الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

BELKACEM BENMAHAMMED

Les turbo-codes dans un contexte OFDM

Mémoire présenté au
Département de l'élelectronique
Faculté de Téchnologie
Université de BLIDA1
Master Télécommunication et Réseaux

Supervisé par **Hocine AIT SAADI & Abdellah BERDAI**

2017

Résumé

Le multiplexage de la division des fréquences orthogonales (OFDM) a été appliqué avec succès sur une grande variété d'applications de communication numérique au cours des dernières années comme le wimax, le wifi et la LTE *etc*.... L'OFDM est une méthode appropriée pour une transmission élevée du débit de données avec des méthodes de correction d'erreur directe (FEC) sur des canaux sans fil.

Les canaux sélectifs en fréquence et sélectifs en temps sont les canaux les plus difficiles à traiter. L'utilisation d'une technique mutliporteuses comme l'OFDM et d'un codage canal puissant tel que les turbo codes sont parmi les meilleures solutions à adopter. De plus, l'utilisation du codage turbo dans l'OFDM, permet atteindre les performances souhaitées à des débits de données plus élevés.

Dans ce projet, nous analysons les performances des systèmes avec un codage turbo sur les modèles des canaux radio mobile du système LTE tels que EPA, EVA et ETU. Ces canaux sont sélectifs en fréquence et variables dans le temps avec des fréquences Doppler allant de 5Hz pour EPA à 300 Hz pour ETU. Au récepteur, la technique d'estimation et d'égalisation du canal MMSE est utilisée avec un décodage canal itératif réalisé avec l'algorithme BCJR.

Les simulations réalisées en Matlab ont montré l'amélioration en BER apportée sur les signaux OFDM par les turbo-codes et l'égalisation MMSE à travers les différents canaux hostiles.

Abstract

Orthogonal frequency division multiplexing (OFDM) has been successfully applied to a wide variety of digital communication applications in recent years. OFDM is a suitable candidate for high data rate transmission with Forward error correction (FEC) methods on wireless channels.

Frequency-selective and time-selective channels are the most difficult to be treated. The use of a mutli-carrier technique such as OFDM and a channel coding such as turbo codes are among the best adopt. In addition, the use of turbo coding in OFDM, achieves the desired performance at higher data rates.

In this project, the OFDM system throughput was improved by adding turbo coding. The use of turbo coding in OFDM is useful for the desired performance at higher data rates. The simulation is done on multi-path fading channel models (EPA, EVA and ETU that are produced in broadband transmissions) with white Gaussian noise, applying the BCJR iterative decoding algorithm to improve performance Of the BER.

A script was developed in Matlab to simulate the various steps throughout the OFDM system chain, beginning with data encoding, modulation (QPSK and 64QAM), channel and noise effects, estimation and equalization of the channel (LS and LMMSE) and finally an iterative process of decoding the symbols.

At the end of this thesis an analysis of the simulation results obtained and an overall conclusion of the project have been established.

Remerciements

Je tiens tout d'abord à remercier l'encadreur de ce projet, Mr. AIT SAADI Hocine, pour m'avoir fait confiance malgré les connaissances plutôt légères que j'avais sur l'OFDM et les turbo codes, puis pour m'avoir guidé, encouragé et conseillé pendant presque six mois tout en me laissant une grande liberté et en me faisant l'honneur de me déléguer plusieurs responsabilités dont j'espère avoir été à la hauteur.

Mes remerciements vont également à Dr. BREDAI Abdellah, pour la gentillesse et la patience qu'il a manifestées à mon égard durant ce projet de fin d'études, pour tous les conseils et les programmes qu'il a bien voulu m'envoyer malgré ces occupations professionnelles et familiale.

Je remercie également mon frère BENMAHAMMED Amine et mon cher amis DAHMANI Ahmed pour leurs sacrifices en temps durant des heures et des heures, exploitants leurs propre PC pour l'exécution des programmes Matlab exigeant des machines très performantes.

Je remercie ma petite famille pour leur patience durant cette période, pour leur compréhensibilité et pour le bon climat de travail qu'ils l'avaient créée.

Finalement je remercie mes très chers parents pour leurs support et leurs prières que je sois toujours à la hauteur.

Table des matières

Résumé ii

Abstract iii

Remerciements iv

Table des matières v

Liste des figures viii

Liste des tableaux x

Acronymes xi

Notation xii

1 Introduction 1

2 Système OFDM : 3

2.1 Introduction : . . . 3

2.2 Caractéristiques du canal : . . . 3

2.2.1 Effet des trajets multiples : . . . 3

2.2.2 Effet Doppler : . . . 4

2.2.3 Canaux sélectifs : . . . 4

2.3 Modulations Multi-porteuses : . . . 5

2.4 La chaîne de transmission OFDM : . . . 7

2.4.1 Génération du signal OFDM : . . . 7

2.4.2 L'orthogonalité : . . . 12

2.4.3 Le préfixe cyclique : . . . 13

2.4.4 Comment choisir le nombre de porteuses : . . . 14

2.4.5 Schéma bloc d'émission-réception d'un système OFDM : . . . 14

2.4.6 Correction d'erreurs : . . . 14

2.5 Avantages et inconvénients des systèmes OFDM : . . . 15

2.5.1 Avantages : . . . 15

2.5.2 Inconvénients : 16

3 Estimation et égalisation du canal : 17
3.1 Estimation du canal : 17
3.1.1 L'estimateur LS (least squares) : 19
3.1.2 L'estimateur LS modifié (least Mean squares) : 19
3.1.3 L'estimateur LMMSE (Least Minimum Mean Square Error) : . 20
3.1.4 L'estimateur LMMSE modifié (Least Minimum Mean Square Error Modified) : 21
3.1.5 L'estimateur ML(Maximum Likehood) : 23
3.2 L'interpolation : 24
3.2.1 L'interpolation linéaire (LI) : 24
3.2.2 L'interpolation du second ordre (SOI) : 24
3.2.3 L'interpolation par splines cubiques (SCI) : 24
3.2.4 L'interpolation passe-bas (LPI) : 25
3.3 L'égaliseur ZF : 25

4 Turbo Codes : 26
4.1 Introduction : 26
4.2 Codes convolutionnels : 26
4.3 L'entrelaceur : 30
4.3.1 Entrelacement polynomial : 31
4.4 Architecture du codeur Turbo : 32
4.5 Décodage Turbo : 33
4.5.1 Décodeur convolutionnel : 33
4.5.2 Viterbi : 34
4.5.3 SISO : 39
4.5.4 BCJR : 41
4.5.5 SOVA : 44
4.6 Architecture du décodeur Turbo : 45

5 Résultats des simulations : 48
5.1 Paramètres de simulation : 48
5.1.1 L'effet du nombre de sous-porteuses : 49
5.1.2 L'effet de modulation : 50
5.1.3 L'effet du canal : 50
5.1.4 L'effet de la méthode d'estimation : 52
5.2 La construction du programme : 53
5.2.1 Les boucles : 53
5.2.2 Génération des données : 53
5.2.3 Codage des données : 53

5.2.4 Formation des symboles binaires : 54
5.2.5 Conversion binaire-décimal : 54
5.2.6 La modulation : 54
5.2.7 Insertion des pilotes : 55
5.2.8 L'IFFT : 55
5.2.9 L'ajout du préfixe cyclique : 55
5.2.10 Le passage au canal : 56
5.2.11 L'ajout du bruit : 57
5.2.12 Enlèvement du préfixe cyclique : 57
5.2.13 FFT : 57
5.2.14 L'estimation LMMSE du canal : 57
5.2.15 La démodulation : 59
5.2.16 Conversion décimal-binaire : 59
5.2.17 Récupération des données codées : 59
5.2.18 Décodage itératif des données récupérées : 59
5.3 Analyse des résultats : 62

6 Conclusion : 65

Bibliographie 68

Liste des figures

2.1 Diagramme simplifié de la modulation OFDM. 7
2.2 Illustration des sous-porteuses d'un systeme OFDM. 12
2.3 L'orthogonalité des sous porteuses OFDM 12
2.4 Illustration du préfixe OFDM. 13
2.5 Illustration de la chaîne de transmission OFDM. 15

3.1 Schéma indiquant le rôle de l'égalisation dans un système de communication. 17
3.2 Insertion des symbole pilote pour une meilleure estimation du canal . . 18
3.3 Shémas d'un estimateur LMMSE modifié. 23

4.1 Illustration d'un codeur convolutif d'un taux de codage 1/2. 27
4.2 Illustration d'un codeur convolutionnel systématique. 28
4.3 Illustration d'un codeur RSC 28
4.4 Illustration d'un treillis du codeur RSC avec le chemin en bleu. 30
4.5 Illustration d'un exemple d'entrelaceur. 31
4.6 Illustration de la corruption des bits dans une séquence due au canal . 31
4.7 Illustration de la structure du codeur turbo 1/3. 32
4.8 décodage par le bit systématique r_i et le bit de parité r_p sortant la séquence décodée u. 34
4.9 Deux chemins à travers le treillis 38
4.10 Un décodeur SISO 40
4.11 Illustration de la structure d'un décodeur Turbo. 46

5.1 L'effet du nombre de sous-porteuses sur la performance du système OFDM. 49
5.2 L'effet de la taille de la constellation sur la performance du système OFDM. 50
5.3 L'effet canal sur la performance du système OFDM. 51
5.4 L'effet de la méthode d'estimation sur la performance du système OFDM. 52
5.5 Constellations des symboles modulés en QPSK et 64QAM. 55
5.6 La réponse fréquentielle du canal ETU. 56
5.7 Comparaison du canal estimé avec le vrai canal. 58
5.8 Turbo décodage des symboles OFDM modulés en QPSK via un canal EPA. 63

5.9 Turbo décodage des symboles OFDM modulés en 64QAM via un canal ETU. 63

Liste des tableaux

2.1 Nombre de bits par constellation 6

4.1 Le tableau du polynôme générateur du codeur RSC 29
4.2 Les paramètres de la branche du schéma de treillis 37
4.3 Décision soft à 8 bit 39

Acronymes

BCJR	L.Bahl, J.Cocke, F.Jelinek, and J.Raviv
BER	Bit error rate
CP	Préfixe cyclique
DFT	La transformée de Fourier discrète
FEC	Forward error correction
FFT	La transformée de Fourier rapide
IEP	Interférences entre porteuses
IES	Interférences entre symboles
LLR	Log-likelihood ratio
LMMSE	Linéaire à erreur quadratique moyenne minimale
LMMSEM	Linéaire à erreur quadratique moyenne minimale modifié
LS	Moindres carrés
LSM	Moindres carrés modifié
MAP	Maximum a posteriori
ML	Vraisemblance maximale
OFDM	Multiplexage par division en fréquences orthogonales
PAPR	Rapport entre les puissances de crête et moyenne
PCCC	Programmable Controller Communications Code
PSAM	Modulation assistée par symboles pilotes
PSK	Modulation par sauts de phase
QAM	Modulation d'amplitude en quadrature
RSC	Code récursive systématique convolutionnel
SISO	Single Input, Single Output
SOVA	Soft Output Viterbi Algorithm
SVD	Décomposition en valeurs singulières
ZF	Forçage à zéro

Notation

Variables utilisées

a	la fonction du canal
B	Largeur de bande utilisée
B_c	Bande de cohérence
c	Chemin de treillis
d	La distance Hamming
E	L'énergie d'un bit
E_b/N_0	Rapport entre énergie par bit et densité spectrale du bruit
$\mathbf{F}$	La matrice de la DFT
f_c	Fréquence porteuse
f_d	Fréquence Doppler
f_m	Fréquence Doppler normalisée
F_m	Fréquence Doppler maximale
F_s	La fréquence d'échantillonnage
g_n	Réponse impulsionnelle du canal
$\mathbf{H}_c$	La matrice circulante du canal
h_n	Réponse fréquentielle du canal
K_f	L'écart fréquentiel entre les sous-porteuses pilotes
K_t	L'écart temporel entre les symboles OFDM pilotes
L	Nombre des trajets du canal
N	Nombre de sous-porteuses
n_k	Bruit blanc additif gaussien
N_g	Nombre d'échantillons dans l'intervalle de garde
N_p	Nombre de sous-porteuses pilotes dans un symbole OFDM
N_s	Nombre d'échantillons dans un symbole OFDM
p	Le rang d'approximation
R_p	Taux d'insertion des symboles pilotes
$S(\tau)$	Profil des puissances des retards
T_c	Temps de cohérence
T_e	Le temps d'échantillonnage

T_g	Durée de l'intervalle de garde
T_s	Durée totale d'un symbole OFDM
T_u	Durée utile d'un symbole OFDM
α	Facteur d'atténuation du canal
β	Constante qui dépende de la constellation du signal
Δf	L'écart fréquentiel entre deux sous-porteuses voisines
ΔF	Le décalage fréquentiel
f	Valeurs associées à la longueur de la séquence
G	Polynome générateur
$\bar{\gamma}$	Rapport SNR moyen
K	La longueur de la séquence d'entrée
L	La longueur de contrainte moins un
λ	Longueur d'onde transmise
M	Les paramètres de la branche de treillis
ν	Vitesse de déplacement du récepteur
P	Probabilité
Φ_{nn}	Densité spectrale de puissance du bruit
r_i	Le bit systématique
r_p	Le bit de parité
S	Les transitions d'un état à un autre état
σ_n^2	Variance du bruit additif
σ_τ	L'étalement rms des retards
τ	Retard de propagation
$\bar{\tau}$	Le retard moyen
$\bar{\tau^2}$	Le moment d'ordre 2 du retard
τ_m	L'étalement maximal de la RIC
θ	Déphasage angulaire des trajets
U	Le bit de sortie codé
V	Valeur attribuée à chaque état dans le treillis
y	La séquence indiquant l'estimation de VA de la séquence transmise
Z	La séquence de vérification de parité de la séquence d'entrée
$E\{\cdot\}$	Espérance mathématique
$\Re\{\cdot\}$	Partie réelle
$\Im\{\cdot\}$	Partie imaginaire
x	Scalaire
$\mathbf{x}$	Vecteur
$\mathbf{X}$	Matrice
$(\cdot)^*$	Conjugué
$(\cdot)^T$	Transposé
$(\cdot)^H$	Transposé conjugué

$\hat{(\cdot)}$	Valeur estimée
$\otimes$	Produit de convolution

Chapitre 1

Introduction

Dans une transmission de type série, les différents symboles sont transmis séquentiellement les uns à la suite des autres. Or, les performances des systèmes séries sont parfois sérieusement limitées, précisèment dans un milieu à trajets multiples, où les systèmes séries nécessitent l'utilisation des circuits d'égalisation complexes afin de lutter contre l'interférence entre symboles (IES). Le débit binaire est alors limité par la capacité des systèmes d'égalisation à suivre les variations du canal.

D'autre part, les systèmes parallèles consistent essentiellement à transmettre à bas débit binaire sur un grand nombre de sous-canaux, plutôt que de transmettre à haut débit binaire sur une seule porteuse. A la fin des années 50, plusieurs chercheurs ont proposé d'utiliser les systèmes de communications parallèles. Ces systèmes nécessitent cependant la conception d'un banc de plusieurs modulateurs qui fonctionne en même temps à des fréquences différentes. En raison de la complexité de sa mise en oeuvre, les systèmes parallèles ont été abandonnés au profit des systèmes séries plus simples.

Il a fallu attendre les années 80 pour que les techniques parallèles fassent un retour en force, grâce au développement des technologies numériques de traitement de signal et aux progrés récents des circuits électroniques et des technologies à haute densité d'intégration. Un canal à trajets multiples présente une réponse fréquentielle qui n'est pas plate, mais comportant des creux (évanouissements) dus aux échos et réflexions du signal transmis par les obstacles qui se trouvent dans l'environnement de transmission. Un trés grand débit impose une grande bande passante et si cette bande passante couvre une partie du spectre comportant des évanouissements, il peut y avoir perte totale de l'information à la fréquence correspondante ; le canal est alors dit sélectif en fréquence. D'autre part, lorsque les symboles sont transmis successivement à haut débit dans un canal sélectif en fréquence, il en résulte des interférences entre les symboles adjacents.

Ces distorsions sont complexes à corriger et font appel à des techniques d'égalisation. Pour remédier à ce désagrément, l'idée est de répartir l'information sur un grand nombre de sous-porteuses, créant ainsi des sous-canaux trés étroits pour lesquels la réponse fréquentielle du canal peut être considérée comme constante. Ainsi, pour ces sous-canaux, le canal est non-sélectif en fréquence, et s'il y a un évanouissement dans la bande utilisée, il n'affectera que certaines fréquences et le système peut récupérer l'information perdue sur les autres fréquences porteuses qui n'auront pas été affectées. Pour que les fréquences des sous-porteuses soient les plus proches possibles et ainsi transmettre un maximum d'information sur une bande de fréquences donnée, l'OFDM utilise des sous-porteuses orthogonales entre elles. Pour cela, les signaux des différentes sous-porteuses se chevauchent mais, grâce à l'orthogonalité, n'interfèrent pas entre elles.

Chapitre 2

Système OFDM :

2.1 Introduction :

Un des problèmes majeurs en télécommunications est d'adapter l'information à transmettre au canal de propagation. Pour des canaux sélectifs en fréquence, une technique est l'utilisation de modulations multi-porteuses dans laquelle un bloc d'information est modulé par une transformée de Fourier. Cette technique connue sous le nom d'OFDM (Orthogonal Frequency Division Multiplexing) a le grand mérite de transformer un canal multi-trajet large bande en un ensemble de sous-canaux mono-trajet très simples à égaliser. De plus, l'utilisation de redondance cyclique à l'émission permet de réduire la complexité des terminaux grâce à l'utilisation d'algorithmes à base de FFT rapides.

2.2 Caractéristiques du canal :

2.2.1 Effet des trajets multiples :

La propagation des ondes dans un canal se fait par voie directe, mais plus souvent par réflexion, réfraction et diffusion des ondes par les obstacles artificiels et naturels qui entourent le récepteur. Les trajets multiples sont le résultat des répliques du signal original qui arrivent au récepteur à des instants différents. Ainsi, le signal reçu sera composé d'une somme de plusieurs répliques. Ces derniers suivront des trajets différents, de ce fait chaque trajet sera affecté d'une atténuation, d'un retard et d'un déphasage

différent.

Un canal multi-trajet est dit sélectif en fréquence lorsque la bande disponible couvre une partie du spectre comportant des évanouissements, ce qui engendre la perte de l'information transmise pour la fréquence correspondante.

2.2.2 Effet Doppler :

Dans un environnement radio mobile, le récepteur est souvent en mouvement par rapport à l'émetteur. Ce mouvement introduit un décalage de fréquence dans le contenu spectral du signal reçu. Ce décalage de fréquence, appelé effet Doppler, dépend de la vitesse du mobile, de la longueur d'onde (fréquence porteuse) et de l'angle d'incidence du trajet par rapport à la trajectoire de déplacement. L'e ?et Doppler peut poser des problèmes significatifs si la technique de transmission est sensible aux décalages de fréquences (le cas des systèmes OFDM).

2.2.3 Canaux sélectifs :

Le bruit dû aux imperfections des systèmes et la nature physique des composants affectent la transmission du signal émis. La déformation du signal au cours de la propagation est également une autre contrainte physique. Elle impose une bonne séparation temporelle des informations émises afin qu'elles restent bien séparées à la réception (pour évite l'interférence entre symboles). Les signaux réfléchis par les immeubles, les voitures ou le sol provoquent un phénomène nommé « affaiblissement par trajets multiples » : selon la longueur des différents chemins parcourus, le signal dévié arrive à l'émetteur plus ou moins longtemps après le signal principal, donc déforme plus ou moins celui-ci (les pertes sont plus faibles pour les basses fréquences et plus importantes pour les hautes fréquences).

Ce phénomène d'évanouissement ou « fading » résulte des variations aléatoires des phases du signal dans le temps. Elles peuvent engendrer des signaux s'ajoutant de façon destructive en réception. Le signal résultant sera alors très faible ou nul. Les signaux multiples peuvent aussi s'ajouter de manière constructive, le signal résultant sera alors plus puissant que le trajet direct. Le canal de propagation peut être modélisé par une

réponse impulsionnelle donnée par :

$$h(t) = \sum_{i=0}^{L-1} a_i g(t - \tau_i) \tag{2.1}$$

— g(t) est le filtre de mise en forme.

— a_i sont les gains du canal de propagation.

— τ_i sont les retards du canal de propagation.

Le signal reçu y(t) est alors le filtrage du signal émis x(t) par le canal de propagation h(t) et peut donc s'écrire :

$$y(t) = \int_{-\infty}^{+\infty} h(\tau)x(t - \tau)d\tau + n(t) \tag{2.2}$$

— n(t) est un bruit gaussien dû aux imperfections du système.

On parle de canaux sélectifs en fréquence quand le signal transmis $x(t)$ occupe une bande de fréquence [-B/2,B/2] plus grande que la bande de cohérence du canal de propagation $h(t)$ (l'inverse du temps de retard maximum du canal de propagation T_{max}) où les composantes fréquentielles de $x(t)$ hors la bande de cohérence subissent des atténuations différentes [1].

2.3 Modulations Multi-porteuses :

Dans les systèmes de transmission numérique haute débit, le signal reçu à un instant t peut s'exprimer comme une somme pondérée du signal émis au même instant et des signaux émis aux instants précédents, multiple de la période d'échantillonnage :

$$y(t) = \sum h(i)x(t - i) \tag{2.3}$$

Lorsque la période d'échantillonnage T est trop petite par rapport au retard T_r (ce qui est le cas pour les transmissions à haut débit), le nombre de coefficients $h(i)$ à

déterminer peut être grand et l'inversion du système devient complexe. La transmission de débits élevés en présence de trajets multiples peut donc rapidement augmenter la complexité et par suite le coût des terminaux.

Les bits sont transmis sous forme de symboles et non tel quel. Le nombre de bits inclus dans chaque symbole désigne la taille de la constellation. Plus cette taille sera grande et plus le débit sera élevé. Les constellations usuelles sont données sous forme de puissance de 2.

Constellation	Nombre de bits
BPSK	1
QPSK	2
8PSK	3
16-QAM	4
32-QAM	5
64-QAM	6

Tableau 2.1 – Nombre de bits par constellation

L'intérêt des modulations multi-porteuses (Multi-Carrier Modulation) est de placer l'information dans une fenêtre temps-fréquence telle que sa durée soit bien plus grande que le temps de retard maximum du canal de propagation. La modulation multi porteuse est basée sur le principe de transformer l'étape d'égalisation dans le domaine temporel par une égalisation simplifiée dans le domaine fréquentielle pour retrouver le signal émis.

L'application de ce principe sur l'équation du signal reçu donne :

$$Y(f) = H(f)X(f) + N(f) \tag{2.4}$$

$Y(f)$, $H(f)$, $X(f)$ et $N(f)$ représente les transformées de Fourier des signaux $y(t)$, $h(t)$, $x(t)$ et $n(t)$ respectivement.

En émission, le signal fréquentiel $X(f)$ est transmis sur un certain nombre de porteuses à l'aide d'une transformée de Fourier inverse. En réception, le signal est démodulé à l'aide d'une transformée de Fourier.

En d'autres termes, chaque composante du signal est multipliée par un coefficient correspondant au gain fréquentiel du canal et le récepteur égalise facilement le canal puisqu'il suffit de diviser chaque signal reçu par le gain correspondant.

De ce fait, à la réception, l'interférence entre symboles est supprimée et les symboles émis ne subissent qu'une atténuation. Chaque sous-canal peut être alors considéré comme une transmission mono-trajet dotée de son propre rapport signal sur bruit et de largeur $\Delta?f$.

2.4 La chaîne de transmission OFDM :

Les systèmes OFDM subdivisent le canal en N sous canaux (porteuses) dont les fréquences centrales sont espacées d'un multiple de l'inverse de la période symbole $1/T$, ces systèmes reposent sur le principe de transmission par bloc. La modulation d'un bloc de symboles est réalisée par une transformée de Fourier inverse.

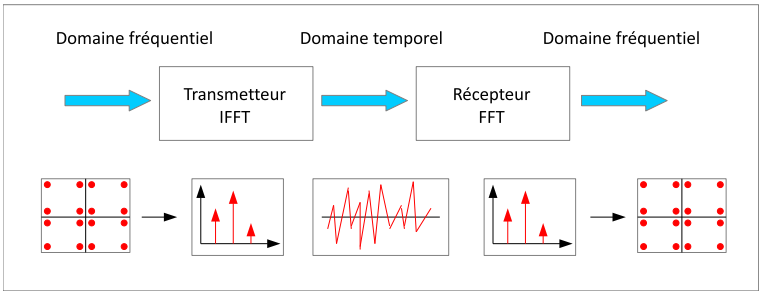

Figure 2.1 – Diagramme simplifié de la modulation OFDM.

2.4.1 Génération du signal OFDM :

La DFT d'un signal $x(n)$ est définie comme suit :

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp^{-j2\pi kn/N}, 1 < k < N \tag{2.5}$$

Et son inverse associé IDFT est dénoté par :

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \exp^{j2\pi kn/N}, 1 < n < N \tag{2.6}$$

Où N représente la taille de la DFT/IDFT.

La matrice de la DFT est composée d'exponentielles complexes définie comme suit :

$$F = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \exp^{-j2\pi/N} & \exp^{-j4\pi/N} & \ddots & \exp^{-j2\pi(N-1)/N} \\ 1 & \exp^{-j4\pi/N} & \exp^{-j8\pi/N} & \ddots & \exp^{-j4\pi(N-1)/N} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \exp^{-j2\pi(N-1)/N} & \exp^{-j4\pi(N-1)/N} & \dots & \exp^{-j2\pi(N-1)(N-1)/N} \end{pmatrix} \tag{2.7}$$

Pour les systèmes OFDM, une autre forme de la DFT est utilisée, appelée la transformée de Fourier rapide (FFT).

Soient X_n les symboles de données ayant des valeurs complexes prises dans un alphabet fini selon la constellation de la modulation choisie, Ts représente la durée d'un symbole OFDM, et les N sous-porteuses sont distantes de Δf. Les symboles d'entrée X_n sont organisés en blocs de taille $N \times 1$. Le kieme bloc est donné par :

$$X_k = (X_{k,0}, X_{k,1}, \dots, X_{k,N-1})^T \tag{2.8}$$

Dans ce qui suit, on omet l'indice k pour la simplicité, avec l'application de l'IDFT au bloc de symboles X, on obtient le vecteur :

$$x = F^H X = (x_0, x_1, \dots, x_{N-1})^T \tag{2.9}$$

Où F désigne la matrice DFT de taille $N \times N$, donnée par la relation (2.7). La matrice FH désigne donc l'opération inverse (IDFT), avec $(\cdot)T$ qui indique le transposé conjugué.

Dans les systèmes OFDM, c'est l'ajout d'une extension cyclique à chaque bloc qui permet de prendre en compte le comportement sélectif du canal. A partir du vecteur x de taille $N \times 1$, on forme un nouveau vecteur x de taille $Ns \times 1$, tel que :

$$\tilde{x} = (x_{N-D}, \dots, x_{N-1}, x_0, x_1, \dots, x_{N-1})^T \tag{2.10}$$

Soit $h = (h_0, h_1, \ldots, h_{(L-1)})^T$ et $n = (n_0, n_1, \ldots, n_{(L-1)})^T$, la réponse impulsionnelle du canal et le bruit AWGN, respectivement. On peut donc définir :

$$H = DFT_N(h) = Fh, \;\; N = Fn \tag{2.11}$$

Au récepteur, le vecteur reçu est donné par :

$$\tilde{y} = \tilde{x} \otimes h + \tilde{n} \tag{2.12}$$

Où $\otimes$ dénote l'opération de convolution. On supprime le préfixe cyclique, donc les D premiers symboles qui contiennent de l'interférence entre les blocs sont supprimés, c'est-à-dire :

$$y = (y_D, y_{D+1}, \ldots, y_{N+D-1})^T \tag{2.13}$$

Sous la supposition que les interférences sont complètement éliminés, on peut dériver le vecteur reçu y de taille $N \times 1$ comme suit :

$$y = \tilde{H}\tilde{x} + n \tag{2.14}$$

Où $\tilde{H}$ est une matrice de taille $N \times N_s$, qui représente l'effet du canal. Elle est donnée par :

$$\tilde{H} = \begin{pmatrix} h_0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ h_{L-1} & h_{L-2} & \cdots & h_0 & \cdots \\ 0 & h_{L-1} & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \cdots & h_0 & 0 \\ 0 & 0 & \cdots & h_1 & h_0 \end{pmatrix} \tag{2.15}$$

Puisque les D premiers éléments de $\tilde{x}$ sont identiques aux D derniers, on peut écrire :

$$y = H_c\tilde{x} + n \tag{2.16}$$

Où H_c est une matrice de taille $N \times N$, obtenue à partir de (IV.11), en superposant les D premières colonnes aux D dernières, ce qui donne :

$$H_c = \begin{pmatrix} h_0 & 0 & . & . & 0 & h_{L-1} & . & . & . & h_1 \\ h_1 & h_0 & . & . & . & 0 & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & h_{L-1} \\ h_{L-1} & h_{L-2} & . & . & h_0 & . & . & . & 0 & 0 \\ 0 & h_{L-1} & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & h_{L-2} & . & . & . & h_0 & 0 \\ 0 & 0 & . & . & h_{L-1} & . & . & . & h_1 & h_0 \end{pmatrix} \tag{2.17}$$

H_c étant une matrice circulante, on peut exprimer sa décomposition en valeurs propres comme suit :

$$H_c = F^H H F \tag{2.18}$$

La matrice H qui contient dans sa diagonale la réponse fréquentielle du canal est donnée par :

$$H = \begin{pmatrix} H_0 & 0 & . & . & . & 0 \\ 0 & H_1 & . & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & . & H_{N-1} \end{pmatrix} \tag{2.19}$$

Où

$$H_k = \sum_{m=0}^{L-1} h_m \exp^{-j2\pi km/N} \tag{2.20}$$

En d'autres mots, H contient sur sa diagonale la DFT à N points de réponse impulsionnelle du canal. On peut écrire le signal reçu comme :

$$y = F^H HX + n \tag{2.21}$$

Pour retrouver les symboles émis, on applique la DFT au vecteur y, ce qui donne :

$$Y = HX + N \tag{2.22}$$

Où sous la forme scalaire suivante :

$$Y_k = H_k X_k + N, \;\; k = 1 \ldots N \tag{2.23}$$

En résumé, l'utilisation du préfixe cyclique transforme la convolution linéaire traditionnelle en une convolution circulaire, qui se traduit par la présence d'une matrice circulante dans le modèle du signal reçu.

Comme le montre l'équation (2.19), cette technique permet de découper un canal sélectif en fréquence en un ensemble de N canaux gaussiens parallèles (un gain complexe suivi par un bruit blanc additif gaussien), comme montré sur la figure 2.1. Chacun de ces canaux à un évanouissement plat avec un gain H_k, c'est-à-dire une des valeurs de la DFT à N points de la réponse impulsionnelle du canal.

L'équation (2.19) montre que l'on peut égaliser très facilement en multipliant chaque sortie de la DFT par le coefficient H_{k1} correspondant. Cependant, selon la valeur de H_k, la transmission sur une sous-bande est dite bonne ou mauvaise.

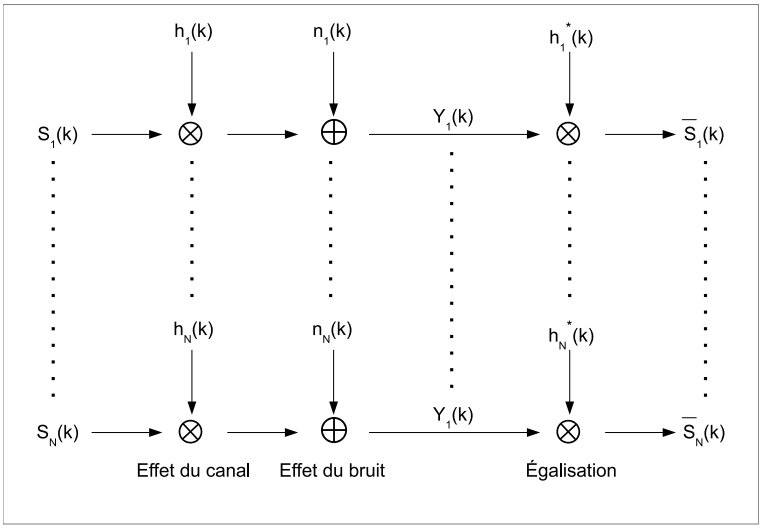

Figure 2.2 – Illustration des sous-porteuses d'un systeme OFDM.

2.4.2 L'orthogonalité :

La différence fondamentale entre les différentes techniques classiques de modulation multi-porteuses et l'OFDM est que cette dernière autorise un recouvrement spectral entre ces sous-porteuses, ce qui permet d'augmenter sensiblement leur nombre. Cependant, pour que ce recouvrement n'ait pas d'effet fatal, les porteuses doivent respecter une contrainte d'orthogonalité illustrée dans la figure suivante :

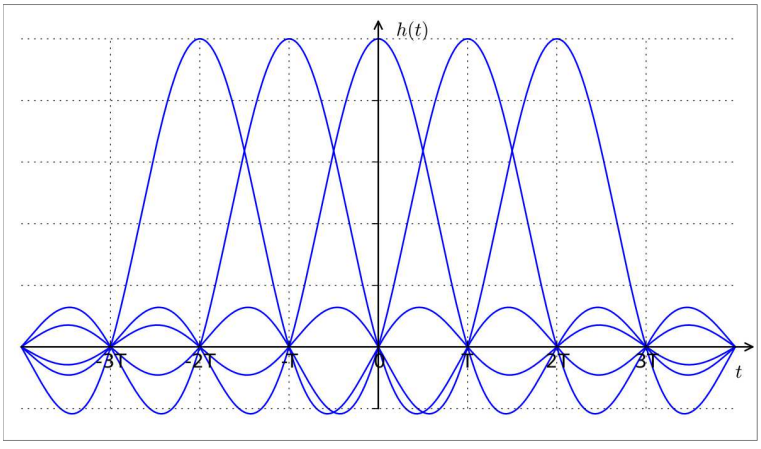

Figure 2.3 – L'orthogonalité des sous porteuses OFDM

2.4.3 Le préfixe cyclique :

Cependant, lorsque le canal de propagation $h(t)$ a une longueur l alors les L dernières composantes du bloc N transmis à l'instant $(k-1)NT$ (obtenu après transformée de Fourier inverse) interfèrent avec les l premières composantes du bloc suivant en raison de la mémoire du canal. Certes, des techniques de pré-égalisation et post-égalisation peuvent être misent en oeuvre pour remédier à ce problème mais au prix d'une complexité accrue.

Afin de préserver une égalisation simplifiée, Les systèmes OFDM actuelles emploient une astuce appelée préfixe cyclique, dont le but est d'introduire de la redondance et de structurer celle-ci afin de transformer le produit de convolution classique en un produit de convolution circulaire. Si le canal est composé de l coefficients, alors chaque bloc est cycliquement étendue après transformée de Fourier inverse de D coefficients tel que $D \geq L-1$. Par conséquent, un vecteur temporel de dimension $N+D$ est émis [2].

A l'aide de la transformée de Fourier, l'opération de convolution cyclique se transforme alors en un produit fréquentiel scalaire très simple à égaliser.

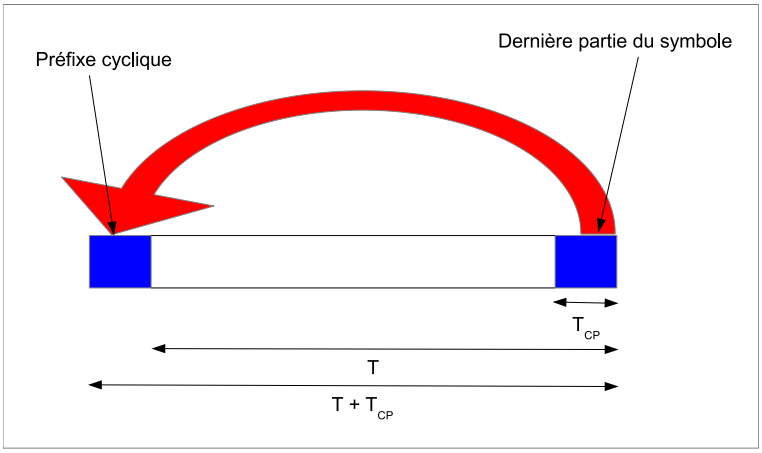

Figure 2.4 – Illustration du préfixe OFDM.

Chaque bloc est cycliquement étendue après transformée de Fourier inverse de D coefficients afin d'éliminer les interférences entre blocs dû à la mémoire du canal. Le terme T_{cp} correspond à la durée du préfixe cyclique alors que T correspond à la durée du bloc de données utiles.

2.4.4 Comment choisir le nombre de porteuses :

Afin de ne pas perdre en débit utile, il est serait judicieux de choisir le nombre de porteuses N très grand devant D puisque le facteur de redondance vaut $N/(N+D)$. En effet, pour D fixé, le facteur de redondance tend vers 1 quand le nombre de porteuses augmente. Cependant, la complexité du modulateur FFT croit avec la taille du bloc. Enfin, l'espacement entre porteuses est lié au facteur $1/NT$ qui diminue lorsque N augmente. Il n'y a donc aucun gain en termes de diversité lorsque N augmente. En pratique, dans les systèmes tels que $IEEE802.11a$, Nest à l'ordre de $4D$.

2.4.5 Schéma bloc d'émission-réception d'un système OFDM :

La figure 2.5 illustre les différents modules composant la chaîne de transmission OFDM. Le modulateur transforme les données binaires bi de durée Tb, en symboles complexes X_k de durée $T_q = \log_2(mT_b)$, où m est la taille de la constellation de la modulation utilisée. Le convertisseur série-parallèle dispose les symboles X_k en groupes (trames) de N symboles, la durée d'une trame Tu est N fois plus grande que la durée d'un symbole en série Tq. Par conséquent, l'e ?et de canal devient moins nuisible. En appliquant ensuite une transformée de Fourier inverse, on obtient la trame (symbole) OFDM. L'IFFT est utilisée afin de transformer le spectre du signal OFDM au domaine temporel pour la transmission à travers le canal. Un préfixe cyclique de durée Tg copie les D derniers symboles de la trame OFDM, et les ajoute ensuite au début de la trame. Après conversion parallèle-série, on obtient enfin le symbole OFDM, qui contient $N_s = N + D$ symboles de durée totale $T_s = T_u + T_g$ que l'on transmet à travers le canal.

À la réception, les opérations inverses sont réalisées, commençant par la suppression du préfixe cyclique, la décomposition spectrale des échantillons reçus calculée en utilisant l'algorithme FFT, et enfin la démodulation pour retrouver les données binaires transmises.

2.4.6 Correction d'erreurs :

Afin de réduire la probabilité d'erreur du message à transmettre, des techniques de diversité et décodage sont employées. On ajoute dans ce cas des bits de correction d'erreur au signal dont les valeurs dépendent de celles des bits du signal qu'il accompagne. Cette opération appelée codage est caractérisée par un coefficient R (plus R est petit

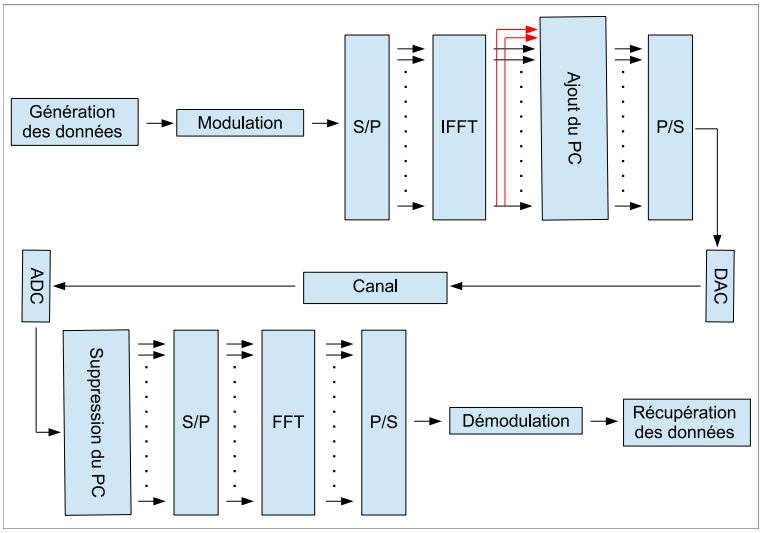

Figure 2.5 – Illustration de la chaîne de transmission OFDM.

et plus grande sera la redondance), le débit utile peut alors devenir négligeable par rapport au débit total.

En réception, le décodeur évalue les bits spéciaux de redondance et, comme il connaît les règles qui les ont engendrés et les erreurs les plus fréquentes, il corrige les désaccords.

Ajouter de la redondance s'avère inutile si tous les bits redondants sont transmis sur la même porteuse du canal affecté d'un évanouissement. Afin d'éviter ces inconvénients, les bits redondants sont transmis sur un grand nombre de porteuses. Comme l'affaiblissement du canal dépend de la fréquence, le message redondant pourra passer sans déformation au moins sur quelques fréquences. Cette « éparpillement de l'information » peut s'effectuer par différentes techniques d'entrelacements (fréquentielle, temporelle ou spatiale).

2.5 Avantages et inconvénients des systèmes OFDM :

2.5.1 Avantages :

— Une utilisation efficace des ressources fréquentielles dû au fait que dans les canaux se chevauchent tout en gardant une orthogonalité parfaite.

— Une égalisation numérique et un décodage simple et optimal grâce à l'utilisation

de l'intervalle de garde.

— Les techniques multi-porteuses sont robustes au bruit impulsif puisque chaque porteuse est affectée d'un bruit indépendant des autres porteuses.

— L'estimation du canal dans le contexte OFDM est facilitée par l'envoi de séquences d'apprentissage dans le domaine fréquentiel. L'identification des coefficients du canal se fait sans inversion de systèmes d'équations.

2.5.2 Inconvénients :

— L'OFDM peut engendrer des symboles temporels à forte amplitude, Ceci crée des contraintes sur les amplificateurs et conduit à une consommation de puissance importante.

— L'OFDM est trés vulnérable aux problèmes de décalage en fréquence qui engendre de l'interférence entre porteuses qui peut détruire l'orthogonalité des porteuses [3].

— L'OFDM est trés vulnérable aux erreurs de synchronisation induisant un déphasage sur les symboles reçus.

— Le modèle OFDM précédent ne s'applique pas quand le préfixe cyclique est plus petit que la longueur du canal. Dans ce cas, un symbole émis sur une porteuse pourra interférer avec les symboles de porteuses adjacentes.

Chapitre 3

Estimation et égalisation du canal :

3.1 Estimation du canal :

L'utilisation de la modulation différentielle dans les systèmes OFDM évite le besoin de suivre les variations temporelles du canal. Cependant, cette modulation limite le nombre de bits par symbole, ce qui engendre une perte en SNR de $3dB$. Par contre la modulation cohérente permet d'utiliser des constellations arbitraires, mais des stratégies efficaces d'estimation de canal sont exigées. La figure suivante montre la structure globale d'un récepteur OFDM cohérent.

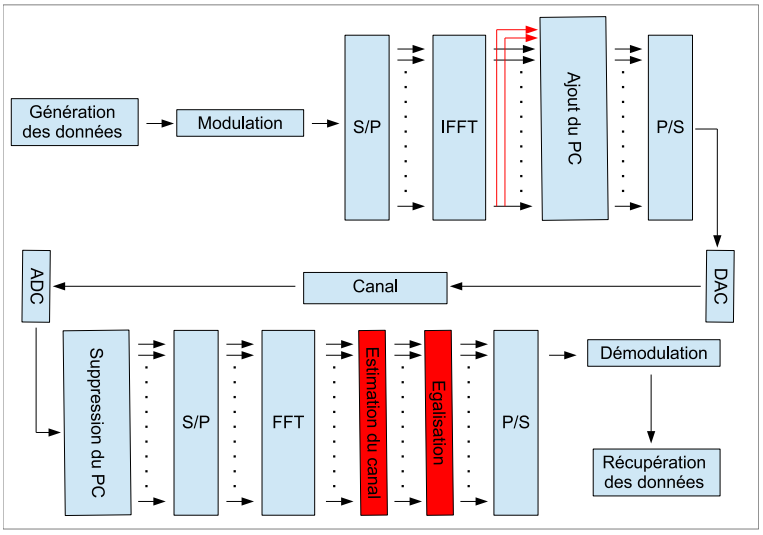

Figure 3.1 – Shémas indiquant le role de l'égalisation dans un système de communication.

Pour l'estimation du canal de transmission on utilise des symboles pilotes (un signal de référence connu par l'émetteur et le récepteur). Cette méthode consiste à utiliser des symboles connus au récepteur pour effectuer l'estimation. L'atténuation des symboles pilotes est mesurée et les atténuations des symboles de données entre ces symboles pilotes sont interpolées. En revanche, un tel système présente le désavantage de consommer une partie du débit disponible.

Il y a deux principaux problèmes à prendre en considération. Le premier est le positionnement des pilotes dans la grille temps-fréquence. Le second est la conception d'un estimateur avec une basse complexité et une grande capacité de suivre les variations du canal. On retrouve les estimateurs utilisant des blocs pilotes basées sur l'algorithme à moindres carrés (LS) et l'algorithme linéaire à erreur quadratique moyenne minimale (LMMSE), ainsi que des estimateurs utilisant des pilotes en forme de peigne incluant les algorithmes précédents avec interpolation, et aussi l'algorithme à vraisemblance maximale (ML).

Les méthodes d'estimation sont basées sur l'insertion des symboles pilotes connus par le récepteur, afin d'estimer et d'interpoler les composantes de la réponse du canal. Ces méthodes, dites PSAM, se divisent en trois grandes classes suivant le type d'insertion effectué (blocs pilotes, porteuses pilotes ou pilotes en treillis), leurs principes de base étant illustré à la figure suivante. Dans la première configuration, on utilise

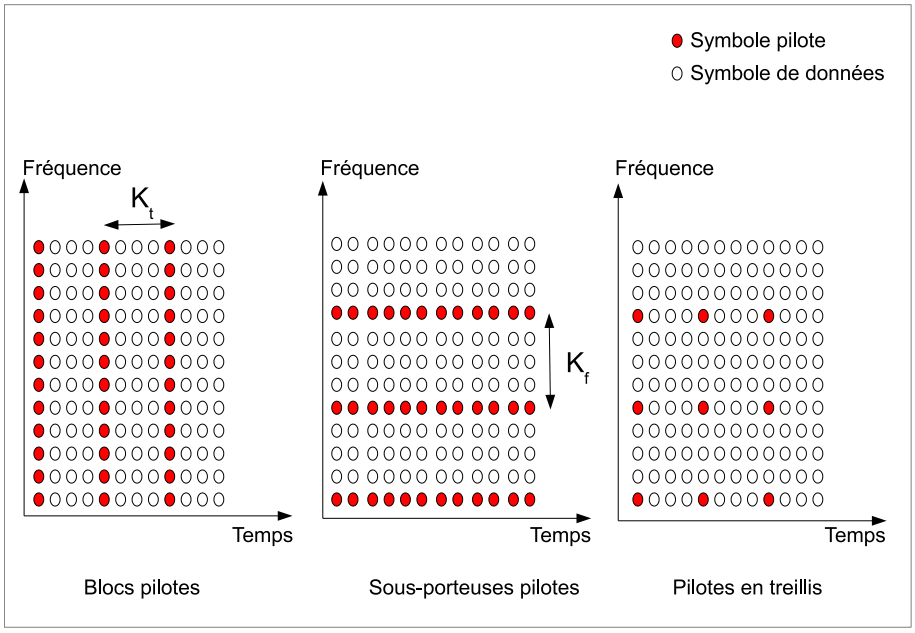

Figure 3.2 – Insertion des symbole pilote pour une meilleure estimation du canal.

périodiquement des symboles pilotes en forme de blocs. Cette configuration est déve-

loppée sous la supposition d'un canal à évanouissement lent. Chaque symbole de ce bloc étant envoyé sur chaque sous-porteuse, les caractéristiques du canal seront connues pour toutes les fréquences, mais à des intervalles de temps K_t.
Tandis que, dans la deuxième configuration, quelques symboles pilotes sont envoyés de façon continue sur certaines sous-porteuses, baptisées donc porteuses pilotes. Dans ce cas, l'état du canal est toujours connu, mais uniquement pour quelques fréquences porteuses, avec un intervalle de fréquence K_f, ce qui impose donc une interpolation fréquentielle. La troisième configuration est une combinaison entre les deux configurations précédentes, où les symboles pilotes sont placés à des intervalles de temps K_t et des intervalles de fréquence K_f.
Pour pouvoir suivre l'évolution dans le temps de la fonction de transfert du canal, il faut que l'écart fréquentiel entre les porteuses pilotes K_f soit inférieur à la bande de cohérence du canal B_c et que l'intervalle de temps entre deux symboles pilotes K_t soit bien inférieur au temps de cohérence du canal T_c.

3.1.1 L'estimateur LS (least squares) :

L'estimateur de canal à moindres carrés LS est le modèle le plus simple puisqu'il consiste en une division du signal reçu sur le signal d'entrée, qui devrait être des symboles pilotes connus. On rappelle qu'au récepteur, les symboles observés à la sortie de la DFT sont :

$$Y = HX + N \tag{3.1}$$

Où la matrice diagonale X contient les symboles transmis sur sa diagonale, et le vecteur H contient la réponse fréquentielle du canal. L'estimateur LS minimise le paramètre $(Y - XH)H(Y - XH)$. Les coefficients de l'estimateur LS sont donnés par la formule suivante [4] :

$$\hat{H}_{LS} = X^{-1}Y = (\frac{X_0}{Y_0}, \frac{X_1}{Y_1}, \ldots, \frac{X_{N-1}}{Y_{N-1}})^T \tag{3.2}$$

L'avantage principal de cet estimateur est sa simplicité. Il exige seulement une simple division par sous-porteuse. L'inconvénient principal est qu'il a une grande erreur quadratique moyenne. C'est dû à l'utilisation d'un modèle de canal trop simplifié et qu'il ne se sert pas des corrélations fréquentielle et temporelle du canal.

3.1.2 L'estimateur LS modifié (least Mean squares) :

Nous étudions, dans cette section, une méthode pour améliorer la précision de l'estimateur à moindres carrés (LS). L'estimateur de canal LS a une structure trés simple.

Cependant, il souffre d'une grande dégradation dans la précision de l'estimation comparée à l'estimateur LMMSE. Cette méthode transfère la réponse impulsionnelle du canal à partir du domaine fréquentiel au domaine temporel et approxime les trajets multiples de faible énergie par des zéros. La méthode utilisée aide à supprimer l'influence de bruit et à améliorer la précision de l'estimation.
Cette méthode est basée sur le fait que la plupart des composants significatifs des trajets dans la réponse impulsionnelle discrète du canal sont concentrés sur un intervalle de temps qui est beaucoup plus petit que la longueur de la trame OFDM. Puisque le bruit AWGN est identiquement distribué sur toute la bande, les composants qui ont une grande énergie sont plus fiables que ceux avec une faible énergie. Par conséquent, exclure les trajets non fiables aidera à améliorer la précision de l'estimation. Nous récapitulons cette méthode dans les étapes suivantes :

1. Estimer initialement le canal grâce à l'information pilote.
2. Obtenir la réponse impulsionnelle discrète du canal dans le domaine temporel $\tilde{h}(n)$ par la transformée de Fourier inverse (IFFT).
3. Ignorer les trajets dont les retards excèdent le plus grand retard du canal, ce qui peut être exprimé comme le montre la relation (26).
4. Estimer le canal au domaine fréquentiel par l'application de la transformée de Fourier FFT sur le signal $\tilde{h}(n)$.

$$\tilde{h}(n) = \begin{cases} \tilde{h}(n) & n \leq \tau_{max} \\ 0 & n \geq \tau_{max} \end{cases} \tag{3.3}$$

3.1.3 L'estimateur LMMSE (Least Minimum Mean Square Error) :

L'estimateur linéaire à erreur quadratique moyenne minimale (LMMSE) minimise l'erreur quadratique moyenne entre le canal réel et le canal estimé en utilisant la corrélation fréquentielle du canal. Ceci est réalisé par une transformation linéaire optimale appliquée à l'estimateur du canal à moindres carrés LS.
On dénote par R_{gg}, R_{hh} et R_{yy} les matrices d'autocovariance de g, h et y, respectivement, et par R_{gy} la matrice de covariance croisée entre g et y. On dénote en outre par $\sigma_n^2 = E|n|^2$ la variance de bruit. On suppose que le vecteur de la réponse impulsionnelle du canal g et le bruit gaussien n sont non-corrélés. On a alors :

$$R_{hh} = E\{hh^H\} = E\{(Fg)(Fg)^H\} = FE\{gg^H\}F^H = FR_{gg}F^H \tag{3.4}$$

$$R_{gy} = E\{gy^H\} = E\{g(XFg+n)^H\} = R_{gg}F^HX^H \tag{3.5}$$

$$R_{yy} = E\{yy^H\} = XFR_{gg}F^HX^H + \sigma_n^2 I_N \tag{3.6}$$

On suppose que R_{gg} (ainsi que R_{hh}) et σ_n^2 sont connus au récepteur à l'avance. De la théorie des filtres adaptatifs, la solution optimale en termes d'erreur quadratique moyenne qui donne l'estimateur LMMSE de la réponse impulsionnelle du canal est :

$$\hat{g}_{LMMSE} = R_{gy} R_{yy}^{-1} Y \tag{3.7}$$

Enfin, on a :

$$\begin{aligned}
\hat{H}_{LMMSE} &= F\hat{g}_{LMMSE} && (3.8)\\
\hat{H}_{LMMSE} &= FR_{gg}F^H X^H (XFR_{gg}F^H X^H + \sigma_n^2 I_N)^{-1} Y && (3.9)\\
\hat{H}_{LMMSE} &= FR_{gg}F^H (XFR_{gg}F^H + \sigma_n^2 (X^H)^{-1})^{-1} Y && (3.10)\\
\hat{H}_{LMMSE} &= FR_{gg}F^H (XFR_{gg}F^H + \sigma_n^2 (X^H)^{-1} X^{-1})^{-1} X^{-1} Y && (3.11)\\
\hat{H}_{LMMSE} &= R_{hh}(R_{hh} + \sigma_n^2 (XX^H)^{-1})^{-1} \hat{H}_{LS} && (3.12)
\end{aligned}$$

L'estimateur LMMSE donne de meilleure performance par rapport à l'estimateur LS, particulièrement dans le cas de faibles SNR. L'inconvénient principal de cet estimateur est qu'il a une complexité trés élevée. L'évaluation de l'inverse de R_{hh} et de XXH implique l'inversion d'une matrice de dimension N x N qui complique le calcul de cet estimateur chaque fois que les données dans X changent.

3.1.4 L'estimateur LMMSE modifié (Least Minimum Mean Square Error Modified) :

La complexité de l'estimateur LMMSE peut être encore réduite en utilisant la décomposition en valeur singulière de la matrice d'autocorrélation R_{hh}. Par conséquent nous pouvons appliquer une réduction de rang et utiliser seulement les éléments les plus significatifs de R_{hh} en utilisant la décomposition en valeur singulière, qui combine les trois techniques de simplification suivantes :
1- La première simplification est de remplacer le terme $(XX^H)^{-1}$ de l'équation (3.12) avec son espérance $E\{(XX^H)^{-1}\}$. Assumant la même constellation du signal sur toutes les sous-porteuses et tous les points équiprobables de la constellation, on a :

$$E\{(XX^H)^{-1}\} = E\{\left|\frac{1}{X_k}\right|^2\} I_N \tag{3.13}$$

Définissant le rapport SNR moyen par :

$$\tilde{\gamma} = \frac{E\{\left|\frac{1}{X_k}\right|^2\}}{\sigma_n^2} \tag{3.14}$$

Alors, dans l'équation (3.12) on peut faire l'approximation suivante :

$$\sigma_n^2(XX^H)^{-1} \approx \frac{\beta}{\tilde{\gamma}} I_N \tag{3.15}$$

où β est une constante qui dépende de la constellation du signal. Elle est donnée par :

$$\beta = \frac{E\{|X_k|^2\}}{E\{\left|\frac{1}{X_k}\right|^2\}} \tag{3.16}$$

2- La deuxième simplification est basée sur l'approximation à bas rang. On sait que la majorité de l'énergie dans le vecteur g est contenue dans les premiers L trajets. Par conséquent, nous pouvons considérer seulement les trajets à énergie significative, c'est-à-dire, le coin gauche supérieur de la matrice d'autocovariance R_{gg}. Ainsi la taille effective de la matrice est réduite aprés l'utilisation de l'approximation à bas rang.
3- La troisième simplification consiste en l'utilisation de la décomposition en valeur singulière (SVD2) de la matrice R_{hh} :

$$R_{hh} = UDU^H \tag{3.17}$$

où U est une matrice unitaire contenant les vecteurs singuliers et D est une matrice diagonale contenant les valeurs singulières,$d_0 \geq d_1 \geq \ldots d_(N-1) \geq 0$, sur sa diagonale. Combinant les techniques de simplification précédentes, l'estimateur LMMSE modifié est expliqué comme suit :
Le système détermine d'abord le rang de la matrice exigé par l'estimateur, dénoté par p, qui ne devrait pas être plus petit que L. Puis, on doit connaître la constellation du signal, la variance de bruit et la matrice d'autocovariance de canal R_{hh}. Au récepteur, on calcule β, $\tilde{\gamma}$, la matrice unitaire U, et les valeurs singulières d_k. On obtient ainsi une matrice diagonale D de taille N x N, ses éléments sont donnés par :

$$\delta_k = \begin{cases} \frac{d_k}{d_k + (\frac{\beta}{\tilde{\gamma}})} & k = 0, 1, ..., p-1 \\ 0 & k = p, p+1, ..., N-1 \end{cases} \tag{3.18}$$

Par la prise en compte des premiers p valeurs singulières correspondant aux valeurs les plus significatives de R_{hh}, la meilleure approximation de rang p de l'estimateur est :

$$\hat{H}_{LMMSE} = UD_p\hat{U}H\hat{H}_{LS} \tag{3.19}$$

où D_p est une matrice de dimension p x p donnée par le coin gauche supérieur de la matrice D. L'estimateur LMMSEM peut être interprété comme suit :
d'abord projetant l'estimateur LS sur un sous-espace de petite dimension (aussi petit que L), la complexité de l'estimateur LMMSEM est inférieure à celle de l'estimateur LMMSE. Cependant, cet estimateur présente des erreurs dûes à la partie du canal ignorée par le sous-espace choisi, comme indiqué sur la figure suivante.

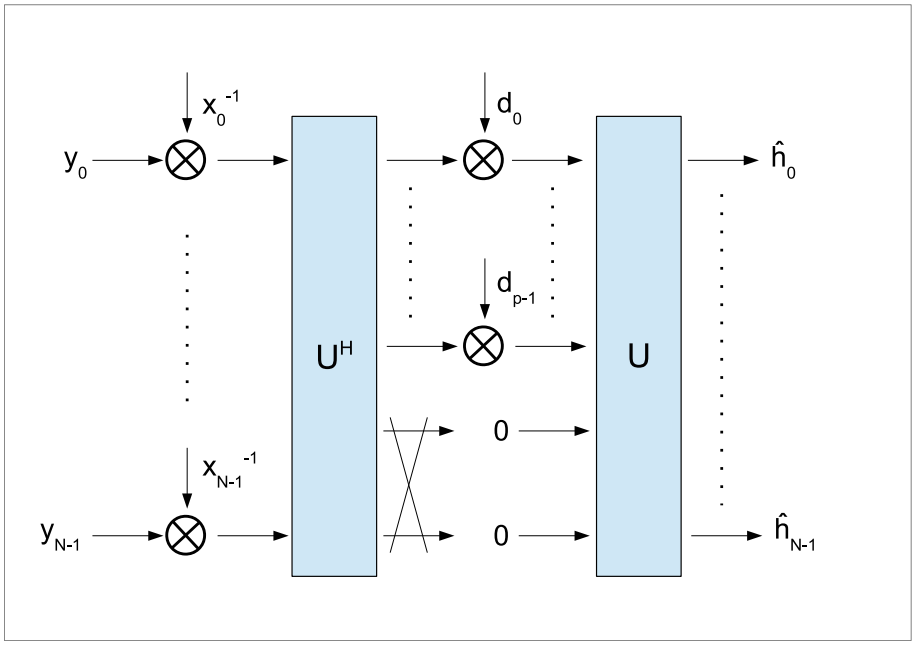

Figure 3.3 – Shémas d'un estimateur LMMSE modifié.

3.1.5 L'estimateur ML(Maximum Likehood) :

Comme mentionné précédemment, la majeure partie de l'énergie dans g est contenue dans les L premiers trajets. De même que la définition de la matrice DFT, nous définissons la matrice DFT non carrée, de dimension A x B, par :

$$F_{A,B} = \left[F_N^{a,b}\right]_{AxB} (0 \leq a \leq A, 0 \leq b \leq B) \tag{3.20}$$

Où

$$F_N^{a,b} = \frac{1}{\sqrt{N}} \exp^{-j2\pi ab/N} \tag{3.21}$$

En outre, nous définissons la matrice DFT uniformément espacée, avec l'espace s, comme suit :

$$\bar{F}_{A,B} = \left[F_N^{as,b}\right]_{AxB} = \left[F_N^{a,bs}\right]_{AxB} \tag{3.22}$$

Il est évident que :

$$\hat{h}(p) = \bar{F}_{N_p,L} g \tag{3.23}$$

où N_p est le nombre de porteuses pilotes utilisées. Ainsi, l'estimateur à vraisemblance maximale (ML) de g est obtenu par :

$$\tilde{g}_{ML} = (\bar{F}^H_{N_p,L} \bar{F}_{N_p,L})^{-1} \bar{F}^H_{N_p,L} \tilde{H}^{(p)}_{LS} \tag{3.24}$$

Finalement, l'estimation complète de toutes les sous-porteuses est calculée par :

$$H_{ML} = F_{N,L} \tilde{g}_{ML} \tag{3.25}$$

3.2 L'interpolation :

Dans l'estimation de canal basée sur le deuxiéme et troisiéme cas d'arrangement des pilotes, une technique efficace d'interpolation est nécessaire afin d'estimer le canal aux sous-porteuses de données en utilisant l'information sur le canal aux sous-porteuses pilotes. L'interpolation est utilisée pour estimer le canal aux sous-porteuses de données, où le vecteur $\hat{H}^{(p)}$ de longueur N_p est interpolé pour donner un vecteur $\hat{H}$ de longueur N ($N_p = N/K_f$), sans utiliser la connaissance additionnelle des statistiques de canal.

3.2.1 L'interpolation linéaire (LI) :

La méthode LI utilisée pour estimer le canal aux sous-porteuses de données entre deux sous-porteuses pilotes $\hat{H}^{(p)}(m)$ et $\hat{H}^{(p)}(m+1)$ est donnée par [5] :

$$\hat{H}(mk_f + l) = \hat{H}^{(p)}(m) + \frac{1}{k_f}(\hat{H}^{(p)}(m+1) - \hat{H}^{(p)}(m)), \;\; 0 \leq l \leq k_f \tag{3.26}$$

3.2.2 L'interpolation du second ordre (SOI) :

La méthode SOI s'exécute mieux que la méthode LI, où l'estimation de canal aux sous-porteuses de données est obtenue par la combinaison linéaire des trois estimations des sous-porteuses pilotes adjacentes. Le canal estimé par l'interpolation de second ordre est donnée par :

$$\hat{H}(mk_f + l) = c_1\hat{H}^{(p)}(m-1) + c_0\hat{H}^{(p)}(m) + c_{-1}\hat{H}^{(p)}(m+1) + ... \tag{3.27}$$

Où

$$c_1 = \alpha(\alpha-1)/2 \tag{3.28}$$
$$c_0 = -(\alpha-1)(\alpha+1) \tag{3.29}$$
$$c_{-1} = \alpha(\alpha+1)/2 \tag{3.30}$$
$$\alpha = l/N \tag{3.31}$$

3.2.3 L'interpolation par splines cubiques (SCI) :

La méthode SCI produit un polynôme continu entre les points de données (la fonction spline dans Matlab).L'interpolation par splines cubiques s'exécute mieux que l'in-

terpolation linéaire.

3.2.4 L'interpolation passe-bas (LPI) :

La méthode *LPI* s'exécute en insérant des zéros dans la séquence originale $\hat{H}^{(p)}$ et puis en appliquant un filtre passe-bas de réponse impulsionnelle finie (FIR) (la fonction interp dans Matlab), qui permet aux données originales de passer sans changement. Cette méthode minimise l'erreur quadratique moyenne entre les points interpolés.

3.3 L'égaliseur ZF :

Dans l'OFDM, tant que le préfixe cyclique demeure assez long par rapport à la réponse impulsionnelle du canal pour maintenir l'orthogonalité entre les sous-porteuses, l'égalisation peut être réalisée simplement par l'égaliseur ZF [6]. L'égaliseur ZF minimise la distorsion maximale entre les symboles à la sortie de l'égaliseur. Le filtre linéaire du canal avec les coefficients h_n et les coefficients de l'égaliseur w_n peuvent être exprimés comme un seul filtre par :

$$q_n = \sum_{j=-\infty}^{+\infty} w_j h_{n-j} = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \tag{3.32}$$

Si on utilise la transformée en z on obtient :

$$Q(z) = W(z)H(z) = 1 \tag{3.33}$$

En d'autres termes, les coefficients de l'égaliseur sont donnés par :

$$W(z) = \frac{1}{H(z)} \tag{3.34}$$

Ainsi, le filtre qui satisfait le critère ZF a les coefficients $W(z)$ qui sont juste l'inverse des coefficients du canal $H(z)$. Il est important de noter que cet égaliseur n'élimine pas l'IES. Cependant, une prolongation cyclique plus longue que la réponse impulsionnelle de canal peut supprimer l'IES et l'IEP. Dans l'OFDM, l'égaliseur ZF aura lieu dans le domaine fréquentiel, donc, le rôle de l'égaliseur devient juste une opération d'ajustement de gain et de phase sur chaque sous-porteuse. L'inconvénient principal d'une telle méthode est quand $H(z)$ est tout prés du zéro, le filtre réciproque $W(z)$ sera trés grand, ce qui amplifiera nettement le bruit ou les erreurs d'approximations.

Chapitre 4

Turbo Codes :

4.1 Introduction :

Les Turbo codes ont été présentés en 1993 par Berrou et Glavieux et Thitimajshima. Ils ont apporté la plus grande avancée en codage depuis les travaux d'Ungerboeck qui avait introduit les Modulations codées (Trellis-Coded Modulation) en 1982. Ils exploitent toutes les potentialités des schémas de codes concaténés proposés par Forney en 1966 [7].

Les turbo-codes peuvent etre considérés comme un raffinement d'une structure de codage concaténé avec la mise en place du décodage itératif à sortie souple. Cette sortie souple permettra de déterminer les symboles émis à partir d'une règle MAP (maximum a posteriori). Une séquence encodée en Turbo est traité par plusieurs encodeurs convolutionnels et Les entrelaceurs, donc le principe de ces sous-blocs est présenté dans ce qui suit pour favoriser la compréhension d'un systéme de codage Turbo.

4.2 Codes convolutionnels :

Un code convolutionnel est un code FEC, qui améliore le BER en ajoutant une redondance á l'information qui doit être transmise. Contrairement á un code traditionnel, la redondance est ajoutée réguliérement aprés chaque bit. Un exemple d'un code convolutionnel est représenté par la figure 4.1. On voit que c'est un codeur de taux de

codage 1/2 (une entrée génére deux sorties), définies comme suit :

$$Sortie_1(k) = Entree(k) + Entree(k-1) \tag{4.1}$$

$$Sortie_2(k) = Entree(k) + Entree(k-2) \tag{4.2}$$

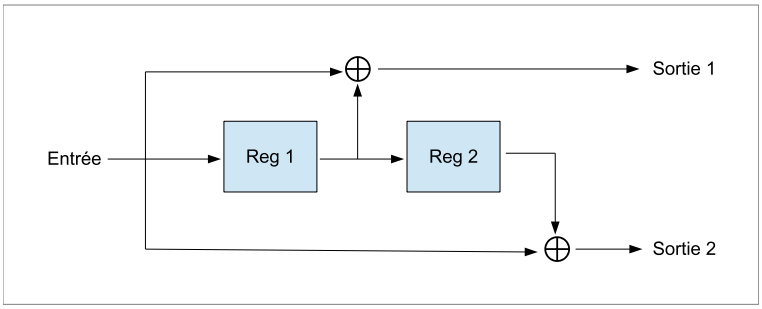

Figure 4.1 – Illustration d'un codeur convolutif d'un taux de codage 1/2.

Outre le codeur présenté dans la figure 4.1, un codeur convolutionnel peut se différencier par l'utilisation d'une sortie systématique. Ici, le bit d'entrée est directement passé comme une partie de la sortie combiné avec un bit de parité.

Notamment les codes Turbo peuvent profiter de cette fonctionnalité, car cela permettra au codeur Turbo qui se compose de deux codeurs de taux de codage 1/2, d'avoir un taux de codage de 1/3 au lieu de 1/4 en raison du fait que le bit systématique transmis peut être utilisé dans les deux décodeurs avec une simple utilisation du motif d'entrelacement dans le décodeur. Le codeur convolutif avec une sortie systématique est illustré par la figure 4.2 :

$$systematique(k) = Entree(k) \tag{4.3}$$

$$parite(k) = Entree(k) + Entree(k-1) + Entree(k-2) \tag{4.4}$$

Pour un codeur convolutionnel, une partie récursive peut être ajoutée, ce qui indique qu'un ou plusieurs des registres á décalage sont connectés dans une boucle de retour. Un tel codeur est illustré par la figure 4.3. Notez que le codeur est également systématique,

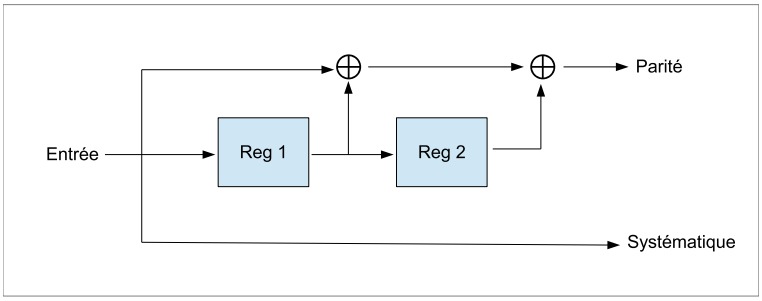

Figure 4.2 – Illustration d'un codeur convolutionnel systématique.

ce qui est commun avec un codeur récursif, cela permet á créer un codeur RSC1, qui est un élément principal dans un codeur Turbo.

$$retour(k) = Entree(k) + retour(k-2) \tag{4.5}$$

$$parite(k) = retour(k) + retour(k-1) + retour(k-2) \tag{4.6}$$

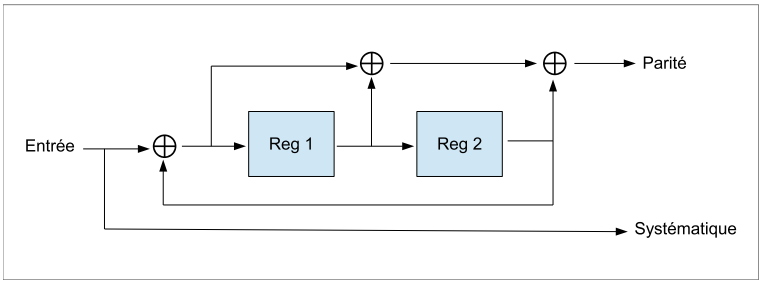

Figure 4.3 – Illustration d'un codeur RSC.

Un codeur convolutionnel avec une longueur de contrainte de 7 diminuera le SNR requis avec plus de 5dB, pour le même BER [8]. Cependant, comme c'est un Taux 1/2, il faut deux fois la bande passante, car il existe un bit de parité pour chaque bit d'information. Non seulement la bande passante accrue, mais aussi le calcul de la puissance requise pour décoder le signal reçu est un paramétre, á prendre en considération.

La complexité augmente exponentiellement avec la de contrainte, cela rend le choix entre la résilience des données et la puissance de calcul au récepteur et non plus la bande passante requise par rapport aux données élastiques.

Les paramétres lorsqu'on considère un codeur RSC sont la longueur de contrainte k, Désignant le nombre de registres plus un, le taux de codage étant l'entrée / sortie n, qui est le nombre d'additionneurs modulo 2. Un codeur convolutif peut être décrit par l'utilisation d'un polynôme générateur, qui est essentiellement un tableau des fonctions de transfert de l'entrée à chaque sortie du codeur.

Ainsi, pour un codeur RSC, le premier élément du tableau est 1, comme le bit est directement passé á la sortie. Le deuxiéme élément, étant la Sortie récursif consistant á un flux vers l'avant et un polynôme vers l'arriére, qui régit l'interconnexion entre les additionneurs de modulo 2 et les registres internes. La fonction de transfert de la sortie récursive est alors définie par g_1/g_0, oú g_1 et g_0 sont définis par un ensemble de n éléments $g_x = [x_1 x_2 \dots x_n]$, $x_1 \dots x_n$ étant 0 ou 1 indiquant une connexion. Par conséquent, un polynôme générateur complet est alors définit comme $G = [1, g_1/g_0]$.

Un codeur avec 2 registres de mémoire, présente initialement quatre états possibles dont chaque état a 2 transitions possibles, d'oú une table de vérité peut être configurée pour la fonctionnalité donnée. Le tableau suivant illustre un exemple avec le polynômes récursif $g_1 = [111]$ et $g_0 = [101]$, oú l'entrée et la sortie systématique sont désigné par U_k et le bit de sortie codé est noté Z_k. Par exemple, il est noté que 00 comme états initial du registre, génére 1 sur les deux sorties et les registres se déplacent vers 10, lorsque 1 est alimenté en entrée.

	Etat initial			
Etat final	00	01	10	11
00	0(00)	1(11)	x	x
10	1(11)	0(00)	x	x
01	x	x	1(10)	1(01)
11	x	x	1(01)	1(10)

Tableau 4.1 – Le tableau du polynôme générateur du codeur RSC

Le code de sortie du générateur peut également être illustré par l'utilisation d'un diagramme en treillis, où les transitions sont affichées plus efficacement. Ce diagramme en treillis est trés illustratif et par conséquent, un exemple sera illustré par la séquence d'entrée U_k.

$$U_k = [101011] \tag{4.7}$$

Le mot de code résultant peut être illustré comme un chemin donné à travers le

treillis, montré à la figure 4.4.

Comme on l'a vu, la sortie mesure $8bits$, ce qui est dû au fait que le treillis a été terminé. Terminer la séquence signifie que l'état final est toujours l'état zéro. Connaître l'état final aidera le décodeur à trouver le bon chemin à travers le treillis.

La séquence d'entrée a la longueur de 6 bits, mais la sortie consiste de 8 transitions d'état, puisque la terminaison ajoute 2 états supplémentaires au treillis.

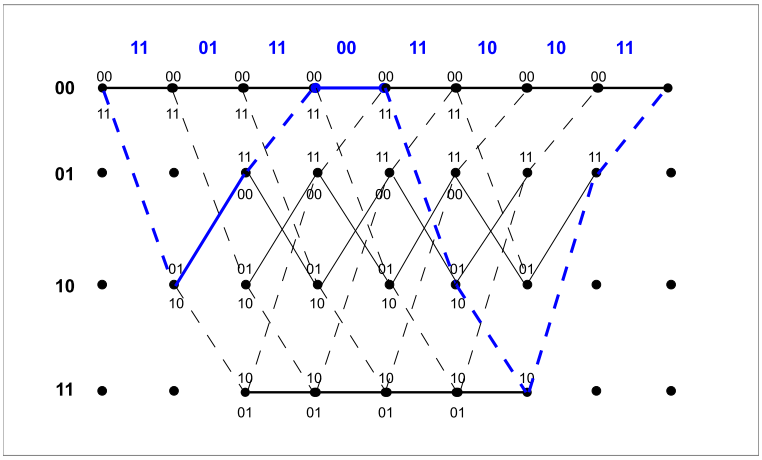

Figure 4.4 – Illustration d'un treillis du codeur RSC avec le chemin en bleu.

La séquence est représentée dans les transitions du diagramme en treillis état par l'état, commençant par l'état 00, puis suivant le tableau de vérité 4.1. le le schéma de treillis est terminé par les bits de queue, qui sont spécifiquement conçus pour que le treillis se termine de nouveau à l'état 00 [9].

4.3 L'entrelaceur :

Un entrelaceur est un algorithme qui réorganise les index de la séquence d'entrée en utilisant l'un des différents types d'entrelacement. Un exemple de ceci est montré à la figure 4.5 :

La figure 4.6 montre comment les erreurs provoquées par un évanouissement instantané dans le canal, sont réparti sur l'ensemble de l'information, ce qui est un avantage lors de l'utilisation du codage convolutionnel. En outre, lors de l'entrelacement des deux codeurs convolutionnels différemment dans le codeur Turbo, le risque de corruption des deux codes est minim.

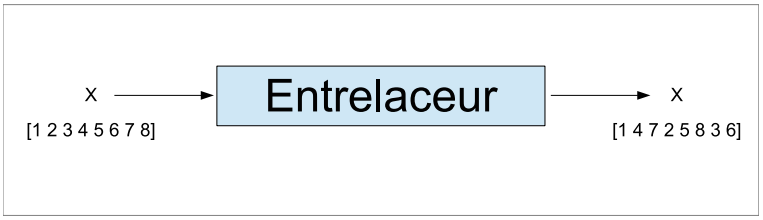

Figure 4.5 – Illustration d'un exemple d'entrelaceur.

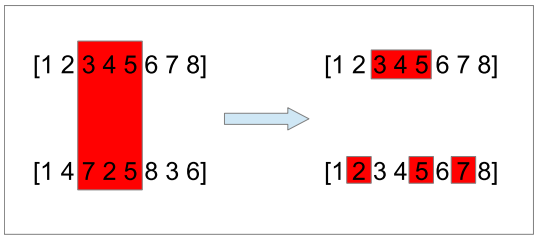

Figure 4.6 – Illustration de la corruption des bits dans une séquence due au canal.

4.3.1 Entrelacement polynomial :

L'entrelaceur utilisé pour les Turbo Codes est un entrelaceur polynomial. Il utilise un polynôme pour réorganiser l'ordre des bits de séquence d'entrée, comme on peut le voir dans l'équation ci-dessous [9].

$$\pi(i) = (f_1.i + f_2.i^2) mod K \tag{4.8}$$

où :

— i est l'index de la séquence d'entrée.

— $\pi(i)$ est l'index de la séquence entrelacée.

— K est la longueur de la séquence d'entrée.

— f_1 et f_2 sont des valeurs associées à la longueur de la séquence.

Les valeurs pour f_1 et f_2 sont définies différemment pour chaque longueur de trame. Par exemple pour une longueur de séquence de 64 bits, f_1 et f_2 seront 7 et 16 respectivement.

4.4 Architecture du codeur Turbo :

Un odeur Turbo est composée de deux codeurs RSC et un entrelaceur. Les codeurs RSC identiques sont mis en parallèle, d'où un autre titre pour les Turbo codes est $PCCC2$. Pour diminuer la corrélation entre les deux codeurs RSC, ils sont séparé par un entrelaceur, qui mélange les bits d'entrée de l'un des deux codeurs.

Étant donné que les codeurs RSC sont à débit 1/2 de polynômes $g_1 = [1101]$ et $g_0 = [1011]$, une des deux séquences de sortie sera la systématique , qui est identique dans les deux codeurs, à part l'entrelacement. Sur cette base, la sortie systématique est négligée par le deuxième codeur convolutionnel , sauf la partie de la queue qui est différente pour les deux et doivent tous les deux être transmis. La structure du codeur Turbo se trouve à la figure 4.7. La clé de ce codage est de rendre les données non corrélées autant que possible. Ceci est dû au fait que, en particulier, les canaux de décoloration sélectifs en fréquence peuvent corrompent un bit, qui peut ensuite être décodés dans un autre intervalle de temps OFDM. Ainsi, un entrelaceur efficace est essentiel pour gérer le canal de décoloration.

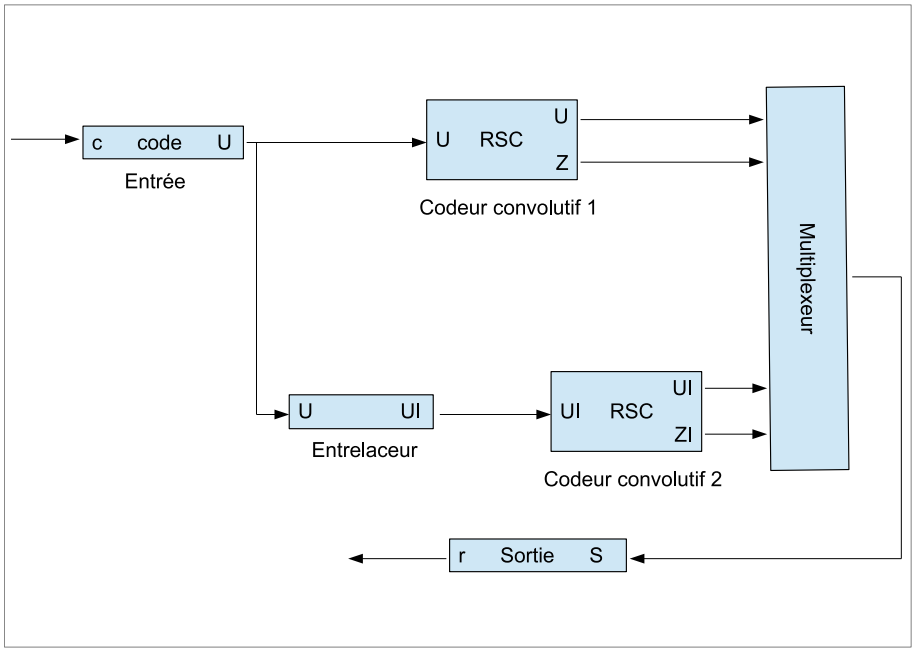

Figure 4.7 – Illustration de la structure du codeur turbo 1/3.

Il sort des encodeurs convolutionnels sont ensuite multiplexés ensemble, afin qu'ils puissent être transmis sur le canal vers le récepteur. La combinaison des sorties avant

la fin du treillis est définie comme [10] :

$$Sortie = [U_0 Z_0 Z_0^{'} U_1 Z_1 Z_1^{'} \dots U_{N-1} Z_{N-1} Z_{N-1}^{'}] \tag{4.9}$$

Et les bits de queue sont triés comme suit :

$$Sortie = [U_N U_{N+1} U_{N+2} Z_N Z_{N+1} Z_{N+2} U_N^{'} U_{N+1}^{'} U_{N+2}^{'} Z_N^{'} Z_{N+1}^{'} Z_{N+2}^{'}] \tag{4.10}$$

où :

— N est la longueur de la séquence d'entrée.
— L est la longueur de contrainte moins un.
— U_k est la séquence d'entrée et la sortie systématique.
— $U_K^{'}$ est la sortie systématique entrelacée.
— Z_k est la séquence de vérification de parité de la séquence d'entrée.
— $Z_K^{'}$ est la séquence de vérification de parité de la séquence d'entrée entrelacée.

4.5 Décodage Turbo :

Pour comprendre le décodeur Turbo, il faut d'abord connaître le principe du décodage des codes convolutifs. Pour exploiter le pré-traitement de données étendu dans l'émetteur, le récepteur se compose de plusieurs éléments dont la seule tâche est de décoder la séquence transmise avec un minimum d'erreurs. Un décodeur MAP3 est l'un de ces éléments, étant la partie principale de l'architecture de décodage. Néanmoins un élément moins complexe en termes de décodeur Viterbi, peut être utilisé dans le cadre de l'architecture.

4.5.1 Décodeur convolutionnel :

Une architecture générale concernant le décodeur convolutionnel dans le récepteur est illustrée par la figure 4.8. Une séquence transmise est désignée par $[U_0 Z_0 U_1 Z_1 \dots U_{N-1} Z_{N-1}]$ alors que la séquence reçue est désignée par $[r_0^i r_0^p r_1^i r_1^p \dots r_{N-1}^i r_{N-1}^p]$ qui est démultiplexé en deux séquences distinctes qui entrent dans le décodeur.

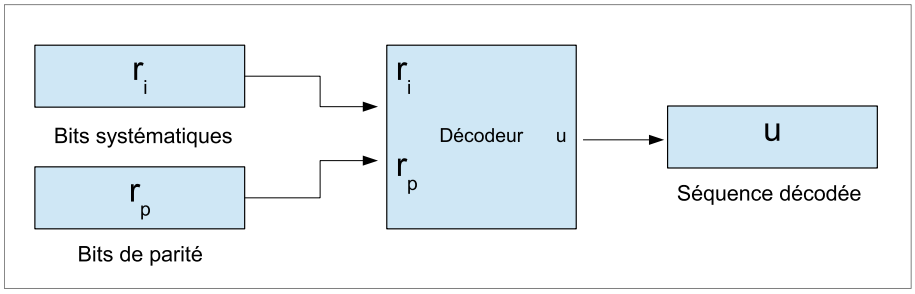

Figure 4.8 – décodage par le bit systématique r_i et le bit de parité r_p sortant la séquence décodée u.

La première partie de cette section décrit l'algorithme de Viterbi qui est un algorithme développé pour diminuer la complexité du décodeur convolutif en profitant du diagramme en treillis. La deuxième partie décrit comment les performances peuvent être augmentées en utilisant des bits de décision souples dans le décodage, seulement des zéros et des uns. La troisième partie est une déduction mathématique de la l'algorithme MAP, qui est un algorithme de décodage très robuste qui produit une décision soft LLR pour les bits donnés, qui peut être utilisée comme une information d'un décodeur itératif, tel que le décodeur Turbo. La dernière partie expliquera l'algorithme SOVA, qui est un décodeur Viterbi d'une décision soft capable d'utiliser les informations extrinsèques mais moins robuste que l'algorithme MAP.

4.5.2 Viterbi :

L'algorithme de Viterbi [8, 11] sert dans une grande variété de récepteurs sans fil, à un niveau faible de complexité limitant la puissance de traitement requise du décodeur, pour obtenir un résultat satisfaisant. L'algorithme peut être implémenté avec deux sorties différentes, en termes de décision (hard ou soft), en se différenciant dans la façon dont les décisions de décodage sont prises. Par conséquent, cette sous-section se concentrera sur D'abord le hard Viterbi après quoi le soft Viterbi suivra.

L'algorithme de Viterbi fonctionne en parcourant le chemin à travers le diagramme en treillis qui a la plus grande probabilité d'être le chemin réel. Pour trouver le le chemin le plus probable, l'algorithme calcule toutes les métriques de branchement $M(S^j)$, qui Sont les transitions d'un état à un autre état S^j. La métrique de la branche est donc Pondéré par les moyens de probabilité, où les branches les plus probables seront Sélectionné comme mesure de branche survivante. Lorsque le treillis complet est ensuite traité, seuls les chemins survivants sont laissés, donc la séquence transmise est la plus

probablement révélé. Le chemin étant estimé comme le vrai chemin, peut être déclaré comme la probabilité conjointe que les bits de code reçus soient égaux aux transitions des treillis, transféré sur un canal [9].

$$P(r/y) = \prod_{k=0}^{N-1}(1-p)(\frac{1-p}{p})^{-d(r_k,y_k)} \tag{4.11}$$

où :

— $P(r/y)$ est la probabilité que la séquence r soit reçue, lorsque la séquence y est transmise

— p indique la probabilité d'erreur dans le canal.

— r_k est la séquence reçue à l'index n.

— y_k est la séquence indiquant l'estimation de VA de la séquence transmise. Quelle est également la séquence qui minimise la distance de hamming à r_k.

— $d(r_k, y_k)$ est la distance Hamming entre r_k et y_k

La distance de hamming de deux séquences de longueur égale est le nombre de les positions auxquelles les symboles correspondants sont différents [12], et est défini par l'équation suivante :

$$d_H(u, u^{'}) = \sum_{k=1}^{K}\delta(u_k, u_k^{'}) \tag{4.12}$$

$$\delta\left(u, u^{'}\right) = \begin{cases} 1 & u \neq u^{'} \\ 0 & u = u^{'} \end{cases} \tag{4.13}$$

Souvent, la fonction log-vraisemblance (log-likelihood) est utilisée, ce qui montre que la métrique de Chacune des branches peut être écrite comme :

$$M(r_k/y_k) = c_1(\log\left((1-p)(\frac{1-p}{p})^{-d(r_k,y_k)}\right) + c_2) \tag{4.14}$$

Lorsque c_1 et c_2 peuvent être choisis pour rendre la sortie proche d'un nombre entier, ce qui est pratique pour la mise en œuvre du décodeur pour éviter la quantification de précision.

Si c_1 est configuré pour enregistrer $\log\left(\frac{1-p}{p}\right)^{-1}$ et c_2 comme $-\log(1-p)$, la métrique peut être calculée comme une distance Hamming pour un décodage dur. Ainsi, la meilleure mesure est le nombre le plus élevée disponible, car cela décrit la séquence avec la moindre quantité d'erreurs.

L'algorithme de Viterbi a le flux suivant :

initialiser :

— $k = 0$ et $V(S_k^j) = \infty$
— $V(S_k^j) = 0$

Etape 1 :

— k=k+1
—

$$\texttt{si}\ V(S_k^j) > V(S_{k-1}^j) - d(r_k, y_k^j) \tag{4.15}$$
$$V(S_k^j) = V(S_{k-1}^j) - d(r_k, y_k^j) \tag{4.16}$$

Etape 1 :

— si $k < N$
aller à l'étape 1

Etape 3 :

— Effectuez un retrait à travers le treillis, à partir de la meilleure métrique de branche résultante et en suivant le meilleur chemin à survie, en économisant N symboles survivants, révélant ainsi la séquence transmise la plus probable.

où :

— $V(S_k^j)$ est une valeur attribuée à chaque état dans le treillis, il tiendra le meilleur Métrique de chemin partiel pour l'état donné.

— y_k^j représente les états dans le treillis.

Un exemple de décodage dur de Viterbi à l'aide de la distance Hamming est fait, en utilisant la même séquence de l'exemple précédent. Deux versions de la séquence seront utilisées, la séquence d'origine sans erreurs et celle où des erreurs de bit sont introduites. Les séquences sont les suivantes, avec des erreurs montré en rouge :

$$r_1 = [1101110011101011]$$
$$r_2 = [0101110001111111]$$

La figure 4.9 montre le chemin à travers le treillis pour la séquence sans erreurs R_1 et la séquence rythmique r_2, montrée dans leurs couleurs appropriées. Quand les séquences sont décodées, les informations suivantes sont dérivées :

$$\hat{u}_1 = [1101110011101011]$$
$$\hat{u}_2 = [0101110001111111]$$

Plusieurs bits sont faux dans u_2, qui peut être attribué directement au mauvais chemin à travers le treillis. La matrice métrique de branche pour u_2 sera illustrée par le tableau 4.2, où le nombre entre parenthèses est l'état où la métrique de branchement est calculée

(-)0	1(1)	1(2)	2(1)	1(1)	1(2)	2(3)	2(3)	2(3)
$(-)-\infty$	$(-)-\infty$	3(1)	3(3)	3(3)	3(3)	3(3)	3(3)	N/A
$(-)-\infty$	(1)1	1(2)	1(2)	2(3)	1(2)	1(2)	N/A	N/A
$(-)-\infty$	$(-)-\infty$	3(3)	4(4)	3(3)	4(3)	3(3)	N/A	N/A

Tableau 4.2 – Le tableau montre les paramètres de la branche du schéma de treillis (les chiffres entre parenthèses représentent l'état de calcul de la métrique de branche). En raison de la terminaison du treillis, le dernier métrique de branche est négligée, puisque l'état final est connu.

Comme on l'a vu du dernier état, la mesure calculée indique également le nombre d'erreurs qui se sont produites dans la séquence reçue.

Une fois les mesures ont été calculées, la trace commence à l'état 1 et suit l'optimum

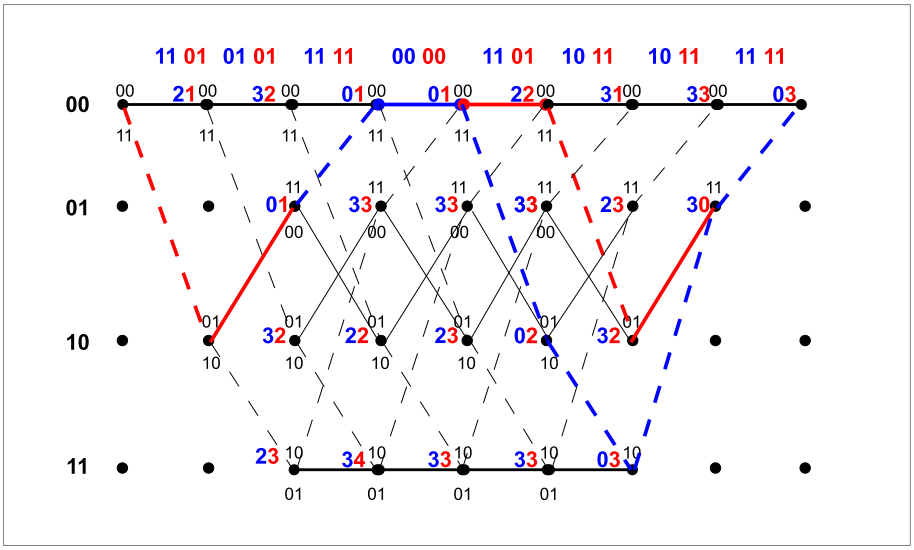

Figure 4.9 – Deux chemins à travers le treillis, étant la vraie séquence transmit représentée en bleue et la version reçue avec erreur en rouge. La couleur fuchsia est utilisée lorsque les deux séquences partagent le même chemin

chemin. Ce chemin est illustré à la figure 4.9. Comme on l'a vu, la séquence reçue avait Quatre bits erronés, mais après le décodage, deux de ces bits ont été corrigés.

Lors du calcul de la métrique de l'état donné, l'algorithme doit seulement savoir quel est l'état actuel et comment la transition serait, d'où l'algorithme ne fonctionne pas sur la séquence complète, uniquement sur deux états, cela rend l'algorithme de Viterbi supérieur aux algorithmes de recherche étendus. Le nombre de chemins différents à travers un treillis augmente exponentiellement avec la longueur, lorsque toutes les branches possibles sont évaluées. Ainsi, la complexité de Viterbi est linéaire avec la longueur mais exponentielle avec le nombre d'états.

L'efficacité du décodeur Viterbi peut être améliorer en faisant le Viterbi utilise un bit soft. La raison de l'utilisation d'une entrée soft, c'est que cela augmente $Eb/N0$ avec $2dB$ pour une résolution fine infinie [13]. Cependant, la complexité des calculs augmente exponentiellement avec la quantification et par conséquent, une quantification de huit bits au plus est habituellement utilisée, ce qui entraînent un gain supérieur à 1,75 dB.

En utilisant cette quantification, un bit sera représenté par les valeurs du tableau 4.3 :

Entrée	Décision
+127	certain 1
+68	probablement 1
0	non plus/ou
-68	probablement 0
-127	certain 0

Tableau 4.3 – Décision soft à 8 bit

Le décodeur Viterbi d'entrée souple utilise l'algorithme MAP pour déterminer si le bit reçu correspond à un 1 ou à un 0 [9].

$$u_k = \begin{cases} 1 & \frac{p(u=+1|r)}{p(u=-1|r)} > 1 \\ 0 & \frac{p(u=+1|r)}{p(u=-1|r)} < 1 \end{cases} \tag{4.17}$$

En utilisant la règle bayes, la condition MAP peut être écrite comme $LLRL(u|r)$.

$$L(u|r) = \ln\left(\frac{p(r|u=+1)}{p(r|u=-1)}\right) + \ln\left(\frac{p(u=+1)}{p(u=-1)}\right) \tag{4.18}$$

où :

— $\ln(\frac{p(r|u=+1)}{p(r|u=-1)}) = L_c(u)$ est la probabilité logique calculée des bits reçus étant 1 ou -1.

— $\ln(\frac{p(u=+1)}{p(u=-1)})$ est le bit a priori d'être $+1$ ou -1.

Compte tenu d'un système de communication symétrique, les deux bits ont une probabilité égale d'etre transmis, d'où le $LLRL(u)$ a priori est 0.

4.5.3 SISO :

Le décodeur SISO qui accepte des entrées d'informations a priori pour générer une sortie soft de décision en $L(u_k)$. La sortie soft est une probabilité qu'un bit soit le estimer correctement, génère donc une probabilité de 0 à 1, qui reflète la certitude

d'une estimation donnée. Cette architecture est illustrée par la figure 4.10.

$$L(\hat{u}_k) = L(u - k|r_k) + L_{ext}(u_k) \tag{4.19}$$

$L(u_k)$ est défini comme la sommation de la probabilité a priori $L_{ext}(u_k)$, également appelé l'information extrinsèque, et la probabilité dérivée trouvée via l'algorithme MAP, appelé valeur intrinsèque. La substitution de l'équation (4.7) pour $L(u_k|r_k)$ dérive comme suit :

$$L(\hat{u}_k) = L_c(r_k) + L(u_k) + L_{ext}(\hat{u}_k) \tag{4.20}$$

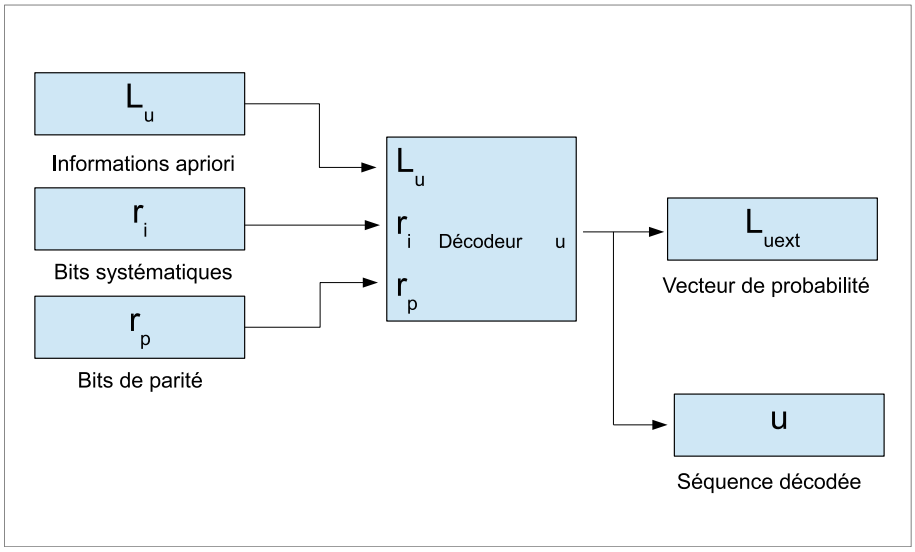

Figure 4.10 – Un décodeur SISO illustré avec ses entrées a priori et la sortie a posteriori $L(\hat{u}_k)$

Où il est évident que $L(u_k)$ se compose de trois éléments, étant $L_c(r_k)$, $L_{ext}(\hat{u}_k)$ et $L(u_k)$. Dans un système non itératif avec une sortie de bit symétrique et aucune fonction favorisant certaines sorties, L_{ext} est égal à 0. Toutefois, comme le décodage Turbo est un processus itératif qui transmet les probabilités de bits au prochain élément décodeur comme information a priori, la valeur extrinsèque joue une partie importante du processus itératif.

En considérant les décodeurs SISO, en relation avec les Turbo Codes, deux variantes sont essentiels pour être couvert :

— L'algorithme BCJR proposé par Bahl, Cocke, Jelinek et Raviv [9] qui requiert une puissance de calcul élevée, mais pour laquelle la sortie est très robuste et bien adapté pour un convoluteur récursif.

— Le SOVA a été proposé par Hagenauer [14] et est largement utilisé dans les Turbo codes comme dernière étape. Il est moins robuste que le BCJR mais nécessite moins puissance de calcul.

4.5.4 BCJR :

L'algorithme BCJR dérive symbole par symbole a posteriori $LLRL(u_k)$ pour la transition de l'état précédent $s^{'}$ 'à l'état actuel s :

$$L(\hat{u}_k) = L(u-k|r) = \ln\left(\frac{\sum_{S+}(P(s^{'},s,r))}{\sum_{S-}(P(s^{'},s,r))}\right) \tag{4.21}$$

Où $P(s^{'},s,r)$ représentent la probabilité indépendante conjointe de la réception de la séquence r. $S-/S+$ indique la sommation comprenant toutes les transitions d'état de $s^{'}$ à s, en raison des bits 1 ou 0, où $-$ indique 0 et $+$ indique 1.

La probabilité conjointe $P(s^{'},s,r)$ peut être évaluée par le produit de trois Probabilités indiquées comme suit :

$P(s^{'},s,r) = \alpha_{k-1}(s^{'})\gamma_k(s^{'},s)\beta_k(s)$ donc :

$$L(\hat{u}_k) = L(u-k|r) = \ln\left(\frac{\sum_{S+}(\alpha_{k-1}(s^{'})\gamma_k(s^{'},s)\beta_k(s))}{\sum_{S-}(\alpha_{k-1}(s^{'})\gamma_k(s^{'},s)\beta_k(s))}\right) \tag{4.22}$$

où

— $\alpha_{k-1}(s^{'}) = P(s^{'}, r_{N<k})$
— $\gamma_k(s^{'},s) = P(u_k)P(r_k|u_k)$
— $\beta_k(s) = P(r_{N>k}|s)$

α, β et γ se rapportent aux probabilités passées, présentes et futures de la séquence r. Les probabilités α et β sont appelées éléments récursifs. α est évalué au fur et à

mesure que la séquence est reçue, du début à la fin du treillis. β agit lorsque la séquence entière est reçue.

Pour calculer $P(u_k)$, la définition de $L_{ext}(u_k)$ est utilisée :
$L_{ext}(u_k) = \ln(\frac{P(u_k=+1)}{P(u_k=-1)}$ donc
$P(u_k) = \frac{\exp(-L_{ext}(u_k)/2)}{1+\exp(-L_{ext}(u_k))} \exp(u_k L_{ext}(u_k)/2)$

$$P(u_k) = A_k \exp(u_k L_{ext}(u_k)/2) \tag{4.23}$$

Afin de calculer $P(r_k|u_k)$ on est besoin de définir :
$r_k = (r_k^s, r_k^p)$ et $z_k = (z_k^s, z_k^p) = (u_k, z_k^p)$

$$\begin{aligned} P(r_k|u_k) &\propto \exp\left(-\frac{(r_k^{s^2}) + u_k^2 + (r_k^{p^2}) + (x_k^{p^2})}{2\sigma^2}\right) \exp\left(\frac{u_k.r_k^s + z_k^p.r_k^p}{\sigma^2}\right) \\ &= B_k.\exp\left(\frac{u_k.r_k^s + z_k^p.r_k^p}{\sigma^2}\right) \end{aligned} \tag{4.24}$$

En combinant les équations (4.21) et (4.22) on aura :

$$\gamma_k(s^{'}, s) \propto A_k B_k \exp(u_k L^e(u_k)/2).\exp(\tfrac{u_k.r_k^s + z_k^p.r_k^p}{\sigma^2})$$

Puisque $\gamma_k(s^{'}, s)$ apparaît dans le numérateur et le dénominateur de 3.20 les termes A_k et B_k vont etre annulé car ils sont indépendants du u_k. Ainsi, l'expression s'effondre jusqu'à :

$$\begin{aligned} \gamma_k(s^{'}, s) &\propto \exp\left(1/2.u_k.(L_{ext}(u_k) + L_c.r_k^s) + 1/2.L_c.r_k^p.z_k^p\right) \\ &= \exp\Big(1/2.u_k.(L_{ext}(u_k) + L_c.r_k^s)\Big).\gamma_k^e(s^{'}, s\Big) \end{aligned} \tag{4.25}$$

Où $Lc = \frac{4E_c}{N_0}.a$, E_c est l'énergie transmise par un bit codé et a est la fonction de canal. Maintenant, le terme récursif et le terme récursif rétro-actif $\alpha_{k-1}(s')\gamma_k(s^{'}, s)$ et $\beta_k(s)\gamma_k(s^{'}, s)$

peuvent être calculés :

$$\alpha_k(s) = \sum_{s' \ni S} \alpha_{k-1}(s')\gamma_k(s', s) \tag{4.26}$$

où la valeur initiale de alpha est : $\alpha_0(0) = 1$ et $\alpha_0(s \neq 0) = 0$

$$\beta_{k-1}(s') = \sum_{s' \ni S} \beta_k(s)\gamma_k(s', s) \tag{4.27}$$

Qui a des conditions limites $\beta_N(0) = 1$ et $\beta_N(s \neq 0) = 0$, ce qui signifie Que β devrait se terminer dans l'état 0 après N bits d'entrée, en raison de la terminaison du treillis.

En raison de l'annulation de paramètres communs dans le dénominateur et de la numérateur dans l'équation (4.19), une normalisation des paramètres α et β est nécessaire afin d'éviter que l'algorithme ne devienne instable. Par conséquent, les termes $\alpha_k(s)$ et $\beta_k(s)$ sont utilisés pour définir les probabilités modifiées $\tilde{\alpha}_k(s)$ et $\tilde{\beta}_k(s)$ [15].

$$\tilde{\alpha}_k(s) = \frac{\sum_s' \tilde{\alpha}_{k-1}(s')\gamma_k(s', s)}{\sum_s \sum_s' \tilde{\alpha}_{k-1}(s')\gamma_k(s', s)} \tag{4.28}$$

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_s' \tilde{\beta}_k(s)\gamma_k(s', s)}{\sum_s \sum_s' \tilde{\alpha}_{k-1}(s')\gamma_k(s', s)} \tag{4.29}$$

Par conséquent :

$$L(\hat{u}_k) = L(u - k|r) = \ln\left(\frac{\sum_{S+}(\tilde{\alpha}_{k-1}(s')\gamma_k(s', s)\tilde{\beta}_k(s))}{\sum_{S-}(\tilde{\alpha}_{k-1}(s')\gamma_k(s', s)\tilde{\beta}_k(s))}\right) \tag{4.30}$$

ou bien :

$$L(\hat{u}_k) = L_c.r_k^s + L_{ext}(u_k) + \ln\left(\frac{\sum_{S+}(\tilde{\alpha}_{k-1}(s')\gamma_k^e(s', s)\tilde{\beta}_k(s))}{\sum_{S-}(\tilde{\alpha}_{k-1}(s')\gamma_k^e(s', s)\tilde{\beta}_k(s))}\right) \tag{4.31}$$

L'algorithme MAP est très lourd en calcul, de sorte que Koch et Bayer ont proposé un algorithme Max-Log-MAP, qui a réduit la complexité, mais aussi la performance [9]. En raison des approximations réalisées dans l'algorithme Max-Log-MAP, la performance est assez faible par rapport à l'algorithme MAP, donc Robertson a proposé l'algorithme Log-MAP, qui utilise une table de consultation afin de calculer $\alpha_k(s)$ et $\beta_k(s)$ précisément plutôt que d'utiliser des approximations. Cela augmente la performance à presque le niveau de l'algorithme MAP, tout en ayant une faible complexité. Cependant, l'algorithme Log-MAP est limité par la taille croissante de la table de recherche [9].

4.5.5 SOVA :

L'algorithme SOVA peut être utilisé pour optimiser la vitesse avec perte de performance. Le SOVA augmente l'efficacité de l'algorithme original de Viterbi en ajoutant des informations extrinsèques L_{ext} de la séquence reçue à l'entrée du décodeur Viterbi. La métrique de la branche est calculée avec l'algorithme Max-Log-MAP 3.21, puisque ceci donne la probabilité maximale a posteriori pour chaque bit [14].

$$M(S_k^s) = M(S_{k-1}^{s'}) + 1/2.u_k.L(u_k) + \frac{L_c}{2}\sum_{l=0}^{n-1} r_{k,l}.z_{k,l} \tag{4.32}$$

où :

- u_k est le bit d'information correspondant à la transition donnée.
- L_c est la mesure de fiabilité des canaux.
- $L(u_k)$ est l'information a priori.
- $\sum_{l=0}^{n-1} r_{k,l}.z_{k,l}$ est la corrélation des bits reçus $r_{k,l}$ et les bits émises $z_{k,l}$ au temps k, où l est le symbole de $r_{k,l}$ et $z_{k,l}$ et n est le nombre de symboles par transmission sur le canal.

Comme on l'a vu, la métrique inclut l'information a priori sur le canal et il contient la fiabilité du canal L_c. L_c est important, car cela décide que la décision soit fondée sur l'information a priori ou a posteriori. Si le canal est bon, L_c est élevé, ce qui provoque l'information a posteriori Pour être plus significatif, sinon L_c sera faible et forçant le résultat de l'algorithme à Être déterminé par l'information a priori.

En outre, il est noté que le résultat de L_c ne change pas pour chaque itération, il suffit donc d'être calculé une fois. Pour un processus itératif , cela peut diminuer la vitesse et augmenter l'usage de la mémoire.

L'algorithme de décision soft de Viterbi aura le flux suivant [11] :

1. pour tout les n :
   - Calculer les paramètres de la branche $M(S_k^s)$.
   - Trouver le meilleur chemin $max_m(M(S_k^{s(m)}))$ pour chaque état, où les valeurs sont différentes pour le même état.
   - Enregistrer $max_m(M(S_k^{s(m)})$ et son correspondant chemin de survie.
   - Calculer et enregistrer $\Delta_k^s = max_m(M(S_k^{s(m)})) - M(S_k^{s(m)})$.
2. Effectuez une traçabilité à travers le treillis, à partir de l'état 0 et suivant la matrice de cheminement survivante, révélant ainsi la séquence de transmissions la plus susceptible.

Lorsque le traçage est terminé, $min(\Delta_k^s)$ pour chaque k sera égal à $L(\hat{u}_k)$.

4.6 Architecture du décodeur Turbo :

Le décodeur Turbo utilise des bits de décision soft comme entrée et crée un bit de décision dure en sortie. Le processus est itératif et profite du fait que les trois versions transmises sont fortement non corrélées, par les différentes erreurs.

Contrairement au codeur PCCC, qui a les N décodeurs convolutionnels en parallèle, le décodeur utilise un couplage en série de N décodeurs convolutionnels, comme le montre la figure 4.11.

Avec :

— Signaux :

- r_k^i est la séquence d'entrée originale.
- r_k^p équivalent à z_k convolutionné avec le décodeur 1.
- $r_{p.2}^{'}$ est z_k entrelacé puis convolutionné avec le décodeur 2.
- $L(u_k)$ est l'estimation de la séquence actuelle.
- Les blocs de décodeurs provoqueront un certain retard au système.

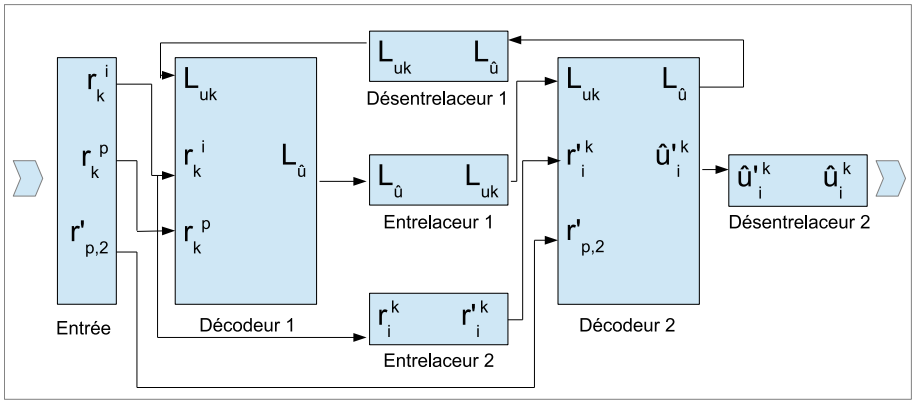

Figure 4.11 – Illustration de la structure d'un décodeur Turbo.

— Blocs :

— Décodeur 1 : prend les entrées r_k^i, r_k^p, $L(u_k)$ et les décode en utilisant un algorithme SISO. La sortie de ce bloc est un vecteur qui désigne la deuxième estimation du décodeur de la séquence transmise. Pour la première itération, $L(u_k)$ est met à 0.

— Entrelaceur : est implémenté car la sortie du décodeur 1 est un vecteur de décision soft qui décrit essentiellement une estimation de la séquence réelle.

— Décodeur 2 : utilise l'algorithme SISO pour décoder ses entrées ; L_1, $\tilde{r}_{p,2}(n)$ et $\tilde{r}(n)$. La sortie du décodeur 2 est donc une autre itération l'estimation, avec l'information a priori du premier décodeur. Quand de nombreuses itérations sont faites, une décision dure est produite.

— Le désentrelaçeur s'assure que la sortie du décodeur 2 peut être utilisé dans le décodeur 1. Pour ce faire, il doit être désentrelacé, tel que les index correspondent aux autres entrées de ce bloc.

Le nombre des itérations peut être décidé soit par un nombre fixe, soit par un seuil pour le BER. Normalement, il faut 4-12 itérations avant que la précision souhaitée ne soit atteinte, selon la longueur de la séquence. Le nombre d'itérations à une grande influence sur la performance du décodeur. S'il ya peu d'itérations la performance souhaitée ne l'atteindra pas, ce qui entraîne un faible débit. D'autre part, si de nombreuses itérations sont faites, le système introduit un délai inutile des données.

L'architecture du décodeur Turbo introduit une flexibilité dans le sens que les différents algorithmes ont chacun leurs propres avantages et inconvénients. Bien que les deux algorithmes MAP fonctionnent bien à un faible SNR, ils sont très complexes et exigent des ressources, les deux SOVA sont moins complexes, à un coût de performance. Il est présenté comment les différents paramètres influent sur la performance, en termes

d'étendre les itérations et comment les différents algorithmes peuvent être utilisés dans l'effort de l'optimisation du système.

Chapitre 5

Résultats des simulations :

Ce chapitre est consacré aux résultats obtenus lors des simulations, ainsi qu'aux discussions des différents résultats. Les résultats sont divisés en quatre sections : système OFDM, canal de transmission, estimation/égalisation du canal et codage/décodage turbo basé sur l'algorithme BCJR.

5.1 Paramètres de simulation :

Les paramètres de simulation sont répartis sur les quatre sections citées ci-dessus :

1. Le système OFDM :
   - Le nombre de sous-porteuses.
   - Le préfixe cyclique.
2. Le canal de transmission :
   - Le type de modulation.
   - La periode du symbole.
   - Les delais du retard.
   - Les puissances relative au delais du retard.
   - La fréquence Doppler.
   - Le rapport signal au bruit.
3. L'estimation/égalisation du canal :
   - L'intervalle entre les symboles pilotes.

— La variance du bruit.

4. Le codage/décodage turbo :

— Les bites systématiques.

— Les bites de parité.

— Le nombre d'itérations.

Puisque chacun de ces paramètres influe directement sur les résultats de la simulation,on va simuler l'effet de chaque groupe de paramètres avant de fixer leurs valeur.

5.1.1 L'effet du nombre de sous-porteuses :

La figure 5.1, démontre l'impacte du nombre de sous-porteuses sur la performance du système, on voit clairement qu'il est directement proportionnel aux valeurs de N :

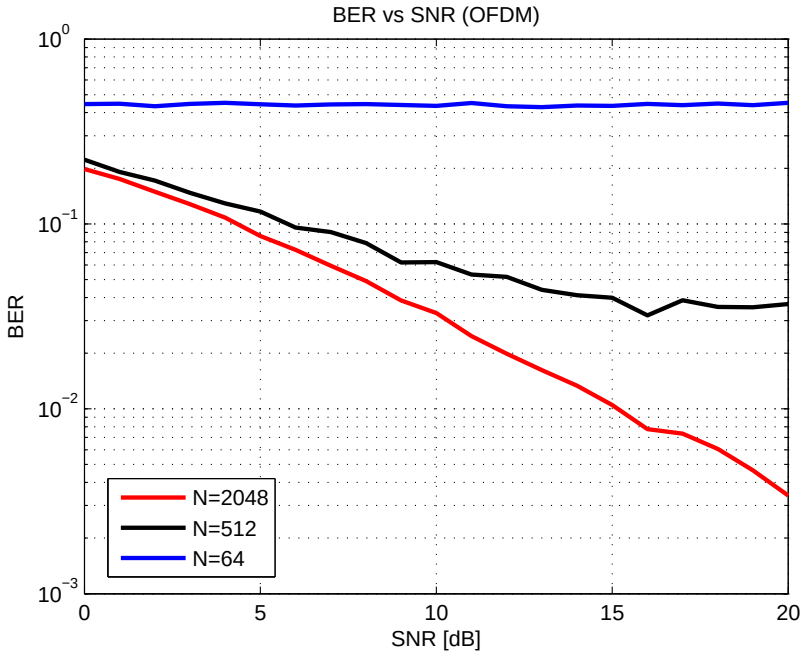

Figure 5.1 – L'effet du nombre de sous-porteuses sur la performance du système OFDM.

Pour le reste du projet le nombre de sous-porteuses est fixé à $N = 2048$.

5.1.2 L'effet de modulation :

Le nombre de bits formant la constellation provoque de nombreux erreurs au fur et à mesure avec sa croissance, La figure 5.2 illustre ce phénomène :

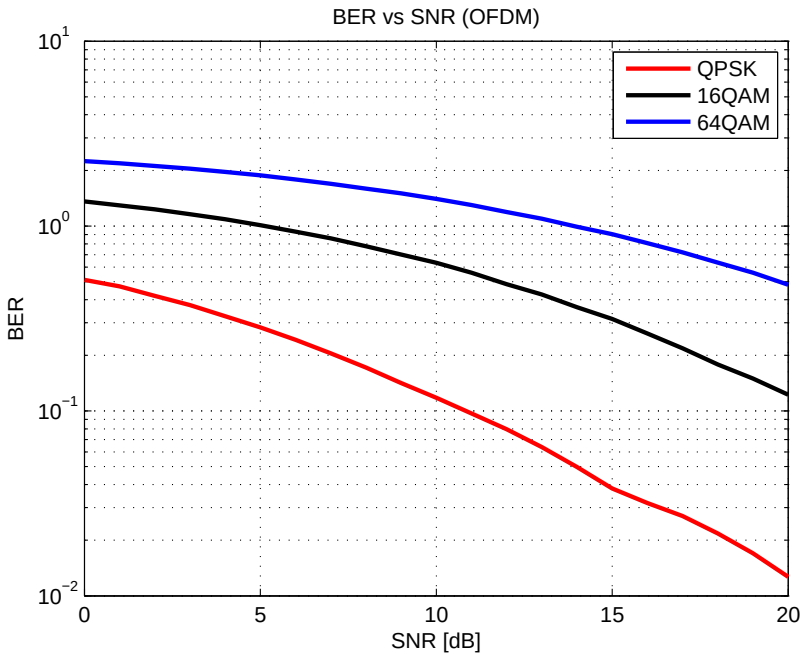

Figure 5.2 – L'effet de la taille de la constellation sur la performance du système OFDM.

Pour le reste du projet on va choisir la QPSK pour le canal moin séléctif (EPA) et la 64QAM pour le canal trés sélectif (ETU), c-à-d le cas le plus favorable VS le plus défavorable.

5.1.3 L'effet du canal :

L'effet du canal dépend essentiellement du vecteur des délais du retard et la fréquence Doppler relatives à ce canal, pour cette simulation on a fait choisir les modèles du système LTE, les modèles les plus proche de la réalité, on distingue trois modèle :

1. Le canal EPA :
   - Vecteur des delais du retard $= [0 \quad 30 \quad 70 \quad 90 \quad 110 \quad 190 \quad 410] * 1e-9\ ns$.
   - Vecteur des puissances $= [0 \quad -1 \quad -2 \quad -3 \quad -8 \quad -17.2 \quad 20.8]\ dB$.
   - La fréquence Doppler $= 5\ Hz$.

2. Le canal EVA :
   - Vecteur des delais du retard $= [0 \quad 30 \quad 150 \quad 310 \quad 370 \quad 710 \quad 1090 \quad 1730 \quad 2510] * 1e-9\ ns$.
   - Vecteur des puissances $= [0 \quad -1.5 \quad -1.4 \quad -3.6 \quad -0.6 \quad -9.1 \quad -7 \quad -12 \quad -16.9]\ dB$.
   - La fréquence Doppler $= 70\ Hz$.
3. Le canal ETU :
   - Vecteur des delais du retard $= [0 \quad 50 \quad 120 \quad 200 \quad 230 \quad 500 \quad 1600 \quad 2300 \quad 5000] * 1e-9\ ns$.
   - Vecteur des puissances $= [-1 \quad -1 \quad -1 \quad 0 \quad 0 \quad 0 \quad -3 \quad -5 \quad -7]\ dB$.
   - La fréquence Doppler $= 300\ Hz$.

On voit que le retard max varie entre $400\ ns$ pour le canal EPA et $5000\ ns$ pour le canal ETU, la figure 5.3 représente l'effet de ce retard sur la performance du système :

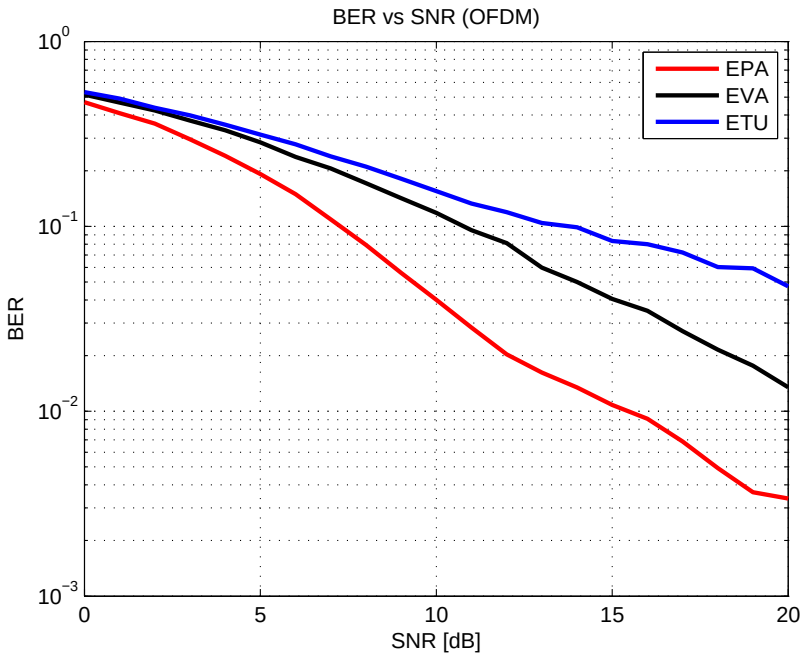

Figure 5.3 – L'effet canal sur la performance du système OFDM.

Pour le reste du projet, on va prendre les deux cas extremes (EPA et ETU) avec les modulations QPSK et 64QAM respectivement.

En ce qui concerne les paramètres du codage/décodage turbo, on a pris les valeurs suivant :

— Les bites systématiques $= 2 * [0, 1; 0, 1; 0, 1; 0, 1] - 0.5$.

— Les bites de parité $= 2 * [0, 1; 1, 0; 0, 1; 1, 0] - 0.5$.

— Le nombre d'itérations $= 4$.

5.1.4 L'effet de la méthode d'estimation :

On a décrit plusieurs méthodes d'estimation du canal dans le chapitre III de cette thèse dont les plus célèbre sont les méthodes LS et le $LMMSE$, la différence entre l'effet des deux méthodes est illustré dans la figure 5.4 :

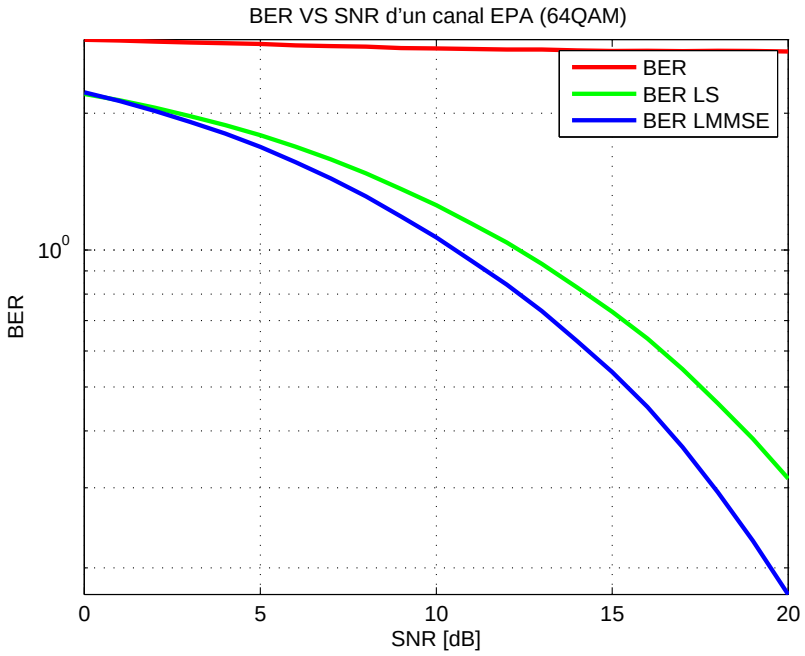

Figure 5.4 – L'effet du la méthode d'estimation sur la performance du système OFDM.

Pour le rest du projet on appliquera l'estimateur LMMSE.

5.2 La construction du programme :

5.2.1 Les boucles :

Le programme contenant deux boucle principales :

— La boucle du snr allant de 0 à 20 avec un pas de 1 dB.
— La boucle de monte-carlo allant de 1 à 1000 avec un pas de 1.

5.2.2 Génération des données :

A l'aide des fonction *rand* et $floor$ on génère une séquence de bit X (des 1 et des 0) d'une dimension de ($Le\ nombre\ de\ sous-porteuses\ *\ l'ordre\ de\ modulation$) x 1, où $l'ordre\ de\ modulation = log_2(M)$ et M représente le nombre de bits par constellation.

5.2.3 Codage des données :

Entrelacement :

La fonction *randperm* permet la génération d'un vecteur contenant une permutation aléatoire des nombres entiers de 1 à ($Le\ nombre\ de\ sous-porteuses * l'ordre\ de\ modulation$), ce vecteur sert à entrelacer la séquence d'entrée X pour nous donner la séquence entrelacée Xe.

Codage :

Selon la méthode citée dans la section 4.3.1, on fait passer la séquence d'entrée X au premier encodeur convolutionnel, alors que la séquence entrelacée Xe sera passer au deuxième encodeur convolutionnel.

$P0$ et $P1$ sont les séquences de sortie des deux encodeurs convolutionnels respectivement.

Il est noté que les processus qui suit s'apliquent au trois séquences X, $P0$ et $P1$ pour qu'ils soient à l'entrée du décodeur itératif.

5.2.4 Formation des symboles binaires :

Les symboles sont formés en appliquant la fonction *reshape* selon le tableau 2.1, les symboles créées contenant deux bit s'il s'agit d'une modulation $QPSK$ et six bit s'il s'agit d'une $64QAM$.

On aura donc trois séquences de symboles d'une dimension de ($Le\ nombre\ de\ sous-porteuses * l'ordre\ de\ modulation$).

5.2.5 Conversion binaire-décimal :

Grace à la fonction *bi2de* on converti les symboles binaires à des symboles décimaux d'une dimendion ($Le\ nombre\ de\ sous-porteuses$ x 1).

5.2.6 La modulation :

A l'aid des objets de modulation créées par l'application des fonctions *pskmod* et *qammod*, on module les symboles décimaux pour avoir les séquences des symboles modulés.

La figures 5.5 illustre les constellations des symboles modulés en QPSK et en 64QAM :

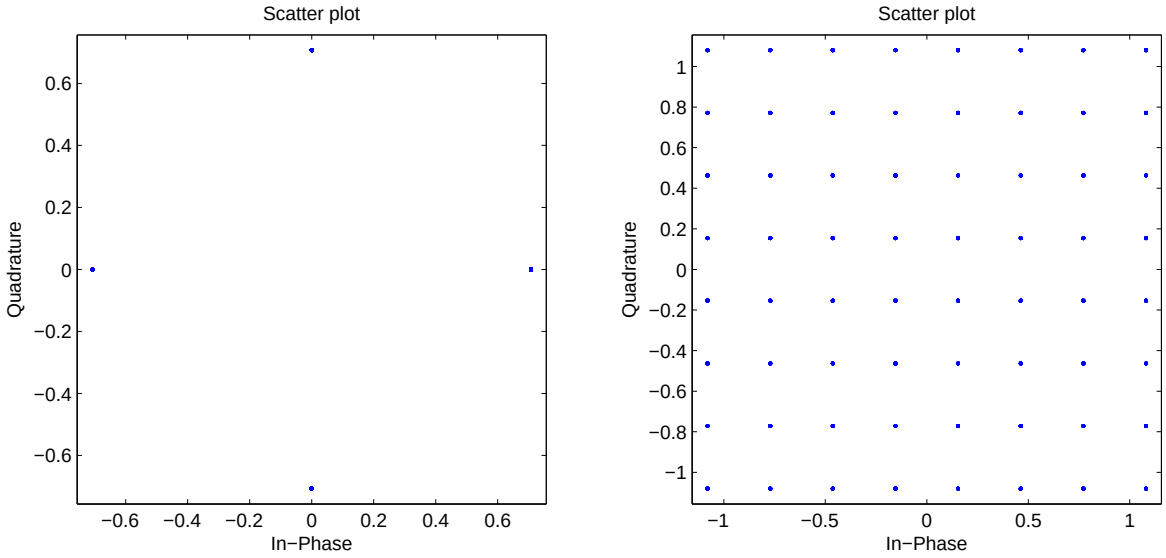

Figure 5.5 – Constellations des symboles modulés en QPSK et 64QAM.

5.2.7 Insertion des pilotes :

On choisit les positions allant de premier symbole au dernier avec un pas de huit symboles pour chacune des séquences modulées, cela nous donne 256 position dans chaque séquence.

5.2.8 L'IFFT :

Les symboles modulés subissent une transformation de Fourier rapide inverse générée par la fonction $ifft$ dont le but est de former des symboles OFDM.

5.2.9 L'ajout du préfixe cyclique :

On prend le dernier quart de chaque séquence et le remettre au début de la séquence, cela permet comme il est mentionné au deuxième chapitre, de transformer la convolution

linéaire en une convolution circulaire afin d'éviter le chevauchement entre le dernier symbole de la séquence précédente et le premier de la séquence en cours.

Cette opération permet la création des séquences d'une dimension (*Le nombre de sous−porteuses + Prefix cyclique*) x 1.

5.2.10 Le passage au canal :

La fonction *rayleighchan* nous donne l'opportunité de créer un canal de Rayleigh personnalisé en fonction des délais de retard, profile de puissances correspondant, la fréquence Doppler et la période du symbole.

Les symboles OFDM seront filtrés par ledit canal pour nous donner les séquences filtrées.

Supposant le signal est stable durant la période du symbole, on peut déduire l'évolution du canal créée à l'aide coefficients *PathGains* engendrés par le filtre du canal. La figure 5.6 illustre un exemple de la réponse fréquentielle dudit canal :

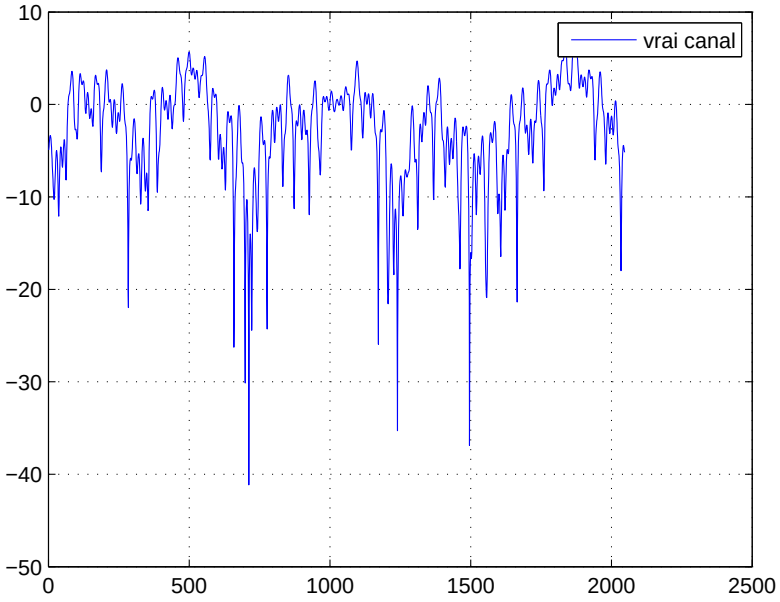

Figure 5.6 – La réponse fréquentielle du canal ETU.

5.2.11 L'ajout du bruit :

D'abord on calcule la puissance des symboles à l'aide de la fonction *mean*, puis le bruit appliqué à chaque séquence en fonction des puissances calculées et la fonction *randn*.

Chaque bruit sera ajouté à la séquence qui lui correspond afin d'introduire les séquences reçues.

5.2.12 Enlèvement du préfixe cyclique :

On enleve les préfixes cycliques premier symboles de chaque séquence pour garder seulement les symboles d'iformation.

5.2.13 FFT :

Aprés la suppression du préfixe cyclique les symboles subissent une transformation de Fourier rapid générée par la fonction fft dont le but est de récupérer les symboles modulés.

5.2.14 L'estimation LMMSE du canal :

L'estimation :

Suivant la section 2.2 du troisième chapitre l'estimation est réalisée en faisant les opérations suivantes :

- Récupérer les symboles pilotes reçus suivant leurs positions citées au-dessus.
- Créer des matrices diagonales en utilisant ces symboles pilotes reçus.
- Déduire les matrices d'estimation en faisant la division des matrices diagonales par le transposé des vecteurs des symboles pilotes reçus.
- Calculer les matrices Rgg en appliquant une autocorrélation sur les matrices d'estimation à l'aide de la fonction ($corrmtx$).

— Déduire les matrices d'estimation LMMSE suivant l'équation (3.18) du troisième chapitre tenant en compte la variance du bruit, les matrices Rgg, les matrices diagonales et les matrices d'estimation.

L'interpolation :

On applique la fonction $interp1(\ldots, (1 : Le\ nombre\ de\ sous - porteuses), 'spline')$ sur les matrices d'estimation LMMSE, on obtiendra les séquences interpolées de dimension $Le\ nombre\ de\ sous - porteuses$ x 1.

L'égalisation :

Cela est fait simplement par la division des séquences récupérées après la transformation de Fourier rapid par les séquences interpolées.

La figure 5.7 présente une comparaison du canal estimé avec le vrai canal.

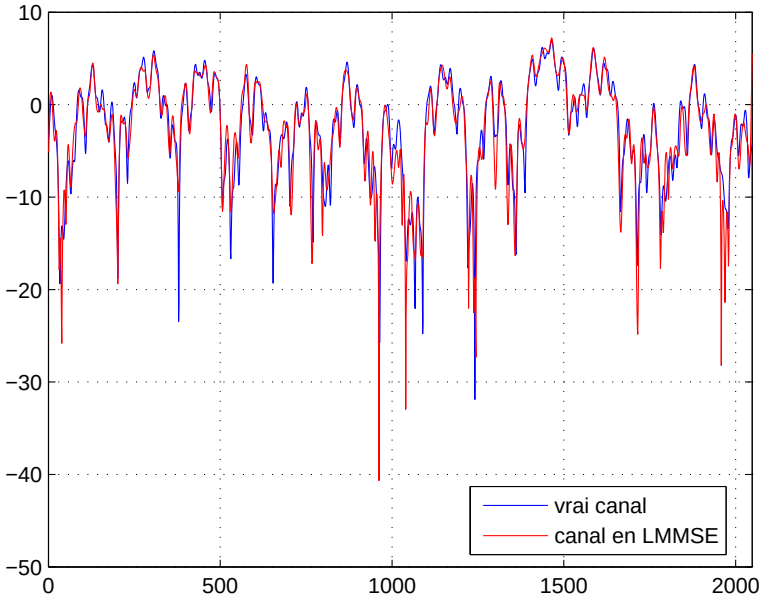

Figure 5.7 – Comparaison du canal estimé avec le vrai canal.

5.2.15 La démodulation :

A l'aid des objets de démodulation créées par l'application des fonctions *pskdemod* et *qamdemod*, on démodule les symboles estimés pour avoir les séquences des symboles démodulés.

5.2.16 Conversion décimal-binaire :

Grace à la fonction *de2bi* on converti les séquences binaires à des séquences décimales d'une dimendion $Le\ nombre\ de\ sous-porteuses$ x $l'ordre\ de\ modulation$.

5.2.17 Récupération des données codées :

On utilise la fonction $rechape(\ldots, numel(\ldots))$ pour récupérer les séquences codées.

Ces séquences vont attaquer le décodeur turbo afin de récupérer la séquence des données originales transmises.

5.2.18 Décodage itératif des données récupérées :

Inicialisation de l'algorithme BCJR :

A l'entrée du décodeur, on introduit trois singaux $R0$, $R1$ et $R2$ représentants les reçus corresponds aux bits transmis, les bits reçus corresponds aux bits de parité du premier codeur convolutionnel et les bits reçus corresponds aux bits de parité du deuxième codeur convolutionnel respectivement. $R0$, $R1$ et $R2$ sont calculés en multipliant les signaux reçus codés par une séquence aléatoire.

On fait l'entrelacement de la première entrée $R0$ pour engendrer $(R0 - pi)$ (voir Figure 4.11). On fait initialiser l'indicateur BCJR à 0 et le vecteur Apriori à des 1, puis le multiplier par 1/2.

La fonction GAMMA :

Cette fonction sert à calculer γ en fonction du $nombre\ de\ sous-porteuses$, le vecteur Apriori, les bits systématiques de l'encodeur convolutionnel, les bits de parité de l'encodeur convolutionnel, SNR, $R0$ et $R1$.

Elle génère une matrice g de dimension 4 x $2 * lenombre\ de\ sous-porteuses$:

où :

$g(j, 2*i) = Apriori(1,i) * (1/(2*\pi)) * \exp(-0.5 * T_j)$

$T_j == ((R0(i) - sqrt(SNR) * (les\ bits\ systmatiques(k,l)))^2) + ((R1(i) - sqrt(SNR) * (les\ bits\ de\ parit(k,l)))^2)$ avec k et l allant de 1 à 4 et j allant de 1 à 8.

La fonction ALPHA :

Cette fonction sert à calculer α en fonction de N et γ.

Elle génère une matrice a de dimension 4 x N :

où :

$a(1,i) = ((GAMMA(l,j) * a(1,i-1)) + (GAMMA(3,j+1) * a(3,i-1)))$.

$a(2,i) = ((GAMMA(3,j) * a(3,i-1)) + (GAMMA(1,j+1) * a(1,i-1)))$.

$a(3,i) = ((GAMMA(2,j) * a(2,i-1)) + (GAMMA(4,j+1) * a(4,i-1)))$.

$a(4,i) = ((GAMMA(4,j) * a(4,i-1)) + (GAMMA(2,j+1) * a(2,i-1)))$.

La fonction BETA :

Cette fonction sert à calculer β en fonction de $nombre\ de\ sous-porteuses$ et γ.

Elle génère une matrice b de dimension 4 x $nombre\ de\ sous-porteuses$:

où :

$b(1,i) = ((GAMMA(1,j) * b(1,i+1)) + (GAMMA(1,j+1) * b(2,i+1))).$

$b(2,i) = ((GAMMA(2,j) * b(3,i+1)) + (GAMMA(2,j+1) * b(4,i+1))).$

$b(3,i) = ((GAMMA(3,j) * b(2,i+1)) + (GAMMA(3,j+1) * b(1,i+1))).$

$b(4,i) = ((GAMMA(4,j) * b(4,i+1)) + (GAMMA(4,j+1) * b(3,i+1))).$

La fonction LAPPR :

Cette fonction sert à calculer LAPPR en fonction de $nombre\ de\ sous-porteuses$ et γ.

Elle génère une matrice l de dimension 1 x $nombre\ de\ sous-porteuses$:

où :

$l(i) = log(p1(i)/p0(i)).$

$p0(i) = (ALPHA(1,i) * GAMMA(1,2*i-1) * BETA(1,i)) + (ALPHA(2,i) * GAMMA(2,2*i-1) * BETA(3,i)) + (ALPHA(3,i) * GAMMA(3,2*i-1) * BETA(2,i)) + (ALPHA(4,i) * GAMMA(4,2*i-1) * BETA(4,i)).$

$p1(i) = (ALPHA(1,i) * GAMMA(1,2*i) * BETA(2,i)) + (ALPHA(2,i) * GAMMA(2,2*i) * BETA(4,i)) + (ALPHA(3,i) * GAMMA(3,2*i) * BETA(1,i)) + (ALPHA(4,i) * GAMMA(4,2*i) * BETA(3,i)).$

Première itération :

Pour la première itération on met $BCJR = 0$ et $Apriori = ones(2,N) * 0.5$ et suivre les étapes suivantes :

— Calculer γ en fonction de $R0$ et $R1$.

— Calculer α en fonction de γ.

— Calculer β en fonction de γ.

— Calculer $LAPPR$ en fonction de α, β et γ.

— Mettre les valeurs positives de $LAPPR$ à 1 ($LAPPR$ représente les bits décodés).

— Calculer le ber de la première itération en comparant $LAPPR$ avec la séquence originale X.

— Entrelacer les bits de $LAPPR$.

— Remettre l'inverse de $1 + \exp(LAPPR)$ à la première ligne de $LAPPR$.

— Remettre $\exp(LAPPR)/(1 + \exp(LAPPR))$ à la deuxième ligne de $LAPPR$.

— Changer la valeur de $BCJR$. (la mettre à 1).

L'itérativité :

Aprés l'incrémentale de *iteration* l'algorithme BCJR refaire les meme étapes jusqu'à la valeur maximale de *iteration*, mais cette fois ci il fait quelques changements sur le processus a savoir :

— Le $LAPPR$ de l'itération précédente sera considéré comme séquence d'entrée.

— γ sera calculé en fonction de $R0 - pi$ et $R2$.

— Ne pas entrelacer les bits de $LAPPR$.

A la fin de la boucle d'itération on aura quatre vecteurs contenant le ber de chaque itération.

Les figures 5.8 et 5.9 démontrent les résultats obtenus en appliquant le programme sur un canal EPA avec une modulation QPSK et un canal ETU avec une modulation 64QAM respectivement.

5.3 Analyse des résultats :

Comme on le voit sur la figure 5.8, on peut voir qu'aprés la première itération, le ber atteint 10^{-5} pour un snr de 10.5 dB, ce qui présente une grande amélioration des performances par rapport au résulta de l'égaliseur LMMSE où le ber n'a pas eu atteindre que 10^{-2} pour la meme valeur du snr. Cela prouves la puissance du décodeur meme avant d'entamer les itérations qui suivent.

Aprés la première itération on remarque que les courbes des différentes itérations commence a se diverger à partir de $snr = 2\ dB$ avec une amélioration du performance

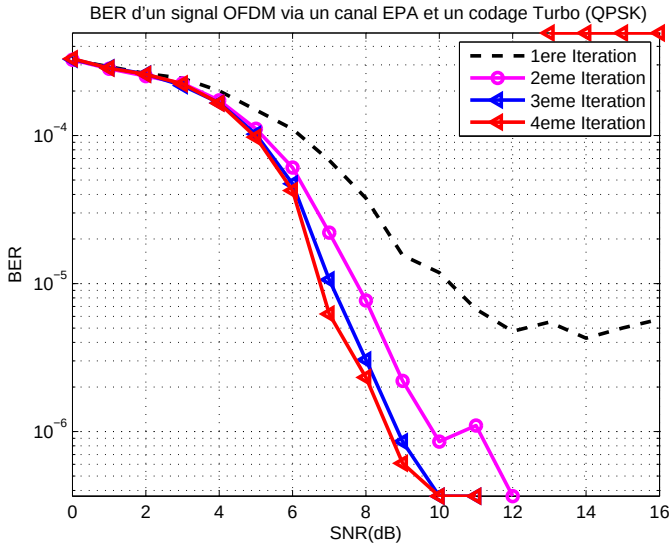

Figure 5.8 – Turbo décodage des symboles OFDM modulés en QPSK via un canal EPA.

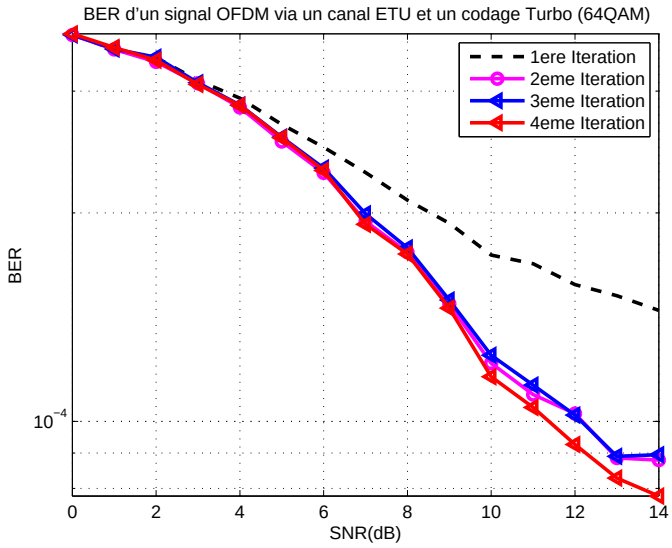

Figure 5.9 – Turbo décodage des symboles OFDM modulés en QPSK via un canal EPA.

pour chaque itération. Par exemple si on prend la valeur de 10^{-5}, on voit qu'elle est atteinte à $snr = 10.5\ dB$ pour la première itération, hors que cette valeur est atteinte à $snr = 8\ dB$ pour la deuxième itération, à $snr = 7\ dB$ pour la troisième itération et à $snr = 6.5\ dB$ pour la quatrième itération. Cela nous indique qu'il ya une amélioration des performances à chaque nouvelle itération, mais cette amélioration est décroissante ce qui explique la limitation du nombre d'itérations.

On remarque aussi que chaque itération converge vers une valeur du ber malgré la croissance du snr, et que cette valeur de convergence s'abaisse au fur et à mesure avec les itérations, elle est à l'ordre de 10^{-5} pour la première itération, de 10^{-6} pour la deuxième et la troisième itération, alors qu'elle s'approche de 10^{-7} pour la quatrième itération.

Finalement, on peut dire que le BER est proche des valeurs attendues. Il est également noté que pour plus d'itérations, le BER semble diminuer pour la même valeur de snr, mais la performance semble stagner autour de quatrième itérations et devient négligeable.

Pour la figure 5.9, on observe la même évolution des courbes avec des valeurs du BER plus élevées dûes à l'effet du type de modulation et le canal choisit.

Chapitre 6

Conclusion :

Avec le besoin grandissant des services de télécommunications sans fil à grande vitesse, l'OFDM a suscité des intérêts croissants en raison de sa robustesse aux évanouissements par trajets multiples et sa capacité de réaliser une efficacité élevée de transmission. Les excellentes performances de l'OFDM ont justifié son adoption dans plusieurs standards pour les systèmes sans fil actuels et futurs.

Dans ce projet de fin d'études, nous nous sommes intéressés à l'étude des performances d'un système OFDM sur différents modèles du canal radio mobile LTE. Ce canal en plus d'etre sélectif en fréquence, peut être à faible, moyenne ou forte sélectivité temporelle. Face à l'hostilité d'un tel canal, nous avons comparé le gain apporté en BER par l'utilisation de la techniques d'estimation du canal MMSE, et par l'utilisation d'un codage canal puissant comme le turbo code.

Au chapitre 3, nous avons présenté cinq techniques d'estimation de canal OFDM, à savoir les estimateurs LS, LSM, LMMSE, LMMSEM et ML. Dans ce chapitre, nous avons concentré notre attention sur les estimateurs LS et LMMSE ainsi que sur l'estimateur ML. Dans les systèmes OFDM, la plupart des méthodes d'estimation de canal utilisent des symboles pilotes. Employer plus de pilotes aide à améliorer la précision de l'estimation mais réduit l'efficacité de transmission. Ainsi, il est souhaitable de garder le nombre de pilotes aussi bas que possible. Au chapitre 4, nous avons présenté les turbo codes en commençant par les codes convolutifs, les entrelaceurs et les différents algorithmes de décodage itératif a savoir le Viterbi, le SISO, le BCJR et le SOVA, tout en notant les avantages et les inconvénients de chaque algoritme.

Les simulations réalisées et exposées au chapitre 5, ont montrées que dans les systèmes OFDM, les techniques robustes d'estimation de canal basées sur les symboles

pilotes exigent typiquement un nombre de symboles pilotes relativement grand pour une estimation précise. De tels estimateurs de canal doivent pouvoir suivre les variations du canal en temps et en fréquence pour que le système fonctionne avec succès dans divers environnements radio. D'autre part, Le principe des turbo codes, comme tout code correcteur d'erreur, est d'introduire une redondance dans le message afin de le rendre moins sensible aux bruits et perturbations subies lors de la transmission. Le codage consiste à utiliser deux codeurs simples, dont les entrées ont été entrelacées ; ainsi, chaque codeur voit une série d'informations différentes à son entrée. Le décodage est le fruit de la collaboration des décodeurs. Ceux-ci vont s'échanger de l'information de manière itérative afin d'améliorer la fiabilité de la décision qui sera prise pour chaque symbole. La séquence reçue, entre dans le premier décodeur, et donne des valeurs soft. Ce qui provient de la séquence reçue est supprimé, pour ne garder que les informations extrinsèques apportées par le décodeur. Puis les données reçus et les données extrinsèques sont entrelacées et fournies à l'entrée du second décodeur. Une itération consiste en l'activation de chaque décodeur une fois. Ainsi, plus il y a d'itérations, plus le décodeur convergera vers la bonne solution ; cependant effectuer plus d'itérations nécessite souvent plus de temps et plus de matériel.

En tout, plusieurs recherches sur les systèmes OFDM et les turbo codes sont encore ouvertes. Nous espérons que ce travail a fourni des éclaircissements sur certains de ces problèmes et fournira une base pour de futures recherches.

.

Bibliographie

[1] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communication Systems Part I : Characterization," *IEEE Communications Magazine*, vol. 35, pp. 90–100, 1997.

[2] A. Peled et A. Ruiz, "Frequency Domain Data Transmission using Reduced Computational Complexity Algorithms," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 964–967, 1980.

[3] T. Pollet, M. van Bladel et M. Moeneclaey, "BER Sensitivity of OFDM Systems to Carrier Frequency Offset and Wiener Phase Noise," *IEEE Transactions on Communications*, vol. 43, pp. 187–190, 1995.

[4] J. J. Van de Beek, O. Edfors, M. Sandell, S. K. Wilson et O. P. Borjesson, "On Channel Estimation in OFDM Systems," *IEEE Vehicular Technology Conference*, vol. 2, pp. 815–819, 1995.

[5] J. Rinne et M. Renfors, "Pilot Spacing in OFDM Systems on Practical Channels," *IEEE Transactions on Consumer Electronics*, vol. 42, pp. 959–962, 1996.

[6] S. U. H. Qureshi, "Adaptive Equalization," *Proceedings of the IEEE*, vol. 73, pp. 1349–1387, 1985.

[7] A. Jantsch, S. Kumar, and A. Hemani, "The rugby meta-model," vol. 5, p. 19, 2000.

[8] C. Flemming, "A tutorial on convolutional coding with viterbi decoding," *Online*, vol. http ://home.netcom.com/ ?chip.f/viterbi/ tutorial.html, 2006.

[9] D. Voulenteers, "Modulation and Coding Techniques in Wireless Communications," 2011.

[10] "Multiplexing and channel coding (FDD)," *TS 25.212, 3GPP Std.*

[11] G. D. Forney, "The viterbi algorithm," vol. 2, p. 11, 1973.

[12] I. Land and B. H. Fleury, "Digital modulation," *lecture notes*, 2007.

[13] J. A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication," vol. 5, p. 13, 1971.

[14] J. Hagenauer et P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," vol. 1, p. 7, 1989.

[15] W. E. Ryan, "A turbo code tutorial."