الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي والـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـتكنولوجيا
FACULTE DE TECHNOLOGIE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de MASTER

Domaine : Sciences et Technologies.
Filière : Télécommunications.
Option : Réseaux et Télécommunications.

Par :
Rezzoug Hadjer
Tidjani Affaf

THEME

Comparaison entre différentes techniques de permutation sur les turbo-codes

Soutenu publiquement devant le jury composé de :

Mme. Mesri mokhtaria	*M.C.*	*Président*
Mr. Regueb mourad	*M.A.*	*Examinateur*
Mr. Chellali safouane	*M.C.*	*Rapporteur*

Année Universitaire 2019/2020

ملخص

يقدم هذا البحث تقييمًا لأداء مختلف انواع المبدلات التي تعتبر عنصرا مهما في تراكيب الشفرات التوربينية من ناحية الأداء، هناك عدة أنواع من المتداخلات. لتحقيق أداء ممتاز، من الضروري تصميم تشذير قوي لكود توربو.

أظهرت نتائج المحاكاة أن تشذير DRP وDGI أفضل مقارنة بالتداخلات الأخرى (ARP وRI وRandom)، بسبب تصميمها الممتاز والمعقد.

بالإضافة إلى نوع المشذر، هناك العديد من العوامل التي تؤثر على أداء الشفرة التوربينية، مثل المولد متعدد الحدود، وعدد تكرارات فك التشفير، وحجم الكتلة، وكلما زادت هذه المعلمات، كان الأداء أفضل، ولكن هذا يزيد من وقت المحاكاة.

الكلمات الرئيسية: التشذير، الشفرة التوربينية، التسلسل الموازي، التبادل، الشفرات التلافيفية، الشفرة المنهجية العودية، وحدة فك الترميز التكرارية، خوارزمية MAP، معدل الخطأ الثنائي، نسبة الإشارة إلى الضوضاء.

Résumé

Cette recherche présente une évaluation des performances de différents types des techniques des entrelaceurs, qui sont considérées comme un élément important dans la structure de turbo codes. Il existe plusieurs types d'entrelaceurs, et pour obtenir une excellente performance, il est nécessaire de concevoir un entrelacement robuste de code turbo.

Les résultats de la simulation ont montré que l'entrelacement Dithered Prime relative interleaver (DRP) et dithered golden interleaver (DGI) sont meilleurs par rapport aux autres entrelaceurs (almost regular permutation (ARP), Entrelaceur régulier (RI) et Aléatoire), grâce à leurs excellentes conceptions sophistiquées. De plus, avec le type de l'entrelaceur, plusieurs facteurs affectent les performances du turbo code, telles que le polynôme générateur, le nombre d'itérations de décodage, la taille du block, en augmentant ces paramètres, les performances paraissent meilleures, mais cela augmente le temps des simulations.

Mots clés : L'entrelacement, turbocode, concaténation parallèle, permutation, codes convolutifs, code systématique récursif, décodeur itératif, algorithme MAP, taux d'erreurs binaire, rapport signal sur bruit.

Abstract

In this research, a performance evaluation of different types of interleavers is presented, which are treated as an important element in the structure of turbo codes. There are several types of interleavers. To obtain excellent performance, it is necessary to design a robust interleaving of turbo code.

The simulation results showed that l'entrelacement Dithered Prime relative interleaver (DRP) et dithered golden interleaver (DGI) interleaving are better compared to the other interleavers

(ARP almost regular permutation (ARP), regular interleaving (RI) and Random), due to its excellent and sophisticated design.

In addition to the type of interleaver, several factors affect the performance of the turbo code, such as the generator polynomial, the number of decoding iterations, the block size, by increasing these parameters, the performances seem better, but that increases the time of the simulations.

Keywords: Interleaving, turbocode, parallel concatenation, permutation, convolutional codes, recursive systematic code, iterative decoder, MAP algorithm, bit error rate, signal to noise ratio.

Dédicace

je dédie mon travail :

A mon père Tidjani Mohamed Alamine

Pour son amour , sa patience, ses conseils et ses considérables sacrifices

pour me parvenir à ce niveau

A ma chère mère Malika

Pour son grand amour, ses sacrifices et toute l'affection qu'elle m'a

toujours offerte

A mes sœurs manal , wissal, et farah et mon frère ahmed

Pour son soutien moral et son affection

À l'âme de mon cher grand-père Bederina Elhadj Lehbib qui a

toujours été fière de mes succès

À ma tante Bederina Zouhra qui s'est tenue à mes côtés et m'a

encouragée à tout moment

A toute ma famille, À mon amie Ferhat Soulaf, et tous mes ami(e)s

de l'option Télécommunications

Tidjani Affaf

Dédicace

Je dédie mon travail :

A mon père Tahar

Pour son amour, son soutien, sa patience, ses conseils et ses considérables sacrifices pour me parvenir à ce niveau

A ma mère Mahboubi Malika

Pour son grand amour, ses sacrifices et toute l'affection qu'elle m'a toujours offerte

À mon deuxième père Noureddine Benjemaa

Pour son soutien, ses conseils et sa grande confiance pour atteindre ce niveau

Pour mes frères et ma sœur

Pour leur soutien moral et leur amour

Pour toute ma famille, Ma Chère tante Affaf

Merci pour tout l'amour que vous m'avez toujours entouré, et ces quelques mots témoignent des sentiments aimables et chaleureux que j'aime pour vous. Qu'ils trouvent ici l'expression de mes meilleurs vœux de bonheur et de santé.et tous mes ami(e)s de l'option Télécommunications, tous ceux qui J'aime qui m'aiment

Rezzoug Hadjer

Remerciements

Mes remerciements à Dieu pour l'aide qu'il nous a apportée, le pouvoir qu'il nous a donné pour faire ce travail. Je tiens à remercier chaleureusement et sincèrement le superviseur, Mr. S. Chellali, pour sa disponibilité, sa générosité intellectuelle, son dynamisme, son objectivité, ses observations pertinentes et la confiance qu'il nous a témoignée, pour réaliser cette mémoire. Nous remercions également tous les jurés d'avoir accepté de revoir notre travail. Et aussi nos chers professeurs qui nous ont enseigné et qui nous ont soutenus par leurs compétences durant nos études. Et enfin nous remercient nos familles et amis qui par leurs prières et leurs encouragements, ont su surmonter tous les obstacles.

Table des matières

Chapitre 3 : les résultats et simulations

Liste des figures

Chapitre 1 :

Chapitre 2 :

Chapitre 3 :

Liste des tableaux

Chapitre 2 :

Chapitre 3 :

Liste des abréviations et symboles :

Les symboles :

A : est un décalage constant,

a : est l'amplitude de l'évanouissement

α_k : La probabilité d'état qu'étant donné le treillis

β_k : La probabilité d'état qu'étant donné le treillis

γ_k : La probabilité d'état qu'étant donné le treillis

σ^2 : est la variance du bruit,

C : la capacité de canal

C : le nombre de colonne de l'entrelaceur.

D : la partie entière de la lecture de la nouvelle matrice qui obtenue de ce fait colonne par colonne

d_{min} :distance minimal de Hamming

$d\ (i)$: est un vecteur « tramé » de longueur C

d : bloc de données

d/n : la fiabilité du code

E_b : est l'énergie transmise par bit,

i : est l'indice séquentiel des positions binaires après entrelacement,

K : La taille de l'entrelaceur

m : les blocs d'informations précédents

n : est la taille du code

P : entier premier par rapport à K

P_0 : est un entier relativement premier à K

Π : permutation de longueur K

R : débit de transmission

r =k/n : le rendement ou taux de codage du code

S : valeur de décalage

SNR : le rapport signal / bruit

u_k : le symbole d'entrée

Y : le bloc reçu

Les abréviations :

ARP: Almost Regular Permutation

AWGN: Aditive White Gaussian. Noise

BCJR : selon les chercheurs Bahl, Cocke, Jelinek et Raviv

BER : Bit Error Rate (taux d'erreurs binaires).

BPSK: Binary Phase Shift Keying

DGI: Dithered Golden Interleaver

DRP: Dithered Prime relative interleaver

DVB-S: Digital Video Broadcasting - Satellite

FEC: Forward error correcting

LDPC: Low Density Parity Check

LLR : Log-Likelihood Ratios

LLR_c : la valeur du canal

LLRa : l'information a priori

LLRe : l'information extrinsèque

MAP: Algorithme Maximum A Posteriori

MIT: Massachusetts Institute of Technology

RSC : récursive systématique convolutionnel

RP : relatif premier

QPP : quadratic permutation polynomial

SNR : Signal to Noise Ratio

SOVA : Soft-Output Viterbi Algorithme

TEB : Le taux d’erreurs par bit

UMTS : *Universal Mobile Telecommunications Syste*

Introduction Générale

Aujourd'hui, les télécommunications numériques sont omniprésentes dans nos vies. Au cours des dernières années, la communication sans fil a considérablement progressé dans notre société, depuis la transmission par satellite, la télévision et la radio jusqu'aux téléphones mobiles universels. La demande croissante d'applications et de services multimédia, comme les jeux interactifs ou la TV à la demande, appellent à introduire une nouvelle génération de systèmes de communication et de radiodiffusion. De tels systèmes se devront d'assurer des transmissions fiables tout en ayant un impact limité sur les ressources spectrales, notamment lorsque des faibles taux d'erreurs sont ciblés. Il y a donc une nécessité de « sécuriser » la transmission : c'est le rôle des codes correcteurs d'erreurs. Ce mémoire contient une description sur les codes de correction d'erreur et des turbo codes avec discussions sur les différents algorithmes de turbo décodage utilisés pour augmenter les performances du décodeur. Les codes de correction d'erreur peuvent détecter les erreurs causées par le bruit et les interférences, et recréer les données d'origine sans erreurs dans les systèmes de communication numériques. Les codes de correction d'erreur sont également appelés codes de correction d'erreur directe ou codes de canal. Il détecte les erreurs causées lors de la transmission du message de l'émetteur au récepteur par le bruit externe. Dans les codes correcteurs d'erreurs, l'introduction de bits de redondance dans un message nous permet de détecter et de corriger les erreurs survenues lors de la transmission. Dans les codes de bloc, un message est divisé en blocs de longueur fixe et les bits de message sont codés et décodés séparément pour chaque bloc. Les codes convolutifs peuvent produire un flux continu de bits avec des bits redondants ajoutés entrelacés entre les bits de données. Les codeurs et décodeurs de codes en bloc et convolutionnels sont mis en œuvre avec une complexité raisonnable car ils sont très structurés. Le résultat de la mise en œuvre pratique de ces codes est inférieur aux limites de codage aléatoires prédites par Shannon en raison de la structure des codes par blocs et convolutionnels. Un turbo code (TC) est la concaténation parallèle de deux ou plusieurs codes composants qui sont des sous-classes de codes de convolution. Les turbo codes sont également structurés mais ils permettent des algorithmes de codage et de décodage pratiques et leur structure sur le canal est construite de manière aléatoire par l'entrelaceur. Par conséquent, les performances (BER) sur les bits et les codes des turbo codes sont plus proches de la limite de Shannon que les performances des codes en bloc et convolutionnels. La clé des performances du turbo codes est l'utilisation d'un algorithme de décodage itératif, qui augmente la complexité de calcul et la latence, mais améliore les performances d'erreur de bit.

Les turbo codes augmentent les performances avec une longueur de contrainte accrue par rapport aux codes convolutifs. Les turbo codes ont des applications limitées en puissance et en interférences. Les performances BER des turbo codes dépendent du nombre d'erreurs binaires dans la séquence reçue. L'entrelaceur aide à rationaliser les modèles d'erreur pour décoder correctement. Il augmente la d_{min} du turbo codes. La perforation est une technique pour augmenter le taux de code. Il s'agit d'un processus dans lequel des bits spécifiques du flux de sortie sont supprimés selon un modèle fixe donné par la matrice de perforation. Il est utilisé pour augmenter l'efficacité de transmission. L'objectifs de la fonction d'entrelacement des turbo codes consiste à « casser » ou éviter les paquets d'erreurs dans les bits encodés, à savoir un groupe consécutif de bits encodés et erronés par le canal. L'entrelaceur lit les bits de la trame d'information d et les écrit dans un ordre permuté dans la trame entrelacée. Ainsi, un paquet d'erreurs présent à l'entrée du décodeur du premier code élémentaire est dispersé dans le deuxième. L'entrelaceur est donc un composant essentiel des turbo codes qui joue un double rôle. Premièrement, il limite la d_{min} de Hamming atteignable par le code et affecte par conséquent sa performance asymptotique. Deuxièmement, due à ses propriétés de dispersion, l'entrelaceur impacte aussi la corrélation affectant l'information extrinsèque échangée par les décodeurs élémentaires lors du processus de décodage itératif. Ce mémoire est organisé en 3 chapitres :

Le premier chapitre est consacré à quelques généralités relatives au système de communication numérique et les codes de correction des erreurs, plus précisément on s'intéresse à l'étude des turbo codes, il contient également le principe de décodage itératif en particulier et les algorithmes de décodages.

Le deuxième chapitre présente les techniques d'entrelacement, ses paramètres, ses différents types d'entrelaceurs en donnant une comparaison et en décrivant son avantage d'utilisation.

Le dernier chapitre contient le détail de la simulation, en analysant les performances des turbo codes selon différents paramètres, à savoir le taux de codage 1/2, la trame d'entrée qui progresse de 400 bits et 1024 bits avec 5 itérations dans chaque simulation, il contient également les résultats obtenus et leurs interprétations. Et en fin on clôture notre mémoire avec une conclusion générale.

Chapitre 1 : La révolution des codes

1.1. Introduction

Dans les systèmes de communications numériques, les informations transmises sont perturbées par le bruit du canal pour cela, un système de codage est utilisé pour détecter, ou même corriger les erreurs provoquées par le bruit. C'est ce qu'on appelle un code correcteur d'erreurs. Dans le secteur des communications numériques, il existe plusieurs codes correcteurs d'erreurs, avec des niveaux de complexité variables. En règle générale, répéter plusieurs fois le même message en code binaire est un pari relativement sûr, mais il est extrêmement coûteux en termes de bande passante et de consommation d'énergie.

Le code le plus puissant est le code Aléatoire qui utilisé par Shannon pour établir la loi du théorème de codage de canal, Mais le problème : pour le code aléatoire pas de possibilité de décodage : Pour chaque bit à décoder : considérer 2^K cas ! (Où **K** : nombre de bits de chaque bloc d'information). La solution pour le codage de canal est de construire un code très proche du code aléatoire, de manière à obtenir un $\mathbf{d_{min}}$ **(distance de Hamming)** très grand, mais décodable, ainsi le turbo code est apparu.

Les turbo codes sont un moyen beaucoup plus développé permettant d'intégrer la redondance des informations, c'est une technique d'encodage sûre et efficace dans la plupart des technologies de communication, basé sur le principe proche du code aléatoire mais en même temps capable de décoder. Les turbo codes sont largement utilisés dans les systèmes actuels (candidat pour de futures normes) de télécommunication sans fil, tels que les normes de téléphonie mobile 3G et 4G (HSPA, LTE), la norme de réseau métropolitain sans fil IEEE 802.16 (WiMAX) et de nombreux autres standards. Classiquement, l'algorithme Logarithmique BCJR (LogMAP) est employé pour le décodage itératif de ces codes convolutifs.

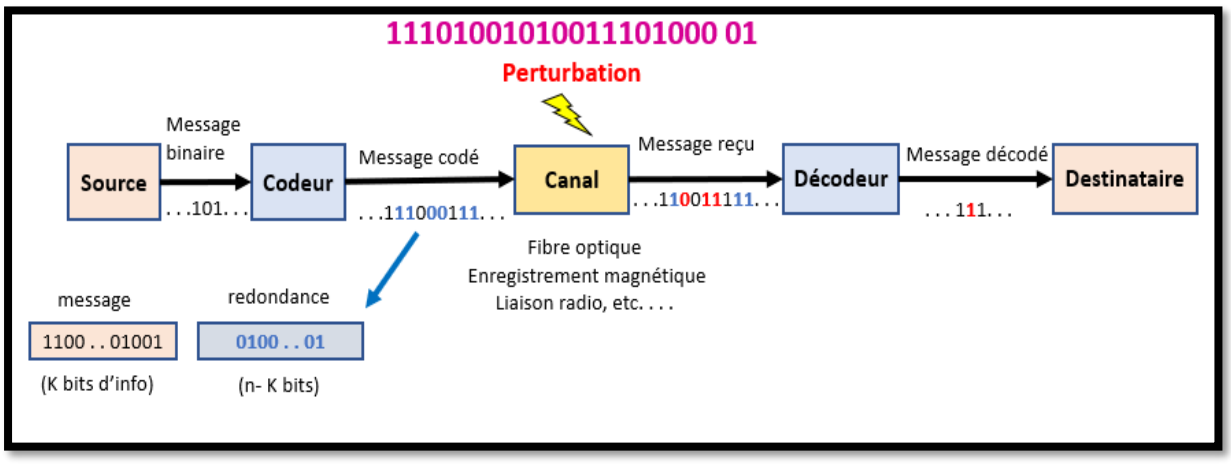


Figure 1. 1:Système de communication numérique

1.2. Théorie de l'information – Théorème de Shannon

Dans tout système de communication numérique, les informations sont transmises d'une source à un récepteur à travers un canal de transmission, ces informations sont représentées par une séquence de symboles dans le système binaire. Ce genre de transmission des données permet l'utilisation de nombreuses techniques de manipulation de l'information Parmi celles-ci, la compression de l'information, au cryptage de cette dernière et aussi aux codes correcteurs d'erreurs.

Les perturbations intervenant sur le canal de transmission induisent des erreurs de transmission que le codage de canal s'efforce de combattre afin d'assurer un taux d'erreur minimal. Le codage de canal est basé sur l'insertion parmi les éléments d'information d'éléments supplémentaires (la redondance) qui suivent une loi connue. On définit un canal de transmission comme un système physique permettant la transmission d'une information entre deux points distants, **(BER)** d'un message est le rapport du nombre de bits erronés par le nombre de bits du message. [1]

En **1948**, Shannon énonce dans **« A Mathematical Theory of Information »** le théorème fondamental de la théorie de l'information :

Cette théorie stipule que de faibles probabilités d'erreur en décodage peuvent être atteintes à tout débit de transmission **R** inférieur à la capacité de canal **C (R < C)** en utilisant de longues longueurs de bloc (block lengths).

Shannon a prouvé que les performances moyennes d'un ensemble de codes choisis au hasard entraînent une diminution significative de la probabilité d'une erreur en décodage avec l'augmentation de la longueur des blocs.

Shannon a démontré que pour un canal à bruit blanc et gaussien, on peut exprimer sa capacité comme suit :

$$C = W\,log_2\left(1 + \frac{S}{N}\right)(bits/symbols) \qquad (1.\,1)$$

Avec :

C : Capacité du canal en bits par seconde (bits/s), W : Largeur de bande du signal (Hz). S : Puissance moyenne reçue (Watt), N : Puissance moyenne du bruit reçu (Watt) dans la bande W

En d'autres termes, on peut obtenir des transmissions aussi fiables que l'on veut, en utilisant des codes de taux plus petits que la capacité du canal. Cependant, ce théorème n'indique pas le moyen de construire de tels codes.

1.3. Les codes de correcteurs d'erreur

Les codes correcteurs d'erreur peuvent être classifiés en deux catégories, à savoir les codes blocs et les codes convolutionnels ; les codes en bloc traitent chaque bloc d'information indépendamment les uns des autres ; par contre dans les codes convolutifs la sortie d'un codeur convolutif dépend d'un symbole courant à coder ainsi que du symbole précédent et du résultat de codage du symbole précédent.

1.3.1. Code en bloc

Dans la théorie du codage, des codes de blocs sont une grande et importante famille de codes correcteurs d'erreurs qui codent les donnes en blocs. Il existe un grand nombre d'exemples de codes de bloc, parmi ces codes : **Reed–Solomon codes**, **Hamming codes**, **Hadamard codes**, **Expander codes**, **Golay codes**, et **Reed–Muller codes**.[2]

Lorsque l'émetteur souhaite envoyer un très long flux de données à l'aide d'un code de block, il divise le flux en morceaux d'une certaine taille fixe, Chacune de ces pièces est appelée bloc dans le contexte des codes de bloc, Chaque bloc est encodé individuellement dans un mot codé, ensuite l'émetteur transmet tous les blocs au récepteur, qui peut à son tour, utiliser un mécanisme de décodage pour récupérer les messages d'origine à partir des blocs reçus éventuellement corrompus. Les performances et le succès de la transmission globale dépendent des paramètres du canal et du code de bloc. Le codage en blocs consiste à associer à un bloc de données **d** de **k** symboles issus de la source d'information un bloc **c**, appelé mot de code, **d** en symbole s avec $\mathbf{n \geq k}$. et n est la taille du code, la différence $\mathbf{(n-k)}$ représente la quantité de redondance introduite par le code. La connaissance de la règle de codage en réception permet de détecter et de corriger, sous certaines conditions, des erreurs. Le rapport $\mathbf{R= k/n}$ est appelé rendement ou taux de codage du code. Plus il est proche de **0** et plus la redondance introduite, et donc le temps de transmission, sont importants. Et le rapport **d/n** renseigne donc sur la fiabilité du code.

On voit qu'il est impossible d'avoir en même temps une fiabilité et un taux élevés. Toute la difficulté de la construction des codes est de trouver le bon compromis.

1.3.2. Codes convolutifs (ou convolutionnels) :

Les codes convolutionnels font partie des codes correcteurs d'erreur et sont parmi les plus utilisés dans les communications sans fil en raison de leur propriété de continuité dans la transmission de l'information. L'approche du codage convolutif consiste à calculer la redondance à partir du bloc d'information courant de taille k et des m blocs précédents. Les n bits en sorties ont calculés par une combinaison linéaire ente les k bits en entrée et les m blocs précédents. Le rendement du code est $\boldsymbol{r = k/n}$ et sa longueur de contrainte K est le nombre maximum de bits associés à une sortie qui peuvent être affectés par un bit quelconque à l'entrée.

Exemple de code convolutif :

Nous considérons un codeur convolutif binaire de taux 1/2 (c'est-à-dire 2 sorties pour 1 entrée), systématique (c'est-à-dire que la première sortie est égale à l'entrée) et d'ordre m (c'est-à-dire que le codeur possède 2^m états, typiquement m=3).[1]

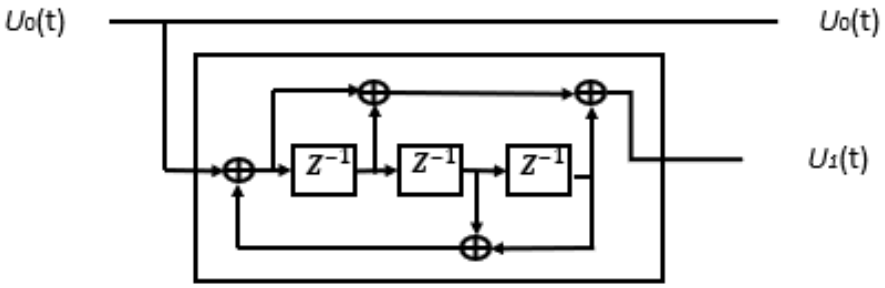


Figure 1. 2:Codeur convolutif systématique de taux 1/2 et d'ordre 3

L'état de la machine au temps t sera un entier, noté e(t), compris entre 0 et 2^m-1. Pour décrire cette machine il faut deux fonctions, l'une pour mettre à jour l'état et l'autre pour calculer le symbole de redondance [1]

$$e(t+1) = E\big(e(t), u0(t)\big) \qquad (1.2)$$

$$u1(t) = S\big(e(t), u0(t)\big) \qquad (1.3)$$

Avec $e(t), e(t+1) \in [0, 2m[$ et $u0(t), u1(t) \in \{0,1\}$. À chaque unité de temps la machine prend en argument un symbole binaire et en produit un second, le taux de transmission (Rapport entre la taille de l'entrée et celle de la sortie) vaut donc $1/2$. Les $u0(t)$ seront appelés symboles d'information et les $u1(t)$ symboles de redondance.

- Il existe deux types des code convolutionnelles, code convolutionnelles systématiques et code convolutionnelles non systématiques, codes récursifs sont toujours systématiques et, à l'inverse, les codes non récurrents ne sont pas systématiques.

1.3.2.1. Représentation graphique des codes convolutionnels

La représentation graphique des codes convolutionnels est nécessaire pour les algorithmes de décodage. Les plus usuelles sont la représentation sous forme d'un treillis, d'un diagramme d'état, ou encore d'un arbre.

a) Le diagramme d'état

Le diagramme d'état est un graphe dont les sommets sont les états et les flèches orientées, qui relient ces derniers, représentent les différentes transitions possibles entre eux. Le diagramme d'état nous permet de déterminer la fonction de transfert du code. Elle est utilisée pour évaluer la distribution des poids de Hamming des différents chemins qui divergent de l'état initial (état nul) et qui convergent plus loin. [1]

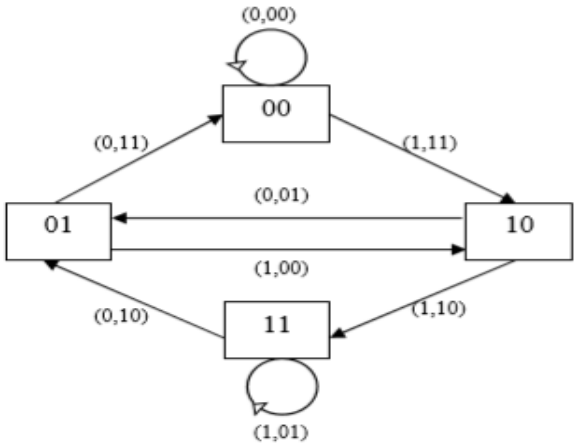


Figure 1. 3:Diagramme d'état de l'encodeur convolutionnel de taux de codage R=1/2

L'inconvénient de diagramme d'état est que ne nous permet pas de percevoir l'évolution dynamique du codeur dans le temps.

b) Arbre

L'arbre d'encodage représente l'ensemble des séquences d'entrées-sorties possibles. Chaque nœud de l'arbre représente un état de l'encodeur. Les nœuds sont reliés par des branches. Deux branches seulement sont issues de chaque nœud : la branche dirigée vers le haut indique que le bit d'information inséré est 0 et la branche dirige vers le bas indique que le bit d'information inséré est 1. Une trajectoire spécifiée dans l'arbre est tracée de gauche à droite conformément à la séquence d'entrée. [1]

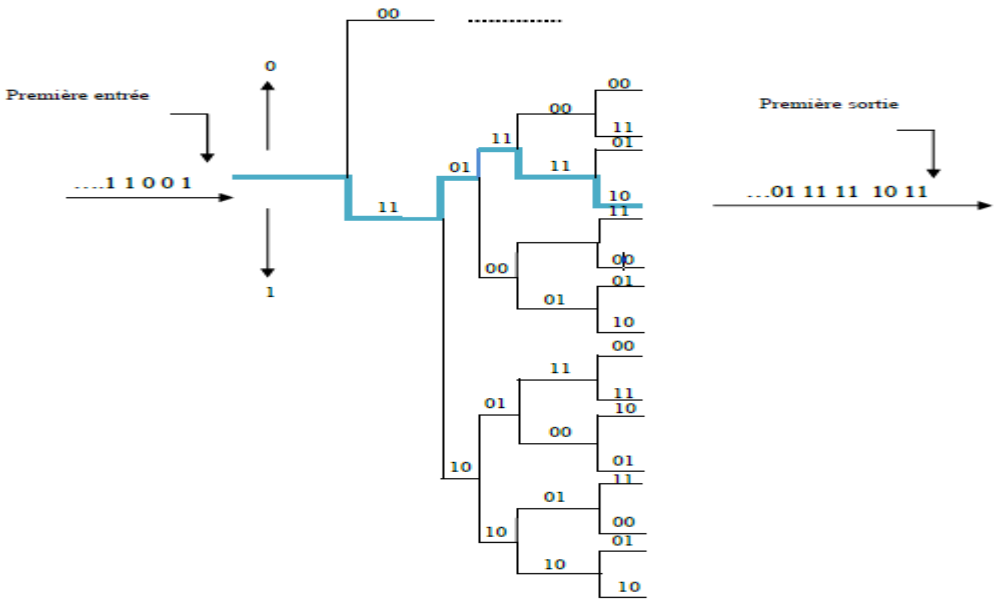


Figure 1. 4:Code arbre pour l'encodeur convolutif

c) Treillis

La représentation en arbre est très redondante. En effet, pour représenter le fonctionnement d'un codeur convolutif, on peut employer un diagramme de transitions d'état, développé en fonction du temps. Ce temps discrétisé est appelé niveau de nœud. L'état est décrit par la mémoire du codeur : pour un codeur de mémoire de taille, 2 états sont accessibles.

On inscrit sur chaque branche associée à une transition possible entre deux états, les bits sortant du codeur quand cette transition a lieu. Les transitions entre les différents états sont représentées par des branches qui relient les nœuds. On trouve deux branches qui sortent d'un nœud et deux autres qui y convergent. [1]

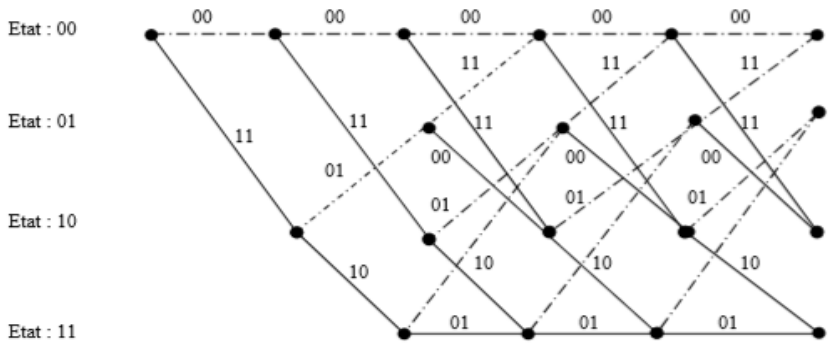


Figure 1. 5:Treillis de l'encodeur convolutif

1.3.2.2. Décodage convolutif :

La contrainte principale du décodage convolutif réside dans le fait que le mot de code est très long, ce qui a tendance à compliquer le circuit décodeur. Les algorithmes de décodage les plus répandus sont :

a) Décodage séquentiel

Il a été proposé par Wonzencraft et Reiffen en 1961 puis améliorée n 1963 par Fano. La stratégie adoptée par Fano consiste en la recherche du chemin le plus probable à l'intérieur du diagramme en arbre, en examinant une transition de mémoire à la fois. Ce type de recherche appartient au genre de recherche depth-first. La recherche du chemin le plus prometteur est conduite à l'aide d'une fonction de métrique PM adaptée à ce genre de recherche. [3]

$$PM^{(i)} = \sum_{np=1}^{B}\left(\mu_{np}^{(i)} - c\right) \qquad (1.4)$$

À cause du nombre d'itérations nécessaires pour le décodage d'un bloc est considéré comme une variable aléatoire, il se peut que le décodage soit défectueux ou la cause d'effacements, Quant à la complexité du décodage, elle est indépendante de la longueur de contrainte du code. Ce type de décodage est approprié au cas des canaux sans mémoire.

b) Algorithme de Viterbi

Proposé en 1967 par A. J. Viterbi, il est basé sur le principe du maximum de vraisemblance, Cet algorithme est une méthode optimale de décodage pour les codes convolutifs, ses performances dépendant de la qualité du canal de transmission utilisé. Par contre la complexité des systèmes de décodage augmentant exponentiellement avec la longueur de contrainte du code utilisé restreint leur emploi aux applications où le code a une petite longueur de contrainte. [3]

c) Décodage par seuil (logique majoritaire)

Proposé par Masseye en 1963. Il diffère des deux autres méthodes de décodages précédant par le fait que la décision finale prise pour un bloc donné est seulement basé sur la longueur de contrainte du bloc en cours de décodage plutôt que sur l'utilisation de toute la séquence reçue,

Ce qui conduit à des préfinances de décodage inferieures aux deux autres méthodes. Ses points forts résident dans la simplicité de son implantation et sa rapidité de décodage. Le décodage de syndrome est approprié pour des transmissions sur des canaux a mémoire où les erreurs apparaissent par paquet. Il est plutôt utilisé dans des applications qui ne demandent pas de grands gains de décodage. On le retrouve dans des applications telles que la téléphonie ou la radio HF. Il est aussi utilisé dans les systèmes de télécommunication à haut débit. [3]

1.3.3. Turbo code

Une classe de codes correcteurs d'erreur décodables itérativement et est un mélange entre codes convolutionnels et codes de bloc, C'est efficace pour un transfert de données fiable sur des liens à bande passante limitée ou à latence limité. Le codage turbo est basé sur deux idées fondamentales : la conception d'un code avec des propriétés aléatoires et la conception d'un décodeur qui exploite le décodage itératif facile à mettre en œuvre. Les turbocodes ont une performance exceptionnellement bonne, en particulier avec des taux d'erreur de bit modérés et pour des longueurs de blocs importantes.

1.3.3.1. Définition

Les Turbo-Codes (TCs), grâce à leurs excellentes capacités de correction d'erreurs, sont considérés comme une partie fondamentale des normes de télécommunication actuelles telles que LTE, HSPA. Le turbo-codeur est la concaténation parallèle de deux codeurs convolutifs séparés par un entrelaceur qui permute les séquences de bits afin de casser les relations de voisinage entre eux, ce qui permet d'accroitre les capacités de correction d'erreurs. Le décodeur est constitué par la concaténation en série de deux décodeurs convolutifs qui sont séparés par deux entrelaceurs. Ces deux décodeurs partagent leurs informations de manière itérative afin de décoder le message reçu. Ce procédé itératif, introduit au niveau du décodage, permet d'obtenir des gains de performances considérables et de se rapprocher à la limite de Shannon

1.3.3.2. Apparition des turbo codes

En 1948, Shannon énonce dans « A Mathematical Theory of Information » le théorème fondamental de la théorie de l'information, où il montrait l'existence d'une borne théorique au-delà de laquelle les communications sont entachées d'erreurs, mais il ne donna pas d'exemple de code réalisable qui permettait d'approcher cette limite.

Puis, en 1960, Robert G. Gallager mit au point les codes LDPC (Low Density Parity Check) dans sa thèse au MIT qui permettaient d'atteindre cette limite mais à l'époque, leur réalisation pratique était infaisable et ils tombèrent aux oubliettes.

Au début des années 1990, Claude Berrou mit au point les turbo codes qui furent les premiers codes approchant la limite de Shannon pour lesquels on réussit une implémentation matérielle, Et qui breveta. [2]

La clé de cette trouvaille aurait résidé dans l'utilisation de codes convolutifs récursifs et dans la mise au point de l'information extrinsèque, résultat de la soustraction de l'information a priori (donnée d'entrée) à l'information a posteriori (donnée de sortie).

C'est, entre autres, cette soustraction permettant d'éviter les phénomènes de propagation d'erreur dans la boucle « turbo » qui fut difficile à mettre au point.

En 1993, Berrou, Glavieux et Thitimajshima ont développé une nouvelle approche aux codes correcteurs d'erreur : les turbo codes. Il s'agit en fait de la concaténation parallèle de deux codeurs convolutionnels récursifs et systématiques (RSC) de faible longueur de contrainte à travers un entrelaceur. Le décodage est effectué de façon itérative par deux décodeurs concaténés en série. Les auteurs cités ci-dessus ont montré que nous nous sommes plus qu'à 0.7 dB de la borne de Shannon. Etant donné qu'avec les codes correcteurs classiques, nous en étions à 2.2 dB, ceci veut dire que l'avènement des turbo codes apportait un gain de codage additionnel de 1.5 dB. [1]

Les performances des turbo codes sont fonctions, non seulement des codes convolutionnels utilisés, mais aussi de l'algorithme de décodage et des entrelaceurs

1.3.3.3. Le Principe de fonctionnement de turbo code

Le principe du Turbo-code fut introduire une redondance dans le message afin de le rendre moins sensible aux bruits et perturbations subies lors de la transmission.

Cette méthode est basée sur la transmission du message initial en trois exemplaires. La première copie est l'information brute non codée. Le second est modifié en codant chaque bit d'information à l'aide d'un algorithme partagé par le codeur et le décodeur. Enfin, une autre version du message est également encodée, mais après modification en l'insérant dans l'entrelaceur qui prend une séquence de symboles à l'entrée et qui produit une séquence de ces mêmes symboles dans un ordre différent. Ces trois versions sont ensuite décodées et comparées afin de retrouver le message d'origine.

La concaténation de deux codes (par exemple deux codes convolutifs) est un moyen simple d'obtenir des distances élevées. Cependant les performances à faible rapport signal à bruit sont dégradées du fait de la répartition de l'énergie de la redondance entre les différents codes constituants. En effet, dans un schéma classique de récepteur ou des décodeurs sont concatènes, l'exploitation de l'information n'est pas optimale. Plus généralement, un d´décodeur composé de sous-décodeurs optimaux ne forment pas un système optimal. Dans le cas d'une concaténation d'un code interne et externe, le décodeur externe ne bénéficie que de l'information contenue dans les symboles de redondance qui lui sont associés.

Le second décodeur bénéficie quant à lui des symboles de redondance et du travail du d´décodeur externe qui le précède. Cette dissymétrie suggère de réinjecter une information

issue du deuxième d´décodeur dans le premier d´décodeur. Cette réinjection de la sortie vers l'entrée est analogue au principe du moteur turbo. Ce principe a été proposé par C. Berrou, A

Glavieux et P. Thitimajshima, en 1993, et a été rendu possible grâce aux travaux de G. Battail, J. Hagenauer et P. Hoeher sur le décodage `a sorties pondérées. Le principe turbo, initialement introduit pour le codage canal, a été ensuite étendu à l'ensemble de la chaîne de réception. Ainsi, on parle de turbo synchronisation, turbo estimation de canal, turbo égalisation et plus généralement de récepteurs itératifs. [4]

Turbo Code généralement dépend de deux composants principaux :

Encodeur : il produit un mot de code avec des propriétés similaires au hasard.

Décodeur : il utilise des valeurs de soft-out et un décodage itératif.

1.3.3.4. Turbo Encoder

La structure générale utilisée dans les turbos encodeurs est la concaténation de deux codeurs convolutifs : le premier encode les bits d'informations **X** dans l'ordre naturel pour générer les bits de parité $\mathbf{p}^{(1)}$, tandis que le second encode les bits d'informations **_(X)** dans l'ordre entrelacé (le message original est introduit dans l'entrelaceur), pour générer les bits de parités $\mathbf{p}^{(2)}$ [5]

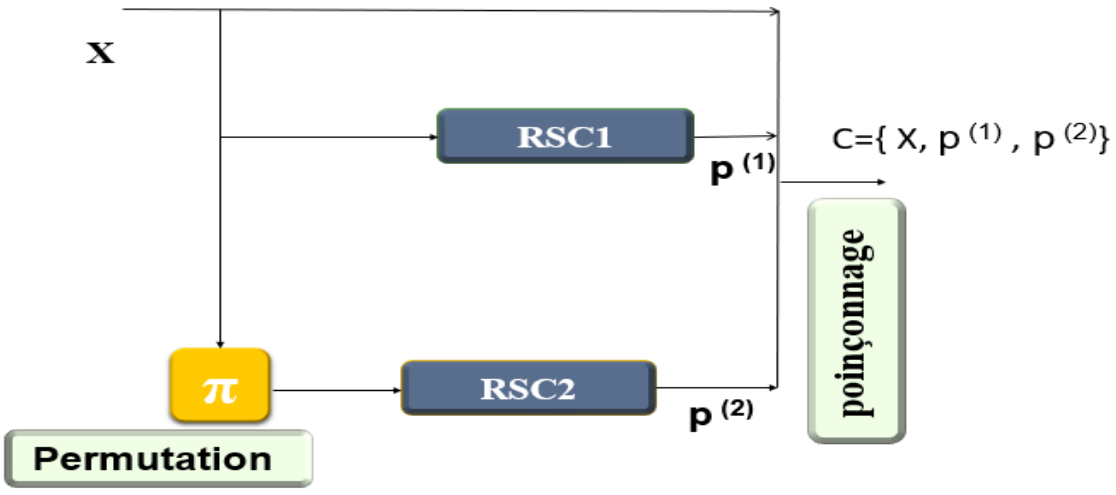


Figure 1. 6 : Schéma de principe d'un Turbo-code, une concaténation parallèle de deux codeurs et un entrelaceur

Donc le turbo codeur possède deux étages. Le premier étage correspond à la transmission de l'information d'entrée, c'est-à-dire que les bits sont transmis sans aucun changement. Ce que nous appelons le deuxième étage correspond à la génération des symboles de parité permettant la correction des erreurs. Cet étage produit deux symboles de parité pour chaque symbole

d'information transmis. Ensuite, pour donner un taux de codage global de la moitié, la moitié des bits de sortie des deux codeurs doivent être perforés. L'arrangement qui est souvent privilégié, et celui que nous avons utilisé dans notre travail, est de transmettre tous les bits systématiques du premier encodeur RSC, et la moitié des bits de parité de chaque encodeur. Notez que les bits systématiques sont rarement perforés, car cela dégrade les performances du code de manière plus spectaculaire que la perforation des bits de parité. [4]

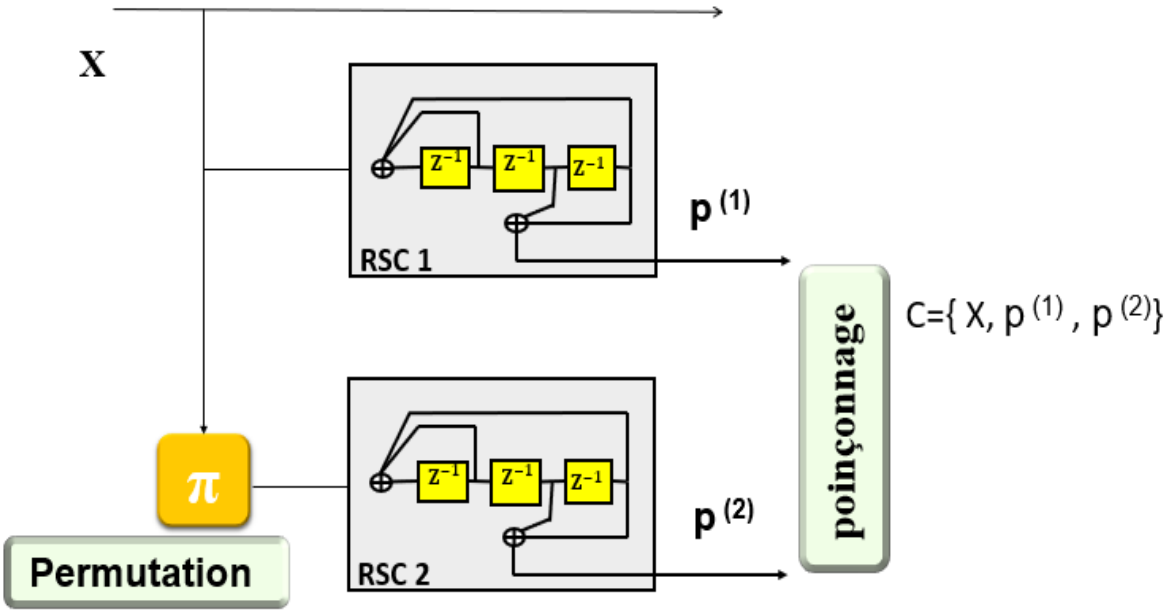


Figure 1. 7:Un turbo-code binaire à mémoire ν = 3 utilisant des codeurs CSR élémentaires identiques.

Dans **La figure 1.7** un turbo codeur, dans sa version la plus classique. Le message binaire d'entrée, de longueur *k*, est codé, dans son ordre naturel et dans un ordre permuté, par deux codeurs RSC appelés *C*1 et *C*2, qui peuvent être terminés ou non. Dans cet exemple, les deux codeurs élémentaires sont identiques (polynômes générateurs 15 pour la récursivité et 13 pour la construction de la redondance) mais ce n'est pas une nécessité. Le rendement de codage naturel, sans poinçonnage, est 1/3. Pour obtenir des rendements plus élevés, un poinçonnage des symboles de redondance *Y*1 et *Y*2 est effectué. Un autre moyen de disposer de rendements plus élevés est d'adopter des codes *m*-binaires.

La fonction de permutation (Π) portant sur un message de taille finie *k*, le turbo-code est par construction un code en bloc. Pour le distinguer toutefois des codes algébriques concaténés décodés à la manière « turbo », comme les codes produits et que l'on a appelés plus tard *turbo-codes en bloc*, ce schéma de turbo-codage est dit *convolutif* ou encore, plus techniquement, *PCCC* (*Parallel Concatenated Convolutional Code*).

a. *m* est le nombre de bits dans les symboles appliqués au turbo-codeur.

Les applications connues à ce jour considèrent des symboles binaires (m = 1) ou double-binaires (m = 2).

b. Chacun des deux codeurs élémentaires $C1$ et $C2$ est caractérisé par

– sa mémoire de code $\mathbf{Z^{-1}}$

– ses polynômes générateurs de récursivité et de redondance

– son rendement

Les valeurs de $\mathbf{Z^{-1}}$ sont en pratique inférieures ou égales à 4. Les polynômes générateurs sont généralement ceux que l'on utilise pour les codes convolutifs classiques et qui ont fait l'objet d'une littérature abondante dans les années 1980-90.

c. La manière dont on effectue la permutation est importante lorsque le taux d'erreurs binaires cible est inférieur à 10^{-5} environ. Au-dessus de cette valeur, la performance est peu sensible à la permutation, à condition bien sûr que celle-ci respecte au moins le principe de dispersion (cela peut être par exemple une permutation régulière). Pour un taux d'erreurs cible faible ou très faible, la performance est dictée par la distance minimale du code et celle-ci est très dépendante de la permutation Π.

d. Le motif de poinçonnage doit être le plus régulier possible, à l'image de ce qui se pratique pour les codes convolutifs classiques. Toutefois, il peut être avantageux d'avoir un motif de poinçonnage faiblement irrégulier quand on recherche de très faibles taux d'erreurs et lorsque la période de poinçonnage est un diviseur de la période du polynôme générateur de récursivité ou de parité.

1.3.3.5. Poinçonnage

L'intérêt du poinçonnage est de supprimer certains bits de la séquence codée afin d'augmenter le rendement.

Le poinçonnage s'effectue classiquement sur les symboles de redondance. Il peut être envisagé de plutôt poinçonner les symboles d'information, pour augmenter la distance minimale du code. Cela se fait au détriment du seuil de convergence du turbo décodeur. De ce point de vue, en effet, poinçonner des données partagées par les deux décodeurs est plus pénalisant que poinçonner des données qui ne sont utiles qu'à l'un des décodeurs. Ce qui sera à considérer de près dans la construction d'un turbo-code et dans son décodage, ce sont les séquences *RTZ*, dont les poids de sortie limitent la distance minimale du code et en fixent les performances asymptotiques. Dans la suite, il sera admis que les motifs d'erreurs qui ne sont pas *RTZ* ne contribuent pas à la DMH du turbo-code et n'auront donc pas à être recensés.[6]

1.3.3.6. Turbo décoder

Le turbo décodeur reçoit les valeurs d'entrée $Y^{(x)}$, $Y^{(1)}$, $Y^{(2)}$ du canal (resp. Pour x, p (1), p (2)). Une itération complète du turbo décodeur est effectuée en deux demi-itérations. Premièrement, le décodeur Code 1 reçoit des valeurs de canal pour le bit de message $Y^{(x)}$, le premier bit de parité $Y^{(1)}$ et la valeur extrinsèque désentrelacée du décodeur Code 2 pour générer une valeur extrinsèque. Ensuite, pendant la seconde moitié d'itération, le décodeur Code 2 crée une valeur extrinsèque à partir des bits de message entrelacés, un deuxième bit de parité $Y^{(2)}$ et une valeur extrinsèque entrelacée à partir du décodeur Code 1. La décision finale concernant les bits de message est prise sur la base des valeurs extrinsèques des deux décodeurs et des valeurs de canal pour les bits de message, après un nombre fixe d'itérations de décodage. [7]

Le décodage de chaque code convolutif peut être réalisé par l'utilisation de l'algorithme MAP-**BCJR** en turbo décodage, les deux décodeurs partagent l'information sur les bits du message reçu. Chaque décodeur fournit l'information extrinsèque à l'autre décodeur afin d'estimer les bits de mot de code transmis par le codeur. Ensuite, dans le turbo décodage, l'information extrinsèque est mise à jour et partagée par les décodeurs durant plusieurs itérations. L'information sortant d'un SISO est ensuite entrelacée (Π) ou desentrelacée (Π-1) pour servir d'information entrante à un autre décodeur, L'algorithme MAP- BCJR est donc appliqué plusieurs fois par chaque décodeur pour obtenir les bits du mot de code les plus probables. Le schéma du turbo décodeur est illustré dans la Figure 1.8. Le décodeur reçoit des échantillons bruités à La Sortie du canal $Y^{(x)}$, $Y^{(1)}$, $Y^{(2)}$. [5]

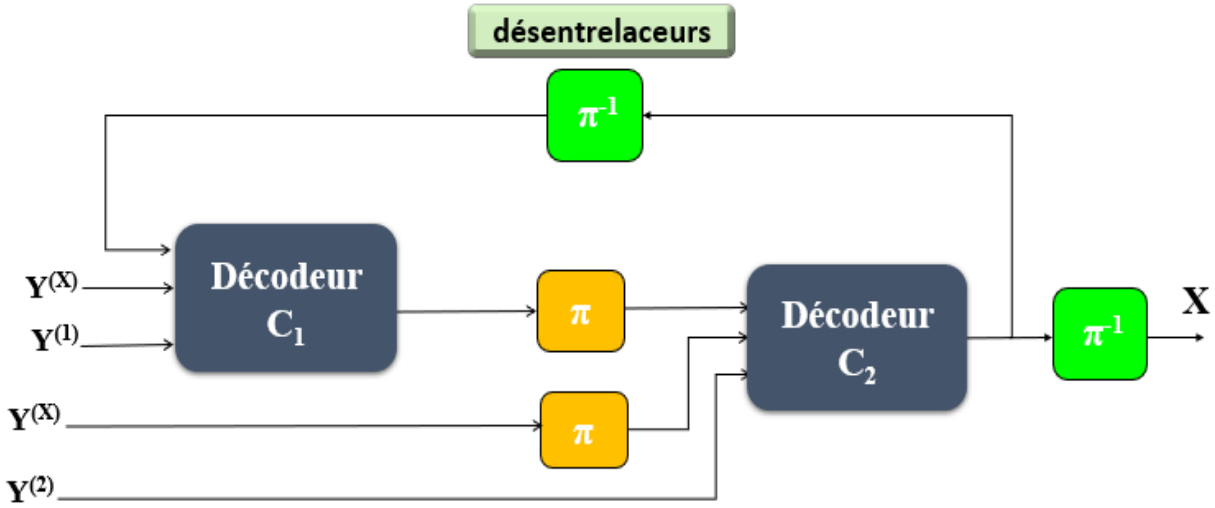


Figure 1. 8: Schéma du décodeur Turbo

Le décodeur de **la figure 1.8** fonctionne de manière itérative, et dans la première itération, le premier décodeur de composant prend uniquement des valeurs de sortie de canal, et produit une soft output comme son estimation des bits de données.

Le soft output du premier codeur est ensuite utilisé comme information supplémentaire pour le deuxième décodeur, qui utilise cette information avec les sorties de canal pour calculer son estimation des bits de données. Maintenant, la deuxième itération peut commencer et le premier décodeur décode à nouveau les sorties de canal, mais maintenant avec des informations supplémentaires sur la valeur des bits d'entrée fournis par la sortie du deuxième décodeur lors de la première itération. Ces informations supplémentaires permettent au premier décodeur d'obtenir un ensemble plus précis de soft output, qui sont ensuite utilisées par le deuxième décodeur comme informations a priori. Ce cycle se répète et à chaque itération, jusqu'à l'obtention des performances de décodage désirées. Cependant, pour des raisons de complexité, seulement 8 itérations sont généralement utilisées. [8]

- **Remarque :**

En raison de l'entrelacement utilisé au niveau du codeur, il faut veiller à bien entrelacer et désentrelacer les LLR qui sont utilisés pour représenter les valeurs douces des bits, comme le montre **la figure 1.8**. En raison de la nature itérative du décodage, il faut veiller à ne pas réutiliser les mêmes informations plusieurs fois à chaque étape de décodage.

D'autres décodeurs non itératifs ont été proposés qui permettent un décodage optimal des turbo codes. Cependant, l'amélioration des performances par rapport aux décodeurs itératifs s'est avérée être seulement d'environ 0,35 dB, et ils sont extrêmement complexes. Par conséquent, le schéma itératif illustré à **la figure 1.8** est couramment utilisé.

Le décodeur MAP fournit une décision douce sur chaque bit décodé. Dans le cas des turbo codes, le codeur fournit un bit systématique qui représente le bit de donnée et deux bits de parités issus des deux codeurs. Donc, nous nous attendons à avoir un accès direct de l'état du canal dû à l'information systématique et une information supplémentaire indirecte qui est le résultat de l'envoi des bits de parités. Le décodeur MAP utilise un ensemble d'algorithmes. [8]

1.3.3.6.1. Algorithme de la vraisemblance (Log-likelihood Ratios)

Le décodeur Turbo nécessite deux décodeurs MAP identiques et fonctionne de manière itérative ; où les probabilités extrinsèques du domaine de consignation produisent $LLR\ (P_{out}(\mu_k|Y))$ d'un décodeur MAP seront transmises à un autre en tant que probabilités a priori du mot de code reçu.

Chaque décodeur réalisant une règle de décision M A P calcule le rapport de vraisemblance logarithmique pour chaque bit d'information dans le bloc Y reçu dans ce décodage itératif de Turbo codes. Le rapport de vraisemblance logarithmique pour le symbole d'entrée μ_k est défini comme ;

$$RLL\ (\mu_k) = ln\frac{p(\mu_k = 1|Y)}{p(\mu_k = 0|Y)} \qquad (1.5)$$

Il peut également s'écrire comme suit :

$$RLL\ (\mu_k) = LLR_c(Y_k^s|\mu_k) + LLR_a(\mu_k) + LLR_e(\mu_k) \qquad (1.6)$$

Où :

$LLR_c(Y_k^s|\mu_k)$:la valeur du canal ; $LLR_a(\mu_k)$: l'information a priori ; $LLR_e(\mu_k)$: l'information extrinsèque.

La valeur de canal $LLR_c(Y_k^s|\mu_k)$ est donnée par ;

$$LLR_c(Y_k^s|\mu_k) = ln\frac{p(Y_k^s|\mu_k = 1)}{p(Y_k^s|\mu_k = 0)} \qquad (1.7)$$

On prend les valeurs -1 et 1 pour le bit u_k, plutôt que 1 et 0. Cette définition des deux valeurs d'une variable binaire ne fait aucune différence conceptuelle, mais elle simplifie légèrement les mathématiques dans les dérivations qui suivent.

$$RLL\ (\mu_k) = ln\frac{p(\mu_k = +1|Y)}{p(\mu_k = -1|Y)} \qquad (1.8)$$

Le signe du LLR L (u_k) d'un bit u_k indiquera si le bit est plus susceptible d'être $+1$ ou -1, et l'amplitude du LLR donne une indication de la probabilité que le signe du LLR donne la valeur correcte de u_k.

Lorsque le $LLR\ L\ (u_k) \approx 0$, nous avons $P\ (u_k = +1) \approx P\ (u_k = -1) \approx 0{,}5$, et nous ne pouvons pas être certains de la valeur de u_k. Inversement, lorsque $L\ (u_k) >> 0$, nous avons $P\ (u_k = +1) >> P\ (u_k = -1)$ et nous pouvons être presque certains que $u_k = +1$

Étant donné le LLR L (u_k), il est possible de calculer la probabilité que $u_k = 1$ ou $u_k = -1$ comme suit. En se rappelant que $P\ (uk = -1) = 1 - P\ (uk = 1)$, et en prenant l'exposant des deux côtés dans l'équation 1.7, nous pouvons écrire :

$$e^{L(u_k)} = \frac{p(u_k=+1)}{1-p(u_k=+1)} \qquad (1.\ 9)$$

Donc

$$p(u_k = +1) = \frac{e^{L(u_k)}}{1 + e^{L(u_k)}}$$

$$= \frac{1}{1+e^{-L(u_k)}} \qquad (1.\ 10)$$

De même,

$$p(u_k = -1) = \frac{1}{1 + e^{+L(u_k)}}$$

$$= \frac{e^{-L(u_k)}}{1+e^{-L(u_k)}} \qquad (1.\ 11)$$

Et donc nous pouvons écrire :

$$p(u_k = \pm 1) = \left(\frac{e^{-\frac{L(u_k)}{2}}}{1+e^{-L(u_k)}}\right) . e^{\pm L(u_k)/2} \qquad (1.12)$$

Notez que le terme entre crochets dans cette équation ne dépend pas de si nous sommes intéressés par la probabilité que u_k = + 1 ou -1, et donc il peut être traité comme une constante dans certaines applications où nous utilisons cette équation dans la dérivation de l'algorithme MAP.

Outre les LLR L (u_k) basés sur les probabilités inconditionnelles $P\ (u_k\ =\ \pm\ 1)$, nous nous intéressons également aux $LLRs$ basés sur les probabilités conditionnelles. Par exemple, dans la théorie du codage de canal, nous nous intéressons à la probabilité que $u_k\ =\ \pm\ 1$ basé, ou conditionné, sur une séquence reçue $\underline{y}$, et donc nous pouvons utiliser le conditionnel $LLR\ \ L\ (u_k \mid \underline{y})$, qui est défini comme :

$$L\left(\mu_k \middle| \underline{y}\right) . 0. = ln \frac{p\left(\mu_k = 1 \middle| \underline{y}\right)}{p\left(\mu_k = -1 \middle| \underline{y}\right)} \qquad (1.\ 13)$$

Les probabilités conditionnelles $p\left(\mu_k = \pm 1 \middle| \underline{y}\right)$ sont connues comme les probabilités a posteriori du bit décodé μ_k, et ce sont ces probabilités a posteriori que nos décodeurs soft-in soft-out décrit dans les sections suivantes tentent de trouver.

Mis à part le LLR L $(\mu_k|y)$ conditionnel basé sur les probabilités a posteriori $P\,(\mu_k = \pm 1 \mid y)$, nous utiliserons également des LLRs conditionnels basés sur la probabilité que la sortie du filtre adapté du récepteur soit y_k étant donné que le bit transmis correspondant x_k était soit + 1 ou − 1. Ce LLR conditionnel s'écrit $L\,(y_k \mid x_k)$ et est défini comme :

$$L\,(y_k|x_k) = ln\frac{p(y_k \mid x_k=+1)}{p(y_k \mid x_k=-1)} \qquad (1.\ 14)$$

Si nous supposons que le bit transmis $x_k = \pm 1$ a été envoyé sur un canal gaussien ou à évanouissement en utilisant la modulation BPSK, alors nous pouvons écrire pour la probabilité de sortie du filtre adapté y_k que :

$$p(y_k \mid x_k = +1) = \frac{1}{\sigma\sqrt{2\pi}} exp\,(-\frac{E_b}{2\sigma^2}(y_k - a)^2) \qquad (1.\ 15)$$

Où :

E_b : est l'énergie transmise par bit,

σ^2 : est la variance du bruit,

a : est l'amplitude de l'évanouissement (nous avons $a = 1$ pour les canaux AWGN sans évanouissement). De même, nous avons

$$p(y_k \mid x_k = -1) = \frac{1}{\sigma\sqrt{2\pi}} exp\,(-\frac{E_b}{2\sigma^2}(y_k + a)^2) \qquad (1.\ 16)$$

Par conséquent, lorsque nous utilisons BPSK sur un canal gaussien (éventuellement décoloré), nous pouvons réécrire l'équation 1.13 sous la forme

$$L(y_k|x_k) = ln\left(\frac{p(y_k \mid x_k = +1)}{p(y_k \mid x_k = -1)}\right)$$

$$= ln\left(\frac{exp\,(-(\frac{E_b}{2\sigma^2})(y_k - a)^2)}{exp\,(-(\frac{E_b}{2\sigma^2})(y_k + a)^2)}\right)$$

$$= (-\frac{E_b}{2\sigma^2}(y_k - a)^2) - (-\frac{E_b}{2\sigma^2}(y_k + a)^2)$$

$$= \frac{E_b}{2\sigma^2} 4a.y_k$$

$$= Lc\ y_k \qquad (1.\ 17)$$

Où

$$Lc = 4a\frac{E_b}{2\sigma^2} \qquad (1.18)$$

Est défini comme la valeur de fiabilité du canal et ne dépend que du rapport signal / bruit (SNR) et de l'amplitude d'évanouissement du canal. Par conséquent, pour BPSK sur un canal gaussien (éventuellement décoloré), le $LLR\ L\ (y_k|\ x_k)$ conditionnel, qui est appelé la sortie douce du canal, est simplement la sortie du filtre adapté y_k multipliée par la valeur de fiabilité du canal Lc.

Après avoir introduit LLR, nous allons maintenant décrire le fonctionnement de l'algorithme MAP, qui est l'un des possibles décodeurs soft-in soft-out composants qui peuvent être utilisés dans un décodeur turbo itératif. [8]

1.3.3.6.2. Algorithme Maximum a posteriori (MAP)

En **1974** algorithme Maximum A Posteriori **(MAP)**, a été proposé par Bahl, Cocke, Jelinek et Raviv, **(BCJR)**, du nom de ses inventeurs, cet algorithme sert pour estimer les probabilités a posteriori des états et les transitions d'une source de Markov observée lorsqu'elle est soumise à un bruit sans mémoire.

Ils ont montré comment l'algorithme pouvait être utilisé pour décoder à la fois des codes de blocs et des codes convolutionnels. Lorsqu'il est utilisé pour décoder des codes convolutifs, l'algorithme est optimal en minimisant directement le BER décodé, contrairement à l'algorithme de Viterbi, qui minimise la probabilité qu'un chemin incorrect à travers le treillis soit sélectionné par le décodeur. Ainsi, l'algorithme de Viterbi peut être considéré comme minimisant le nombre de groupes de bits associés à ces chemins de treillis, plutôt que le nombre réel de bits, qui sont décodés de manière incorrecte. Néanmoins, comme indiqué par Bahl et al., dans la plupart des applications, les performances des deux algorithmes seront presque identiques. Cependant, l'algorithme MAP examine tous les chemins possibles à travers le treillis du décodeur convolutif et, par conséquent, semblait initialement être d'une complexité insurmontable pour une application dans la plupart des systèmes. Il n'était donc pas largement utilisé avant la découverte des turbo codes.

Cependant, l'algorithme MAP fournit non seulement la séquence de bits estimée, mais également les probabilités pour chaque bit qui a été décodé correctement. [8]

a) Théorème de Bayes :

La règle de Bayes est utilisé pour donner la probabilité conjointe d'**a** et **b**, $P\ (a \wedge b)$, en termes de probabilité conditionnelle d'un b donné comme :

$$P\ (a|b) = \frac{P\ (a \wedge b)}{P(b)} = \frac{P\ (b|a)P(a)}{P(b)}$$

$$P\ (a \wedge b) = P\ (a|b)\ .P(b). \tag{1. 19}$$

Probabilité : $P\ (a|b)$, la probabilité des données (observations) dans certains modèles

Probabilité a priori (Prior, probabilité) : $P(a)$, une probabilité a priori que l'observateur a sur le système ou modèle avant l'observation.

Probabilités postérieures (la probabilité postérieure) : $P\ (a|b)$, après l'incident (après les observations effectuées) probabilité lors de l'incident à un modèle spécifique

Une conséquence utile de la règle de Bayes est que :

$$P\ (\{a \wedge b\}|c) = P\ (a|\{b \wedge c\}).P\ (b|c)\ . \tag{1. 20}$$

Qui peut être dérivée de l'équation 1.18 en considérant $x \equiv a \wedge b$ et $y \equiv b \wedge c$ comme suit. À partir de l'équation 1.18, nous pouvons écrire :

$$P\ (\{a \wedge b\}|c) \equiv P\ (x|c) = \frac{P(x \wedge c)}{P(c)} \tag{1. 21}$$

$$= \frac{P(x \wedge b \wedge c)}{P(c)} \equiv \frac{P(a \wedge y)}{P(c)}$$

$$= \frac{P\ (a|y).P(y)}{p(c)} \equiv P(a|\{a \wedge c\}).\frac{P(b \wedge c)}{P(c)}$$

$$= P(a|\{b \wedge c\}).p\ (b|c)\ . \tag{1. 22}$$

L'algorithme MAP donne, pour chaque bit décodé u_k, la probabilité que ce bit soit +1 ou -1, étant donné la séquence de symboles reçue $\underline{y}$. Cela équivaut à trouver le LLR L a posteriori $(u_k \mid \underline{y})$, où :

$$L\left(u_k\middle|\underline{y}\right) = ln\left(\frac{p\left(u_k = +1\middle|\underline{y}\right)}{p\left(u_k = -1\middle|\underline{y}\right)}\right) \tag{1. 23}$$

La règle de Bayes nous permet de réécrire cette équation comme suit :

$$L\left(u_k\middle|\underline{y}\right) = ln\left(\frac{p(u_k=+1\wedge\underline{y})}{p(u_k=-1\wedge\underline{y})}\right) \quad (1.\ 24)$$

La figure 9 montrant les transitions possibles pour le code RSC pour K = 3, g0=7, g1=5.
Pour ce code K = 3 il y a quatre états de codeur, et puisque nous considérons un code binaire, dans chaque état de codeur, deux transitions d'état sont possibles, en fonction de la valeur de ce bit. L'une de ces transitions est associée au bit d'entrée de -1 représenté par une ligne continue, tandis que l'autre transition correspond au bit d'entrée de +1 représenté par une ligne discontinue.

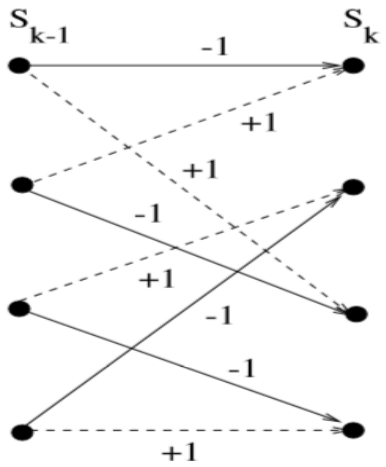


Figure 1. 9:Transitions possibles dans le code composant K = 3RSC.

Si l'état précédent $Sk-1$ et l'état actuel S_k sont connus, alors la valeur du bit d'entrée μ_k, qui a provoqué la transition entre ces deux états, sera connue. Par conséquent, la probabilité que $uk = +1$ soit égale à la probabilité que la transition de l'état précédent $Sk - 1$ à l'état actuel Sk soit l'une des quatre transitions possibles qui peuvent se produire lorsque $uk = +1$ (c'est-à-dire les transitions représentées par lignes brisées). Cet ensemble de transitions est mutuellement exclusif (c'est-à-dire qu'une seule d'entre elles aurait pu se produire au niveau du codeur), et donc la probabilité que l'une d'entre elles se produise est égale à la somme de leurs probabilités individuelles. [8]

$$L\left(u_k\middle|\underline{y}\right) = ln\left(\frac{\sum(s,\acute{s})\Longrightarrow_{u_k=+1} p(S_{k-1}=\acute{s}\wedge S_k=s\wedge\underline{y})}{\sum(s,\acute{s})\Longrightarrow_{u_k=-1} p(S_{k-1}=\acute{s}\wedge S_k=s\wedge\underline{y})}\right) \quad (1.\ 25)$$

Où $(s', s) \Rightarrow u_k = +1$ est l'ensemble des transitions de l'état précédent $S_{k-1} = s'$ à l'état actuel S_k = s qui peuvent se produire si le bit d'entrée $u_k = +1$, et de même pour $(s', s) \Rightarrow uk = -1$. Par souci de concision, nous écrirons $P\ (S_{k-1} = s' \wedge S_k = s \wedge y)$ comme $P\ (s' \wedge s \wedge y)$. Nous écrirons $P\ (S_{k-1} = \acute{s} \wedge s_k = s \wedge \underline{y})$ comme $P\ (\acute{s} \wedge s \wedge \underline{y})$.

La séquence reçue y peut être divisée en trois sections :

Le mot de code reçu associé à la transition actuelle $\underline{y_k}$,

La séquence reçue avant la transition actuelle $\underline{y_j} < k$

La séquence reçue après la transition actuelle $\underline{y_j} > k$.

Cette division est montrée dans la figure suivants :

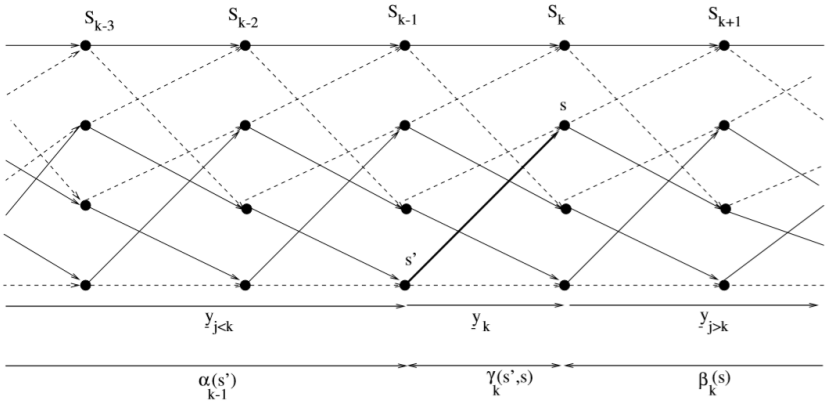


Figure 1. 10:Treillis de décodeur MAP pour code RSC K = 3

Par utilisé la règle de bayes on peut écrire la probabilité $p(\acute{s} \wedge s \wedge \underline{y})$ comme suit :

$$p\left(\acute{s} \wedge s \wedge \underline{y}\right) = p\left(\underline{y}_{j>k} \middle| s\right) . p\left(\acute{s} \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right)$$

$$= p\left(\underline{y}_{j>k} \middle| s\right) . p\left(\left\{\underline{y}_k \wedge s\right\} \middle| \left\{\acute{s} \wedge \underline{y}_{j<k}\right\}\right) . p\left(\acute{s} \wedge \underline{y}_{j<k}\right)$$

$$= p\left(\underline{y}_{j>k} \middle| s\right) . p\left(\left\{\underline{y}_k \wedge s\right\} \middle| \acute{s}\right) . p\left(\acute{s} \wedge \underline{y}_{j<k}\right)$$

$$= \beta_k(s) . \gamma_k(\acute{s}, s) . \alpha_{k-1}(\acute{s}), \qquad (1.\ 26)$$

Où :

La probabilité que les treillis créent s 'à l'instant $k - 1$ et la séquence de canaux reçue jusqu'à ce point est $\underline{y}_{j<k}$:

$$\alpha_{k-1}(\acute{s}) = P\left(S_{k-1} = \acute{s} \wedge \underline{y}_{j<k}\right) \qquad (1.\ 27)$$

La probabilité qu'étant donné le treillis : dans l'état s au moment k, la future séquence de canaux reçus sera $\underline{y}_{j>k}$:

$$\beta_k(s) = P\left(\underline{y}_{j<k} \middle| S_k = s\right) \quad (1.28)$$

La probabilité qu'étant donné le treillis : dans l'état s au moment k-1, la future séquence de canaux reçus sera $\underline{y}_k$:

$$\gamma_k(\acute{s}, s) = \left(\left\{\underline{y}_k \wedge S_k = s\right\} \middle| S_{k-1} = \acute{s}\right) \quad (1.29)$$

MAP permet de trouve $\alpha_k(s)$ et $\beta_k(s)$ pour tous les états s tout au long du treillis, c'est-à-dire pour $k = 0, 1, \ldots, K-1$ et $\gamma_k(\acute{s}, s)$ pour toutes les transitions possibles de état $S_{k-1} = \acute{s}$ à l'état $S_k = s$, encore pour $k = 0, 1, \ldots, K-1$.Ces valeurs sont ensuite utilisées pour trouver les probabilités $P(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})$, qui sont ensuite utilisées dans l'équation 1.24 pour donner les $LLR\ L(uk \mid y)$ pour chaque bit u_k. [8]

b) Calcul des probabilités d'état :

Les probabilités d'état peuvent être exprimées récursivement. Ainsi,

$$\alpha_k(s) = \sum_{\acute{s}} \alpha_{k-1}(\acute{s})\gamma_k(s, \acute{s}) \qquad et \qquad \beta_{k-1}(s) = \sum_{\acute{s}} \beta_k(\acute{s})\gamma_k(s, \acute{s}) \quad (1.30)$$

En supposant que le treillis commence et termine dans l'état s_0, les conditions initiales pour ces deux probabilités sont :

$$\alpha_0(s) = \begin{cases} 1 & si\ \ s = s_0 \\ 0 & sinon \end{cases} \qquad et \qquad \beta_k(s) = \begin{cases} 1 & si\ \ s = s_0 \\ 0 & sinon \end{cases}$$

Calcul des probabilités de transition Tout d'abord, il est à noter que si la transition dans une tranche de treillis entre l'état s et l'état s' n'existe pas, alors $\gamma_k(s, s') = 0$. Ensuite, en partant de la définition de γ_k et en utilisant la relation de Bayes, nous obtenons :

$$\gamma_k(s, \acute{s}) = p(m_k)p(y_k|c_k) \quad (1.31)$$

c) Forward récursion :

Calculer $\boldsymbol{\alpha_k(s)}$:

$$\boldsymbol{\alpha_{k-1}(\acute{s}) = p\left(s_{k-1} = \acute{s} \wedge \underline{y}_{j<k}\right)}$$

$$\boldsymbol{\alpha_k(s) = p\left(S_k = s \wedge \underline{y}_{j<k+1}\right)}$$

$$\boldsymbol{= P\left(s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right)}$$

$$\boldsymbol{= \sum_{all\ \acute{s}} P\left(s \wedge \acute{s} \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right)} \quad (1.32)$$

En utilisant la règle de Bayes et supposant que le canal est sans mémoire, nous pouvons procéder comme suit :

$$\alpha_k(s) = \sum_{all\ \acute{s}} \boldsymbol{P}\left(s \wedge \acute{s} \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right)$$

$$= \sum_{all\ \acute{s}} \boldsymbol{P}\left(\left\{s \wedge \underline{y}_k\right\} \middle| \left\{\acute{s} \wedge \underline{y}_{j<k}\right\}\right) . \boldsymbol{P}\left(\acute{s} \wedge \underline{y}_{j<k}\right)$$

$$= \sum_{all\ \acute{s}} \boldsymbol{P}\left(\left\{s \wedge \underline{y}_k\right\} \middle| \acute{s}\right) . \boldsymbol{P}\left(\acute{s} \wedge \underline{y}_{j<k}\right)$$

$$= \sum_{all\ \acute{s}} \boldsymbol{\gamma_k}(\acute{s}, s) . \boldsymbol{\alpha_{k-1}}(\acute{s}) \qquad (1.\ 33)$$

Calculer $\boldsymbol{\beta_{k-1}}$:

On a :

$$\boldsymbol{\beta_k}(s) = \boldsymbol{p}(\underline{y}_{j>k} / \ s_k = s) \qquad (1.\ 34)$$

$$\boldsymbol{\beta_{k-1}}(\acute{s}) = \boldsymbol{p}(\underline{y}_{j>k} / \acute{s}) \qquad (1.\ 35)$$

En utilisant la règle de Bayes et supposant que le canal est sans mémoire,

$$\beta_{k-1}(\acute{s}) = P\left(\underline{y}_{j>k-1} \middle| \acute{s}\right)$$

$$= \sum_{all\ s} P\left(\underline{y}_{j>k-1} \wedge s \middle| \acute{s}\right)$$

$$= \sum_{all\ s} P\left(\left\{\underline{y}_k \wedge \underline{y}_{j>k} \wedge s\right\} \middle| \acute{s}\right)$$

$$= \sum_{all\ s} P\left(\underline{y}_{j>k} \middle| \left\{\acute{s} \wedge s \wedge \underline{y}_k\right\}\right) . P\left(\left\{\underline{y}_k \wedge s\right\} \middle| \acute{s}\right)$$

$$= \sum_{all\ s} P\left(\underline{y}_{j>k} \middle| s\right) . P\left(\left\{\underline{y}_k \wedge s\right\} \middle| \acute{s}\right)$$

$$= \sum_{all\ s} \beta_k(s) . \gamma_k(s, \acute{s}) \qquad (1.\ 36)$$

Calculer $\gamma_k(s, \acute{s})$:

$$\gamma_k(s, \acute{s}) = p\left[\underline{y}_k \wedge S_k = s\right] / (S_{k-1} = \acute{s})$$

$$\gamma_k(s, \acute{s}) = P\left[\left(s \wedge \underline{y}_k\right) / \acute{s}\right]$$

$$\gamma_k(s, \acute{s}) = P\left[\underline{y}_k / (s \wedge \acute{s})\right] . P(u_k) \qquad (1.\ 37)$$

Où :

$$P(u_k) = \left[\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}}\right].e^{\left(u_k\left(\frac{L(u_k)}{2}\right)\right.} \quad (1.\ 38)$$

A partir l'équation 1.24 et 1.25 on peut écrire :

$$L\left(u_k\middle|\underline{y}\right) = ln\left(\frac{\sum(s,\acute{s}) \underset{u_k=+1}{\Longrightarrow} p(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})}{\sum(s,\acute{s}) \underset{u_k=-1}{\Longrightarrow} p(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})}\right.$$

$$= ln\left[ln\left(\frac{\sum(s,\acute{s}) \underset{u_k=+1}{\Longrightarrow} \beta_k(s).\gamma_k(\acute{s},s).\alpha_{k-1}(\acute{s})}{\sum(s,\acute{s}) \underset{u_k=-1}{\Longrightarrow} \beta_k(s).\gamma_k(\acute{s},s).\alpha_{k-1}(\acute{s})}\right.\right] \quad (1.\ 39)$$

Avec $L\left(u_k\middle|\underline{y}\right)$ est le LLR conditionnel délivré par le décodeur MAP

1.3.3.6.3. L'algorithme Max-Log-MAP

L'algorithme Max-Log-MAP simplifie cela en transférant ces équations dans le domaine logarithmique puis en utilisant l'approximation :

$$ln(\sum_i e^{x_i}) \approx max(x_i), \quad (1.\ 40)$$

Où $max_i\ (x_i)$ signifie la valeur maximale de x_i. Ensuite, avec $A_k\ (s)$, $B_k\ (s)$ et $\Gamma_k\ (s', s)$ définis comme :

$$A_k(s) \triangleq ln\left(\alpha_k(s)\right), \quad (1.\ 41)$$

$$B_k(s) \triangleq ln\left(\beta_k(s)\right), \quad (1.\ 42)$$

$$\Gamma_k(\acute{s}, s) \triangleq ln\left(\gamma_k(\acute{s}, s)\right), \quad (1.\ 43)$$

a) Calcul de A_k :

$$A_k(s) \triangleq ln\left(\alpha_k(s)\right)$$

$$= ln\left(\sum_{all\ \acute{s}} \alpha_{k-1}(\acute{s})\,\gamma_k(\acute{s}, s)\right)$$

$$= ln\left(\sum_{all\ \acute{s}} exp[A_{k-1}(\acute{s}) + \Gamma_k(\acute{s}, s)]\right)$$

$$\approx max_{\acute{s}}(A_{k-1}(\acute{s}) + \Gamma_k(\acute{s}, s). \quad (1.\ 44)$$

b) Calcul de B_{k-1} :

$$B_{k-1}(\acute{s}) \triangleq ln\big(\beta_{k-1}(\acute{s})\big)$$

$$= ln\left(\sum_{all\ s} \beta_k(s)\gamma_k(\acute{s},s)\right)$$

$$= ln\left(\sum_{all\ s} exp[\beta_k(s) + \Gamma_k(\acute{s},s)\ ]\right)$$

$$\approx max_s\big(B_k(s) + \Gamma_k(\acute{s},s)\big) \qquad (1.45)$$

c) Calcul de Γ_k :

$$\Gamma_k(\acute{s},s) \triangleq ln\big(\gamma_k(\acute{s},s)\big)$$

$$= ln\left(C.e^{(u_k L(u_k/2)}\, exp\left[\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^{n} y_{kl}x_{kl}\right]\right)$$

$$= ln\left(C.e^{(u_k L(u_k/2)}\, exp\left[\frac{L_c}{2} 2a \sum_{l=1}^{n} y_{kl}x_{kl}\right]\right)$$

$$= \hat{C} + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl}, \qquad (1.46)$$

Où :

E_b : l'Energie transmise par bit, σ^2 : la variance de bruit, a : l'amplitude d'évanouissement.

$$L_c = 4a\frac{E_b}{2\sigma^2} \qquad (1.47)$$

C : est une constante dans la sommation en numérateur dénominateur.

$\hat{C} = ln(C)$, où $\hat{C}$ est une constante qui ne dépend pas de $u_k$ou bien de mot code transmit $\underline{x}_k$

Enfin, on peut écrire pour l'a posteriori LLRs L (uk | y) que l'algorithme Max-Log-MAP calcule que :

$$L\left(u_k\middle|\underline{y}\right) = ln\left(\frac{\sum_{(\acute{s},s)\Longrightarrow u_k=+1} \alpha_{k-1}(\acute{s}).\gamma_k(\acute{s},s).\beta_k(s)}{\sum_{(\acute{s},s)\Longrightarrow u_k=-1} \alpha_{k-1}(\acute{s}).\gamma_k(\acute{s},s).\beta_k(s)}\right)$$

$$= ln\left(\frac{\sum_{(\acute{s},s)\Longrightarrow u_k=+1} exp(A_{k-1}(\acute{s}) + \Gamma_k(\acute{s},s) + B_k(s)}{\sum_{(\acute{s},s)\Longrightarrow u_k=-1} exp(A_{k-1}(\acute{s}) + \Gamma_k(\acute{s},s) + B_k(s)}\right)$$

$$\approx \max_{(\acute{s},s)\Longrightarrow u_k=+1}\big(A_{k-1}(\acute{s}) + \Gamma_k(\acute{s},s) + B_k(s)\big) - \max_{(\acute{s},s)\Longrightarrow u_k=-1}\big(A_{k-1}(\acute{s}) + \Gamma_k(\acute{s},s) + B_k(s)\big) \quad (1.48)$$

1.3.3.6.4. L'algorithme Max-Log-MAP (correction de l'approximation) :

L'algorithme Max-Log-MAP donne une légère dégradation des performances par rapport à l'algorithme MAP en raison de l'approximation de l'équation 1. 49. Lorsqu'elle est utilisée pour le décodage itératif de turbo codes, cette dégradation a été trouvée par Robertson et al comme entraînant une baisse de performance d'environ 0,35 dB. Cependant, l'approximation de l'équation 1.38 peut être rendue exacte en utilisant le logarithme jacobien :

$$\begin{aligned} ln(e^{x_1} + e^{x_2}) &= max(x_1, x_2) + ln\left(1 + e^{-|x_1 - x_2|}\right) \\ &= max(x_1, x_2) + f_c(|x_1 - x_2|) \\ &= g(x_1, x_2), \end{aligned} \tag{1. 50}$$

Où $f_c\ (x)$ peut être considéré comme un terme de correction. C'est alors la base de l'algorithme Log-MAP proposé par Robertson et al.

Semblable à l'algorithme Max-Log-MAP, les valeurs pour $A_k\ (s)\ ln\ (\alpha_k\ (s))$ et $B_k\ (s)\ ln\ (\beta_k\ (s))$ sont calculées en utilisant une récursion en avant et en arrière. [8]

1.3.3.6.5. L'algorithme SOVA :

L'algorithme Souft out put verturbi (SOVA) est un algorithme de décodeur de composants pour les turbos codes qui contient deux modifications à l'algorithme classique de Viterbi :

Premièrement, les métriques de chemin utilisées sont modifiées pour tenir compte des informations a priori lors de la sélection du chemin maximum likelihood à travers le treillis,

Deuxièmement, l'algorithme est modifié de manière à fournir un soft output sous la forme du $LLR\ L\left(u_k \middle| \underline{y}\right)$ a posteriori pour chaque bit décodé. [8]

Pour réaliser la première modification, Considérons la séquence d'états $\underline{S}_k^s$qui donne les états le long du chemin de survie à l'état $S_k = s$ au stade k du treillis. La probabilité que ce soit le bon chemin à travers le treillis est donnée par :

$$p\left(\underline{S}_k^s \middle| \underline{y}_{j \le k}\right) = \frac{p\left(\underline{S}_k^s \wedge \underline{y}_{j \le k}\right)}{p\left(\underline{y}_{j \le k}\right)} \tag{1. 51}$$

$$M\left(\underline{S}_k^s\right) = \left(\underline{S}_{k-1}^{\acute{s}}\right) + ln\left(\gamma_k(\acute{s}, s)\right) \tag{1. 52}$$

Où :

$M\left(\underline{S}_k^s\right)$ La métrique de chemin pour $\underline{S}_k^s$

$\gamma_k(\acute{s}, s)$: est la probabilité de transition de branche pour le chemin de $S_{k-1} = \acute{s}$ à $S_k = s$

À partir de l'équation 1.52, nous pouvons écrire :

$$ln\left(\gamma_k(\dot{s},s)\right) \triangleq \Gamma_k(\dot{s},s) = \hat{C} + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl} \qquad (1.53)$$

Le terme $\hat{C}$ peut être supprimé car il est constant afin que nous puissions réécrire l'équation $M\left(\underline{S}_k^s\right)$ comme suit :

$$M\left(\underline{S}_k^s\right) = M\left(\underline{S}_{k-1}^{\dot{s}}\right) + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl} \qquad (1.54)$$

Pour la deuxième modification de l'algorithme, Pour donner un changement des soft output, dans un treillis binaire, il y aura deux chemins atteignant l'état $S_k = s$ au stade k dans le treillis, qui sont $\underline{S}_k^s$ et $\hat{\underline{S}}_k^s$, et qui ont les deux métriques de chemins respectivement $M\left(\underline{S}_k^s\right)$ et $M\left(\hat{\underline{S}}_k^s\right)$ qui peut être calculé à partir de l'équation 1.50, On peut définit la différence métrique Δ_k^s comme :

$$\Delta_k^s = M\left(\underline{S}_k^s\right) - M\left(\hat{\underline{S}}_k^s\right) \geq 0 \qquad (1.55)$$

La probabilité que nous ayons prise le bon chemin lorsque nous avons sélectionné le chemin S_k^s comme survivant et le chemin rejeté $\hat{S}_k^s$ est alors :

$$p(décision\ correcte\ \text{à}\ S_k = s) = \frac{e^{M\left(\underline{S}_k^s\right)}}{e^{M\left(\underline{S}_k^s\right)} + e^{M\left(\hat{\underline{S}}_k^s\right)}}$$

$$= \frac{e^{\Delta_k^s}}{1+e^{\Delta_k^s}} \qquad (1.56)$$

Et le LLR que c'est la bonne décision est donnée par :

$$L(décision\ correcte\ \text{à}\ S_k = s) = \ln\left(\frac{p(décision\ correcte\ \text{à}\ S_k = s)}{1 - p(décision\ correcte\ \text{à}\ S_k = s)}\right)$$

$$= \Delta_k^s \qquad (1.57)$$

La valeur de $LLR_s L\left(u_k \middle| \underline{y}\right)$ à posteriori délivrés par le décodeur SOVA :

$$LLR_s L\left(u_k \middle| \underline{y}\right) = |\Delta_k^s|_{\max\left(M\left(\underline{S}_k^s\right)\right)} \qquad \text{Pour } s = 0,1,2,\ldots,2^{k-1}$$

$|\Delta_k^s|_{\max\left(M\left(\underline{S}_k^s\right)\right)}$: est la valeur absolue de Δ_k^spour le chemin $\underline{S}_k^s$pour obtenir $M(\underline{S}_k^s)$maximum dans le treillis à l'instant K

1.3.3.7. Canal de transmission

Le canal de transmission est divisé généralement en trois parties : modulation, canal physique de transmission et démodulation. Nous allons présenter la modulation et les modèles de canaux physiques utilisés dans nos travaux.[6]

1.3.3.7.1. Modulation BPSK (Binary Phase Shift Keying)

BPSK est une modulation très utilisée dans l'étude théorique des communications numériques. Effectivement, elle est caractérisée par une faible probabilité d'erreur par bit, et par son efficacité au niveau de la puissance de transmission par rapport aux autres types de modulation. Par exemple, la modulation BPSK offre un gain de 3 dB par rapport à la modulation binaire de fréquence (BFSK).

En BPSK, les phases opposées de la porteuse (0 et π) sont transmises toutes les T_b secondes en considérant T_b la durée temporelle d'un bit. Ces phases sont aussi représentées par +1 et -1. Ces derniers viennent des expressions cos $(w_p t)$ et cos $(w_p t + \pi)$. Ceci se représente sous forme de constellation à deux points. Chaque bit entrant dans le modulateur BPSK se retrouvera en l'un de ces deux points de cette constellation. Ceci est alors une modulation par phase. [6]

Mathématiquement on peut exprimer un signal BPSK de la façon suivante

$$s_1(t) = \sqrt{\frac{2E_b}{T_b}} cos(2\pi f_c t) \ \ Pour\ le\ "\ 1\ " \qquad (1.58)$$

$$s_0(t) = \sqrt{\frac{2E_b}{T_b}} cos(2\pi f_c t + \pi) = -\sqrt{\frac{2E_b}{T_b}} cos(2\pi f_c t) \ \ Pour\ le\ "\ 0\ " \qquad (1.55)$$

$s_0 t$: Signal BPSK lorsqu'on transmet un 0

$s_1 t$: Signal BPSK lorsqu'on transmet un 1

T_b : Durée de transmission d'un bit

f_b : Fréquence porteuse

E_b : Energie par bit du signal transmis

Le problème du démodulateur BPSK est de récupérer la bonne phase pour démoduler. Ceci constitue la principale difficulté du récepteur BPSK.

1.3.3.7.2. Canal physique de transmission

Dans un système de communication, un canal physique de transmission s'occupe de l'acheminement de l'information d'un expéditeur (source) vers le destinataire), on peut le considérer comme un milieu de propagation. Le canal est un véritable problème pour les transmissions de données. En effet, il est source de bruits de toutes sortes. Il existe des modèles pour identifier ce bruit

A) Canal à bruit additif, blanc et gaussien (AWGN)

Le bruit gaussien blanc additif est un type de canal bruyant dans lequel passe le signal modulé. L'entrée peut être un signal multicanal réel ou complexe ainsi qu'un traitement basé sur la trame. La modification de la période de symbole dans le bloc de canaux AWGN affecte la variance du bruit ajouté / échantillon, ce qui entraîne également une modification du taux d'erreur final. Lorsque vous utilisez l'un des modes de variance avec des entrées complexes, la valeur de variance est également répartie entre les composantes réelle et imaginaire du signal d'entrée. Ce modèle implique que le bruit du canal est une variable aléatoire n qui s'additionne au signal modulé transmis. [6]

Communications numériques de base : cas des canaux parfaits.

Conception de l'émetteur (passage des bits au signal) : modulations numériques.

Conception du récepteur (passage du signal aux bits) : cas des canaux parfaits (AWGN).

Communications numériques avancées : cas des canaux réels.

Etude des transmissions numériques

Conception de l'émetteur et du récepteur dans le cas d'un canal idéal : canal AWGN (Additive White Gaussian Noise)

Intérêt du canal AWGN : établissement des performances de référence

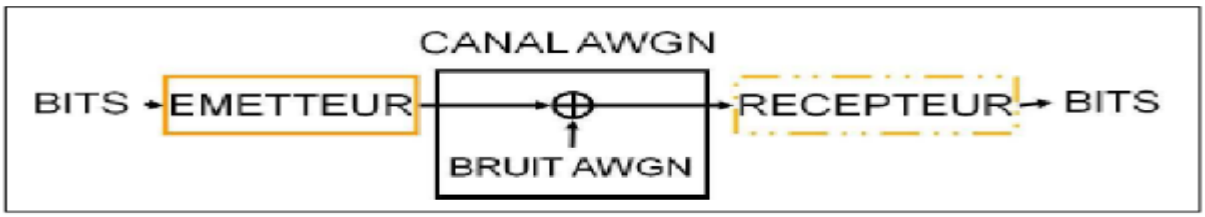


Figure 1. 11: Modèle de canal gaussien

Mathématiquement, la densité de probabilité du bruit blanc gaussien et additif

$$p(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n^2}{2\sigma^2}} \qquad (1.59)$$

La densité spectrale de ce type de bruit est constante, symétrique et uniforme de valeur $\frac{N_0}{2}$.

Si nous considérons la variable du signal modulé x_k, il nous est possible de donner la sortie du canal comme :

$$r_k = x_k + n \qquad (1.60)$$

$$p(r_k|x_k) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(r_k - x_k)^2}{N_0}} \qquad (1.61)$$

Le modèle de bruit blanc gaussien et additif est très simple et très pratique, comme nous l'avons dit, pour les calculs théoriques. Ce type de bruit ne reflète pas bien la réalité dans les communications spatiales. [6]

B) Canal de Rayleigh

Dans les canaux de transmission radio-mobiles, les signaux reçus présentent des évanouissements profonds, provoqués par les phénomènes des chemins multiples et l'effet Doppler. En fait, l'antenne à la réception reçoit non seulement le signal utile mais aussi d'autres signaux appelés chemins multiples, générés par les réflexions sur différents obstacles comme les immeubles, les arbres.... Ces signaux arrivent d'une part en retard, c'est-à-dire avec différentes phases que le signal utile, et d'autre part avec une fréquence décalée proportionnelle à la vitesse du mobile désignée par fréquence Doppler. Par conséquent, en théorie, le modèle du bruit AWGN ne peut pas bien représenter la variation statistique d'un tel signal. On utilise plutôt un autre modèle qui suit une loi de Rayleigh. [6]

Le modèle d'un canal de Rayleigh est schématisé, où r_k est la version bruitée du signal x_k à l'entrée du canal. Ce modèle est caractérisé par deux paramètres. L'un de ces deux paramètres est une variable aléatoire n de distribution gaussienne, et l'autre variable est l'enveloppe du signal a_k.

Le modèle d'un canal de Rayleigh est schématisé à la **figure 1.13**, où r_k est la version bruitée du signal x_k à l'entrée du canal. Ce modèle est caractérisé par deux paramètres. L'un de ces deux paramètres est une variable aléatoire n de distribution gaussienne, et l'autre variable est l'enveloppe du signal a_k. [6]

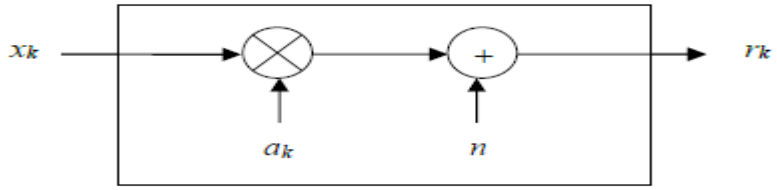


Figure 1. 12:Modèle de canal de Rayleigh

Le signal reçu x_k s'exprime dans ce cas par :

$$r_k = a_k x_k + n \qquad (1.62)$$

$$p(r_k|x_k, a_k) = \frac{1}{2\pi N_0} e^{-\frac{(r_k - a_k x_k)^2}{2N_0}} \qquad (1.63)$$

Où p présente la densité de probabilité de bruit sur le canal de Rayleigh.

Ce modèle prendre en compte la modification de l'amplitude du signal x_k. La fonction de densité de l'enveloppe du signal $a_k$est :

$$p(a_k) = a_k e^{\frac{a_k^2}{2}} \quad , a_k \geq 0 \qquad (1.64)$$

La méthode la plus simple pour obtenir l'enveloppe dont la puissance moyenne est unitaire est de générer deux variables gaussiennes b_k et b_k de variance $1/2$ et de moyenne nulle. En considérant que le processus d'évanouissement est décorrélé, l'enveloppe du signal sera alors donnée par :

$$a_k = \sqrt{b_k^2 + c_k^2} \qquad (1.65)$$

1.3.3.7.3. Capacité du canal BI-AWGN

La limite préalablement présentée considère un alphabet infini et réel présent à l'entrée du canal. Or, lorsqu'une modulation numérique est employée, les entrées sont discrètes et font partie d'un alphabet fini. Dans le cadre d'une modulation à changement de phase binaire (BPSK), chaque bit du mot de code est translaté sur $\{-1\ ;\ 1\}$. Ce canal AWGN contraint sur son entrée est appelé canal AWGN à entrer binaire (BI-AWGN). [9]

$$C_{BI-AWGN} = -\int_{-\infty}^{\infty} p(y)\, log_2\big(p(y)\big)dy\ 0{,}5\, log_2(2\pi e\, \sigma^2) \qquad (1.66)$$

Avec :

$$p(y) = \frac{1}{2}(p(y/x = +1) + (p(y/x = -1)$$

$$= \frac{1}{2}\frac{1}{\sqrt{2\pi\sigma}}\left(exp\left(\frac{-(y+1)^2}{2\sigma^2}\right) exp\left(\frac{-(y-1)^2}{2\sigma^2}\right)\right. \qquad (1.67)$$

1.3.3.7.4. Gain de codage

La probabilité d'erreur sur le canal de transmission est fonction du rapport signal à bruit. Comme esquissé précédemment, il peut être exprimé comme le rapport entre l'énergie moyenne par élément binaire transmis E_b et la densité spectrale mono-latérale du bruit N_0. En l'absence de codage, le rapport signal à bruit à l'entrée du récepteur est égal à E_b/N_0. En

revanche, dans le cadre d'un codage de canal de rendement R, la probabilité d'erreur sur le canal est fonction de R E_b/N_0 [9]

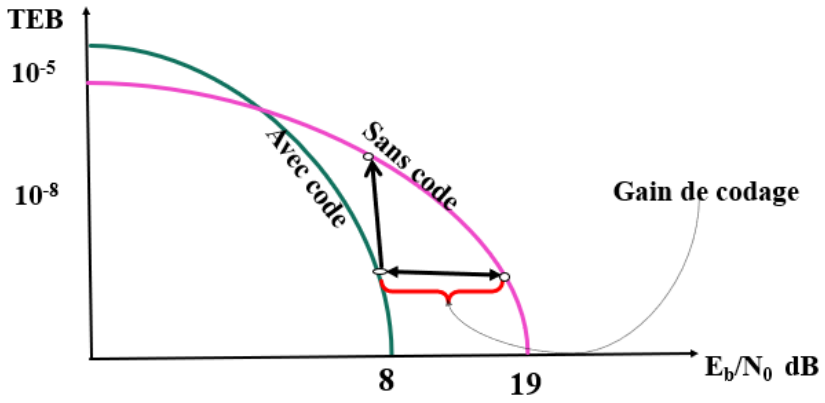


Figure 1. 13:le gain de codage

1.3.3.8. Les turbo codes standardisés

Les performances de décodage des turbo codes, plusieurs comités de standardisation se sont intéressés à cette nouvelle famille de codes correcteurs d'erreurs.

Ainsi, en 1999, le Comité Consultatif pour les Systèmes de Données Spatiales (CCSDS, qui regroupe différentes agences spatiales à travers le monde), propose dans leur livre de recommandations d'employer un turbo code à 16 états dans le cadre des communications avec l'espace lointain. Aujourd'hui, nous retrouvons des turbo codes essentiellement dans des contextes de télécommunication mobile et de diffusion vidéo numérique.

Depuis la fin des années 90, le 3GPP a adopté un turbo code à 8 états dans le cadre de la technologie de téléphonie mobile de troisième génération UMTS. En même temps, son concurrent, le 3GPP2, fait de même pour la norme CDMA2000, avec un turbo code très proche de celui de l'UMTS mais proposant plus de rendements. Enfin, dans le cadre de la technologie de téléphonie mobile de quatrième génération, les standards LTE etLTE-Advanced adoptent les mêmes codes constituants mais un nouvel entrelacementbasé sur des permutations QPP est retenu. Le consortium européen DVB, dès 1994, s'est intéressé aux turbo codes pour le standard de diffusion vidéo numérique terrestre DVB-T. Cependant, ce n'est qu'en 2000 que les turbo codes sont standardisés dans ce contexte afin d'assurer la voie retour permettant l'ajout de services tels que des programmes TV interactifs ou l'accès à internet. Ainsi deux standards, le DVB-RCS et le DVB-RCT spécifient respectivement la voie retour du standard DVB-S et la voie retour du standard DVB-T.

Ces deux turbo codes standardisés sont des codes double binaires 8 états et leur entrelaceur est basé sur une permutation ARP. Depuis 2012, la seconde génération du standard DVB-RCS2 comprend un turbo code similaire mais à 16 états.

D'autres standards utilisent des turbo codes tels que le WiMAX pour des communications numériques sans fil. Ces trois turbo codes sont similaires à celui du DVB-RCS. [6]

	Binaire		Double binaire	
	LTE	**CCSDS**	**DVB-RCS**	**DVB-RCS2**
Type de communication	Mobiles		Satellitaires	
Nombre d'états	8	16	8	16
Terminaison du treillis	Retour à l'état 0		Circulaire	
Tailles de trame (bits)	$40 \rightarrow 6144$	$1784 \rightarrow 8920$	$96 \rightarrow 1728$	$128 \rightarrow 3008$
Rendements	Rate-Matching	$1/6 \rightarrow 1/2$	$1/3 \rightarrow 6/7$	$1/3 \rightarrow 7/8$

Tableau 1 Les turbo codes standardisés

1.4. Conclusion

Les turbo codes sont les uns des plus importants codes utilisés pour la correction d'erreurs qui ont révolutionné le domaine des communications numériques, Malgré leur simple principe, ils contribuent de manière significative à réduire le taux d'erreur dans les transmissions numériques. Ils sont utilisés dans de nombreuses technologies de télécommunications. Mais leur inconvénient, est le délai mis dans le processus de décodage itérative, plus le nombre d'itérations dans le logarithme est grand, plus le processus de décodage prend de délai.

Dans ce chapitre nous avons tout d'abord introduit les systèmes de communication, Ensuite, nous avons répertorié les différents codes de correction d'erreur, et en particulier le code convolutionnelle, et après nous avons présenté une définition de turbo code et leur principe, Cela inclut le codage et le décodage avec ses différents algorithmes.

2.1. Introduction

L'entrelacement ou permutation, est une technique de traitement de signal standard utilisée dans une variété de systèmes de communication, y compris le turbo code. L'idée initiale et fondamentale d'introduire un entrelaceur dans la structure concaténée parallèle des turbocodes était de lutter efficacement contre l'occurrence d'éclat d'erreurs (burst errors), sur au moins l'une des dimensions du code composite. Où il est basé sur le principe de brassage de l'ordre des donnée les emplacements des erreurs introduites dans la transmission, permettant l'utilisation de codes de correction d'erreurs aléatoires au niveau du récepteur.

Il existe deux sortes d'entrelaceurs classiques, le bloc et le convolutionnel. Le premier consiste à insérer des bits dans une matrice le long des lignes et de lire les bits de sortie colonne par colonne. L'entrelaceur aléatoire est un dérivé de ce dernier étant donné que des bits sont mis en mémoire de façon séquentielle et lus à la sortie de façon aléatoire. Le principe du deuxième entrelaceur (convolutionnel) est différent de celui du bloc dans le sens où la notion de délai est induite. Dans ce chapitre, nous étudierons tous ces types et verrons le principe de chacun.

2.2. Définition de l'enterlaceur

L'entrelacement est un processus de réorganisation de l'ordre d'une séquence de symboles. Où l'entrelaceur prend des symboles d'un alphabet fixe à l'entrée et produit les symboles identiques à la sortie dans un ordre temporel différent. L'usage classique de l'entrelacement consiste à " Aléatoire" les emplacements des erreurs introduites dans la transmission, permettant l'utilisation de codes de correction d'erreurs aléatoires au niveau du récepteur. En turbo-codage, l'entrelacement est utilisé avant que le deuxième codeur ne veuille saisir les données d'information dans le codeur.

L'entrelacement est exprimé à partir d'une séquence de permutation $\Pi = \{\pi_1, \pi_2, \pi_3 \ldots, \pi_n\}$ où la séquence $\{\pi_1, \pi_2, \pi_3 \ldots, \pi_n\}$ représente la permutation des entiers de 1 jusqu'à n. La fonction d'entrelacement consiste à générer deux bits de parités complètement différents une fois qu'ils sont introduits dans deux codeurs. En utilisant l'entrelaceur qui conduit à des permutations aléatoires pour des milliers de bits, nous pouvons obtenir les meilleures performances du turbo code. [1]

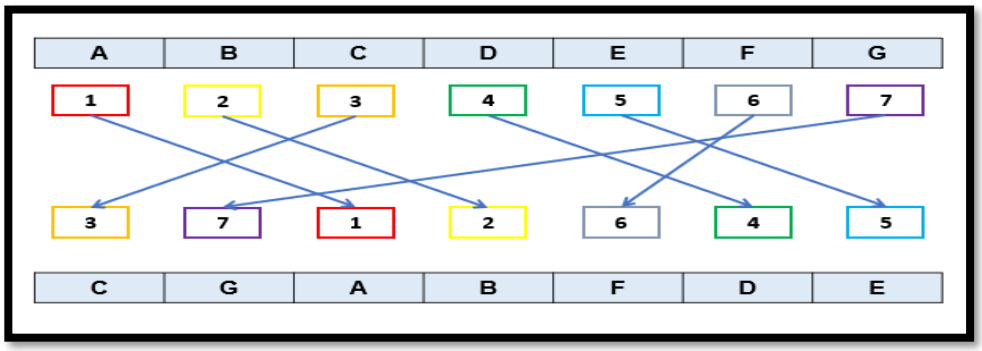


Figure 2. 1 Fonction d'entrelacement

La figure 2.1 représente un exemple de règle d'entrelacement représenté sous la forme d'une séquence de données A, B, C, D, E, F, G, Ce qui, après avoir été entrelacé selon la règle de permutation représentée par les séquences numériques, nous donnera la prochaine séquence de données C, G, A, B, F, D, E

Le rôle de base d'un entrelacement est de construire le code de bloc long à partir de petits codes convolutionnels de mémoire, tant qu'il peut approcher la limite de capacité de Shannon.

Le rôle final de l'entrelaceur est de rompre les séquences d'entrée de faible poids, et donc d'augmenter la distance de Hamming sans code ou de réduire le nombre de mots de code avec de petites distances dans le spectre de distance de code. La taille et la structure des entrelaceurs jouent un rôle majeur dans la performance des turbo codes.

Les deux principaux problèmes dans la conception d'entrelaceur sont la taille de l'entrelaceur et la carte de l'entrelaceur. La taille de l'entrelaceur joue un rôle important dans le compromis entre les performances et le temps (délai) car les deux sont directement proportionnels à la taille. D'un autre côté, la carte de l'entrelaceur joue un rôle important dans la définition des performances du code. Un autre rôle clé de l'entrelaceur est de façonner la distribution de poids du code, qui contrôle finalement ses performances.

L'entrelacement fait généralement référence à un processus qui permute les symboles d'une séquence d'entrée. Il est particulièrement utilisé dans le codage de correction d'erreur directe pour réduire l'effet du bruit impulsif et des erreurs de salve dans les canaux à évanouissement et à trajets multiples. Pour la même raison, il est également appliqué dans les systèmes d'enregistrement magnétique.

2.3. Paramètres d'un entrelaceur

Les paramètres essentiels de l'entrelaceur sont :

2.3.1. Le facteur d'étalement (Spreading)

Nous disons qu'un entrelaceur a des facteurs de propagation (s, t) ,si la condition est remplie :

$|i - j| < s$ Alors $|\pi(i) - \pi(j)| \geq t$, Notez que cette définition est symétrique en ce sens que chaque fois C'est l'équivalent $|\pi(i) - \pi(j)| < t$,alors $|\pi(i) - \pi(j)| \geq s$, dans ce l'entrelacement, les bits individuels d'une rafale de longueur inférieure à t en entrée seront séparés en sortie en blocs distincts de longueur supérieure ou égale à s. Si un entrelaceur a des facteurs d'étalement (s, t) alors un désentrelaceur correspondant à des facteurs d'étalement (t, s). Un entrelaceur donné peut avoir un certain nombre de facteurs d'étalement. [10]

Soit π : $Z_K \rightarrow Z_K$ une permutation, La propagation de π est le plus grand entier s tel que :

$$|i - j| < s \implies |\pi(i) - \pi(j)| \geq s, \ pour\ 1 \leq s \leq K$$

Où i, j sont des valeurs d'entrée telles que i ≠ j.

Exemple : Pour la permutation suivante, nous calculons spread :

$$\Pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 5 & 0 & 3 & 1 \end{pmatrix}$$

Nous regardons toutes les paires pour lesquelles | i − j | < 2, et vérifiez si | π (i) −π (j) | ≥ 2. Ainsi,

$$|\pi(1) - \pi(0)| = |2 - 4| = 2,$$

$$|\pi(2) - \pi(1)| = |5 - 1| = 4,$$

$$|\pi(3) - \pi(2)| = |0 - 5| = 5,$$

$$|\pi(4) - \pi(3)| = |3 - 0| = 3,$$

$$|\pi(5) - \pi(4)| = |1 - 3| = 2,$$

Comme nous pouvons le voir, toutes les paires d'entrées qui sont éloignées de moins de 2 ont une distance permutée d'au moins 2 les unes des autres. Par conséquent, s = 2 est un écart possible. Nous devons maintenant considérer le cas pour lequel s = 3. Nous examinons toutes les paires pour lesquelles | i - j | <3 et vérifier si | π (i) −π (j) | ≥ 3. Premièrement, nous considérerons les valeurs d'entrée qui sont à une distance de 2 les unes des autres ; si la

définition est valable pour ces entrées, nous vérifierons les valeurs d'entrée qui sont éloignées l'une de l'autre. Donc

$$|\pi(2) - \pi(0)| = |4 - 5| = 1,$$

$$|\pi(3) - \pi(1)| = |2 - 0| = 2,$$

$$|\pi(4) - \pi(2)| = |5 - 3| = 2,$$

$$|\pi(5) - \pi(3)| = |0 - 1| = 1.$$

Nous pouvons voir que les valeurs d'entrée 2 et 0 produisent des valeurs de sortie qui ne sont éloignées que de 1, et donc l'écart ne peut pas être 3. Par conséquent, selon la proposition 38, l'écart de π est 2. S = 2. [11]

2.3.2. La Dispersion

Cette propriété nous aidera à comparer la distance entre deux valeurs en Z_k et la distance entre leurs images permutées, où Z_K est défini comme l'ensemble des entiers {0,1, ..., K− 1} Pour tout entier positif q, et K la taille ou la longueur (bloc) de l'entrelaceur. [12]

Une dispersion élevée indique une variété de distances permutées entre les éléments. La première étape du calcul de la dispersion consiste à acquérir une liste complète des déférences.

Étant donné une fonction d'entrelacement π de Z_K, la liste des déférences de π est définie comme l'ensemble :

$$D(\pi) = \{(j - i, \pi(j) - \pi(i) | 0 \leq i < j < q)\} \qquad (2.\ 1)$$

Exemple : Nous calculons la liste des déférences pour la permutation suivante :

$$\Pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 5 & 0 & 3 & 1 \end{pmatrix}$$

Nous commençons par calculer toutes les paires pour lesquelles les valeurs d'entrée sont à distance 1 les unes des autres :

$$\big(1 - 0, \pi(1) - \pi(0)\big) = (1, -2),$$

$$\big(2 - 1, \pi(2) - \pi(1)\big) = (1,3),$$

$$\big(3 - 2, \pi(3) - \pi(2)\big) = (1, -5),$$

$$\big(4 - 3, \pi(4) - \pi(3)\big) = (1,3),$$

Et

$$\big(5-4,\pi(5)-\pi(4)\big)=(1,-2).$$

Ainsi, les premiers éléments de l'ensemble D (π) sont (1, −2), (1,3) et (1, −5). La liste des déférences étant un ensemble, les éléments répétés ne doivent pas être comptés deux fois. Nous pouvons continuer à calculer les paires restantes en vérifiant les entrées qui sont 2,3,4 et 5 éloignées les unes des autres. Par conséquent, la permutation a la liste de différences suivante :

$$D(\pi)\ =\ \{(1,-2),(1,3),(1,-5),(2,1),(2,-2),(3,-4),(3,1),(4,-1),(5,-3)\}.$$

Par conséquent,

$$|D(\pi)|\ =\ 9. \qquad (2.\ 2)$$

Étant donné une permutation π, la dispersion de π est donnée par :

$$\frac{|D(\pi)|}{\binom{K}{2}}=\frac{|D(\pi)|}{\frac{K(K-1)}{2}} \qquad (2.\ 3)$$

Dans l'exemple précédent, on voit que K = 6 car la permutation agit sur 6 objets. De plus, nous savons que la cardinalité de D (π) est 9. Nous pouvons maintenant calculer la dispersion en utilisant la définition : [12]

$$\frac{|D(\pi)|}{\frac{K(K-1)}{2}}=\frac{9}{\frac{6(5)}{2}}=\frac{9}{\frac{30}{2}}=\frac{3}{5} \qquad (2.\ 4)$$

2.3.3. Le délai

Le délai d'un entrelaceur est un paramètre très important dans le design d'un système de transmission. Il est très important de connaître à l'avance ou de pouvoir évaluer le délai que notre système Va introduire. [11]

2.3.4. Mémoire

Tout entrelaceur a besoin d'une certaine mémoire pour s'entrelacer. Un entrelaceur est Causal si la sortie au temps i ne dépend que des entrées précédentes ou courantes.

La mémoire requise est alors le nombre de bits qui changent de position, Si notre entrelaceur N'est pas causal, la mémoire requise est tout simplement la longueur de I'entrelaceur. [11]

2.4. Les types des entrelaceurs

Il existe deux types d'entrelaceurs : l'entrelacement de type bloc, et l'entrelacement de type convolutionnelle, pour le turbo code, on utilise les entrelaceurs de type bloc.

2.4.1. Entrelaceur de type bloc

Ce type d'entrelaceur est caractérisé par le traitement en bloc plutôt qu'en série (comme dans le cas des entrelaceurs convolutionnels) des symboles d'une séquence à réordonner. Les entrelaceurs de blocs sont spécifiés par la conception d'une permutation sur les entiers {0, 1,2, ..., K - I}. La conception implique de nombreux éléments qui sont utilisés pour optimiser l'entrelaceur en termes de complexité et de performances (par exemple, paramètre s ou dispersion). Parfois, une permutation est conçue pour avoir des propriétés supplémentaires. Par exemple, une permutation génère un entrelaceur pair-impair si la permutation satisfait l'équation.

Les entrelaceurs de bloc généralement divisé en trois familles, les entrlaceurs Aléatoire, régulier (déterministe) comme les entrelaceurs algébriques, et des entrlaceurs irrégulier (hybride).

2.4.1.1. L'entrelaceur réguliers

Le point de départ dans la conception d'un entrelacement est la permutation régulière, qui est décrite en **figure 2.2** sous deux formes différentes. La première suppose que le bloc contenant k bits peut être organisé comme un tableau de *m* lignes et *n* colonnes. L'entrelacement consiste alors à écrire les données dans une mémoire ad hoc, ligne par ligne, et à les lire colonne par colonne (Figure 2.2 (a)). La seconde s'applique sans hypothèse sur la valeur de k. Après écriture des données dans une mémoire linéaire (adresse i, $0 \leq i \leq k-1$), le bloc est assimilé à un cercle, les deux extrémités (i = 0 et i = k−1) étant alors contiguës (figure 2.2 (b)). [2]

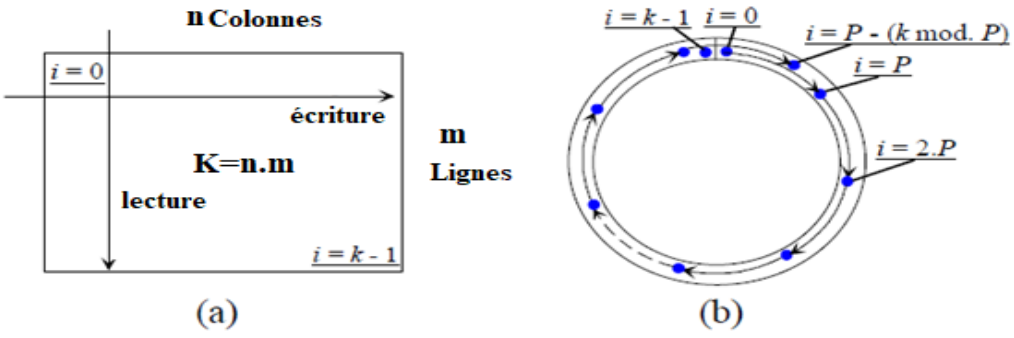


Figure 2. 2: Permutation régulière sous formes rectangulaire (a) et circulaire (b).

2.4.1.2. L'entrelaceur irréguliers

Nous ne ferons pas, dans cette section, une description de toutes les permutations irrégulières qui ont pu être imaginées jusqu'à ce jour et qui ont fait l'objet de nombreuses publications ou de plusieurs chapitres d'ouvrages Nous avons plutôt retenu de présenter ce qui semble être, pour le moment, le type de permutation à la fois le plus simple et le plus performant. Il s'agit de permutations circulaires presque régulières, appelées *ARP* (*almost regular permutation,* ou *DRP* (*dithered relatively prime),* suivant les auteurs. Dans tous les cas, l'idée est de ne pas trop s'éloigner de la permutation régulière, bien adaptée aux motifs d'erreurs *RTZ* simples et d'instiller un petit désordre contrôlé pour contrer les motifs d'erreurs *RTZ* multiples. [4]

2.4.1.3. Les entrelaceur hybrides

Ces entrelaceurs portent les caractéristiques des « entrelaceurs aléatoires » et des « entrelaceurs régulier » ensemble.

2.5. Quelques types des entrelaceurs

2.5.1. Entrelaceur en matrice

Le principe de fonctionnement de ce type d'entrelacement est d'écrire les données ligne par ligne et ensuite ces données sont lues colonne par colonne. Cette méthode est très simple mais l'inconvénient est qu'elle fournit très peu d'aléatoire. Ce type de l'interlaceur est un enterlaceur de période, la taille de cet entrelaceur est définie par : K=n.m [1]

- **Exemple :** Supposons les entrées du turbo codeur comme suit :

0	1	2
3	4	5
6	7	8

Tableau 2. 1:les entrées du turbo codeur

La sortie de l'entrelaceur est comme suivants : **0 3 6 1 4 7 2 6 8**

2.5.2. Entrelaceur hélicoïdal

Dans un entrelaceur hélicoïdal, les données sont écrites ligne par ligne dans les mémoires et les lit en diagonale à partir de l'entrée en haut à gauche de l'entrelaceur en utilisant la permutation suivante :

$$\pi(t) = i_r n + j_r \qquad (2.5)$$

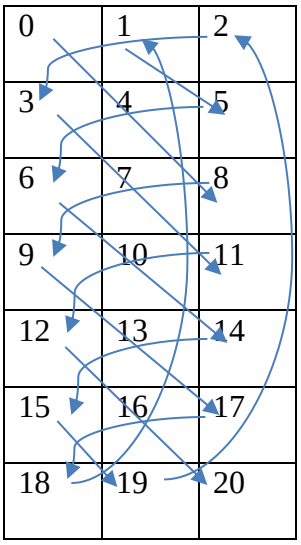
$$i_r = n * m - 1 - t(mod\ m) \qquad (2.\ 6)$$

$$j_r = t(mod\ n) \qquad (2.\ 7)$$

m et *n* sont premiers, Où *m* représentent le nombre de ligne et *n* représentent le nombre de colonne de l'entrelaceur. [1]

- **Exemple :** Supposons les entrées du turbo codeur comme suit :

0	1	2
3	4	5
6	7	8
9	10	11
12	13	14
15	16	17
18	19	20



Dans la sortie, nous obtenons la séquence de symboles suivante :

0	4	8	9	13	17	18	1	5	6	10	14	15	19	2	3	7	11	12	16	20

Figure 2. 3:Entrelaceur hélicoïdal

2.5.3. Entrelaceur bloc classique

L'entrelacement classique repose sur l'échange les positions des symboles d'une séquence deux à deux sans répétition, c-à-d, pour deux symboles A et B de positions initiales respectives I et J, si A à la position J après entrelacement alors nécessairement B devra occuper celle de I. dans la plupart des cas, l'entrelaceur et le désentrelaceur sont implémentés séparément puisqu'ils sont différents. Cela nécessite un énorme espace mémoire surtout pour des entrelaceurs de grandes tailles. [1]

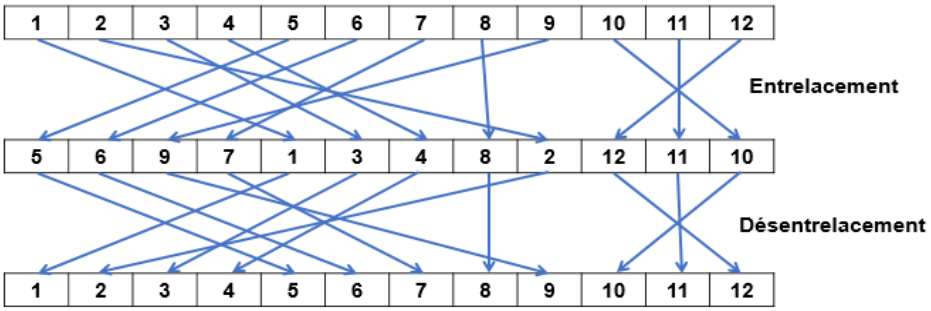


Figure 2. 4 : Entrelaceur et désentrelaceur bloc classique

2.5.4. Entrelaceur de berrou-glavieux

Ce type d'entrelaceur est une amélioration pour de l'entrelaceur ligne-colonne en termes de performances dans les applications des turbos codes, Il a été conçu par Berrou et Glavieux

Le principe de fonctionnement de cet entrelaceur est le suivant : La taille K de l'entrelaceur est choisie comme étant une puissance de 2, avec le nombre de lignes doit être égal au nombre de colonnes où : k=n*n. [1]

Les permutations sont obtenues en appliquant les formules suivantes :

$$i_r = \left(\frac{n}{2} + 1\right)(i + j)(mode\ n) \qquad (2.\ 8)$$

$$\xi = (i + j)(mod\ 8) \qquad (2.\ 9)$$

$$j_r = [p(\xi)(j + 1)] - 1(mod\ n) \qquad (2.\ 10)$$

ξ	$p(\xi)$
0	17
1	37
2	19
3	29
4	41
5	23
6	13
7	7

Tableau 2. 2:La fonction pseudo-aléatoire ξ

Où i et j sont la ligne et la colonne d'écriture d'un symbole et i_r et $j_r$donnent la position de lecture. En outre, $p(\xi)$ est un nombre premier avec *w*, qui est déterminée par les valeurs présentées dans **le tableau 2.2**.

2.5.5. Entrelaceur régulier (RI)

Les données binaires sont alors extraites de telle sorte que la j[ème] donnée lue ait été préalablement écrite à la place i, de valeur :

$$i = \pi(j) = p_j + i_0 \ mod \ k \qquad (2.\ 11)$$

Où : *P* est un entier premier avec *k* et *i*0 est l'indice de départ 1.

Pour une permutation circulaire, définissons la distance spatiale cumulée S (j1, j2) comme la somme des deux distances spatiales séparant deux bits, avant et après permutation, dont les indices de lecture sont j1 et j2 :

$$S(j_1, j_2) = f(j_1, j_2) + f(\pi(j_1), \pi(j_2)) \qquad (2.\ 12)$$

Où

$$f(u, v) = min\{|u - v|, k - |u - v|\} \qquad (2.\ 13)$$

La fonction *f* est introduite pour tenir compte du caractère circulaire des adresses. Finalement, nous appelons S_{min} la plus petite des valeurs de *S* (*j*1, *j*2), pour toutes les paires *j*1 et *j*2 possibles :

$$s_{min} = min\{s(j_1, j_2)\} \qquad (2.\ 14)$$

Avec une permutation régulière, la valeur de P qui maximise S_{min} est :

$$sup\ s_{min} = \sqrt{2k} \qquad (2.\ 15)$$

Cette borne supérieure n'est atteinte que dans le cas d'une permutation régulière et avec les conditions :

$$P = P_0 = \sqrt{2k} \qquad (2.\ 16)$$

Et :

$$k = \frac{P_0}{2} \ mod \ P_0 \qquad (2.\ 17)$$

En pratique, pour satisfaire le critère de l'espace total maximal distance, P est choisi comme un entier proche de P_0, et premier avec k. Permutation presque régulière (ARP).

2.5.6. L'entrelaceur almost regular permutation (ARP)

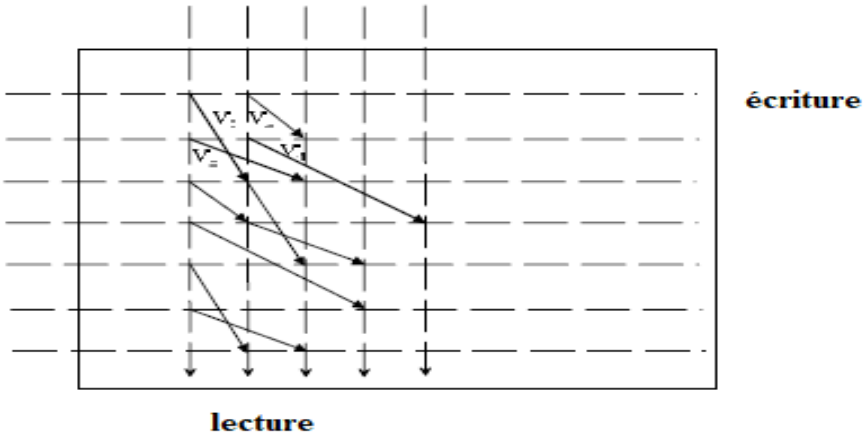


Figure 2. 4: Permutation ARP

L'ARP est un modèle d'entrelaceur basé sur un entrelacement global circulaire régulier dans laquelle on insérer un certain degré de désordre par le biais d'une permutation locale

La structure d'entrelacement ARP est dérivée de l'entrelaceur régulier (RI) :

$$i = \pi(j) = P_j + Q(j) + i_0 mod\ K \qquad (2.\ 18)$$

Si l'on choisit

$$Q(j) = A(j)P + B(j) \qquad (2.\ 19)$$

Où P est la période RI qui doit être relativement prime à K. Les modèles d'erreur de retour à zéro rectangulaire (RTZ) ne peuvent pas être efficacement évités avec cette permutation en raison de sa structure régulière.

Où les entiers positifs $A(j)$ et $B(j)$ sont périodiques, de cycle C (diviseur de k), alors ces grandeurs correspondent à des décalages positifs appliqués respectivement avant et après la permutation régulière. C'est la différence avec la permutation DRP, dans laquelle les perturbations en écriture et en lecture se réalisent à l'intérieur de petits groupes de données et non par décalage.

Pour que la permutation soit bien une bijection, les paramètres $A(j)$ et $B(j)$ ne sont pas quelconques. Une condition suffisante pour assurer l'existence de la permutation est que les paramètres soient tous multiples de C. Cette condition n'est pas très contraignante vis-à-vis de l'efficacité de la permutation. (2.13) peut alors être récrit sous la forme

$$Q(j) = C(\alpha(j)P + \beta(j)) \qquad (2.\ 20)$$

Où $\alpha(j) et\ \beta(j)$sont le plus souvent de petits entiers, de valeurs 0 à 8. Par ailleurs, puisque les propriétés d'une permutation circulaire ne sont pas modifiées par une simple rotation, l'une des valeurs *Q*(*j*) peut être systématiquement 0. [4]

- Deux jeux typiques de valeurs *Q*, avec un cycle 4 et α = 0 ou 1, sont donnés ci-dessous

Si $j = 0 \mod 4, \quad alors \quad Q = 0$

Si $j = 1 \mod 4, \quad alors \quad Q = 4P + 4\beta_1$

Si $j = 2 \mod 4, \quad alors \quad Q = 4\beta_2$

Si $j = 3 \mod 4, \quad alors \quad Q = 4P + 4\beta_3$

Si $j = 0 \mod 4, \quad alors \quad Q = 0$

Si $j = 1 \mod 4, \quad alors \quad Q = 4\beta_1$

Si $j = 2 \mod 4, \quad alors \quad Q = 4P + 4\beta_2$

Si $j = 3 \mod 4, \quad alors \quad Q = 4P + 4\beta_3$

Ces modèles requièrent la connaissance de seulement quatre paramètres ($P, \beta 1, \beta 2\ et\ \beta 3$) ont été validés sur 8 différents ou turbo-codes à 16 états, qui peuvent être déterminés selon la procédure détaillée. L'utilisation de codes *m*-binaires, au lieu de codes binaires, demande simplement de remplacer *k* par *k/m*. En particulier, les permutations définies pour les turbo-codes double-binaires (*m* = 2) des normes DVB-RCS, DVB-RCT et WiMax sont inspirées.

Où i : est l'indice séquentiel des positions binaires après entrelacement, $0 \leq i \leq K - 1$

$\pi\ (i)$: est l'indice binaire avant entrelacement correspondant à la position i,

K : est la taille du bloc d'informations en bits.

2.5.7. L'entrelaceur Dithered Prime relative interleaver (DRP)

Crozier proposé l'entrelaceur DRP (Dithered Prime relative interleaver) qui vise à augmenter la distance minimale des bits adjacents dans les flux binaires à l'entrée d'entrelaceur. Cet entrelaceur fonctionne selon les étapes suivantes : Tout d'abord, les flux binaires ayant une longueur *K* sont subdivisés à des blocs plus petits comme fenêtres avec des tailles *R*. Les données de chaque fenêtre sont permutées localement à l'entrée de l'entrelaceur, le soi-disant

processus de tramage de la donnée d'entrée. Ensuite, les données obtenues sont cycliquement décalées en fonction de permutation suivante :

$$j = i + sp \qquad (2.\ 21)$$

Où :

S : valeur de décalage, **P** : entier premier par rapport à *K*

Enfin, un tramage de sortie similaire au tramage à l'entrée est effectué. Les entrelaceurs DRP fonctionnent bien lorsque les deux codeurs RSC sont terminés à l'état zéro. [4]

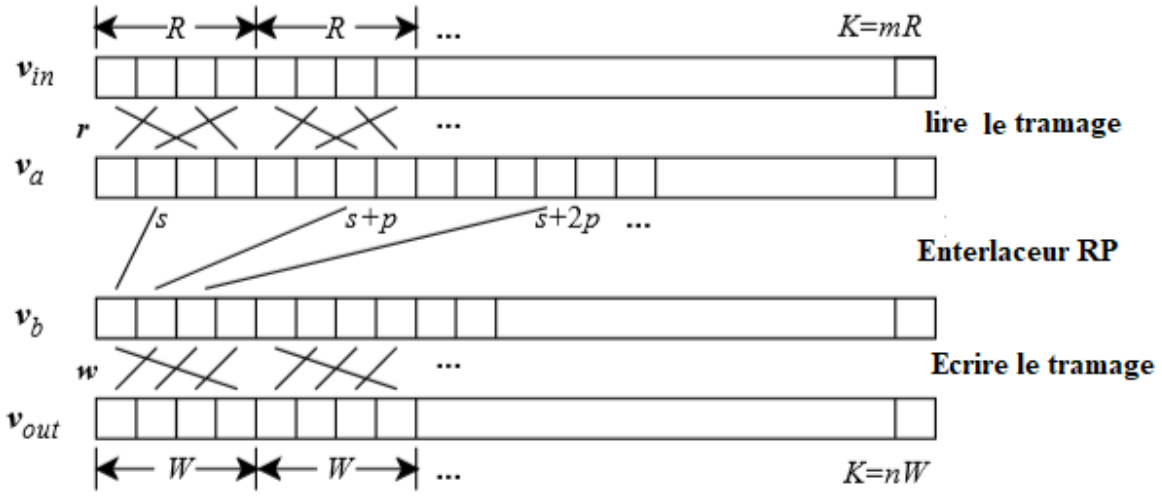


Figure 2. 5: enterlaceur DRP

La figure 2.6 montre l'approche utilisée pour concevoir des enterlaceurs DRP, qu'il comprend trois étapes :

Tout d'abord, le vecteur d'entrée V_{in} est tramé (permuté localement) à l'aide d'un petit vecteur de tramage de lecture *r* de longueur *R* (e.g. 2,4,8,16). Ensuite, le vecteur résultant, V_a est permuté en utilisant un entrelaceur relatif premier (RP), avec un indice de départ s et un incrément premier relatif *p*, pour obtenir une bonne répartition. Enfin, le vecteur résultant, V_b est tramé à l'aide d'un petit vecteur de tramage d'écriture, w, de longueur W, pour générer le vecteur de sortie V_{out}. La longueur de l'entrelaceur, K, doit être un multiple de R et W. Les vecteurs de tramage de lecture et d'écriture courts ne détruiront pas les bonnes propriétés d'étalement d'un entrelaceur RP, mais auront tendance à réduire quelque peu l'étalement. Le vecteur d'entrelacement global équivaut I, est ensuite déterminé. [13]

Les équations pour les différents vecteurs entrelaceurs DRP représentés sur **la figure 2.6** peuvent être exprimées comme suit :

$$v_a(i) = v_{in}\big(I_a(i)\big)$$

$$v_b(i) = v_a\big(I_b(i)\big)$$

$$v_{out}(i) = v_b\big(I_c(i)\big) \qquad i = 0 \dots K-1 \qquad (2.\ 22)$$

Où :

$$I_a(i) = R(i/R) + r(i\ modulo\ R) \qquad i = 0 \dots K-1 \qquad (2.\ 23)$$

$$I_b(i) = (s + ip)\ modulo\ K) \qquad i = 0 \dots K-1 \qquad (2.\ 24)$$

$$I_c(i) = W(i/W) + w(i\ modulo\ W) \qquad i = 0 \dots K-1 \qquad (2.\ 25)$$

Ainsi, le vecteur d'entrée peut être entrelacé en utilisant :

$$v_{out}(i) = v_{in}\big(I(i)\big) \qquad i = 0 \dots K-1 \qquad (2.\ 26)$$

$$I(i) = I_a\left(I_b\big(I_c(i)\big)\right) \qquad i = 0 \dots K-1 \qquad (2.\ 27)$$

2.5.7. Entrelacement Chaotic Golden Interleaver (CGI)

Conception d'entrelaceur d'or chaotique Puisque les signaux chaotiques peuvent générer des séquences aléatoires de haute performance, nous nous tournons pour discuter de la façon de concevoir un entrelaceur efficace pour le codage turbo, en combinant la séquence chaotique avec la génération du vecteur doré tramé, nous pouvons améliorer le caractère aléatoire de l'entrelaceur doré. Un tel entrelaceur est appelé entrelaceur doré chaotique tramé (CI). Le tramage d (i) dans la structure DGI est maintenant obtenu en appliquant une certaine séquence chaotique. Soit x (i) la séquence chaotique considérée, définie comme :

$$X(i+1) = \big(p + q * X(i)\big) mod\ k \qquad (2.\ 28)$$

Où x0 est la séquence initiale, p, q et k sont des nombres entiers. La séquence x (i) sera aléatoire si l'on fait un bon choix sur les paramètres p, q et k.

2.5.8. Entrelacement dithered golden interleaver (DGI)

Il a été constaté pour les Turbo-codes que les entrelaceurs avec un certain caractère aléatoire ont tendance à mieux fonctionner que les entrelaceurs complètement structurés, en particulier pour les grandes tailles de bloc de l'ordre de 1000 bits ou plus. Cependant, les propriétés

d'étalement de l'entrelaceur doré sont toujours très souhaitables, à la fois pour maintenir une bonne distance minimale (une courbe d'erreur abrupte à des SNR élevés) et pour assurer une convergence rapide en étalant efficacement les rafales d'erreur dans tout le bloc. Ces deux caractéristiques sont englobées dans l'entrelaceur doré tramé. La seule différence entre l'entrelaceur doré et l'entrelaceur doré tramé est l'inclusion d'un vecteur de perturbation réel (tramage) d, dans le vecteur doré v.

$$v(n) = s + nc + d(n), \ \ modulo\ K, \ \ n = 0 \dots K - 1, \qquad (2.\ 29)$$

Où d (n) est le n-th composant de tramage. Le tramage ajouté est uniformément réparti entre 0 et KD, où D est la largeur normalisée de la distribution de tramage. Le vecteur doré tramé v est trié et les indices d'entrelaceur sont générés d'une manière similaire à celle de l'entrelaceur doré décrit ci-dessus.

Crozier a également introduit un autre entrelaceur dépend de la définition de la section d'or et appelée entrelacement doré intermittent, Il fonctionne selon le principe suivant :

Pour un segment de longueur 1, il est de divisé le segment total en un segment long de longueur g et un segment plus court de 1−g de telle sorte que le rapport du segment long à la longueur totale est égal au rapport entre le plus court segment au segment plus long. Donc :

$$\frac{g}{1} = \frac{1-g}{g} \approx 0.618 \qquad (2.\ 30)$$

Il est prévu que ce type de l'entrelaceur augmente la distance entre les bits voisins de flux binaires d'entrée dans les données entrelacées. [1]

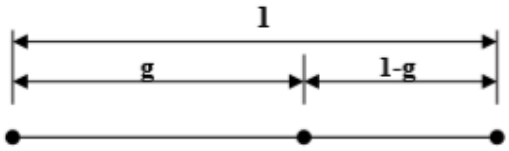


Figure 2. 6:Principe de la section d'or

L'entrelaceur or tramé (DGI) est identifié par le vecteur v, dont les éléments est calculée par

$$v(n) = [s + nC + d(n)] mod\ K \qquad (2.\ 31)$$

Où : S : constante de compensation ; K : la taille de l'etrelaceur ;

$d(n)$: est le nième component de vecteur de tramage qui est uniformément distribué entre 0 et KD, avec D est la largeur normalisée de la distribution *d(n)*, dans les turbos codes D qui représente la valeur de décalage est mis à 0.01, et présentée par :

$$C = K(g^m + j)/r \qquad (2.32)$$

Où : $m,\ j$ sont des entiers avec $m, j > 0$;

r : est un espacement d'indice entre les éléments voisins.

Dans les réalisations typique, j est mis à 0 et r est mis à 1

Par conséquent, **l'équation 2.18** devient comme suit :

$$C = K(g^m) \qquad (2.33)$$

Le vecteur de tramage peut être exprimée en tant que :

$$d(n) = C * \alpha(n\ mod\ KD) + \beta(n\ mod\ KD) \qquad (2.34)$$

Où : $\alpha(.)$ et $\beta(.)$: sont des vecteurs générés aléatoirement de longueur *KD*, appliqués périodiquement pour 0 ≤ n ≤ K-1 [1]

2.5.9. L'entrelaceur Quadratic Permutation Polynomial (QPP)

Pour un bloc d'informations de taille K, un entrelaceur QPP de taille K est défini par le polynôme suivant :

$$\pi(i) = (f_1 i + f_2 i^2) mod\ K \qquad 0 \le i \le K-1 \qquad (2.35)$$

Où : i : est l'indice séquentiel des positions de bit après entrelacement,

$\pi\ (i)$: est l'indice de bit avant entrelacement correspondant à la position i,

f_1, f_2 : Sont les coefficients qui définissent la permutation, les possibilités de ces coefficients sont liées à la factorisation de K.

Où 0 ≤ i ≤ K-1 est l'indice séquentiel de la position de bit après entrelacement, π (i) est l'indice de bit avant entrelacement correspondant à la position i, et f1 et f2 sont les coefficients qui définissent la permutation. Les conditions sur f1 et f2 pour lesquelles (6) définit une permutation valide sont largement prises en compte. Les possibilités de ces coefficients sont liées à la factorisation de K.Par exemple, lorsque K est pair, les conditions sont: i) f1 est impair

(relativement premier à K), et ii) tous les facteurs premiers de K sont également des facteurs de f2.

Par exemple, lorsque K est pair, les conditions sont :

i) f_1 Est impair (relativement premier à K),

ii) Tous les facteurs premiers de K sont également des facteurs de f2.

Les entrelaceurs QPP (et les polynômes de permutation en général) peuvent être calculés de manière récursive avec uniquement des additionneurs (c'est-à-dire que les multiplications dans l'équation 2.25 peuvent être réécrites avec des ajouts pour une réalisation récursive). Les entrelaceurs QPP sont sans contention et vectorisables au maximum, c'est-à-dire qu'ils fournissent un accès à la mémoire CF pour tout M qui facteur K [15]

A. Conception d'entrelaceurs ARP et QPP

Les entrelaceurs ARP et QPP peuvent être conçus sur la base de métriques de conception d'entrelaceurs classiques telles que l'étalement, le tramage et la distance minimale d_{min}. De plus, ces permutations ont des propriétés de distance périodiques qui contribuent à réduire la complexité de la recherche. En outre, l'effet de la terminaison du treillis sur la distance minimale doit être géré de manière appropriée. Pour les entrelaceurs basés sur QPP, fournit une nouvelle métrique de conception basée sur la combinaison des métriques de dispersion-dispersion, tandis que traite de la conception d'entrelaceurs ARP basée sur le calcul de la distance minimale en utilisant la méthode d'impulsion d'erreur. Une combinaison appropriée de ces techniques de conception peut être utilisée pour trouver des entrelaceurs candidats pour chaque taille de bloc.

B. Conception d'entrelaceurs ARP et DRP

Les entrelaceurs ARP et DRP sont basés sur un entrelacement global circulaire régulier, dans laquelle on insère un certain degré de désordre par le biais d'une permutation locale. Et sont tout à fait appropriés pour valider des critères de conception tels que l'envergure et la circonférence de corrélation pendant la sélection de leurs paramètres. [15]

C. Comparaison des entrelaceurs algébriques

Selon la description des entrelaceurs DRP, QPP et ARP, entrelacement similaire des propriétés peuvent être attendues de ces familles d'entrelaceurs. En effet, une bonne durée peut être atteint par leur composante RI. De plus, tous permettent l'introduction de désordre

dans la séquence de données d'entrelacement. Il est obtenu par les vecteurs de tremblement dans le cas DRP, le terme quadratique dans le cas QPP, et le vecteur de changements dans le Boîtier ARP. En revanche, seuls quelques paramètres doivent être stockés pour leur la mise en oeuvre. En outre, ils sont également bien adaptés à la mise en œuvre parallèle de turbo décodeur. Même si l'entrelaceur QPP est sans contention maximale, un degré de parallélisme peut être atteint avec chacun d'eux parce que le parallélisme très élevé les diplômes ne conviennent pas à une mise en œuvre pratique. Enfin, il convient de noter que Les entrelaceurs DRP et ARP sont tout à fait appropriés pour valider des critères de conception tels que l'envergure et la circonférence de corrélation pendant la sélection de leurs paramètres. [16]

2.5.10. Entrelaceurs aléatoire

Le concept fondamental d'un entrelaceur aléatoire est simple mais sa réalisation en pratique est plus complexe que celle des entrelaceurs bloc classique et hélicoïdal. Une fois que les symboles d'un bloc sont introduits dans un entrelaceur aléatoire, les symboles en sortie sont choisis d'une façon aléatoire de telle manière qu'il ne faut pas répéter le même symbole déjà sélectionné. Comme la sélection est aléatoire, il sera impossible de connaître les positions des symboles à la sortie d'entrelaceur. Par conséquent, il serait utile de garder une table de correspondance entre les anciennes et les nouvelles positions des symboles entrelacés afin de pouvoir désentrelacerces derniers. La figure 4.10 montre un exemple d'entrelaceur aléatoire. [1]

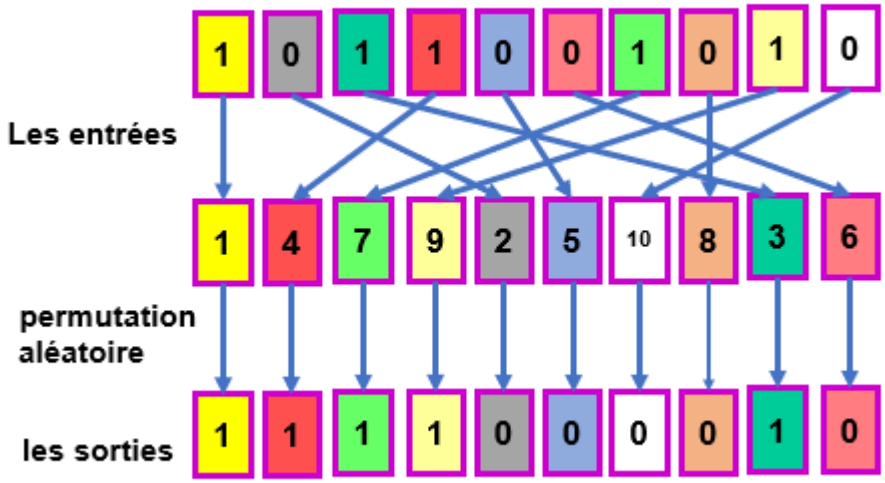


Figure 2. 7:entrelaceur aléatoire

2.5.11. Entrelaceur pseudo-aléatoire

Pratiquement, toutes les techniques de conception d'entrelacement des turbo codes dans la littérature sont basés sur l'algorithme de génération d'entrelacement s-aléatoire, Cet algorithme s'appuie essentiellement sur le choix au hasard d'un entrelaceur avec une limitation sur son étalement. Les symboles dans les sorties sont choisis au hasard afin que vous n'ayez pas à répéter le même symbole déjà sélectionné. Parce que la sélection est aléatoire, il est difficile de connaître les symboles à la sortie de l'interlaceur, Par conséquent, il serait utile de garder une table de correspondance entre les anciennes et les nouvelles positions des symboles entrelacés afin de pouvoir désentrelacer ces derniers. L'opération inverse consiste à écrire dans le désentrelaceur suivant le même ordre pseudo-aléatoire puis à lire la séquence selon l'ordre croissant.[16]

2.5.12. Entrelaceur S-aléatoire

Cet entrelaceur change de façon aléatoire les séquences de blocs de de telle manière que leurs symboles restent au moins espacés d'au moins une distance S. Chaque position d'un symbole sélectionné est ensuite comparée aux emplacements S déjà sélectionné précédemment. Ces positions ne sont conservées que position est distante d'au moins ±S par rapport aux S positions précédentes. Le reste sera exclu. Cela se fait jusqu'à la sélection de tous les éléments. Ainsi, pour s=1 on retrouve l'entrelaceur pseudo-aléatoire classique. Généralement $S < \sqrt{\frac{K}{2}}$, où K est la longueur du bloc à entrelacer. [5]

Par conséquent, l'algorithme de construire un entrelaceur s-aléatoire de taille N et de facteur S de dispersion est donné comme suit :

1. Considérer le vecteur I ϵ [0, …, K-1] ;
2. Choisir une valeur entière pour le facteur de dispersion $S < \sqrt{\frac{K}{2}}$ afin que l'algorithme soit réussi.
3. Comparez i aux S précédents. Pour chacun des S entiers, comparez i pour voir s'il se situe dans ± S. Si i se trouve dans la plage, revenez à l'étape 1. Sinon, gardez i.
4. Revenez à l'étape 1 jusqu'à ce que tous les postes L aient été pourvus.

Exemple :

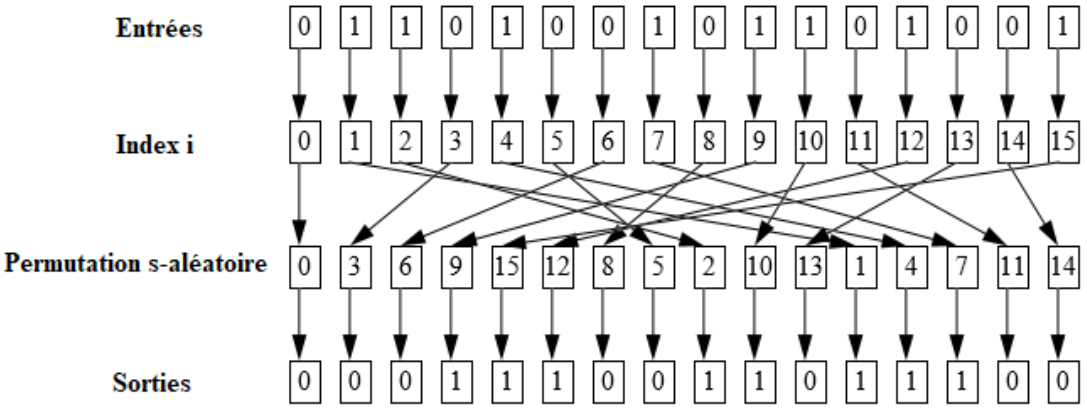


Figure 2. 8: l'Entrelaceur S-aléatoire de longueur 18 et S = 3

2.5.13. Les entrelaceus pseudo-aléatoires pair-impair

Les entrelaceus pseudo-aléatoires pair-impair dépend d'une condition simple qu'il est les symboles avec leurs positions initiales paires (respectivement impaires) seront entrelacés aléatoirement en sortie dans des positions paires (respectivement impaires). [5]

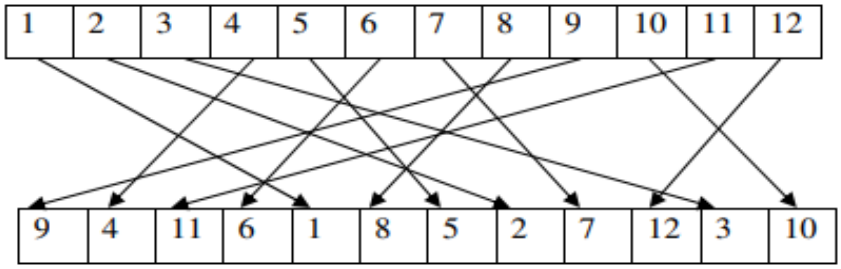


Figure 2. 9:Entrelacement pseudo-aléatoire pair-impair

2.5.14. Enterelaceur pseudo-aléatoire pair-impair symétrique

L'idée réside dans d'ajouter une contrainte supplémentaire à l'entrelaceur pseudo-aléatoire pair-impair. En d'autres termes, elle consiste à introduire le critère de symétrie, où permet d'échanger les positions, que ce soit paires ou impaires.de deux symboles entre eux. [2]

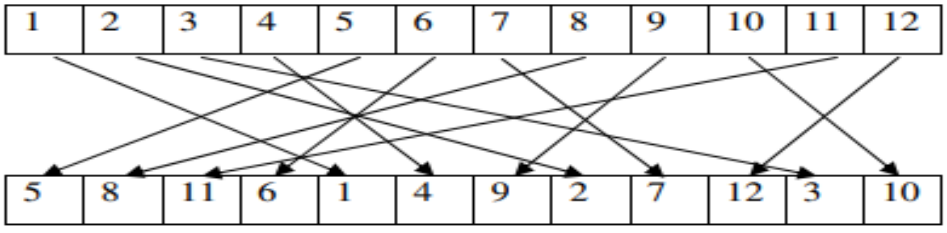


Figure 2. 10:l'Entrelacement pseudo-aléatoire pair-impair symétrique

2.5. Avantage d'utilisation des entrelaceurs

Dans la théorie des codes correcteurs d'erreurs, plusieurs propriétés associées avec les mots de code ont été définies.

Les deux propriétés importantes sont la distance de Hamming et le poids de Hamming.

- La distance de Hamming entre deux mots codés est le nombre total de positions, où les deux mots codes sont différents.
- Le poids de Hamming d'un mot de code est défini comme étant le nombre total des positions par lesquelles il est différent par rapport au vecteur nul.

L'une des propriétés qui définissent la qualité d'un code, est appelée la distance minimale d_{min} elle est définie comme la plus petite distance entre les mots de code distincts, et donne une mesure de la qualité de code pour détecter et corriger les erreurs. Un code à distance minimale $\mathbf{d_{min}}$ peut détecter jusqu'à $\mathbf{(d_{min} - 1)}$ erreurs et peut corriger Jusqu'à $\boldsymbol{t}$ erreurs, $\boldsymbol{t}$ vérifie la relation suivante :

$$d_{min} \geq 2t + 1 \text{ où } t = floor\left(\frac{d_{min}-1}{2}\right) \qquad (2.36)$$

Considérons un code qui peut corriger jusqu'à, $\boldsymbol{t}$ erreurs. Nous prenons un exemple de transmission de données simplifiée sur un canal bruyant comme le montre la **figure 2.11**

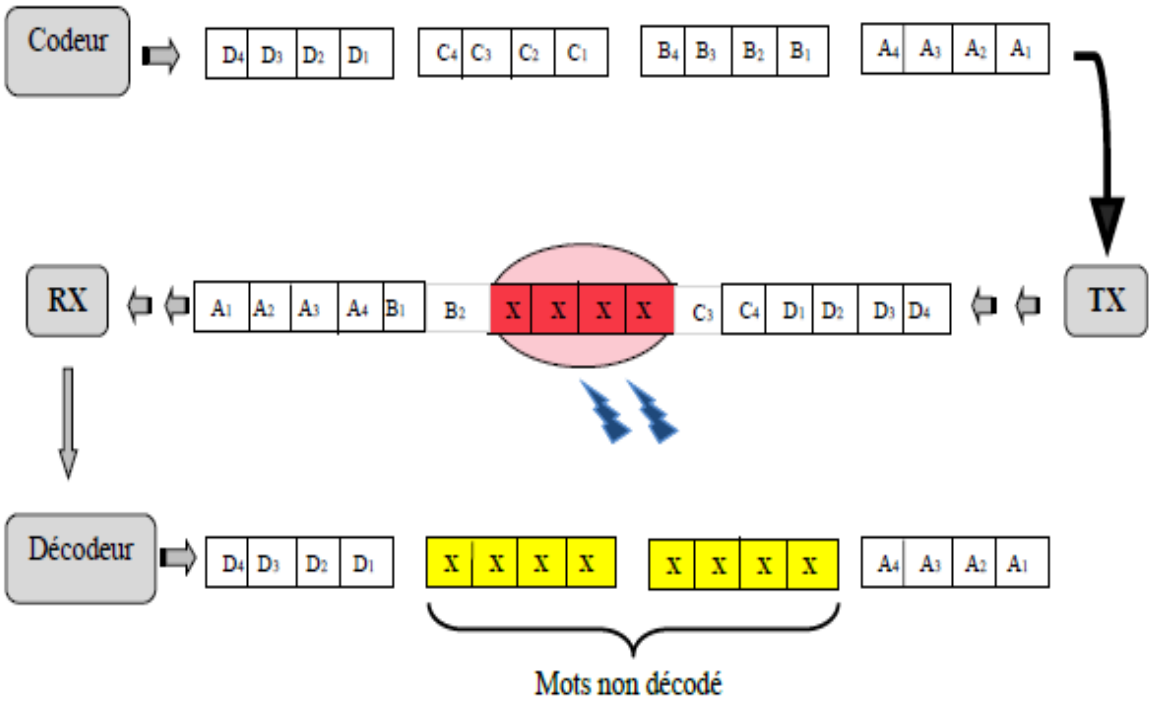


Figure 2. 11:transmission et de réception sur un canal bruité

Nous voyons que certaines données sont perdues lors de la transmission parce que deux erreurs apparus dans ces mots de code. Comme le décodeur peut corriger une seule erreur dans

chaque mot de code ainsi ces deux mots de code ont fini comme non décodé. Une solution pour améliorer la fiabilité de transmission est d'augmenter la puissance de transmission, mais cette solution est couteuse et difficile à réaliser dans le domaine spatial.

Une autre façon pour améliorer la fiabilité de transmission contre le bruit de canal est d'introduire un entrelaceur entre le codeur et l'émetteur. Vice versa un désentrelaceur doit être incorporé entre le récepteur et le décodeur comme représenté sur la **figure 2.11** Les bits transmis sont les mêmes, mais commandés d'une manière spéciale. Cette réorganisation des bits est appelée permutation des données.

Considérant même effet du bruit, comme dans **la figure 2.12**, les données reçues sont corrompues sous la forme d'un paquet d'erreurs.

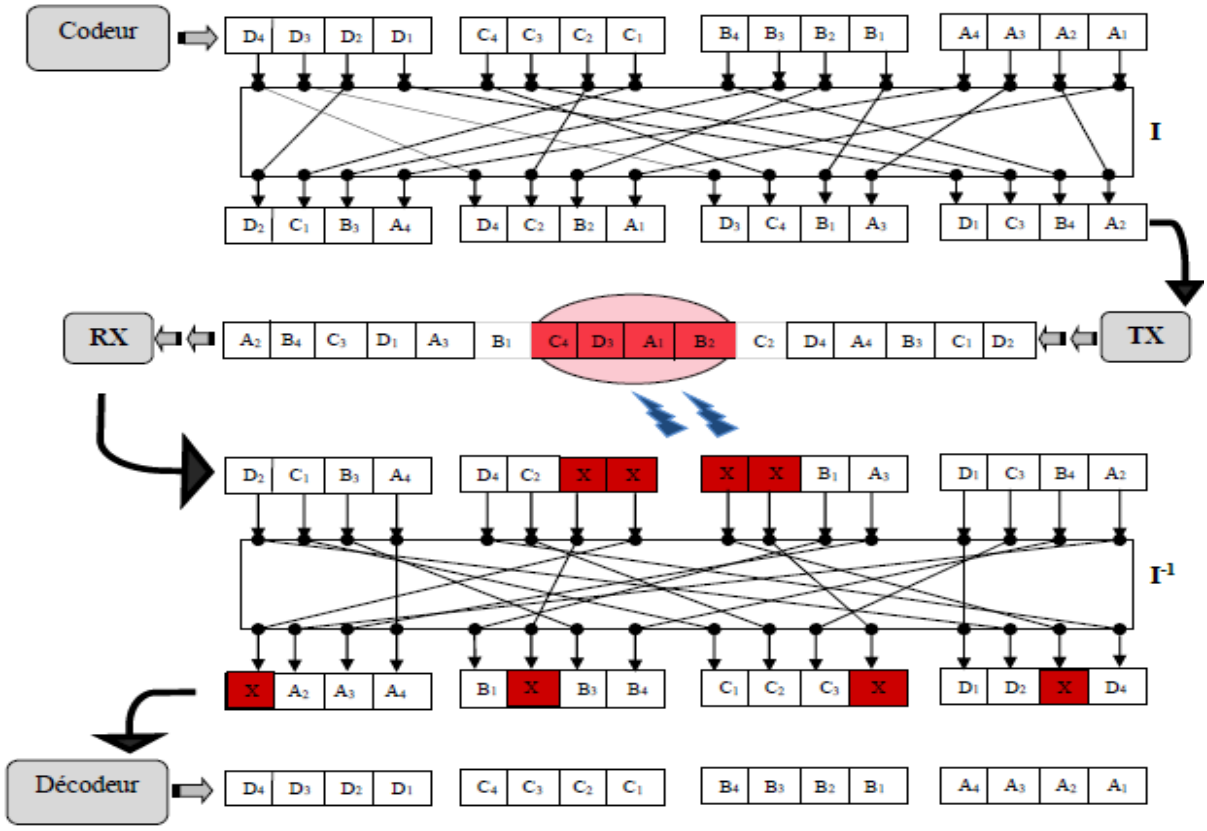


Figure 2. 12:transmission et de réception sur un canal bruité avec entrelaceur et désentrelaceur

Dans ce cas, au lieu de fournir directement les données reçues vers le décodeur, il est tout d'abord transmis au désentrelaceur. Le réarrangement effectué par le désentrelaceur donne l'avantage que chacun des mots de code au maximum une erreur. Puisque le décodeur est capable de corriger un mot de code avec une erreur, par conséquent, tous les mots de code sont récupérés et décodés correctement. L'entrelaceur fait le réarrangement de chaque mot de code sur un domaine spatial plus large qui donne une robustesse au même code contre les erreurs fatales induites par le bruit sur le canal de transmission. [1]

2.6. Conclusion

L'entrelacement est un élément important dans les performances de turbo code, utilisé pour lutter efficacement contre l'occurrence d'éclat d'erreurs (burst errors), Il vise à changer l'ordre des informations et non à modifier les informations elles-mêmes, La performance d'un turbo code est essentiellement améliorée quand la longueur d'entrelaceur augmente. Il existe différentes formes d'entrelaceurs, Dans ce chapitre nous nous sommes intéressés à l'objectif d'entrelacement, ses différentes techniques, et le principe de chacune de ces techniques.

L'entrelaceur de section d'or pour le turbo code est mis en œuvre et les performances sont comparées à l'entrelaceur de blocs pour différentes tailles de blocs et différents rapports signal / bruit. Les résultats en taux d'erreur sur les bits s'avèrent plus performants en cas de section d'or. Et les performances s'améliorent avec l'augmentation du SNR ainsi que l'augmentation de la taille des blocs des paquets. Dans le troisième chapitre, nous allons comparer chacun de ces entrelacs en termes d'efficacité et de performance.

Chapitre 3 : les résultats et simulations

3.1. Introduction

Le taux d'erreur binaire (BER) est un paramètre d'une très grande importance dans une chaîne de transmission numérique. Afin d'assurer une transmission fiable, le BER doit rester en dessous d'un seuil donné qui est fonction du système utilisé.

Le BER en fonction du Eb/N0 nous donne le gain du code, c.-à-d., le gain en puissance du code.

Pour une meilleure approximation du BER, une simulation de la chaîne de transmission s'avère nécessaire, nous avons effectué une comparaison des performances entre les entrelaceurs DRP, DGI, ARP, aléatoire, et RI

3.2. Interprétation du modèle de simulation

La description standard du système est le schéma de principe, où chaque bloc représente une opération de traitement du signal. Les paramètres standard WCDMA sont utilisés pour cette enquête et les bits de sortie du codeur turbo sont ensuite modulés à l'aide d'un modulateur BPSK (Binary Phase Shift Keying). [18]

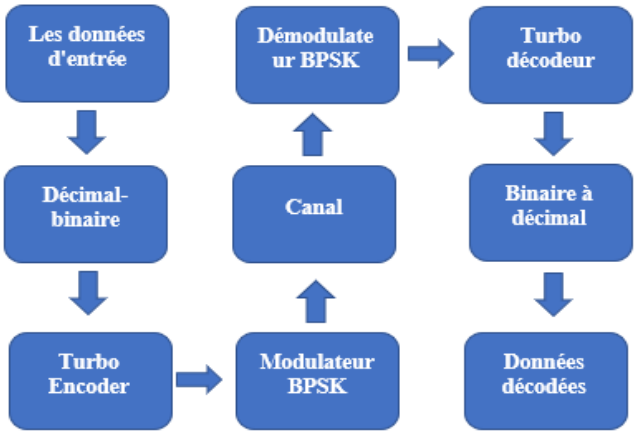


Figure 3. 1 : Schéma fonctionnel du modèle de simulation

Les étapes suivantes :

1. **Les données d'entrée :** On utilise des valeurs aléatoires pour la taille du données (trame). Dans cette simulation, on utilise **K = 400 bits** et **K = 1024 bits** pour les valeurs de taille de trame. Également ont utilisé le polynôme générateur **G = [15,13]$_8$**

La ligne ci-dessous est utilisée dans le code source pour générer des valeurs aléatoires :

$$x = round\big(rand(1, L_{total} - m)\big)$$

2. Décimal-binaire

Les données d'entrée dans le codeur et le modulateur sont sous forme binaire. Ainsi, la fonction de ce module est de transformer la source de la valeur décimale en valeur binaire. [14]

3. Turbo Encoder

Tout comme ce que nous avons vu dans le premier chapitre, le turbo encodeur est composé de de deux encodeurs RSC identiques. Un entrelaceur aléatoire sépare ces deux codeurs et il s'agit d'une permutation de l'ordre des bits dans un flux binaire.

Dans cette simulation, on utilise essentiellement un taux de code de 1/2 (perforé).

La figure 3.2 illustre l'organigramme du turbo encodeur dans ce modèle de simulation.

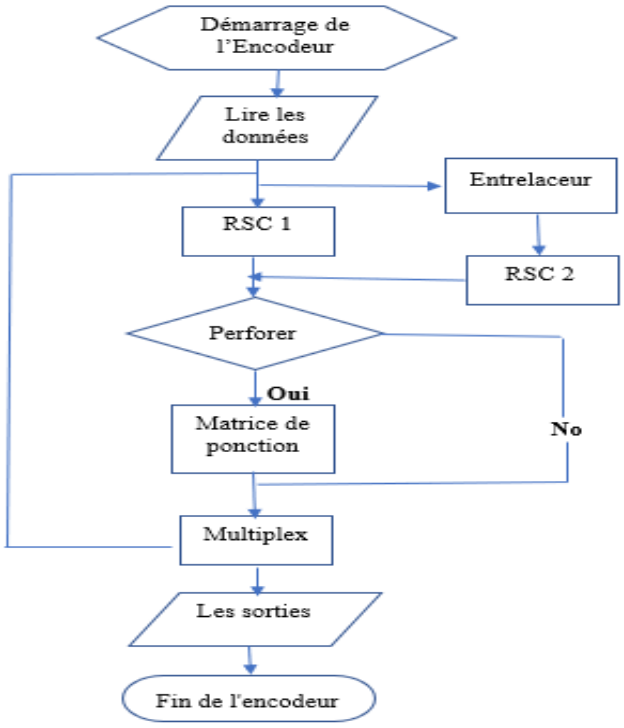


Figure 3. 2: L'organigramme de l'encodeur

4. Canal

Dans cette simulation, le modèle de canal (AWGN) est utilisé car il est utile pour simuler le comportement sous-jacent réalisé d'un système, et le canal AWGN est également une bonne approximation pour de nombreuses liaisons de communication par satellite et dans l'espace lointain. Le bruit de Gaussion est facile à construire à partir de la distribution de Gaussion avec une moyenne de zéro et un écart type d'un. [18]

5. Turbo décodeur

Le système de décodeur de code turbo décodeur possède plusieurs types, notamment : soft input soft output (SOVA) et the log maximum a posterior algorithm (log-MAP), et MAX-Log-MAP. Dans ces simulations, on utilise l'algorithme log-Map pour enquêter. L'organigramme de la simulation du turbo décodeur est illustré à la figure 3.3.

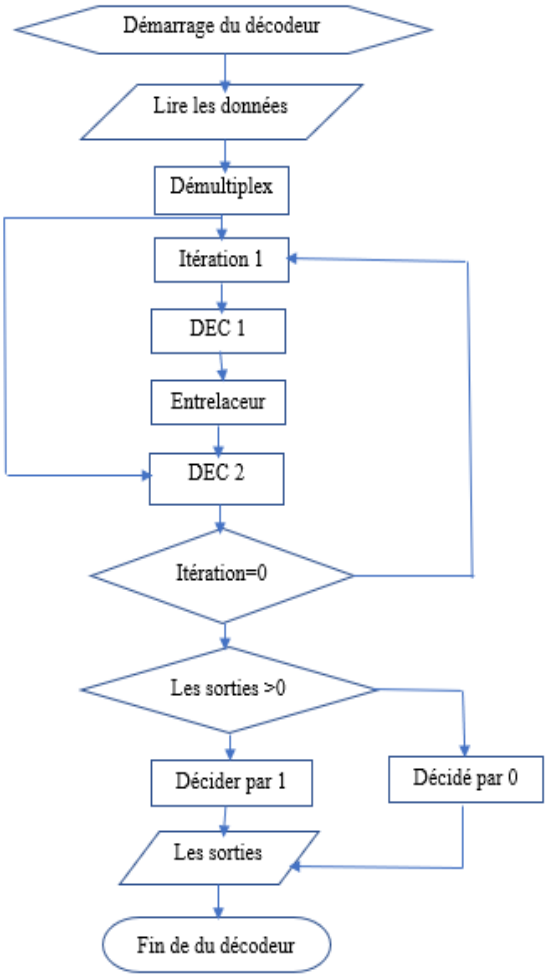


Figure 3. 3:L'organigramme du turbo décodeur

6. Binaire en décimal

La sortie du décodeur est binaire mais le système la lit comme une valeur décimale, il doit donc transformer le nombre binaire en nombre décimal et le récupérer à l'entrée de l'encodeur. **[18]**

3.3. Les étapes de travail

- La simulation suit la procédure suivante :

1. Générer aléatoirement les bits d'information.

2. Encoder les bits d'information par le codage turbo, à chaque étape de simulation nous changeons le type d'entrelaceur utilisé pour construire le turbo-code.
3. Utilisez la modulation BPSK pour convertir la donnée à la sortie de code turbo, en signaux complexes.
4. Introduire un bruit pour simuler les erreurs sur le canal de transmission. En première tempe nous supposons que les signaux sont transmis sur un canal AWGN ensuite en ajoutant l'effet multi trajets.
5. A la réception, les opérations inverses sont effectuées pour démoduler et décoder la séquence des symboles reçues.
6. Comptez le nombre des bits erronés en comparant la séquence des bits décodés avec la séquence des bits transmis.
7. Calculer le taux d'erreurs binaire (BER) et frame erreur bit (FER).

- Voici la liste des fonctions utilisées pour appeler le programme principal :

- **bin_state :** pour convertir un entier en un vecteur de bits binaires.
- **Démultiplex :** pour obtenir le mot de code de chaque encodeur au niveau du récepteur.
- **encode_bit :** Cette fonction prend en entrée un seul bit à encoder, ainsi que les coefficients des polynômes générateurs et du vecteur d'état courant. Il renvoie en sortie n bits de données codés, où 1 / n est le débit de code.
- **Fonction encoderm:** Cette fonction entrelace l'entrée du deuxième encodeur, si elle n'est pas percée, produit une sortie au taux 1/3 de longueur fixe, si elle est perforée, produit une sortie au taux 1/2. Le multiplexeur choisit les bits de contrôle impairs de RSC1, même les bits de contrôle de RSC2, et détermine la longueur de contrainte (K), la mémoire (m) et le nombre de bits d'information plus les bits de queue. Exécute également la modulation antipodale BPSK : + 1 / -1.
- **Fonction int_state:** Cette fonction convertit un vecteur ligne de m bits en un entier (base 10)
- **Rsc_encode.m**, code un bloc de données à l'aide d'un encodeur RSC, pour générer un mot de code.
- **logmapo:** pour calculer la sortie logicielle, le rapport log-vraisemblance des symboles dans la trame à l'aide de l'algorithme Log MAP.
- **Treillis** : Pour configurer le treillis pour un générateur de code donné, G (D)

Et enfin le programme principal pour commencer simulation :

• **Turbo sys .m** : turbo encodage et simulation de système de turbo décodage.

3.4. Les conditions et les paramètres de simulation

Les fichiers m de code Turbo développés dans ce projet étaient basés sur des travaux originaux de Yufer Wu (YuFei, 2005).

- La structure matlab Turbo Code utilisée par Wu est basée sur la structure décrite dans l'article original de Berrou. (Berrou, Glavieux et Thitimajshima, 1993). Cette simulation est basée sur le système classique de codes convolutionnels concaténés parallèles (PCCC).
- La machine utilisée est un PC (**logiciel MATLAB 2018**).

Nous étudierons les courbes de taux d'erreurs binaires (BER) et aussi Taux d'erreur de trame (FEE) en fonction du rapport signal à bruit Eb/N0 pour les turbo codes utilisant les entrelaceurs: DRP, DGI, ARP, aléatoire, et RI ; et nous avons fixé les paramètres suivant :

1. Le type de canal : AWGN.
2. L'algorithme de décodage : Log-MAP.
3. Le générateur de code : G= $(15, 13)_8$
4. Le taux du code R=1/2 (perforer).
5. Le nombre d'itérations pour chaque bloc : 5.
6. Le nombre des blocs erronés pour arrêter la simulation : 60.
7. Le nombre des bites erronés pour arrêter la simulation : 300.
8. Le nombre des itérations est fixé à 5 itérations et deux longueurs d'entrelacement sont considérées dans la simulation : K=20x20 bits, K = 32x32 bits.

- Pour le paramètre de taille de trame, nous utilisons les valeurs k = 400 bits et k = 1024bits, puis nous les comparons pour voir laquelle est la meilleure en termes de performances.

3.5. Les résultats de simulation

⇨ **K=400 (20x20) bits :**

Après avoir sélectionné la taille de trame **K = 400 bits**, nous avons obtenu les valeurs de taux d'erreurs binaires **(BER)** et Taux d'erreur de trame **(FER)** indiquées dans **le tableau 3.1**, qui étaient représentées en fonction d'**E_b/N_0** sur la **figure 3.4** et **figure 3.5.**

Tableau 3. 1:Les résultats de la simulation pour la taille de block K=400 bits

E_b/N_0		0db	0.5db	1db	1.5db	2db	2.5db
Aléatoire	Ber	1.3140e-01	5.4576e-02	1.4589e-02	5.7603e-03	1.5120e-04	1.8238e-05
	Fer	1.0000e+00	8.0000e-01	2.5000e-01	1.0853e-01	7.8383e-03	1.4726e-03
DRP (P=27, M=4)	Ber	1.2175e-01	8.0605e-02	1.7572e-02	3.5055e-03	1.6383e-04	4.3234e-06
	Fer	1.0000e+00	1.0000e+00	3.5714e-01	7.5000e-02	5.9328e-03	2.3936e-04
DGI (D=0.01)	Ber	1.1125e-01	5.2729e-02	1.3780e-02	1.4694e-03	1.4228e-04	4.5421e-06
	Fer	1.0000e+00	7.3333e-01	3.6111e-01	4.3651e-02	3.5303e-03	3.0819e-04
ARP (α=0,β=4, C=4, P=27)	Ber	1.0327e-01	5.1427e-02	1.8892e-02	1.8420e-03	3.4023e-04	6.6097e-06
	Fer	1.0000e+00	6.6667e-01	2.8571e-01	4.9096e-02	8.2418e-03	3.2462e-04
RI P=27	Ber	1.0579e-01	6.7590e-02	2.6952e-02	2.2416e-03	1.5461e-04	1.0248e-05
	Fer	1.0000e+00	7.5000e-01	5.0000e-01	5.8104e-02	4.9283e-03	2.8865e-04

BER

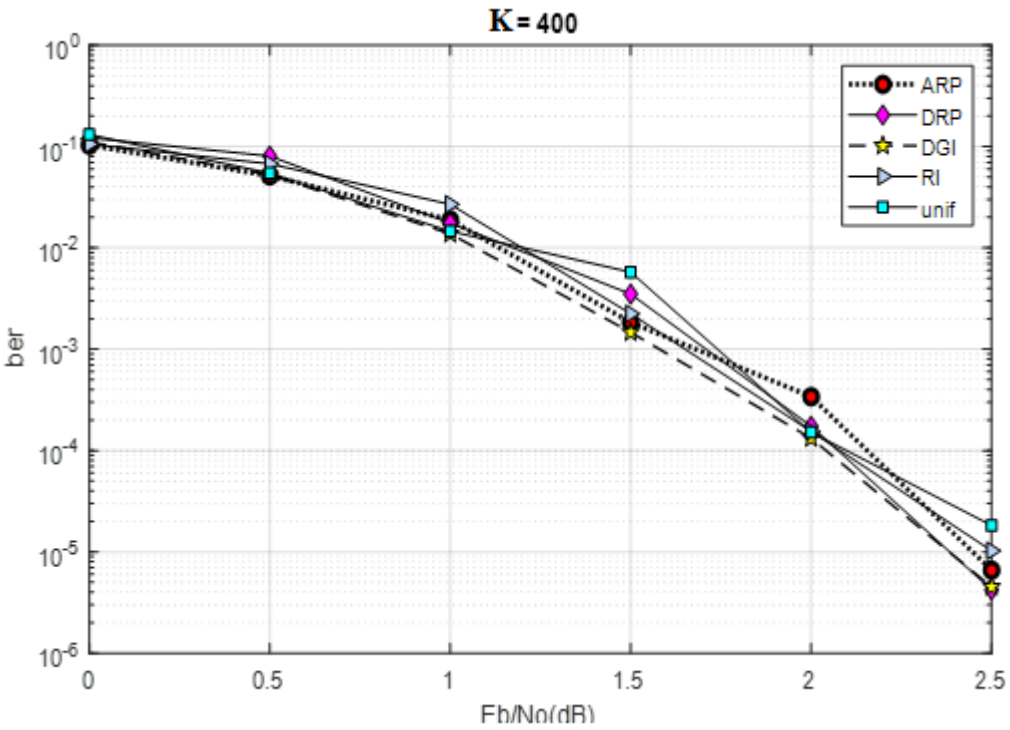


Figure 3. 4: le taux d'erreurs binaires (BER) en fonction de Eb/N0 pour K=400 bits.

La figure 3.4 montre les performances BER pour les entrelaceurs DRP, DGI, ARP, aléatoire, et RI dans un canal AWGN. Pour la longueur K = 400 bits, on remarque une différence de performance entre ces entrelaceurs. Pour un BER = 1.8238×10^{-05}, nous avons un E_b/N_0 = 2,5 dB pour l'entrelaceur Aléatoire et un E_b/N_0 = 2.334 dB pour l'entrelaceur DRP et DGI, et un E_b/N_0 = 2.417 dB pour le RI, un E_b/N_0 = 2.2384 dB pour le ARP. Les gains des codes entre chaque courbe sont : de0 dB la différence entre (DGI et DRP), 0.05 dB la différence entre (ARP et DGI) et (ARP et DRP),0.083 dB la différence entre (RI et DGI) et (RI et DRP), 0.166 dB la différence entre (Aléatoire et DGI) et (Aléatoire et DRP), 0.033 dB la différence entre (RI, ARP), 0.083 dB la différence entre (Aléatoire et RI) et 0.116 dB la différence entre (Aléatoire et ARP).

FER

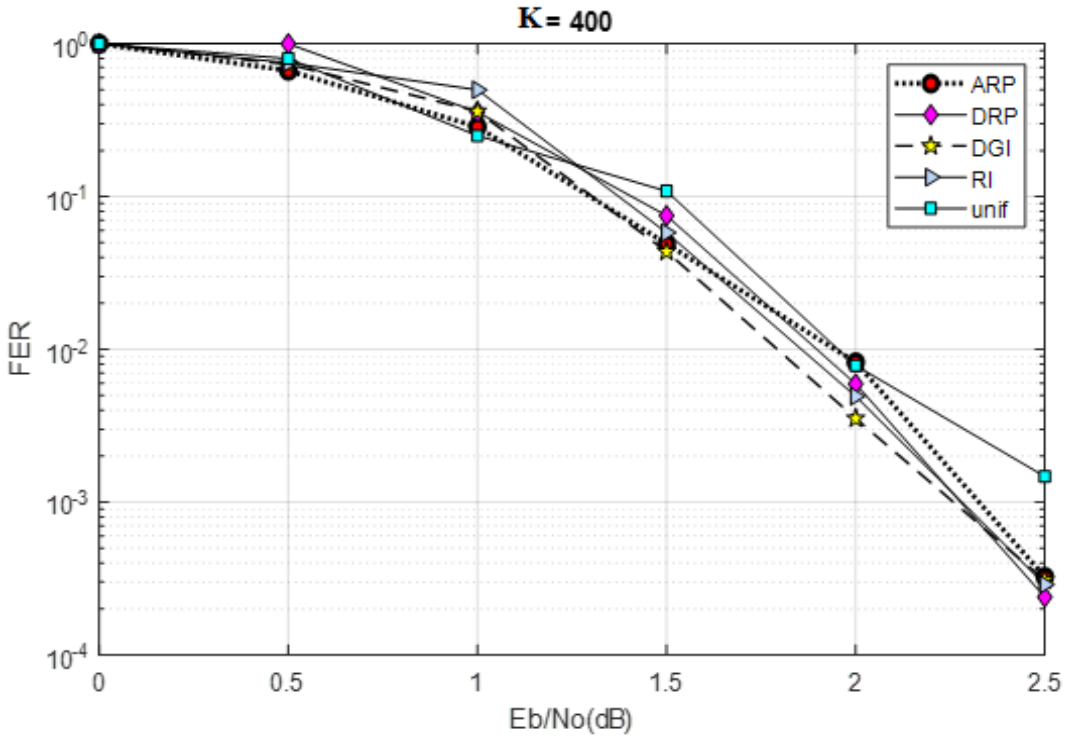


Figure 3. 5:Le taux d'erreur de trame (FEE) en fonction de Eb/N0 pour K=400 bits.

La figure 3.5 montre les performances FER pour les entrelaceur DRP, DGI, ARP, aléatoire, et RI dans un canal AWGN. Pour la longueur K = 400 bits, La performance est la même et presque identique lorsque vous utilisez des paquets de faible taille, Sauf qu'il y a une différence de performance pour l'entrelaceur l'aléatoire à point E_b/N_0= 2.5dB où FER=1.4726×10^{-03}. Cela nous montre une préférence relative pour les entrelaceurs : DGI, le DRP, ARP et RI sur l'entrelaceur aléatoire.

Nous avons également pour le point FER =1.4726×10^{-03}, un E_b/N_0 = 2.268 dB pour l'entrelaceur ARP et un E_b/N_0 =2.219 dB pour l'entrelaceur DRP et E_b/N_0 =2.202dB pour

l'entrelaceur DGI, et un E_b/N_0 = 2.235 dB pour le RI. Les gains des codes entre chaque courbe sont : de 0.017 dB la différence entre (DGI et DRP), 0.066 dB la différence entre (ARP et DGI) et 0.049 dB entre (ARP et DRP), 0.033 dB la différence entre (RI et DGI) et 0.016 dB entre (RI et DRP), 0.298 dB la différence entre (Aléatoire et DGI) et 0.281 dB entre (Aléatoire et DRP), 0.033 dB la différence entre (RI, ARP), 0.265 dB la différence entre (Aléatoire et RI) et 0.232 dB la différence entre (Aléatoire et ARP).

- **N=1024 (32x32) bits :**

Nous faisons la même étude précédente, mais cette fois, nous modifions la taille du cadre à **K= 1024 bits**, nous obtenons donc les résultats suivants :

Tableau 3. 2:Les résultats de la simulation pour la taille de block K=1024 bits

E_b/N_0		0db	0.5db	1db	1.5db	2db
Aléatoire	**BER**	1.1296e-01	8.0313e-02	1.0744e-02	2.4851e-04	1.9923e-05
	FER	1.0000e+00	1.0000e+00	3.6364e-01	2.1891e-02	1.8893e-03
DRP (P=45,M=8)	**BER**	1.2863e-01	8.1619e-02	9.5005e-03	1.2243e-04	1.6838e-06
	FER	1.0000e+00	1.0000e+00	4.3333e-01	1.2821e-02	3.5205e-04
DGI (D=0.01)	**BER**	1.1231e-01	6.3010e-02	1.4365e-02	1.3852e-04	2.8694e-06
	FER	1.0000e+00	1.0000e+00	2.8571e-01	2.1739e-02	4.0308e-04
ARP (α=0,β=8, C=8, P=45)	**BER**	1.1818e-01	6.4643e-02	6.4542e-03	2.3409e-04	7.9414e-06
	FER	1.0000e+00	1.0000e+00	2.8205e-01	1.8385e-02	1.2244e-03
RI P=45	**BER**	1.3190e-01	1.0121e-01	5.4631e-03	3.5216e-04	1.4531e-05
	FER	1.0000e+00	1.0000e+00	2.0000e-01	2.9963e-02	2.2479e-03

BER

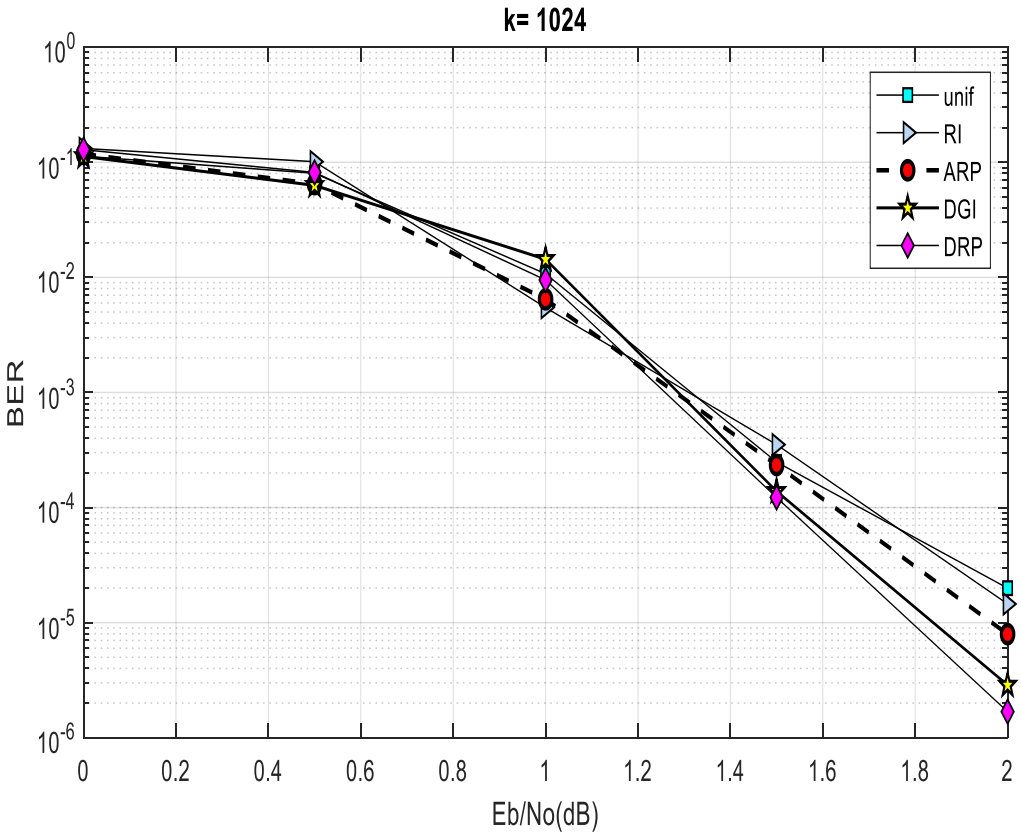


Figure 3. 6:le taux d'erreurs binaires (BER) en fonction de Eb/N0 pour K=1024 bits.

La figure 3.6 montre les performances BER pour les entrelaceur DRP, DGI, ARP, unif, et RI dans un canal AWGN. Pour la longueur K = 1024bit. Nous notons que les meilleures performances sont dans le cas de l'entrelaceur DRP et DGI par rapport à l'autre entrelaceur, Où au point BER $=1.9923 \times 10^{-05}$, un E_b/N_0 = 2 dB pour l'entrelaceur unif et un E_b/N_0 =1.732 dB pour l'entrelaceur DRP et E_b/N_0=1.764 dB pour l'entrelaceur DGI, E_b/N_0= 1,87 dB pour l'entrelaceur ARP, et un E_b/N_0= 1.974 dB, pour le RI. Les gains des codes entre chaque courbe sont : de 0.032 dB la différence entre (DGI et DRP), 0.236 dB la différence entre (unif et DGI) et 0.268 dB entre (unif et DRP), 0.21 dB la différence entre (RI et DGI) et 0.242 dB entre (RI et DRP), 0.106 dB la différence entre (ARP et DGI) et 0.138 dB entre (ARP et DRP), 0.026 dB la différence entre (RI, unif), 0.104 dB la différence entre (ARP et RI) et 0.13 dB la différence entre (ARP et unif).

FER

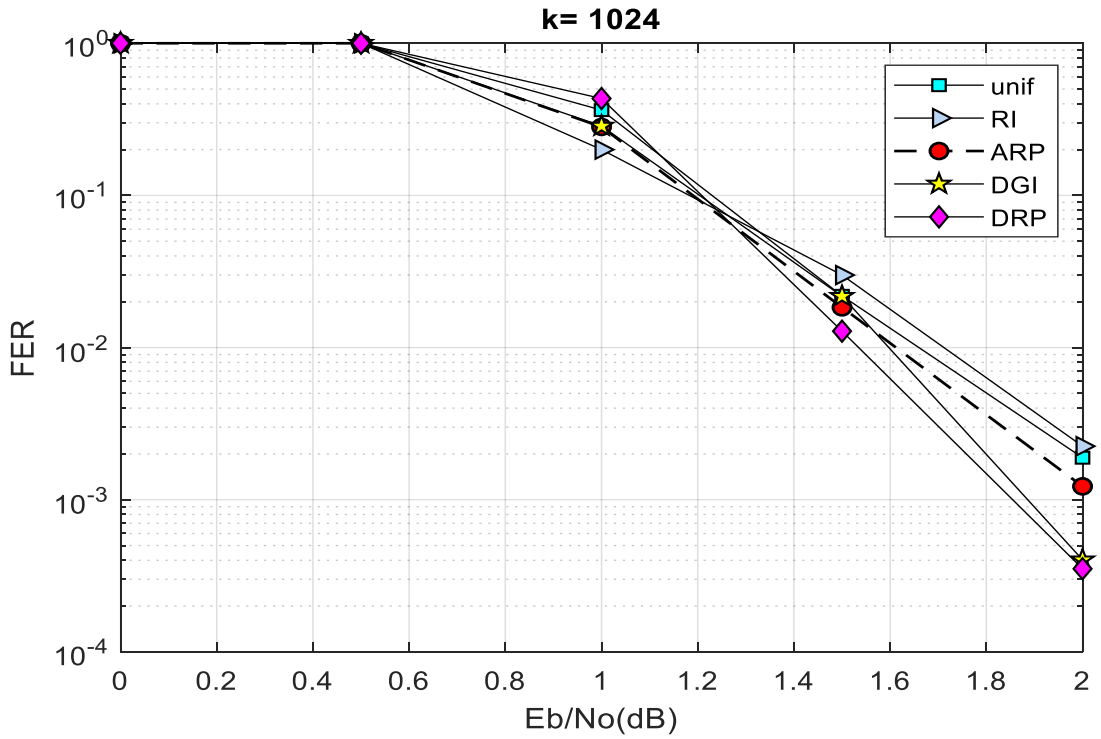


Figure 3. 7:Le taux d'erreur de trame (FEE) en fonction de Eb/N0 pour N=1024 bits.

La figure 3.7 montre les performances FER pour les entrelaceur DRP, DGI, ARP, aléatoire, et RI dans un canal AWGN. Pour la longueur K = 1024bit. Nous notons que les meilleures performances sont dans le cas de l'entrelaceur DRP et DGI par rapport à l'autre entrelaceur, Où au point FER = 2.2479e-03, un E_b/N_0 = 1.893 dB pour l'entrelaceur Aléatoire et un E_b/N_0 =1.736 dB pour l'entrelaceur DRP et E_b/N_0=1.747 dB pour l'entrelaceur DGI, et un E_b/N_0= 2 dB pour le RI et un E_b/N_0= 1.972 pour l'entrelaceur ARP. Les gains des codes entre chaque courbe sont : de 0.011 dB la différence entre (DGI et DRP), 0.146 dB la différence entre (Aléatoire et DGI) et 0.157 dB entre (Aléatoire et DRP), 0.253 dB la différence entre (RI et DGI) et 0.264 dB entre (RI et DRP), 0.225 dB la différence entre (ARP et DGI) et 0.236 dB entre (ARP et DRP), 0.107 dB la différence entre (RI, Aléatoire), 0.028 dB la différence entre (ARP et RI) et 0.079 dB la différence entre (Aléatoire et ARP).

La comparaison entre les différents types d'entrelaceurs en fonction du rapport signal sur bruit peut se faire à partir des courbes des figures **3.4 et 3.5, 3.6 et 3.7**. Ces figures présentent le **BER** et **FER** du système de codage obtenus par simulation des différentes entrelaceurs : **DRP DGI ARP RI, Aléatoire** sur un canal de **AWGN.** Le nombre des itérations est fixé à 5 itérations et deux longueurs d'entrelacement sont considérées dans la simulation : K=20x20 bits, K = 32x32 bits. Où, à travers cette étude, nous concluons qu'en termes de type l'entrelaceur, les entrelaceurs **DRP** et **DGI** sont les plus performants par rapport les autres entrelaceurs.

3.6. Les effets qui affectent les performances du turbo code

➔Effets du nombre d'itérations de décodage

Lorsque le nombre d'itérations augmente, les performances du turbo-code augmentent. Cependant, l'amélioration du BER ne sera pas significative dans certaines régions de E_b/N_0. Mais, le délai de traitement augmentera à mesure que l'itération augmentera. En effet, cette action augmentera la complexité et réduira l'efficacité du système. Dans cette étude, nous utiliserons le nombre 5 et 6 pour le nombre d'itérations et le générateur **$(7,5)_8$**

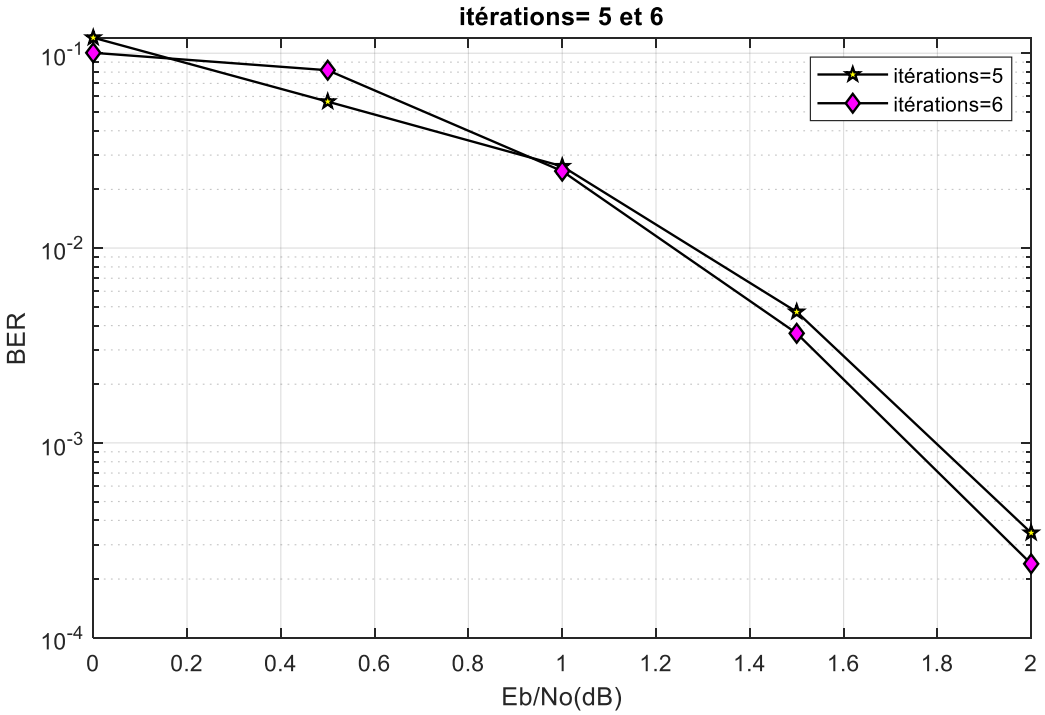


Figure 3. 8:Le taux d'erreurs binaires (BER) en fonction de E_b/N_0 en utilise l'entrelaceur DRP pour les nombres d'itérations 5 et 6.

La **figure 3.8** représente la différence entre les performances du turbo-code pour des entrelacs DRP en utilisant la taille de block K=400 bits et les nombres d'itérations 5 et 6 pour chaque bloc, Où nous notons que les performances sont meilleures lors de l'utilisation le nombre

d'itérations 6, Où l'on remarque qu'au point E_b/N_0= 1.5db, le ber =3.6501e-03 pour l'itération 6, et ber = 4.7030e-03 pour l'itération 5. Et aussi pour le point E_b/N_0 = 2db, où ber = 2.3952e-04 pour l'itération 6 et ber = 3.4526e-04 dans l'itération 5.

Nous avons pour le point BER = 3.4526e-04, un E_b/N_0= 2 dB pour l'itération 5 et E_b/N_0 =1.947db Pour l'itération 6, Le gain des codes entre les deux courbes est 0.053db

➔Effets du polynôme générateur

Parmi les méthodes qui d'améliorer les performances et facilitent l'utilisation de la tâche du décodeur, est utiliser un code contenant principalement des mots de code de poids élevé.

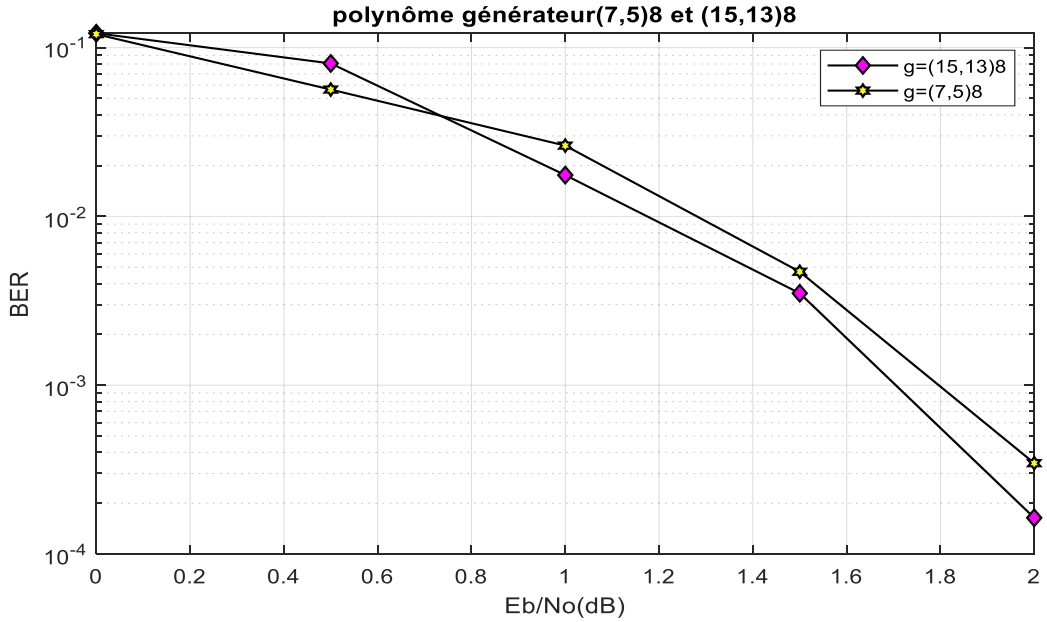


Figure 3. 9:Le taux d'erreurs binaires (BER) en fonction de E_b/N_0 en utilise l'entrelaceur DRP pour le polynôme générateur g= $(15,13)_8$ et g= $(7,5)_8$.

La figure 3.9 représente la différence entre les performances du turbo-code pour des entrelacs DRP en utilisant la taille de block K=400 bits et les polynômes générateur g =$(15,13)_8$ et g=$(7,5)_8$, Où nous notons que les performances sont meilleures lors de l'utilisation le polynôme générateur g=$(15,13)_8$, Où l'on remarque qu'au point E_b/N_0= 1db, le ber = 1.7572e-02 pour le polynôme générateur g=$(15,13)_8$, et ber = 2.6277e-02 pour le polynôme générateur g=$(7,5)_8$. Et aussi pour le point E_b/N_0= 1.5db, où le ber = 3.5055e-03 pour le polynôme générateur g=$(15,13)_8$ et ber = 4.7030e-03 pour le polynôme générateur g=$(7,5)_8$, et pour le point E_b/N_0= 2db, le ber = 1.6383e-04 pour le polynôme générateur g=$(15,13)_8$, et ber = 3.4526e-04 pour le polynôme générateur g=$(7,5)_8$

Nous avons pour le point BER = 3.4526e-04, un E_b/N_0= 2 dB pour le polynôme générateur g=$(15,13)_8$ et E_b/N_0=1.894 db Pour le polynôme générateur g=$(15,13)_8$, Le gain des codes entre les deux courbe est 0.106 db

➔Effets de taille de block

La taille de block affectera les performances du code turbo. Une grande taille de trame signifie que la plus grande distance entre chaque trame peut être entrelacée et que le décodeur peut obtenir de meilleures performances car la corrélation entre deux bits adjacents deviendra plus petite. Mais, l'augmentation de la taille de la trame augmente également le temps de traitement car le code turbo est un type de code de bloc, il doit attendre la fin du décodage de tout le bloc avant d'obtenir la sortie décodée. Pour cela, nous devons limiter la taille de la trame pour éviter un délai de traitement plus élevé en raison d'une plus grande quantité de données transmises. Dans cette simulation, nous utilisons **K = 400 bits et N = 1024 bits**, et nous pouvons voir que le turbo code avec une plus grande taille de trame à de meilleures performances.

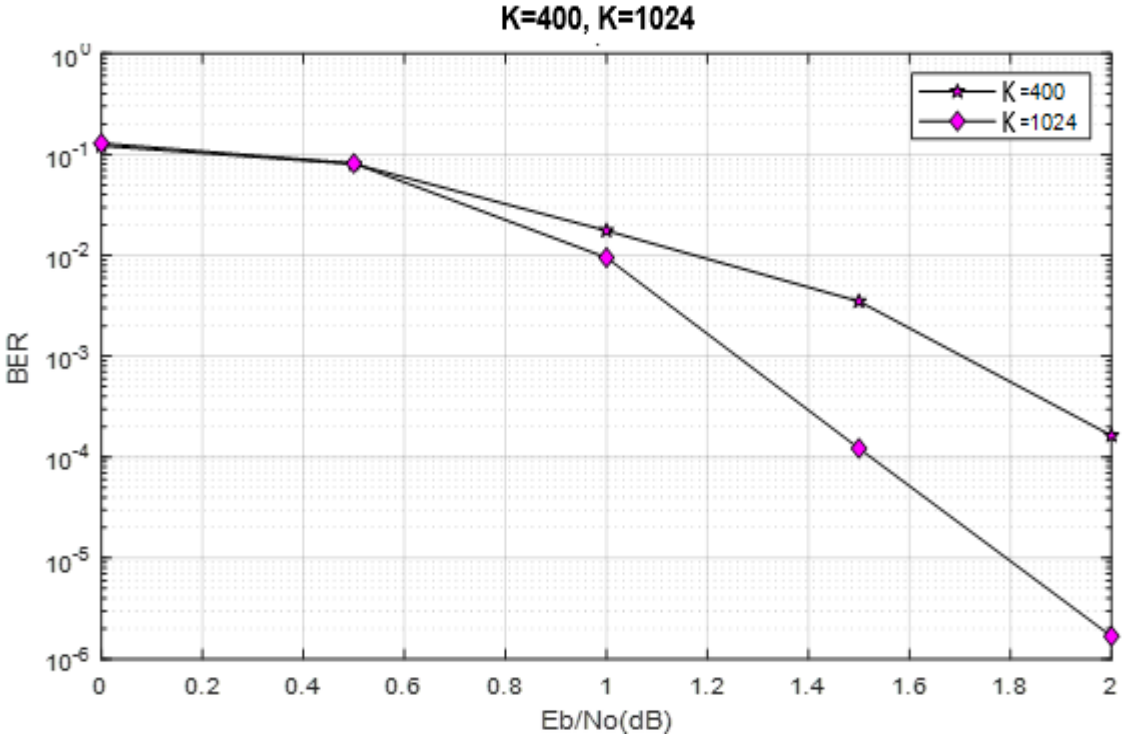


Figure 3. 10:le taux d'erreurs binaires (BER) en fonction de Eb/N0 en utilise l'entrelaceur DRP pour K=1024 bits et K=400 bits

La figure 3.10 représente la différence entre les performances du turbo-code pour des entrelacs DRP en utilisant les tailles de block K= 400 bits et k=1024 bits, où il est clairement montré que les meilleures performances sont à la taille de la trame 1024 bits.

Nous avons pour le point BER = 1.6383e-04, un E_b/N_0 = 2 dB pour la taille de block K= 400 bits et E_b/N_0=1.457 db les tailles de block K= 1024 bits, Le gain des codes entre les deux courbes est 0.543 db

3.6. Conclusion

Les entrelaceurs aléatoires parmi les entrelaceurs les plus efficaces pour les codes turbo. Cependant, en raison de leurs permutations aléatoires, une représentation compacte d'entrelaceur est complexe. Ainsi, à ce jour, de nombreuses recherches ont été menées sur la conception des entrelaceurs déterministes ayant des performances proches d'entrelaceurs aléatoires. Ces entrelaceurs sont principalement construits comme entrelaceurs bloc. Les techniques d'entrelacement pour les analyses TC ont été. À des fins de mise en œuvre, ces entrelaceurs présentent également une complexité réduite. Comme des capacités de parallélisation efficaces. La maximisation de la durée, la corrélation de la circonférence et le degré de désordre dans les symboles permutés constituant leurs principales exigences de performance. Cependant, la maximisation de ces paramètres ne conduit toujours à la meilleure performance. En conséquence, de nouveaux critères de conception sont proposés au chapitre 2. De plus, il a été identifié que les entrelaceurs DRP, DGI et ARP ont des propriétés similaires. Tout entrelaceur est caractérisé par deux propriétés : la propriété d'étalement, donc la distance en position entre l'index adjacent dans la trame d'information avant la permutation (entrelacement), et la propriété aléatoire qui fournit une fonction d'indexation non fixe, qui est un bon facteur de correction de codage itératif.

Il existe plusieurs facteurs qui affectent les performances de la pale de turbo, y compris le type d'entrelaceur. Nous avons vu que les entrelaceurs DRP et DGI sont meilleurs en termes de performances par rapport à ARP, RI et uniforme. Il y a aussi la longueur du bloc, qui joue un rôle important dans les performances de l'entrelaceur, plus la longueur de block est élevée, mieux c'est. Quant au polynôme du générateur, l'utilisation d'un code contenant principalement des mots de code de poids élevé améliore les performances et facilite l'utilisation de la tâche de décodeur.

Conclusion générale

L'objectif de ce travail est de faire une comparaison entre les différents types des entrelaceurs et de discuter leur effet sur les performances des turbo codes.

Cette mémoire contient deux parties : théorique et pratique ;

La partie théorique étudie les turbo codes et leur fonctionnement, et comment encoder et décoder et les algorithmes qui sont utilisés pour cela, et on a étudié les différentes techniques d'entrelacement et leurs fonctions et on a comparés, et vu les caractéristiques de chacun.

Pour la partie pratique, on a étudié les courbes de taux d'erreurs binaires (BER) en fonction du rapport signal à bruit Eb/N0 pour les entrelaceurs : DRP, DGI, ARP, unif, et RI ; En utilisant le programme Matlab, puis nous avons comparé ces interlocuteurs, qui sont meilleurs en termes de performances en gain de code (en puissance).

Théoriquement, les entrelaceurs aléatoires sont meilleurs que les autres car ils permettent d'obtenir une très grande distance de hamming par rapport aux autres, Cependant les résultats de la simulation montrent que DRP et DGI sont les plus performants par rapport aux autres entrelaceurs.

Nous avons également testé l'effet de certains facteurs qui affectent les performances du turbo-code, tels que le polynôme générateur, le nombre d'itérations de décodage et la taille de block.

Les bibliographiques :

[1]. IFTENE Essedik. "Maxime Cote. ETUDE DES STRUCTURES D'ENTRELECEURS POUR LE CODAGE TURBO DU CANAL POUR L'OPTIMISATION DES SYSTEMES DE COMMUNICATION PAR SATELLITE". Mémoire, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf. Année Universitaire : 2015/2016.

[2] Mlle. Khadidja SERIR. « Application des codes correcteurs d'erreurs Reed Muller ». Mémoire de fin d'études Master en Informatique. Université Abou Bakr Belkaid– Tlemcen. octobre 2011.

[3] Mohamed El Amine M'SIR. « Conception D'ARCHITECTURES RAPIDES POUR CODES CONVOLUTIFS EN Télécommunications : Application Aux TURBO-CODES » THÈSE DE DOCTEUR de l'Université de Metz. Le 19 Septembre 2003.

[4]. Claude, Berrou. Collection IRIS ISBN dirigée par nicolas puech. « Codes et turbocodes. » . École Nationale Supérieure des Télécommunications de Bretagne CS 83818. Springer-Verlag France 2007

[5] TERRAS, Dahou. GUERANDI Khedidja. « Entrelaceurs Turbo à base des permutations cryptographiques » Mémoire de Master. Option : Technologies de Communications. Université Dr. Tahar Moulay Saida. 29/10/2019

[6]. Thibaud Tonnellier. « Contribution à l'amélioration des performances de décodage des turbo codes : algorithmes et architecture ». THÈSE DE DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX École Doctorale des Sciences de l'Ingénieur Spécialité : Électronique. Le5 Juillet 2017

[7] Aroua Briki. « UNE NOUVELLE APPROCHE DE PLACEMENT DE DONNEES EN MEMOIRE : APPLICATION A LA CONCEPTION D'ARCHITECTURES D'ENTRELACEURS PARALLELES ». THESE DOCTORAT. UNIVERSITE DE BRETAGNE-SUD Ecole doctorale SICMA.14 Jan 2014.

[8] John Wiley & Sons Ltd « Turbo Coding, Turbo Equalisation and Space–Time Coding. EXIT-Chart-Aided Near-Capacity Designs for Wireless Channels ». The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom Publication 2011.

[9] MOSTARI LATIFA. Codage de Canal. Polycopié de Cours & Travaux Dirigés. UNIVERSITE HASIBA BEN BOUALI CHLEF FACULTE DE TECHNOLOGIE DEPARTEMENT D'ELECTRONIQUE. 2016/2017.

[10] Jason. Dolloff, Zackary Kenz†, Jacquelyn Rische, Danielle Ashley Rogers§, Laura Smith, Edward MosteigkPomona and Loyola Marymount « Algebraic Interleavers in Turbo Codes. ». Université polytechnique de l'État de Californie, Pomona et l'Université Loyola Marymount Rapport technique du Département de mathématiques, August 2006.

[11] Chris Heegard Alantro Communications, Inc. et Cornell University. Université Stephen B. Wicker Cornell, « TURBO CODING ». Copyright © 1999 par Springer Science + Business Media New York. Publié à l'origine par Kluwer Academic Publishers en 1999.

[12] Benjamin Moreno, Laura Smith†, Andrea Viteri, Kouadio David Yao. « Exploring Interleavers in Turbo Coding ». Department of Mathematics Technical Report. California State Polytechnic University, Pomona and Loyola Marymount University. August 2005

[13] Stewart Crozier and Paul Guinand**.. «** Distance Upper Bounds and True Minimum Distance Results for Turbo-Codes Designed with DRP Interleavers. » Communications Research Centre, 3701 Carling Ave., P.O. Box 11490, Station H, Ottawa Canada, K2H 8S2, 28/02/2020.

[14] Ajit Nimbalker, Yufei Blankenship, Brian Classon, T. Keith Blankenship « ARP and QPP interleavers for LTE turbo coding Wireless and Solutions » Research Motorola Labs 1301 E. Algonquin Road, MD 2928 Schaumburg, IL 60196 USA 12/01/2020

[15] Ronald Garzon Bohorquez, Charbel Abdel Nour, Catherine Douillard. "On the Equivalence of Interleavers for Turbo Codes". IEEE wireless communications letters, IEEE comsoc, 2015, 4 (1), pp.58 61. 10.1109/LWC.2014.2367517. hal-01170165. Le17 Feb 2020.

[16] Ronald Edicson Garzón Bohórquez « Advanced coding and interleaving techniques for next generation communication and broadcasting systems ». Thèse de Doctorat ’Université européenne 12/03/2020.

[17] M. AMAMRA Imed. « Codage Canal et Techniques Efficaces de Décodage Ttératif ». THÈSE DE DOCTORAT. Université 20 Aout 1955 de Skikda 18/4/2020.

[18] LOOI CHUN HUI. « Generator Polynomial in Turbo Code System". Un rapport de projet soumis dans le respect partiel des exigences pour l'octroi du baccalauréat en génie (avec spécialisation) génie électronique et de la communication. Faculté de génie et des sciences Universiti Tunku Abdul Rahman. April 2012 13/04/2020