ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΤΕ

Ασύμφωνα OFDM MIMO συστήματα με διαμόρφωση MFSK

Non-Coherent OFDM MIMO Systems with MFSK Modulation

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
ΤΟΥ

ΕΜΜΑΝΟΥΗΛ ΜΟΥΤΑΦΗ
513124
ΚΩΔΙΚΟΣ ΠΤΥΧΙΑΚΗΣ: 17175Μ

Επιβλέπων: Αθανάσιος Ιωσηφίδης, Αναπληρωτής Καθηγητής

Θεσσαλονίκη, Ιούνιος 2017- Σεπτέμβριος 2018

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία εξετάζεται ένα σχήμα πολλαπλών κεραιών (πολλαπλών εισόδων πολλαπλών εξόδων) με ορθογωνική πολυπλεξία διαίρεσης συχνότητας σε συνδυασμό με διαμόρφωση μετατόπισης συχνότητας και ασύμφωνη αποδιαμόρφωση ή MIMO OFDM-MFSK (Multiple-Input Multiple-Output, Orthogonal Frequency Division Multiplexing M-ary Frequency Keying) [1] σε περιβάλλον Λευκού θορύβου (AWGN) και διαλείψεων Rayleigh. Με τη χρήση του λογισμικού MATLAB υπολογίστηκαν οι ρυθμοί λαθών bit συναρτήσει της σηματοθορυβικής σχέσης και για διάφορες τιμές *M* της διαμόρφωσης FSK. Είναι ένα σύστημα υποψήφιο για χρήση στην επόμενη γενιά κινητών επικοινωνιών ειδικότερα για εφαρμογές που απαιτούν απλούς δέκτες όπως το Διαδίκτυο των Πραγμάτων (Internet of Things – IoT). Το εξεταζόμενο σχήμα διαμόρφωσης είναι εξαιρετικά ανθεκτικό σε γρήγορες διαλείψεις και το ιδιαίτερο χαρακτηριστικό που αναλύεται είναι πως χρησιμοποιείται στον δέκτη ασύμφωνη αποδιαμόρφωση, η οποία βοηθά στην ευκολότερη υλοποίηση του συστήματος. Επίσης, ένα ακόμη σημαντικό πλεονέκτημα είναι πως έχει χαμηλή κατανάλωση ισχύος. Στο πλαίσιο αυτό, αναλύεται σε βάθος ο τρόπος δημιουργίας του συμβόλου προς εκπομπή OFDM-MFSK καθώς και η κωδικοποίηση που υπόκειται στον χώρο-χρόνο-συχνότητα έτσι ώστε να αποδοθεί στο MIMO κανάλι. Επίσης αναλύεται το σχήμα κωδικοποίησης ελέγχου ισοτιμίας χαμηλής πυκνότητας (Low Density Parity Check – LDPC) των bits πληροφορίας που επιπρόσθετα χρησιμοποιείται για την περαιτέρω βελτίωση του ρυθμού λαθών bit. Τα αποτελέσματα δείχνουν ότι το προτεινόμενο σχήμα παρουσιάζει πολύ ικανοποιητική επίδοση στα υπό μελέτη κανάλια και αποτελεί σημαντική εναλλακτική λύση εφαρμογών που δεν απαιτούν υψηλή φασματική απόδοση.

ABSTRACT

This thesis presents the study of a a multiple antennas (multi-input and multi-output – MIMO) scheme with orthogonal frequency division multiplexing in combination with M-ary frequency shift keying (MIMO OFDM-MFSK) [1] and non-coherent detection in a white noise (AWGN) and Rayleigh fading channel. MATLAB was used in order to simulate and calculate the Bit Error Rate (BER) of the system for various Signal-to-Noise ratio and different *M* values of FSK modulation. This scheme is considered to be a useful system for next generation of mobile communications (5G), especially for low complexity receiver applications such as the ones involved inthe Internet of Things. This particular modulation system is extremely resistant at fast fading channels and and the special feature that is analyzed is the non-coherent detection capability that leads to low complexity and low power consumption receivers. In this context, a detailed analysis of the OFDM-MFSK symbols construction is given together with a the Space-Time-Frequency (STF) coding techinque, which is applied in order to use the MIMO channel effectively. In addition, a low density parity check (LDPC) coding schme of the information bits is used for BER enhancemment. The results show that the proposed scheme has very good performance under the specific channels that are studied and prove that it is a significant alternative solution for wireless application that do not require high spectral efficiency.

Λέξεις-κλειδιά

OFDM-MFSK, κανάλι MIMO, Space-Time-Frequency κωδικοποίηση, IoT, Ασύμφωνη αποδιαμόρφωση, LDPC κωδικοποίηση-αποκωδικοποίηση

Πρόλογος

Τα προσωπικά οφέλη που αποκτήθηκαν λόγω της περάτωσης αυτής της πτυχιακής εργασίας είναι αρκετά. Αρχικά, αποκτήθηκε εμπειρία ως προς την αναζήτηση έγκυρων πηγών και μελέτης αυτών για την σωστή αξιοποίησή τους. Διαπιστώθηκε ο τρόπος προσέγγισης ενός παντελώς αγνώστου θέματος έτσι ώστε να υπάρχει κάποια εμπειρία για την αντιμετώπιση μελλοντικών εργασιών ή project. Αποκτήθηκε γνώση πάνω στο προγραμματιστικό περιβάλλον της MATLAB και συγκεκριμένα ως προς την προσομοίωση ενός τηλεπικοινωνιακού συστήματος. Τα διάφορα προβλήματα που προέκυψαν και η επίλυσή τους θα μπορούσαν να προσομοιάζουν μια πραγματική κατάσταση που θα μπορούσε να προκύψει σε εργασιακό περιβάλλον. Επίσης, σημαντική εμπειρία ήταν και η συνεργασία μεταξύ δύο ατόμων που υπήρξε στην διάρκεια περάτωσης της πτυχιακής εργασίας και υπενθυμίζει πως πάντα η ομαδική δουλειά είναι το παν σε οποιοδήποτε εργασιακό ή ακαδημαϊκό περιβάλλον.

Ευχαριστώ τον κύριο Αθανάσιο Ιωσηφίδη για την έμπνευση, την συνεργασία και την υποστήριξη που μου πρόσφερε.

1 ΕΙΣΑΓΩΓΗ

Στη σημερινή εποχή όπου η 5η γενιά κινητών επικοινωνιών τείνει όλο και περισσότερο να γίνει διαθέσιμη για εμπορική χρήση, , το Διαδίκτυο των Πραγμάτων (Internet of Things) έρχεται ακόμη πιο κοντά στις ανθρώπινες ζωές. Το Internet of Things (IoT) είναι ένας όρος ο οποίος χρησιμοποιείται για να περιγράψει καθημερινά αντικείμενα και όχι μόνο που χρησιμοποιεί ένας άνθρωπος στη ζωή του, τα οποία είναι συνδεδεμένα στο ιντερνέτ και επικοινωνούν με αυτό παρέχοντας ποικίλες πληροφορίες σε φυσικά πρόσωπα ή σε άλλες συσκευές. Έχει πολλές εφαρμογές σε όλους σχεδόν τους τομείς της ζωής, στην απλή καθημερινότητα, στην ιατρική, στην βιομηχανία, στη γεωργία, στις ανανεώσιμες πηγές κ.α. Εφαρμογές που ξεκινούν από μικρούς wearable αισθητήρες που παρακολουθούν τα ζωτικά χαρακτηριστικά ενός ανθρώπου μέχρι απομακρυσμένη χρήση ιατρικών απεικονιστικών μηχανημάτων, και από απλά αισθητήρια υγρασίας μέχρι έλεγχο βιομηχανικών μηχανών σε πραγματικό χρόνο εμπεριέχονται στις εφαρμογές που θα υποστηρίζει ακόμη πιο ενεργά το Internet of Things [9]. Μια κλασσική εφαρμογή Internet of Things η οποία και παρουσιάζεται συνήθως σαν παράδειγμα κατανόησης, είναι αυτή με το ψυγείο και το άδειο κουτί γάλα εντός

του ψυγείου· το ψυγείο ‘καταλαβαίνει’ πως το γάλα έχει τελειώσει και έτσι επικοινωνεί με τον χρήστη μέσω της κινητής του συσκευής υπενθυμίζοντάς του να αγοράσει γάλα.

Το δίκτυο του Internet of Things θα είναι μια σύνθετη τηλεπικοινωνιακή οντότητα η οποία θα έχει ανάγκη από διάφορα αποδοτικά τηλεπικοινωνιακά συστήματα τα οποία θα πρέπει να μπορούν να συνυπάρχουν στο ίδιο δίκτυο και ταυτοχρόνως να εναρμονίζονται με τις τεχνολογίες της προηγούμενης γενιάς.

Ένα από τα πιο σημαντικά είδη επικοινωνιών στο Internet of Things είναι αυτή του machine to machine (M2M) ή αλλιώς device to device, όπου δυο μηχανές επικοινωνούν μεταξύ τους δίχως την παρέμβαση ανθρώπου [9]. Οι M2M επικοινωνίες υπάρχουν σε πολλές ‘smart’ εφαρμογές, όπου διάφορα αισθητήρια στέλνουν μαζικά πληροφορίες σε μια συσκευή ελέγχου ή κάποιον server. Μπορεί να καταλάβει κανείς πως οι αισθητήρες που παρέχουν πληροφορίες δεν είναι πάντοτε κοντά στη συσκευή ελέγχου και ίσως να βρίσκονται σε απομακρυσμένη τοποθεσία και χωρίς δυνατότητες παροχής μόνιμης τροφοδότησης. Οι αισθητήρες είναι απλές συσκευές στις οποίες η ασύμφωνη αποδιαμόρφωση θα ήταν ένα σημαντικό πλεονέκτημα.

Σ’ αυτήν την εργασία λοιπόν εξετάζεται ένα σύστημα το οποίο βασίζεται στην κλασική ψηφιακή διαμόρφωση μετατόπισης συχνότητας *M* συμβόλων (MFSK) με χρήση ορθογώνιας πολυπλεξίας διαίρεσης συχνότητας (OFDM) – που χρησιμοποιείται ευρέως στη $4^{η}$ γενιά κινητών επικοινωνιών – και μαζί συνθέτουν το OFDM-MFSK. Το OFDM-MFSK σχήμα θεωρείται πως είναι κατάλληλο για εφαρμογές όπου η χαμηλή κατανάλωση ενέργειας είναι απαραίτητη [1]. Επίσης, λόγω της φύσης του OFDM, η διαμορφωμένη πληροφορία είναι εξαιρετικά ανθεκτική σε κανάλια με γρήγορες διαλείψεις. Ένα σημαντικό μειονέκτημα που χαρακτηρίζει το σύστημα αυτό είναι η χαμηλή φασματική απόδοση [2]. Οι αισθητήρες δεν έχουν την ανάγκη να στείλουν μεγάλο και περίπλοκο όγκο δεδομένων, οπότε σ’ αυτήν την περίπτωση, το OFDM-MFSK κρίνεται κατάλληλο για χρήση. Άλλες δημοσιεύσεις περιγράφουν συστήματα για τα οποία μπορεί να μεταδοθεί μεγαλύτερος όγκος πληροφοριών με υβριδικό OFDM-MFSK-MPSK σχήμα, στο οποίο εκμεταλλεύονται όχι μόνο οι συχνότητες των συνιστωσών αλλά και οι φάσεις αυτών έτσι ώστε να μπορεί να αποσταλεί επιπλέον πληροφορία [2]. Μια άλλη τεχνική είναι να μειωθεί ο αριθμός συμβόλων *M*. Η χρήση σύμφωνης αποδιαμόρφωσης είναι καλή για κινητούς σταθμούς αργής ταχύτητας. Σε περιβάλλοντα με πολύ γρήγορες διαλείψεις είναι αδύνατο να υπάρχει γνώση του καναλιού στο δέκτη. Ένα πολύ σημαντικό προτέρημα του συστήματος που παρουσιάζεται είναι ότι μπορεί να κατασκευαστεί ασύμφωνος αποδιαμορφωτής. Αυτό αυτομάτως σημαίνει ότι η γνώση του καναλιού για αποδιαμόρφωση δεν είναι απαραίτητη. Επακόλουθα, ο

αποδιαμορφωτής είναι πολύ χαμηλής περιπλοκότητας και πολύ πιο εύκολος ως προς την κατανόηση και την υλοποίησή του [4].

Πριν τη διαμόρφωση του συμβόλου OFDM-MFSK χρησιμοποιείται στο σύστημα μια κωδικοποίηση διόρθωσης λαθών κωδίκων LDPC η οποία αναλύεται στο 3^{ο} κεφάλαιο. Η κωδικοποίηση βελτιώνει τη πιθανότητα σφάλματος bit στον δέκτη σε μεγάλο βαθμό όπως θα διαπιστωθεί και στο 4^{ο} κεφάλαιο. Με τη χρήση LDPC το μέγεθος της βασικής πληροφορίας μειώνεται ανάλογα με τον ρυθμό rc που χαρακτηρίζει έναν συγκεκριμένο κώδικα. Ένα αρνητικό ως προς τη προσομοίωση είναι πως για να τρέξει μια φορά ένας αλγόριθμος LDPC καθυστερεί αρκετά, λόγω του μεγάλου όγκου των πράξεων που γίνονται. Οπότε για την παραγωγή και συλλογή αποτελεσμάτων χρειάζεται αρκετός χρόνος.

Στο πρακτικό μέρος της εργασίας χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον MATLAB για να κατασκευαστεί το τηλεπικοινωνιακό μοντέλο και να γίνει η προσομοίωση. Στο 3^{ο} κεφάλαιο θα αναλυθεί περιεκτικά όλη η διαδικασία μοντελοποίησης με πομπό κανάλι και δέκτη και τη βοήθεια μεταβλητών-κλειδιά και διαγράμματος ροής του προγράμματος. Επίσης, θα επεξηγηθούν οι ειδικές συναρτήσεις που χρησιμοποιήθηκαν για την υλοποίηση του LDPC. Με τη βοήθεια του διαγράμματος ροής και την αναφορά των βασικών μεταβλητών θα επεξηγηθούν τα κρίσιμα και ενδιαφέροντα σημεία του κώδικα. Στη συνέχεια, στο 4^{ο} κεφάλαιο τα αποτελέσματα των προσομοιώσεων θα απεικονιστούν και θα συγκριθούν. Για καλύτερη σύγκριση αποτελεσμάτων στην προσομοίωση κατασκευάστηκαν επίσης, εκτός από διαφορετικά *M* της FSK διαμόρφωσης , επιπλέον καμπύλες από σύστημα SISO καθώς και από θεωρητικές καμπύλες.

Το σύστημα εξετάστηκε κυρίως σε 64FSK διαμόρφωση με κώδικα LDPC ρυθμού ½ με 408 bits κωδικοποιημένης λέξης. Τα συμπεράσματα της συνολικής περάτωσης της εργασίας αναφέρονται στο 5^{ο} και τελευταίο κεφάλαιο της εργασίας καθώς και πιθανές βελτιώσεις αυτής.

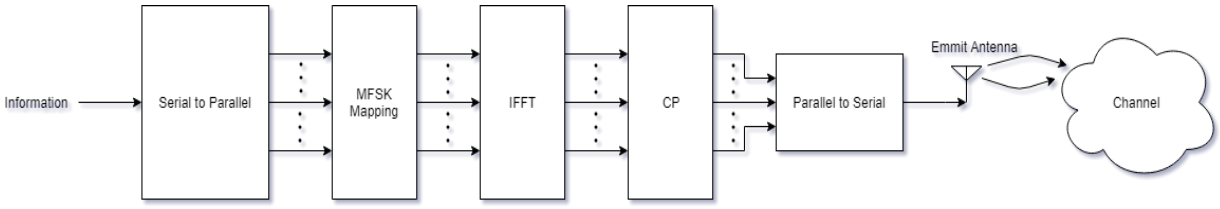
2 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ OFDM-MFSK

Στο παρόν κεφάλαιο θα παρουσιαστεί βήμα βήμα και αναλυτικά όλη η επεξήγηση του συστήματος OFDM-MFSK, η κατασκευή του συμβόλου με Space-Time-Frequency κωδικοποίηση, η εκπομπή του στο MIMO κανάλι καθώς και πως ανιχνεύεται το σήμα στο δέκτη και η επεξεργασία που γίνεται σ' αυτόν. Επίσης θα αναλυθεί ο τρόπος κωδικοποίησης διόρθωσης λαθών της χρήσιμης πληροφορίας και επίσης ο τρόπος με τον οποίον γίνεται η αποκωδικοποίηση.

2.1 ΤΑ ΒΑΣΙΚΑ ΤΟΥ OFDM

Στο σύστημα χρησιμοποιείται η πολυπλεξία OFDM, η οποία χρησιμοποιείται σε όλα τα σύγχρονα συστήματα ασύρματων και κινητών επικοινωνιών. Η κλασσική του υλοποίηση θα επεξηγηθεί συνοπτικά σε αυτήν την ενότητα. Η διαδικασία που ακολουθείται είναι να χωριστούν τα σύμβολα βασικής ζώνης (συνήθως N) σε N πολλαπλά υποφέροντα με χρήση του Αντίστροφου Μετ/σμού Fourier (IFFT). Το σύνολο των N υποφερόντων τα οποία θα δεχτούν IFFT ονομάζεται σύμβολο OFDM.

Στην περίπτωση του σχήματος διαμόρφωσης αυτής της εργασίας, δεδομένου ότι η ψηφιακή διαμόρφωση MFSK απαιτεί M φέρονται (εκ των οποίνων μόνο το ένα φέρον έχει ενέργεια κάξε φορά όπως επεξηγείται σε επόμενη ενότητα) θεωρήθηκε για απλότητα ότι $M = N$, δηλαδή ότι ο αριθμό των υποφερόντων του OFDM είναι ίσος με τον αριθμό φερόντων (και άρα τον αριθμό συμβόλων) του MFSK. Στη γενική περίπτωση απατείται ο λόγος $n = N/M$ να είναι ακέραιος έτσι ώστε n συνολικά διαμορφωμένα σύμβολα MFSK να «φορτώνονται» στο OFDM σύμβολο των N υποφερόντων. Η περίπτωση που επιλέχτηκε δεν επιδρά στην επίδοση του συστήματος αφού τα κανάλια που χρησιμοποιούνται είναι απλά και δε έχουν χαρακτηριστικά επιλεκτικότητας στη συχνότητα. Παρακάτω στο Σχήμα 1 απεικονίζεται σε μπλοκ διάγραμμα ένας διαμορφωτής OFDM.



Σχήμα 1: Ένα κλασσικό μπλοκ διάγραμμα OFDM

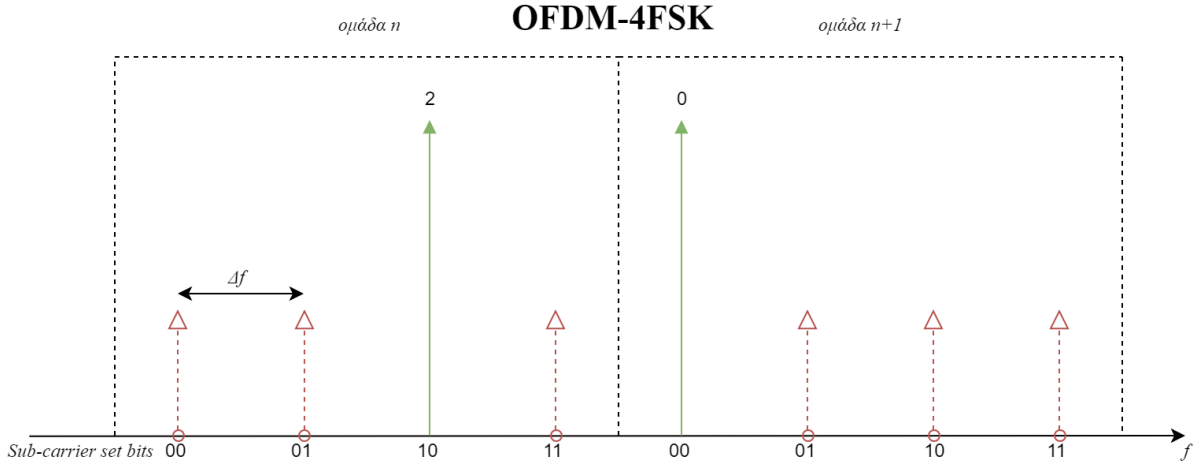
Η πληροφορία, αρχικά σε ψηφιακή μορφή, περνάει από σειριακή προσπέλαση σε παράλληλη και το επόμενο βήμα είναι να επιλεχθεί ο τρόπος διαμόρφωσης. Σε αυτό το σύστημα επιλέγεται διαμόρφωση MFSK. Αφού επιτευχθεί η διαμόρφωση λαμβάνει χώρα ο αλγόριθμος του Αντίστροφου Μετ/σμού Fourier, όπου η διαμορφωμένη πληροφορία ενοποιείται σε ένα σύνολο N δειγμάτων. Στη συνέχεια, εφαρμόζεται μέθοδος προφύλαξης του σήματος από τη διασυμβολική παρεμβολή με την προσθήκη κυκλικού προθέματος (cyclic prefix). Έπειτα τα OFDM σύμβολα ξαναγυρνάνε σε σειριακή προσπέλαση και εκπέμπονται στο κανάλι. Η αποδιαμόρφωση ακολουθεί την αντίστροφη ακολουθία [3].

2.2 ΤΟ OFDM-MFSK

Ο τρόπος λειτουργίας του σχήματος διαμόρφωσης στο σύστημα OFDM-MFSK δεν είναι σύνθετος και μπορεί να εξηγηθεί εύκολα. Τα bits πληροφορίας αντιστοιχίζονται σε MFSK σύμβολα όπου το κάθε σύμβολο έχει $\log_2 M$ bits, και τα bits με τη σειρά τους αντιστοιχίζονται ή με βάση το φυσικό δυαδικό σύστημα (στην περίπτωση της εργασίας) ή με Gray κωδικοποίηση. Στην FSK διαμόρφωση κάθε σύμβολο είναι μια ξεχωριστή συχνότητα, (η κάθε συχνότητα μπορεί να αποτυπωθεί μαθηματικά με μια ξεχωριστή διάσταση ανά σύμβολο) με απόσταση Δf (Hz) από την γειτονική της, οπότε για παράδειγμα ένα σύστημα 4FSK ($M = 4$), θα υπάρχουν 4 υποφέρουσες που στη καθεμία αντιστοιχίζεται ένα σύμβολο

των $\log_2 4 = 2$ bits [3]. Έπειτα τα MFSK σύμβολα ομαδοποιούνται σε *n*-άδες. Κάθε *n*-άδα είναι ένα OFDM σύμβολο. Είναι σημαντικό να τονισθεί πως σε MFSK σύμβολο που καταλαμβάνει *M* υποφέρουσες, μόνο μια υποφέρουσα είναι μη-μηδενική, δηλαδή μόνο μία συχνότητα περιέχει την διαμορφωμένη πληροφορία ενώ κάθε άλλη υποφέρουσα υπάρχει μόνο ο θόρυβος [1]. Σ' αυτήν την πρακτική οφείλεται η χαμηλή φασματική απόδοση. Όσο μεγαλύτερο το *M* τόσο χαμηλότερη θα είναι η φασματική απόδοση [2].

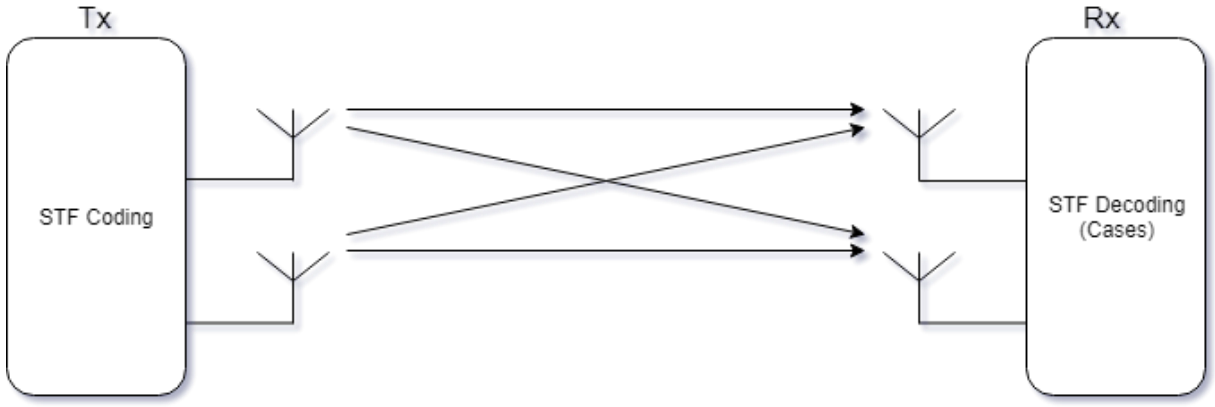
Το παρακάτω σχήμα Σχήμα 2 απεικονίζει ένα OFDM-4FSK σύστημα. Ο οριζόντιος άξονας είναι συχνότητα. Με κόκκινες κάθετες γραμμές συμβολίζονται τα ανενεργά sub-carriers τα οποία τοποθετήθηκαν με μικρό πλάτος διότι εμπεριέχουν μόνο θόρυβο. Με πράσινες γραμμές είναι τα ενεργά sub-carrier ένα για κάθε ομάδα *n*. Κάτω από τα sub-carriers αναφέρονται οι συνδυασμοί των bits που χρειάζονται για να ενεργοποιήσουν το συγκεκριμένο sub-carrier και πάνω από αυτά συμβολίζεται ο δεκαδικός αριθμός του ενεργού. Το σχήμα δείχνει τι γίνεται στις πρώτες δύο ομάδες. Για τα τέσσερα υποφέροντα υπάρχουν τέσσερα ζεύγη bits τα οποία τα ενεργοποιούν. Ο συνδυασμός 10 στην $1^{η}$ ομάδα υποφερόντων ενεργοποιεί το υπ' αριθμόν $3^{ο}$ υποφέρον (Η αρίθμηση γίνεται από το μηδέν, οπότε το υποφέρον ονομάζεται στο δεκαδικό 2). Για την $2^{η}$ ομάδα ο συνδυασμός 00 ενεργοποιεί το $1^{ο}$ υποφέρον (το υποφέρον ονομάζεται στο δεκαδικό 0)



Σχήμα 2: Η γραφική αναπαράσταση ενός συστήματος OFDM με 4FSK διαμόρφωση

2.2.1 MIMO

Για να μπορέσει να υπάρξει καλύτερη ανίχνευση του σήματος στο δέκτη, το σύστημα χρησιμοποιεί ένα 2×2 MIMO κανάλι (2 κεραίες κατά την εκπομπή και 2 στη λήψη). Γενικά το MIMO είναι μια προχωρημένη τεχνική μετάδοσης με κέρδος κατά τη λήψη το άθροισμα των δύο κεραιών επί τη σηματοθορυβική σχέση [3].



Σχήμα 3: Το μπλοκ διάγραμμα του MIMO 2X2 συστήματος με πομπό και δέκτη

Για να μπορέσει να υλοποιηθεί το MIMO έτσι ώστε στο δέκτη να μπορεί να γίνει ασύμφωνη αποκωδικοποίηση χωρίς δηλαδή τη γνώση του καναλιού υπάρχουν διάφορες τεχνικές κωδικοποίησης. Αυτές οι τεχνικές μπορούν να υλοποιηθούν στο πεδίο του χώρου, του χρόνου αλλά και της συχνότητας. Μια γνωστή κωδικοποίηση είναι η Alamouti Space-Time (ST). Η εργασία βασισμένη σε ένα σχήμα κωδικοποίησης ST διαμόρφωσης PPM, θα εξετάσει μια κωδικοποίηση η οποία εκτός από τον χώρο-χρόνο θα υλοποιηθεί και στη συχνότητα και θα λέγεται Space-Time-Frequency (STF) (OFDM) [1] με διαμόρφωση FSK. Στο Σχήμα 3 αποτυπώνεται ένα στοιχειώδες μπλοκ διάγραμμα ενός 2×2 MIMO καναλιού με τις βαθμίδες χώρου-χρόνου-συχνότητας κωδικοποίησης στον πομπό και αποκωδικοποίησης στον δέκτη. Η κάθε κεραία εκπομπής απευθύνεται και στέλνει δεδομένα και στις δύο κεραίες λήψης.

2.3 ΟΡΓΑΝΩΣΗ ΠΛΗΡΟΦΟΡΙΑΣ ΠΟΜΠΟΥ

2.3.1 ΚΩΔΙΚΟΠΟΙΗΣΗ ΚΑΙ ΕΚΠΟΜΠΗ ΣΥΜΒΟΛΩΝ STF

Με την STF κωδικοποίηση έχει αποδειχθεί ότι καταπολεμούνται οι διαλείψεις σε μεγάλο βαθμό και ενισχύονται οι ρυθμοί πληροφορίας. Με κατάλληλη παραμετροποίηση μπορεί να χρησιμοποιηθεί και σε ασύμφωνη αποδιαμόρφωση. Τα m bits πληροφορίας αφού περάσουν από τον Forward Error Correction (FEC) Encoder όπου υλοποιείται με τη χρήση κωδίκων LDPC –ο οποίος θα αναλυθεί σε επόμενο κεφάλαιο-, θα διαμορφωθούν με FSK. Το σύστημα που παρουσιάζεται χρησιμοποιεί 2 ομάδες συμβόλων για να φτιάξει την κωδική λέξη. Οπότε τα συνολικά bits που χρειάζονται για την κατασκευή της λέξης είναι $2\log_2 M$. M bits για το σύμβολο $\mathbf{s}_1$ και M bits για το σύμβολο $\mathbf{s}_2$. Το επόμενο στάδιο είναι να κατασκευαστεί η κωδική λέξη που θα εκπεμφθεί από τον πομπό. Αφού τα δύο δυαδικά σύμβολα κατασκευαστούν, το σύμβολο $\mathbf{s}_2$ πολ/ζεται με έναν πίνακα Ω κυκλικής μετακίνησης ο οποίος

κατεβάζει τα στοιχεία του $\mathbf{s}_2$ κατά μία θέση, κυκλικά. Η εκπομπή χωρίζεται ανά κεραία και ανά time-slot.

2.3.2 ΠΑΡΑΓΩΓΗ ΣΥΜΒΟΛΩΝ

Χάριν επεξήγησης θα υποτεθεί η απλούστερη διαμόρφωση 4FSK, δηλαδή $M = 4$. Άρα τα $\mathbf{s}_1$ και $\mathbf{s}_2$ είναι δύο δυαδικοί 1×4 πίνακες, ένας για κάθε κεραία εκπομπής. Από τα $2\log_2 M = 4$ bits του πίνακα της πληροφορίας τα πρώτα 2 υποδηλώνουν ποια θέση, έστω m, του 4×1 πίνακα θα «ενεργοποιηθεί» δηλαδή που θα υπάρχει το μη-μηδενικό υποφέρον. Ο Ω όπως εξηγήθηκε είναι πίνακας που κάνει κύλιση προς τα κάτω τα στοιχεία του $\mathbf{s}_2$. Έτσι προκύπτει μια νέα μη μηδενική θέση στον αντίστοιχο πίνακα, έστω n.

Η χρήση του πίνακα Ω είναι ζωτικής σημασίας για την αποδιαμόρφωση των συμβόλων. Όπως θα εξηγηθεί παρακάτω για τον υπολογισμό της απόφασης του δέκτη χρειάζονται δύο όροι Ε και δ. Εάν ο Ω δεν υπήρχε πολλές αποφάσεις για τα m (δείκτης-εκτίμηση του 1ου συμβόλου) και n (δείκτης-εκτίμηση του 2ου συμβόλου) θα είχαν τα ίδια μέτρα σύγκρισης κι έτσι πολλές από τις αποφάσεις του δέκτη θα είναι λάθος. Με την χρήση του Ω και άρα με την κύλιση των στοιχείων του $2^{ου}$ συμβόλου προς τα κάτω αυτή η πιθανότητα εξαλείφεται, εξασφαλίζοντας ότι τα m και n έχουν διαφορετικά μέτρα σύγκρισης [1].

2.3.3 ΣΧΗΜΑΤΙΣΜΟΣ ΚΩΔΙΚΗΣ ΛΕΞΗΣ ΣΥΜΒΟΛΩΝ

Η κωδική λέξη χωρίζεται σε δύο υποσύνολα $\mathbf{X}_1$ και $\mathbf{X}_2$ τα οποία υποδηλώνουν το time-slot και με την σειρά τους αυτά αποτελούνται από τα $\mathbf{s}_1$ και $\mathbf{s}_2$. Τα $\mathbf{s}_1$ και $\mathbf{s}_2$ είναι οι συχνότητες που εκπέμπονται από τις δύο κεραίες ανάλογα του time-slot. Πιο συγκεκριμένα, στο $1^{ο}$ time-slot εκπέμπεται το $\mathbf{s}_1$ από την πρώτη κεραία και το $\mathbf{s}_2$ από την δεύτερη κεραία. Στο $2^{ο}$ time-slot εκπέμπεται το $\Omega\mathbf{s}_2$ (που θα αναλυθεί παρακάτω) από την πρώτη κεραία και το $\mathbf{s}_1$ από τη δεύτερη κεραία. Στην ουσία τα μη-μηδενικά bits των $\mathbf{s}_1$ και $\mathbf{s}_2$ ενεργοποιούν συχνότητες-υποφέροντα [1]:

$$\left.\begin{array}{l} M = 4 \rightarrow bits = 2\log_2 M = 4 \\ b = \begin{bmatrix} b1 & b2 & b3 & b4 \end{bmatrix} \end{array}\right\} \mathbf{s}_1 = \begin{bmatrix} s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{1,4} \end{bmatrix} \mathbf{s}_2 = \begin{bmatrix} s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{2,4} \end{bmatrix}$$

όπου ο πρώτος δείκτης στο $\mathbf{s}_{xy}$ αφορά το προς εκπομπή σύμβολο και ο δεύτερος δείκτης τον αύξοντα αριθμό των υποφερόντων, από τα οποία μόνο το ένα είναι ενεργό. Οπότε το $\mathbf{X}_1$ το οποίο υποδηλώνει το $1^{ο}$ time-slot θα είναι: $X_1 = [s_{1,1} \quad s_{1,2} \quad s_{1,3} \quad s_{1,4} \quad s_{2,1} \quad s_{2,2} \quad s_{2,3} \quad s_{2,4}]^T$

όπου τα πρώτα 4 στοιχεία του εκπέμπονται από την πρώτη κεραία και τα 4 επόμενα από τη δεύτερη κεραία. Για το $\mathbf{X}_2$ χρησιμοποιείται το γινόμενο του πίνακα $\mathbf{s}_2$ με τον πίνακα Ω:

$$\Omega\mathbf{s}_2 = \begin{bmatrix} \mathbf{0}_{1x3} & 1 \\ \mathbf{I}_{3x3} & \mathbf{0}_{3x1} \end{bmatrix} \cdot \begin{bmatrix} s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{2,4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{2,4} \end{bmatrix} = \begin{bmatrix} s_{2,4} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \end{bmatrix}$$

Άρα το $\mathbf{X}_2$ που υποδηλώνει το 2ο time-slot θα είναι $X_2 = \begin{bmatrix} \Omega s_2 & s_1 \end{bmatrix}^T$. Αναλυτικότερα:

$X_2 = [s_{2,4} \quad s_{2,1} \quad s_{2,2} \quad s_{2,3} \quad s_{1,1} \quad s_{1,2} \quad s_{1,3} \quad s_{1,4}]^T$ όπου και πάλι τα πρώτα 4 στοιχεία του εκπέμπονται από την πρώτη κεραία και τα 4 επόμενα από τη δεύτερη κεραία.

Άρα η συνολική κωδική λέξη **X** θα γίνει :

$$\mathbf{X} = \begin{bmatrix} \mathbf{s}_1 & \Omega\mathbf{s}_2 \\ \mathbf{s}_2 & \mathbf{s}_1 \end{bmatrix} \triangleq [\mathbf{X}_1 \quad \mathbf{X}_2] = \begin{bmatrix} s_{1,1} & s_{2,4} \\ s_{1,2} & s_{2,1} \\ s_{1,3} & s_{2,2} \\ s_{1,4} & s_{2,3} \\ s_{2,1} & s_{1,1} \\ s_{2,2} & s_{1,2} \\ s_{2,3} & s_{1,3} \\ s_{2,4} & s_{1,4} \end{bmatrix}$$

Αφού η κωδική λέξη κατασκευαστεί, εφαρμόζεται σ' αυτήν ανάστροφος μετασχηματισμός Fourier (IFFT) και μετατρέπεται σε σύμβολο OFDM. Οι δυο στήλες συμβόλων $\mathbf{s}_1$ και $\mathbf{s}_2$ λαμβάνονται σε δύο χρονικές στιγμές (time-slots) $\mathbf{Y}_t$ για t = 1,2 από τις δύο λαμβανόμενες κεραίες. Ο $\mathbf{H}_t$ είναι ένας πίνακας πινάκων στοιχείων *h*, ο οποίος εκφράζει το gain του MIMO καναλιού για κάθε συνδυασμό κεραιών Δέκτης(Rx)-Πομπός (Tx) ,οι οποίες φαίνονται στον πάνω δείκτη $h_{t,sc}^{(i,j)}$ (ο *i* αφορά την Rx, ο *j* την Tx). Ο κάτω δείκτης προσδιορίζει το time-slot και το υποφέρον του STF κώδικα $\mathbf{H}_{t,sc}$ (2x2 πίνακας , κάθε στοιχείο δημιουργεί έναν διαγώνιο πίνακα 4x4.) και $\mathbf{X}_{t,sc}$. Ο $\mathbf{X}_t$ πίνακας στην ουσία είναι η κωδική λέξη του STF κώδικα. Όπου $\sqrt{E_s}$ η ενέργεια συμβόλου.

$$\mathbf{Y}_t = \begin{bmatrix} \mathbf{y}_t^{(1)} \\ \mathbf{y}_t^{(2)} \end{bmatrix} = \sqrt{E_s} \cdot \mathbf{H}_t \mathbf{X}_t + \mathbf{W}_t \rightarrow$$

$$\mathbf{Y}_1 = \begin{bmatrix} y_1^{(1)} \\ y_1^{(2)} \end{bmatrix} = \sqrt{E_s} \cdot \begin{bmatrix} h_{1,1}^{(11)} & 0 & 0 & 0 & h_{1,1}^{(12)} & 0 & 0 & 0 \\ 0 & h_{1,2}^{(11)} & 0 & 0 & 0 & h_{1,2}^{(12)} & 0 & 0 \\ 0 & 0 & h_{1,3}^{(11)} & 0 & 0 & 0 & h_{1,3}^{(12)} & 0 \\ 0 & 0 & 0 & h_{1,4}^{(11)} & 0 & 0 & 0 & h_{1,4}^{(12)} \\ h_{1,1}^{(21)} & 0 & 0 & 0 & h_{1,1}^{(22)} & 0 & 0 & 0 \\ 0 & h_{1,2}^{(21)} & 0 & 0 & 0 & h_{1,2}^{(22)} & 0 & 0 \\ 0 & 0 & h_{1,3}^{(21)} & 0 & 0 & 0 & h_{1,3}^{(22)} & 0 \\ 0 & 0 & 0 & h_{1,4}^{(21)} & 0 & 0 & 0 & h_{1,4}^{(22)} \end{bmatrix} \cdot \begin{bmatrix} s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{1,4} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{2,4} \end{bmatrix} + \mathbf{W} =$$

$$= \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(11)} s_{1,1} + \sqrt{E_s} h_{1,1}^{(12)} s_{2,1} + w_{1,1} \\ \sqrt{E_s} h_{1,2}^{(11)} s_{1,2} + \sqrt{E_s} h_{1,2}^{(12)} s_{2,2} + w_{1,2} \\ \sqrt{E_s} h_{1,3}^{(11)} s_{1,3} + \sqrt{E_s} h_{1,3}^{(12)} s_{2,3} + w_{1,3} \\ \sqrt{E_s} h_{1,4}^{(11)} s_{1,4} + \sqrt{E_s} h_{1,4}^{(12)} s_{2,4} + w_{1,4} \\ \sqrt{E_s} h_{1,1}^{(21)} s_{1,1} + \sqrt{E_s} h_{1,1}^{(22)} s_{2,1} + w_{1,5} \\ \sqrt{E_s} h_{1,2}^{(21)} s_{1,2} + \sqrt{E_s} h_{1,2}^{(22)} s_{2,2} + w_{1,6} \\ \sqrt{E_s} h_{1,3}^{(21)} s_{1,3} + \sqrt{E_s} h_{1,3}^{(22)} s_{2,3} + w_{1,7} \\ \sqrt{E_s} h_{1,4}^{(21)} s_{1,4} + \sqrt{E_s} h_{1,4}^{(22)} s_{2,4} + w_{1,8} \end{bmatrix} \Rightarrow$$

Αν γίνει διαχωρισμός ανά time-slot και ανά κεραία, τότε το λαμβανόμενο σήμα θα μοιάζει κάπως έτσι για το $1^{ο}$ time-slot:

Rx = 1

$$y_1^{(1)} = \sqrt{E_s} \cdot \begin{bmatrix} h_{1,1}^{(11)} & 0 & 0 & 0 & h_{1,1}^{(12)} & 0 & 0 & 0 \\ 0 & h_{1,2}^{(11)} & 0 & 0 & 0 & h_{1,2}^{(12)} & 0 & 0 \\ 0 & 0 & h_{1,3}^{(11)} & 0 & 0 & 0 & h_{1,3}^{(12)} & 0 \\ 0 & 0 & 0 & h_{1,4}^{(11)} & 0 & 0 & 0 & h_{1,4}^{(12)} \end{bmatrix} \cdot \begin{bmatrix} s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{1,4} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{2,4} \end{bmatrix} + \mathbf{W} =$$

$$\mathbf{y}_1^{(1)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(11)} s_{1,1} + \sqrt{E_s} h_{1,1}^{(12)} s_{2,1} + w_{1,1} \\ \sqrt{E_s} h_{1,2}^{(11)} s_{1,2} + \sqrt{E_s} h_{1,2}^{(12)} s_{2,2} + w_{1,2} \\ \sqrt{E_s} h_{1,3}^{(11)} s_{1,3} + \sqrt{E_s} h_{1,3}^{(12)} s_{2,3} + w_{1,3} \\ \sqrt{E_s} h_{1,4}^{(11)} s_{1,4} + \sqrt{E_s} h_{1,4}^{(12)} s_{2,4} + w_{1,4} \end{bmatrix}$$

Rx = 2

$$y_1^{(2)} = \sqrt{E_s} \cdot \begin{bmatrix} h_{1,1}^{(21)} & 0 & 0 & 0 & h_{1,1}^{(22)} & 0 & 0 & 0 \\ 0 & h_{1,2}^{(21)} & 0 & 0 & 0 & h_{1,2}^{(22)} & 0 & 0 \\ 0 & 0 & h_{1,3}^{(21)} & 0 & 0 & 0 & h_{1,3}^{(22)} & 0 \\ 0 & 0 & 0 & h_{1,4}^{(21)} & 0 & 0 & 0 & h_{1,4}^{(22)} \end{bmatrix} \cdot \begin{bmatrix} s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{1,4} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{2,4} \end{bmatrix} + \mathbf{W} =$$

$$\mathbf{y}_1^{(2)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(21)} s_{1,1} + \sqrt{E_s} h_{1,1}^{(22)} s_{2,1} + w_{1,5} \\ \sqrt{E_s} h_{1,2}^{(21)} s_{1,2} + \sqrt{E_s} h_{1,2}^{(22)} s_{2,2} + w_{1,6} \\ \sqrt{E_s} h_{1,3}^{(21)} s_{1,3} + \sqrt{E_s} h_{1,3}^{(22)} s_{2,3} + w_{1,7} \\ \sqrt{E_s} h_{1,4}^{(21)} s_{1,4} + \sqrt{E_s} h_{1,4}^{(22)} s_{2,4} + w_{1,8} \end{bmatrix}$$

Για το 2^{o} time-slot :

Rx = 1

$$y_2^{(1)} = \sqrt{E_s} \cdot \begin{bmatrix} h_{2,1}^{(11)} & 0 & 0 & 0 & h_{2,1}^{(12)} & 0 & 0 & 0 \\ 0 & h_{2,2}^{(11)} & 0 & 0 & 0 & h_{2,2}^{(12)} & 0 & 0 \\ 0 & 0 & h_{2,3}^{(11)} & 0 & 0 & 0 & h_{2,3}^{(12)} & 0 \\ 0 & 0 & 0 & h_{2,4}^{(11)} & 0 & 0 & 0 & h_{2,4}^{(12)} \end{bmatrix} \cdot \begin{bmatrix} s_{2,4} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{1,4} \end{bmatrix} + \mathbf{W} =$$

$$\mathbf{y}_2^{(1)} = \begin{bmatrix} \sqrt{E_s} h_{2,1}^{(11)} s_{2,4} + \sqrt{E_s} h_{2,1}^{(12)} s_{1,1} + w_{2,1} \\ \sqrt{E_s} h_{2,2}^{(11)} s_{2,1} + \sqrt{E_s} h_{2,2}^{(12)} s_{1,2} + w_{2,2} \\ \sqrt{E_s} h_{2,3}^{(11)} s_{2,2} + \sqrt{E_s} h_{2,3}^{(12)} s_{1,3} + w_{2,3} \\ \sqrt{E_s} h_{2,4}^{(11)} s_{2,3} + \sqrt{E_s} h_{2,4}^{(12)} s_{1,4} + w_{2,4} \end{bmatrix}$$

Rx = 2

$$\mathbf{y}_2^{(2)} = \sqrt{E_s} \cdot \begin{bmatrix} h_{2,1}^{(21)} & 0 & 0 & 0 & h_{2,1}^{(22)} & 0 & 0 & 0 \\ 0 & h_{2,2}^{(21)} & 0 & 0 & 0 & h_{2,2}^{(22)} & 0 & 0 \\ 0 & 0 & h_{2,3}^{(21)} & 0 & 0 & 0 & h_{2,3}^{(22)} & 0 \\ 0 & 0 & 0 & h_{2,4}^{(21)} & 0 & 0 & 0 & h_{2,4}^{(22)} \end{bmatrix} \cdot \begin{bmatrix} s_{2,4} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{1,4} \end{bmatrix} + \mathbf{W} =$$

$$\mathbf{y}_2^{(2)} = \begin{bmatrix} \sqrt{E_s} h_{2,1}^{(21)} s_{2,4} + \sqrt{E_s} h_{2,1}^{(22)} s_{1,1} + w_{2,5} \\ \sqrt{E_s} h_{2,2}^{(21)} s_{2,1} + \sqrt{E_s} h_{2,2}^{(22)} s_{1,2} + w_{2,6} \\ \sqrt{E_s} h_{2,3}^{(21)} s_{2,2} + \sqrt{E_s} h_{2,3}^{(22)} s_{1,3} + w_{2,7} \\ \sqrt{E_s} h_{2,4}^{(21)} s_{2,3} + \sqrt{E_s} h_{2,4}^{(22)} s_{1,4} + w_{2,8} \end{bmatrix}$$

Συλλογικά οι πίνακες για κάθε time-slot και για κάθε κεραία λήψης:

$$\mathbf{y}_1^{(1)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(11)} s_{1,1} + \sqrt{E_s} h_{1,1}^{(12)} s_{2,1} + w_{1,1} \\ \sqrt{E_s} h_{1,2}^{(11)} s_{1,2} + \sqrt{E_s} h_{1,2}^{(12)} s_{2,2} + w_{1,2} \\ \sqrt{E_s} h_{1,3}^{(11)} s_{1,3} + \sqrt{E_s} h_{1,3}^{(12)} s_{2,3} + w_{1,3} \\ \sqrt{E_s} h_{1,4}^{(11)} s_{1,4} + \sqrt{E_s} h_{1,4}^{(12)} s_{2,4} + w_{1,4} \end{bmatrix} \mathbf{y}_2^{(1)} = \begin{bmatrix} \sqrt{E_s} h_{2,1}^{(11)} s_{2,4} + \sqrt{E_s} h_{2,1}^{(12)} s_{1,1} + w_{2,1} \\ \sqrt{E_s} h_{2,2}^{(11)} s_{2,1} + \sqrt{E_s} h_{2,2}^{(12)} s_{1,2} + w_{2,2} \\ \sqrt{E_s} h_{2,3}^{(11)} s_{2,2} + \sqrt{E_s} h_{2,3}^{(12)} s_{1,3} + w_{2,3} \\ \sqrt{E_s} h_{2,4}^{(11)} s_{2,3} + \sqrt{E_s} h_{2,4}^{(12)} s_{1,4} + w_{2,4} \end{bmatrix}$$

$$\mathbf{y}_1^{(2)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(21)} s_{1,1} + \sqrt{E_s} h_{1,1}^{(22)} s_{2,1} + w_{1,5} \\ \sqrt{E_s} h_{1,2}^{(21)} s_{1,2} + \sqrt{E_s} h_{1,2}^{(22)} s_{2,2} + w_{1,6} \\ \sqrt{E_s} h_{1,3}^{(21)} s_{1,3} + \sqrt{E_s} h_{1,3}^{(22)} s_{2,3} + w_{1,7} \\ \sqrt{E_s} h_{1,4}^{(21)} s_{1,4} + \sqrt{E_s} h_{1,4}^{(22)} s_{2,4} + w_{1,8} \end{bmatrix} \mathbf{y}_2^{(2)} = \begin{bmatrix} \sqrt{E_s} h_{2,1}^{(21)} s_{2,4} + \sqrt{E_s} h_{2,1}^{(22)} s_{1,1} + w_{2,5} \\ \sqrt{E_s} h_{2,2}^{(21)} s_{2,1} + \sqrt{E_s} h_{2,2}^{(22)} s_{1,2} + w_{2,6} \\ \sqrt{E_s} h_{2,3}^{(21)} s_{2,2} + \sqrt{E_s} h_{2,3}^{(22)} s_{1,3} + w_{2,7} \\ \sqrt{E_s} h_{2,4}^{(21)} s_{2,3} + \sqrt{E_s} h_{2,4}^{(22)} s_{1,4} + w_{2,8} \end{bmatrix}$$

Παράδειγμα με bits πληροφορίας :

$$\left.\begin{array}{l} M = 4 \rightarrow bits = 2\log_2 M = 4 \\ b = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \end{array}\right\} \mathbf{s} = \begin{bmatrix} s_{1,1} \\ s_{1,2} \\ s_{1,3} \\ s_{1,4} \\ s_{2,1} \\ s_{2,2} \\ s_{2,3} \\ s_{2,4} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{και } \mathbf{X} = \begin{bmatrix} \mathbf{s}_1 & \Omega\mathbf{s}_2 \\ \mathbf{s}_2 & \mathbf{s}_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

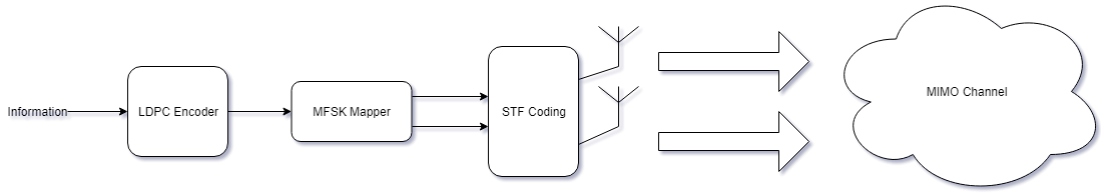
και για κάθε κεραία και time-slot ξεχωριστά :

$$y_1^{(1)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(11)} + w_{1,1} \\ w_{1,2} \\ w_{1,3} \\ \sqrt{E_s} h_{1,4}^{(12)} + w_{1,4} \end{bmatrix} \quad y_2^{(1)} = \begin{bmatrix} w_{2,1} \\ \sqrt{E_s} h_{2,2}^{(11)} + w_{2,2} \\ w_{2,3} \\ \sqrt{E_s} h_{2,4}^{(12)} + w_{2,4} \end{bmatrix}$$

$$y_1^{(2)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(21)} + w_{1,5} \\ w_{1,6} \\ w_{1,7} \\ \sqrt{E_s} h_{1,4}^{(22)} + w_{1,8} \end{bmatrix} \quad y_2^{(2)} = \begin{bmatrix} w_{2,5} \\ \sqrt{E_s} h_{2,2}^{(21)} + w_{2,6} \\ w_{2,7} \\ \sqrt{E_s} h_{2,4}^{(22)} + w_{2,8} \end{bmatrix}$$

Στο παραπάνω παράδειγμα παρατηρείται ότι σε κάθε time-slot οι κεραίες στέλνουν εναλλάξ τις ίδιες πληροφορίες. Δηλ. η $y^{(2)}$ στο 1^{o} time-slot στέλνει ότι και η $y^{(1)}$ στο 2^{o} time-slot, όπως και στον Alamouti ST [3]. Παρακάτω θα διαπιστωθεί πως αυτό δεν συμβαίνει σε όλες τις περιπτώσεις.

Παρακάτω απεικονίζεται ένα απλό μπλοκ διάγραμμα του πομπού. Η πληροφορία αφού κωδικοποιηθεί με LDPC διαμορφώνεται με MFSK διαμόρφωση. Στη συνέχεια τα MFSK σύμβολα πολυπλέκονται με OFDM και κωδικοποιούνται με STF κωδικοποίηση. Τέλος, το κωδικοποιημένο σήμα εκπέμπεται στο κανάλι.



Σχήμα 4: Το μπλοκ διάγραμμα του MIMOOFDM-MFSK πομπού.

2.4 ΒΕΛΤΙΣΤΟΣ ΑΣΥΜΦΩΝΟΣ ΔΕΚΤΗΣ

2.4.1 ΑΝΙΧΝΕΥΣΗ ΣΥΜΒΟΛΩΝ

Μόλις το λαμβανόμενο σήμα που αναλύθηκε παραπάνω ανιχνευθεί από το δέκτη, τότε διαχωρίζεται σε 3 διαφορετικές περιπτώσεις σύμφωνα με τους όρους: Θέτονται 2 αριθμοί-δείκτες *m* και *n* οι οποίοι δείχνουν για κάθε πίνακα s σε ποια θέση βρίσκεται το μη-μηδενικό στοιχείο. Επιπλέον υπάρχει ένας δείκτης *q* ο οποίος χρησιμοποιείται στη δημιουργία των κατανομών, όπου $q = (n+1) \bmod M$ όπου mod είναι το υπόλοιπο της ακέραιας διαίρεσης modulo με *M*. Για την καλύτερη επεξήγηση των περιπτώσεων, θα χρησιμοποιηθεί για κάθε περίπτωση παράδειγμα για συγκεκριμένα *m* και *n*.

Οπότε για την πρώτη περίπτωση :

- Περίπτωση 1: $m \neq n, q$. Κάθε πίνακας *y* για κάθε κεραία λήψης και time-slot έχει από δύο ενεργές συχνότητες, ενώ τα υπόλοιπα στοιχεία εμπεριέχουν μόνο θόρυβο. π.χ.

  όπως στο αρχικό παράδειγμα με *M*=4. *Για m=3 και n=0 , q = (0+1)*mod*4 = 1* -> Ενεργές θέσεις $4^{η}$ και $1^{η}$ (η αρίθμηση ξεκινάει από το 0) Οι πίνακες για κάθε κεραία και time-slot:

$$y_1^{(1)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(11)} + w_{1,1} \\ w_{1,2} \\ w_{1,3} \\ \sqrt{E_s} h_{1,4}^{(12)} + w_{1,4} \end{bmatrix} \quad y_2^{(1)} = \begin{bmatrix} w_{2,1} \\ \sqrt{E_s} h_{2,2}^{(11)} + w_{2,2} \\ w_{2,3} \\ \sqrt{E_s} h_{2,4}^{(12)} + w_{2,4} \end{bmatrix}$$

$$y_1^{(2)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(21)} + w_{1,5} \\ w_{1,6} \\ w_{1,7} \\ \sqrt{E_s} h_{1,4}^{(22)} + w_{1,8} \end{bmatrix} \quad y_2^{(2)} = \begin{bmatrix} w_{2,5} \\ \sqrt{E_s} h_{2,2}^{(21)} + w_{2,6} \\ w_{2,7} \\ \sqrt{E_s} h_{2,4}^{(22)} + w_{2,8} \end{bmatrix}$$

Για τη δεύτερη περίπτωση :

- Περίπτωση 2: $m = n$. Σ' αυτήν την περίπτωση, στο $1^{ο}$ time-slot υπάρχει μόνο ένα μη-μηδενικό στοιχείο σε κάθε πίνακα για τις 2 κεραίες. Στο $2^{ο}$ time-slot ισχύει ότι και στην πρώτη περίπτωση.

π.χ.

Για *m=n=2* ->Ενεργές θέσεις $3^{η}$ και $3^{η}$.Οι πίνακες για κάθε κεραία και time-slot:

$$y_1^{(1)} = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ \sqrt{E_s} h_{1,3}^{(11)(12)} + w_{1,3} \\ w_{1,4} \end{bmatrix} \quad y_2^{(1)} = \begin{bmatrix} w_{2,1} \\ w_{2,2} \\ \sqrt{E_s} h_{2,3}^{(12)} + w_{2,3} \\ \sqrt{E_s} h_{2,4}^{(11)} + w_{2,4} \end{bmatrix}$$

$$y_1^{(2)} = \begin{bmatrix} w_{1,5} \\ w_{1,6} \\ \sqrt{E_s} h_{1,3}^{(21)(22)} + w_{1,7} \\ w_{1,8} \end{bmatrix} \quad y_2^{(2)} = \begin{bmatrix} w_{2,5} \\ w_{2,6} \\ \sqrt{E_s} h_{2,3}^{(22)} + w_{2,7} \\ \sqrt{E_s} h_{2,4}^{(21)} + w_{2,8} \end{bmatrix}$$

Για την τρίτη περίπτωση:

- Περίπτωση 3: $m = q$ Είναι ακριβώς η ανάστροφη της $2^{ης}$ περίπτωσης. Στο $2^{ο}$ time-slot εμπεριέχεται ένα μη-μηδενικό στοιχείο για κάθε πίνακα, ενώ στο $1^{ο}$ time-slot εμπεριέχονται δύο μη-μηδενικά στοιχεία.

π.χ.

$m=q=(0+1) \bmod 4=1n=0$

$$y_1^{(1)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(12)} + w_{1,1} \\ \sqrt{E_s} h_{1,2}^{(11)} + w_{1,2} \\ w_{1,3} \\ w_{1,4} \end{bmatrix} \quad y_1^{(2)} = \begin{bmatrix} \sqrt{E_s} h_{1,1}^{(22)} + w_{1,5} \\ \sqrt{E_s} h_{1,2}^{(21)} + w_{1,6} \\ w_{1,7} \\ w_{1,8} \end{bmatrix}$$

$$y_2^{(1)} = \begin{bmatrix} w_{2,1} \\ \sqrt{E_s} h_{2,2}^{(22)(21)} + w_{2,2} \\ w_{2,3} \\ w_{2,4} \end{bmatrix} \quad y_2^{(2)} = \begin{bmatrix} w_{2,5} \\ \sqrt{E_s} h_{2,2}^{(21)(22)} + w_{2,6} \\ w_{2,7} \\ w_{2,8} \end{bmatrix}$$

Αφού κατανεμηθούν οι αποφάσεις του δέκτη στις σωστές περιπτώσεις, τότε ξεκινάει η διαδικασία της αποδιαμόρφωσης: Αυτό που κάνει ο δέκτης είναι να βλέπει τις λαμβανόμενες τιμές και να εκτιμάει με βάση τις πιθανότητες ποιο subcarrier εστάλη από τον πομπό. Η γενική μορφή της εξίσωσης εκφράζει την συνολική πιθανότητα για όλες τις κεραίες και time-slots το σήμα να είναι το *y* υπό συνθήκη X και είναι: $\prod_{t=1,2} \prod_{k=1,2} p(y_t^{(k)} \mid X_t)$. Η εξίσωση απλοποιείται σε δύο όρους για κάθε περίπτωση οι οποίες υπολογίζουν τη συνολική πιθανότητα του σήματος. Οι όροι υπολογίζονται με βάση τη σηματοθορυβική σχέση και τα διάφορα στοιχεία *y* που έχουν λάβει οι κεραίες. Συγκεκριμένα οι όροι που υπολογίζονται είναι το E και το *δ*. Στην τελική εξίσωση υπάρχει και ένας επιπλέον όρος *C* αλλά επειδή υπολογίζεται για όλες τις περιπτώσεις και είναι ο ίδιος απλοποιείται και δεν υπάρχει νόημα να επεξηγηθεί.

Περίπτωση 1

$$\mathrm{E} = \frac{E_s}{N_0} \sum_{k=1}^{2} \frac{| y_{1,m}^{(k)} |^2 + | y_{1,n}^{(k)} |^2 + | y_{2,q}^{(k)} |^2 + | y_{2,m}^{(k)} |^2}{E_s + N_0}$$

$$\delta = \ln\left(\frac{E_s}{N_0}+1\right)^8$$

Περίπτωση 2

$$\mathrm{E} = \frac{E_s}{N_0}\sum_{k=1}^{2}\left(\frac{2\,|\,y_{1,m}^{(k)}\,|^2}{2E_s+N_0}+\frac{|\,y_{2,q}^{(k)}\,|^2+|\,y_{2,m}^{(k)}\,|^2}{E_s+N_0}\right)$$

$$\delta = \ln\left(\frac{2E_s}{N_0}+1\right)^2\left(\frac{E_s}{N_0}+1\right)^4$$

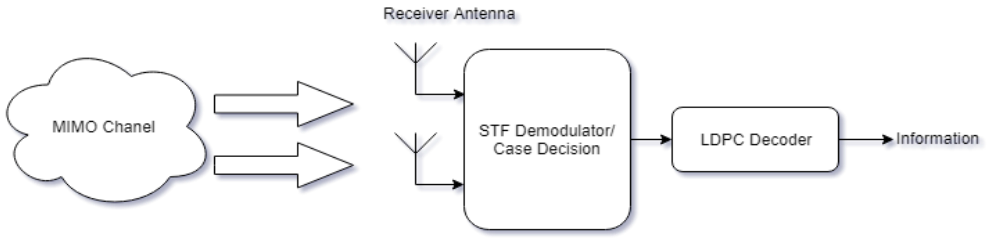
Περίπτωση 3

$$\mathrm{E} = \frac{E_s}{N_0}\sum_{k=1}^{2}\left(\frac{2\,|\,y_{2,m}^{(k)}\,|^2}{2E_s+N_0}+\frac{|\,y_{1,m}^{(k)}\,|^2+|\,y_{1,n}^{(k)}\,|^2}{E_s+N_0}\right)$$

$$\delta = \ln\left(\frac{2E_s}{N_0}+1\right)^2\left(\frac{E_s}{N_0}+1\right)^4$$

Από τις τρεις περιπτώσεις παρατηρείται πως το δ μπορεί να απλοποιηθεί κι έτσι για οπτική διευκόλυνση μπορεί να οριστεί ως : $\delta = 4\ln\left(\frac{E_S}{N_0}+1\right)$για $m\neq n,q$ και $\delta = 2\ln\left(\frac{2E_S}{N_0}+1\right)$,αλλού.

Μετά τον υπολογισμό των όρων E και δ η τελική απόφαση του δέκτη υπολογίζεται από τον όρο μέγιστης πιθανοφάνειας (ML) ο οποίος στη γενική του μορφή είναι : $(\hat{m},\hat{n}) = \underset{(m,n)}{\arg\max}\, p(Y\,|\,X)$.Η συνολική πιθανότητα $p(Y\,|\,X)$ μπορεί να γραφτεί ως το υπόλοιπο των δυο όρων E και δ, που στην ουσία είναι πίνακες και να βρεθούν τα μέγιστα στοιχεία αυτών $(\hat{m},\hat{n}) = \underset{(m,n)}{\arg\max}(\mathrm{E}-\delta)$. Ο μέγιστος όρος εκτιμάται ότι είναι πιθανώς η θέση του μη-μηδενικού στοιχείου από το σήμα που εκπέμφθηκε, οπότε και αποδιαμορφώνεται οδηγώντας στη λήψη των αρχικών bits [1].



Σχήμα 5: Μπλοκ διάγραμμα του MIMO OFDM-MFSK αποκωδικοποιητή

Παραπάνω, στο Σχήμα 4 απεικονίζεται η διαδικασία που ακολουθεί ο δέκτης. Γίνεται η λήψη του σήματος και στη συνέχεια στον αποδιαμορφωτή αφαιρείται το CP. Έπειτα, γίνεται αποδιαμόρφωση του σήματος με την επιλογή περιπτώσεων. Το αποδιαμορφωμένο πλέον σήμα, περνάει από έναν αποκωδικοποιητή που χρησιμοποιεί διάφορους αλγόριθμους αποκωδικοποίησης LDPC. Τέλος, στην έξοδο του αποκωδικοποιητή LDPC βρίσκεται η αρχική πληροφορία που έστειλε ο πομπός.

2.4.2 ΑΠΟΚΩΔΙΚΟΠΟΙΗΣΗ SOFT-ΠΛΗΡΟΦΟΡΙΑΣ

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, τα bits πληροφορίας πριν τη διαμόρφωσή τους και την κωδικοποίηση σε STF σύμβολα για την εκπομπή τους, κωδικοποιούνται με μια FEC τεχνική, την LDPC. Ο Decoder στην είσοδό του χρησιμοποιεί δυο ειδών πληροφορίας, την Hard και την Soft. Σ' αυτήν την εργασία εξετάστηκαν 2 αποκωδικοποιητές, ένας για κάθε περίπτωση.

Οι αποκωδικοποιητές θα αναλυθούν σε επόμενο κεφάλαιο. Όπως και να 'χει, μετά τις δοκιμές, παρατηρήθηκε πως η soft-bit πληροφορία έχει πολύ καλύτερα αποτελέσματα ως προς το ρυθμό σφάλματος bit. Η βιβλιογραφία επίσης επιβεβαιώνει πως ο αποκωδικοποιητής STF χρειάζεται στην είσοδό του soft-bit πληροφορία για να λειτουργήσει βέλτιστα. Η hard πληροφορία είναι στην ουσία, αυτούσια τα αποδιαμορφωμένα bits έτσι όπως εκτιμήθηκαν από τον ανιχνευτή, η οποία με διάφορους αλγόριθμους του αποκωδικοποιητή διορθώνεται και ενημερώνεται η τελική λαμβανόμενη πληροφορία. Η soft πληροφορία δείχνει μια πιθανοτική εκτίμηση (soft τιμή) για το αν από τη θέση που προήλθε το bit η τιμή του είναι 1 η 0. Μια soft πληροφορία είναι και το αποτέλεσμα του λόγου μεγίστης πιθανοφάνειας που αναφέρθηκε πιο πάνω.

Μια βελτίωση ως προς την περιπλοκότητα του λόγου μέγιστης πιθανοφάνειας είναι ο υπολογισμός των **Log-Likelihood-Ratios(LLR)**. Τα LLR ορίζονται ως $\Lambda(b_j) = \ln\left(\frac{\sum_{m,n:bj=1} p(Y \mid X)}{\sum_{m,n:bj=0} p(Y \mid X)}\right), j = 0,...,2\log_2 M - 1$. Λογαριθμίζοντας τον λόγο της συνολικής

πιθανότητας του λαμβανόμενου σήματος υπό συνθήκη της κωδικής λέξης *X* των bit να είναι 1 προς τη συνολικής πιθανότητας του λαμβανόμενου σήματος υπό συνθήκη της κωδικής λέξης *X* των bit να είναι 0 λαμβάνεται μια soft εκτίμηση της πληροφορίας, η οποία είναι απλούστερη ως προς τους υπολογισμούς που εκτελεί ο δέκτης, αφού πλέον οι πράξεις που θα πρέπει να κάνει είναι πρόσθεση και αφαίρεση, ενώ στη γενική περίπτωση ML θα πρέπει να γίνουν οι υπολογισμοί με γινόμενα. Ο δέκτης , αφού υπολογίζονται έτσι κι αλλιώς τα E και *δ,* μπορεί να τα εκμεταλλευτεί και να υπολογίσει τα LLR από τα υπόλοιπα των μεγίστων τιμών για όλες τις φυσικές δυαδικές θέσεις που περιέχουν 1 και για όλες τις θέσεις που περιέχουν 0 $\Lambda(b_j) = \max_{m,n:b_j=1}(\mathrm{E}-\delta) - \max_{m,n:b_j=0}(\mathrm{E}-\delta)$ [1].

2.5 LDPC

2.5.1 ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ ΚΩΔΙΚΕΣ LDPC

Οι LDPC κώδικες πρωτοδημοσιεύθηκαν σαν διδακτορική διατριβή το 1963 στην ακαδημαϊκή εφημερίδα του MIT σε μια μονογραφία του επιστήμονα Robert G. Gallager. Οι κώδικες που εισήγαγε ήταν πίνακες με δυαδικά στοιχεία τα οποία στην πλειοψηφία τους ήταν 0 με μερικούς διάσπαρτους άσσους σε συγκεκριμένα σημεία. Απέδειξε πως όσο μεγαλύτερος ο κώδικας LDPC σε σχέση με τη βασική πληροφορία, τόσο αποδοτικότερη διόρθωση λαθών θα γίνεται αλλά με την προσαύξηση των στοιχείων του πίνακα η πολυπλοκότητά του ως προς την αποκωδικοποίηση αυξάνεται ταυτοχρόνως. Εκείνη την εποχή είχε θεωρηθεί από την επιστημονική κοινότητα πως οι κώδικες αυτοί δεν μπορούν να εφαρμοστούν στην πράξη για τον παραπάνω λόγο. Το 1993 ανακαλύφθηκαν από ηλεκτρολόγους μηχανικούς οι turbo κώδικες τους οποίους η επιστημονική κοινότητα προσπάθησε να καταρρίψει. Τελικά η επιστημονική κοινότητα όχι μόνο παρατήρησε πως λειτουργούν αλλά επίσης βρήκαν ξανά τους LDPC κώδικες οι οποίοι ήταν αποδοτικότεροι από τους turbo, μόνο που είχαν ανακαλυφθεί τη δεκαετία του ’60. Τελικά ο επιστήμονας D.J.C. MacKay και ο συνεργάτης του R.M. Neal έφεραν ξανά τους LDPC κώδικες στην επιφάνεια και πρότειναν έναν νέο αλγόριθμο αποκωδικοποίησης και απέδειξαν ότι οι LDPC όπως και οι turbo φτάνουν κοντά στο όριο Shannon. Επίσης απέδειξαν ότι οι LDPC και πιο συγκεκριμένα αυτοί με ρυθμούς ½ και $^2/_3$ αποδίδουν καλύτερα σε σχέση με άλλους κώδικες. Πλέον οι LDPC κώδικες χρησιμοποιούνται σε πολλά τηλεπικοινωνιακά πρότυπα ασύρματα αλλά και ενσύρματα. [12][13]

Οι block κώδικες LDPC (Low Density Parity Check) είναι μια μέθοδος διόρθωσης λαθών (ECC) η οποία ανήκει στην κατηγορία FEC (Forward Error Correction). Οι FEC κώδικες προσθέτουν επιπλέον bits κωδικοποίησης στην αρχική λέξη ανάλογα τον ρυθμό του κώδικα

rc = k/n όπου k = bits πληροφορίας και n = bits πληροφορίας + επιπρόσθετα bits ισοτιμίας. Για παράδειγμα, ένας κώδικας ½ με 408 bits, θα έχει μόνο 204 bits πληροφορίας, ενώ τα υπόλοιπα θα είναι επιπρόσθετα bits κώδικα για διόρθωση λαθών. Ένας κώδικας LDPC μπορεί να είναι συστηματικός η μη-συστηματικός.

Ένας συστηματικός LDPC κώδικας συνθέτει την κωδική λέξη απλά τοποθετώντας τα bits ισοτιμίας σε σειρά με την πληροφορία [3]. Εδώ να τονισθεί πως τα bits ισοτιμίας παράγονται με συγκεκριμένη τεχνική η οποία λαμβάνει υπ' όψιν της τη συγκεκριμένη πληροφορία που κωδικοποιούν και τον πίνακα ελέγχου ισοτιμίας LDPC ο οποίος είναι αυτός που χαρακτηρίζει ένα σχήμα κωδικοποίησης LDPC. Ένας LDPC πίνακας ισοτιμίας, είναι διαστάσεων KxN και αποτελείται κυρίως από μηδενικά και αραιούς άσσους, συνήθως με συγκεκριμένο αριθμό ανά στήλη ή ανά γραμμή. Έχει τρεις τρόπους αναπαράστασης, ένας εκ των οποίων είναι γραφικός. Μπορεί να αναπαρασταθεί με τον πίνακα ελέγχου ισοτιμίας ή αυτούσιος, με γράφο Tanner και με τις εξισώσεις ισοτιμίας του κώδικα. Και από τους τρεις τρόπους μπορεί κανείς να καταλάβει τα βασικά στοιχεία του κώδικα [6].

Παρακάτω ακολουθεί ένα παράδειγμα [5] κώδικα 9x12 έτσι ώστε να απεικονιστούν όλοι οι τρόποι με πιο κατανοητό τρόπο :

Πινάκας ισοτιμίας : Ο παρακάτω πίνακας ισοτιμίας έχει διαστάσεις 9x12, άρα έχει ρυθμό κώδικα rc = 0.75 . Κάθε γραμμή είναι και κωδική λέξη.

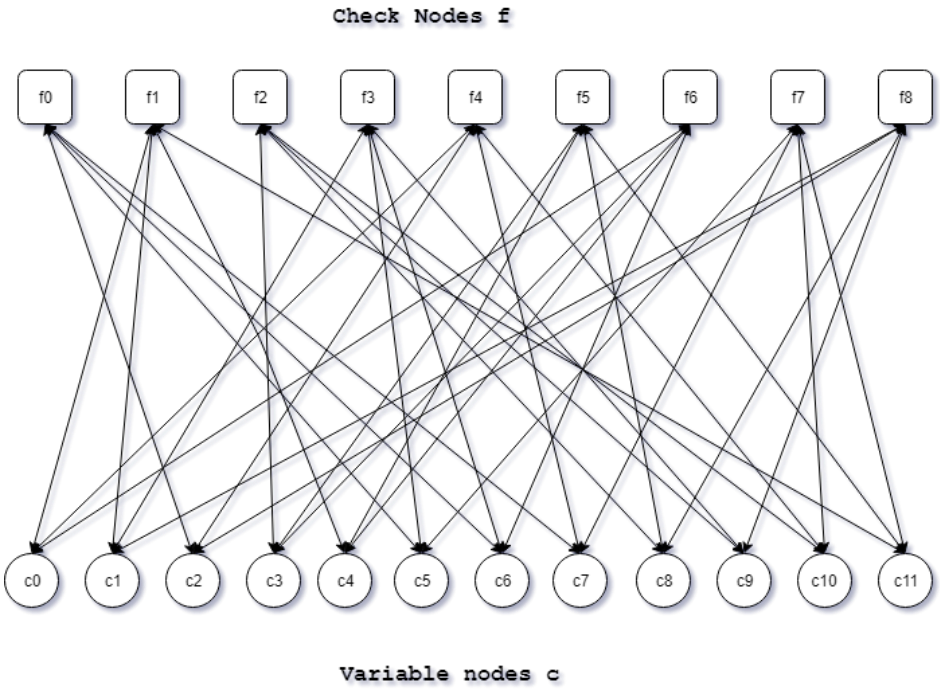
$$H = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

↕ *C*heck nodes *f*i

↔ *V*ariable nodes *c*

Ο πίνακας έχει βάρος στήλης τρεις άσσους και βάρος γραμμής τέσσερις άσσους. Παρατηρήθηκε από την προσομοίωση ότι το βάρος κάποιας από τις γραμμές ή τις στήλες μπορεί να διαφέρει ελάχιστα ανά περιπτώσεις.

Γράφος Tanner: Ο γράφος Tanner είναι μια γραφική αναπαράσταση ενός LDPC πίνακα ισοτιμίας Η. Αποτελείται από δυο κατηγορίες κόμβων: τα variable nodes *c* και τα check nodes *f*. Το πλήθος των variable nodes *c* υποδηλώνουν το μήκος της κωδικής λέξης που παράγει ο κώδικας, και σ' αυτά αντιστοιχίζονται τα bits μιας κωδικής λέξης. Τα check nodes *f* δείχνουν το πόσες διαφορετικές κωδικές λέξεις περιγράφονται σ' έναν πίνακα Η. Τα nodes *c* συνδέονται με τα nodes *f* μέσω γραμμών. Οι γραμμές υποδηλώνουν πόσους άσσους ανά γραμμή έχει και πόσους άσσους ανά στήλη έχει ο πίνακας Η. Με τον γράφο Tanner μπορεί κανείς να καταλάβει πως λειτουργεί ένας αλγόριθμος αποκωδικοποίησης. Σε επόμενο κεφάλαιο θα αναλυθεί συνοπτικά ένας τέτοιος αλγόριθμος.



Σχήμα 6: Ο Γράφος Tannertου παραδείγματος ρυθμού κώδικα 9/12

Εξισώσεις ισοτιμίας: Παρακάτω αναλύονται οι εξισώσεις ισοτιμίας του παραδείγματος. Για να ικανοποιείται ένα LDPC σύστημα, δηλαδή για να είναι σίγουρο πως ο κώδικας δεν έχει λάθος bits, χρησιμοποιεί τις εξισώσεις ισοτιμίας. Όταν όλες οι εξισώσεις ισοτιμίας ισούνται με 0, αυτό αυτομάτως σημαίνει πως ο κώδικας δεν έχει λάθος λέξεις. Οι εξισώσεις υπολογίζονται από το modulo 2 άθροισμα των Variable nodes *c* που είναι συνδεδεμένα σε ένα check node *f*.

$$f_0 = (c_2 + c_5 + c_6 + c_7)_2 = 0$$
$$f_1 = (c_0 + c_1 + c_4 + c_{11})_2 = 0$$
$$f_2 = (c_3 + c_8 + c_9 + c_{10})_2 = 0$$
$$f_3 = (c_1 + c_5 + c_6 + c_9)_2 = 0$$
$$f_4 = (c_0 + c_2 + c_7 + c_{10})_2 = 0$$
$$f_5 = (c_3 + c_4 + c_8 + c_{11})_2 = 0$$
$$f_6 = (c_0 + c_3 + c_4 + c_6)_2 = 0$$
$$f_7 = (c_5 + c_7 + c_{10} + c_{11})_2 = 0$$
$$f_8 = (c_1 + c_2 + c_8 + c_9)_2 = 0$$

2.5.2 SUM PRODUCT DECODING

Οι LDPC κώδικες χρησιμοποιούν διάφορους αλγόριθμους για έλεγχο λαθών και αποκωδικοποίηση. Είναι συνήθως επαναληπτικοί, δηλαδή κάνουν εκτίμηση των bit πληροφορίας σε πολλές επαναλήψεις για καλύτερα αποτελέσματα. Ο πιο γνωστός αλγόριθμος και αυτός που χρησιμοποιήθηκε για την υλοποίηση του συστήματος είναι ο Sum-Product Algorithm (SPA). Ο SPA είναι ένας Message Passing Algorithm όπως ονομάζεται, διότι μεταφέρει τα bits πληροφορίας μπρος και πίσω στα variable και check nodes εκτιμώντας κάθε φορά την προέλευση της αρχικής πληροφορίας. Μπορεί να υλοποιηθεί στο Probability domain και στο Log domain με το δεύτερο να είναι η βελτιωμένη εκδοχή του πρώτου ως προς την υλοποίηση και την πολυπλοκότητα των πράξεων.

Ξεκινώντας από την hard αποκωδικοποίηση bit flipping και για λόγους ευκολίας, θα εξηγηθεί πως λειτουργούν οι επαναλήψεις και η εκτίμηση της πληροφορίας. Η μόνη διαφορά της hard αποκωδικοποίησης με τη soft είναι πως τα μεγέθη στη δεύτερη αφορούν πιθανότητες άρα και πιο περίπλοκες πράξεις, ενώ στη πρώτη καθαρά bits. Ο αλγόριθμος εκτίμησης παραμένει ίδιος και για τις δύο περιπτώσεις. Μόλις ο δέκτης λάβει το σήμα, κάνει κάποια εκτίμηση της πληροφορίας και τη μετατρέπει σε bit. Τα bits εισέρχονται στα variable nodes, για κάθε bit αντιστοιχίζεται ένα variable node. Το επόμενο βήμα είναι τα bits να αποσταλούν στα συγκεκριμένα check nodes που συνδέονται συγκεκριμένα variable nodes και να ελεγχτούν οι εξισώσεις ισοτιμίας με modulo 2 πρόσθεση. Όπως εξηγήθηκε πιο πριν, για να ικανοποιείται η αποκωδικοποίηση και να μην υπάρχουν λάθη, θα πρέπει όλες οι εξισώσεις να ισούνται με 0. Αν τώρα κάποιο bit έχει παραλλαχθεί, πολύ πιθανόν να γίνει λάθος στην εξίσωση ισοτιμίας οπότε θα βγάλει αποτέλεσμα 1. Αφού βρεθεί το λάθος, αυτό που γίνεται είναι με διάφορους τρόπους να γίνει αναστροφή του λάθους bit. Ένας τρόπος είναι να βρεθεί με ποια κοινά variable nodes συνδέονται οι λάθος check nodes, έτσι ώστε αν υπάρχει π.χ. για

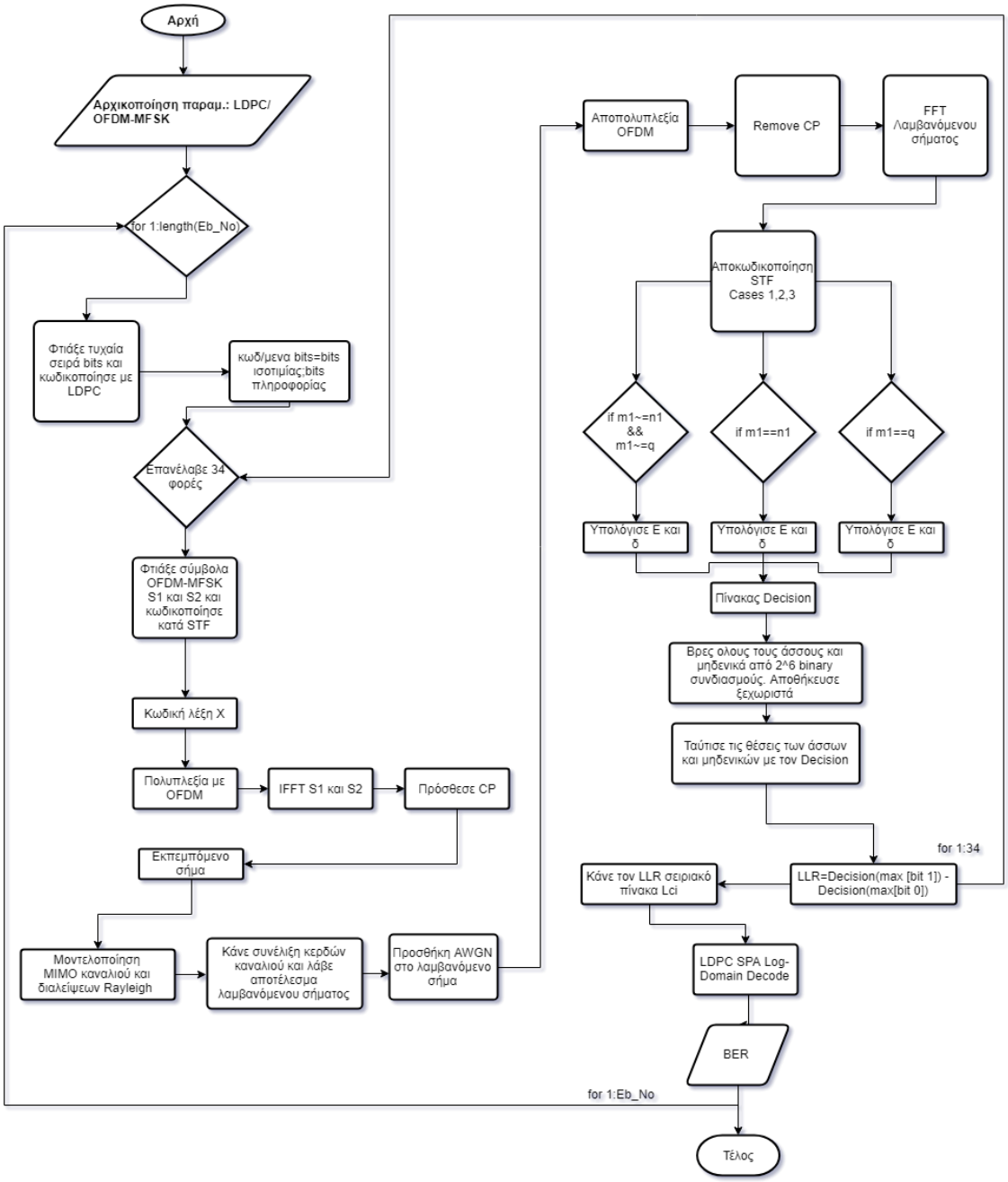
τα 2 check nodes από τα 9 μόνο 1 λάθος variable, τότε να γίνει η αναστροφή εκείνου του variable node [7]. Ένας άλλος τρόπος είναι, τα αποτελέσματα να γυρίσουν στα variable nodes και εκεί με πλειοψηφία από όλες τις εκτιμήσεις συν την αρχική πληροφορία να γίνει αναστροφή. Στη συνέχεια οι νέες διορθωμένες τιμές αποθηκεύονται στα variable nodes και αρχίζει εκ νέου επανάληψη του αλγορίθμου γι' αυτές τις νέες τιμές [8].

Με παρόμοιο τρόπο γίνεται και η soft αποκωδικοποίηση. Αυτό που διαφέρει είναι ο τρόπος που γίνονται οι πράξεις γιατί πλέον η πληροφορία εκτιμάται σε πιθανότητες η στις λογαριθμισμένες πιθανότητες. Ο αλγόριθμος που χρησιμοποιήθηκε είναι στο Log domain ο οποίος είναι καλύτερος σε απόδοση από τον Prob domain οπότε και η ανάλυση δεν θα γίνει για αυτόν. Ο Log domain decoder παίρνει σαν είσοδο τα υπολογισμένα από τον δέκτη LLR που εξηγήθηκαν σε προηγούμενο κεφάλαιο. Τώρα, τα variable nodes έχουν την εκτίμηση $L(ci) = LLR$ την οποία στέλνουν στα check nodes και εκεί με τρόπους οι οποίοι θα αναλυθούν σε επόμενη ενότητα υπολογίζουν το πρόσημο και το πλάτος των LLR. Με αυτά τα δεδομένα κατασκευάζονται μοντέλα συνολικών πιθανοτήτων για υπολογισμό των εκτιμήσεων των check nodes. Και αφού υπολογιστούν οι εκτιμήσεις αθροίζονται στα αρχικά $L(ci)$ για την εκ νέου εκτίμηση του σήματος. Η διεργασία επαναλαμβάνεται πολλές φορές και στο τέλος γίνεται η εκτίμηση των bits από τα πλέον καλά εκτιμημένα $L(ci)$. Σε επόμενο κεφάλαιο θα αναλυθεί ο αλγόριθμος υλοποίησης της μεθόδου Sum Product.

Κλείνοντας, να επισημανθεί πως υπάρχουν και αλγόριθμοι Log domain χαμηλότερης πολυπλοκότητας όπως ο Min Sum και ο Min Sum Correction άρα και μεγαλύτερης ταχύτητας οι οποίοι δεν επιλέχθηκαν γιατί δεν δίνουν τα βέλτιστα αποτελέσματα [6].

3 ΥΛΟΠΟΙΗΣΗ ΤΟΥ OFDM-MFSK

Η υλοποίηση του συστήματος σε αυτήν την εργασία έγινε με την βοήθεια του προγραμματιστικού περιβάλλοντος MATLAB. Με τη βοήθεια διαγραμμάτων ροής και την επισύναψη μικρών και κριτικών κομματιών από το αρχείο του κώδικα, θα επεξηγηθεί ο τρόπος υλοποίησης του συστήματος. Παρακάτω ακολουθεί το διάγραμμα ροής της προσομοίωσης. Είναι χωρισμένο οπτικά σε δύο μέρη. Το αριστερά μέρος αφορά τον πομπό με όλα του τα περιεχόμενα και το κανάλι, και το δεξιά μέρος αφορά τον δέκτη.



Σχήμα 7: Διάγραμμα ροής του βασικού προγράμματος

Στις επόμενες ενότητες ακολουθούν συνοπτικά οι τρόποι με τους οποίος έγινε η μοντελοποίηση του συστήματος. Για καλύτερη και αναλυτικότερη επεξήγηση, θα εισαχθούν μερικές **μεταβλητές-κλειδιά** από το πρόγραμμα προσομοίωσης :

Πομπός

`ParityMatrix`: Πίνακας ισοτιμίας του κώδικα LDPC.

`s1`: Σύμβολο 1 της κωδικής λέξης

s2: Σύμβολο 2 της κωδικής λέξης

InfoBits: Τυχαία δυαδική ακολουθία μήκους 204bits

ParityBits: Bits ισοτιμίας κωδικής λέξης

u: [ParityBits;InfoBits]

omega: Πίνακας κυκλικής μετακίνησης [zeros(1,M-1),1;eye(M-1),zeros(M-1,1)]

TxA(1,2)s(1,2): Εκπεμπόμενα σύμβολα με OFDM (όπου **A**=κεραία και **s**=timeslot)

Κανάλι

h(1,2)(1,2)_s(1,2): κέρδος καναλιού (πρώτος δείκτης=κεραία εκπομπής, δεύτερος δείκτης κεραία λήψης, s=timeslot)

RxA(1,2)s(1,2): Λαμβανόμενα σήματα (A=κεραία και s=timeslot)

Δέκτης

y(1,2)s(1,2):λαμβανόμενο σήμα αποπολυπλεγμένο από OFDM

Epsilon και **delta**: Μεταβλητές για εύρεση της κωδικής λέξης ανά case

decision():Epsilon-delta

LLR: Log-Likelihood Ratios

Lci: Όλα τα LLRσε μία γραμμή

vhat: Αποκωδικοποιημένα bits (Αρχική πληροφορία)

3.1 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΠΟΜΠΟΥ

Το πρόγραμμα ξεκινάει με αρχικοποίηση όλων των βασικών μεταβλητών για τον LDPC κώδικα όπως οι διαστάσεις αυτού και ο πίνακας ισοτιμίας (**ParityMatrix**), επίσης οι αρχικοποιήσεις των s1 και s2 καθώς και σημαντικές παράμετροι για τις συναρτήσεις που χρησιμοποιούνται στο κύριο πρόγραμμα. Το MFSK που επιλέχθηκε αρχικά ήταν για $M = 64$.

Για πίνακας ισοτιμίας 'H' δοκιμάστηκε πίνακας ρυθμού ½ μεγέθους (204x408) και με τρείς άσσους, από την ιστοσελίδα του MacKay [10]. Ο κώδικας που εφαρμόστηκε είναι συστηματικός. Βρέθηκε επίσης έτοιμη συνάρτηση η **makeLdpc()**[11] η οποία κατασκευάζει τον 'H', ανάλογα των ''1'' ανά στήλη που προτιμάται, η οποία δούλεψε εξίσου καλά στην κωδικοποίηση της λέξης και για καλύτερη ταχύτητα χρησιμοποιήθηκε αυτή. Το επόμενο βήμα είναι να κατασκευαστεί ο κωδικοποιητής. Χρησιμοποιήθηκε συνάρτηση από την παραπομπή [11] η **makeParityChk()** στην οποία εισάγονται τα **InfoBits**, ο **ParityMatrix** και επιλέγεται ένα από τα 3 strategies που έχει διαθέσιμη η συνάρτηση. Στην έξοδο της συνάρτησης δημιουργούνται τα μοναδικά για κάθε λέξη **ParityBits** και ένας νέος πίνακας **ParityMatrix**ο **newH**. Η μέθοδος για την κατασκευή των bits ισοτιμίας είναι

γνωστή ως LU Decomposition και ο αλγόριθμός της βρίσκεται στο [11]. Ο πίνακας **newH** χρησιμοποιείται αργότερα στον δέκτη για την αποκωδικοποίηση της κωδικοποιημένης λέξης. Ο **newH** έχει μεγάλο ενδιαφέρον να αναλυθεί περισσότερο γιατί στη βιβλιογραφία αναφέρεται πως για την αποκωδικοποίηση χρησιμοποιείται ο αρχικός πίνακας ισοτιμίας και τα bits ισοτιμίας.

Για να κατασκευαστεί ο **newH** γίνονται αλλαγές στις στήλες του H με συγκεκριμένο τρόπο. Επιλέγεται για την i στήλη του **newH** να βρεθεί και να τοποθετηθεί η πρώτη στήλη του H η οποία θα έχει άσσο στην j γραμμή. Παρακάτω αναλύεται ο αλγόριθμος του πίνακα:

```
Αλγόριθμος newH:

Βρες σε ποιές θέσεις του H υπάρχουν άσσοι. Στο r-> indices of rows c-> indices of columns.

Χρησιμοποίησε την minimum product στρατηγική. Βρες το βάρος των στηλών από i και μετά και αφαίρεσε 1 .

Βρες το βάρος της i γραμμής και αφαίρεσε 1.
```

Παρατήρηση: Παρόλο που στην δημιουργία του Parity Check Matrix με την συνάρτηση makeLdpc() έχει επιλεγεί να δημιουργείται πίνακας με τρείς άσσους ανά στήλη, μερικές στήλες έχουν δύο.

```
Από τους δείκτες των άσσων r στις γραμμές , βρες τις θέσεις για τις οποίες το περιεχόμενό τους είναι ίσο με τον μετρητή i και αποθήκευσε στο rowindex

Βρες το minimum product: Πολ/σε το βάρος των γραμμών (συνήθως είναι ο ίδιος αριθμός σε κάθε i) με το βάρος των στηλών από i και μετά(c(rowindex)). Αποθήκευσε το ελάχιστο γινόμενο σε ix.

Επέλεξε τη στήλη που θα τοποθετηθεί πρώτη στον newH με βάση τον δείκτη του ελάχιστου γινομένου από τον πίνακα c.

newH στήλες του i = επιλεγμένη στήλη.

Επανέλαβε 204 φορές.
```

Με βάση αυτόν τον πίνακα κατασκευάζονται και τα parity bits τα οποία θα «κολλήσουν» διαδοχικά με την πληροφορία.

Τα bits αφού κωδικοποιηθούν θα πρέπει να ομαδοποιηθούν και να γίνουν σύμβολα OFDM-MFSK. Η κωδικοποίηση παράγει 408 bits σε σειρά, οπότε για μήκος συμβόλου 6 bits ($M = 2^6$) και για δύο σύμβολα άρα συνολικά 12 bits θα πρέπει να γίνει ομαδοποίηση σε 34 διπλά σύμβολα. Έπρεπε να γίνει επιλογή του πρώτου συμβόλου διαδοχικά με το δεύτερο σύμβολο κ.ο.κ. για 34 φορές. Το πρόβλημα λύθηκε με τον εξής τρόπο:

Κατασκευάστηκε ένας βρόχος for από 1 μέχρι 34 (N = 408, a = $\log_2(M)$)

```
for kk=1:N/(2*a)
```

Το πρώτο σύμβολο παίρνει αρχικά τα bits του u από τις θέσεις 1:6

```
b1 = u(2*(kk-1)*a+1:(2*kk-1)*a);
```

Το δεύτερο σύμβολο παίρνει αρχικά τα bits του u από τις θέσεις 7:12

```
b2 = u((2*kk-1)*a+1:2*kk*a);
```

Στην 2η εκτέλεση της for το `b1`θα ξεκινήσει από το στοιχείο 13 του `u` και θα τελειώσει στο 18 και το `b2` θα ξεκινήσει στο 19 και θα τελειώσει στο 24 κ.ο.κ. Το επόμενο στάδιο είναι τα δύο σύμβολα να μεταφραστούν από δυαδική σε δεκαδική αρίθμηση με την εντολή **`binaryVectorToDecimal()`.**

```
b1dec=binaryVectorToDecimal(b1');
b2dec=binaryVectorToDecimal(b2');
```

Η αρίθμηση τέθηκε στο πρόγραμμα να ξεκινάει από το 1 και όχι από το 0 οπότε προστίθεται και η μονάδα για απαλοιφή των μηδενικών.

```
B1=b1dec+1;
B2=b2dec+1;
```

Τα s1 και s2 παίρνουν τιμές (ενεργοποίηση συχνότητας) και είναι έτοιμα για πολυπλεξία.

```
s1(B1)=1/sqrt(2);
s2(B2)=1/sqrt(2);
```

Για την κατασκευή της κωδικής λέξης, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο θα πρέπει το σύμβολο s2 να πολλαπλασιαστεί και με τον πίνακα omega. Αφού όλες οι παραπάνω εργασίες ολοκληρωθούν γίνεται ο ανάστροφος μετ/σμος Fourier και προστίθεται το cyclic prefix μήκους 16 όπως φαίνεται και στο Σχήμα 6. Τα εκπεμπόμενα σήματα στη MATLAB για κάθε κεραία εκπομπής και για κάθε time-slot:

```
TxA1s1 = ofdm_s1_CP;
TxA2s1 = ofdm_s2_CP;
TxA1s2 = ofdm_os2_CP;
TxA2s2 = ofdm_s1_CP;
```

Η STF κωδικοποίηση έχει πλέον επιτευχθεί με όλες τις παραπάνω διεργασίες. Το επόμενο στάδιο είναι η κατασκευή του MIMO καναλιού καθώς και οι Rayleigh διαλείψεις.

3.1.2 ΚΑΝΑΛΙ MIMO ΚΑΙ ΛΑΜΒΑΝΟΜΕΝΟ ΣΗΜΑ

Τα κέρδη φτιάχτηκαν με βάση τις εξισώσεις που αναφέρονται στο κεφάλαιο 2.3.3. Οι διαλείψεις Rayleigh επιλέχτηκαν να έχουν 5 samples [0,2,5,10,14] και τυπική απόκλιση √0.2. Το συνολικό κανάλι περιγράφεται παρακάτω με τις γραμμές του κώδικα. Το κανάλι περιγράφεται με μιγαδικούς και τυχαίους αριθμούς. Υπάρχουν 8 μεταβλητές, μια για κάθε

συνδυασμό κεραιών ανά time slot. Παρακάτω βρίσκεται ενδεικτικά μια απ' αυτές τις μεταβλητές. Η σημασία των αριθμών στην ονομασία της μεταβλητής υπάρχει παραπάνω στις μεταβλητές-κλειδιά στην εισαγωγή του κεφαλαίου 3.

```
h11_s1(delays+1)=
    sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/
    sqrt(2);
```

Η συνάρτηση **randn()** παράγει τυχαίους αριθμούς σε επιθυμητές διαστάσεις.

Το λαμβανόμενο σήμα πλέον θα είναι η συνέλιξη των κερδών του καναλιού με το εκπεμπόμενο σήμα.

Στη συνέχεια, εισάγεται τυχαίος θόρυβος AWGN στο λαμβανόμενο σήμα ανάλογα της σηματοθορυβικής σχέσης (σ). Ο Gaussian θόρυβος περιγράφεται κι αυτός επίσης όπως οι διαλείψεις με μιγαδικούς αριθμούς. Δημιουργήθηκαν συνολικά τέσσερις συνιστώσες θορύβου μια για κάθε λαμβανόμενο σήμα. Μία από αυτές παρουσιάζεται σαν παράδειγμα πιο κάτω.

Ο μιγαδικός αριθμός πολλαπλασιάζεται με το σ το οποίο περιέχει την πληροφορία της σηματοθορυβικής σχέσης. Μία από τις τέσσερις μεταβλητές:

```
no1=sigma*(randn(1,length(RxA1s1))+1i*randn(1,length(Rx
A1s1)));
```

Το σήμα είναι πλέον έτοιμο προς επεξεργασία στον δέκτη.

3.2 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΔΕΚΤΗ

Αφού το σήμα όπως εξηγήθηκε και παραπάνω υποστεί Rayleigh διαλείψεις και Gaussian θόρυβο, το επόμενο βήμα είναι να γίνει αποδιαμόρφωση και αποκωδικοποίηση των συμβόλων. Σε προηγούμενο κεφάλαιο εξηγήθηκε πως για την αποδιαμόρφωση του STF συμβόλου υπάρχει συγκεκριμένος τρόπος με τρία συνολικά διαφορετικά cases ανάλογα των ενεργών subcarriers *m* και *n* και επιπλέον του αριθμού *q*, για τα οποία υπολογίζονται οι μεταβλητές Ε και *δ*. Πριν χωριστεί το λαμβανόμενο σήμα σε cases θα γίνει αποδιαμόρφωση κατά OFDM όπως φαίνεται και στο Σχήμα 6. Η υλοποίηση των cases πραγματοποιήθηκε σχετικά εύκολα και ως προς τους υπολογισμούς βασίστηκε στις εξισώσεις των Περιπτώσεων 1,2 και 3 από το 2.4.1. Ο υπολογισμός έγινε για όλους τους δυνατούς συνδυασμούς των *n* και *m*. Τα cases υλοποιήθηκαν με απλή συνθήκη else-if και εν τέλει αποθηκεύτηκαν σε έναν πίνακα `decision()` 64x64.

```
for m1=0:M-1;
 for n1=0:M-1;
       q=mod(n1+1,M);
       if m1~=n1 && m1~=q
```

```
            %Υπολόγισε Ε, δ για αυτήν την περίπτωση
            elseifm1==n1

            %Υπολόγισε Ε, δ για αυτήν την περίπτωση

        elseifm1==q

        %Υπολόγισε Ε, δ για αυτήν την περίπτωση

        end
        decision(n1+1,m1+1)=Epsilon-delta;
        end
```

Το αποδιαμορφωμένο σήμα είναι έτοιμο για να εισέλθει στον αποκωδικοποιητή LDPC.

3.2.1 ΥΛΟΠΟΙΗΣΗ ΑΠΟΚΩΔΙΚΟΠΟΙΗΣΗΣ LDPC

Γενικά ο αλγόριθμος sum product είναι ένας αρκετά διάσημος αλγόριθμος LDPC Decoding και χρησιμοποιείται στην εργασία για soft information input αποκωδικοποίηση. Η σχέση $\Lambda(b_j) = \ln\left(\frac{\sum_{m,n:bj=1} p(Y \mid X)}{\sum_{m,n:bj=0} p(Y \mid X)}\right), j = 0,...,2\log_2 M - 1$ αναφέρει πως η αποδιαμορφωμένη πληροφορία θα πρέπει να υποστεί επεξεργασία για να μπορέσει να λειτουργήσει ο Log-Domain αποκωδικοποιητής σωστά. Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, ο αποκωδικοποιητής χρησιμοποιεί τα L(*ci*) για αποκωδικοποίηση, τα οποία είναι στην πραγματικότητα τα LLR σε σειριακή μορφή. Προτού επεξηγηθεί ο τρόπος με τον οποίον κατασκευάστηκαν τα L(*ci*) θα εξηγηθεί συνοπτικά ο αλγόριθμος και έπειτα σε λίγες γραμμές θα αποτυπωθεί ο τρόπος υλοποίησής του.

Το πρώτο του βήμα είναι να υπολογιστούν ποιες οι πιθανότητες του λαμβανόμενου σήματος να είναι bit 1 και ποιες να είναι 0. Στην περίπτωση της εργασίας υπολογίζονται τα Log-Likelihood Ratios (LLR) τα οποία στην περίπτωση του συγκεκριμένου συστήματος απλοποιημένα είναι το υπόλοιπο της μέγιστης απόφασης του δέκτη για όλες τις περιπτώσεις στις οποίες η πληροφορία είναι 1 πλην της μέγιστης απόφασης του δέκτη για όλες τις περιπτώσεις για τις οποίες η πληροφορία είναι 0. Ένας variable node με βάση τα LLR που εισήλθαν σ αυτόν, και τα LLR που έλαβε από τους συνδεδεμένους μ' αυτόν κόμβους έλεγχου (check nodes), κάνει τις πρώτες εκτιμήσεις των LLR. Αφού υπολογιστούν οι πρώτες εκτιμήσεις των LLR, και εφόσον οι parity συναρτήσεις ικανοποιούνται είναι δηλαδή μηδέν, ο αλγόριθμος σταματάει και δίνει την αποκωδικοποιημένη λέξη διορθωμένη. Αλλιώς ξεκινάει το επόμενο βήμα, το Horizontal step. Τα LLR από τους variable nodes περνάνε στους check nodes οι οποίοι με συγκεκριμένους τρόπους κάνουν μια εκτίμηση για το LLR του κάθε

variable node με βάση όλα τα υπόλοιπα LLR που έφτασαν στον συγκεκριμένο check node. Στο επόμενο βήμα (Vertical step) γίνεται η ανάποδη διαδικασία. Αφού υπολογίστηκαν όλες οι εκτιμήσεις των νέων LLR, γυρνάνε στους variable nodes όπου και εκτελείται η δεύτερη διαδικασία ξανά, δηλαδή ο έλεγχος των εξισώσεων απ' τα νέα LLR. Η διαδικασία επαναλαμβάνεται για n φορές. Παρακάτω αναγράφεται ο αλγόριθμος.

Αλγόριθμος Log-domain:

Στους άσσους του newH, αντιστοίχισε τις τιμές του πίνακα L(ci) (LLR). Ο νέος πίνακας είναι ο LQ(ji)

Iteration:
Σε δυο πίνακες αντιστοίχισε τα πρόσημα και τις απόλυτες τιμές του LQ(ji). A(ji), B(ji).

Horizontal step:

Από i=1 μέχρι 204 , βρες τις θέσεις των άσσων του H στη γραμμή I και για αυτές τις τιμές πλην της πρώτης , το στοιχείο minB(ij)=B(j) σε γραμμή i και στήλη cl(l), μέχρι να βρεθεί το μικρότερο B(ij) στη γραμμή i.

Πολ/σε τις θέσεις των μη-μηδενικών cl με το πρόσημο που προκύπτει από τον πολ\σμό της γραμμής I και των θέσεων με τους άσσους (6).-> Prod(A(ij))

Βάλε σε νέο πίνακα, το γινόμενο Prod(A(ij)) επί την χαμηλότερη τιμή του B(ij) (στις θέσεις που καθορίζει το cl).

Κάνε το παραπάνω για όλες τις γραμμές και σώσε στο L(rji).

Vertical step:

Από j=1 μέχρι 408 βρες τις θέσεις των άσσων στην j στήλη (συνήθως 3 άσσοι [column weight])

Σύγκρινε το Lc(ij) (Τα LLR) με το άθροισμα των μη-μηδενικών Lr(ji) για την στήλη j πλην το στοιχείο Lr(ji) στη γραμμή k και στήλη j και ενημέρωσε τα L(qij) .

Για να προσδιοριστεί η τελική απόφαση 'άθροισε την αρχική πληροφορία στη θέση j με το άθροισμα της στήλης j και των γραμμών rl του L(rji).
Οπότε , εάν το αποτέλεσμα είναι μικρότερο του 0 τότε το bit πληροφορίας είναι 1, αλλιώς είναι0.
Επανέλαβε και για τα 408 bits.

Για τα καλύτερα πλέον εκτιμημένα L(qij)(LLR) υπολόγισε ξανά πρόσημα και απόλυτες τιμές, και άλλαξε το τελικό vHat.

Ξεκίνα την αρχική διαδικασία για το επόμενο iteration.

Παρατήρηση: η δηλωμένη μεταβλητή pibetaij δεν χρησιμοποιείται.

Στην περίπτωση του κανονικού log-domain decoder υπάρχει μια διαφορά στον υπολογισμό των απόλυτων τιμών οι οποίες δεν υπολογίζονται με το B(ij) αλλά υπολογίζονται με την μεταβλητή pibetaij η οποία ορίζεται ως $\prod B(ij) = \log(\frac{e^{B(ij)+1}}{e^{B(ij)-1}})$. Ο αλγόριθμος λαμβάνει υπ' όψιν του και τον θόρυβο (η συνάρτηση λειτούργησε και χωρίς την προσθήκη θορύβου αλλά είχε ως αποτέλεσμα απρόβλεπτα λάθη).

Αφού υπολογιστεί το pibetaij ξεκινάει το horizontal step:

Πρόσθεσε τα μη-μηδενικά στοιχεία του pibetaij από την γραμμή i και απ' το αποτέλεσμα αφαίρεσε το k μη-μηδενικό στοιχείο του pibetaij. Αποθήκευσε στο sumofpibetaij.

Κάνε κανονικοποίηση τις πολύ υψηλές τιμές αν χρειαστεί και κάνε update τον pibetaij αυτήν τη φορά με τις τιμές από το sumofpibetaij.

Για τον υπολογισμό του ελάχιστου L(rji) χρησιμοποίησε τον παραπάνω πίνακα. (Ο simple χρησιμοποιεί το minB(ij)).

Ο παραπάνω αλγόριθμος περιγράφει τη συνάρτηση **decodeLogDomainOurs()** η οποία χρησιμοποιήθηκε με μικρή παραμετροποίηση ως προς τον τρόπο λήψης των **Lci** για την υλοποίηση του αποκωδικοποιητή στο σύστημα. Η συνάρτηση παίρνει τέσσερα ορίσματα. Τα L(*ci*) που αναφέρθηκαν και πιο πάνω, τον πίνακα **newH** που δημιουργήθηκε παράλληλα με τα bits ισοτιμίας για την κωδικοποίηση της πληροφορίας από τη συνάρτηση **makeParityChk()**, τον θόρυβο σήματος **No** και την επιθυμητή ποσότητα επαναλήψεων (**iter**) ελέγχου που ζητείται από την συνάρτηση να εκτελεστούν. Έγιναν δοκιμαστικές προσομοιώσεις επίσης με τη συνάρτηση **decodeLogDomainSimpleOurs()** η οποία έχει ως στόχο με μικρές διαφοροποιήσεις στον αλγόριθμο να κάνει εξοικονόμηση χρόνου προσομοίωσης. Αυτή η συνάρτηση δεν λαμβάνει υπ' όψιν της τον θόρυβο **No** του σήματος οπότε δεν βγάζει πιο ικανοποιητικά αποτελέσματα απ' ότι η **decodeLogDomainOurs()**. Παρ' όλα αυτά τα αποτελέσματα αποθηκεύτηκαν για να γίνει σύγκριση μεταξύ των 2 συναρτήσεων.

Προκειμένου να λειτουργήσει η συνάρτηση, έπρεπε να εφευρεθεί τρόπος για να υπολογιστούν τα LLR. Η απλούστερη εξίσωση LLR που αναφέρθηκε σε προηγούμενο κεφάλαιο η $\Lambda(b_j) = \max_{m,n:b_j=1}(\mathrm{E}-\delta) - \max_{m,n:b_j=0}(\mathrm{E}-\delta)$ περιγράφει τη μορφή της πληροφορίας που ζητάει ο αποκωδικοποιητής να έχει στην είσοδό του. Αρχικά κατασκευάστηκε ένας πίνακας **A** με 64 διαφορετικούς φυσικούς δυαδικούς συνδυασμούς, με προφανές μήκος ανά λέξη 6.

```
A=decimalToBinaryVector(0:63,6);
```

Στη συνέχεια, για τον πίνακα **A** βρέθηκαν όλες οι θέσεις του στις οποίες υπήρχαν bit 0 και όλες οι θέσεις για τις οποίες το περιεχόμενό τους ήταν bit 1. Αφού οι τιμές βρέθηκαν χρησιμοποιήθηκαν οι γραμμές οι οποίες δέχτηκαν ανασχηματισμό έτσι ώστε να είναι πιο διαχειρίσιμες στη συνέχεια.

```
[row,col]=find(A==0);
```

Οι θέσεις των μηδενικών είναι:

```
row0 = reshape(row,32,6);
            [row,col]=find(A);
```

Οι θέσεις των άσσων είναι:

```
row1 = reshape(row,32,6);
```

Με την χρήση βρόχου for έγινε συσχέτιση των στοιχείων του πίνακα του αποδιαμορφωμένου σήματος **decision** με τον **row0** και **row1**. Με αυτόν τον τρόπο:

```
for bitnumber=1:6


CCm0 = decision(row0(:,bitnumber),:);
CCm1 = decision(row1(:,bitnumber),:);
maxm0 = max(CCm0(:));
maxm1 = max(CCm1(:));

LLRa(bitnumber) = maxm1-maxm0;
end


for bitnumber2=1:6
CCn0 = decision(:,row0(:,bitnumber2));
CCn1 = decision(:,row1(:,bitnumber2));
maxn0 = max(CCn0(:));
maxn1 = max(CCn1(:));
%LLRb(bitnumber2) = maxn0-maxn1;
LLRb(bitnumber2) = maxn1-maxn0;
end

LLR=[LLRb LLRa];
Lcia(kk,:)=LLR;
```

Στη μεταβλητή **CCm0** αποθηκεύονται κάθε φορά όλες οι στήλες των στοιχείων από τον πίνακα decision που βρίσκονται στις θέσεις 1 έως 32 για την πρώτη for, 1 έως 16 και 33 έως 48 για τη δεύτερη, από 1 έως 8 και από 17 μέχρι 25 καθώς και 33:40 και 49:56 για την τρίτη for κ.ο.κ. για όλες τις θέσεις των 0 bit. Στη μεταβλητή **CCm1** αντιθέτως αποθηκεύονται όλες οι στήλες των στοιχείων από τον πίνακα **decision** που βρίσκονται τα bits τα οποία είναι 1. Δηλ. στις θέσεις 33 έως 64 για την πρώτη for, 17:32 και 49:64 για τη δεύτερη, 9:16 και 26:32 και 41:48 και 57:64 κ.ο.κ. για όλα τα 1 bit. Στη συνέχεια υπολογίζονται οι μέγιστοι όροι των **CCm0** και **CCm1** και το υπόλοιπό τους σχηματίζει το πρώτο LLR που θα χρησιμοποιηθεί στον αποκωδικοποιητή. Η επανάληψη for για τα δεύτερα LLR γεμίζει αυτήν την φορά τις γραμμές αντί για τις στήλες. Έτσι, η μεταβλητή **CCn0** θα γεμίσει τις γραμμές της με τα περιεχόμενα των θέσεων του πίνακα decision που στον **A** περιέχουν μηδενικά και αντιθέτως η **CCn1** θα γεμίσει τις γραμμές της με τις θέσεις που περιέχουν άσσους. Το υπόλοιπο των μεγίστων **CCn0** και **CCn1** είναι το δεύτερο πλέον LLR που θα μπει στον αποκωδικοποιητή. Τέλος, οι δύο πίνακες **LLRa** και **LLRb** επεξεργάζονται και ανασχηματίζονται σε έναν πίνακα **Lci** διαστάσεων 1x408.

3.3 SISO ΚΑΝΑΛΙ

Στην εισαγωγή αναφέρθηκε πως έγινε επιπλέον προσομοίωση με SISO κανάλι για λόγους σύγκρισης. Ο κώδικας σε αυτήν την περίπτωση είναι πολύ απλοποιημένος σε σχέση με το MIMO κανάλι. Στο SISO υπάρχει μια κεραία εκπομπής και μια λήψης, η χρήση του πίνακα κυκλικής μετακίνησης δεν είναι απαραίτητη. Ο τρόπος ενεργοποίησης του υποφέροντος δεν επηρεάζεται από το κανάλι οπότε παραμένει ίδιος, τα δύο σύμβολα s1 και s2 θα διαμορφωθούν με OFDM και θα προστεθεί κυκλικό πρόθεμα στο καθένα. Το κέρδος του καναλιού είναι ένα και μοναδικό. Η επιλογή του δέκτη είναι πολύ πιο εύκολη αφού υπολογίζονται χωρίς τη χρήση των τριών περιπτώσεων (cases). Παρακάτω αποτυπώνεται συνοπτικά ο κώδικας για την εκπομπή των συμβόλων και την αποδιαμόρφωσή του. Στη SISO προσομοίωση δεν χρησιμοποιήθηκε κώδικας LDPC.

```
SISOTxA1s1 = SISOofdm_s1_CP;
SISOTxA1s2 = SISOofdm_s2_CP;

h11_s1(delays+1)=sd.*(randn(1,length(delays))+1i*rand
n(1,length(delays)))/sqrt(2);

SISORxA1s1 = conv(SISOTxA1s1,h11_s1);
SISORxA1s2 = conv(SISOTxA1s2,h11_s1);
```

```
SISOy1s1 = fft(SISORxA1s1_rCP)/sqrt(M);
SISOy1s2 = fft(SISORxA1s2_rCP)/sqrt(M);

        %Demodulation
[SISOCC1,II1] = max(abs(SISOy1s1));
 [SISOCC2,II2] = max(abs(SISOy1s2));
SISOmbits = decimalToBinaryVector(II1-1,log2(M));
SISOnbits = decimalToBinaryVector(II2-1,log2(M));
SISOb_hat = [SISOmbits,SISOnbits];
```

3.4 ΥΠΟΛΟΓΙΣΜΟΣ ΛΑΘΩΝ ΚΑΙ BIT ERROR RATE

Ο πίνακας Lci μαζί με τον newH εισέρχονται στη συνάρτηση αποκωδικοποίησης καθώς και επίσης ο θόρυβος του σήματος No, και όπως αναφέρθηκε σε προηγούμενο κεφάλαιο λαμβάνει χώρα ο αλγόριθμος Sum Product και το αποτέλεσμα της συνάρτησης vhat είναι κατά βάση η αρχική πληροφορία. Για να επιβεβαιώνεται αν όντως είναι όλα τα bits πληροφορίας σωστά χωρίς λάθη, τοποθετήθηκε ένας έλεγχος ο οποίος συλλέγει κάθε λάθος bit που έγινε από το δέκτη.

```
        bit_errors_LDPC(ii)=bit_errors_LDPC(ii)+sum(abs
(u-vhat'));
```

Γίνεται αφαίρεση της αρχικής λέξης με την λαμβανόμενη. Για να μην υπάρχουν αρνητικοί άσσοι σαν αποτέλεσμα της αφαίρεσης το αποτέλεσμα βγαίνει με απόλυτη τιμή. Στη συνέχεια και αν δεν υπάρχει κανένα λάθος, το υπόλοιπο των δυο λέξεων θα πρέπει να είναι μηδέν, ειδάλλως γίνεται καταμέτρηση του πλήθους των άσσων από το υπόλοιπο οι οποίοι υποδηλώνουν λάθη στην λέξη. Τα λάθη αποθηκεύονται στη μεταβλητή **`bit_errors_LDPC(ii)`** και χρησιμοποιούνται στο τέλος για την εύρεση του BER. Ολόκληρη η προσομοίωση ελέγχεται από έναν βρόχο while και την ποσότητα λαθών bits που κάνει ο αποκωδικοποιητής. Για να σταματήσει ο βρόχος while και να προχωρήσει το πρόγραμμα σε επόμενη σηματοθορυβική σχέση για νέους υπολογισμούς θα πρέπει να γίνουν συνολικά 2000 λάθη σε bits. Τέλος, γίνεται ο υπολογισμός του BER με τον λόγο $BER = \frac{bit_{errors}}{bit_{total}}$ και την εντολή **`bercoded = bit_errors_LDPC./total_bits;`** και η απεικόνιση των ημιλογαριθμικών γραφικών παραστάσεων.

3.5 ΠΡΟΒΛΗΜΑΤΑ ΠΟΥ ΑΝΤΙΜΕΤΩΠΙΣΤΗΚΑΝ

Τα κυριότερα προβλήματα που αντιμετωπίστηκαν αφορούσαν την κωδικοποίηση κατά LDPC. Δεν ήταν δυνατό να κατασκευαστεί κωδικοποιητής και αποκωδικοποιητής έτσι ώστε να ολοκληρωθεί το σύστημα. Αρχικά η προσομοίωση έγινε χωρίς τη χρήση κώδικα LDPC,

γρήγορα όμως διαπιστώθηκε πως υπάρχει χαμηλή απόδοση στο σύστημα οπότε και ξεκίνησε η αναζήτηση κωδίκων LDPC. Το πρώτο βήμα ήταν να ελεγχθεί από το [10] τι δεδομένα υπήρχαν. Τελικά βρέθηκε πίνακας ισοτιμίας Η υπό τη μορφή δεικτών και αξιοποιήθηκε αρχικά. Στη συνέχεια έγινε αναζήτηση στο μενού help της MATLAB προκειμένου να βρεθεί συνάρτηση LDPC κωδικοποίησης και αποκωδικοποίησης. Η MATLAB παρέχει στο Communications toolbox συναρτήσεις LDPC, τις **`comm.LDPCEncoder`** και **`comm.LDPCDecoder`** οι οποίες όμως δεν μπορούσαν να συμβαδίσουν με τα δεδομένα του προγράμματος και παρουσίαζαν συνεχώς προβλήματα. Με λεπτομερή αναζήτηση στο Διαδίκτυο παρατηρήθηκε πως τα περισσότερα τηλεπικοινωνιακά συστήματα με κωδικοποίηση ήταν γραμμένα σε γλώσσα προγραμματισμού C όπως τα πακέτα προσομοίωσης του Valenti κι έτσι ήταν αδύνατο να χρησιμοποιηθούν στη MATLAB. Τελικά ανακαλύφθηκε ένα site [11] το οποίο περιείχε ένα πακέτο με custom συναρτήσεις LDPC της MATLAB, καθώς και demo πρόγραμμα που λειτουργούσε εκπαιδευτικά και το οποίο επιβεβαίωνε πως οι συναρτήσεις λειτουργούσαν σωστά.

4 ΑΠΟΤΕΛΕΣΜΑΤΑ

Σε αυτό το κεφάλαιο απεικονίζονται όλα τα αποτελέσματα της προσομοίωσης. Σαν αρχικό μέτρο σύγκρισης χρησιμοποιήθηκε μια θεωρητική καμπύλη σε διαλείψεις Rayleigh για να επιβεβαιωθεί και η σωστή λειτουργία του συστήματος. Για τον υπολογισμό της καμπύλης χρησιμοποιήθηκε το μαθηματικό πρόγραμμα Mathematica. Η αρχική συνάρτηση που δοκιμάστηκε για την κατασκευή της καμπύλης είναι η

$$P_l = \frac{1}{(L-1)!}\sum_{m=1}^{M-1}\frac{(-1)^{m+1}\binom{M-1}{m}}{(1+m+m\bar{\gamma}_c)^L}\times\sum_{k=0}^{m(L-1)}\beta_{km}(L-1+k)!\left(\frac{1+\bar{\gamma}_c}{1+m+m\bar{\gamma}_c}\right)^k$$ [15] [εξ. (13.4-49)] η

οποία χρησιμοποιείται για 1 diversity branch άρα L = 1. Απλοποιώντας το δεύτερο άθροισμα η συνάρτηση γίνεται $P_l = \sum_{m=1}^{M-1}\frac{(-1)^{m+1}\binom{M-1}{m}}{(1+m+m\bar{\gamma}_c)^L}$ [15] [εξ. (13.4-50)]. Ο διωνυμικός όρος $\binom{M-1}{m}$ για μεγάλα M είναι πάρα πολύ μεγάλος και πρακτικά το άθροισμα γίνεται μη υπολογίσιμο. Για αυτόν τον λόγο στον υπολογισμό της πιθανότητας σφάλματος

χρησιμοποιήθηκε το αρχικό ολοκλήρωμα

$$P_l = 1 - \int_0^{\infty} \frac{1}{(2\sigma_1^2)^L (L-1)!} u_1^{L-1} \exp\left(-\frac{u_1}{2\sigma_1^2}\right) \times \left[1 - \exp\left(-\frac{u_1}{2\sigma_2^2}\right) \sum_{k=0}^{L-1} \frac{1}{k!} \left(\frac{u_1}{2\sigma_2^2}\right)^k\right]^{M-1} du_1 \quad [15] \quad [\text{ξ}$$

(13.4-47)] και τέλος η συνολική πιθανότητα δίνεται από το $P_b = \frac{2^{\log_2 M - 1}}{2^{\log_2 M} - 1} P_l$ [3] (Γ2- σελ 58).

Για το SISO κανάλι έγινε προσομοίωση με προσθήκη μόνο Gaussian θορύβου αλλά και σε περιβάλλον διαλείψεων για $M = 16$ και $M = 64$. Για υπόλοιπα συστήματα έγινε προσομοίωση μόνο σε περιβάλλοντα διαλείψεων. Αναλυτικότερα, έγινε προσομοίωση MIMO καναλιού για $M = 64$ χωρίς κωδικοποίηση, με αποκωδικοποίηση Log-Domain Simple για 20 iterations καθώς και για 50 iterations, με αποκωδικοποίηση Log-Domain επίσης για 20 και 50 iterations. Τέλος έγινε υπολογισμός του ρυθμού λαθών ανά πακέτο για την αποκωδικοποίηση Log-Domain.

Στις καμπύλες μόνο με AWGN αναζητείται η ορθή λειτουργία του συστήματος η οποία παρατηρείται στο Σχήμα 8 πως λειτουργεί σωστά. Με προσαύξηση των συμβόλων από 16 σε 64 υπάρχει μια μικρή βελτίωση στο BER της τάξης περίπου του 1dB. Στα 5dB τα 64 σύμβολα έχουν BER περίπου 8*10-3 ενώ τα 16 σύμβολα φτάνουν το ίδιο BER σε κάτι παραπάνω από ένα dB αργότερα.

Όπως φαίνεται στο Σχήμα 9, το MIMO σύστημα και για FSK με $M = 64$ είναι πολύ πιο αποδοτικό σε ρυθμό λαθών bit από το SISO σύστημα για τον ίδιο αριθμό συμβόλων $M = 64$ σε περιβάλλοντα με διαλείψεις περίπου κατά 10dB. Η θεωρητική καμπύλη συμπίπτει με αυτή του SISO για 64 σύμβολα.

Χρησιμοποιώντας επιπλέον κωδικοποίηση LDPC στα bits πληροφορίας πριν τη διαμόρφωσή τους ο ρυθμός λαθών bits για τα MIMO συστήματα και για M = 64 βελτιώνεται περισσότερο και οι καμπύλες πέφτουν κατακόρυφα στα 2.5-3dB όπως φαίνεται και στο Σχήμα 10. Αναλυτικότερα, ο κώδικας LDPC Simple των 20 iterations και για M = 64 παρουσιάζει 10^{-4} BER για σηματοθορυβική των περίπου 2.5dB ενώ χωρίς LDPC και για το ίδιο BER η καμπύλη βρίσκεται στα περίπου 12.5dB.

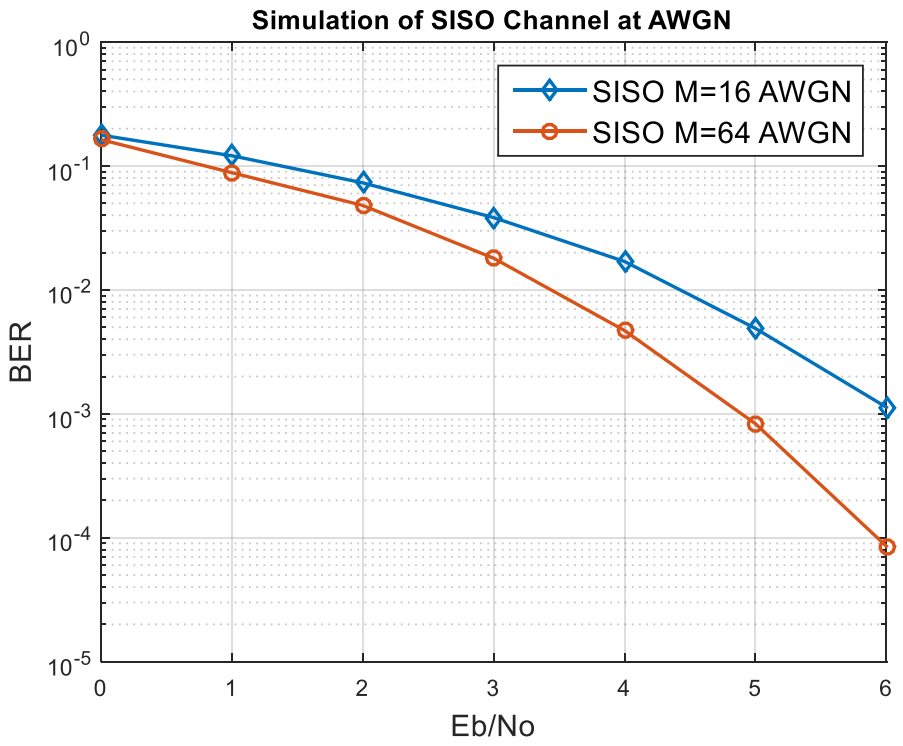
Οι δοκιμές για 50 iterations στο Σχήμα 11 έδειξαν ότι έχουν καλύτερη απόδοση από των 20 iterations με τον κανονικό αλγόριθμο να υπερισχύει και εδώ.

Παρατηρήθηκε πως υπάρχει διαφορά με τον κανονικό LDPC κώδικα των 20 iterations ο οποίος στα 2.5dB παρουσιάζει 4*10-5 λάθη κάνοντας τον simple κώδικα υποδεέστερο του. Με τα 50 iterations παρατηρήθηκε μικρή βελτίωση σε σχέση με τα 20 iterations όπως φαίνεται και στο Σχήμα 12.

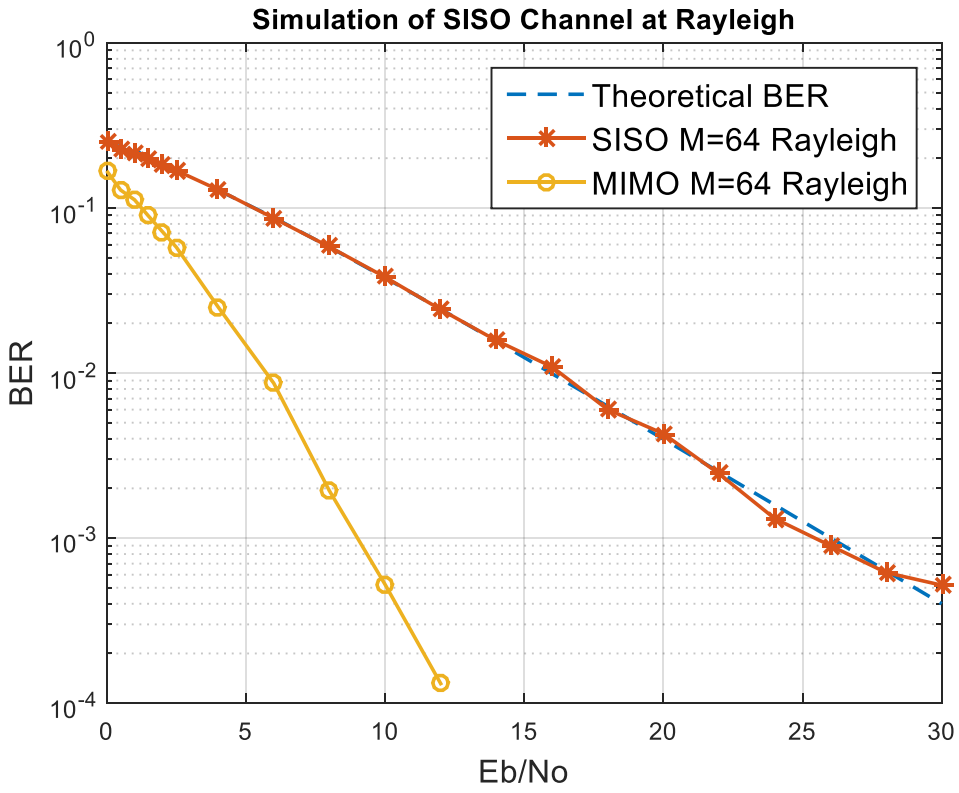
Για να υπάρξει λάθος σε ένα πακέτο θα πρέπει έστω κι ένα bit του πακέτου να είναι λάθος. Με αυτόν τον τρόπο υπολογίστηκε η καμπύλη του Σχήματος 13 για $M = 64$ και με LDPC κώδικα των 50 επαναλήψεων.

Στο Σχήμα 14 φαίνεται η γενική εικόνα των αποτελεσμάτων της εργασίας. Φαίνεται πως γενικά το MIMO κανάλι υπερισχύει από αυτό του SISO και με την προσθήκη LDPC κωδίκων το σύστημα γίνεται ακόμη πιο αξιόπιστο.

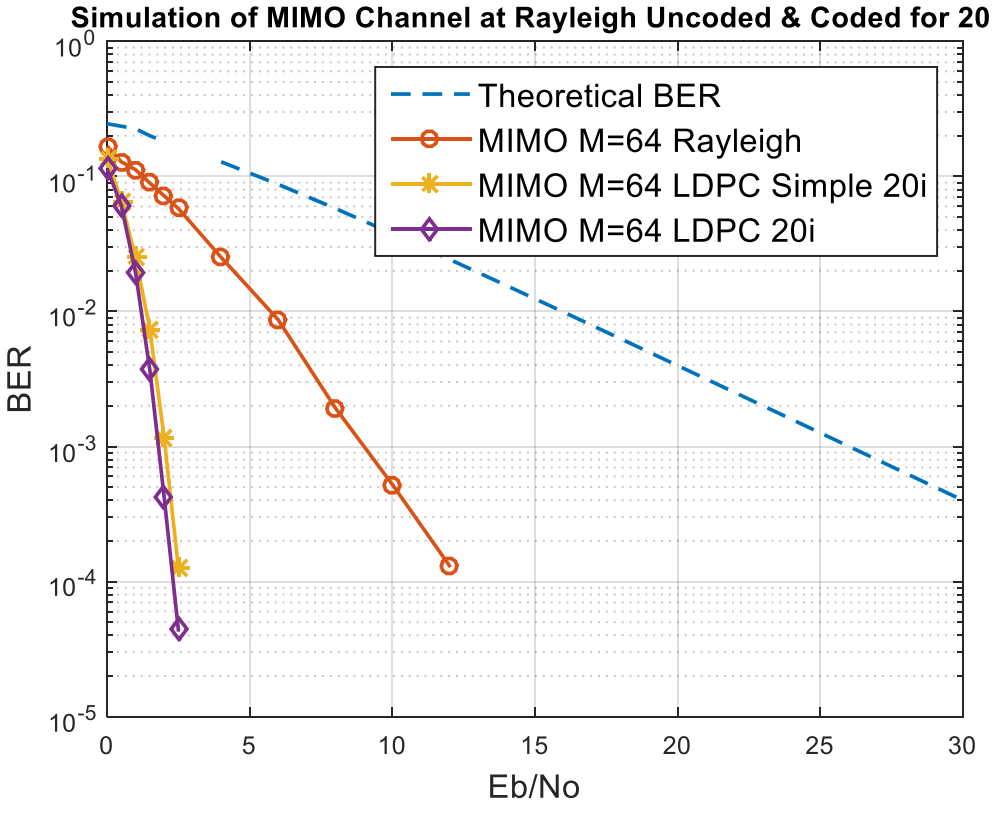
Παρακάτω ακολουθούν οι γραφικές παραστάσεις των αποτελεσμάτων:



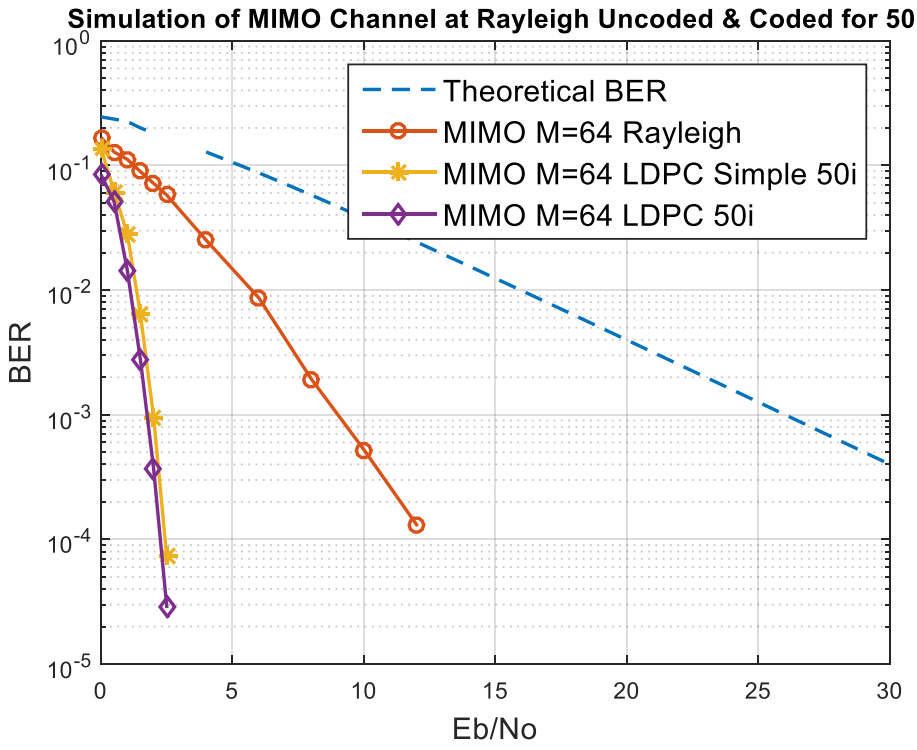
Σχήμα 8: Προσομοίωση BER SISO καναλιού μόνο με AWGN θόρυβο και για M=16 & M=64



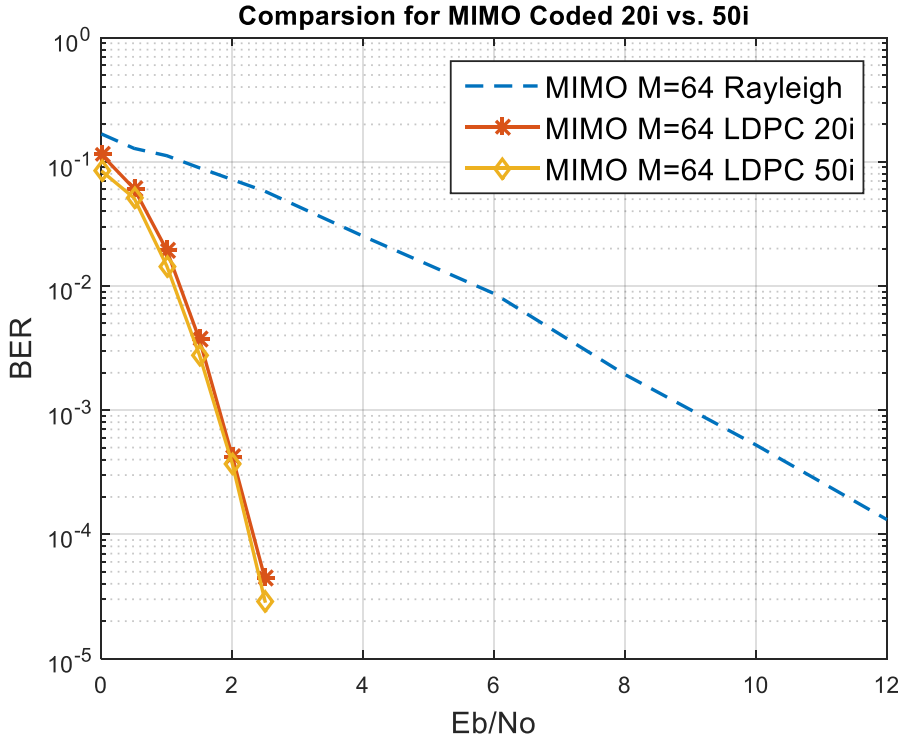
Σχήμα 9:Προσομοίωση BER SISO καναλιού για M=64 & M=16 καθώς και MIMO καναλιού για M=64 σε Rayleigh διαλείψεις



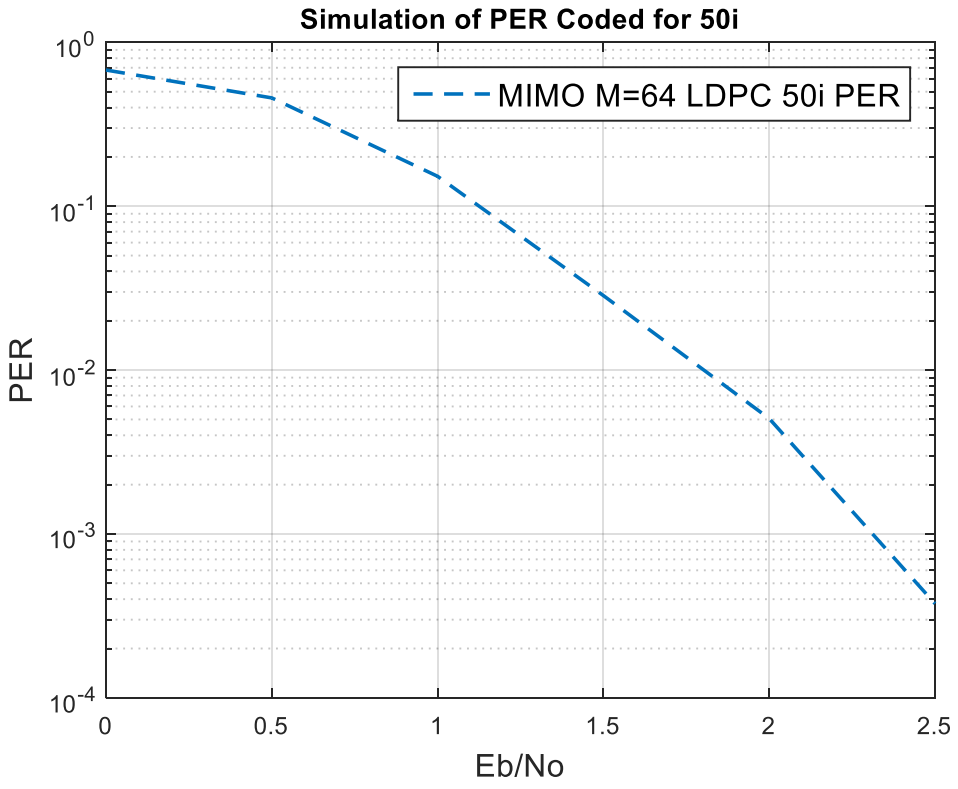
Σχήμα 10:Προσομοίωση BER MIMO καναλιού για M=64 σε Rayleigh διαλείψεις χωρίς αποκωδικοποίηση και με LDPC 20i



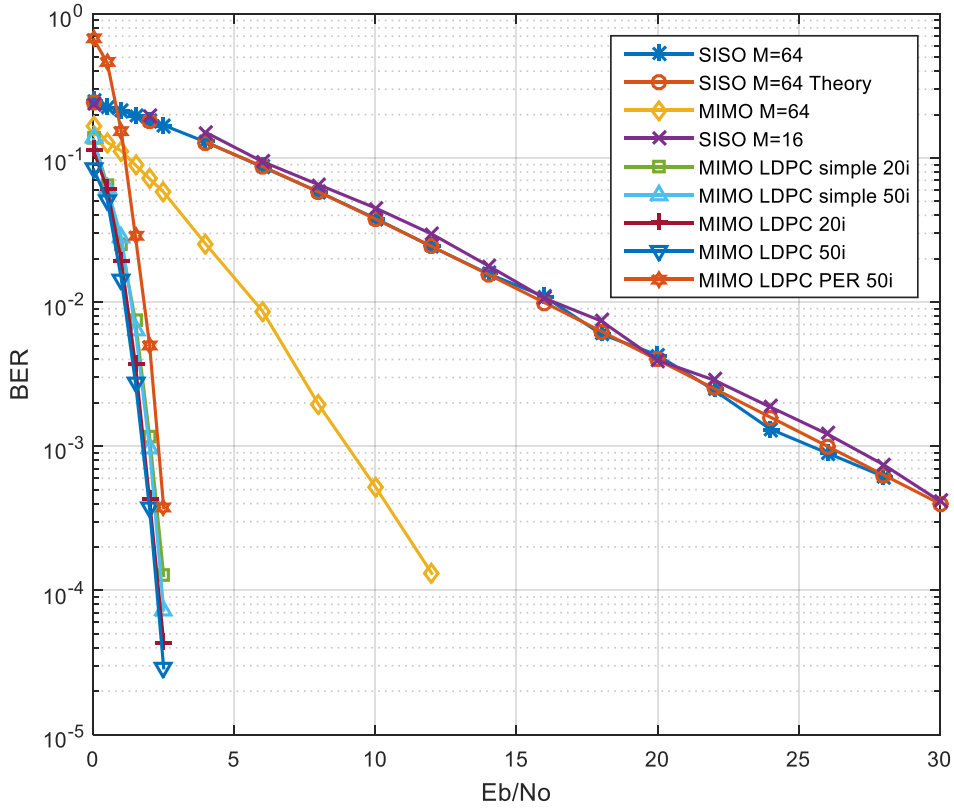
Σχήμα 11:Προσομοίωση BER MIMO καναλιού για M=64 σε Rayleigh διαλείψεις χωρίς αποκωδικοποίηση και με LDPC 50i



Σχήμα 12:Προσομοίωση BER MIMO καναλιού για M=64 σε Rayleigh διαλείψεις με LDPC 20iκαι 50i



Σχήμα 13:Προσομοίωση PER MIMO καναλιού για M=64 σε Rayleigh διαλείψεις με LDPC 50i



Σχήμα 14:Συλλογικά όλα τα αποτελέσματα της προσομοίωσης με και χωρίς LDPC

5 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα πτυχιακή εργασία μελετήθηκε ένα τηλεπικοινωνιακό σύστημα MIMO OFDM διαμόρφωσης MFSK και ασύμφωνης αποδιαμόρφωσης σε περιβάλλον τυχαίων Rayleigh διαλείψεων και με προσθήκη AGWN, το οποίο υλοποιήθηκε στο πρόγραμμα MATLAB. Το σύστημα για τυχαίο Rayleigh κανάλι φαίνεται να λειτουργεί σωστά. Η χρήση του κώδικα διόρθωσης λαθών (408,204) LDPC είναι κριτικής σημασίας για το σύστημα διότι βελτιώνει κατά πολύ το BER στον δέκτη όπως φάνηκε και στο 4^{ο} κεφάλαιο.

Αναλυτικότερα, το σχήμα λειτουργεί πολύ καλύτερα σε MIMO κανάλι με τις δύο κεραίες εκπομπής και τις δύο κεραίες λήψης απ' ότι σε SISO με μία κεραία σε πομπό και μία σε δέκτη. Σε AGWN με τυχαίο Rayleigh κανάλι, με προσθήκη του LDPC κώδικα και συγκεκριμένα του σκέτου αλγόριθμου (όχι του simple) των 50 επαναλήψεων παράχθηκαν τα καλύτερα αποτελέσματα της πτυχιακής εργασίας, περίπου $3*10^{-5}$ λάθη για SNR των 2.5 dB. Η χρήση του αλγόριθμου simple και των 20 επαναλήψεων από την άλλη πλευρά, βγάζει αποδεκτά αποτελέσματα, αλλά υποδυέστερα του σκέτου αλγόριθμου. Έχει όμως το προτέρημα της ταχύτερης προσομοίωσης.

Ένα επόμενο βήμα για την βελτίωση της πτυχιακής εργασίας είναι να γίνουν μελέτες σε πραγματικά και πρακτικά περιβάλλοντα διαλείψεων όπως το EPA το EVA και το ETU, τα

κυρίαρχα πρότυπα στο LTE. Επίσης, η πρακτική υλοποίηση ολόκληρου ή μέρους του συστήματος θα μπορούσε να είναι εφικτή με την χρήση διαφόρων kit προγραμματιζόμενων ολοκληρωμένων κυκλωμάτων Field Programmable Gate Arrays (FPGA) τα οποία χρησιμοποιούνται εκτός από άλλες εφαρμογές και για την υλοποίηση τηλεπικοινωνιακών συστημάτων. Επιπλέον, η μελέτη της GFDM (Generalized Frequency Divided Multiplex) πολυπλεξίας στην οποία πολλά από τα μειονεκτήματα του OFDM βελτιώνονται, όπως η Out-Of-Band εκπομπή, η χρονική του ευαισθησία καθώς και το βασικότερο πρόβλημα, ο υψηλός λόγος PAPR (Peak-to-Average Power Ratio), θα μπορούσε να είναι η μελλοντική εξέλιξη και κατεύθυνση της εργασίας [14].

ΠΑΡΑΡΤΗΜΑ

ΚΩΔΙΚΑΣ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

```
clearall
closeall
clc

%foropenloop=1:10

M=64;%input('Set number M of Subcarriers: ')
%load matrix from MacKay
% load LDPCmatrix arra
% for i=1:length(arra)
%     for j=1:3
%        f= arra(i,j);
%        arrc(i,f)=1;
%     end
% end
% ParityMatrix=arrc';
%%%%%%%%%%%%%%%
%LDPC Variables
K = 204;
N = 408;
onePerCol = 3;
ParityMatrix = makeLdpc(K, N, 1, 1, onePerCol);
iter = 20;
strategy=2;
%%%%%%%%%%%%%%%
%cyclic prefix
CP=16;
%alphabet of b
ab=[1 0];
%Set Symbols s1 & s2
s1=zeros(M,1);
s2=zeros(M,1);
%Length of b
a=log2(M);
%Set Eb/No & Es/No rates to simulate
%Es_No_dB=0:5:60;
%Eb_No_dB=0:0.5:2.5;
Eb_No_dB=2.5;
Es_No_dB=Eb_No_dB+10*log10(a);
%Eb_No_dB = Es_No_dB1-10*log10(log2(M));
%Set BER Variables
bit_errors_no_coding=zeros(1,length(Es_No_dB));
bit_errors_no_coding_SISO=zeros(1,length(Es_No_dB));
bit_errors_LDPC=zeros(1,length(Es_No_dB));
packet_errors=zeros(1,length(Es_No_dB));
total_bits=zeros(1,length(Es_No_dB));
total_packets=zeros(1,length(Es_No_dB));
%Main simulation
for ii=1:length(Es_No_dB)
    required_errors=2000;
xxxx=0;
%while bit_errors_LDPC(ii)<required_errors continue for loop
while bit_errors_LDPC(ii)<required_errors
xxxx=xxxx+1;
        total_packets(ii)=xxxx;
        total_bits(ii)=total_bits(ii)+N;
        EsNo = 10.^(0.1*Es_No_dB(ii));
```

```
    %InfoBits = round(rand(K,1));
            InfoBits = round(rand(K,1)); % change N to K for the coding case
    %Encode message
    %[ParityBits, newH] = makeParityChkOurs(InfoBits, ParityMatrix,
strategy);
            [ParityBits, newH] = makeParityChk(InfoBits, ParityMatrix,
strategy);
    %Make systemic codeword
            u = [ParityBits; InfoBits];
    %u = InfoBits;
    %N0=[];
            u_hat = [];
            SISOu_hat = [];
    bbb =[];
    %STF coding begins
    for kk=1:N/(2*a)
                b1 = u(2*(kk-1)*a+1:(2*kk-1)*a);
                b2 = u((2*kk-1)*a+1:2*kk*a);
                b=[b1;b2];
    %h thesi toy energoy subcarrier stoys pinakes s1 kai s2(arithmisi apo to
1)
                b1dec=binaryVectorToDecimal(b1');
                b2dec=binaryVectorToDecimal(b2');
                B1=b1dec+1;
                B2=b2dec+1;
    %ta subcarriers ginontai energa
                s1=zeros(M,1);
                s2=zeros(M,1);
    s1(B1)=1/sqrt(2);
    s2(B2)=1/sqrt(2);

    %dimiourgia leksis X
                omega=[zeros(1,M-1),1;eye(M-1),zeros(M-1,1)];
    %x1=[s1;s2];
    %x2=[omega*s2;s1];
    %Make OFDM symbols
                ofdm_s1 = sqrt(M)*ifft(s1).';
                ofdm_s2 = sqrt(M)*ifft(s2).';
                ofdm_os2 = sqrt(M)*ifft(omega*s2).';
                ofdm_s1_CP = [ofdm_s1(end-CP+1:end),ofdm_s1];
                ofdm_s2_CP = [ofdm_s2(end-CP+1:end),ofdm_s2];
                ofdm_os2_CP = [ofdm_os2(end-CP+1:end),ofdm_os2];
    %             SISOofdm_s1 = sqrt(2)*sqrt(M)*ifft(s1).';
    %             SISOofdm_s2 = sqrt(2)*sqrt(M)*ifft(s2).';
    %             SISOofdm_s1_CP = [SISOofdm_s1(end-CP+1:end),SISOofdm_s1];
    %             SISOofdm_s2_CP = [SISOofdm_s2(end-CP+1:end),SISOofdm_s2];

                TxA1s1 = ofdm_s1_CP;
                TxA2s1 = ofdm_s2_CP;
                TxA1s2 = ofdm_os2_CP;
                TxA2s2 = ofdm_s1_CP;
    %             SISOTxA1s1 = SISOofdm_s1_CP;
    %             SISOTxA1s2 = SISOofdm_s2_CP;

    %pdp=[0,-2,-3,-3,-7,-10]; %dB
    %sd = db2mag(pdp);
    sd = sqrt([.2,.2,.2,.2,.2]);
    delays=[0,2,5,10,14];

    %             sd = 1;
    %             delays=0;
```

```
                h11_s1 = zeros(1,max(delays)+1);
                h12_s1 = zeros(1,max(delays)+1);
                h21_s1 = zeros(1,max(delays)+1);
                h22_s1 = zeros(1,max(delays)+1);
                h11_s2 = zeros(1,max(delays)+1);
                h12_s2 = zeros(1,max(delays)+1);
                h21_s2 = zeros(1,max(delays)+1);
                h22_s2 = zeros(1,max(delays)+1);

                h11_s1(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
                h12_s1(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
                h21_s1(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
                h22_s1(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
    %             h11_s2=h11_s1;
    %             h12_s2=h12_s1;
    %             h21_s2=h21_s1;
    %             h22_s2=h22_s2;
    %             h11_s2=1;
    %             h12_s2=1;
    %             h21_s2=1;
    %             h22_s2=1;
                h11_s2(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
                h12_s2(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
                h21_s2(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
                h22_s2(delays+1) =
sd.*(randn(1,length(delays))+1i*randn(1,length(delays)))/sqrt(2);
    %             h1 = sd.*(randn(1,1)+1i*randn(1,1))/sqrt(2);
                RxA1s1 = conv(TxA1s1,h11_s1)+conv(TxA2s1,h21_s1);
                RxA2s1 = conv(TxA1s1,h12_s1)+conv(TxA2s1,h22_s1);
                RxA1s2 = conv(TxA1s2,h11_s2)+conv(TxA2s2,h21_s2);
                RxA2s2 = conv(TxA1s2,h12_s2)+conv(TxA2s2,h22_s2);
                RxA1s2(1:max(delays)) = RxA1s2(1:max(delays))+RxA1s1(end-
max(delays)+1:end);
                RxA2s2(1:max(delays)) = RxA2s2(1:max(delays))+RxA2s1(end-
max(delays)+1:end);
    %             SISORxA1s1 = conv(SISOTxA1s1,h11_s1);
    %             SISORxA1s2 = conv(SISOTxA1s2,h11_s1);
    %             SISORxA1s2(1:max(delays)) =
SISORxA1s2(1:max(delays))+SISORxA1s1(end-max(delays)+1:end);
    %             RxA1s1 = TxA1s1+TxA2s1;
    %             RxA2s1 = TxA1s1+TxA2s1;
    %             RxA1s2 = TxA1s2+TxA2s2;
    %             RxA2s2 = TxA1s2+TxA2s2;

    %noise
    sigma=1/sqrt(EsNo);
    sigma=sigma/sqrt(2);

    %             A=norm(RxA1s1).^2;
    %             B=norm(RxA2s1).^2;
    %             C=norm(RxA1s2).^2;
    %             D=norm(RxA2s2).^2;
    %             symenergy1=(A+C)/2;
```

```
    %             symenergy2=(B+D)/2;
    %      EsNo=80;
    %      esnonorm=db2mag(EsNo);

    %no=sqrt(symenergy/(2*esnonorm))*(sigma*randn(1,length(RxA1s1))+1i*sigma
*randn(1,length(RxA1s1)));

no1=sigma*(randn(1,length(RxA1s1))+1i*randn(1,length(RxA1s1)));

no2=sigma*(randn(1,length(RxA1s1))+1i*randn(1,length(RxA1s1)));

no3=sigma*(randn(1,length(RxA1s1))+1i*randn(1,length(RxA1s1)));

no4=sigma*(randn(1,length(RxA1s1))+1i*randn(1,length(RxA1s1)));
    %no1=zeros(1,length(RxA1s1)); no2=no1; no3=no1; no4=no1;
                RxA1s1_n=RxA1s1+no1;
                RxA2s1_n=RxA2s1+no2;
                RxA1s2_n=RxA1s2+no3;
                RxA2s2_n=RxA2s2+no4;
    %             SISORxA1s1_n = SISORxA1s1+no1;
    %             SISORxA1s2_n = SISORxA1s2+no3;

                RxA1s1_rCP = RxA1s1_n(CP+1:CP+M);
                RxA2s1_rCP = RxA1s1_n(CP+1:CP+M);
                RxA1s2_rCP = RxA1s2_n(CP+1:CP+M);
                RxA2s2_rCP = RxA2s2_n(CP+1:CP+M);
    %             SISORxA1s1_rCP = SISORxA1s1_n(CP+1:CP+M);
    %             SISORxA1s2_rCP = SISORxA1s2_n(CP+1:CP+M);

                y1s1 = fft(RxA1s1_rCP)/sqrt(M);
                y2s1 = fft(RxA2s1_rCP)/sqrt(M);
                y1s2 = fft(RxA1s2_rCP)/sqrt(M);
                y2s2 = fft(RxA2s2_rCP)/sqrt(M);
    %             SISOy1s1 = fft(SISORxA1s1_rCP)/sqrt(M);
    %             SISOy1s2 = fft(SISORxA1s2_rCP)/sqrt(M);

    %             x1=[ofdm_s1;ofdm_s2];
    %             x2=[ofdm_os2;ofdm_s1];
    %             X=[x1 x2];
    %cases(arithmisi apo to 0)
                Es=1;%symenergy;
                No=sigma^2;
    decision=zeros(M,M);
    for m1=0:M-1;
    for n1=0:M-1;
                        q=mod(n1+1,M);
    %detecting
    %c=((pi*sigma)^(-4*M))*exp((-norm(X).^2)/sigma)
    if m1~=n1 && m1~=q
    %disp('This codeword is a case 1 example')
                            y1s1m=abs(y1s1(m1+1))^2;
                            y1s1n=abs(y1s1(n1+1))^2;
                            y1s2q=abs(y1s2(q+1))^2;
                            y1s2m=abs(y1s2(m1+1))^2;
                            y2s1m=abs(y2s1(m1+1))^2;
                            y2s1n=abs(y2s1(n1+1))^2;
                            y2s2q=abs(y2s2(q+1))^2;
                            y2s2m=abs(y2s2(m1+1))^2;

Epsilon=(Es/No)*((y1s1m+y1s1n+y1s2q+y1s2m+y2s1m+y2s1n+y2s2q+y2s2m)/(Es+No))
;
```

```
delta=4*log(Es/No+1);
elseif m1==n1
%disp('This codeword is a case 2 example')
                        y1s1m=abs(y1s1(m1+1))^2;
                        y2s1m=abs(y2s1(m1+1))^2;
                        y1s2m=abs(y1s2(m1+1))^2;
                        y2s2m=abs(y2s2(m1+1))^2;
                        y1s2q=abs(y1s2(q+1))^2;
                        y2s2q=abs(y2s2(q+1))^2;
                        e1=2*(y1s1m+y2s1m)/(2*Es+No);
                        e2=(y1s2q+y2s2q+y1s2m+y2s2m)/(Es+No);
                        Epsilon=(Es/No)*(e1+e2);
delta=2*log(2*Es/No+1);

elseif m1==q
%disp('This codeword is a case 3 example')
                        y1s1m=abs(y1s1(m1+1))^2;
                        y2s1m=abs(y2s1(m1+1))^2;
                        y1s2m=abs(y1s2(m1+1))^2;
                        y2s2m=abs(y2s2(m1+1))^2;
                        y1s1n=abs(y1s1(n1+1))^2;
                        y2s1n=abs(y2s1(n1+1))^2;
                        e1=(y1s1m+y2s1m+y1s1n+y2s1n)/(Es+No);
                        e2=2*(y1s2m+y2s2m)/(2*Es+No);
                        Epsilon=(Es/No)*(e1+e2);
delta=2*log(2*Es/No+1);
end
decision(n1+1,m1+1)=Epsilon-delta;
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%              ww=M/2;
%
            A=decimalToBinaryVector(0:63,6);
            [row,col]=find(A==0);
            row0 = reshape(row,32,6);
            [row,col]=find(A);
            row1 = reshape(row,32,6);

for bitnumber=1:6
                CCm0 = decision(row0(:,bitnumber),:);
                CCm1 = decision(row1(:,bitnumber),:);
                maxm0 = max(CCm0(:));
                maxm1 = max(CCm1(:));
%LLRa(bitnumber) = maxm0-maxm1;
LLRa(bitnumber) = maxm1-maxm0;
end

for bitnumber2=1:6
                CCn0 = decision(:,row0(:,bitnumber2));
                CCn1 = decision(:,row1(:,bitnumber2));
                maxn0 = max(CCn0(:));
                maxn1 = max(CCn1(:));
%LLRb(bitnumber2) = maxn0-maxn1;
LLRb(bitnumber2) = maxn1-maxn0;
end
            LLR=[LLRb LLRa];
Lcia(kk,:)=LLR;
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %               [CC,II]=max(decision(:));
    %               %element indexing starts from 1
    %               [n,m]=ind2sub(size(decision),II);
    %               mbits=decimalToBinaryVector(m-1,log2(M));
    %               nbits=decimalToBinaryVector(n-1,log2(M));
    %               b_hat=[mbits,nbits];
    %               u_hat = [u_hat b_hat];
    %               SISO Channel Decision
    %               [SISOCC1,II1] = max(abs(SISOy1s1));
    %               [SISOCC2,II2] = max(abs(SISOy1s2));
    %               SISOmbits = decimalToBinaryVector(II1-1,log2(M));
    %               SISOnbits = decimalToBinaryVector(II2-1,log2(M));
    %               SISOb_hat = [SISOmbits,SISOnbits];
    %               SISOu_hat = [SISOu_hat,SISOb_hat];
    end
            Lcib=reshape(Lcia',1,408);
            Lci=-Lcib/max(Lcib);
    %Lci=(1./Lcib);
    %         bit_errors_no_coding(ii)=bit_errors_no_coding(ii)+sum(abs(u-
u_hat'));
    %
bit_errors_no_coding_SISO(ii)=bit_errors_no_coding_SISO(ii)+sum(abs(u-
SISOu_hat'));
    %vhat = decodeProbDomain(u_hat, ParityMatrix, sigma, iter);
    %N0 = 1/(exp(Es_No_dB(ii)*log(10)/10));

    vhat = decodeLogDomainOurs(Lci,newH,No,iter);
    %vhat = decodeLogDomainSimpleOurs(Lci,newH,iter);

    %vhat = decodeProbDomainnew(u_hat, ParityMatrix, N0, iter);
            bit_errors_LDPC(ii) = bit_errors_LDPC(ii)+sum(abs(u-vhat'));
    if sum(abs(u-vhat'))>0
                packet_errors(ii) = packet_errors(ii)+1;
    end
    %         fprintf('%3d Eb/No=%2d bit errors uncoded=%4d  total bits=%5d
BER=%e\n',xxxx,
Eb_No_dB(ii),bit_errors_no_coding(ii),total_bits(ii),bit_errors_no_coding(i
i)/total_bits(ii));
            fprintf('%3d Eb/No=%2d bit errors coded=%4d  total bits=%5d
BER=%e\n',xxxx,
Eb_No_dB(ii),bit_errors_LDPC(ii),total_bits(ii),bit_errors_LDPC(ii)/total_b
its(ii));
            fprintf('%3d Eb/No=%2d packet errors coded=%4d  total
packets=%5d PER=%e\n',xxxx,
Eb_No_dB(ii),packet_errors(ii),xxxx,packet_errors(ii)/xxxx);
    %         fprintf('%3d Eb/No=%2d bit errors uncoded SISO=%4d  total
bits=%5d BER=%e\n',xxxx,
Eb_No_dB(ii),bit_errors_no_coding_SISO(ii),total_bits(ii),bit_errors_no_cod
ing_SISO(ii)/total_bits(ii));
    %         fprintf('%3d Es/No=%2d bit errors coded=%4d total bits=%5d
BER=%e\n',xxxx,
Es_No_dB(ii),bit_errors_LDPC(ii),total_bits(ii),bit_errors_LDPC(ii)/total_b
its(ii));
    end
    end
    %bercoded=(bit_errors_LDPC./total_bits);
    beruncoded=bit_errors_no_coding./total_bits;
    beruncodedSISO = bit_errors_no_coding_SISO./total_bits;
    bercoded = bit_errors_LDPC./total_bits;
    percoded=packet_errors./total_packets;
```

```
    semilogy(Eb_No_dB,beruncoded,Eb_No_dB,beruncodedSISO,Eb_No_dB,bercoded,E
b_No_dB,percoded)
    xlabel('Es/No')
    legend('Uncoded MIMO', 'Uncoded SISO', 'Coded MIMO','Packet Error Rate')
    grid
```

ΚΩΔΙΚΕΣ ΓΡΑΦΙΚΩΝ ΠΑΡΑΣΤΑΣΕΩΝ

1

```
    clearall
    clc

    [numinfo,titles]=xlsread('results.xlsx');
    titles=titles(1:2,:);
    awgndB=numinfo(1:8,1);
    rayleighdB=numinfo(12:31,1);
    awgndata=numinfo(1:8,1:11);
    rayleighdata=numinfo(12:31,2:end);

    semilogy(awgndB,awgndata(1:8,4),'-o',awgndB,awgndata(1:8,11),'-
d','LineWidth',1.4)
    xlabel('Eb/No')
    ylabel('BER')
    legend('SISO M=64 AWGN only', 'SISO M=16 AWGN only')
    grid
    figure
    semilogy(rayleighdB,rayleighdata(:,3),'-
*',rayleighdB,rayleighdata(:,4),'-o',rayleighdB,rayleighdata(:,7),'-
d',rayleighdB,rayleighdata(:,10),'-x',rayleighdB,rayleighdata(:,13),'-
s',rayleighdB,rayleighdata(:,16),'-^',rayleighdB,rayleighdata(:,19),'-
+',rayleighdB,rayleighdata(:,22),'-v',rayleighdB,rayleighdata(:,25),'-
h','LineWidth',1.4)
    xlabel('Eb/No')
    ylabel('BER')
    legend('SISO M=64', 'SISO M=64 Theory','MIMO M=64','SISO M=16','MIMO
LDPC simple 20i','MIMO LDPC simple 50i','MIMO LDPC 20i','MIMO LDPC
50i','MIMO LDPC PER 50i')
    grid
```

2

```
    closeall
    clearall

    ebnodbrayleigh=[0 0.5 1 1.5 2 2.5 4 6 8 10 12 14 16 18 20 22 24 26 28
30];

    ebnodbawgn=[0:6];

    bertheory=[0.245385 0.235385 0.225385 0.20 0.181741 0 0.128301 0.087349
0.0579548 0.0377821 0.0243455 0.0155688 0.0099075 0.00628516 0.00397929
0.00251621 0.0015898 0.00100396 0.000633803 0.000400041];

    siso64awgn=[0.163296569 0.088322829 0.048058439 0.018021915 0.00470006
0.000830813 8.53507e-05];

    siso16awgn=[0.177521008 0.121031746 0.073168973 0.038334865 0.016955412
0.004887372 0.001129661];
```

```
    siso64rayleigh=[0.248145204 0.226004599 0.212920364 0.196544086
0.181547006 0.168604942 0.128903413 0.08667859 0.058114604 0.038111578
0.024235882 0.015730481 0.010893246 0.006008471 0.004289216 0.00245098
0.001313306 0.000900012 0.000618304 0.00051830];

    siso16rayleigh=[0.235962567 0.22 0.21 0.20 0.194381599 0.18 0.150230681
0.094362745 0.065144479 0.045008913 0.029677534 0.017890368 0.010627958
0.007404775 0.003952304 0.002879614 0.001869455 0.001220613 0.000746795
0.000417179];

    mimo64=[0.16772655 0.128207456 0.111964618 0.08928931 0.071855385
0.057854392 0.025160064 0.00867803 0.001934475 0.000523977 0.000131382];

    mimoLDPCsimple20i=[0.137663399 0.064903234 0.025004951 0.007418788
0.001172784 0.000126538];

    mimoLDPCsimple50i=[0.137519873 0.061486323 0.028247891 0.006402754
0.000959991 7.3001e-05];

    mimoLDPC20i=[0.114398085 0.060802032 0.019389345 0.003684757 0.000424965
4.38719e-05];

    mimoLDPC50i=[0.084579595 0.051087623 0.014466419 0.002764131 0.000371049
2.90278e-05];

    per=[0.677966102 0.458333333 0.152298851 0.0285 0.005063733
0.000375665];

    figure

    semilogy(ebnodbawgn,siso16awgn,'-d',ebnodbawgn,siso64awgn,'-
o','LineWidth',1.4)
    title('Simulation of SISO Channel at AWGN')
    xlabel('Eb/No','FontSize',13)
    ylabel('BER','FontSize',13)
    legend({'SISO M=16 AWGN','SISO M=64 AWGN'},'FontSize',13 )
    grid


    figure

    semilogy(ebnodbrayleigh,bertheory,'--',ebnodbrayleigh,siso16rayleigh,'-
x',ebnodbrayleigh,siso64rayleigh,'-*',ebnodbrayleigh(1:11),mimo64,'-
o','LineWidth',1.4)
    title('Simulation of SISO Channel at Rayleigh')
    xlabel('Eb/No','FontSize',13)
    ylabel('BER','FontSize',13)
    legend({'Theoretical BER','SISO M=16 Rayleigh','SISO M=64
Rayleigh','MIMO M=64 Rayleigh'},'FontSize',13 )
    grid

    figure

    semilogy(ebnodbrayleigh,bertheory,'--',ebnodbrayleigh(1:11),mimo64,'-
o',ebnodbrayleigh(1:6),mimoLDPCsimple20i,'-
*',ebnodbrayleigh(1:6),mimoLDPC20i,'-d','LineWidth',1.4)
    title('Simulation of MIMO Channel at Rayleigh Uncoded & Coded for 20i')
```

```
    xlabel('Eb/No','FontSize',13)
    ylabel('BER','FontSize',13)
    legend({'Theoretical BER','MIMO M=64 Rayleigh','MIMO M=64 LDPC Simple
20i','MIMO M=64 LDPC 20i'},'FontSize',13 )
    grid

    figure

    semilogy(ebnodbrayleigh,bertheory,'--',ebnodbrayleigh(1:11),mimo64,'-
o',ebnodbrayleigh(1:6),mimoLDPCsimple50i,'-
*',ebnodbrayleigh(1:6),mimoLDPC50i,'-d','LineWidth',1.4)
    title('Simulation of MIMO Channel at Rayleigh Uncoded & Coded for 50i')
    xlabel('Eb/No','FontSize',13)
    ylabel('BER','FontSize',13)
    legend({'Theoretical BER','MIMO M=64 Rayleigh','MIMO M=64 LDPC Simple
50i','MIMO M=64 LDPC 50i'},'FontSize',13 )
    grid

    figure

    semilogy(ebnodbrayleigh(1:11),mimo64,'--
',ebnodbrayleigh(1:6),mimoLDPC20i,'-*',ebnodbrayleigh(1:6),mimoLDPC50i,'-
d','LineWidth',1.4)
    title('Comparsion for MIMO Coded 20i vs. 50i')
    xlabel('Eb/No','FontSize',13)
    ylabel('BER','FontSize',13)
    legend({'MIMO M=64 Rayleigh','MIMO M=64 LDPC 20i','MIMO M=64 LDPC
50i'},'FontSize',13 )
    grid

    figure

    semilogy(ebnodbrayleigh(1:6),per,'--','LineWidth',1.4)
    title('Simulation of PER Coded for 50i')
    xlabel('Eb/No','FontSize',13)
    ylabel('PER','FontSize',13)
    legend({'MIMO M=64 LDPC 50i PER'},'FontSize',13)

    grid
```

ΚΩΔΙΚΑΣ ΘΕΩΡΗΤΙΚΗΣ ΚΑΜΠΥΛΗΣ ΣΤΗ MATHEMATICA

```
M=64
For [ii= 0,ii<=30, ii=ii+2,
 EbNodB = ii;
 EsNodB = ii+10*Log10[Log2[M]];
 g = 10^(0.1*EsNodB);
 Ps = Sum[Binomial[M-1,i]*(-1)^(i+1)/(i+1+i*g),{i,1,M-1}];
 Ps1 = 1-NIntegrate[Exp[-y/(1+g)]/(1+g)*(1-Exp[-y])^(M-1),{y,0,Infinity}];
 Pb1 = 2^(Log2[M]-1)/(2^Log2[M]-1)*Ps1;
    (*Print[ii,",",EbNodB,",",Ps,",",Ps1,",",Pb1,";"]*)Print[Pb1]]
```

ΠΙΝΑΚΕΣ EXCELME ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Πίνακας 1: AWGN αποτελέσματα χωρίς LDPC

AWGN	M=64 SISO				M = 64 MIMO			M=16 SISO		
Eb/No	bit errors	total bits	BER	BER Theory	bit errors	total bits	BER	bit errors	total bits	BER
0	1066	6528	0.163297					1014	5712	0.17752101
1	1009	11424	0.088323					1037	8568	0.12103175
2	1000	20808	0.048058					1015	13872	0.07316897
3	1000	55488	0.018022					1001	26112	0.03833487
4	1001	212976	0.0047					1010	59568	0.01695541
5	1002	1206048	0.000831					1005	205632	0.00488737
6	1002	11739792	8.54E-05					1002	886992	0.00112966
7								1000	6020040	0.00016611

Πίνακας 2:Rayleigh αποτελέσματα με LDPC

Rayleigh	MIMO LDPC simple 20i			MIMO LDPC simple 50i			MIMO LDPC 20i			MIMO LDPC 50i		
Eb/No	bit errors	total bits	BER	bit errors	total bits	BER	bit errors	total bits	BER	bit errors	total bits	BER
0	2022	14688	0.137663	2076	15096	0.13752	2007	17544	0.114398	2036	24072	0.08458
0.5	2039	31416	0.064903	2032	33048	0.061486	2059	33864	0.060802	2001	39168	0.051088
1	4040	161568	0.025005	2063	73032	0.028248	2041	105264	0.019389	2054	141984	0.014466
1.5	2028	273360	0.007419	2035	317832	0.006403	2001	543048	0.003685	2039	737664	0.002764
2	2036	1736040	0.001173	1007	1048968	0.00096	1058	2489616	0.000425	867	2336616	0.000371
2.5	1486	11743464	0.000127	1796	24602400	7.3E-05	983	22406136	4.39E-05	599	20635416	2.9E-05

Πίνακας 3:Rayleigh αποτελέσματα χωρίς LDPC

Rayleigh	M=64 SISO				M = 64 MIMO			M=16 SISO		
Eb/No	bit errors	total bits	BER	BER Theory	bit errors	total bits	BER	bit errors	total bits	BER
0	3746	15096	0.248145	0.245385	2532	15096	0.167727	1059	4488	0.235963
0.5	7469	33048	0.226005		4237	33048	0.128207			
1	15550	73032	0.21292		8177	73032	0.111965			
1.5	62468	317832	0.196544		28379	317832	0.089289			
2	190437	1048968	0.181547	0.181741	75374	1048968	0.071855	1031	5304	0.194382
2.5	161796	959616	0.168605		55518	959616	0.057854			
4	2840	22032	0.128903	0.128301	1006	39984	0.02516	1042	6936	0.150231
6	5234	60384	0.086679	0.087349	1002	115464	0.008678	1001	10608	0.094363
8	13444	231336	0.058115	0.0579548	1000	516936	0.001934	1010	15504	0.065144
10	32654	856800	0.038112	0.0377821	1002	1912296	0.000524	1010	22440	0.045009
12	86354	3563064	0.024236	0.0243455	772	5876016	0.000131	1005	33864	0.029678
14	80065	5089800	0.01573	0.0155688				1000	55896	0.01789
16	1000	91800	0.010893	0.0099075				1006	94656	0.010628
18	1010	168096	0.006008	0.0062852				1000	135048	0.007405
20	1001	233376	0.004289	0.0039793				1003	253776	0.003952
22	1001	408408	0.002451	0.0025162				1001	347616	0.00288
24	1002	762960	0.001313	0.0015898				1003	536520	0.001869
26	1001	1112208	0.0009	0.001004				1001	820080	0.001221
28	1001	1618944	0.000618	0.0006338				1000	1339056	0.000747
30				0.0004				1001	2399448	0.000417

Πίνακας 4:PER αποτελέσματα για LDPC 50i

Rayleigh	LDPC Packets 50i		
Eb/No	packet errors	total packets	PER
0	40	59	0.677966
0.5	44	96	0.458333
1	53	348	0.152299
1.5	57	2000	0.0285
2	29	5727	0.005064
2.5	19	50577	0.000376

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Evgeny Tsimbalo, Robert J. Piechocki, Andrew Nix, and Paul Thomas ‘’*Non-Coherent MIMO Scheme Based on OFDM-MFSK*’’ IEEE WIRELESS COMMUNICATIONS LETTERS, VOL. 6, NO. 3, JUNE 2017, pg. 406-409

[2] Matthias Wetz, Ivan Peri_sa, Werner G. Teich, J¨urgen Lindner ‘’*Robust transmission over fast fading channels on the basis of ofdm-mfsk*’’

[3] Αθανάσιος Ιωσηφίδης, *Επίκουρος καθηγητής ΑΤΕΙΘ,* Σημειώσεις στο μάθημα ‘’Συστήματα Ασύρματων Επικοινωνιών’’ ΑΤΕΙ Θεσσαλονίκης.

[4] Moritz Borgmann, *Student Member, IEEE,* and Helmut Bolcskei, *Senior Member, IEEE* ‘’*Noncoherent Space-Frequency Coded MIMO-OFDM*’’ IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 23, NO. 9, , pg. 1799-1810, SEPTEMBER 2005

[5]] Jack Keil Wolf, ‘’*AN INTRODUCTION TO ERROR CORRECTING CODES Part 3’’* ECE 154 C Spring 2010

[6] William E. Ryan, ‘’*An Introduction To LDPC Codes*’’, Department of Electrical and Computer Engineering, The University of Arizona, August 19, 2003

[7] Sibonginkosi Ntuli, ‘’*BIT FLIPPING DECODING FOR BINARY PRODUCT CODES*’’, University Of The Witwatersarand Johannesburg, September 2013

[8] Ashwini Wanjari, Sandeep Kakde and Atish Khobragade, ‘’*Error Performance of LDPC Decoder using Bit Flip Algorithm*’’ International Conference on Communication and Signal Processing, India, April 6-8, 2016,

[9] Karen Rose, Scott Eldridge, Lyman Chapin ‘’*THE INTERNET OF THINGS: AN OVERVIEW*’’, The Internet Society (ISOC), October 2015

[10] http://www.inference.org.uk/mackay/codes/

[11] https://sites.google.com/site/bsnugroho/

[12] http://news.mit.edu/2010/gallager-codes-0121

[13] D.J.C. MacKay and R.M. Neal ‘’ *Near Shannon limit performance of low density parity check codes*’’ ELECTRONICS LETTERS 13TH MARCH 1997 VOL.33 NO.6 pg. 457-458.

[14] Ghaith Al-Juboori, Evgeny Tsimbalo, Angela Doufexi and Andrew R. Nix ‘’*A Comparison of OFDM and GFDM-based MFSK Modulation Schemes for Robust IoT*

Applications'' Communication Systems and Networks Group-Department of Electrical and Electronic Engineering, University of Bristol, Bristol, United Kingdom.

[15] John G. Proakis and Masoud Salehi, *Digital Communications 5th Edition*, New York, 2008, McGraw-Hill Higher Education, pg. 863-864

Τα σχήματα και τα διαγράμματα έγιναν με τη βοήθεια της ιστοσελίδας:

https://draw.io