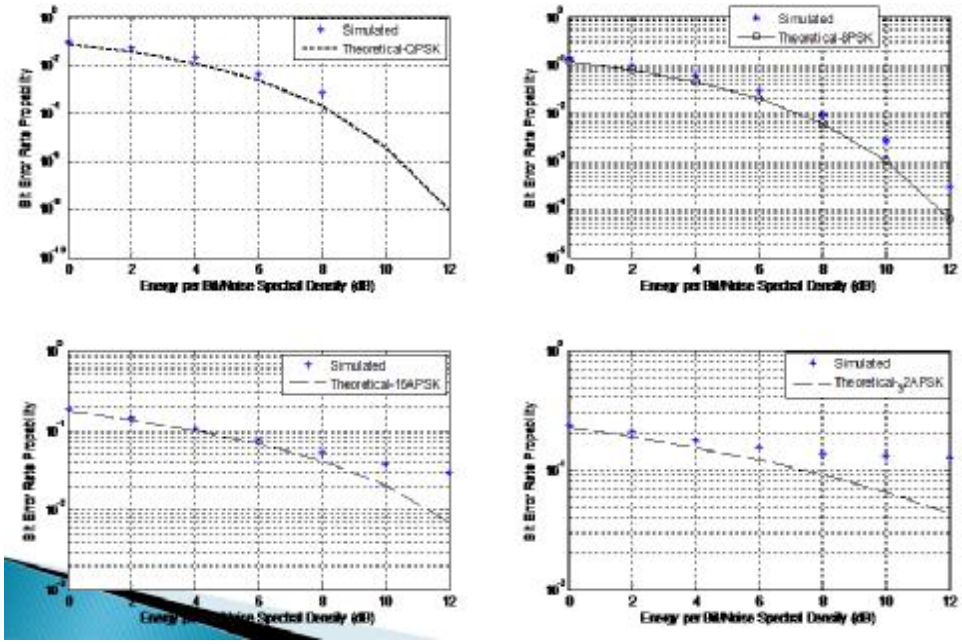
τέσσερεις επαναληπτικές δομές, μία για κάθε ένα από τα συστήματα QPSK, 8PSK, 16APSK και 32APSK. Η κάθε ρουτίνα εκτελεί το TxRx για όλες τις τιμές του EbNo από 2 ως 12 και αποθηκεύει τις τιμές της μεταβλητής Bit Error Rate στους πίνακες ber1, ber2, ber3 και ber4 αντίστοιχα.

Αφού τελειώσει με τις επαναλήψεις, η εφαρμογή υπολογίζει τις θεωρητικές τιμές των συστημάτων από τις ενσωματωμένες εντολές της matlab berawgn και τις αναπαριστά μαζί με τα BER που μόλις υπολογίσαμε.

4.3 Παρουσίαση αποτελεσμάτων

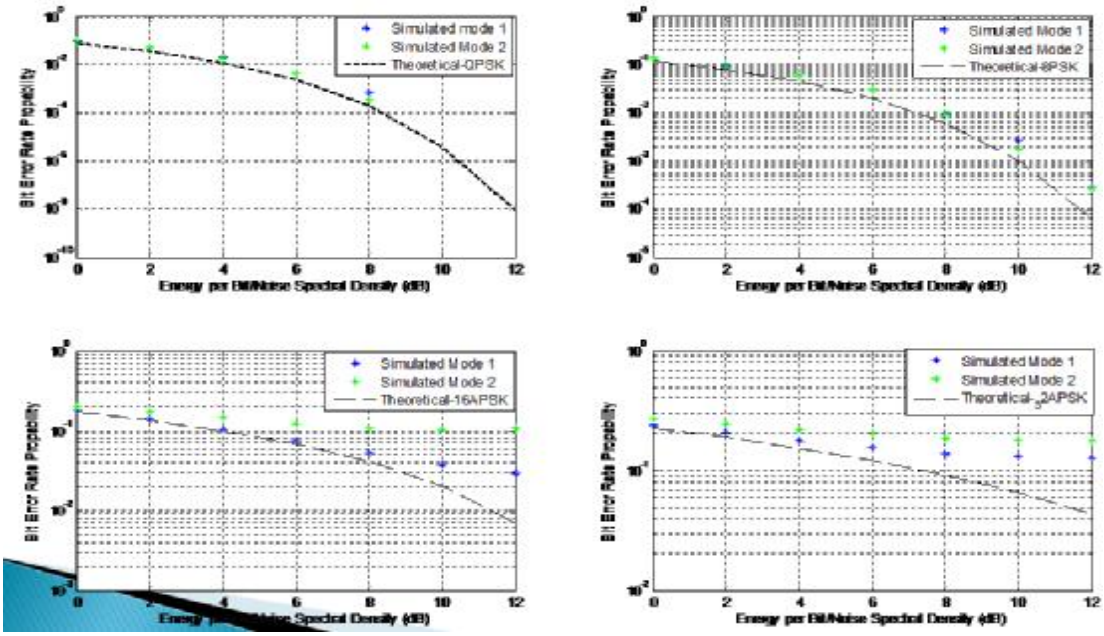
Για πειραματικούς λόγους αλλά και για να επαληθεύσουμε τις θεωρίες που χρησιμοποιήθηκαν κατά την εργασία, υπολογίσαμε την απόδοση του συστήματος και στις τρεις περιπτώσεις ενίσχυσης του σήματος στο Δέκτη. Το πείραμα μας έδωσε τα εξής αποτελέσματα ανά περίπτωση

Α. Χρήση φίλτρου SRRC με ίδια χαρακτηριστικά με εκείνο του πομπού.



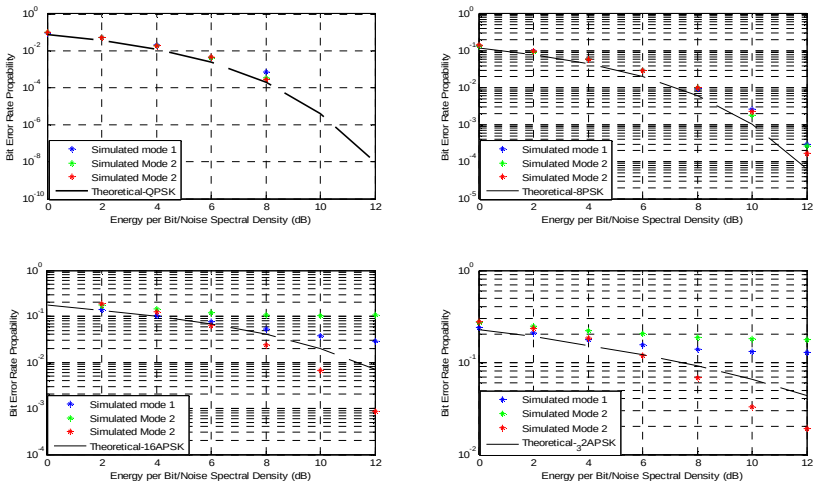
Ο κάθετος άξονας στα διαγράμματα αντιπροσωπεύει την ποσότητα Bit Error Rate Propability και η οριζόντια την Energy per Bit /Noise Spectral Density. Η μαύρη γραμμή δείχνει τις θεωρητικές τιμές ενώ τα μπλε σημεία είναι τα αποτελέσμτα του πειράματος. Χρησιμοποιώντας «matching filters» στα συστήματα QPSK και 8PSK στα πρώτα δύο συστήματα έχουμε πολύ καλή προσέγγιση των θεωρητικών τιμών ενώ στα άλλα δύο, όσο μεγαλώνει ο θόρυβος, παρουσιάζεται απόκλιση.

Β. Προσθέτω τις τιμές για χρήση φίλτρου SRRC με πειραματική ενίσχυση φίλτρου (gain=Ps/mean(abs(Rxnoisy)^2)) και τα διαγράμματα παίρνουν την μορφή :



Με πράσινο φαίνεται η νέα προσέγγιση. Στα συστήματα QPSK και 8PSK φαίνεται να έχει μεγαλύτερη εγγύτητα με τις θεωρητικές τιμές ενώ στα 16APSK και 32APSk φαίνεται να υπάρχει μεγαλύτερη απόκλιση.

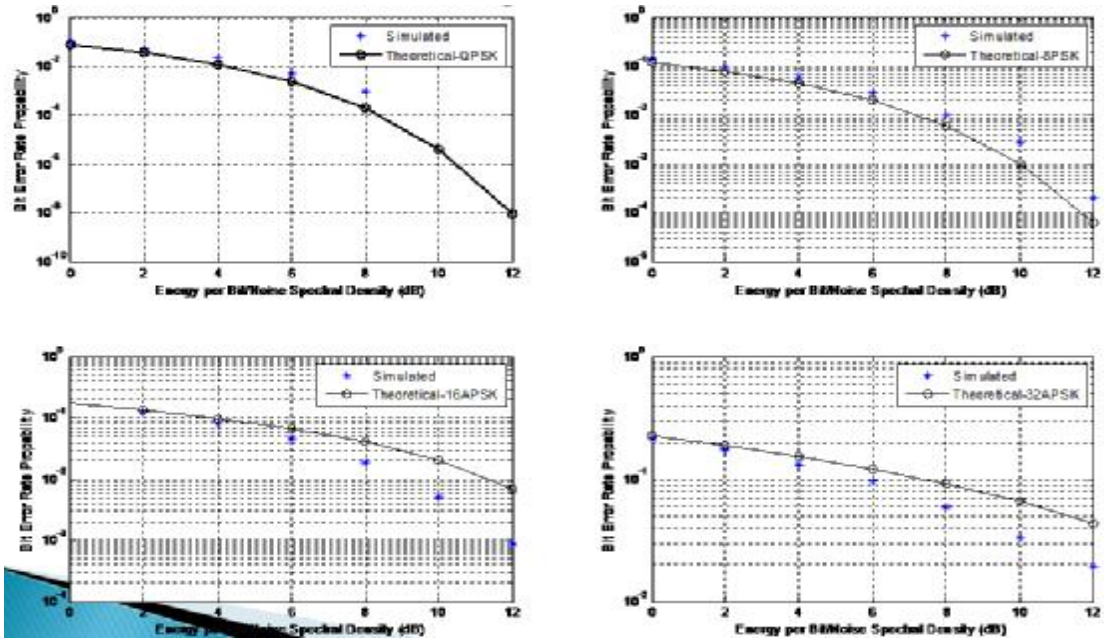
Γ. Αν στα διαγράμματα προσθέσω τις πειραματικές τιμές από τη μέθοδο της προενίσχυσης με το ίδιο gain (mean(abs(Rxnoisy)^2)) το αποτέλεσμα θα δείχνει κάπως έτσι



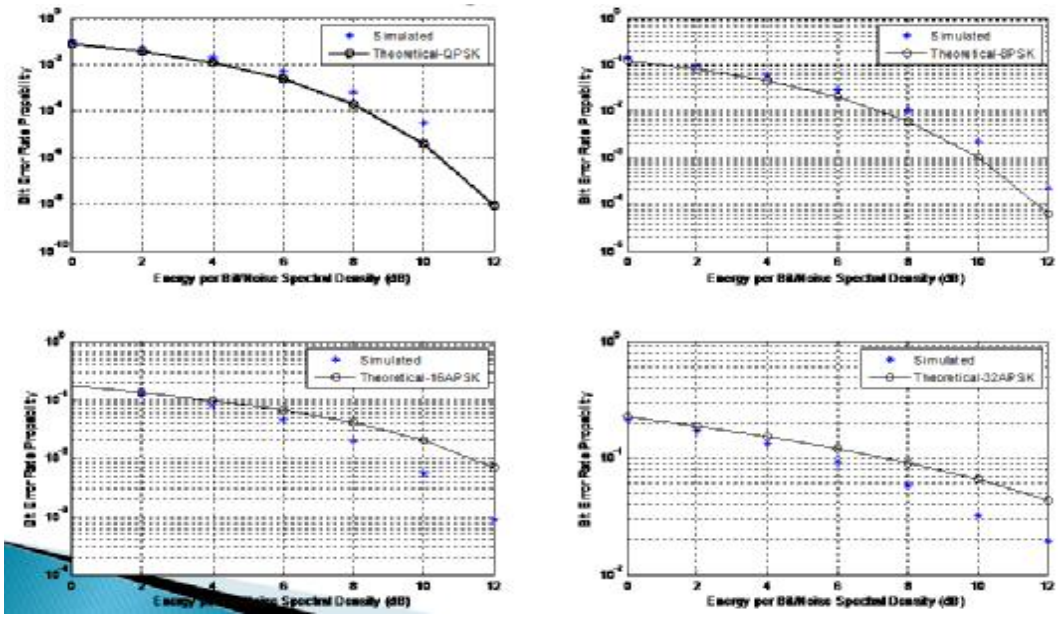
Εδώ είναι προφανές ότι αυτή η μέθοδος, όχι μόνο έχει καλύτερη προσέγγιση στις θεωρητικές τιμές των QPSK και 8PSK αλλά στα 16APSK και 32APSK έχουμε απόδοση καλύτερη από το θεωρητικό μοντέλο

Μια άλλη ερευνητική πρόταση που διαμορφώθηκε στις αρχές της εργασίας ήταν ότι χρειαζόμαστε δείγμα τουλάχιστον 1Mbot (= 1.000.000 σύμβολα) για να εξάγουμε ασφαλή συμπεράσματα για το μοντέλο μας.Τα πειράματά μας έδειξαν τα εξής :

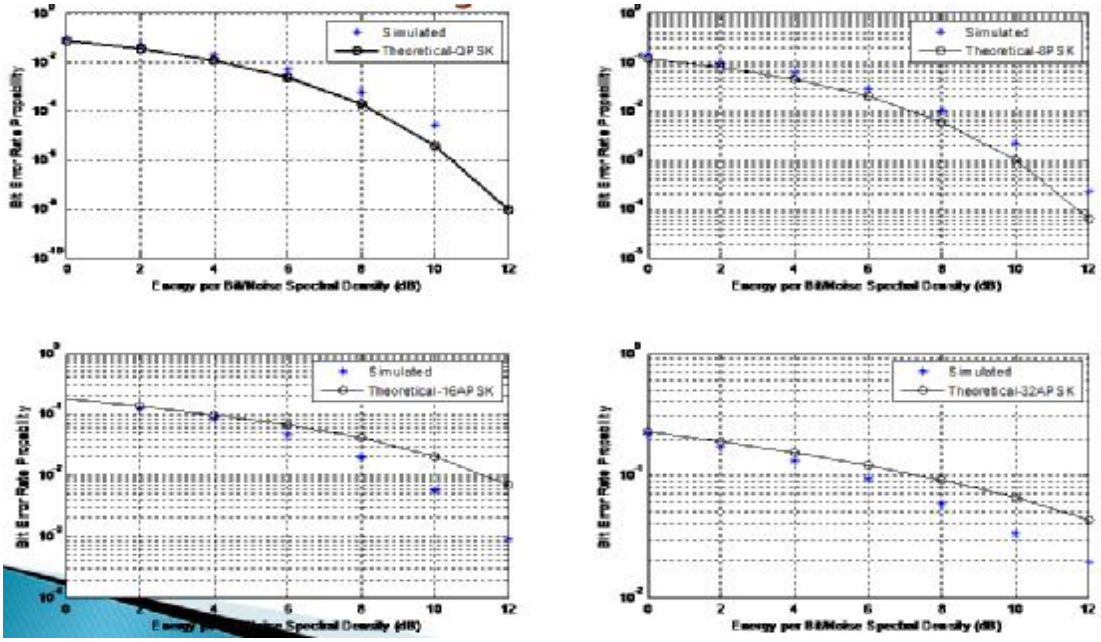
Για το ερευνητικό μας μοντέλο (το τελευταίο που περιγράφηκε) για 10.000 σύμβολα το πρόγραμμα αξιολόγησης (ber_ebno) μας δίνει



Για 100.000 σύμβολα :



Και για 1.000.000 σύμβολα (1 Mbot)



Από τα παραπάνω είναι προφανές ότι η διαφορά ανάμεσα στα δύο πρώτα διαγράμματα είναι αμελητέα ενώ ανάμεσα στο δεύτερο και στο τρίτο είναι μη ορατή. Αυτό μας οδηγεί στο να συμπεράνουμε ότι δεν χρειαζόμαστε 1Mbot για να εξάγουμε συμπεράσματα για το σύστημά μας και πως η ποσότητα των 100.000 συμβόλων είναι αρκετή αφού πλέον αυτών δεν μπορούμε να παρατηρήσουμε διαφορά.

5. Αποτίμηση

Μελετήσαμε ένα πλήρες δορυφορικό σύστημα από την δημιουργία μέχρι και την αποκωδικοποίηση όσον αφορά τη χρήση των αστερισμών QPSK 8PSK, 16APSK και 32 APSK το οποίο σχεδιάσαμε και υλοποιήσαμε στην μαθηματική γλώσσα προγραμματισμού Matlab.

Δημιουργήσαμε έναν αλγόριθμο ανίχνευσης θορύβου σε πρώιμο στάδιο του δέκτη με ακρίβεια μεγαλύτερη του 90%.

Υλοποιήσαμε ένα ολόκληρο πρόγραμμα αξιολόγησης δορυφορικών μοντέλων σε Matlab με χρήση επαναληπτικών δομών και υλοποιήσαμε ένα μοντέλο δέκτη με απόδοση αποκωδικοποίησης καλύτερη από την θεωρητική.

6. Βιβλιογραφία

Καλαλάς Χ. , Αντιστάθμιση Μη Γραμμικών Φαινομένων Σε Δορυφορικούς Αναμεταδότες. Εφαρμογή Στο Πρότυπο Dvb-s2, Εθνικό Μετσόβειο Πολυτεχνείο, Αθήνα, 13-Ιού-2011

Κωτσόπουλος Σ."Συστήματα Ευρείας Εκπομπής", , Εκδόσεις Πανεπιστημίου Πατρών

Σταυροπούλου Α., Προαντιστάθμιση Μη Γραμμικής Ενίσχυσης Σε Σήματα Dvb-s2, Εθνικό Μετσόβειο Πολυτεχνείο, Αθήνα, 19-Απρ-2013

Φραζή Γ. , Μελέτη και αντιστάθμιση μη γραμμικής συμπεριφοράς στο δορυφορικό κανάλι του προτύπου DVB-S2. Καραγιαννίδης Γεώργιος , Θεσσαλονίκη, Ελλάδα.

A. R. Hammons, Jr. and H. El Gamal, "On the theory of space&ndash,time codes for PSK modulation", IEEE Trans. Inform. Theory, 2000

A. R. Hammons, Jr., P. V. Kumar, A. R. Calderbank, N. J. A. Sloane, and P. Sole, "The z_4 linearity of Kerdock, Preparate, Goethals and related codes", IEEE Trans. Inform. Theory, vol. 40, pp.301 -319 1994

Bit-Error-Rate Simulation Using Matlab James E. Gilley Chief Scientist Transcrypt International, Inc. jgilley@transcrypt.com August 19 2003

BORKO, F., SYED, A. Handbook of Mobile Broadcasting, DVB-H, DMB, ISDB-T and MEDIAFLO. Taylor & Francis Group, LCC, 2008.

C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," IEEE Trans. Commun., vol. 44, no. 10, pp. 1261-1271, 1996.

C. Rapp, "Effects of HPA - Nonlinearity on a 4-DPSK / OFDM signal for a digital sound broadcasting system," in 2th Europ. Conf. on Satellite Commun., Oct 1991, pp. 179-184.

C. Xia-xia,J. Hui-juan,W. Ming-gang,(The 54th Research Institute of CETC,Shijiazhuang Hebei 050081,China), "Implementation of Soft Demapping Technique for 8PSK?16APSK and 32APSK Modulation", 2011-04

D. Aktas and M. P. Fitz, "Distance specta for space&ndash,time trellis coded modulations", Int. Symp. on Information Theory, 2001

D. Xiaodai , N. C. Beaulieu and P. H. Wittke "Error probabilities of two-dimensional M-ary signaling in fading", IEEE Trans. Commun., vol. 47, pp.352 -355 1999

Decoding of low-complexity signals transmitted by a constellation modulation, US Patent 7336718, Feb. 2008.

DVB Fact Sheet, Digital Video Broadcasting (DVB); DVB-SH Satellite Services to Handhelds, 2010.

E. Casini, R. De Gaudenzi, A Ginesi, "DVB-S2 modem algorithms design and performance over typical stellite channels ", International Journal on Sattlelite Communication Networks. 2004 ; 22

E. Teletar, Capacity of multi-antenna Gaussian channels, 1995 :AT&amp,T Bell Labs

E. W. Cheney, Introduction to Approximation Theory, American Mathematical Society, Rhode Island, 1998.

E.Chen, J. L. Koslov. V. Mignone, J. Santoru, "DVB-S2 Backward-compatible modes: a Bridge Between the Present and the Future". International Journal on Satellite Communication Networks, 2004; 22

ETSI EN 300 421 v 1.1.2 : "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services"

ETSI EN 302 307 V1.2.1, "Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications," Apr. 2009.

ETSI TS 102 584 V1.2.1 (2011-01). Digital Video Broadcasting (DVB), DVB-SH Implementation Guidelines. Technical Specification. ETSI, 2011.

ETSI TS 302 583, Digital Video Broadcasting (DVB); Framing Structure, channel coding and modulation for Satellite Services to Handheld devices (SH) below 3 GHz, v.1.1.2, 2010.

f. harris, “Multirate Signal Processing in Transmitter & Receiver Designs,” UCLA Extension course, November, 2000.

Fish A., Gurevich S., Hadani R., Sayeed A., and Schwartz O.. Computing the Matched Filter in Linear Time (December 2011).. arXiv:1112.4883. Arxiv.

G. E. Corazza et al., "Performance evaluation of APSK modems," ARCES - Thales Italia Partner., Tech. Rep., 2006.

G. E. Corazza, Ed., Digital Satellite Communications. Springer, 2007.

G. J. Foschini and M. Gans, "On the limits of wireless communication in a fading environment when using multiple antennas", Wireless Pers. Commun., vol. 6, pp.311 -335 1998

G. Karam and H. Sari, "A data predistortion technique with memory for QAM radio systems," IEEE Trans. Commun., vol. 39, no. 2, pp. 336-344, 1991.

G.Karam and H.Sari, " A Data Pre-distortion Technique with Memory for QAM Radio Systems", IEEE Transcations on Communications, Vol.Com-39. No.2, pp336-344, February 1991.

Graychip, Inc., “GC4016 MULTI-STANDARD QUAD DDC CHIP DATA SHEET,” Revision 1.0, August 27, 2001.

H. Bischl1, H. Brandt1, T.de Cola1, R. De Gaudenzi2,*, E. Eberlein3, N. Girault4, E. Alberty4, S. Lipp3, R. Rinaldo2, B. Rislow5, J.A. Skard5, J. Tousch6 and G. Ulbricht3, "Adaptive coding and modulation for satellite broadband networks: From theory to practice" 27 MAR 2009

H. El Gamal and A. R. Hammons Jr., "On the design of algebraic space&ndash,time codes for block fading channels", IEEE Trans. Inform. Theory,

H. El Gamal and A. R. Hammons Jr., "On the design of algebraic space&ndash,time codes for block fading channels", ISIT',01, 2001

H. G. Myung and D. J. Goodman, Single Carrier FDMA: a New Air Interface for Long Term Evolution. Wiley, 2008.

H. Takara "Distance-adaptive super-wavelength routing in elastic optical path network (SLICE) with optical OFDM", Proc. ECOC 2010, pp.1 -3

HRACH, P. Simulation of the RF Transmission Channel for the DVB-H and DVB-SH. Brno: BUT, FEEC. Department of Radioelectronics, 2010.

I. W. Selesnick and C. S. Burrus, “Exchange algorithms that complement the Parks-McClellan algorithm for linear-phase FIR filter design,” IEEE Trans. on Circuits and Systems II, 44(2):137-142, February 1997. Suggested by R. A. Vargas, in private communication, 2001.

J. G. Proakis Digital Communications, 2000 :McGraw-Hill

J. Grimm, Transmit diversity code design for achieving full diversity on Rayleigh fading channels, 1998 :Purdue Univ.

J. Grimm, M. P. Fitz, and J. V. Krogmeier, "Further results in space&ndash,time coding for Rayleigh fading", Proc. 36th Allerton Conf. on Communications, Control and Computing, 1998

J. H. McClellan, T. W. Parks, and L. R. Rabiner, “A computer program for designing optimum FIR linear phase digital filters,” IEEE Trans. Audio Electroacoust. AU-21, 506-526, 1973.

J.-C. Guey, M. R. Bell, M. P. Fitz, and W.-Y. Kuo, "Signal design for transmitter diversity wireless communication systems over Rayleigh fading channels", IEEE Vehicular Technology Conf., pp.136 -140 1996

J.G. Proakis . (1989) , Digital Communications.

Ji-Won Jung, Jin-Hee Jeong, Min-Hyuk Kim, "Demapping Algorithm for Applying the Multilevel Modulation Scheme to LDPC Decoding Based on DVB-S2", Journal of The Korea Electromagnetic Engineering Society, vol. 17, no. 7. 2006.7.

L. B. Du and A. J. Lowery "Improved nonlinearity precompensation for long-haul high-data-rate transmission using coherent optical OFDM", Opt. Express, vol. 16, no. 24, pp.19920 -19925 2008

L. Giugno and M. Luise, "Adaptive pre- and post-compensation of nonlinear distortions for high-level data modulations," IEEE Trans. Wireless Commun., vol. 3, no. 5, pp. 1490-1495, 2004.

L. Hanzo, S. X. Ng, T. Keller, W. Webb, "Quadrature Amplitude Modulation", John Wiley & Sons Inc., 2003. ISBN: 9780470094686.

L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing, Prentice Hall, Englewood Cliffs, 1975.

M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, Simulation of Communication Systems: modeling, methodology and techniques, 2nd ed. Kluwer Academic, 2002.

M. C. Thomas, M. Y. Weidner, and S. H. Durrani, "Digital amplitude-phase keying with M-ary alphabets," IEEE Trans. Commun., vol. 22, no. 2, pp. 168-180, 1974.

M. C. Valenti and J. Sun, "The UMTS turbo code and efficient decoder implementation suitable for software-defined radios," Int. J. Wireless Information Networks, vol. 8, no. 4, pp. 203-215, 2002.

M. Varsamou, P. Savvopoulos, N. Papandreou and Th. Antonakopoulos, "From Matlab/Simulink Models to Prototype Implementation:A Communication Systems Development Environment" Copenhagen, Denmark, OPctober 2003

Melvin, Willian L. "A STAP Overview." IEEE Aerospace and Electronic Systems Magazine 19 (1) (January 2004): 19-35.

Morello A. and V. Mignone, "DVB-S2: The Second Generation Standard for Satellite Broad-band Services," Proc. IEEE, vol. 94, no. 1, pp. 210-227, Jan. 2006.

N. J. Fliege, Multirate Digital Signal Processing, John Wiley & Sons Ltd. Chichester, 1994.

Nix, A.R., Castle, R.J., McGeehan, J.P.: `The application of 16APSK to Mobile Fading Channels', Proc. IEE 6th Int. Conf. on Mobile Radio and Personal Communications, 1991, p. 233–240.

North, D. O. (1943). "An analysis of the factors which determine signal/noise discrimination in pulsed carrier systems". RCA Labs., Princeton, NJ, Rep. PTR-6C.

O. Hermann, and H. W. Sch¨ussler, “Design of nonrecursive digital filters with minimum phase,” Electron. Lett. 6,pp. 329-330, 1970.

P. P. Vaidyanathan and T. Q. Nguyen, “Eigenfilters: a new approach to least squares FIR filter design and applications including Nyquist filters,” IEEE Trans. Circuits Syst. CAS-34, pp. 11-23, 1987.

P. Savvopoulos, Synchronization in satellite communications devices : the multiple ring constellations case, Univ. of Patras, Patras, 2010-04-09

POLAK, L., KRATOCHVIL, T. DVB- SH Digital Television Transmission and its Simulation in MATLAB. In Proceedings of the 20th Conference Radioelektronika 2010. Brno (Czech Republic), 2011, pp. 75-78.

POLAK, L., KRATOCHVIL, T. Simulation of the DVB-H Channel Coding and Transmission in MATLAB. In Proceedings of the 21st Conference Radioelektronika 2011. Brno (Czech Republic), 2010, pp. 57-60.

R. D. J. V. N. A. R. Prasad OFDM for Wireless Multimedia Communications, 2000 :Artech House

R. De Gaudenzi, A. Guillen i Fabregas, and A. Martinez, "Turbo coded APSK modulations design for satellite broadband communications," Int. J. Satellite Commun. Networking, vol. 24, pp. 261-281, 2006. (Pubitemid 44085974)

R. Lyons, “Interpolated narrowband lowpass FIR filters,” IEEE Signal Proc. Mag., pp. 50-57, January, 2003.

R.De Gaudenzi, A.Guillen I Fabregas, A Martinez Vicente, "Turbo-coded .4PSK Modulations for Satellite Broadcasting and Multicasting-Part II: "End to End Performance", submitted to IEEE Trans. On Wireless Communications 2003

Raju, M.S Ramesh, A, Chockalingam,A, "LLR based BER analysis of orthogonal STBCs using QAM on Rayleigh fading channels, Personal, Indoor and Mobile Radio Communications, 2004 PIMRC 2004, 15

S. Baro, G. Bauch, and A. Hansman, "Improved codes for space&ndash,time trellis codes modulation", IEEE Commun. Lett., pp.20 -22 2000

S. Benedetto , E. Biglieri , V. Castellani . (1987) , Digital Transmission Theory.

S. J. Orfanidis, Introduction to Signal Processing, Prentice Hall, Upper Saddle River, 1996.

S. L. Jansen , I. Morita , T. C. W. Schenk , N. Takeda and H. Tanaka "Coherent optical 25.8-Gb/s OFDM transmission over 4160-km SSMF", J. Lightw. Technol., vol. 26, no. 1, pp.6 -15 2008

S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, 1983 :Prentice-Hall

S. M. Alamouti, "A simple transmitter diversity scheme for wireless communications", IEEE J. Select. Areas Commun., 1998

Saleh A. A. , "Frequency - independent and frequency - dependent nonlinear models of TWT amplifiers," IEEE Trans. Commun., vol. 29, no. 11, pp. 1715-1720, 1981.

T. Marzetta and B. Hochwald, "Capacity of a mobile multiple antenna communication link in Rayleigh flat fading", IEEE Trans. Inform. Theory, vol. 45, pp.139 -158 1999

T. S. Rappaport, Wireless Communications. Prentice Hall, 2001.

T. Saramaki, "A class of FIR Nyquist (Nth-band) filters with zero intersymbol interference," IEEE Trans. Circuits Syst. CAS-34, pp. 1182-1190, 1987.

T. Saramaki, Y. Neuvo, and S. K. Mitra, "Design of computationally efficient interpolated FIR filters," IEEE Trans. on Circuits and Systems, vol. 35, N0. 1,pp. 70-88, January 1988.

T. Saramaki. "Finite impulse response filter design," Chapter 4 in Handbook for Digital Signal Processing, S. K. Mitra and J. F. Kaiser eds. John Wiley and Sons, New York, 1993.

Turin, George L. "An introduction to matched filters." IRE Transactions on Information Theory 6 (3) (June 1960): 311- 329..

U. REimers, A. Morello, "DVB-S2, the second generation standard for satellite broadcasting and unicasting", International Journal on Satellite Communication Network, 2004; 22

V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space&ndash,time block codes from orthogonal designs", IEEE Trans. Inform. Theory, vol. 45, pp.1456 -1467 1999

V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space&ndash,time block coding for wireless communications: Performance results", IEEE J. Select. Areas Commun., vol. 17, pp.451 -460 1999

V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space&ndash,time codes for high data rate wireless communication: Performance criterion and code construction", IEEE Trans. Inform. Theory, vol. IT-44, pp.744 - 765 1998

W. Sung1, S. Kang1, P. Kim2, D.I. Chang2 and D. Shin3, "Performance analysis of APSK modulation for DVB-S2 transmission over nonlinear channels" 20 MAY 2009

W.-R. Peng , K. Takeshima , I. Morita , H. Takahashi and H. Tanaka "Scattered pilot channel tracking method for PDM-CO-OFDM transmissions using polar-based intra-symbol frequency-domain average", Proc. OFC 2011, pp.1 -3

Woodward P.M. Probability and Information Theory with Applications to Radar, Norwood, MA: Artech House, 1980.

X. Liu , Q. Yang , S. Chandrasekhar and W. Shieh "Transmission of 44-Gb/s coherent optical OFDM signal with trellis-coded 32-QAM subcarrier modulation", Proc. OFC 2010, pp.1 -3

X. Liu and F. Buchali "Improved nonlinear tolerance of 112-Gb/s PDM-OFDM in dispersion-uncompensated transmission with efficient channel estimation", Proc. ECOC 2008, pp.1 -2

Y. Neuvo, C.-Y. Dong, and S. K. Mitra, “Interpolated finite impulse response filters,” IEEE Trans. onAcoust. Speech and Signal Proc., vol. ASSP-32, pp 563-570, June 1984.

Y. R. Zheng and C. Xiao, "Simulation models with correct statistical properties for Rayleigh fading channels," IEEE Trans. Commun., vol. 51, no. 6, pp. 920-928, 2003.

7. Παράρτημα

TXRX – Η Βασική εφαρμογή

```
% Καθαρισμός όλων των μεταβλητών εκτός απο PSK και SNRG
%----------------- Switchpanel -------------------------------
%-------------------------------------------------------------------
Ps=1;           % Μέση Ενέργεια ανα σύμβολο = Ισχύς
coderate =2/3;       % Ρυθμός κωδικοποίησης 16APSK
coderate32=3/4;     % Ρυθμός κωδικοποίησης 32APSK
Radius3=0;     % Αρχικοποίηση μεταβλητών ακτίνας 32APSK
Radius2=0;     % Αρχικοποίηση μεταβλητών ακτίνας 16APSK
% Μεταβλητές φίλτρου
Nsym=6; overSamp = 8;
rollOff = .35;
gain=0.5;
if exist('Txlength','var')==0
Txlength=50;    % Μέγεθος δείγματος
end
if exist('PSK','var')==0
PSK=0;    % Επιλογή κωδικοποίησης 1.QPSK 2.8PSK 3.16APSK 4.32APSK
end
if exist('overrideplot','var')==0
   overrideplot=0;    % Επιλογή παραβίασης των απαγορεύσεων σχεδίασης γραφικών
παραστάσεων
end

awgnmode=0;
if exist('ebnoinput','var')==1     % Αναλόγως τις επιλογές μας ενεργοποιείται αυτόματα η
μέθοδος απόδοσης θορύβου
   awgnmode=1;                % Με SNR ή με Eb/No δίνοντας τιμές στις μεταβλητές
   typos='Eb/No';

else                          % snrinput kai ebnoinput αντίστοιχα
   typos='SNR';
   awgnmode=0;
   if exist('snrinput','var')==1    % Η μεταβλητή awgnmode είναι ο διακόπτης για τον τύπο
του καναλιού
   SNRG=snrinput;              % Για awgnmode=1 το κανάλι δουλεύει με ebno αλλιώς με
snr
   else
   SNRG=5; % Θόρυβος καναλιού
   end
end

% Διακόπτες ενεργοποίησης πομπού, καναλιού, δέκτη
% Αυτοί οι διακόπτες έχουν τοποθετηθεί για να απενεργοποιούμε όποιο
% απο τα τρία μέρη του αλγορίθμου (πομπός- κανάλι- Δέκτης) θέλουμε
% κατα την εκτέλεση
% Αν κάποιο απο τα μέρη αυτά απενεργοποιηθεί με διακόπτη, αυτόματα
% απενεργοποιείται και η δυνατοτήτα Γραφικής παράστασης των αποτελεσμάτων
% Υπάρχει ένας τρόπος να Εμφανιστούν οι γραφικες παραστάσεις
% με τη χρήση του διακόπτη παραβίασης overideplot
if exist('TxSW','var')==0
TxSW=1;
end
```

```
if exist('ChSW','var')==0
ChSW=1;
end
if exist('RxSW','var')==0
RxSW=1;
end

% ----------------------------------------------------------------
% ----------------------------------------------------------------

filtOrder=overSamp*Nsym;       % Μήκος φίλτρου
delay = filtOrder/(overSamp*2); % Καθυστέρηση φίλτρου

while (PSK< 1) || (PSK>4)
 disp('Please Select INPUT MODE : ')
 disp('1.    QPSK ');
 disp('2.    8APSK ');
 disp('3.    16APSK ');
 disp('4.    32APSK ');
 disp('selection :  ');
 text=' ';
 PSK= input(text);
end


%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxx                    xxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxx      ΠΟΜΠΟΣ     xxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxx                    xxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
if TxSW==1
%========== Δημιουργία τυχαίου σήματος =============%
startsample=samplegen(PSK,Txlength);
% ---------------- Συνάρτηση Αστερισμού --------------------
Tx=zeros(Txlength,1);
Txnoisy=zeros(Txlength,1);
switch PSK
   case 1
Tx=mapQPSK(startsample,Ps);
output=['Creation of sample and coding in QPSK successful',' Symbol Length:
',num2str(Txlength)];
   case 2
Tx=map8PSK(startsample,Ps);
output=['Creation of sample and coding in 8PSK successful',' Symbol Length:
',num2str(Txlength)];
   case 3
% Δημιουργία ακτίνων για τον αστερισμό
%16APSK απο Ισχύ σήματος και coderate
[R1,R2]=Radius16APSK(Ps,coderate,Radius2);
Tx=map16APSK(startsample,R1,R2);
output=['Creation of sample and coding in 16APSK successful',' Symbol Length:
',num2str(Txlength)];
   case 4
% Δημιουργία ακτίνων για τον αστερισμό
%16APSK απο Ισχύ σήματος και coderate
[R1,R2,R3]=Radius32APSK(Ps,coderate32,Radius3);
Tx=map32APSK(startsample,R1,R2,R3);
```

```
output=['Creation of sample and coding in 32APSK successful ',' Symbol Length:
',num2str(Txlength)];
end
disp(output);
%============Φιλτράρισμα δεδομένων=========%
%Πρώτα υπερδειγματοληπτώ
Txup=upsample(Tx,overSamp);
% Έπειτα φιλτράρω
[Txfiltered,HT]=transmitfilter(Txup,gain,delay,overSamp,Nsym,rollOff);
%====================================%
disp('Transmition was filtered sucesfully')
else
   ploted=0;
end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx



%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxx                              xxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxx   ΚΑΝΑΛΙ ΜΕ ΘΟΡΥΒΟ GAUSS    xxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxx                              xxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
if ChSW==1

if awgnmode==1
   switch PSK
   case 1
   M=4;
   Rc=1;
   case 2
   M=8; %Number of Constellation points M=2^k for 8-PSK k=3
   Rc=1;   %Rc = code rate for a coded system. Since no coding is used Rc=1
     case 3
      M=16;
      Rc=1;
   case 4
      M=32;
      Rc=1;
   end
   Rm=log2(M);   %Rm=log2(M) for 8-PSK M=8


   EBNO=ebnoinput;
   disp(['EB/no : ',num2str(EBNO)]);
   EbNo = 10.^(ebnoinput/10); %Converting Eb/N0 dB value to linear scale
   Sigma = sqrt(2)*sqrt(1./(2*Rm*Rc*EbNo)); %Standard deviation for AWGN Noise

    noise = Sigma*(randn(1,Txlength*overSamp)+randn(1,Txlength*overSamp)*1j)';
   Rxnoisy = Txfiltered + noise;
   Txnoisy=Tx+downsample(noise,overSamp);
else
Rxnoisy=awgn(Txfiltered,SNRG,'measured');
```

```
Txnoisy=awgn(Tx,SNRG,'measured');

end
%Υπολογίζω το Sound to Noise Ratio (SNR) απο τον τύπο 10*log(ps/pn)
% βρίσκω την ισύ του σήματος Ps=Σ(|χ|^2)/n
ps=mean(abs(Txfiltered).^2);
% βρίσκω την ισύ του θορύβου Pn=Σ(|noise|^2)/n
noise=Rxnoisy-Txfiltered;
pn=mean(abs(noise).^2);
snr=10*log10(ps/pn);
disp('Signal noised through channel succesfully')

else
   disp('Signal skipped the channel')
   Rxnoisy=Txfiltered;
   Txnoisy=Tx;
   ploted=0;
end
%radius adjustment
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx



%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxx                          xxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxx          ΔΕΚΤΗΣ            xxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxx                          xxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%radad=sqrt(mean(abs(Tx)).^2/mean(abs(Rxnoisy).^2));
%Rxnoisy=radad*Rxnoisy;
if RxSW==1
% Εφαρμόζω το φίλτρο του δέκτη
[Rxfiltered,HR]=transmitfilter(Rxnoisy,0.5,delay,overSamp,Nsym,rollOff);
% ------- Υποδειγματοληψία για να πάρω τις αρχικές τιμές----------
RxDown=downsample(Rxfiltered,overSamp);
radad=sqrt(mean(abs(Tx)).^2/mean(abs(RxDown).^2));
RxDown=radad*RxDown;

% Το snr2 υπολογίζεται απο το θορυβημένο σήμα και το φιλτραρισμένο απο το
% φίλτρο του δέκτη ενώ το αρχικό snr απο το θορυβημένο και το φιλτραρισμένο
% απο το φίλτρο του πομπού
switch PSK
   case 1
[Rx,BER]=receiverQPSK(RxDown,startsample,Ps);
   case 2
[Rx,BER]=receiver8PSK(RxDown,startsample,Ps);
   case 3
[Rx,BER]=receiver16APSK(RxDown,startsample,R1,R2);
   case 4
[Rx,BER]=receiver32APSK(RxDown,startsample,R1,R2,R3);
end
%---------------- Ανίχνευση σφάλματος--------------------
% Εναλλακτικός τρόπος είναι να συγκρίνω το θορυβημένο
```

```
% κατευθείαν με το φιλτραρισμένο μετά το κανάλι
ps2=mean(abs(Rxfiltered).^2);
pn2=mean(abs(Rxnoisy-Rxfiltered).^2);
snr2=10*log10(ps2/pn2);
disp('Signal Reception was succesfull')
else
   ploted=0;
end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


if overrideplot==1
   % Αυτή είναι η μεταβλητή που υπερισχύει της απαγόρευσης
   % προβολής των γραφικών παραστάσεων.. σε περίπτωση που χρειαστεί να
   % αναπαραστήσουμε ενώ δεν είναι ενεργός ο πομπός
  ploted=1;
end

% Απενεργοποίηση Γραφικών παραστάσεων
if Txlength>400
   ploted=0;
end

% Γραφική παράσταση του σήματος μετά το κανάλι πριν
% και μετά το φίλτρο SRRC
switch PSK
   case 1
      title1='QPSK ';
   case 2
      title1='8PSK ';
   case 3
      title1='16APSK ';
   case 4
      title1='32APSK ';
end

if exist('ploted','var')==0 || ploted==1
   A=zeros(3,1);
   A(1)=max(max(abs(real(Txfiltered))),max(abs(real(Txup))));
   A(2)=max(max(abs(real(Rxnoisy))),max(abs(real(Rxfiltered))));
   A(3)=max(abs(real(Rx)));
   Amp2=max(A);
Txuplength=length(Txfiltered);
figure;
sh(1)=subplot(3,1,1);
plot(real(Txfiltered),'rd-.');
hold on;
stem(real(Txup),'b');
xlabel('Symbols');
ylabel('Amplitude');
legend('Transmitted Filtered Signal','Original Tx Signal');
axis([0 Txuplength -Amp2 Amp2]);
grid on;
```

```
set(gcf,'Color','w')
% Γραφική παράσταση του αρχικού σήματος και
% του σήματος εκπομπής (μετά το φίλτρο SRRC)
sh(2)=subplot(3,1,2);
plot(real(Rxnoisy),'ko-');
hold on;
plot(real(Rxfiltered),'r-');
stem(upsample(downsample(real(Rxfiltered),overSamp),overSamp),'r')
xlabel('Symbols (Nsym)');
ylabel('Aplitude');
grid on;
axis([0 Txuplength -Amp2 Amp2]);
title(title1);
legend('Noised Signal ','Received Filtered Signal','Sampling Point');
% Γραφική παράσταση του τελικού σήματος στη μονάδα του χρόνου
sh(3)=subplot(3,1,3);
hold on;
stem(upsample(real(Rx),overSamp),'b');
xlabel('Symbols (Nsym)');
ylabel('Amplitude');
legend('Transmit Signal');
axis([0 Txuplength -Amp2 Amp2]);
grid on;
set(gcf,'Color','w')
linkaxes(sh,'xy');
legend('Final Rx Signal');

figure;
title('Transmit and Receive filter graphs')
subplot(1,2,1);
stem(HT,'b');
hold on;

xlabel('Number of Samples (Nsym)');
ylabel('Amplitude');
grid on;
legend('SRRC Transmit Filter Impulse Responce');
plot(HT,'b');
hold off;

subplot(1,2,2)
stem(HR,'r-');hold on;
plot(HR,'r-');

xlabel('Number of Samples (Nsym)');
ylabel('Amplitude');
legend('SRRC Receive Filter Impulse Rsponce')
grid on;
hold off;


   figure;
   axes();hold on;
   plot(real(Txnoisy),imag(Txnoisy),'r*');
   plot(real(Tx),imag(Tx),'ko','MarkerFaceColor','g','MarkerSize',7);
   if awgnmode==0
```

```
  titlos=['GREEN : Constellation of ideal ',title1,'',10,'RED : Noised Singal ',10,' Sound To
Noise Ratio (SNR) ',num2str(SNRG),' dB '];
    else
  titlos=['GREEN : Constellation of ideal ',title1,'',10,'RED : Noised Singal ',10,' Energy per
Bit/ Noise Spectral Density (Eb/No) ',num2str(EBNO),' dB '];
    end
  title(titlos);
minx=-abs(min(min(real(Txnoisy)),min(real(Tx))));
miny=-abs(min(min(imag(Txnoisy)),min(imag(Tx))));
maxx=abs(max(max(real(Txnoisy)),max(max(Tx))));
maxy=abs(max(max(imag(Txnoisy)),max(imag(Tx))));
xax=max(-minx,maxx);
yax=max(-miny,maxy);
ax=max(xax,yax);

% drawcircles
t = 0:0.1:2*pi;

switch PSK
    case 1
x = Ps*cos(t);
y = Ps*sin(t);
plot(x,y,'k-');
    case 2
x = Ps*cos(t);
y = Ps*sin(t);
plot(x,y,'k-');
    case 3
x = R1*cos(t);
y = R1*sin(t);
plot(x,y,'k-');
x = R2*cos(t);
y = R2*sin(t);
plot(x,y,'k-');
    case 4
x = R1*cos(t);
y = R1*sin(t);
plot(x,y,'k-');
x = R2*cos(t);
y = R2*sin(t);
plot(x,y,'k-');
x = R3*cos(t);
y = R3*sin(t);
plot(x,y,'k-');

end
plot([-ax ax],[0 0],'k-');
plot([0 0],[-ax ax],'k-');

  figure;
    axes();hold on;
    plot(real(RxDown),imag(RxDown),'b*');
    plot(real(Tx),imag(Tx),'ko','MarkerFaceColor','g','MarkerSize',7);
    if awgnmode==0
  titlos=['GREEN : Constellation of ideal ',title1,'',10,'BLUE : Downsampled Received Signal
',10,' Sound To Noise Ratio (SNR) ',num2str(SNRG),' dB ',' Bit Error Rate
',num2str(BER*100),'%'];
```

```
   else
  titlos=['GREEN : Constellation of ideal ',title1,'',10,'BLUE : Downsampled Received Signal
',10,' Energy per Bit/ Noise Spectral Density (Eb/No) ',num2str(EBNO),' dB ',' Bit Error Rate
',num2str(BER*100),'%'];
   end
  title(titlos);
minx=-abs(min(min(real(RxDown)),min(real(Tx))));
miny=-abs(min(min(imag(RxDown)),min(imag(Tx))));
maxx=abs(max(max(real(RxDown)),max(max(Tx))));
maxy=abs(max(max(imag(RxDown)),max(imag(Tx))));
xax=max(-minx,maxx);
yax=max(-miny,maxy);
ax=max(xax,yax);

% drawcircles
t = 0:0.1:2*pi;

switch PSK
   case 1
x = Ps*cos(t);
y = Ps*sin(t);
plot(x,y,'k-');
   case 2
x = Ps*cos(t);
y = Ps*sin(t);
plot(x,y,'k-');
   case 3
x = R1*cos(t);
y = R1*sin(t);
plot(x,y,'k-');
x = R2*cos(t);
y = R2*sin(t);
plot(x,y,'k-');
   case 4
x = R1*cos(t);
y = R1*sin(t);
plot(x,y,'k-');
x = R2*cos(t);
y = R2*sin(t);
plot(x,y,'k-');
x = R3*cos(t);
y = R3*sin(t);
plot(x,y,'k-');

end
plot([-ax ax],[0 0],'k-');
plot([0 0],[-ax ax],'k-');
end
clear overrideplot ploted TxSW ChSW RxSW ebnoinput snrinput
```

Κωδικοποίηση - Αποκωδικοποίηση QPSK

```
function Transmitter=mapQPSK(sample,PowerPerSymbol)
txlength=length(sample);
% ---------------- MAPPING FUNCTION --------------------
% -------------------------QPSK-----------------------------
Transmitter=zeros(txlength,1);
%         step=20*round(txlength/100);

     for i=1:txlength

%           if mod(i,step) ==0
%           tell=10*i/step;
%              disp(tell);
%           end

         bl=sample(i,:);
         if bl==[0 0]
         Transmitter(i)=PowerPerSymbol*(cos(pi/4)+sin(pi/4)*1j);
         elseif bl==[1 0]
         Transmitter(i)=PowerPerSymbol *(cos(3*pi/4)+sin(3*pi/4)*1j);
         elseif bl==[1 1]
         Transmitter(i)=PowerPerSymbol*(cos(5*pi/4)+sin(5*pi/4)*1j);
         elseif bl==[0 1]
         Transmitter(i)=PowerPerSymbol*(cos(7*pi/4)+sin(7*pi/4)*1j);
         end
     end
    function [ReceiveSignal,ber]=receiverQPSK(RxDown,startsample,PowerPerSample)

%  Αρχικοποίηση τελικού σήματος και τιμών
% -------Δημιουργία τελικού σήματος απο το σήμα λήψης----------------
%  Αρχικοποίηση τελικού σήματος και τιμών
txlength=length(RxDown);
ReceiveSignal=zeros(txlength,1);

r1=zeros(txlength,2);
ber=0;
       for i=1:txlength
          if real(RxDown(i))>0 && imag(RxDown(i))>0
          ReceiveSignal(i)=PowerPerSample*(cos(pi/4)+sin(pi/4)*1j);
          r1(i,:)=[0 0];
          elseif real(RxDown(i))<0 && imag(RxDown(i))>0
          ReceiveSignal(i)=PowerPerSample*(cos(3*pi/4)+sin(3*pi/4)*1j);
          r1(i,:)=[1 0];
         elseif real(RxDown(i))<0 && imag(RxDown(i))<0
          ReceiveSignal(i)=PowerPerSample*(cos(5*pi/4)+sin(5*pi/4)*1j);
          r1(i,:)=[1 1];
          elseif real(RxDown(i))>0 && imag(RxDown(i))<0
          ReceiveSignal(i)=PowerPerSample*(cos(7*pi/4)+sin(7*pi/4)*1j);
          r1(i,:)=[0 1];
          end

       end

ber=sum(sum(xor(r1,startsample)));
```

```
ber=ber/(2*txlength);
```

Κωδικοποίηση - Αποκωδικοποίηση 8PSK

```
function Transmitter=map8PSK(sample,PowerPerSymbol)
Txlength=length(sample);
% ---------------- MAPPING FUNCTION --------------------
% -------------------------8PSK-----------------------------
Transmitter=zeros(Txlength,1);


        for i =1:Txlength
           bl=sample(i,:);
%            if mod(i,step) ==0
%             tell=10*i/step;
%             disp(tell);
%           end

              if        bl==[0 0 0];
              Transmitter(i) = PowerPerSymbol*(cos(1*pi/4)+sin(1*pi/4)*1j);
              elseif     bl==[0 0 1];
             Transmitter(i) = PowerPerSymbol*(cos(0*pi/4)+sin(0*pi/4)*1j);
              elseif     bl==[0 1 0];
             Transmitter(i) = PowerPerSymbol*(cos(4*pi/4)+sin(4*pi/4)*1j);
              elseif    bl==[0 1 1];
             Transmitter(i) = PowerPerSymbol*(cos(5*pi/4)+sin(5*pi/4)*1j);
              elseif    bl==[1 0 0];
             Transmitter(i) = PowerPerSymbol*(cos(2*pi/4)+sin(2*pi/4)*1j);
              elseif    bl==[1 0 1];
             Transmitter(i) = PowerPerSymbol*(cos(7*pi/4)+sin(7*pi/4)*1j);
              elseif    bl==[1 1 0];
             Transmitter(i) = PowerPerSymbol*(cos(3*pi/4)+sin(3*pi/4)*1j);
              elseif    bl==[1 1 1];
             Transmitter(i) = PowerPerSymbol*(cos(6*pi/4)+sin(6*pi/4)*1j);
               end
       end
        % ---------------------------------------------------
        function [ReceiveSignal,ber]=receiver8PSK(RxDown,startsample,PowerPerSample)
txlength=length(RxDown);
%  Αρχικοποίηση τελικού σήματος και τιμών
% -------Δημιουργία τελικού σήματος απο το σήμα λήψης----------------
%  Αρχικοποίηση τελικού σήματος και τιμών
ReceiveSignal=zeros(txlength,1);


         % Αρχικοποίηση τιμών

          D=ones(1,3);
          Rx1=RxDown;
          ber=0;
          r1=zeros(txlength,3);
```

```
                % Αποφασίζω σε ποιό τεταρτημόριο βρίσκεται το κάθε στοιχείο
                % και ψάχνω σε ποιο απο τα 2 στοιχεία του αρχικού σήματος είναι
                % αλγεβρικά πιο κοντά

                for i=1:length(Rx1)

        %         if mod(i,step) ==0
        %         tell=10*(i/step+5);
        %           disp(tell);
        %         end

                if real(Rx1(i))>=0 && imag(Rx1(i))>=0  % 1ο Τεταρτημόριο
                  D(1) = abs(Rx1(i)-PowerPerSample*(1+0*1j));                % Σημείο (1,0)
                  D(2)    =    abs(Rx1(i)-PowerPerSample*(cos(pi/4)+cos(pi/4)*1j));    %    Σημείο
(cos(π/4),cos(π/4))
                  D(3) = abs(Rx1(i)-PowerPerSample*(0+1*1j));                % Σημείο (ο,1)
                 minD=min(D);
                  switch minD
                     case D(1)
                        ReceiveSignal(i)= PowerPerSample*1;
                        r1(i,:)=[0 0 1];
                     case D(2)
                        ReceiveSignal(i)=PowerPerSample*(cos(pi/4)+cos(pi/4)*1j);
                        r1(i,:)=[0 0 0];
                     case D(3)
                        ReceiveSignal(i)=PowerPerSample*1j;
                        r1(i,:)=[1 0 0];
                  end
                elseif real(Rx1(i))<0 && imag(Rx1(i))>=0  % 2ο Τεταρτημόριο
                  D(1) = abs(Rx1(i)-PowerPerSample*(0+1*1j));                % Σημείο (0,1)
                  D(2)    =    abs(Rx1(i)-PowerPerSample*(-cos(pi/4)+cos(pi/4)*1j));    %    Σημείο    (-
cos(π/4),cos(π/4))
                  D(3) = abs(Rx1(i)-PowerPerSample*(-1+0*1j));                % Σημείο (-1,0)
                 minD=min(D);
                  switch minD
                     case D(1)
                        ReceiveSignal(i)= PowerPerSample*1j;
                        r1(i,:)=[1 0 0];
                     case D(2)
                        ReceiveSignal(i)=PowerPerSample*(-cos(pi/4)+cos(pi/4)*1j);
                        r1(i,:)=[1 1 0];
                     case D(3)
                        ReceiveSignal(i)=PowerPerSample*(-1);
                        r1(i,:)=[0 1 0];
                  end
                elseif real(Rx1(i))<0 && imag(Rx1(i))<0  % 3ο Τεταρτημόριο
                  D(1) = abs(Rx1(i)-PowerPerSample*(-1+0*1j));                % Σημείο (-1,0)
                  D(2) = abs(Rx1(i)-PowerPerSample*(-cos(pi/4)-cos(pi/4)*1j));  % Σημείο (-cos(π/4),-
cos(π/4))
                  D(3) = abs(Rx1(i)-PowerPerSample*(0-1j));                % Σημείο (ο,-1)
                 minD=min(D);
                  switch minD
                     case D(1)
                        ReceiveSignal(i)= PowerPerSample*(-1);
                        r1(i,:)=[0 1 0];
                     case D(2)
                        ReceiveSignal(i)=PowerPerSample*(-cos(pi/4)-cos(pi/4)*1j);
```

```
                    r1(i,:)=[0 1 1];
                case D(3)
                    ReceiveSignal(i)=PowerPerSample*(-1j);
                    r1(i,:)=[1 1 1];
            end
        elseif real(Rx1(i))>=0 && imag(Rx1(i))<0  % 4ο Τεταρτημόριο
            D(1) = abs(Rx1(i)-PowerPerSample*(0-1*1j));                % Σημείο (0,-1)
            D(2) =  abs(Rx1(i)-PowerPerSample*(cos(pi/4)-cos(pi/4)*1j));  % Σημείο  (cos(π/4),-
cos(π/4))
            D(3) = abs(Rx1(i)-PowerPerSample*(1+0*1j));                % Σημείο (1,0)
           minD=min(D);
            switch minD
                case D(1)
                    ReceiveSignal(i)= PowerPerSample*(-1j);
                    r1(i,:)=[1 1 1];
                case D(2)
                    ReceiveSignal(i)=PowerPerSample*(cos(pi/4)-cos(pi/4)*1j);
                    r1(i,:)=[1 0 1];
                case D(3)
                    ReceiveSignal(i)=PowerPerSample*1;
                    r1(i,:)=[0 0 1];
            end
        end

%           b1=startsample(i,:);
%           b2=r1(i,:);
%              ber=ber+sum(xor(b1,b2));
        end

        ber=sum(sum(xor(r1,startsample)));
        ber=ber/(3*txlength);
```

Κωδικοποίηση - Αποκωδικοποίηση 16APSK

```
function Transmitter=map16APSK(sample,Radius1,Radius2)
% ---------- MAPPING FUNCTION ----------
% ------------------------------------------------------------
Txlength=length(sample);
Transmitter=zeros(Txlength,1);
% step=20*round(txlength/100);

for i=1:Txlength

%           if mod(i,step) ==0
%           tell=10*i/step;
%              disp(tell);
%           end

   bl=sample(i,:);
if bl == [0 0 0 0];
Transmitter(i)=  Radius2 * exp(1j*pi/12*3);
elseif bl == [0 0 0 1];
Transmitter(i)= Radius2 * exp(1j*pi/12*21);
elseif bl == [0 0 1 0];
Transmitter(i) = Radius2 * exp(1j*pi/12*9);
```

```
elseif bl == [0 0 1 1];
Transmitter(i) =  Radius2 * exp(1j*pi/12*15);
elseif bl == [0 1 0 0];
Transmitter(i) = Radius2 * exp(1j*pi/12);
elseif bl == [0 1 0 1];
Transmitter(i) = Radius2 * exp(1j*pi/12*23);
elseif bl == [0 1 1 0];
Transmitter(i) =  Radius2 * exp(1j*pi/12*11);
elseif bl == [0 1 1 1];
Transmitter(i) = Radius2 * exp(1j*pi/12*13);
elseif bl == [1 0 0 0];
Transmitter(i) = Radius2*exp(1j*pi/12*5);
elseif bl == [1 0 0 1];
Transmitter(i) = Radius2 * exp(1j*pi/12*19);
elseif bl == [1 0 1 0];
Transmitter(i) = Radius2 * exp(1j*pi/12*7);
elseif bl == [1 0 1 1];
Transmitter(i) = Radius2 * exp(1j*pi/12*17);
elseif bl == [1 1 0 0];
Transmitter(i) = Radius1 * exp(1j*pi/4 );
elseif bl == [1 1 1 0];
Transmitter(i) = Radius1 * exp(1j*pi/4 *3);
elseif bl == [1 1 0 1];
Transmitter(i) = Radius1 * exp(1j*pi/4 *7);
elseif bl == [1 1 1 1];
Transmitter(i) = Radius1 * exp(1j*pi/4 *5);
end
end

clearvars txlength bl gamma

function [ReceiveSignal,ber]=receiver16APSK(RxDown,startsample,R1,R2)
txlength=length(RxDown);
%               Αρχικοποίηση τελικού σήματος και τιμών
% -------Δημιουργία τελικού σήματος απο το σήμα λήψης----------------
% ------------------------------------------------------------------------------
% ------------------------------------------------------------------------------
%  Αρχικοποίηση τελικού σήματος και τιμών
ReceiveSignal=zeros(txlength,1);

        % Αρχικοποίηση τιμών
        D=ones(1,4);
        r1=zeros(txlength,4);
        ber=0;
        Rx1=RxDown;

        % Αποφασίζω σε ποιό τεταρτημόριο βρίσκεται το κάθε στοιχείο
        % και ψάχνω σε ποιο απο τα 4 στοιχεία του αρχικού σήματος είναι
        % αλγεβρικά πιο κοντά

        for i=1:txlength
%           if mod(i,step) ==0
%           tell=10*(i/step+5);
%              disp(tell);
%           end
%
       if real(Rx1(i))>=0 && imag(Rx1(i))>=0  % 1ο Τεταρτημόριο
```

```
                D(1) = abs(Rx1(i)-R2*(cos(pi/12)+sin(pi/12)*1j));%  (R2cos(π/12),R2sin(π/12))
                D(2) = abs(Rx1(i)-R2*(cos(pi/4)+sin(pi/4)*1j));   %  (R2cos(π/4),R2sin(π/4))
                D(3)          =          abs(Rx1(i)-R2*(cos(5*pi/12)+sin(5*pi/12)*1j));          %
(R2cos(5π/12),R2sin(5π/12))
                D(4) = abs(Rx1(i)-R1*(cos(pi/4)+sin(pi/4)*1j));  % (r1cos(π/4),r1sin(π/4))

                minD=min(D);

                switch minD
                   case D(1)
                      ReceiveSignal(i)= R2*(cos(pi/12)+sin(pi/12)*1j);
                      r1(i,:)= [0 1 0 0];
                   case D(2)
                      ReceiveSignal(i)=R2*(cos(pi/4)+sin(pi/4)*1j);
                      r1(i,:) = [0 0 0 0];
                   case D(3)
                      ReceiveSignal(i)=R2*(cos(5*pi/12)+sin(5*pi/12)*1j);
                      r1(i,:)= [1 0 0 0];
                   case D(4)
                      ReceiveSignal(i)=R1*(cos(pi/4)+sin(pi/4)*1j);
                      r1(i,:)= [1 1 0 0];
                end

             elseif real(Rx1(i))<0 && imag(Rx1(i))>=0  % 2ο Τεταρτημόριο

                D(1) = abs(Rx1(i)-R2*(-cos(pi/12)+sin(pi/12)*1j));%       (-R2cos(π/12),R2sin(π/12))
                D(2) = abs(Rx1(i)-R2*(-cos(pi/4)+sin(pi/4)*1j));   %       (-R2cos(π/4),R2sin(π/4))
                D(3)      =      abs(Rx1(i)-R2*(-cos(5*pi/12)+sin(5*pi/12)*1j));      %                (-
R2cos(5π/12),R2sin(5π/12))
                D(4) = abs(Rx1(i)-R1*(-cos(pi/4)+sin(pi/4)*1j));  %         (-r1cos(π/4),r1sin(π/4))

                minD=min(D);

                switch minD
                   case D(1)
                      ReceiveSignal(i)= R2*(-cos(pi/12)+sin(pi/12)*1j);
                      r1(i,:)= [0 1 1 0];
                   case D(2)
                      ReceiveSignal(i)=R2*(-cos(pi/4)+sin(pi/4)*1j);
                      r1(i,:)= [0 0 1 0];
                   case D(3)
                      ReceiveSignal(i)=R2*(-cos(5*pi/12)+sin(5*pi/12)*1j);
                      r1(i,:)= [1 0 1 0];
                   case D(4)
                      ReceiveSignal(i)=R1*(-cos(pi/4)+sin(pi/4)*1j);
                      r1(i,:)= [1 1 1 0];
                end

             elseif real(Rx1(i))<0 && imag(Rx1(i))<0  % 3ο Τεταρτημόριο

                D(1) = abs(Rx1(i)-R2*(-cos(pi/12)-sin(pi/12)*1j));%  (-R2cos(π/12),-R2sin(π/12))
                D(2) = abs(Rx1(i)-R2*(-cos(pi/4)-sin(pi/4)*1j));   %  (-R2cos(π/4),-R2sin(π/4))
                D(3)    =    abs(Rx1(i)-R2*(-cos(5*pi/12)-sin(5*pi/12)*1j));    %        (-R2cos(5π/12),-
R2sin(5π/12))
                D(4) = abs(Rx1(i)-R1*(-cos(pi/4)-sin(pi/4)*1j));  % (-r1cos(π/4),-r1sin(π/4))
```

```
                minD=min(D);

                switch minD
                   case D(1)
                      ReceiveSignal(i)= R2*(-cos(pi/12)-sin(pi/12)*1j);
                      r1(i,:)= [0 1 1 1];
                   case D(2)
                      ReceiveSignal(i)=R2*(-cos(pi/4)-sin(pi/4)*1j);
                      r1(i,:)= [0 0 1 1];
                   case D(3)
                      ReceiveSignal(i)=R2*(-cos(5*pi/12)-sin(5*pi/12)*1j);
                      r1(i,:)= [1 0 1 1];
                   case D(4)
                      ReceiveSignal(i)=R1*(-cos(pi/4)-sin(pi/4)*1j);
                      r1(i,:)= [1 1 1 1];
                end
             elseif real(Rx1(i))>=0 && imag(Rx1(i))<0  % 4ο Τεταρτημόριο

                D(1) = abs(Rx1(i)-R2*(cos(pi/12)-sin(pi/12)*1j));%  (R2cos(π/12),-R2sin(π/12))
                D(2) = abs(Rx1(i)-R2*(cos(pi/4)-sin(pi/4)*1j));   %  (R2cos(π/4),-R2sin(π/4))
                D(3)    =    abs(Rx1(i)-R2*(cos(5*pi/12)-sin(5*pi/12)*1j));    %         (R2cos(5π/12),-
R2sin(5π/12))
                D(4) = abs(Rx1(i)-R1*(cos(pi/4)-sin(pi/4)*1j));  % (r1cos(π/4),-r1sin(π/4))

                minD=min(D);

                switch minD
                   case D(1)
                      ReceiveSignal(i)= R2*(cos(pi/12)-sin(pi/12)*1j);
                      r1(i,:)= [0 1 0 1];
                   case D(2)
                      ReceiveSignal(i)=R2*(cos(pi/4)-sin(pi/4)*1j);
                      r1(i,:) = [0 0 0 1];
                   case D(3)
                      ReceiveSignal(i)=R2*(cos(5*pi/12)-sin(5*pi/12)*1j);
                      r1(i,:)= [1 0 0 1];
                   case D(4)
                      ReceiveSignal(i)=R1*(cos(pi/4)-sin(pi/4)*1j);
                      r1(i,:)= [1 1 0 1];
                end
             end

%            b1=startsample(i,:);
%            b2=r1(i,:);
%
%             ber=ber+sum(xor(b1,b2));
%
          end

ber=sum(sum(xor(r1,startsample)));
      ber=ber/(4*txlength);
```

Κωδικοποίηση - Αποκωδικοποίηση 32PSK

```
function Transmitter=map32APSK(sample,Radius1,Radius2,Radius3)
```

```
% 32APSK mapping (according to ETSI EN 302 583)
   Txlength=length(sample);
      % ---------------------------------------------------------------------
      % |  Πίνακας αντιστοίχισης coderate - γ1=R2/R1 γ2=R3/R1  |
      % ---------------------------------------------------------------------
Transmitter=zeros(Txlength,1);

for i=1:Txlength
   bl=sample(i,:);

%          if mod(i,step) ==0
%          tell=10*i/step;
%             disp(tell);
%          end

            if bl == [1 0 0 0 1];
               Transmitter(i) = Radius1 * exp(1i*pi/4 ); %
            elseif bl == [1 0 1 0 1];
               Transmitter(i) = Radius1 * exp(1j*pi/4 *3); %
            elseif bl == [1 0 1 1 1];
               Transmitter(i) = Radius1 * exp(1i*pi/4 *5); %
            elseif bl == [1 0 0 1 1];
               Transmitter(i) = Radius1 * exp(1j*pi/4 *7); %
            elseif bl == [1 0 0 0 0];
               Transmitter(i) = Radius2 * exp(1j*pi/12); %
            elseif bl == [0 0 0 0 0];
               Transmitter(i) = Radius2 * exp(1i*pi/12*3); %
            elseif bl == [0 0 0 0 1];
               Transmitter(i) = Radius2 * exp(1j*pi/12*5); %
            elseif bl == [0 0 1 0 1];
               Transmitter(i) = Radius2 * exp(1j*pi/12*7); %
            elseif bl == [0 0 1 0 0];
               Transmitter(i) = Radius2 * exp(1i*pi/12*9); %
            elseif bl == [1 0 1 0 0];
               Transmitter(i) = Radius2 * exp(1j*pi/12*11); %
            elseif bl == [1 0 1 1 0];
               Transmitter(i) = Radius2 * exp(1j*pi/12*13); %
            elseif bl == [0 0 1 1 0];
               Transmitter(i) = Radius2 * exp(1j*pi/12*15); %
            elseif bl == [0 0 1 1 1];
               Transmitter(i) = Radius2 * exp(1j*pi/12*17); %
            elseif bl == [0 0 0 1 1];
               Transmitter(i) = Radius2 * exp(1j*pi/12*19); %
            elseif bl == [0 0 0 1 0];
               Transmitter(i) = Radius2 * exp(1j*pi/12*21); %
            elseif bl == [1 0 0 1 0];
               Transmitter(i) = Radius2 * exp(1j*pi/12*23); %
            elseif bl == [1 1 0 0 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *0); %
            elseif bl == [0 1 0 0 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *1); %
            elseif bl == [1 1 0 0 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *2); %
            elseif bl == [0 1 0 0 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *3); %
            elseif bl == [0 1 1 0 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *4); %
```

```
            elseif bl == [1 1 1 0 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *5); %
            elseif bl == [0 1 1 0 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *6); %
            elseif bl == [1 1 1 0 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *7); %
            elseif bl == [1 1 1 1 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *8); %
            elseif bl == [0 1 1 1 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *9); %
            elseif bl == [1 1 1 1 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *10); %
            elseif bl == [0 1 1 1 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *11); %
            elseif bl == [0 1 0 1 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *12); %
            elseif bl == [1 1 0 1 1];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *13); %
            elseif bl == [0 1 0 1 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *14); %
            elseif bl == [1 1 0 1 0];
               Transmitter(i) = Radius3 * exp(1j*pi/8 *15); %
            end

      end
      function [ReceiveSignal,ber]=receiver32APSK(RxDown,startsample,R1,R2,R3)
      txlength=length(RxDown);
      % -------Δημιουργία τελικού σήματος απο το σήμα λήψης-----------------
      % ------------------------------------------------------------------------
      % ------------------------------------------------------------------------
            %  Αρχικοποίηση τελικού σήματος και τιμών
            ReceiveSignal=zeros(txlength,1); % Τελικό σήμα
            D=ones(1,9);  %  Αποστάσεις απο των σημείων απο τις θεωρητικές τιμές
            ber=0;
            Rx1=RxDown;

          % Αποφασίζω σε ποιό τεταρτημόριο βρίσκεται το κάθε στοιχείο
          % και ψάχνω σε ποιο απο τα 4 στοιχεία του αρχικού σήματος είναι
          % αλγεβρικά πιο κοντά
            r1=zeros(txlength,5);
          for i=1:length(Rx1)

      %       if mod(i,step) ==0
      %       tell=10*(i/step+5);
      %          disp(tell);
      %       end

          if real(Rx1(i))>=0 && imag(Rx1(i))>=0  % 1ο Τεταρτημόριο

            D(1) = abs(Rx1(i)-R3*(1+0*1j));                    % (R3,0)
            D(2) = abs(Rx1(i)-R3*(cos(pi/8)+sin(pi/8)*1j));  % (R3cos(π/8),R3sin(π/8))
            D(3) = abs(Rx1(i)-R3*(cos(pi/4)+sin(pi/4)*1j));%  (R3cos(π/4),R3sin(π/4))
            D(4)          =          abs(Rx1(i)-R3*(cos(3*pi/8)+sin(3*pi/8)*1j));          %
(R3cos(3π/8),R3sin(3π/8))
```

```
                D(5) = abs(Rx1(i)-R3*(0+1j));                    %  (0,R3)
                D(6)                =                abs(Rx1(i)-R2*(cos(pi/12)+sin(pi/12)*1j));%
(R2cos(π/12),R2sin(π/12))
                D(7) = abs(Rx1(i)-R2*(cos(pi/4)+sin(pi/4)*1j));   %  (R2cos(π/4),R2sin(π/4))
                D(8)          =          abs(Rx1(i)-R2*(cos(5*pi/12)+sin(5*pi/12)*1j));          %
(R2cos(5π/12),R2sin(5π/12))
                D(9) = abs(Rx1(i)-R1*(cos(pi/4)+sin(pi/4)*1j));  % (R1cos(π/4),R1sin(π/4))

                minD=min(D);

                switch minD
                   case D(1)
                      ReceiveSignal(i)=R3*(1+0*1j);
                       r1(i,:)= [1 1 0 0 0];
                   case D(2)
                      ReceiveSignal(i)=R3*(cos(pi/8)+sin(pi/8)*1j);
                      r1(i,:)= [0 1 0 0 0];
                   case D(3)
                      ReceiveSignal(i)=R3*(cos(pi/4)+sin(pi/4)*1j);
                      r1(i,:)= [1 1 0 0 1];
                   case D(4)
                      ReceiveSignal(i)= R3*(cos(3*pi/8)+sin(3*pi/8)*1j);
                      r1(i,:)= [0 1 0 0 1];
                   case D(5)
                      ReceiveSignal(i)=R3*(0+1j);
                      r1(i,:)= [0 1 1 0 1];
                   case D(6)
                      ReceiveSignal(i)=R2*(cos(pi/12)+sin(pi/12)*1j);
                      r1(i,:)= [1 0 0 0 0];
                   case D(7)
                      ReceiveSignal(i)=R2*(cos(pi/4)+sin(pi/4)*1j);
                      r1(i,:)= [0 0 0 0 0];
                   case D(8)
                      ReceiveSignal(i)=R2*(cos(5*pi/12)+sin(5*pi/12)*1j);
                      r1(i,:)= [0 0 0 0 1];
                   case D(9)
                      ReceiveSignal(i)=R1*(cos(pi/4)+sin(pi/4)*1j);
                      r1(i,:)= [1 0 0 0 1];
                end

             elseif real(Rx1(i))<0 && imag(Rx1(i))>=0  % 2ο Τεταρτημόριο

                D(1) = abs(Rx1(i)-R3*(-1+0*1j));                    % (-R3,0)
                D(2) = abs(Rx1(i)-R3*(-cos(pi/8)+sin(pi/8)*1j));  % (-R3cos(π/8),R3sin(π/8))
                D(3) = abs(Rx1(i)-R3*(-cos(pi/4)+sin(pi/4)*1j));%  (-R3cos(π/4),R3sin(π/4))
                D(4)      =      abs(Rx1(i)-R3*(-cos(3*pi/8)+sin(3*pi/8)*1j));      %           (-
R3cos(3π/8),R3sin(3π/8))
                D(5) = abs(Rx1(i)-R3*(0+1j));                    %  (0,R3)
                D(6)        =        abs(Rx1(i)-R2*(-cos(pi/12)+sin(pi/12)*1j));%                (-
R2cos(π/12),R2sin(π/12))
                D(7) = abs(Rx1(i)-R2*(-cos(pi/4)+sin(pi/4)*1j));   %  (-R2cos(π/4),R2sin(π/4))
```

```
                D(8) = abs(Rx1(i)-R2*(-cos(5*pi/12)+sin(5*pi/12)*1j)); % (-
R2cos(5π/12),R2sin(5π/12))
                D(9) = abs(Rx1(i)-R1*(-cos(pi/4)+sin(pi/4)*1j)); % (-R1cos(π/4),R1sin(π/4))

                minD=min(D);

                switch minD
                  case D(1)
                    ReceiveSignal(i)=R3*(-1+0*1j);
                    r1(i,:)= [1 1 1 1 0];
                  case D(2)
                    ReceiveSignal(i)=R3*(-cos(pi/8)+sin(pi/8)*1j);
                    r1(i,:)= [1 1 1 0 0];
                  case D(3)
                    ReceiveSignal(i)=R3*(-cos(pi/4)+sin(pi/4)*1j);
                    r1(i,:)= [0 1 1 0 0];
                  case D(4)
                    ReceiveSignal(i)= R3*(-cos(3*pi/8)+sin(3*pi/8)*1j);
                    r1(i,:)= [1 1 1 0 1];
                  case D(5)
                    ReceiveSignal(i)=R3*(0+1j);
                    r1(i,:)= [0 1 1 0 1];
                  case D(6)
                    ReceiveSignal(i)=R2*(-cos(pi/12)+sin(pi/12)*1j);
                    r1(i,:)= [1 0 1 0 0];
                  case D(7)
                    ReceiveSignal(i)=R2*(-cos(pi/4)+sin(pi/4)*1j);
                    r1(i,:)= [0 0 1 0 0];
                  case D(8)
                    ReceiveSignal(i)=R2*(-cos(5*pi/12)+sin(5*pi/12)*1j);
                    r1(i,:)= [0 0 1 0 1];
                  case D(9)
                    ReceiveSignal(i)=R1*(-cos(pi/4)+sin(pi/4)*1j);
                    r1(i,:)= [1 0 1 0 1];
                end

              elseif real(Rx1(i))<0 && imag(Rx1(i))<0  % 3ο Τεταρτημόριο

                D(1) = abs(Rx1(i)-R3*(-1-0*1j));                % (-R3,0)
                D(2) = abs(Rx1(i)-R3*(-cos(pi/8)-sin(pi/8)*1j)); % (-R3cos(π/8),-R3sin(π/8))
                D(3) = abs(Rx1(i)-R3*(-cos(pi/4)-sin(pi/4)*1j));% (-R3cos(π/4),-R3sin(π/4))
                D(4) = abs(Rx1(i)-R3*(-cos(3*pi/8)-sin(3*pi/8)*1j)); % (-R3cos(3π/8),-
R3sin(3π/8))
                D(5) = abs(Rx1(i)-R3*(0-1j));                % (0,-R3)
                D(6) = abs(Rx1(i)-R2*(-cos(pi/12)-sin(pi/12)*1j));% (-R2cos(π/12),-
R2sin(π/12))
                D(7) = abs(Rx1(i)-R2*(-cos(pi/4)-sin(pi/4)*1j)); % (-R2cos(π/4),-R2sin(π/4))
                D(8) = abs(Rx1(i)-R2*(-cos(5*pi/12)-sin(5*pi/12)*1j)); % (-R2cos(5π/12),-
R2sin(5π/12))
                D(9) = abs(Rx1(i)-R1*(-cos(pi/4)-sin(pi/4)*1j)); % (-R1cos(π/4),-R1sin(π/4))
```

```
minD=min(D);

switch minD
   case D(1)
      ReceiveSignal(i)=R3*(-1-0*1j);
      r1(i,:)= [1 1 1 1 0];
   case D(2)
      ReceiveSignal(i)=R3*(-cos(pi/8)-sin(pi/8)*1j);
      r1(i,:)= [0 1 1 1 0];
   case D(3)
      ReceiveSignal(i)=R3*(-cos(pi/4)-sin(pi/4)*1j);
      r1(i,:)= [1 1 1 1 1];
   case D(4)
      ReceiveSignal(i)= R3*(-cos(3*pi/8)-sin(3*pi/8)*1j);
      r1(i,:)= [0 1 1 1 1];
   case D(5)
      ReceiveSignal(i)=R3*(0-1j);
      r1(i,:)= [0 1 0 1 1];
   case D(6)
      ReceiveSignal(i)=R2*(-cos(pi/12)-sin(pi/12)*1j);
      r1(i,:)= [1 0 1 1 0];
   case D(7)
      ReceiveSignal(i)=R2*(-cos(pi/4)-sin(pi/4)*1j);
      r1(i,:)= [0 0 1 1 0];
   case D(8)
      ReceiveSignal(i)=R2*(-cos(5*pi/12)-sin(5*pi/12)*1j);
      r1(i,:)= [1 0 0 1 1];
   case D(9)
      ReceiveSignal(i)=R1*(-cos(pi/4)-sin(pi/4)*1j);
      r1(i,:)= [1 0 1 1 1];
end

elseif real(Rx1(i))>=0 && imag(Rx1(i))<0  % 4ο Τεταρτημόριο

   D(1) = abs(Rx1(i)-R3*(1-0*1j));                  %     (R3          , -0)
D(2) = abs(Rx1(i)-R3*(cos(pi/8)-sin(pi/8)*1j));   %              (R3cos(π/8) , -
R3sin(π/8))
D(3) = abs(Rx1(i)-R3*(cos(pi/4)-sin(pi/4)*1j));%                 (R3cos(π/4) , -
R3sin(π/4))
D(4) = abs(Rx1(i)-R3*(cos(3*pi/8)-sin(3*pi/8)*1j)); %         (R3cos(3π/8),-
R3sin(3π/8))
D(5) = abs(Rx1(i)-R3*(0-1j));                    %        (0           , -R3)
D(6) = abs(Rx1(i)-R2*(cos(pi/12)-sin(pi/12)*1j));%            (R2cos(π/12) , -
R2sin(π/12))
D(7) = abs(Rx1(i)-R2*(cos(pi/4)-sin(pi/4)*1j));    %           (R2cos(π/4)    , -
R2sin(π/4))
D(8) = abs(Rx1(i)-R2*(cos(5*pi/12)-sin(5*pi/12)*1j)); %   (R2cos(5π/12), -
R2sin(5π/12))
D(9) = abs(Rx1(i)-R1*(cos(pi/4)-sin(pi/4)*1j));   %           (R1cos(π/4)    , -
R1sin(π/4))
```

```
            minD=min(D);

            switch minD
               case D(1)
                  ReceiveSignal(i)=R3*(1-0*1j);
                  r1(i,:)= [1 1 0 0 0];
               case D(2)
                  ReceiveSignal(i)=R3*(cos(pi/8)-sin(pi/8)*1j);
                  r1(i,:)= [1 1 0 1 0];
               case D(3)
                  ReceiveSignal(i)=R3*(cos(pi/4)-sin(pi/4)*1j);
                  r1(i,:)= [0 1 0 1 0];
               case D(4)
                  ReceiveSignal(i)= R3*(cos(3*pi/8)-sin(3*pi/8)*1j);
                  r1(i,:)= [1 1 0 1 1];
               case D(5)
                  ReceiveSignal(i)=R3*(0-1j);
                  r1(i,:)= [0 1 0 1 1];
               case D(6)
                  ReceiveSignal(i)=R2*(cos(pi/12)-sin(pi/12)*1j);
                  r1(i,:)= [1 0 0 1 0];
               case D(7)
                  ReceiveSignal(i)=R2*(cos(pi/4)-sin(pi/4)*1j);
                  r1(i,:)= [0 0 0 1 0];
               case D(8)
                  ReceiveSignal(i)=R2*(cos(5*pi/12)-sin(5*pi/12)*1j);
                  r1(i,:)= [0 0 0 1 1];
               case D(9)
                  ReceiveSignal(i)=R1*(cos(pi/4)-sin(pi/4)*1j);
                  r1(i,:)= [1 0 0 1 1];
            end

         end
%
%          b1=startsample(i,:);
%          b2=r1(i,:);
%         ber=ber+sum(xor(b1,b2));
         end

      ber=sum(sum(xor(r1,startsample)));
      ber=ber/(5*txlength);
```

Η Εφαρμογή αξιολόγησης

```
clear;
Txlength=10000;
EbNodB=0:2:12;
ebno=EbNodB;
ber1=zeros(1,length(ebno));
ber2=zeros(1,length(ebno));
ber3=zeros(1,length(ebno));
ber4=zeros(1,length(ebno));
```

```
snrarrayQ=zeros(1,length(ebno));
snrcalcarrayQ=zeros(1,length(ebno));

snrarray8=zeros(1,length(ebno));
snrcalcarray8=zeros(1,length(ebno));

snrarray16=zeros(1,length(ebno));
snrcalcarray16=zeros(1,length(ebno));

snrarray32=zeros(1,length(ebno));
snrcalcarray32=zeros(1,length(ebno));


for l=1:length(ebno)
   clearvars -EXCEPT Tx Txfiltered startsample ps pn BER Txlength snrdb ber1 ber2 ber3
ber4 ebno l snrarrayQ snrcalcarrayQ snrarray8 snrcalcarray8 snrarray16 snrcalcarray16
snrarray32 snrcalcarray32

   if l==1
   ploted=0;
   else
   TxSW=0;
   end
   ebnoinput=ebno(l);
   PSK=1;
   TxRx
   ber1(l)=BER;
   snrarrayQ(l)=snr;
   snrcalcarrayQ(l)=snr2;
end
clearvars -EXCEPT ber1 ber2 ber3 ber4 Txlength ebno snrarrayQ snrcalcarrayQ snrarray8
snrcalcarray8 snrarray16 snrcalcarray16 snrarray32 snrcalcarray32
for l=1:length(ebno)
  clearvars -EXCEPT Tx Txfiltered startsample ps pn snr BER Txlength snrdb ber1 ber2 ber3
ber4 ebno l snrarrayQ snrcalcarrayQ snrarray8 snrcalcarray8 snrarray16 snrcalcarray16
snrarray32 snrcalcarray32

   if l==1
   ploted=0;
   else
   TxSW=0;
   end
   ebnoinput=ebno(l);
   PSK=2;
   TxRx
   ber2(l)=BER;

   snrarray8(l)=snr;
   snrcalcarray8(l)=snr2;
end
clearvars -EXCEPT ber1 ber2 ber3 ber4 Txlength ebno snrarrayQ snrcalcarrayQ snrarray8
snrcalcarray8 snrarray16 snrcalcarray16 snrarray32 snrcalcarray32
for l=1:length(ebno)
  clearvars -EXCEPT R1 R2 Tx Txfiltered startsample ps pn snr BER Txlength snrdb ber1 ber2
ber3 ber4 ebno l snrarrayQ snrcalcarrayQ snrarray8 snrcalcarray8 snrarray16
snrcalcarray16 snrarray32 snrcalcarray32
```

```
   if l==1
   ploted=0;
   else
   TxSW=0;
   end
   ebnoinput=ebno(l);
   PSK=3;
   TxRx
   ber3(l)=BER;

   snrarray16(l)=snr;
   snrcalcarray16(l)=snr2;
end
clearvars -EXCEPT ber1 ber2 ber3 ber4 Txlength ebno snrarrayQ snrcalcarrayQ snrarray8
snrcalcarray8 snrarray16 snrcalcarray16 snrarray32 snrcalcarray32
for l=1:length(ebno)
  clearvars -EXCEPT R1 R2 R3 Tx Txfiltered startsample ps pn snr BER PSK SNRG Txlength
ber1 ber2 ber3 ber4 ebno l snrarrayQ snrcalcarrayQ snrarray8 snrcalcarray8 snrarray16
snrcalcarray16 snrarray32 snrcalcarray32

   if l==1
   ploted=0;
   else
   TxSW=0;
   end
   ebnoinput=ebno(l);
   PSK=4;
   TxRx
   ber4(l)=BER;

   snrarray32(l)=snr;
   snrcalcarray32(l)=snr2;
end
clearvars -EXCEPT ber1 ber2 ber3 ber4 Txlength ebno snrarrayQ snrcalcarrayQ snrarray8
snrcalcarray8 snrarray16 snrcalcarray16 snrarray32 snrcalcarray32




%ber1=log10(ber1);
figure;
title(' Γραφική Παράσταση Bit Erorr Rate - Eb/No ')
subplot(2,2,1);
theoreticalBER4 =0.5*erfc(sqrt(10.^(ebno/10)));
theoreticalber4= berawgn(ebno,'psk',4,'nondiff');
semilogy(ebno,ber1,'b*');hold on;
xlabel('Energy per Bit/Noise Spectral Density (dB)');
ylabel('Bit Error Rate Propability');

plotHandle=semilogy(ebno,theoreticalber4,'k-o');hold off;
set(plotHandle,'LineWidth',1.5);
legend('Simulated','Theoretical-QPSK');
grid on;
```

```
subplot(2,2,2);
theoreticalber8 = berawgn(ebno,'psk',8,'nondiff');

semilogy(ebno,ber2,'b*');hold on
% Plot theoretical results.
xlabel('Energy per Bit/Noise Spectral Density (dB)');
ylabel('Bit Error Rate Propability');

semilogy(ebno,theoreticalber8,'k-o');hold off
legend('Simulated','Theoretical-8PSK');
grid on

subplot(2,2,3);
theoreticalber16= berawgn(ebno,'psk',16,'nondiff');


semilogy(ebno,ber3,'b*');hold on
xlabel('Energy per Bit/Noise Spectral Density (dB)');
ylabel('Bit Error Rate Propability');

semilogy(ebno,theoreticalber16,'k-o');hold off
legend('Simulated','Theoretical-16APSK');
grid on

subplot(2,2,4);
theoreticalber32= berawgn(ebno,'psk',32,'nondiff');

semilogy(ebno,ber4,'b*');hold on
xlabel('Energy per Bit/Noise Spectral Density (dB)');
ylabel('Bit Error Rate Propability');

semilogy(ebno,theoreticalber32,'k-o');hold off
legend('Simulated','Theoretical-32APSK');
grid on



% SNR Detection tables
snrerrorQ=snrcalcarrayQ-snrarrayQ;
snrerror8=snrcalcarray8-snrarray8;
snrerror16=snrcalcarray16-snrarray16;
snrerror32=snrcalcarray32-snrarray32;

f1 = figure('Position',[100 100 900 450]);
datas=[snrarrayQ ; snrcalcarrayQ ; snrerrorQ ;];
cnames = {'0','2','4','6','8','10','12'};
rnames = {'SNR QPSK','Calculated SNR','error'};
t1 = uitable('Data',datas,'ColumnName',cnames,'RowName',rnames,'Tag','QPSK SNR
CALCULATION',...
'TooltipString','Table 3','Parent',f1,'Position',[25 340 850 100]);

datas=[snrarray8 ; snrcalcarray8; snrerror8;];
cnames = {'0','2','4','6','8','10','12'};
rnames = {'SNR 8PSK','Calculated SNR','error'};
t2 = uitable('Data',datas,'ColumnName',cnames,'RowName',rnames,'Tag','8PSK SNR
CALCULATION',...
'TooltipString','Table 3','Parent',f1,'Position',[25 230 850 100]);
```

```
datas=[snrarray16 ; snrcalcarray16; snrerror16;];
cnames = {'0','2','4','6','8','10','12'};
rnames = {'SNR 16PSK','Calculated SNR','error'};
t3 = uitable('Data',datas,'ColumnName',cnames,'RowName',rnames,'Tag','16PSK SNR
CALCULATION',...
'TooltipString','Table 3','Parent',f1,'Position',[25 120 850 100]);

datas=[snrarray32 ; snrcalcarray32; snrerror32;];
cnames = {'0','2','4','6','8','10','12'};
rnames = {'SNR 32PSK','Calculated SNR','error'};
t4 = uitable('Data',datas,'ColumnName',cnames,'RowName',rnames,'Tag','32PSK SNR
CALCULATION',...
'TooltipString','Table 3','Parent',f1,'Position',[25 10 850 100]);

f2 = figure('Position',[100 100 900 450]);
subplot(2,2,1)
plot(ebno,snrarrayQ,'-b');hold on
plot(ebno,snrcalcarrayQ,'-r');hold off
legend('Real SNR','');
xlabel('Eb/No');
ylabel('Sound To Noise Ratio');

subplot(2,2,2)
plot(ebno,snrarray8,'-b');hold on
xlabel('Eb/No');
ylabel('Sound To Noise Ratio');
plot(ebno,snrcalcarray8,'-r');hold off

subplot(2,2,3)
plot(ebno,snrarray16,'-b');hold on
xlabel('Eb/No');
ylabel('Sound To Noise Ratio');
plot(ebno,snrcalcarray32,'-r');hold off

subplot(2,2,4)
plot(ebno,snrarray32,'-b');hold on
xlabel('Eb/No');
ylabel('Sound To Noise Ratio');
plot(ebno,snrcalcarray32,'-r');hold off
```