الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA

FACULTE DE TECHNOLOGIE

Département du Génie Électrique



Mémoire de Fin d'Études

Présenté pour l'obtention du Diplôme deMaster en

Télécommunication

THEME

Etude et implementation d'un code correcteur d'erreur par turbo code

Proposé et dirigé par,

Mr. BAKIR TERKI OTHMAN

Préparé et présenté par,

BOUKHELKHAL MOHAMED YASSINE

BENDALI BRAHEM MERIEM

Remerciement :

D'abord, nous remercions Dieu le tout puissant et miséricordieux. C'est grâce à lui que nous avons eu le foie et la force pour accomplir ce travail.

Nous remercions vivement notre encadreur, Mr othman Bakir torkie, pour nous avoir proposé ce sujet d'actualité assez passionnant ; sa patience et son organisation nous ont permis de surmonter de nombreuses difficultés liées à ce travail.

A toute personne qui a contribué de près ou de loin, directement ou indirectement à l'accomplissement de ce travail.

Nos remerciements s'étendent également aux membres du jury pour l'intérêt qu'ils ont porté à notre mémoire en acceptant d'examiner notre travail.

Nos remerciements les plus vifs, vont également à nos parents qui nous ont donné un environnement idéal durant toute notre enfance et ils nous ont enseigné l'humilité et l'honnêteté avec lesquelles nous avant essaye de mener à terme ce travail scientifique.

Dédicace

Je dédie ce travail à :

A mes très chers parents, pour leurs patience, leurs soutien et Leurs confiance ; mon père BOUKHALKHAL NASR ELDINE qui s'est sacrifié afin que rien n'entrave le déroulement de mes études, ma mère ELGUELLA NADIA qui m'encourager, et me soutenir dans les moments difficiles ; qui a sa su m'entourer de tout son amour et son affectation pour que je ne puisse manquer de rien.

A ma chère femme et mon binôme BENDALI BRAHAM MERIEM qui ma soutenu avec leurs encouragement et leur amour.

A mon chère frère HOUSSEM, et mes chère sœur NESRINE et FARAH.

""A mon petit ange ""

A l'esprit de mon chère père BOUKHALKHAL KHALAD " Allah yarhmo".

A ma 2eme famille, ma belle mère YAMINA et mon beau père SALIME et mes belle sœur MICHA, mon beau frère IDRIS.

A toute ma famille BOUKHALKHAL et ELGUELLA.

A tout mes cousins et mes cousines

A tous ceux qui m'aiment, et ceux que j'aime.

Dédicace

Je dédie ce travail à :

A mes très chers parents, pour leurs patience, leurs soutien et Leurs confiance ; mon père BENDALI BRAHAM SALIME qui s'est sacrifié afin que rien n'entrave le déroulement de mes études, ma mère EL MOHRI AMINA qui m'encourager, et me soutenir dans les moments difficiles ; qui a sa su m'entourer de tout son amour et son affectation pour que je ne puisse manquer de rien.

A mon chère marie et mon binôme BOUKHALKHAL MOHAMED YASSINE qui ma soutenu avec leur amour, leur compréhension, et pour tout ce qu'il a fait pour moi.

A mon chère frère IDRIS et ma chère sœur MAISSA

"" A mon petit ange ""

A ma 2eme famille, ma belle mère NADIA et mon beau père NASR ELDINNE et mes belle sœur FARAH et NESRINE, mon beau frère HOUSSEM.

A toute mes cousins et mes cousines surtout SELMA et la petite IHCEN.

A toute ma famille BENDALI et EL MOHRI.

A tous ceux qui m'aiment, et ceux que j'aime.
A l'esprit de ma grand mère KHDAWEJ "Allah yerhamha"

Résumé :
Dans ce travail, nous avons proposé une programmation de la performance de turbo code pour corriger les erreurs dans d'information binaire transmise sur le canal AWGN où le but est de la programmer en utilisant le logiciel de calcule Matlab.
Pour arriver à ce but, nous avons suivi trois étapes principales représentées dans le codage en utilisant deux codeurs identiques puis l'entrelacement des bits d'information pour le deuxième et à la fin le décodage, où déduisent la complexité de ce dernier d'une part et la durée de fonctionnement un peu long, d'autre part.
Mots clés :
Bruit blanc gaussien AWGN, codage, entrelacement, décodage, Code turbo, concaténation parallèle, codes convolutifs, code systématique récursif, décodeur itératif, algorithme MAP, TEB, modulation BPSK, SNR.
Abstract:
In this work, we proposed programming the performance of turbo code to correct the errors in binary information transmitted by the AWGN channel where the goal is to program using the Matlab software calculates.
To get this goal, we followed three main steps shown in coding using two identical encoders and interleaving the information bits for the second and at the end where the decoding, we deduce the complexity of the last on the one hand and a little long term operation on the other hand.
Keywords:
Gaussian white noise AWGN, coding, interleaving, decoding, turbo code, convolution code, Systematic Convolutional code, decoder iterative, MAP Algorithm, BER, modulation BPSK, SNR.

Table des matières

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SYMBOLES ET ABREVIATION

Introduction générale 1

Chapitre I : Systèmes de communication numériques.

I.1. Introduction 3

I.2. le système de communication numérique classique 3

I.2.1. le codage de source …….4

I.2.2. le codage de canal ……..4

I.2.3. modulateur …….4

la modulation BPSK..……4

I.2.4. canal de transmission ……6

I.2.4.1. canal symétrique ……6

I.2.4.2. canal à bruit blanc gaussien …….7

I.2.5. partie réceptrice…….8

I.3. conclusion …….9

Chapitre II : Concaténation parallèle des codes CRS.

II.1. introduction ……..10

II. 2. Principe de base de codage convolutionnel 10

II.3. type de codeur convolutionnel ……12

II.3.1. code convolutionnel systématique ……...12

II.3.2. code convolutionnel récursif systématique ……12

II.4. représentation graphique des codes convolutionnels ……...16

II.4.1. diagramme d'états ……16

II.4.2. treillis d'encodage ……17

II.5. l'entrelacement ……..…18

II.5.1. l'avantage d'utilisé l'entrelaceur ……18

II.5.2.principe des entrelaceurs ……..20

II.5.3. Entrelaceurs de type bloc ……..……22

II.5.3.1. Entrelaceurs en matrice ……22

II.5.3.2. Entrelaceurs hélicoïdal ……..23

II.5.3.3. Entrelaceurs aléatoire ……...…….23
II.5.3.3.1. Entrelaceurs pseudo-aléatoire……...…….23
II.5.3.3.2. Entrelaceurs pseudo-aléatoire (pair-impair) ……24
II.6. concaténation parallèle des codes CRS ……..24
II.7. l'encodeur turbo ……...25
II.8. conclusion ……..26
Chapitre III : Décodage itératif des codes concaténés
III.1. Introduction 27
III.2. principe de maximum à postériori …….....27
III.3. probabilité conjointe …….....30
III.4. l'algorithme MAP utilisé ……...31
III.4.1. expression de la Métrique d'état en avant…….....31
III.4.2. expression de la Métrique d'état en arrière…….....32
III.4.3. expression de la Métrique d'état de branche ……..33
III.5. décodage itératif ……33
III.5.1.principe de décodage itératif …….….33
III.6. conclusion ……...35
Chapitre IV : Programmation et discussion des résultats
IV.1. Introduction.....36
IV.2. Principe de turbo code …….…….36
IV.3 Le codeur Turbo ……36
IV.4. Transmission de symboles modulés par BPSK par canal AWGN……...38
IV.5. Décodeur turbo ……40
IV.6. Conception d'algorithme ……41
IV.7. résultats de simulation ……..43
IV.8. conclusion ……..47
Conclusion générale …….…..48
Bibliographie …….….49

Liste des figures :

Figure I.1 : modèle de système de transmission numérique……..……3
Figure I.2 : modulateur BPSK…….5
Figure I.3 : modèle du canal bruit blanc gaussien additif……...……5
Figure I.4 : canal symétrique binaire de probabilité d'erreur Pe……..…….6
Figure I.5 : canal additif à bruit blanc gaussien…….7
Figure II.1 : principe d'un codeur convolutionnel……10
Figure II.2 : codeur convolutionnel de taux de codage R=1 /2……..……..11
Figure II.3 : autre représentation du code R=1/2, K=3, $G_1 = 5\,, G_2 = 7.$ ……13
Figure II.4 : représentation poly nominale du code R=1/2, K=3, $G_1 = 5\,, G_2 = 7.$ ……13
Figure II.5 : structure d'un codeur récursif systématique K=3, R=1/2……..15
Figure II.6 : Structure du deuxième codeur récursif systématique (K=3, R=1/2)……...16
Figure II.7 : Diagramme d'état du codeur de la figure II.2 (K=3. R=1/2)…….17
Figure II.8 : Représentation en treillis du code convolutionnel de la figure II.2…….18
Figure II.9 : modèle simplifie de transmission et de réception sur le canal bruité……..19
Figure II.10 : modèle simplifie de transmission et de réception sur le canal bruité avec Entrelaceurs et désentrelaceur……20
Figure II.11 : principe d'entrelacement…….21
Figure II.12 : vecteur de permutation d'un entrelaceur……22
Figure II.13 : Entrelaceurs en matrice de taille 3*3……..22
Figure II.14 : Entrelaceurs hélicoïdal……23
Figure II.15 : Entrelaceur pseudo-aléatoire pair-impair……..24
Figure II.16 : schéma de concaténation parallèle des codes CRS……25
Figure II.17 : schéma de principe d'un codeur turbo……26
Figure III.1 : principe de décodage itératif……34
Figure IV.1 : architecture d'un turbo encodeur pour code $G(D) = [1\,\frac{1+D+D^2}{1+D^2}]$……...37
Figure IV.2 : diagramme en state……..38
Figure IV.3 diagramme en treillis…….38
Figure IV.4 : principe de codage turbo…….39
Figure IV.5 : architecture d'un décodeur BJCR……..40
Figure VI.6 : principe de décodage turbo…….……...42
Figure VI.7 : TEB après 1^{er} itération…….43
Figure VI.8 : TEB après 2^{eme} itération……44

Figure VI.9 : TEB après 3eme itération……44
Figure VI.10 : TEB après 4eme itération……..45
Figure VI.11 : comparaison de TEB avec la limite théorique……46

Liste des tableaux :

Tableau IV.1 : Valeurs de TEB pour 4^{eme} itération et comparaison avec TEB théorique du code convolutionnel correspondant (Certaines valeurs sont supérieures à 1, à cause de bande théorique) ……47

Liste des Symboles et abréviations :

K : Longueur de contrainte d'un code convolutionnel.

M : Mémoire d'un code convolutionnel.

V : Nombre des additionneurs modulo 2 d'un code convolutionnel.

R : Taux de codage d'un code convolutionnel.

N : Taille d'une séquence d'information binaire.

P : Matrice de perforation

I : Entrelaceurs

$\mathbf{I}^{-}$: Désentrelaceur

$\boldsymbol{E_b}$: Energie du signal par bit.

$\boldsymbol{N_0}$: Densité spectrale de puissance de bruit.

$\boldsymbol{E_b/N_0}$: Rapport signal sur bruit.

P(B) : Probabilité d'erreur par bit.

$\boldsymbol{\sigma^2}$: Variance du bruit dans le canal.

B : Bande passante.

C : Capacité de canal.

$\boldsymbol{R_{gp}}$: Taux de codage pour des codeurs montés en parallèle.

$\boldsymbol{R_1}$: Taux de codage de code convolutionnel 1.

$\boldsymbol{R_2}$: Taux de codage de code convolutionnel 2.

$\boldsymbol{R_k = (r_k^s, r_k^p)}$: La version bruitée reçue.

$\boldsymbol{Y_k^j}$ Symbole systématique à la sortie de l'encodeur turbo.

$\boldsymbol{Y_k^{1p}}$ Premier symbole de parité à la sortie de l'encodeur turbo.

$\boldsymbol{Y_k^{2p}}$ Deuxième symbole de parité à la sortie de l'encodeur turbo.

APP: A posteriori Probabilité.

BABG: Bruit blanc Gaussien additive.

BCJR: Bahl, Cock, Jelinec, Raviv.

BER: Bit Error Rate.

BPSK: Binary Phase Shift Keying.

CNS : Code Non Systématique.

CRS : Code Récursif Systématique.

MAP: Maximum A posteriori.

NRNSC : Non Récursif Non Systématique Code.

SNR : Rapport Signal sur Bruit.

LLR : Logarithme du rapport de vraisemblance.

SOVA : Soft Output Viterbi Algorithme.

TEB : Taux d'Erreurs Binaire.

FSM : Métrique d'état en avant.

BSM : Métrique d'état en arrière.

BM : Métrique de branche.

$\boldsymbol{a_k}$: Enveloppe du signal Rayleigh.

$\boldsymbol{\gamma_k}$ Métrique de branche de l'état à m l'instant k.

$\boldsymbol{\alpha_k}$ Métrique d'état en avant de l'état m à l'instant k.

$\boldsymbol{\beta_k}$Métrique d'état en arrière de l'état m à l'instant k.

$\boldsymbol{\lambda_k^i}$ Probabilité conjointe.

Introduction générale :

Dans tout système de communication numérique, on cherche à transmettre l'information provenant d'une source vers un récepteur à travers un canal de transmission. Le terme numérique est relatif à la séquence des symboles qu'il utilise pour représenter l'information. En général, pour les communications numériques l'alphabet utilisé est binaire. Ce genre de transmission des données est très intéressant dans le sens où il permet l'utilisation de nombreuses techniques de manipulation de l'information [1]. Parmi celles-ci, nous pouvons penser à la compression de l'information, au cryptage de cette dernière et aussi aux codes correcteurs d'erreurs. Ce sont ces derniers qui font l'objet de beaucoup de recherches actuellement. Les perturbations intervenant sur le canal de transmission induisent des erreurs de transmission que le codage de canal s'efforce de combattre. L'objectif est alors 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.

La théorie de l'information fondée par Glaude E. Shannon en 1918 fournit un ensemble de développements théoriques qui permettent l'évaluation de la capacité des canaux de communication [3]. Celle-ci assure qu'on peut trouver une probabilité d'erreur aussi petite que l'on veut, si l'on emploie un code correcteur d'erreur approprié et si le taux de transmission du code est inférieur à la capacité du canal. En 1993, Berrou, Glavieux et Thitimajshima inventent les codes turbo [2] qui approchent les limites données par la théorie de l'information.

Les codes correcteurs d'erreurs et le domaine spatial qui a permis le développement de ces codes correcteurs d'erreur. En effet, les premières utilisations de ceux-ci en communication devaient permettre de surmonter la perte de puissance lors de la propagation entre le transmetteur de l'engin spatial et le récepteur de la station terrestre [4]. Les engins envoyés dans l'espace, à l'origine se devaient d'être assez petits pour des raisons économiques, mais aussi des raisons d'énergie. Par conséquent, emporter une grande antenne, ou de grosses batteries, ou encore de grands panneaux solaires dans ces appareils était hors de question. Réduire la taille de l'antenne entraînait obligatoirement une perte de puissance à la réception et le signal transmis avait donc une puissance moindre. Néanmoins, le bruit est toujours le même, donc les erreurs pouvaient facilement détériorer le signal. L'idée des codes correcteurs d'erreur est d'apporter un gain en puissance. En général, nous parlons de rapport signal sur bruit.

Le gain de codage est la différence entre un rapport signal sur bruit (E_b/N_0) requis pour atteindre une certaine performance d'erreur par bit (TEB) avec un système codé et (E_b/N_0)) requis pour atteindre la même performance d'erreur avec un système non codé.

Nous allons maintenant parler du choix de la modulation. En ce qui concerne les transmissions spatiales, la modulation binaire de phase BPSK est la plus sensée car elle offre une meilleure efficacité énergétique par rapport aux autres modulations [5] [6]. Nous venons de voir que les codes correcteurs d'erreur permettent de réduire la valeur du (E_b/N_0) nécessaire à l'entrée du démodulateur pour obtenir la même performance d'erreur sans codage. Ceci a un impact notamment sur les exigences requises vis-à-vis des paramètres de liaison de communication.

Grâce au codage, des communications seront fiables pour de plus grandes distances étant donné que la puissance d'émission peut être réduite. Il est également possible de réduire la taille des antennes ce qui a, comme nous l'avons vu, un impact matériel et économique dans les projets spatiaux.

En effet, il est nécessaire de rappeler qu'il n'existe pas d'opération de codage/décodage parfaite car la probabilité d'erreur n'est jamais nulle. En 1948, Shannon a montré qu'il est théoriquement possible de transmettre l'information à travers un canal avec une probabilité d'erreur arbitrairement faible à condition que le débit de la transmission soit inférieur à la capacité du canal. Shannon a défini la capacité de canal en considérant le cas du bruit blanc additif et gaussien (AWGN).

Les codes correcteurs d'erreur peuvent être classifiés en deux catégories, à savoir les codes blocs et les codes convolutionnels. Les codes convolutionnels sont parmi les plus utilisés dans les systèmes de communications en raison de leur propriété de continuité dans la transmission de l'information. En effet, nous n'avons pas à attendre que le bloc de bits soit arrivé avant de transmettre. Leur performance est excellente et ceci grâce à la puissance des algorithmes de décodage disponibles. Parmi ces algorithmes, nous pouvons penser au décodage de viterbi qui fut une révolution.

En 1993, Berrou, Glavieux et Thitimajshima ont développé une nouvelle approche aux codes correcteurs d'erreur: les codes turbo [2]. Il s'agit en fait de la concaténation parallèle de deux codeurs convolutionnels récursifs et systématiques (CRS) de faible longueur de contrainte à travers un entrelaceur. Le décodage est effectué de façon itérative par deux décodeurs concaténés en série.

Les performances des codes turbo sont fonctions, non seulement des codes convolutionnels utilisés, mais aussi de l'algorithme de décodage et des entrelaceurs [3]. De nombreuses recherches ont portés sur les deux derniers points. Dans ce mémoire, nous nous sommes concentrés sur l'étude et l'implémentation d'un code correcteur d'erreur par turbo code.

Le premier chapitre de ce mémoire porte sur systèmes de communication numérique.

Le 2eme chapitre de ce mémoire prote sur la concaténation parallèle des codes CRS.

Le 3eme chapitre porte sur le décodage itératif des codes concaténés.

Le 4eme chapitre sur la programmation et résultat de simulation.

Chapitre I :

Systèmes de communication numériques.

Chapitre I :

I.1.Introduction

Dans tout système de communication numérique, on cherche à transmettre l'information provenant d'une source vers un récepteur à travers un canal de transmission. Le terme numérique est relatif à la séquence des symboles qu'il utilise pour représenter l'information. En général, pour les communications numériques l'alphabet utilisé est binaire. Ce genre de transmission des données est très intéressant dans le sens où il permet l'utilisation de nombreuses techniques de manipulation de l'information [1].

Les signaux transportés peuvent être soit directement d'origine numérique, comme dans les réseaux de données, soit d'origine analogique (parole, image...) mais convertis sous une forme numérique. La tâche du système de communication est d'acheminer l'information de la source vers le destinataire avec le plus de fiabilité possible ces systèmes sont caractérisés par leur fréquence de transmission et leur débit et en utilisant différents technique de codage et de modulation. [13]

I.2. Systèmes de communication numérique classique :

Le but principal d'un système de communication numérique est de transmettre l'information avec un maximum d'efficacité et de fiabilité d'un usager à un autre, le plus souvent éloigné. Le système que nous considérons est numérique : une séquence de symboles d'un alphabet est utilisée pour transmettre cette information. Dans notre cas, il s'agit de l'alphabet binaire. Comme nous l'avons déjà mentionné, utiliser les transmissions numériques permet en particulier de pouvoir corriger les erreurs. Les différents éléments d'un système de communication sont représentés à la (figure I.1) [7]. Différentes étapes sont illustrées, comme le codage de source et le codage de canal.

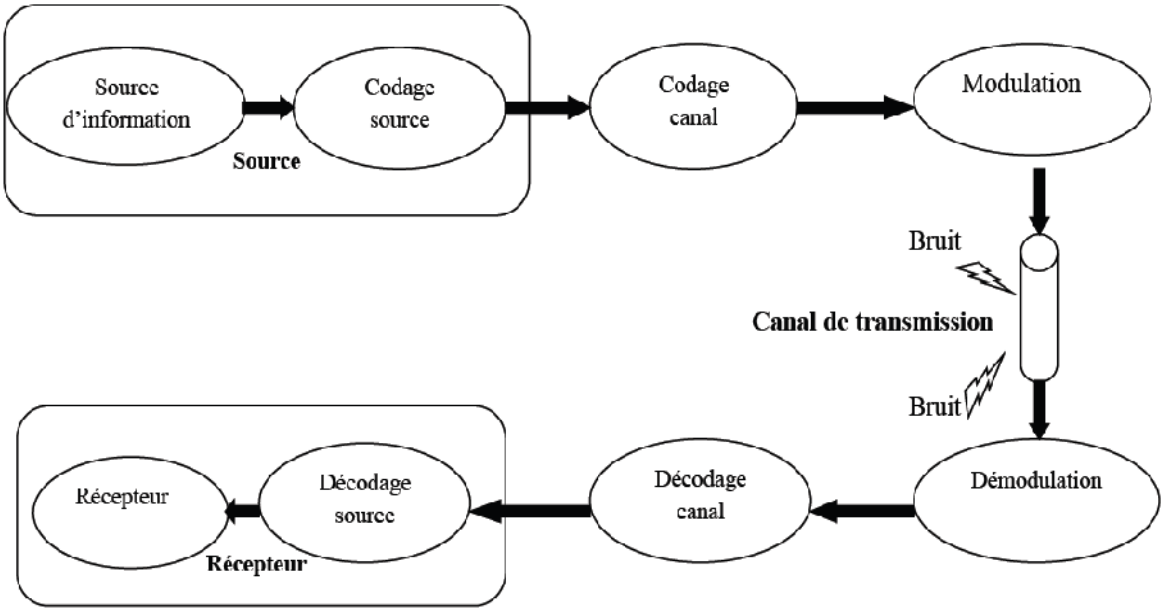


Figure I.1. Modèle de système de transmission numérique.

I.2.1. Le codage de source :

Le codage de source est principalement utilisé pour réduire la redondance dans la séquence d'information. Il existe deux types de sources de données. Tout d'abord, la source de données la plus répandue est analogique, il s'agit tout simplement de signaux électriques. Par exemple, pour un cellulaire, la source de données sera le microphone de l'appareil. En général, lorsque nous sommes confrontés à ce type de source de données, pour pouvoir effectuer le codage, il est nécessaire de numériser l'information d'origine. Ceci nous amène alors au deuxième type des données qui est tout simplement une source numérique. En effet, le numérique se retrouve en grande partie dans les ordinateurs. Lorsque nous parlons de transmissions à l'aide des ordinateurs, la source de données est directement numérique et n'a nullement besoin d'être numérisée. Dans ce cas, le codage de source s'applique directement à la source de données. La plupart du temps, la source est analogique, il faut donc la numériser. Le signal qui provient de la source doit, dans un premier temps, être échantillonné, ensuite, il faut le quantifier et enfin le transformer en une suite binaire de "0" et de "1". Cette transformation s'effectue à l'aide des techniques de conversion analogique-numérique comme la modulation à impulsion codée (PCM). Après avoir codé la source, le codage de canal est effectué. [8]

I.2.2. Le codage de canal :

Le codage de canal consiste à ajouter de la redondance au message de façon à pouvoir corriger les erreurs qui pourraient se transmettre dans le canal. Ces erreurs sont le plus souvent introduites par du bruit qui pourrait faire passer un bit ‘’0’’ à l'état de bit ‘’1’’ ou bit ‘’1’’ à l'état de bit ‘’0’’. Il existe deux grandes familles des codes correcteurs d'erreurs. La première est celle des codes en blocs où une séquence de k bits d'information est codée dans un bloc de n symboles (n > k). La deuxième est celle des codes convolutionnels utilisés pour la transmission des symboles d'information qui arrivent en série sans structure en bloc [8]. Dans nos travaux, la technique de codage utilisée est celle des codes convolutionnels.

I.2.3. Le modulateur :

Les deux étapes de codage (codage de source et codage de canal) que nous venons de décrire sont à effectuer dans l'ordre indiqué. Enfin, le modulateur joue le rôle d'adapteur de canal. Son rôle est de faire en sorte que la séquence convient au canal physique par lequel les données sont transmises. La source de données ainsi modifiée passe au travers d'un canal physique. [16]

La modulation BPSK :

Après avoir codé la séquence d'information, celle-ci est modulée. Le procédé de modulation utilisé est le BPSK (en anglais : Binary Phase Shift Keying*).* Le nombre de bits par symbole pour ce type de modulation est égal à 1. En appliquant la sortie Y_k du codeur à l'entrée du modulateur de la figure I.2, la sortie générée par celui-ci est :

$$\boldsymbol{Y_k = 2d_k - 1} \qquad (\mathbf{1.1})$$

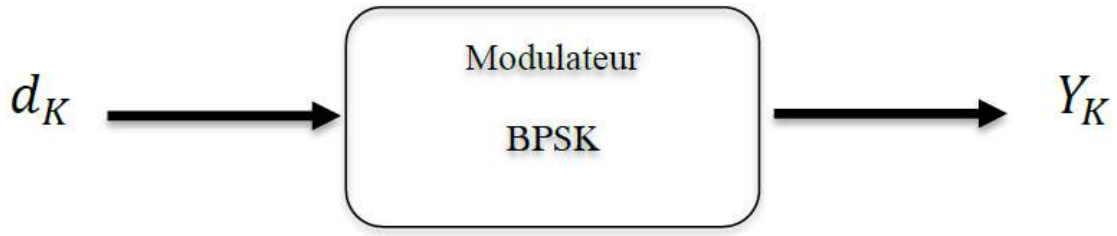


Figure I.2 : Le modulateur BPSK.

Une fois modulée, la séquence obtenue est transmise dans un canal sans mémoire, auquel s'ajoute un bruit blanc gaussien de moyenne nulle et de densité spectrale unilatérale *No* (voir figure I.3). [13]

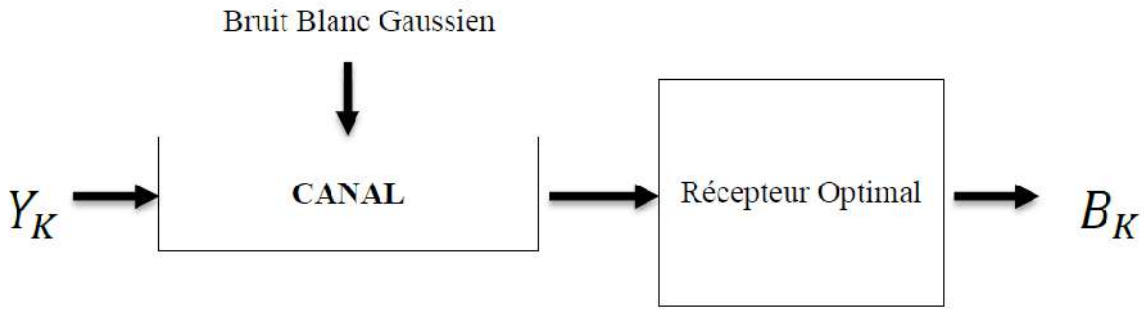


Figure I.3 : Modèle du canal à bruit blanc gaussien additif.

Le bruit blanc gaussien généré par le canal est ajouté à la séquence modulée pour obtenir à l'entrée du décodeur le symbole bruité suivant :

$$\boldsymbol{B_k = Y_k + P_k = 2d_k - 1 + P_k} \qquad (\mathbf{1.2})$$

Où Y_k représente le symbole codé et pk représente la composante du bruit blanc gaussien (p(x)) correspondant au symbole codé Y_k . La densité de probabilité de la variable aléatoire gaussien x est exprimée par :

$$\boldsymbol{P(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) dx} \qquad (\mathbf{1.3})$$

Où σ2 est la variance du bruit blanc gaussien, et vaut :

$$\boldsymbol{\sigma^2 = \frac{1}{2\mathrm{Rtot}} \frac{\mathrm{Eb}}{\mathrm{N0}}} \qquad (\mathbf{1.4})$$

Rtotal étant le taux de codage du système entier, comprenant les deux codeurs de taux R1 et R2 et E_b/N_0 étant le rapport entre l'énergie par bit transmis et le bruit introduit par le canal. Pour que notre système puisse correspondre au système réel, le programme conçu utilise les valeurs réelles (non quantifiées) des symboles bruités (B_k) afin d'estimer la séquence transmise. Évidemment, cela risque d'augmenter le temps de calcul ainsi que la mémoire

exigée par rapport à un système quantifié, car tous les calculs seront basés sur les valeurs réelles et non pas sur des valeurs entières. [13]

I.2.4. Canal 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 (figure 2.1), 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.

Le traitement de l'information souple à de meilleures performances par rapport au traitement de l'information dure. Mais ce premier traitement accroît la complexité algorithmique et matérielle du système.

Il existe de nombreuses représentations des perturbations du canal. Le canal à bruit blanc additif gaussien (AWGN) est le canal de référence pour étudier les codes correcteurs d'erreurs et leurs algorithmes de décodage. [16]

I.2.4.1. Canal symétrique :

Pour un canal binaire, la probabilité d'un symbole de sortie $\boldsymbol{Y_K}$ par rapport à celle d'entrée $\boldsymbol{d_K}$ est constante, dite probabilité d'erreur de transition. Le canal est sans mémoire si le symbole reçu Yk ne dépend que de l'élément $\boldsymbol{d_K}$. Lorsque la probabilité d'erreur est constante et indépendante du symbole $\boldsymbol{d_K}$ le canal est appelé canal symétrique binaire, comme montré sur (la figure I.4), où $\boldsymbol{P_e}$ représente la probabilité d'erreur par élément binaire. La probabilité d'erreur est alors symétrique et peut être écrite de la manière suivante :

P_e (0→1)= P_e (1→0)= P_e.

P_e (0→0)= P_e(1→1)=1-P_e.

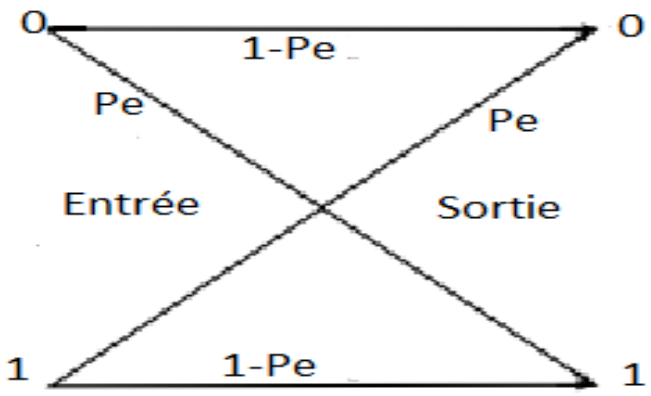


Figure I.4 Canal symétrique binaire de probabilité d'erreur Pe.

I.2.4.2. Canal à bruit blanc gaussien additif :

Le canal binaire symétrique se focalise sur le codeur et le décodeur de canal. Le modulateur et le démodulateur sont supposés avoir un fonctionnement idéal et nous ne nous préoccupons pas de la forme d'onde choisie. Un autre modèle de canal élémentaire est celui qui au contraire permet de prendre en compte les caractéristiques du modulateur et du démodulateur. Classiquement, le canal additif à bruit blanc gaussien (ABBG) ou AWGN (Additive White Gaussien Noise) est le plus utilisé.

Canal AWGN :

C est un modèle de bruit de base utilisé pour imiter l'effet de nombreux processus aléatoires qui se produisent dans la nature.et on dit :

Additif : parce qu'il est ajouté à tout bruit qui pourrait être intrinsèque au système d'information.

White: se réfère à l'idée qu'il a une puissance uniforme dans la bande de fréquences pour le système d'information. C'est une analogie avec la couleur blanche qui a des émissions uniformes à toutes les fréquences dans le spectre visible.

Gaussien : parce qu'il a une distribution normale dans le domaine temporel avec une valeur de domaine temporel moyenne de zéro.

Pour ce canal, l'entrée $\boldsymbol{d_k}$ est discrète binaire, à valeur dans l'alphabet {$\mathbf{0}$, $\mathbf{1}$} et la sortie est continue, constituée d'échantillons analogiques perturbés par un bruit $\boldsymbol{d_k}$, additif, blanc, gaussien, stationnaire, centré et indépendant des éléments $\boldsymbol{d_k}$ à transmettre. Ce modèle de canal correspond à la transmission d'un message continu, sans interférence entre symboles (sans mémoire) dans un milieu à atténuation constante avec réception cohérente et décision pondérée. Le canal est représenté symboliquement par la (figure I.5).

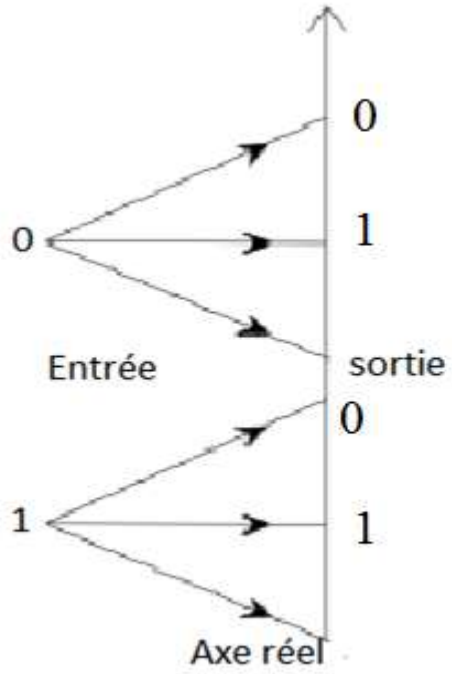


Figure I.5 Canal à bruit blanc gaussien additif.

Le bruit additif blanc gaussien (**BABG**) dispose les propriétés suivantes. Sa densité spectrale de puissance est constante sur une bande de fréquence donnée, dit bruit blanc. Les échantillons du bruit représentent une distribution gaussienne avec une variance $\boldsymbol{\sigma_2}$. La sortie du canal est la somme de l'entrée discrète et d'un échantillon du bruit. C'est alors que la mesure physique à la sortie du canal à l'instant donné est une valeur réelle et donc continue (voir l'axe réel de la figure I.5).

Considérons que le **BABG** est une variable aléatoire gaussienne B de moyenne nulle (centré) et de variance $\boldsymbol{\sigma_2}$ et **b** est une réalisation de **B**. Après filtrage adapté et échantillonnage périodique, l'amplitude des échantillons représente une densité de probabilité **p(b),** exprimée de la manière suivante :

$$\boldsymbol{P(b) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\left(\frac{b^2}{2\sigma^2}\right)}} \qquad (\mathbf{1.5})$$

Où $\boldsymbol{\sigma_2 = N_0/2}$ dont $\boldsymbol{N_0}$ est la densité spectrale de puissance mono latérale constante. [12]

I.2.5. partie réceptrice :

Enfin, du côté récepteur, les fonctions de démodulation et de décodage sont les inverses respectifs des fonctions de modulation et de codage situées du côté émetteur.

Puisque nous avions effectué plusieurs opérations sur l'information avant la transmission, ce qui implique évidemment qu'à la réception, Pour réduire la quantité d'information à transmettre et donc le coût de la transmission, il est nécessaire de compresser les messages numériques avant de les envoyer aux destinataires. La compression de message, dite codage de source, permet de rendre le message plus concis et donc de réduire le débit de transmission. Le message après codage de source est noté $\boldsymbol{d_k}$ dans (la figure I.1). Pour lutter contre le bruit introduit par le canal de transmission, le codeur de canal ajoute une partie de redondance. Cette redondance est associée au message à transmettre et forme un mot de code. La règle de codage doit respecter une loi fixée *a* priori, connue à la réception. Cet ajout de redondance, en outre, permet de détecter voire de corriger les erreurs de transmission à l'aide de l'étape de décodage. Le mot de code est classiquement représenté par une suite de symboles sous forme discrète. L'émetteur, quant à lui, transforme le message numérique en un signal électrique dont les caractéristiques sont adaptées aux propriétés du canal. Dans un environnement radio mobile, l'émetteur a en charge l'opération de modulation qui associe un ou plusieurs éléments binaires à un symbole caractérisé par son amplitude et sa phase.

Réciproquement, le récepteur (figure I.1), qui a pour fonction de reconstituer le message émis, effectue une démodulation pour distinguer le mot de code du signal reçu. Nous introduisons deux types de décision : décision dure et décision souple. La décision dure signifie que toutes les valeurs sont prises dans un alphabet binaire A = {0,1}. Plus précisément, une valeur est égale à 0 si elle est négative, 1 sinon. Pour la décision souple, il s'agit tout d'abord d'une quantification sur le message reçu. Chaque valeur prise est donc composée d'une suite d'éléments binaires. La quantification des données engendre une perte d'information pour le décodeur.

D'après la nature du canal, un circuit de décision prend soit une décision dure, soit une décision souple qui alimente donc le décodeur de canal. Lorsque le canal est discret, le récepteur fournit une suite des symboles numériques binaires d'après le message reçu.

Dans ce cas, le récepteur est sous-optimal puisque la prise de décision dure par canal provoque une perte irréversible d'information. A l'inverse, lorsque le canal et le démodulateur fournissent une représentation souple du symbole, le décodeur délivre alors le mot le plus vraisemblable par rapport au mot reçu, noté $\boldsymbol{d_k}$, en vérifiant si la loi a priori est bien respectée. La démodulation avec décision souple permet alors d'améliorer de façon significative les performances du système correspondant au codage de source, le décodeur de source s'occupe de reconstituer la source initiale $\boldsymbol{a_k}$ à partir du résultat. Le résultat de reconstitution $\boldsymbol{a_k}$ est ensuite transmis aux destinataires en transformant le signal numérique en signal analogique. [12]

I.3. conclusion :

Dans ce chapitre nous avons présenté les éléments de théorie essentiels qui nous serviront comme base pour les chapitres suivants.

Dans le deuxième chapitre nous allons présenter les codes convolutionnels récursif systématiques et la concaténation parallèle de ces derniers.

Chapitre II :

Concaténation parallèle des codes CRS.

Chapitre II :

II.1. introduction :

Les codes convolutionnels font partie des codes correcteurs d'erreur et sont parmi les plus utilisés dans les communications sans fil où une probabilité d'erreur par bit faible est requise. Il existe deux types de codage, à savoir le bloc et le convolutionnel. Les codeurs blocs prennent une séquence d'entrée et la séparent en plusieurs blocs de N bits. Le codeur doit donc attendre d'avoir eu N bits à son entrée pour effectuer son opération d'encodage et transmettre son bloc codé. Les codeurs convolutionnels ont une différente approche dans le sens où le flux de bits à l'entrée et à la sortie est continu. Il ne sépare pas les bits d'entrée en blocs, quand un bit entre dans le codeur, un ou plusieurs symboles codés sont générés à la sortie. Les codes convolutionnels sont basés sur l'introduction de la redondance à l'aide de registres à décalage.

II.2. principe de base des codes convolutionels :

Le principe d'un codeur convolutionnel est illustré par le schéma de (la figure II.1).

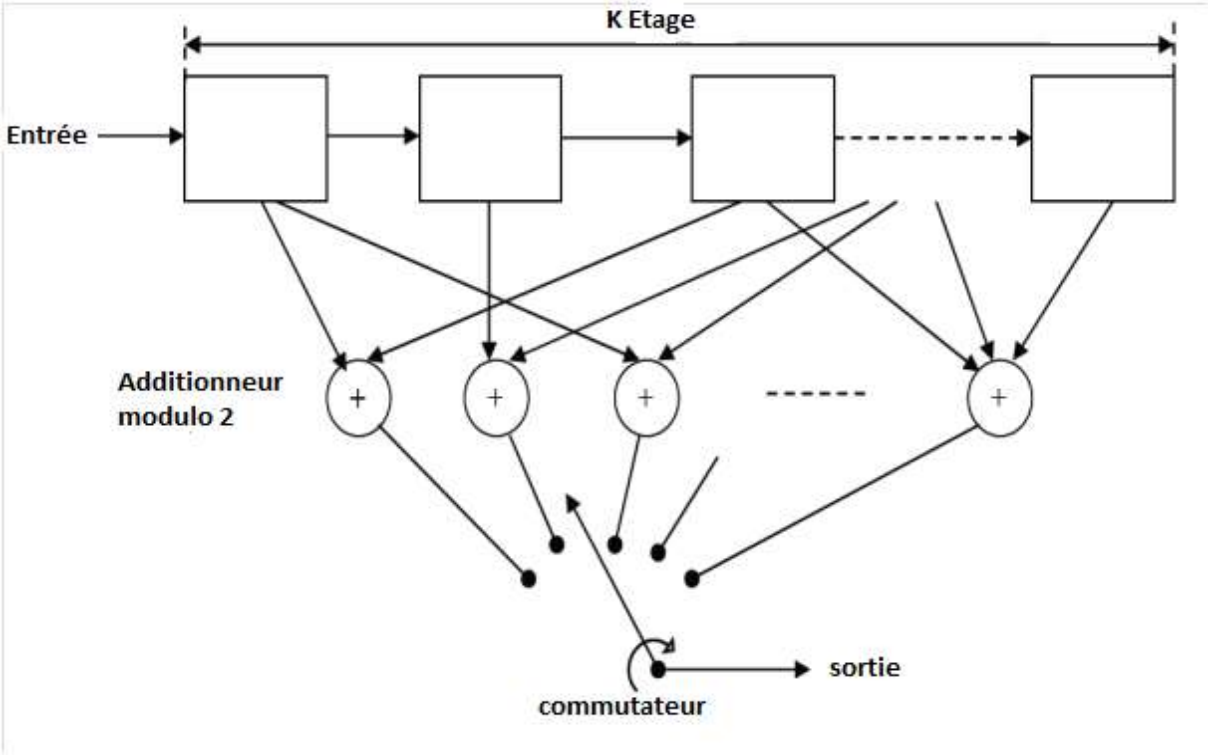


Figure II.1- Principe d'un codeur convolutionnel.

Un codeur convolutionnel de taux 1/n est une machine linéaire constituée d'un registre à décalage de K étages, de n additionneurs modulo 2 qui sont reliés à certaines des cellules de registre et d'un commutateur qui sélectionne séquentiellement les sorties des n additionneurs. La valeur K est appelée la longueur de contrainte du codeur [8]. Le mot de code dépend ainsi non seulement du message d'information à son entrée, mais aussi des états de l'encodeur (les bits de message qui ont déjà été codés) et qui sont stockés dans les registres à décalage.

Au début du codage, toutes les cellules de l'encodeur sont initialisées à zéro. Les bits d'information sont introduits dans le registre un par un et la sortie de l'encodeur est égale au produit de convolution entre le symbole d'information présent à l'entrée du codeur

convolutionnel et les (K - 1) cellules du registre peut prendre une des $\mathbf{2^{k-1}}$valeurs possibles et donc : Nombres d'états d'un codeur = $\mathbf{2^{k-1}=2^{M}}$. La mémoire M du codeur convolutionnel est égale à K - 1. A la fin du codage d'une séquence de bits d'information, nous devrons réinitialiser les cellules de registre par l'insertion d'une séquence additionnelle connue, appelée queue ou terminaison du treillis. Cette dernière est très importante pour coder correctement d'autres séquences [4]. Pour illustrer mieux les codes convolutionnels, nous allons présenter un exemple d'un encodeur convolutionnel.

Les trois principaux paramètres qui caractérisent un codeur convolutionnel sont : la longueur de contrainte, le taux de codage, et le polynôme générateur.

- La longueur de la contrainte d'un code convolutif, est définie comme le nombre de changement produits en sortie par un seul bit du message. Si la mémoire de l'encodeur est égale à M alors le nombre de changement K =M + 1. D'où la longueur de la contrainte de l'encodeur est K.

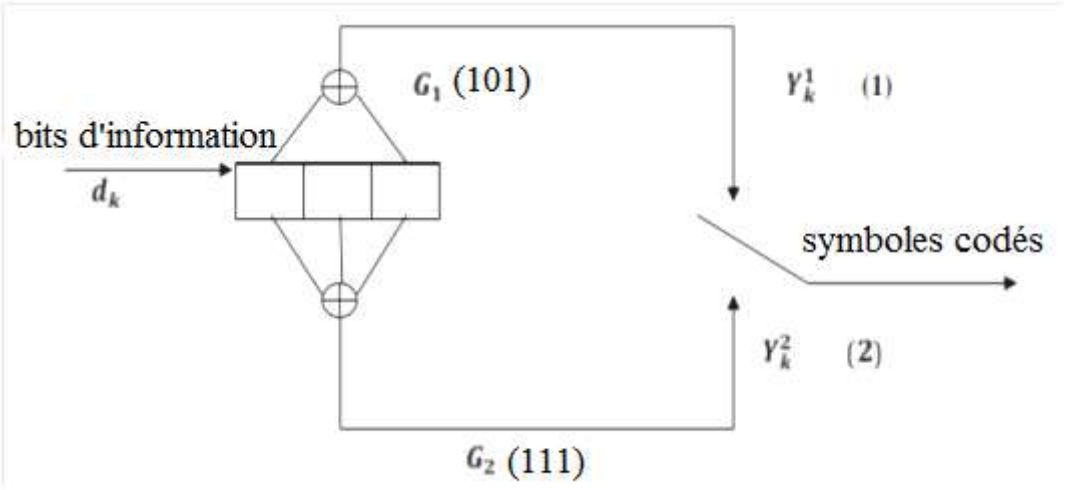


Figure II.2- Codeur convolutionnel de taux de codage R=1/2.

- Le taux de codage R=1/n: en général, il s'agit d'une fraction dont le numérateur correspond au nombre d'entrées et le dénominateur correspond au nombre de sorties.

- Les polynômes générateurs $G_1, \ldots\ldots, G_n$: ceux-ci caractérisent le code et sont les opérations binaires qui permettent d'obtenir les symboles codés de sortie. Chaque polynôme générateur est défini par le paramètre binaire g_{ij}. Ce paramètre est égal à 1 s'il y a une connexion à l'additionneur et 0 s'il n'y en a pas.

Dans notre exemple de (la figure II.2), nous avons alors, le taux de codage R = 1/2, la longueur de contrainte K = 3. A la sortie, comme nous avons un taux de codage de 1/2, deux symboles de parité sont générés. Ces derniers sont définis par l'addition modulo 2 des bits d'information. Les additions sont définies par les générateurs du code. Dans le cas de notre exemple, les vecteurs générateurs sont:

$$\boldsymbol{G_1 = (g_{10}, g_{11}, g_{12}) = (101)}$$
$$\boldsymbol{G_2 = (g_{20}, g_{21}, g_{22}) = (111)}$$

Ceci est exprimé en base binaire. Néanmoins, ces vecteurs générateurs sont le plus souvent exprimés en base octale. Ainsi, dans notre cas, nous obtiendrons : $G_1 = 5 \quad G_2 = 7$ De façon mathématique, soit d_K le bit d'information à l'entrée du codeur à l'instant K, nous pouvons écrire que le symbole codé correspondant Y_k^j peut s'écrire :

$$\boldsymbol{Y_k^j = \sum_{i=0}^{k-1} g_{ji}\, d_{k-i}\ (mod 2)\ j: 1 \dots \dots n} \qquad (\mathbf{2.1})$$

La séquence de sortie est donc une combinaison linéaire des entrées présentes et passées.

II.3. Type de codeur convolutionnel :

Il existe deux types de codeurs convolutionnels: codeur convolutionnel systématique, et codeur convolutionnel récursif et systématique.

II.3.1. Code convolutionnel systématique :

Un codeur convolutionnel est dit systématique si les symboles codés comportent une réplique exacte des bits d'information de l'entrée. Autrement dit, un codeur systématique de taux de codage R = 1/2, laisse passer directement un bit d'information de l'entrée à la sortie comme symbole systématique. Les autres symboles à la sortie de l'encodeur sont appelés symboles de parités.

Les codeurs systématiques sont des codes non catastrophiques, c'est-à-dire, on ne peut trouver une boucle de poids zéro autre que la branche qui boucle sur l'état zéro dans le diagramme d'état. Cela revient du fait que la séquence d'information qui se présente à l'entrée de codeur convolutionnel systématique, se retrouve automatiquement à sa sortie. Pour des faibles rapports signal sur bruit, ces codes offrent des performances de correction d'erreurs meilleures que les codes non-systématiques [3]. Cependant, pour des fortes valeurs du rapport signal à bruit, les codes non-systématiques sont plus performants que les codes systématiques.

II.3.2. code convolutionels systématique récursif :

Un codeur convolutionnel récursif systématique (CRS) est obtenu à partir du codeur convolutionnel qui lui est équivalent au niveau de la longueur de contrainte et des générateurs. Les deux propriétés de cette classe de codes, comme le nom l'indique déjà sont la récursivité et la propriété systématique. La récursivité consiste en une boucle de retour qui est ajouté dans le codeur. Quant à la propriété systématique, elle se manifeste par le fait que les bits d'entrée sont toujours retrouvés intacts dans la séquence de symboles codés à la sortie du codeur.

Selon Forney, il existe pour chaque code non systématique, un code systématique avec une boucle de retour, possédant les mêmes propriétés de distance.

Afin d'examiner la construction des codes récursifs et systématiques, on commence par fournir la représentation polynomiale d'un code convolutionnel. Une représentation équivalente du codeur de (la figure II.1) est illustrée par (la figure II.3).

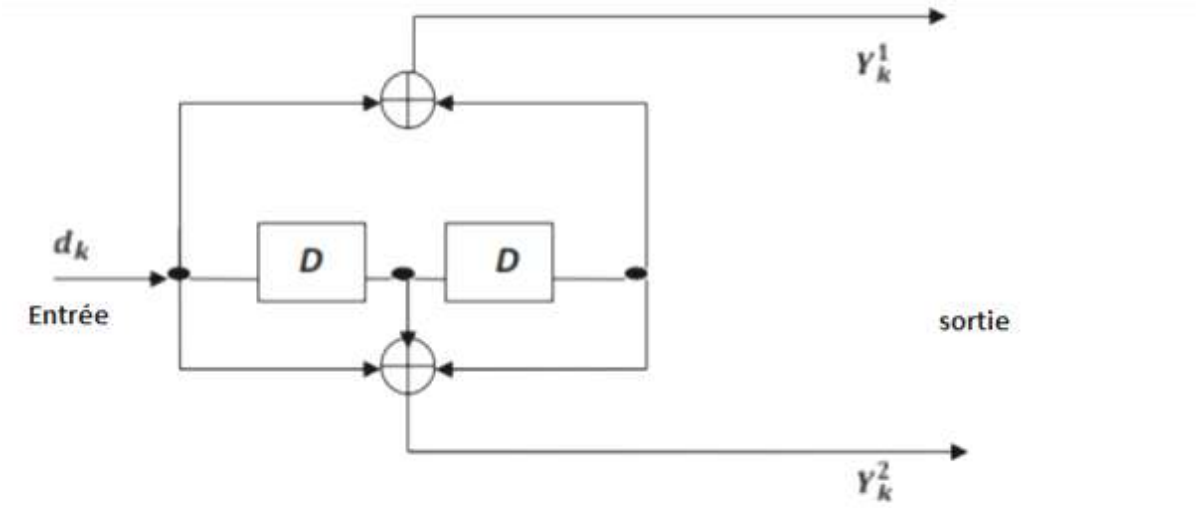


Figure II.3 : Autre représentation du code R=1/2, *K=3, G1=5,* G2=7.

Cette représentation permet de mettre plus en évidence l'aspect mémoire du code. Le registre à décalage à K cellules correspond à K-l opérateurs de retard D, et on définit alors l'état du codeur à un instant k par la valeur des bits en mémoire à cet instant. Le nombre d'états que peut prendre le codeur est donc $M = 2^{k-1}$.
L'opérateur de retard D représenté à la figure II.3 peut être utilisé comme indicateur de temps. Dans ce dernier cas, l'équation devient :

$$\mathbf{Y_k^j(D) = G_j(D)\, d_k(D) \quad j = 1 \ldots\ldots n.} \qquad (\mathbf{2.2})$$

Où $G_j(D)$ sont les polynômes générateurs en D du code définis par :

$$\mathbf{G_j(D) = \sum_{i=0}^{k-1} g_{ji}\, D^i \qquad j = 1 \ldots\ldots n.} \qquad (\mathbf{2.3})$$

Le codeur de la figure II.3 est repris à la figure II.4 :

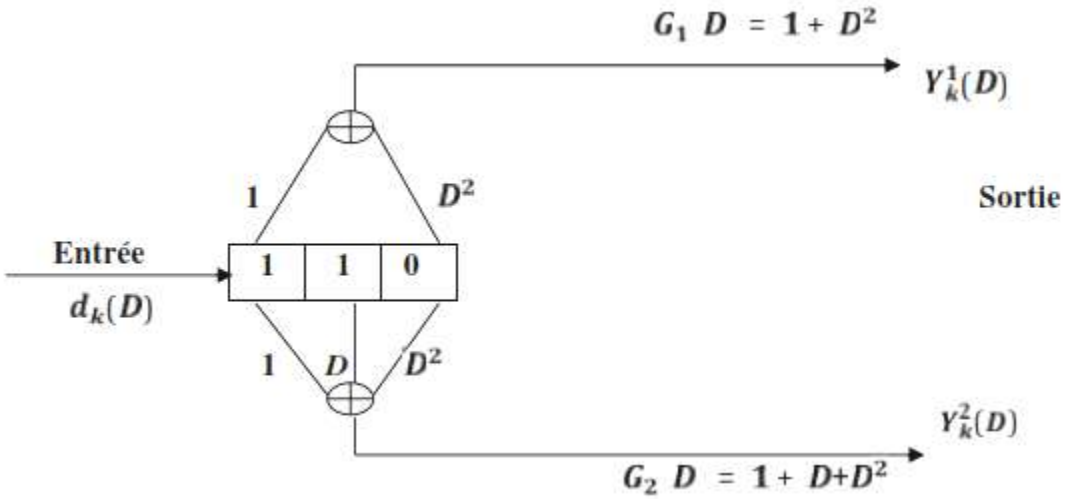


Figure II.4 : Représentation polynomiale du code R=1/2, K=3, G1=5, G2=7.

Pour obtenir la version systématique du code de (la figure II.4), l'une des deux sorties $Y_k^j(D)$, j=l, 2. Doit être égale à $d_k(D)$**.** Pour ce faire**,** on peut diviser les termes de l'équation (2.2) soit par $G_1(D)$Soit par $G_2(D)$. On choisit $G_1(D)$. Dans cet exemple On obtient alors :

$$\mathbf{\overline{Y_k^1}(D) = d_k(D) = \overline{G_1}(D)\, d_k(D).} \qquad (\mathbf{2.4.a})$$

$$\boldsymbol{\overline{Y_k^2}(D) = \frac{G_2(D)}{G_1(D)}\, d_k(D) = \overline{G_2}(D)\, d_k(D).} \qquad (\mathbf{2.4.b})$$

Avec :

$$\boldsymbol{\overline{Y}_k^j = \frac{Y_k^j(D)}{G_1(D)} \quad , \quad j = 1 \ldots\ldots n.} \qquad (\mathbf{2.5})$$

Les polynômes générateurs du code ainsi obtenu sont alors :

$$\boldsymbol{\overline{G_1}(D) = 1.} \qquad (\mathbf{2.6.a})$$

$$\boldsymbol{\overline{G_2}(D) = \frac{G_2(D)}{G_1(D)}.} \qquad (\mathbf{2.6.b})$$

Dans toute la suite, nous adoptons la notation $G = (\overline{G_1}, \overline{G_2})$ soit pour un code systématique $G = (1, \overline{G_2}) = (1, G_2/G_1)$.

En introduisant le polynôme $A_k(D)$ défini par :

$$\boldsymbol{A_k(D) = \frac{d_k(D)}{G_1(D)}} \qquad (\mathbf{2.7})$$

Les équations (4.a) et (4..b) deviennent :

$$\boldsymbol{\overline{Y_k^1}(D) = d_k(D)} \qquad (\mathbf{2.8})$$

$$\boldsymbol{\overline{Y_k^2}(D) = G_2(D) A_k(D)} \qquad (\mathbf{2.9})$$

Le code ainsi construit. Dont les sorties sont définies par les équations (8) et (9) est systématique et récursif**.** Le caractère récursif apparait plus clairement en ramenant l'équation (7) dans l'espace temporel. Elle devient alors :

$$\boldsymbol{d_k = \sum_{i=0}^{k-1} g_{1i}\, a_{k-i}} \qquad (\mathbf{2.10})$$

En tenant compte du fait que toutes les opérations sont faites modulo 2, et en faisant l'hypothèse que $g_{10} = 1$**,** le symbole a_k peut s'exprimer récursivement en fonction des symboles $a_{k-i}(i = 1,2 \ldots\ldots, k-1)$ et du symbole d_k.

$$\boldsymbol{a_k = d_k + \sum_{i=0}^{k-1} g_{1i}\, a_{k-i}} \qquad (\mathbf{2.11})$$

Par conséquent. Dans le cas d'un code CRS**,** ce sont désormais les symboles $\boldsymbol{a_k}$ qui sont contenus dans le registre à décalage du codeur. A partir des relations (8) et (9) (10), les sorties Y_k^1 *et* Y_k^2 du code CRS à un instant k peuvent s'écrire sous la forme :

$$\boldsymbol{Y_k^1 = d_k = \sum_{i=0}^{k-1} g_{1i} a_{k-i}} \qquad (\mathbf{2.12})$$

$$Y_k^2 = \sum_{i=0}^{k-1} g_{2i} a_{k-i} \qquad (2.13)$$

La construction du code CRS à partir du code CNS s'est donc faite en conservant les mêmes séquences génératrices, et en substituant simplement les symboles a_k aux symboles $\boldsymbol{d}$. Un codeur CRS équivalent au codeur de (la figure II.2) est représenté à (la figure II.5). Comme (la figure II.5) l'indique, les bits d'information à l'entrée du codeur représentent une des deux sorties du codeur. Cette sortie du codeur est appelée la sortie systématique.

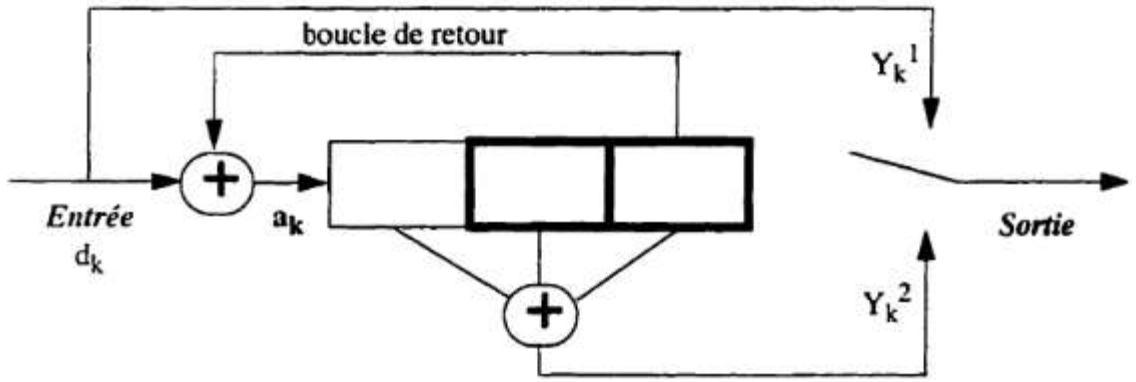


Figure II.5 : structure d'un codeur récursif systématique (K=3, R= 1/2)

A l'arrivée d'un bit d_k celui-ci est additionné modulo 2 aux cellules du registre selon les connexions de l'additionneur à l'entrée du codeur. Le bit a_k résultant de l'addition est alors inséré dans le registre à décalage, et la sortie Y_k^2 calculée.

En comparant les structures des deux codeurs équivalents, c'est à dire le codeur convolutionnel non-systématique et le codeur récursif systématique (figures II.2 et II.5 respectivement), on remarque que le vecteur générateur du code diffère d'un cas à l'autre. On peut être exprimé comme suit :

$\underline{\boldsymbol{G}}\ \boldsymbol{convolutionnel} = (\boldsymbol{G1}, \boldsymbol{G2}) = (\mathbf{5}, \mathbf{7})$ **(2.14)**

Tandis que le vecteur générateur du codeur récursif systématique illustré à la figure II.5 vaut :

$\underline{\boldsymbol{G}}$ **Récursif systématique 1 = (1, 7/5)** **(2.15)**

Le principe de fonctionnement du codeur CRS est le même que celui du codeur convolutionnel non-systématique (CNS) sauf que due à la boucle de retour les bits d'information ne sont pas injectés directement dans le registre à décalage (voir la figure II.5) comme pour les codes CNS. Ainsi, le contenu de la première cellule de registre à décalage ne dépend pas seulement du bit d'information à l'entrée mais aussi du contenu des (K-1) cellules de registre. Par conséquent. Il ne suffit plus d'envoyer une queue de (K-1) zéros afin de remettre le codeur en état initial. Une fois la séquence d'information transmise, les (K-1) bits de la queue doivent être calculés en fonction de l'état final du registre à décalage. Il faut noter qu'on considère les (K-1) dernières cellules du registre comme étant la mémoire du code CRS. À partir de chaque codeur convolutionnel non-récursif et non-systématique de taux R=1/V. On peut obtenir V codeurs CRS en choisissant un des V additionneurs modulo-2 pour la boucle de retour. Le deuxième codeur récursif systématique (par rapport à celui de la figure II.5) obtenu à partir du même codeur de (la figure II.2) est représenté à (la figure II.6). Dans ce cas, le vecteur générateur du code devient :

$\underline{G}$ Récursif systématique 2 = (1,5/7) **(2.16)**

Notons qu'il n'existe jamais de connexion entre la première cellule du registre et la boucle de retour pour les codeurs récursifs.

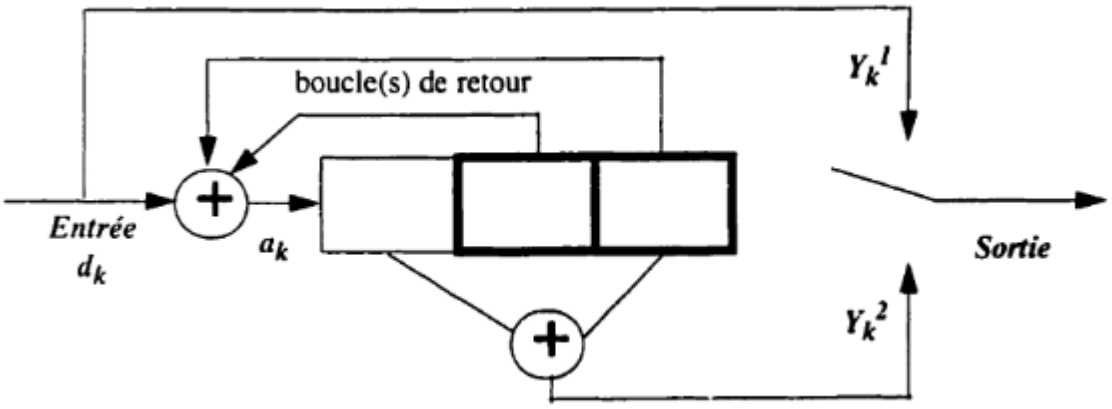


Figure II.6 : Structure du deuxième codeur récursif systématique (K=3, R=1/2).

La structure du treillis de code CRS est la même que celle du code CNS à partir duquel il a été construit, c'est-à-dire que, pour une même transition entre états, les mots de code sont identiques. Toutefois, les séquences de sorties des codes CRS et des codes CNS ne correspondent pas à la même séquence d'entrée. Cela implique qu'au niveau du spectre du code, les deux types de codes possèdent la même distance libre, les mêmes nombre de mots code de poids de Hamming mais le nombre de bits d'information correspond à tous les mots code est différents. Puisque les deux types de codes ont la même distance Libre, ils peuvent alors être considérés comme étant équivalents à des rapports signal à bruit élevés.

Les performances des codes récursifs systématiques à des faibles rapports signal à bruit sont légèrement supérieures aux codes convolutionnels non-récursifs.

Dans un système contenant deux codeurs/décodeurs concaténés en parallèle ou en série, il a été prouvé que l'utilisation des codes CRS apporte une amélioration significative des performances par rapport à des codes CNS. [13]

II.4. Représentation graphique des codes convolutionnels :

Un codeur convolutionnel est une machine à états finis. Pour un taux de codage R = 1/n, l'état de codeur est le contenu des K - 1 premières cellules (mémoire du codeur). Le nombre d'états possibles est égal à 2^{k-1}. La sortie de l'encodeur dépend de son entrée et de son état présent. Après chaque introduction d'un nouveau symbole, l'état du codeur change. Le nombre des transitions entre les différents états est fini, par conséquent, il est possible de présenter graphiquement ces transitions en indiquant l'entrée et les sorties du codeur.

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 [7] [8]. Pour mieux expliquer ces représentations, considérons notre exemple de (la figure II.2).

II.4.1 Diagramme d'états :

Un diagramme d'états du code convolutionnel représente la relation entre les symboles codés. Les bits d'information et les états du codeur. Cette représentation est rendue possible par le

fait que le codeur ne peut prendre qu'un nombre fini d'états. (La figure II.4) représente le diagramme d'état du codeur de (la figure II.2).
Les nœuds du diagramme d'état représentent les états du codeur et le résultat de transition entre deux états représente les v symboles codés qui correspondent au bit d'information à l'entrée du codeur. Chaque branche de (la figure II.7) représente les symboles codés correspondant au bit d'information.

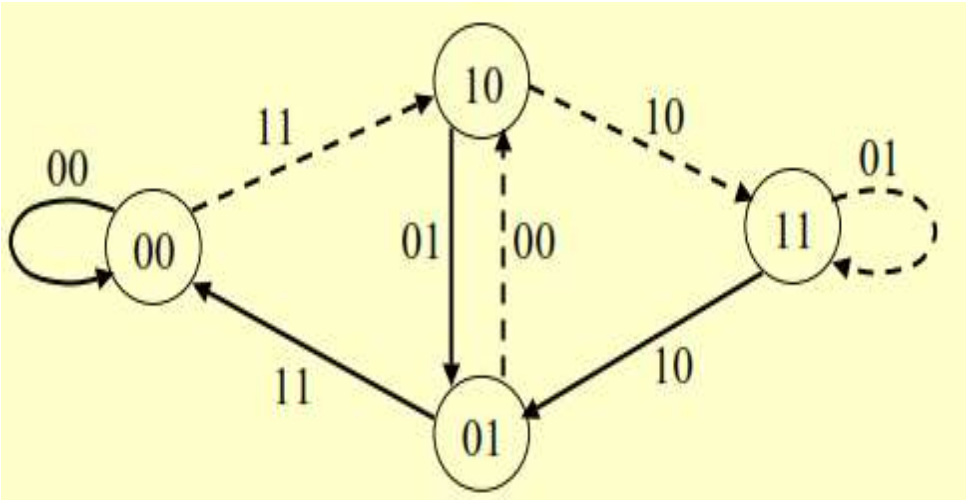


Figure II.7 : Diagramme d'état du codeur de la figure II.2 (K=3. R=1/2)

Le diagramme d'état permet d'analyser et de déterminer les performances d'un code particulier. Par exemple, la fonction de transfert du code utilisée pour déterminer les distances de Hamming de tous les mots de code par rapport au mot de code 0 = **(**0,0 ... 0) peut être obtenue à partir du diagramme d'états du code. L'inconvénient est que cela devient beaucoup trop complexe lorsque K augmente et cette méthode n'est utilisée que pour des codes de faible longueur de contrainte. D'autre part, le diagramme d'état n'explicite pas la relation entre la dynamique d'encodage et la profondeur dans la séquence des bits d'information. C'est pour cette raison qu'on utilise plus souvent le treillis d'encodage pour représenter un code convolutionnel. [13]

II.4.2. Treillis d'encodage :

Un treillis d'encodage est une représentation du codeur convolutionnel qui tient compte du fait que le nombre d'états du codeur est fini ainsi que de la propriété de convergence. Cette dernière caractéristique traduit le fait que lorsque deux séquences d'information sont identiques pendant au moins (K-1) bits consécutifs, alors le codeur se trouve dans un état identique pour les deux séquences. Le treillis est constitué des nœuds représentant les états du codeur en question et des branches reliant les nœuds du treillis, représentant les transitions entre les états du codeur. Par rapport au diagramme d'état, la représentation en treillis introduit une information de plus, celle du temps. La notion du temps joue un rôle très important, comme on le verra par la suite, lors du décodage. La longueur du treillis représente la profondeur dans la séquence des bits d'information (notion du temps) et sa largeur, le nombre d'états du codeur. Le treillis du codeur convolutionnel de (la figure II.1) (K=3, R= 1/2**,** G_1=5**.** G_2=**7)** est représenté par (la figure II.3), où les états sont identifiés sur la colonne de gauche.

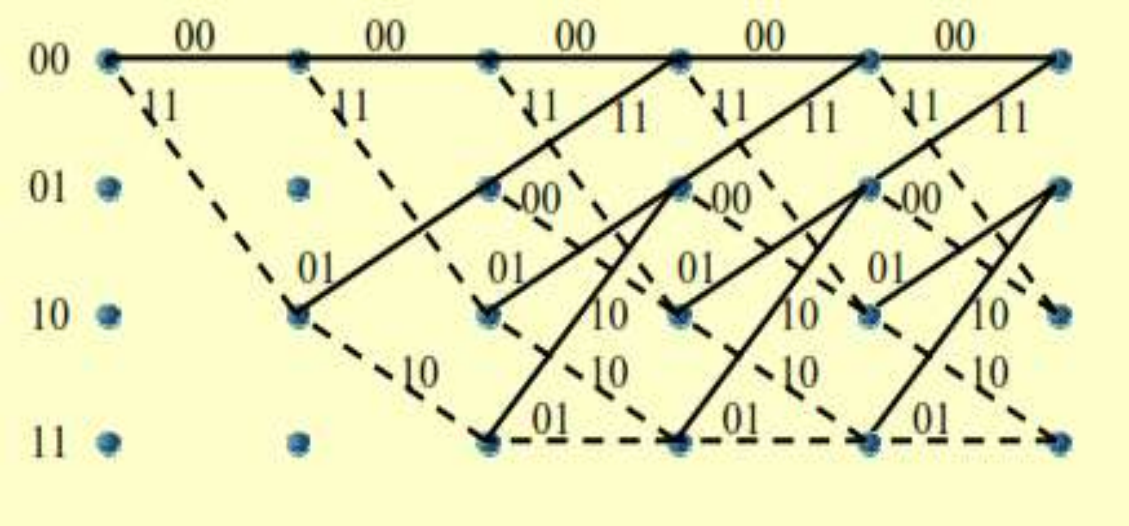


Figure II.8 : Représentation en treillis du code convolutionnel de la figure II.2.

Comme on peut le constater à (la figure II.3), le treillis du code de taux de codage R=1/ V a toujours deux branches sortant de chaque état. Dans le cas des taux R=1/ V, il y a un bit à l'entrée du codeur à chaque instant et par conséquent, deux possibilités pour le prochain état. En général. Pour le taux de codage R=b/ V on a b bits à l'entrée du codeur à chaque instant, ce qui est équivalent à 2^b possibilités pour l'état suivant ou encore, 2^bbranches arrivant à chaque nœud et 2^b branches sortant de chaque nœuds du treillis. [13]

II.5. l'entrelacement :

II.5.1. Avantage d'utilisation des entrelaceurs :

Dans le cas des canaux à des erreurs qui arrivant en rafales (burst error), le problème est qu'une rafale d'erreur peut toucher un seul mot de code. Les codes ne sont pas capables de corriger les erreurs s'il y a plus d'un certain nombre de bits en erreurs dans un mot de code (capacité de correction de code). Le but de l'entrelacement est alors de faire en sorte qu'une rafale d'erreur se retrouve dans plusieurs mots de code différents**.**

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 d_{min} peut détecter jusqu'à ($d_{min} - 1$) erreurs et peut corriger jusqu'à *t* erreurs, *t* vérifie la relation suivante :

$$\boldsymbol{d_{min} \geq 2t + 1 \; ou \; t = floor\left(\frac{d_{min} - 1}{2}\right)} \qquad (\mathbf{2.17})$$

Considérons un code qui peut corriger jusqu'à *t* erreurs. Nous prenons un exemple de transmission de données simplifiée sur un canal bruyant comme le montre (la figure II.6). 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 II.7).

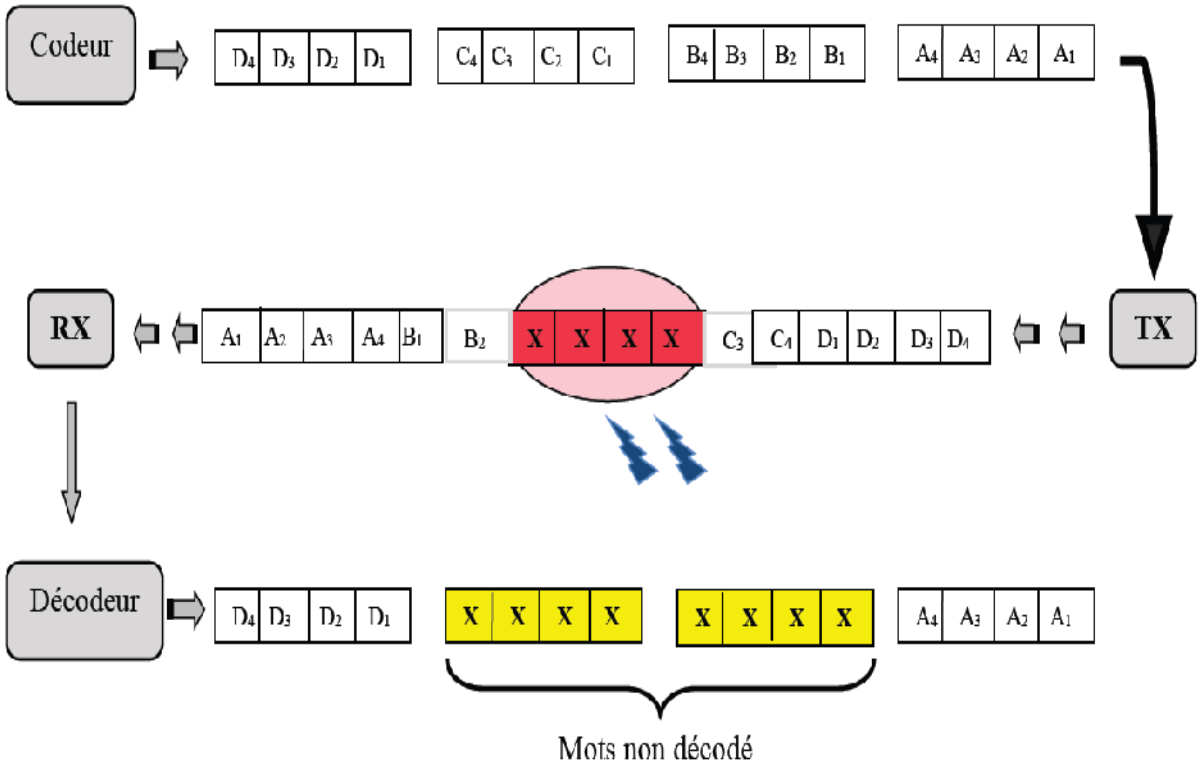


Figure II.9. Modèle simplifiée de transmission et de réception sur un canal bruité

Les bits transmis sont les mêmes, mais commandés d'une manière spéciale. Cette réorganisation des bits est appelé permutation des données.
Considérant même effet du bruit, comme dans (la figure II.6), les données reçues sont corrompues sous la forme d'un paquet d'erreurs. 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 a 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. [16]

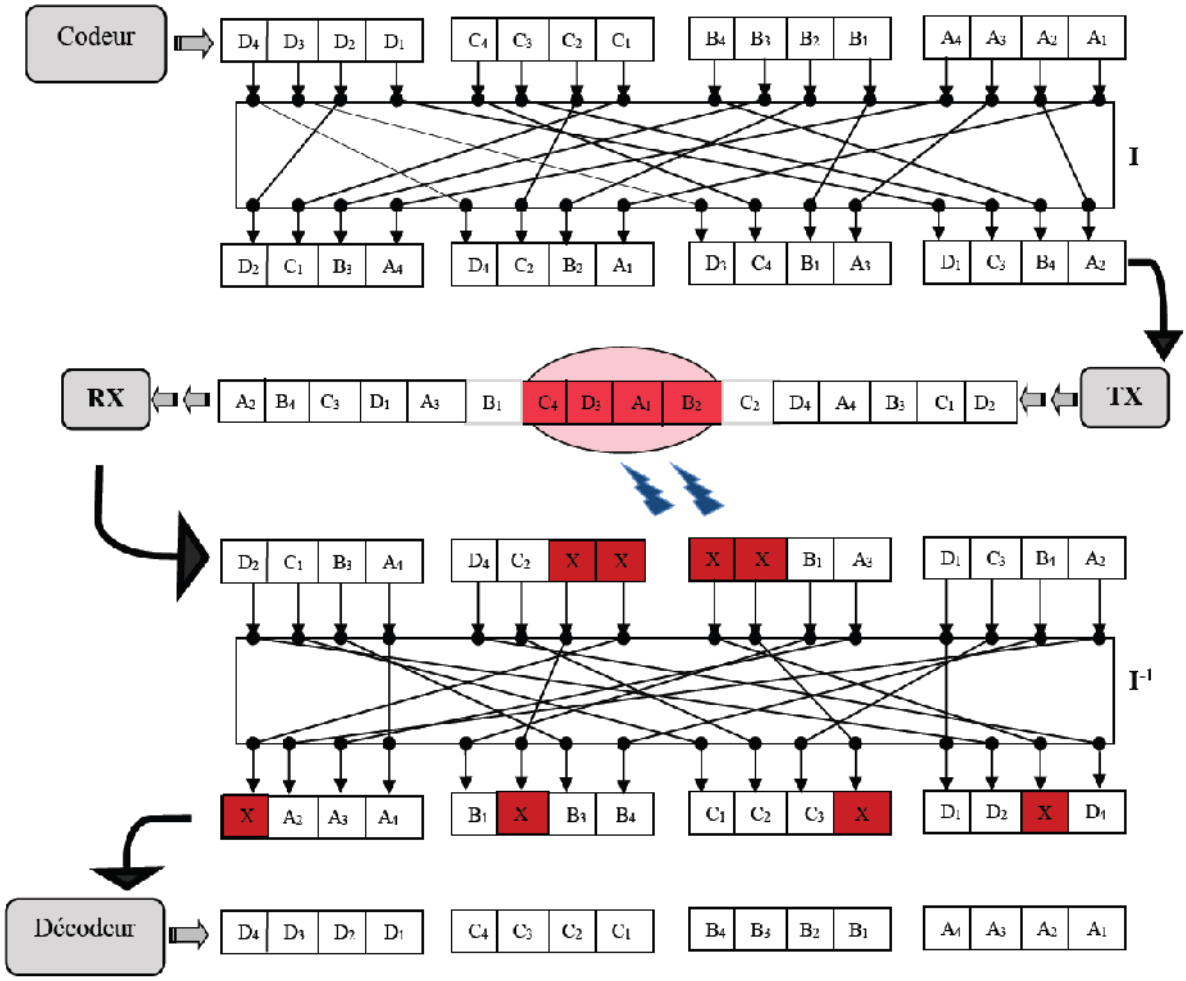


Figure II.10 Modèle simplifiée de transmission et de réception sur un canal bruité avec Entrelaceur et désentrelaceur.

II.5.1.Principe d'entrelacements :

L'entrelacement est une technique très utilisée dans un grand nombre de systèmes de communications numériques. En fait, un entrelaceur prend une séquence de symboles en entrée et reproduit la même séquence à la sortie mais dans un ordre temporel complètement différent. On peut dire alors qu'un entrelaceur est un système qui permute les éléments d'une séquence, sans bien sûr, aucune répétition. [16]

La technique d'entrelacement ou de permutation consiste à disperser les erreurs dans le temps, elle a toujours rendu de grands services en communications numériques. On l'utilise pour réduire les effets des atténuations plus ou moins longues dans les transmissions affectées d'évanescences, et plus généralement dans des situations où des perturbations peuvent altérer des symboles consécutifs. Dans le cas de turbo code aussi, la permutation permet de lutter efficacement contre l'apparition de paquets d'erreurs, sur l'une au moins des dimensions du code composite.

Les fonctions de permutation ou d'entrelacement, utilisées entre codeurs élémentaires dans un schéma concaténé, ont un double rôle. D'une part, elles assurent, à la sortie de chaque décodeur élémentaire, une dispersion temporelle des erreurs qui peuvent être en produites en rafales. Ces paquets d'erreurs deviennent alors erreurs isolées pour le décodeur qui suit, avec

des effets de corrélation bien amoindris. Cette technique de dispersion des erreurs est utilisée dans un cadre plus large que celui du codage de canal.
Notant qu'un délaceur permet de restaurer l'ordre de la séquence d'entrée de l'entrelaceur.
La combinaison de deux codeurs CRS et un entrelaceur a permis de créer un code avec de meilleures propriétés de distance de Hamming [4]. D'ailleurs, il a été introduit comme un élément principal dans la concaténation des codes en parallèle ou en série pour pouvoir d'une part utiliser des codes avec des longueurs de contrainte K petites et d'autre part pour diminuer la complexité des algorithmes de décodage tout en gardant presque les mêmes performances obtenues en utilisant des codeurs classiques de grande mémoire [2].
Au niveau du décodage, l'entrelaceur détient un autre rôle, il permet de décorrélé les entrées des décodeurs surtout lorsque nous utilisons un algorithme de décodage itératif basé sur l'échange de l'information entre ces derniers [7] [9]. Autrement dit, dans le cas de décodage itératif utilisant deux décodeurs, si les séquences à l'entrée de ces derniers sont décorrélées alors il y aura une forte probabilité que les erreurs indétectables par l'un seront corrigées par l'autre.
La notation que nous utiliser pour un entrelaceur de période T est la suivante :

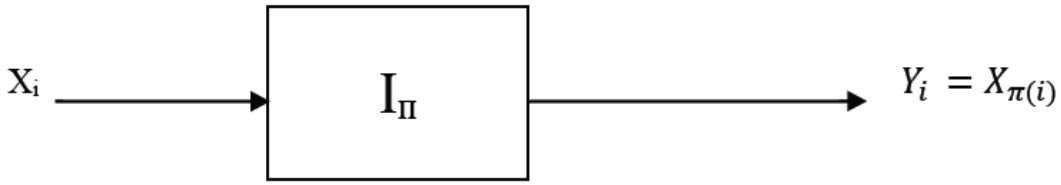


Figure II.11. Principe d'entrelacement.

Le bit x_i est inséré à l'entrée de l'entrelaceur *I*, à l'instant i. La sortie, après une certaine latence, est alors $y_i = x_{\pi(i)}$ Cette notation correspond au temps réel. Si maintenant, nous exprimons la même chose en termes de séquences, nous obtenons $y = I_\pi(x)$ De façon mathématique, l'entrelaceur I_π est décrit par la permutation : $\pi: Z \to Z$. L'entrelaceur est alors une permutation sur les entiers de l'alphabet Z.
Il faut indiquer qu'un désentrelaceur joue le même rôle qu'un entrelaceur, la différence se situe dans le fonctionnement inverse de ce dernier. Le désentrelaceur est un système qui opère à la sortie de l'entrelaceur ou plus loin dans un système de communication et qui remet en place les symboles tels qu'ils étaient à l'origine. Nous utiliserons la notation I_d, pour le désentrelaceur. Avec les notations que nous utilisons, il vient tout de suite que :

$$\boldsymbol{I_d = I_\pi^{-1}} \qquad (\mathbf{2.18})$$

De façon logique, pour qu'un entrelaceur soit réalisable, il faut qu'il soit périodique de période finie. De façon mathématique, pour vérifier qu'un entrelaceur est bien de période finie *T*, il suffit d'effectuer une simple opération de délai sur cet entrelaceur. Si l'entrelaceur commute avec l'opération délai, alors ce dernier possède une période finie. Un entrelaceur qui a une période T satisfait l'équation suivante :

$$\boldsymbol{\pi(i) - T = \pi(iT)} \qquad (\mathbf{2.19})$$

Ayant maintenant introduit la notion de période finie d'un entrelaceur, il nous est possible de représenter les permutations de ce dernier. Il s'agit d'illustrer les images de chacun des éléments de l'entrelaceur. Une des façons de faire est illustrée à (la figure II.9).
Cette notation correspond à la permutation fondamentale de l'entrelaceur. Pour obtenir les autres permutations lors d'une communication numérique, il suffit de combiner cette permutation fondamentale avec la périodicité de l'entrelaceur. [16]

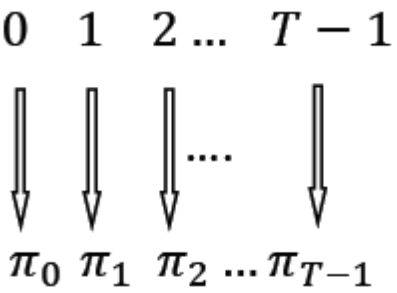


Figure II.12. Vecteur de permutation d'un entrelaceur.

II.5.3. 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. Il existe différentes catégories des entrelaceurs en bloc: [16]

II.5.3.1 Entrelaceur en matrice :
Un entrelaceur en matrice est un entrelaceur de période P=N, M, P représente la taille de la matrice d'entrelacement. Ce type d'entrelaceur se caractérise par un processus dans lequel les données sont écrites ligne par ligne dans une matrice et ensuite ces données sont lues colonne par colonne. Nous allons essayer à l'aide d'un exemple d'illustrer le fonctionnement d'entrelaceur en matrice. Supposons qu'une séquence de 9 symboles (N = 3, M= 3, P = 9) soit reçue à l'entrée de l'entrelaceur. Nous allons d'abord placer ces 9 symboles dans une matrice (voir figure (II.10)) dans un ordre croissant des nombres indiques dans les cases (c.à.d. le premier symbole reçu est placé dans la case 0, le deuxième dans la case 1 et ainsi de suite). Les symboles sont réécrits à la sortie de l'entrelaceur en matrice dans l'ordre suivant :

Sortie de l'entrelaceur : 036147268

0	1	2
3	4	5
6	7	8

Figure II.13. Entrelaceur en matrice de taille 3x3.

II.5.3.2 Entrelaceur hélicoïdal :

L'entrelaceur hélicoïdal est un autre entrelaceur ligne-colonne modifié qui écrit des données 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:

$$\boldsymbol{\pi(t) = i_r c + j_r} \qquad \mathbf{(2.20)}$$

$$\boldsymbol{i_r = C * R - 1 - t(mod\ R)} \qquad \mathbf{(2.21)}$$

$$\boldsymbol{j_r = t(mod\ C)} \qquad \mathbf{(2.22)}$$

Où R et C sont premiers et représentent le nombre de ligne et colonne de l'entrelaceur, respectivement. (La figure II.11) présente un exemple d'entrelacement hélicoïdal.

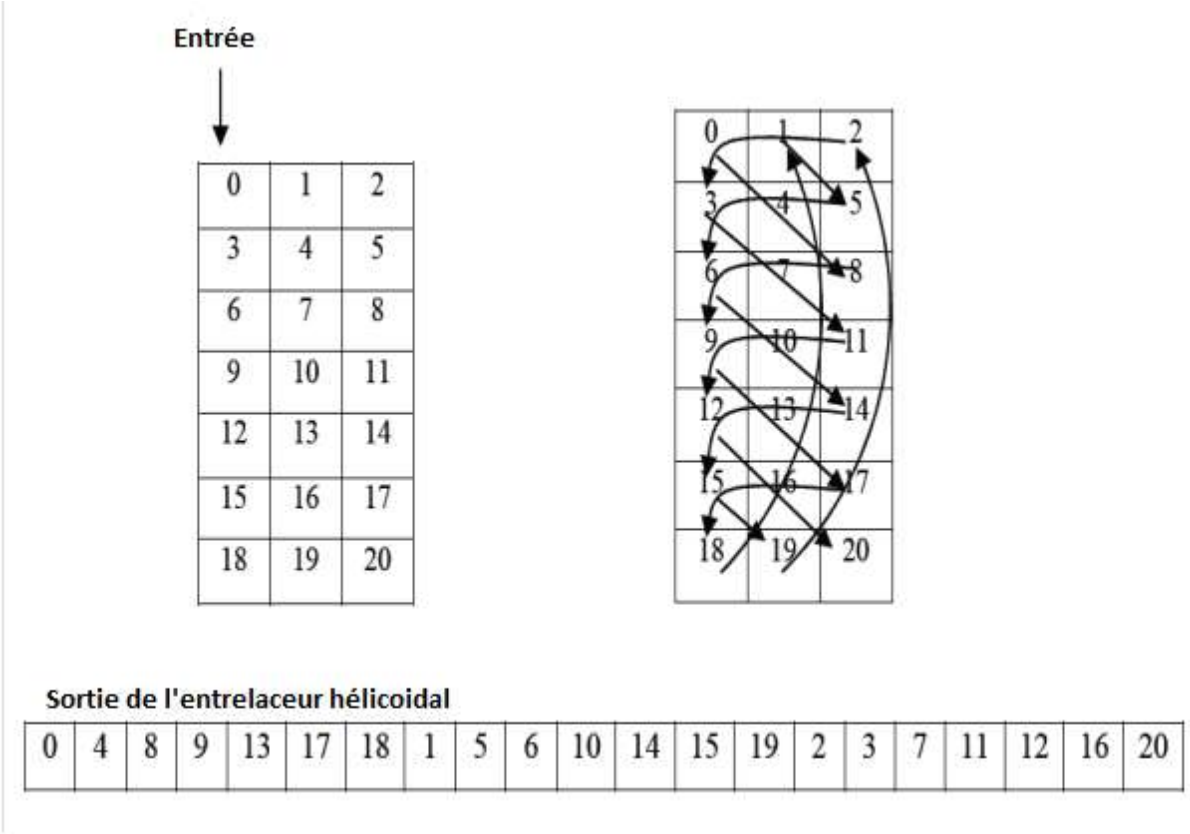


Figure II.14 : Entrelaceur hélicoïdal.

II.5.3.3. Entrelaceur aléatoire :

II.5.3.3.1. Entrelaceur pseudo-aléatoire :

Le concept fondamental d'un entrelaceur pseudo-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 la matrice d'un entrelaceur pseudo-aléatoire, les symboles en sortie sont choisis d'une façon quasi-aléatoire de telle manière qu'il ne faut pas répéter le même symbole déjà sélectionné. Comme la sélection est aléatoire, II sera impossible de connaître les positions des symboles à la sortie de la matrice d'entrelacement. Par conséquent. Il serait utile de garder une table de correspondance entre les anciennes et les nouvelles positions des symboles entrelacés afin de pouvoir délacer ces derniers. [14]

II.5.3.3.2. Entrelaceur pseudo-aléatoire pair-impair :

Les entrelaceurs pseudo-aléatoires pair-impair sont des entrelaceurs pseudo-aléatoires avec une simple condition : les symboles avec leurs positions initiales paires (respectivement impaires) seront entrelacés aléatoirement en sortie dans des positions paires (respectivement impaires).

La figure II.12 montre un exemple d'entrelaceur pseudo-aléatoire pair-impair. [14]

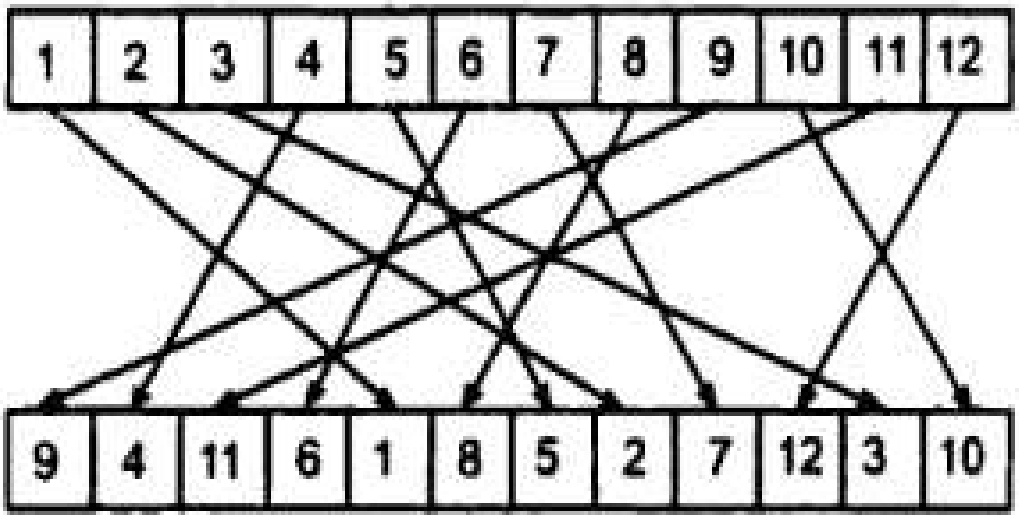


Figure II.15 : Entrelaceur pseudo-aléatoire pair-impair.

II.6. concaténation parallèle des codes CRS :

La concaténation parallèle utilise deux ou plusieurs codeurs reliés par un entrelaceur. Dans le cas des codes turbo, les deux codeurs sont de préférence identiques et de type convolutionnel récursifs systématiques. Un codeur turbo consiste à la concaténation parallèle de deux codeurs identiques de type CRS séparés par un entrelaceur. À l'instant K le bit est codé par le codeur supérieur lequel produit le bit de parité Y_k^{1P}. L'ensemble des bits d_k sont ensuit entrelacés et codés par le codeur inferieur qui produit le second bit de parité Y_k^{2P}. Les deux bits de parité peuvent être perforés ou non afin de produire le vecteur Y_k^p. qui sera multiplexé avec le bit d'entrée Y_k^S.

Les séquences codées générées par un codeur concaténé en parallèle ont permis d'atteindre des performances d'erreur surprenantes qui sont de l'ordre de quelques dixièmes de dB de la limite de Shannon. Mentionnons que l'utilisation de codeurs CRS se justifie par le fait que pour des petits rapports signal sur bruit les codeurs CRS sont plus performants que les codeurs NRNSC pour la même longueur de contrainte. Ajoutons que ceci est aussi valable lorsque le taux de codage est élevé. Par contre pour des rapports signal à bruit élevés, les codeurs NRNSC donnent une performance d'erreur meilleure que les codeurs CRS.

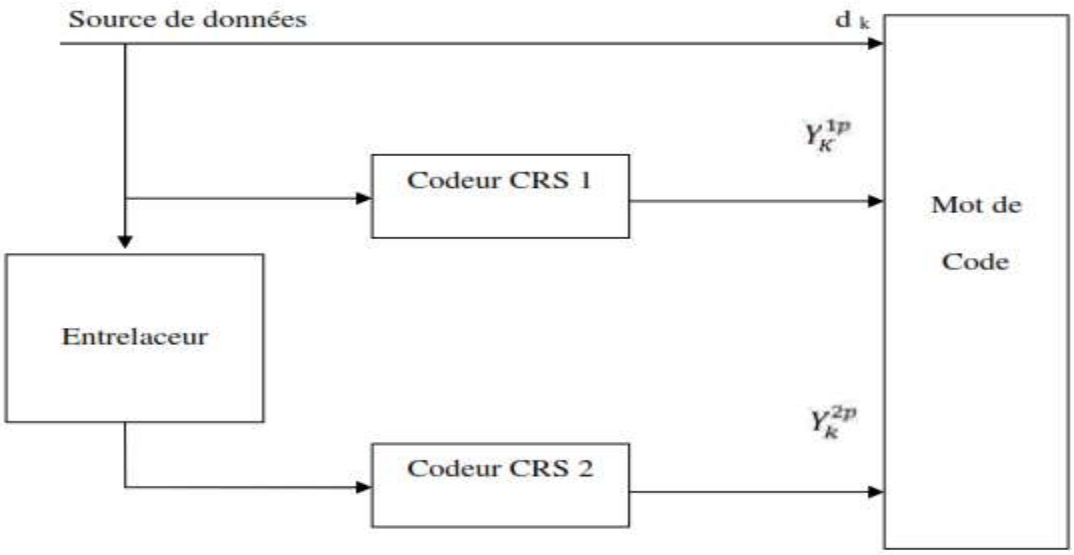


Figure II.16 : Schéma d'une concaténation parallèle des codes CRS.

II.7. L'encodeur Turbo :

L'encodeur turbo est constitué de deux ou plusieurs codeurs convolutionnels récursifs, systématiques concaténés en parallèle et séparés par un ou plusieurs entrelaceurs. Cette structure a été originalement introduite par Berrou, Glavieux et Thitimajshima. Le terme concaténation en parallèle provient du fait que les codeurs utilisent la même séquence d'information mais dans un ordre temporel différent.

Le taux de codage, noté R_{gp} de l'encodeur turbo obtenu à la suite d'une concaténation parallèle d'un premier codeur de taux de codage R_1 et d'un deuxième codeur de taux de codage R_2 vérifie la relation :

$$\frac{\mathbf{1}}{\boldsymbol{R_{gp}}} = \frac{\mathbf{1}}{\boldsymbol{R_1}} + \frac{\mathbf{1}}{\boldsymbol{R_2}} - \mathbf{1} \qquad (\mathbf{2.23})$$

Soit :

$$\boldsymbol{R_{gp}} = \frac{\mathbf{1}}{\frac{\mathbf{1}}{\boldsymbol{R_1}} + \frac{\mathbf{1}}{\boldsymbol{R_2}} - \mathbf{1}} \qquad (\mathbf{2.24})$$

$$\boldsymbol{R_{gp}} = \frac{\boldsymbol{R_1 R_2}}{\boldsymbol{R_1 + R_2 - R_1 R_2}} \qquad (\mathbf{2.25})$$

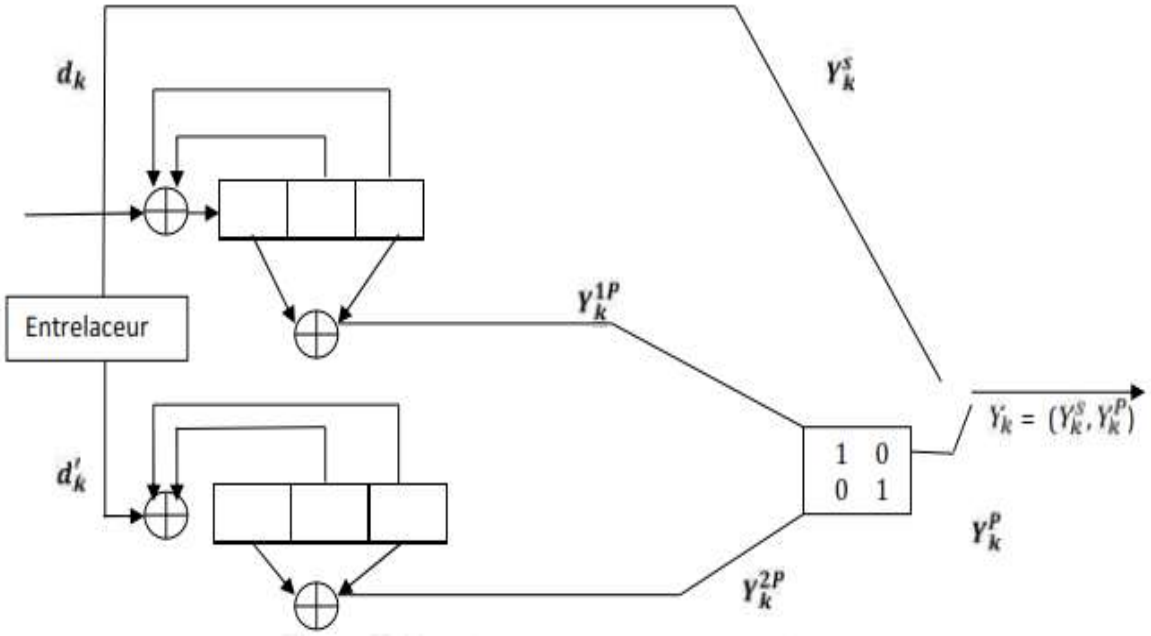


Figure II.17 : Schéma de principe d'un encodeur Turbo.

Pour mieux illustrer l'encodeur turbo, nous avons représenté sur (la Figure II.14) un exemple d'une concaténation parallèle de deux codes élémentaires dont le taux de codage global est égal à Rgp=1/2 après perforation. Nous trouvons dans cet exemple, en plus des composantes déjà mentionnées, comme les codeurs convolutionnels et I'entrelaceur, l'existence d'une matrice de perforation et d'une unité de multiplexage. L'unité de multiplexage est utile pour multiplexer les sorties de l'encodeur turbo. [15]

II.8.conclusion :

Dans ce chapitre nous avons tout d'abord introduit le principe de codage convolutionnel et nous avons présenté quelques types d'entrelaceur et la concaténation parallèle. Nous intéressons, par la suite au chapitre suivante concernant la partie décodage itératif en utilisant le décodeur APP avec l'algorithme de décodage MAP.

Chapitre III :

Décodage itératif des codes concaténés.

Chapitre III :

III.1. introduction :

Dans le chapitre précédent, nous avons décrit la structure de code turbo. Dans ce genre de système, il existe d'une partie de codage et une partie de décodage. Dans cette deuxième partie, l'élément le plus important est l'algorithme de décodage, car c'est ce dernier qui permet en grande partie de minimiser la probabilité d'erreur par bits (TEB).

De nombreuses recherches ont été effectuées dans ce domaine. En 1966, Chang et Hancock ont développé un algorithme minimisant la probabilité d'erreur par symbole. Cet algorithme est appelé **MAP** pour Maximum a Posteriori, ensuite des nouveaux travaux ont adapté cet algorithme aux codes correcteurs d'erreur [7]. Le choix d'un algorithme de décodage est un problème majeur. En effet, deux algorithmes de décodage ont été proposés pour les codes turbo. Le premier est le **SOVA** (Soft Output Viterbi Algorithm) qui est une variante de l'algorithme de Viterbi. Le deuxième est celui que nous allons traiter dans la première partie de ce chapitre et qui porte le nom de l'algorithme **MAP**. La caractéristique principale de ces deux algorithmes est qu'ils fournissent une mesure de fiabilité sous forme probabiliste (appelée aussi métrique) sur chaque symbole décodé. Dans ce chapitre, nous allons seulement nous intéresser à l'algorithme **MAP** et le décodage itératif.

Dans un premier lieu, nous allons introduire le principe du maximum a posteriori qui est la base de l'algorithme **MAP**. Ensuite nous allons présenter l'algorithme dans le cadre des codes convolutionnels. Dans un deuxième lieu, nous allons présenter le principe de décodage itératif et le décodeur turbo. [16]

III.2 Principe du maximum a posteriori :

La procédure de décodage d'un code turbo est effectué par les algorithmes du Maximum de Vraisemblance (Maximum Likelihood : **ML**), comme le Soft Output Viterbi Algorithm (**SOVA**), ou l'algorithme de Maximum a Posteriori (**MAP**) [17].

Typiquement, les algorithmes de **ML** sont employés pour estimer la séquence d'information la plus probable qui a été transmis, alors que l'algorithme **MAP** est employé quand les bits d'information les plus probables déjà transmis ont besoin d'être estimé [17] [18].

Dans le cas du **MAP** on parlera de la règle du maximum a posteriori c.à.d. de prendre le maximum des probabilistes a posteriori (**APP**) du symbole ou bit envoyé dans le canal. Le calcul de **l'APP** se base sur l'observation de la séquence reçue, d'où la nomination "a posteriori". Nous allons maintenant expliquer le terme de maximum a posteriori. Dans notre cas, celui de l'algorithme **MAP**, nous parlerons de décision pondérée.
Soient les événements $A_i = 1,2, \dots, M$ Ceci constitue l'ensemble des messages transmis dans un intervalle de temps donné, $p(A_i)$ représente la probabilité a priori de ces événements.
Maintenant, considérons **B**, le signal reçu, qui n'est qu'un des A_i corrompu par du bruit. B est une variable aléatoire continue de densité de probabilité $p(B)$, La probabilité a posteriori (**APP**) est définie par la probabilité que nous reconnaissions le message A_i étant donné que

nous avons reçu le message **B**. Le détecteur **MAP** cherche à maximiser la probabilité a posteriori $p(A_i/B)$. Cette dernière s'exprime sous la forme [10] [11] :

$$\boldsymbol{p(A_i/B) = \frac{p(B/A_i)\,p(A_i)}{p(B)}} \qquad (\mathbf{3.1})$$

Où $p(B/A_i)$ est la fonction de densité de probabilité conditionnelle de B sachant que le vecteur A_i a été transmis, et $p(A_i)$ est la probabilité a priori du ième signal transmis. Le dénominateur de l'équation (3.1) est donné par :

$$\boldsymbol{p(B) = \sum_{i=0}^{M} p(B/A_i)\,p(A_i)} \qquad (\mathbf{3.2})$$

Notons que la probabilité $p(B/A_i)$ est appelée aussi la fonction de vraisemblance. Le critère de décision qui consiste à prendre le maximum de $p(B/A_i)$ parmi les M signaux est le principe du **ML**. Si la distribution des signaux A_i est uniforme (c.à.d. les signaux A_i sont équiprobables), $p(B)$ est indépendant du signal A_i transmis, donc maximiser $p(A_i/B)$ revient à maximiser l'expression $p(B/A_i)$. Donc le détecteur basé sur le principe **MAP** fera la même décision que celui basé sur le principe **ML**.

Dans notre étude les événements A_i appartiennent au champ de Galois GF (2) dont les éléments sont $\{0, 1\}$ où 0 est l'élément neutre sous l'addition $\oplus$, donc M=2. Essayons maintenant de définir cette probabilité plus adéquatement. En notons R la séquence reçue et d_k, le symbole transmis, cette équation devient :

$$\boldsymbol{p(d_k = i/R) = \frac{p(R/d_k = i)p(d_k = i)}{p(R)} \quad i = .01} \qquad (\mathbf{3.3})$$

L'équation (3.1) développée au paragraphe précédent exprime la probabilité a posteriori. Nous avons déjà mentionné que le principe de l'algorithme **MAP** consiste à choisir le maximum de cette probabilité, Nous voyons que l'équation est différente selon que i=0 ou i=1. La règle de décision de l'algorithme **MAP** consiste alors à comparer les probabilités pour ces deux cas et de prendre le maximum, c'est-à-dire la décision la plus probable. Deux cas peuvent se présenter:

$$\begin{cases} \boldsymbol{p(d_k = 1/R) < p(d_k = 0/R) \rightarrow H_0} \\ \boldsymbol{p(d_k = 1/R) \geq p(d_k = 0/R) \rightarrow H_1} \end{cases} \qquad (\mathbf{3.4})$$

Dans l'équation (3.4) l'hypothèse H_1 signifie que le détecteur **MAP** assigne d_k à la valeur 1 dans le cas où $p(d_k = 1/R)$ est supérieur à $p(d_k = 0/R)$. Dans le cas contraire le détecteur **MAP** choisit $d_k = 0$ ce qui correspond à l'hypothèse H_0. Afin de faire le lien entre l'équation (3.4) et le rapport de vraisemblance, il suffit de remplacer l'équation (3.3) dans l'équation (3.4).Il vient que :

$$\begin{cases} \boldsymbol{\frac{p(R/d_k = 1)p(d_k = 1)}{p(R)} < \frac{p(R/d_k = 0)p(d_k = 0)}{p(R)} \rightarrow H_0} \\ \boldsymbol{\frac{p(R/d_k = 1)p(d_k = 1)}{p(R)} \leq \frac{p(R/d_k = 0)p(d_k = 0)}{p(R)} \rightarrow H_1} \end{cases} \qquad (\mathbf{3.5})$$

Nous pouvons alors simplifier cette double inéquation :

$$\begin{cases} \boldsymbol{p(R/d_k = 1)p(d_k = 1) < p(R/d_k = 0)p(d_k = 0) \rightarrow H_0} \\ \boldsymbol{p(R/d_k = 1)p(d_k = 1) \geq p(R/d_k = 0)p(d_k = 0) \rightarrow H_1} \end{cases} \quad (3.6)$$

Que nous pouvons mettre aussi sous la forme plus conventionnelle :

$$\begin{cases} \boldsymbol{\dfrac{p(R/d_k = 1)p(d_k = 1)}{p(R/d_k = 0)p(d_k = 0)} < 1 \longrightarrow H_0} \\ \boldsymbol{\dfrac{p(R/d_k = 1)p(d_k = 1)}{p(R/d_k = 0)p(d_k = 0)} \geq 1 \longrightarrow H_1} \end{cases} \quad (3.7)$$

En général, la probabilité que l'on ait a priori un bit "1" ou un bit "0" à l'entrée est équiprobable. Il est donc justifié de dire que :

$$\boldsymbol{p(d_k = 1) = p(d_k = 0)} \quad (3.8)$$

Ce qui nous simplifie de façon intéressante l'équation (3.7):

$$\begin{cases} \boldsymbol{\dfrac{p(R/d_k = 1)}{p(R/d_k = 0)} < 1 \longrightarrow H_0} \\ \boldsymbol{\dfrac{p(R/d_k = 1)}{p(R/d_k = 0)} \geq 1 \longrightarrow H_1} \end{cases} \quad (3.9)$$

Grâce à la règle de Bayes, on peut écrire :

$$\begin{cases} \boldsymbol{p(R/d_k = 1) = \dfrac{p(d_k = 1/R)p(R)}{p(d_k = 1)}} \\ \boldsymbol{p(R/d_k = 0) = \dfrac{p(d_k = 0/R)p(R)}{p(d_k = 0)}} \end{cases} \quad (3.10)$$

Sachant que les bits "0" et "1" sont équiprobables, en divisant ces deux termes, (3.10) devient:

$$\begin{cases} \boldsymbol{\dfrac{p(d_k = 1/R)}{p(d_k = 0/R)} < 1 \rightarrow H_0} \\ \boldsymbol{\dfrac{p(d_k = 1/R)}{p(d_k = 0/R)} \geq 1 \rightarrow H_1} \end{cases} \quad (3.11)$$

Ceci constitue ce que nous appelons le rapport de vraisemblance. Le fait que ce soit supérieur ou inférieur à "1" implique qu'il serait judicieux de prendre le logarithme de cette expression et de comparer à zéro. Ainsi la règle de décision devient :

$$\boldsymbol{LLR(d_k) = \log\left[\dfrac{p(d_k = 1/R)}{p(d_k = 0/R)}\right]} \quad (3.12)$$

Ceci veut donc dire que:

$$\begin{cases} \boldsymbol{si\ LLR(d_k) \geq 0\ le\ bit\ d\acute{e}cod\acute{e}\ est\ "1"} \\ \boldsymbol{si\ LLR(d_k) < 0}\ le\ bit\ d\acute{e}cod\acute{e}\ est\ "0" \end{cases} \quad (3.13)$$

Ce rapport de vraisemblance est la base de l'algorithme **MAP** que nous allons développer.

III.3 Probabilité conjointe :

Nous allons maintenant appliquer ce que nous avons déjà introduit au cas des codes turbo. Pour ce faire, nous allons considérer un codeur systématique récursif de taux de codage R=1/2 (voir figure II.5). Nous allons adopter les notations suivantes :

- $d_k = i \;\; i = 0.1$, représente le bit d'information présent à l'entrée du codeur à l'instant k.
- M=K-1 bits, la mémoire du codeur.
- $S_k = m\,, m = 0.1 \ldots \ldots, 2^M$, représente l'état du codeur à l'instant K
- *N*, est la taille de la séquence à coder (ou encore la longueur du bloc exprimée en bits).
- $(d_1, \ldots, d_k, \ldots, d_N) = (Y_1^s, Y_k^s, Y_N^s)$. la séquence à coder (aussi la séquence de l'information systématique).
- $(Y_1^p, \ldots, Y_k^p, \ldots, Y_N^p)$ la séquence composée des symboles de parité à la sortie du codeur CRS.
- $R_k = (r_k^s, r_k^p)$, la version bruitée reçue de (Y_k^s, Y_k^p) à l'instant k.
- $R_1^N = (R_1, \ldots, R_k, \ldots, R_N)$ la séquence bruitée reçue de longueur N

Avec ces notations, l'équation du rapport de vraisemblance devient :

$$\boldsymbol{LLR(d_k) = \log\left[\frac{p(d_k = 1/R_1^N)}{p(d_k = 0/R_1^N)}\right]} \quad (3.14)$$

Le calcul de la probabilité a posteriori $p(d_k = i/R_1^N)\,, i = 1{,}0$ s'avère complexe d'où l'idée d'introduire la probabilité conjointe (JD) définie par:

$$\boldsymbol{\lambda_k^i(m) = p(d_k = i, S_k = m/R_1^N)} \quad (3.15)$$

Etant donné que nous avons 2^M états, donc **l'APP** d'un bit décodé d_k . peut s'exprimer en fonction de la JD et est égale à :

$$\boldsymbol{p(d_k = i/R_1^N) = \sum_{m=0}^{2^M-1} \lambda_k^i(m)} \quad (3.16)$$

Où *i=1,0* et la sommation est sur les 2^M états du codeur. En remplaçant l'expression de la probabilité conjointe dans l'équation (3.14), nous obtenons :

$$\boldsymbol{LLR(d_k) = \log\left[\frac{\sum_{m=0}^{2^M-1} \lambda_k^1(m)}{\sum_{m=0}^{2^M-1} \lambda_k^0(m)}\right]} \quad (3.17)$$

Maintenant que nous avons introduit la notion de probabilité conjointe, nous pouvons décrire l'algorithme **MAP**, ce que nous allons faire dans la prochaine section. [16]

III.4 Algorithme MAP utilisé :

Dans cette partie, nous allons introduire des nouvelles notions qui vont nous permettre de mieux calculer le LLR. Nous allons introduire les notions de métrique d'état en avant, de métrique d'état en arrière et de métrique de branche. En tout premier lieu, nous allons donner quelques définitions nécessaires au développement de l'algorithme **MAP**.

III.4.1 Définitions :

Nous définissons ici trois notions essentielles au développement de la probabilité conjointe et de ce fait, à la construction de l'algorithme **MAP**. Ces trois notions sont :

- La métrique d'état en avant (FSM):

$$\begin{cases} \boldsymbol{a_k(m) = p\left(R_1^{k-1}/d_k = i, S_k = m, R_k^N\right)} \\ \boldsymbol{a_k(m) = p\left(R_1^{k-1}/S_k = m\right)} \end{cases} \quad (\mathbf{3.18})$$

Car les événements après l'instant K n'influencent en rien les événements avant K.

- la métrique d'état en arrière (BSM):

$$\boldsymbol{\beta_k(m) = (R_k^N/S_k = m)} \quad (\mathbf{3.19})$$

- la métrique de branche (BM):

$$\boldsymbol{\gamma_k^i(m) = p(d_k = i, S_k = m, R_1^k)} \quad (\mathbf{3.20})$$

III.4.1. Expression de la métrique d'état en avant (FSM) :

Nous pouvons remarquer que la FSM à l'instant K dépend de la séquence reçue R_1^k qui précède l'instant K (ou au plus jusqu'à l'instant K). En utilisant la règle de Bayes, nous pouvons exprimer la métrique d'état en avant sous la forme :

$$\boldsymbol{\alpha_k(m) = p\left(R_1^{k-1}/S_k = m\right)} \quad (\mathbf{3.21})$$

$$\boldsymbol{\alpha_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} p(d_{k-1} = j, S_{k-1} = i, R_1^{k-1}/S_k = m)} \quad (\mathbf{3.22})$$

$$\boldsymbol{\alpha_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} p\left(R_1^{k-2}/S_k = m, d_{k-1} = j, S_{k-1} = i, R_{k-1}\right) * p(d_{k-1} = j, S_{k-1} = i, R_{k-1}/S_k = m).} \quad (\mathbf{3.23})$$

Sachant que le passage de l'état $S_{k-1} = m$ a été provoqué par le bit $d_{k-1} = j$, nous pouvons compacter ces informations sous la forme $S_{k-1} = b(j, m)$ Ceci signifie que l'état S_{k-1} est l'état qui précède l'état $S_k = m$ sachant que le bit d'entrée au codeur à l'instant K-1 est égale à j.

$$\alpha_k(m) = \sum_{j=0}^{1} p\left(R_1^{k-2}/S_{k-1} = b(j,m)\right) p(d_{k-1} = j, S_{k-1} = b(j,m), R_{k-1})$$

$$= \sum_{j=0}^{1} \alpha_{k-1}\left(b(j,m)\right) \gamma_{k-1}^{j}\left(b\,(j,m)\right) \qquad (3.24)$$

À partir de cette dernière équation, nous constatons que cette métrique d'état à l'instant k s'exprime en fonction de la même métrique d'état à l'instant précédent. Ceci explique pourquoi nous lui donnons le nom de métrique en avant : il faut connaître les instants précédents, donc la valeur initiale pour pouvoir la calculer. [15]

III.4.2 Expression de la métrique d'état en arrière (BSM)

Au paragraphe précédent, nous avons développé l'expression de la métrique en avant. Nous Intéressons maintenant à la deuxième métrique, à savoir la métrique d'état en arrière. Nous avons également conclu que la métrique d'état en avant dépendait des instants précédents. II en est différemment de la métrique d'état en arrière comme l'indique son nom. En fait, cette métrique, à l'instant k**,** dépend de la séquence reçue (instant N) après l'instant k. Nous avons défini cette métrique comme :

$$\beta(m) = p\left(R_k^N/S_k = m\right) \qquad (3.25)$$

$$\beta_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} p(d_k = j, S_{k+1} = i, R_k^N/S_k = m) \qquad (3.26)$$

$$\beta_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} p\left(R_{k+1}^N/S_k = m, d_k = j, S_{k+1} = i, R_k\right) * p(d_k = j, S_{k+1} = i, R_k/S_k = m) \qquad (3.27)$$

Comme pour la métrique d'état en avant, nous pouvons définir une fonction $S_{k+1} = f(j,m)$ de façon à simplifier cette expression. Il vient alors que : [16]

$$\beta_k(m) = \sum_{j=0}^{1} p\left(R_k^N/S_{k+1} = f(j,m)\right) p(d_k = j, S_k = m, R_k) \qquad (3.28)$$

$$\beta_k(m) = \sum_{j=0}^{1} \gamma_k^j(m)\, \beta_{k+1}\left(f\,(j,m)\right) \qquad (3.29)$$

Nous en déduisons que cette métrique à l'instant k est fonction de la même métrique à l'instant k+1**,** d'où la nomination de métrique en arrière, étant donné qu'il faut connaitre l'instant k = N pour reculer dans le treillis et calculer notre métrique (elle dépend des conditions terminales). [15]

III.4.3. Calcul de la métrique de branche (BM) :

Comme nous l'avons indiqué précédemment le calcul de la métrique de branche (**BM**) est nécessaire afin de calculer le logarithme du rapport de vraisemblance. D'après l'expression de $\boldsymbol{\gamma}_k^i$ (Équation (3.20)), la **BM** à l'instant k dépend seulement de l'instant même et non plus des instants précédent ou suivant d'où la nomination métrique de branche. Nous avons vu que :

$$\boldsymbol{\gamma_k^i(m) = p(d_k = i, S_k = m, R_k)} \qquad \mathbf{(3.30)}$$

Donc, toujours grâce à la règle de Bayes:

$$\boldsymbol{\gamma_k^i(m) = p(R_k/d_k = i, S_k = m) * p(d_k = i, S_k = m)} \qquad \mathbf{(3.31)}$$

$$\boldsymbol{\gamma_k^i(m) = p(R_k/d_k = i, S_k = m) * p(S_k = m/d_k = i) * p(d_k = i)} \qquad \mathbf{(3.32)}$$

En remplaçant $R_k = (r_k^s, r_k^p)$ dans l'équation précédente nous obtenons :

$$\boldsymbol{\gamma_k^i(m) = p(r_k^s \backslash d_k = i, S_k = m) * p\left(r_k^p \backslash d_k = i, S_k = m\right) * p(S_k = m \backslash d_k = i) * p(d_k = i)} \qquad \mathbf{(3.33)}$$

Les deux premiers termes de la métrique de branche dépendent des symboles recueillis à la sortie du canal. Donc le type de canal utilisé affecte l'expression de la BM. [16]

III.5. Décodage itératif :

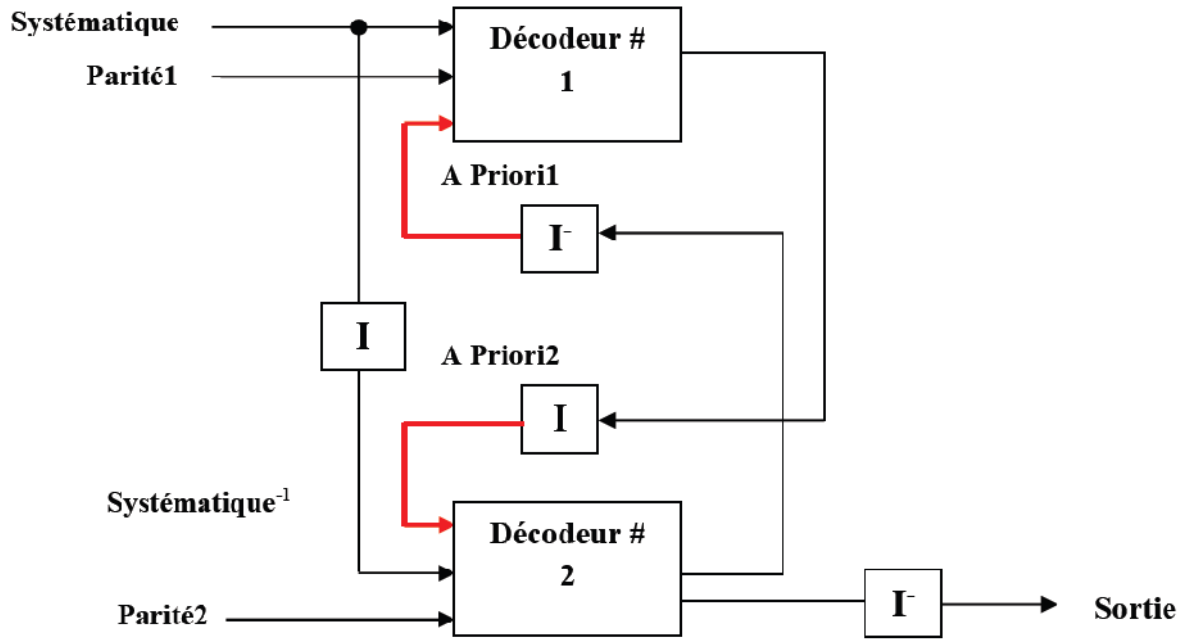
Nous avons présenté l'algorithme **MAP** qui est utilisée pour le décodage des codes turbo sans toutefois traiter l'architecture d'un décodeur turbo. En examinant soigneusement le codeur turbo, nous pourrions nous attendre à avoir autant de décodeurs **MAP** que de codeurs CRS. Les deux décodeurs sont en série et sont séparés par des entrelaceurs du même type que ceux utilisés au codage. Chaque décodeur va se concentrer surtout sur les symboles provenant du codeur qui lui est associée. Les deux décodeurs vont s'échanger de l'information douce qui inclue une estimation de la vraisemblance sur l'information commune (symbole systématique) au codage. Ce va et vient sous forme d'échange de l'information qui va se répéter jusqu'à l'obtention des performances requises nous amène tout droit vers le principe itératif.

Dans le reste de ce chapitre, nous allons d'abord développer et préciser le type d'information qui est échangée entre les décodeurs **MAP** composant le décodeur turbo. Ensuite une description détaillée du décodeur turbo sera présentée. L'information échangée est considérée comme une donnée ou variable supplémentaire que le décodeur reçoit. Donc ceci modifie l'expression des paramètres de décodage (**BSM**, **FSM** et **BM**) mais c'est surtout la métrique de branche (**BM**) qui va inclure dans son expression cette information échangée entre les décodeurs. [16]

III.5.1. Principe de décodage itératif :

Le principe de décodage itératif est très simple. Il s'agit de répéter plusieurs fois l'échange d'information pondérée sur les bits d'information entre les deux décodeurs jusqu'à l'obtention des performances de décodage désirées. En fait, le décodage itératif se fait en plusieurs itérations. Chaque itération de décodage implique deux décodeurs. Nous avons vu que les symboles fournis par le canal au décodeur sont de trois sortes. Il s'agit en effet des symboles

d'information et des symboles de parité générés par le premier codeur et ceux par le deuxième. Le décodage itératif va donc prendre ces trois types de symboles et minimiser la probabilité d'erreur par bit. Le principe de décodage itératif est illustré à (la figure III.1) ci-dessous :



I: entrelaceur , I^{-}: désentreleceur

Figure III.1 : principe de décodage itératif.

La technique de décodage illustrée par cette figure se fait par itérations de deux étapes. En effet, comme nous le voyons, contrairement au codage, les blocs décodeurs sont en série et non en parallèle, donc chaque décodeur agit l'un après l'autre.

Pour une première itération et après avoir décodé les symboles associés au premier codeur, le décodeur 1 fait passer une information vers le décodeur 2. Ce dernier dispose de cette information ainsi que des symboles bruités issus du deuxième codeur pour pouvoir corriger certaines erreurs qui n'ont pas été détectés par le décodeur 1. A la deuxième itération, si le décodeur 1 peut avoir accès à plus d'informations alors sûrement il fournira des performances meilleures que celles obtenues durant la première étape. Autrement dit, le décodeur 1 reçoit en plus des mêmes symboles bruités issus du premier codeur, une information provenant du décodeur 2. Ainsi, d'une itération a une autre, le processus d'échange d'information se répète jusqu'à une saturation où les deux décodeurs ne peuvent pas améliorer la performance d'erreur. La séquence d'information supplémentaire échangée qui se présente à l'entrée de l'un des deux décodeurs s'appelle séquence d'information à priori. Elle doit être indépendante de toutes les entrées de l'autre décodeur. En plus, elle doit contenir des symboles erronés qui sont réordonnés et non regroupés ensemble afin d'augmenter l'efficacité de décodage. D'où l'intérêt de la présence de l'entrelaceur ou de celle du désentrelaceur entre les deux décodeurs. [16]

III.6.conclusion :

Dans ce chapitre, nous avons présenté l'algorithme **MAP** que nous avons utilisé pour le décodage turbo de nos simulations. Ce dernier a été choisi car il permet la minimisation de la probabilité d'erreur par bit ainsi que la génération d'information de fiabilité pour chaque bit. II est important de noter que cet algorithme est complexé et qu'il induit un certain délai dans le décodage. Nous avons également vu qu'il existait des simplifications à cet algorithme. Enfin**,** il est important de savoir que d'autres algorithmes, comme le **SOVA** sont disponibles pour le décodage turbo, même si nous ne l'avons pas utilisé.

Le choix de l'algorithme de décodage est extrêmement important pour le décodage turbo**.** Néanmoins, il n'est pas le seul paramètre à prendre en compte étant donné que ce décodage est très complexé.

Chapitre IV :

Programmation et discussion des résultats.

Chapitre IV :

IV.1.introduction :

Dans cette partie de travail, nous allons présenter les résultats obtenus. Commençant par une présentation générale d'un algorithme de codage et décodage, puis nous présentons les différentes étapes de conception de notre codeur et du décodeur. Le guide de l'utilisation est disponible dans les deux organigrammes.

Les codes Turbo sont Également connus sous le nom de codes concaténés parallèles et la complexité du décodage est faible pour la dimension du code, Alors que la longueur du code est très longue.

Les codes turbo sont dérivés des codes de convolution parentale. Ils sont une classe de codes de correction d'erreur directe Les codes Turbo utilisent le codeur Récursive Systématique Convolution (CRS) et le décodeur Itérative BCJR (Bahl Cocke Jelinek Raviv). Sa performance est bien meilleure que le code de convolution correspondant. Et tout en raison de la complexité du décodage Devient petit

Donc on va aussi présent dans Ce chapitre les étapes de travail sure le passage de message par un codeur de décodeur turbo dans un canal AWGN et voire les résultats de code correcteur d'erreurs par le turbo code.

IV.2. Principe de turbo code :

Le principe des turbos codes, comme tout code correcteur d'erreur, est d'introduire une redondance dans le message afin de le rendre moins sensible aux bruits et perturbations subies lors de la transmission. Le codage consiste à utiliser deux codeurs simples, dont les entrées ont été entrelacées ; ainsi, chaque codeur voit une série d'informations différentes à son entrée.
Le décodage est une collaboration entre les décodeurs, chacun donnant son « avis » (notion de confiance) sur chaque bit décodé. Cette information est ensuite fournie à l'entrée du prochain décodeur, et ainsi de suite. D'où l'appellation « turbo ».
Donc on va implémenter notre signal codes par un code correcteur d erreurs qui va être le encodeur turbo et avent on le passer dans canal AWGN on applique la modulation BPSK
Au début on va coder notre signal avec un codeur qui s appel le codeur turbo.

IV.3 Le codeur Turbo:

Le codeur Turbo est composé d'un codeur ½ CRS de 2 vitesses, comme indiqué sur (la figure IV.1). Le premier codeur prend les bits d'information d'entrée et génère des bits de parité L'entrelaceurs entrelace les bits d'information pour générer des informations intercalées Le deuxième codeur utilise et génère des bits de parité.

Encodeur turbo (CRS-0 et CRS-1) :

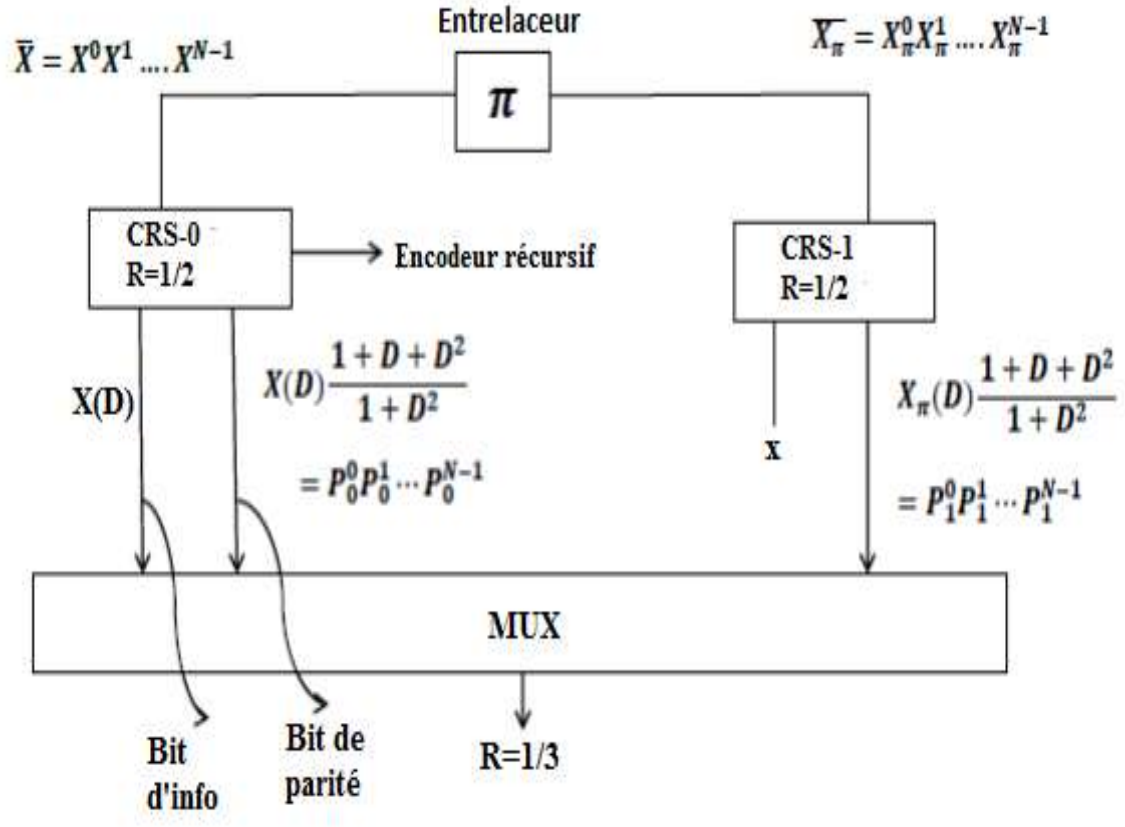


Figure IV.1 : architecture d'un turbo encodeur pour code $\mathbf{G(D)} = [\mathbf{1}\ \frac{\mathbf{1+D+D^2}}{\mathbf{1+D^2}}]$.

Chaque codeur 1/2 CRS faire le travail suivent :

$$\mathbf{G(D)} = \left[\mathbf{1}\ \frac{\mathbf{1+D+D^2}}{\mathbf{1+D^2}}\right] \qquad (\mathbf{4.1})$$

Nous donne : $\boldsymbol{C_0(D) = X(D)\ et\ C_1(D) = X(D)\frac{1+D+D^2}{1+D^2}}$ $(\mathbf{4.2})$

On suppose que : $\boldsymbol{F(D) = \frac{X(D)}{1+D^2}}$ $(\mathbf{4.3})$

Donc on obtient : $\boldsymbol{C_0^j = X^j et\ C_1^j = X^j + F^{j-1} ou\ F^j = F^{j-2} + X^j}$ $(\mathbf{4.4})$

Si nous considérons : (F^{j-1}, F^{j-2}).

Pour l'état actuel, le diagramme d'état (Figure VI.2) et le diagramme Treillis (Figure VI.3) sont obtenus :

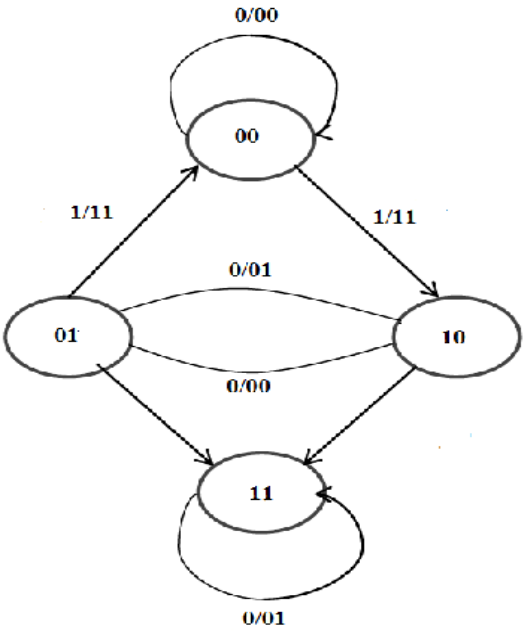


Figure IV.2 : diagramme en state.

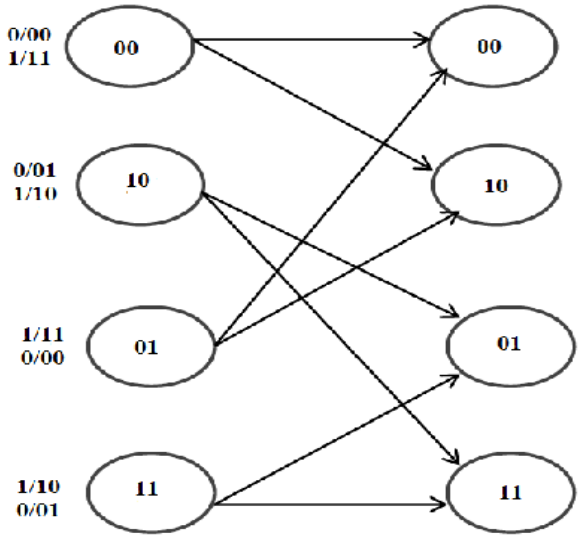


Figure IV.3 : Diagramme en Treillis.

Organigramme des étapes de codage :

Pour stockage de BER simulé (chaque colonne correspond à une itération).

Pour tous les cas l'état 0, 1, 2,3 représente 00, 01, 10,11 respectivement.

Former les bites de parité à la forme matricielle.

Initialisation du 1er êta 00.

Intercalation des bites d'entrer pour l'encodeur CRS-1.

Modélisation BPSK.

IV.4. Transmission de symboles modulés par BPSK par canal AWGN:

Modulation utilisant la modulation BPSK:

$$u_0^j = 2X^j - 1 \qquad (4.5)$$

$$u_1^j = 2P_0^j - 1 \qquad (4.6)$$

$$u_2^j = 2P_1^j - 1 \qquad (4.7)$$

Pour j=0,1,…... N-1.

La transmission vers AWGN :

$$R_i^j = \sqrt{P}u_i^j - n_i^j \qquad (4.8)$$

Pour j=1,2,…, N-1 et i=0, 1,2. Ici P c'est un signal de puissance.

Les étapes de codage sont illustrées dans l'organigramme suivant :

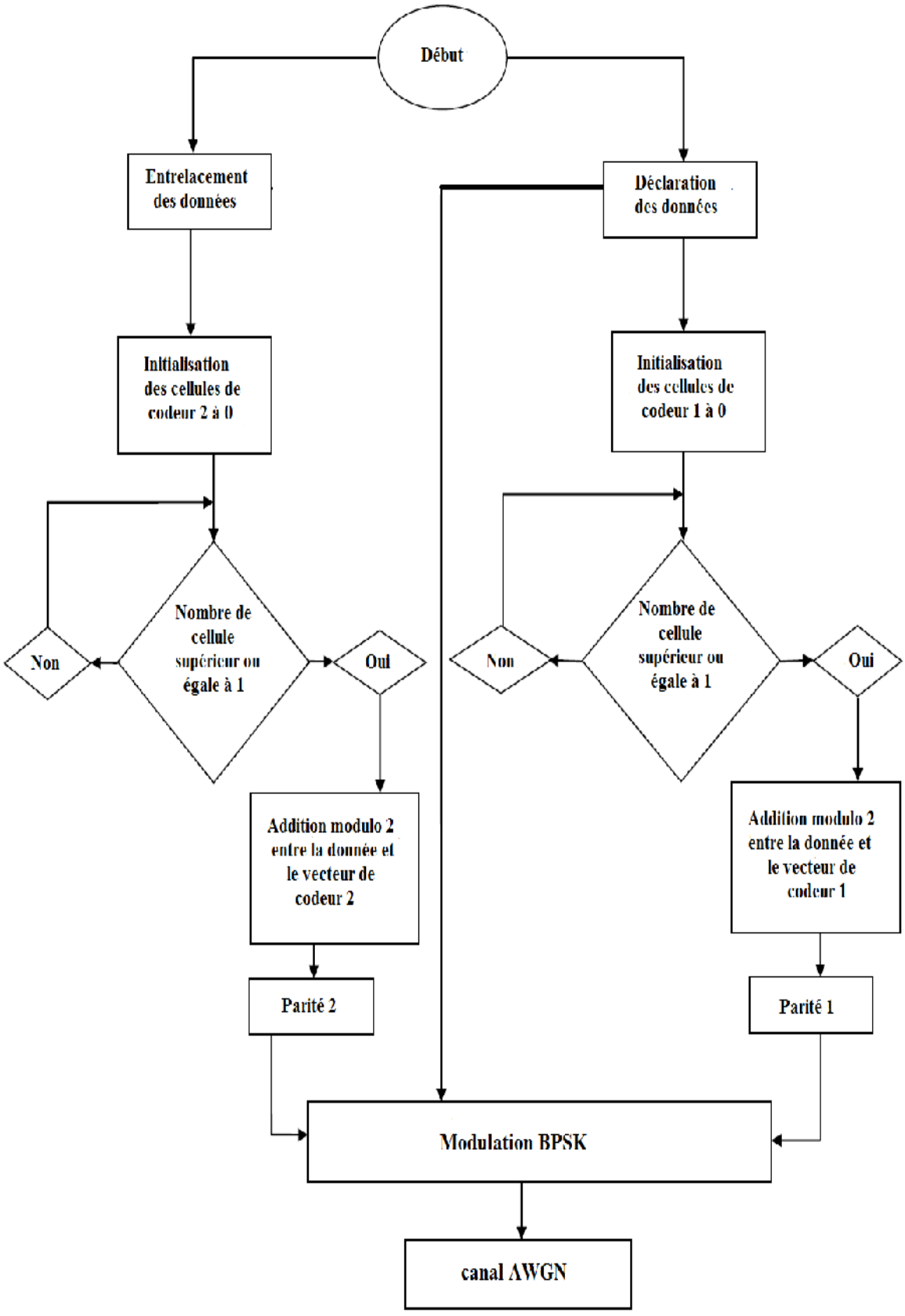


Figure IV.4 : principe de codage turbo

IV.5. Décodeur turbo :

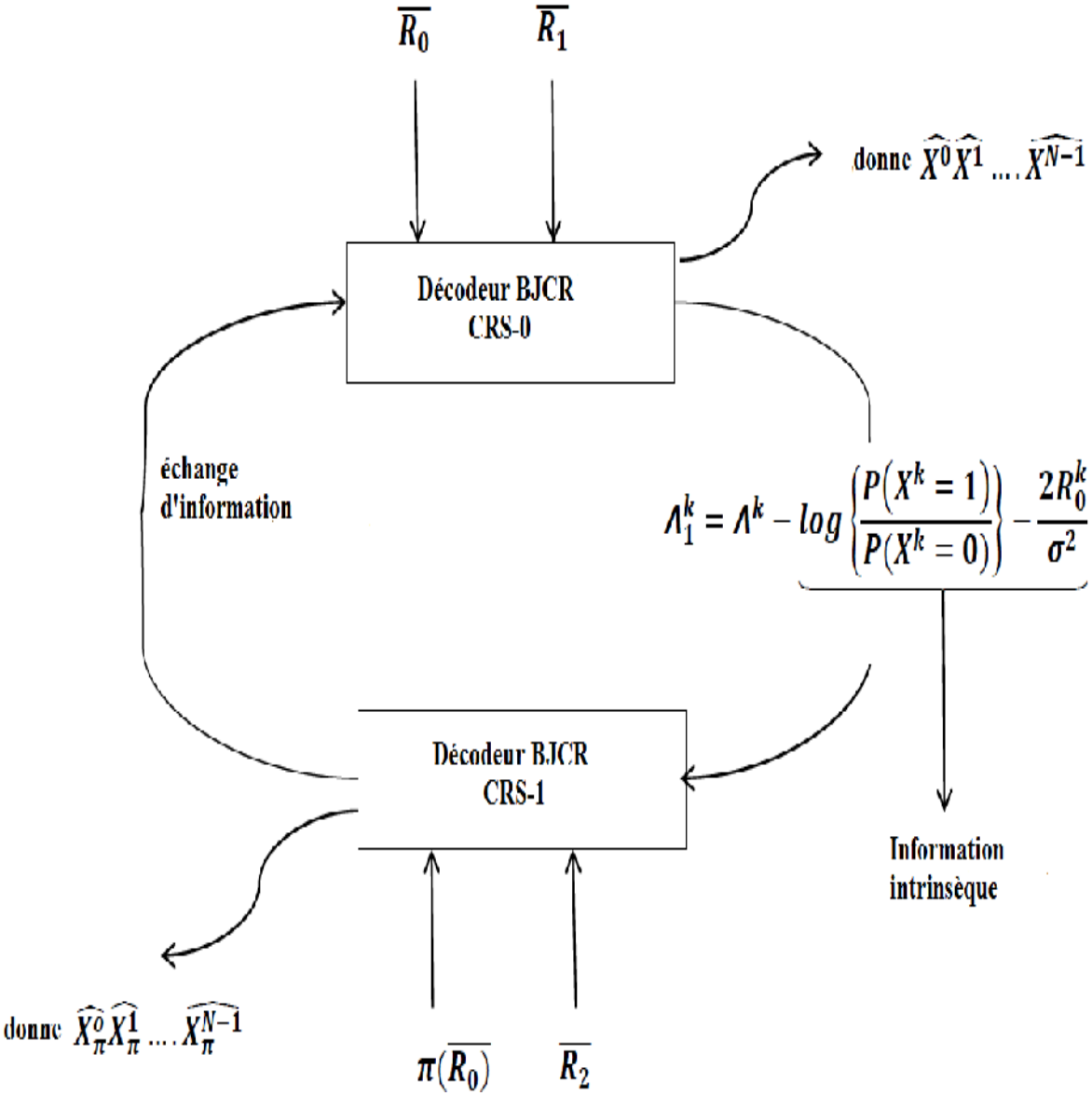


Figure IV.5 : Architecture d'un décodeur BJCR.

La première étape sera effectuée par BCJR-0

Première ligne pour la probabilité. (De i/p 0 et deuxièmes lignes pour la probabilité. De i/p 1)

Initialisation de tous les prieurés à ½ (0,5).

Si BCJR est 0, passez R0 et R1 pour calculer GAMMA.

Si BCJR est 1, passez R0_pi et R2 pour calculer GAMMA.

Parcelles pour BER simulé

IV.6. Conception d'algorithme :

Après réception de bits transmis au récepteur avec bruit blanc gaussien additif (AWGN), nous commençons par l'initialisation de probabilités à priori à 0,5 pour l'entrée 0 et 1. Ensuite, nous calculons les probabilités $\boldsymbol{\gamma}$ à toutes les étapes pour les deux entrées. L'initialisation des probabilités $\boldsymbol{\alpha}$ pour première étape se fait en supposant l'encodeur pour être dans l'état 00 initialement. Donc α est initialisé comme suite :

α0 (00) = 1, α0 (01) = 0, α0 (10) = 0, α0 (11) = 0.

Ensuite, nous utilisons une relation récursive de α pour calculer α de l'étape suivante.
De mêmes, les probabilités **β** sont initialisés à l'étape final N-1, où N est la longueur du bloc. Si on suppose que le codeur suivant est dans l'état 00 au début puis β peuvent être initialisés que : $\beta_{N-1}(00) = 1, \beta_{N-1}(01) = 0, \beta_{N-1}(10) = 0, \beta_{N-1}(11) = 0.$

Après le calcul de α et β et γ, le rapport Λ^k est calculé pour toutes les étapes. En observant les valeurs de Λ^k qu'ils soient positifs ou négatifs, les bits sont décodés en 1 et 0 respectivement.

Après la première itération du premier décodeur (BCJR-0), les valeurs de Λ^k passent au deuxième décodeur à l'exclusion des informations intrinsèques. Le deuxième décodeur utilise ces probabilités pour calculer les probabilités à priori à toutes les étapes à l'aide de la relation suivante :

$$\boldsymbol{P(X^k = 1) = \frac{e^{\Lambda_1^k}}{1 + e^{\Lambda_1^k}}} \qquad (\mathbf{4.9})$$

$$\boldsymbol{P(X^k = 0) = \frac{1}{1 + e^{\Lambda_1^k}}} \qquad (\mathbf{4.10})$$

Le 2emme décodeur fait le même travaille que le 1er pour chaque bit et dans la simulation on termine lorsque la probabilité priori pour l'un de l'entrée devient 1.

Expression théorique pour BER

Résultats de la simulation.

Les étapes de décodage sont illustrées dans l'organigramme suivant :

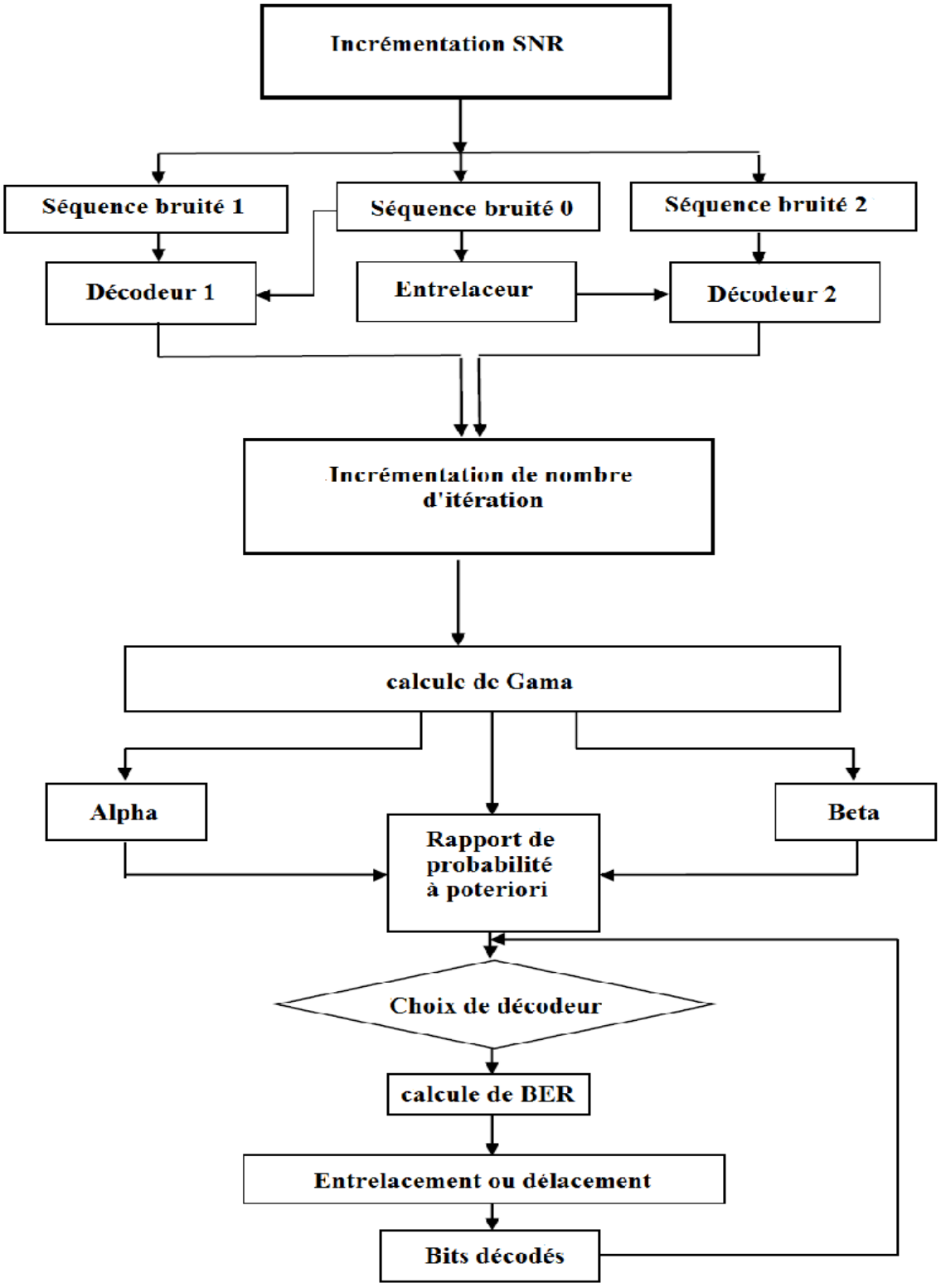


Figure IV.6 : Principe de décodage turbo

VI.7. Résultats de simulation :

Dans la simulation de programme (code) nous définie :

N=10000 bites d'informations.

SNR varie entre 0 et 9 dB.

Nombre d'itérations= 4.

Et toutes es valeurs SNR est incrémenter de 0.5dB.

Donc il soit 19 valeurs de SNR, dans e temps totale d'environ 39 secs.

Le temps pris pour chaque valeurs SNR est 2.05 sec.

Le temps pris pour chaque itérations est 0.51 sec.

On présente le TEB dans les graphes suivants :

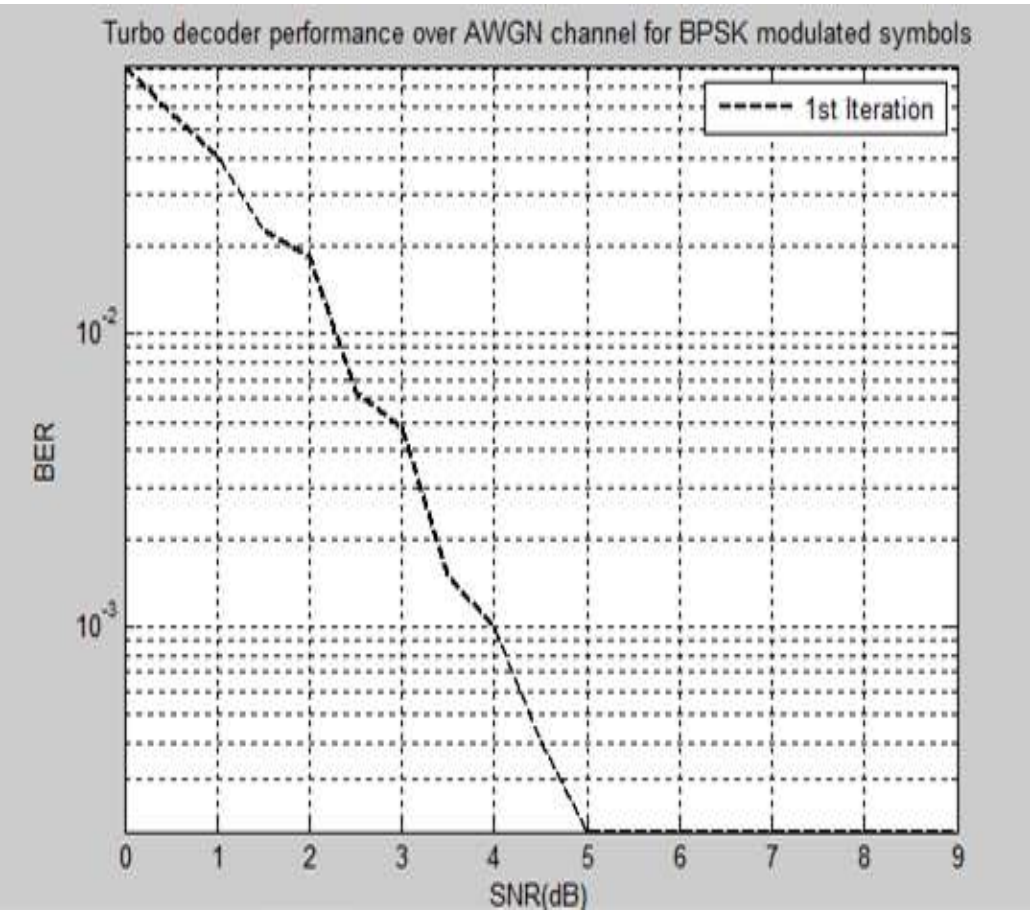


Figure IV.7 : TEB après 1er itération.

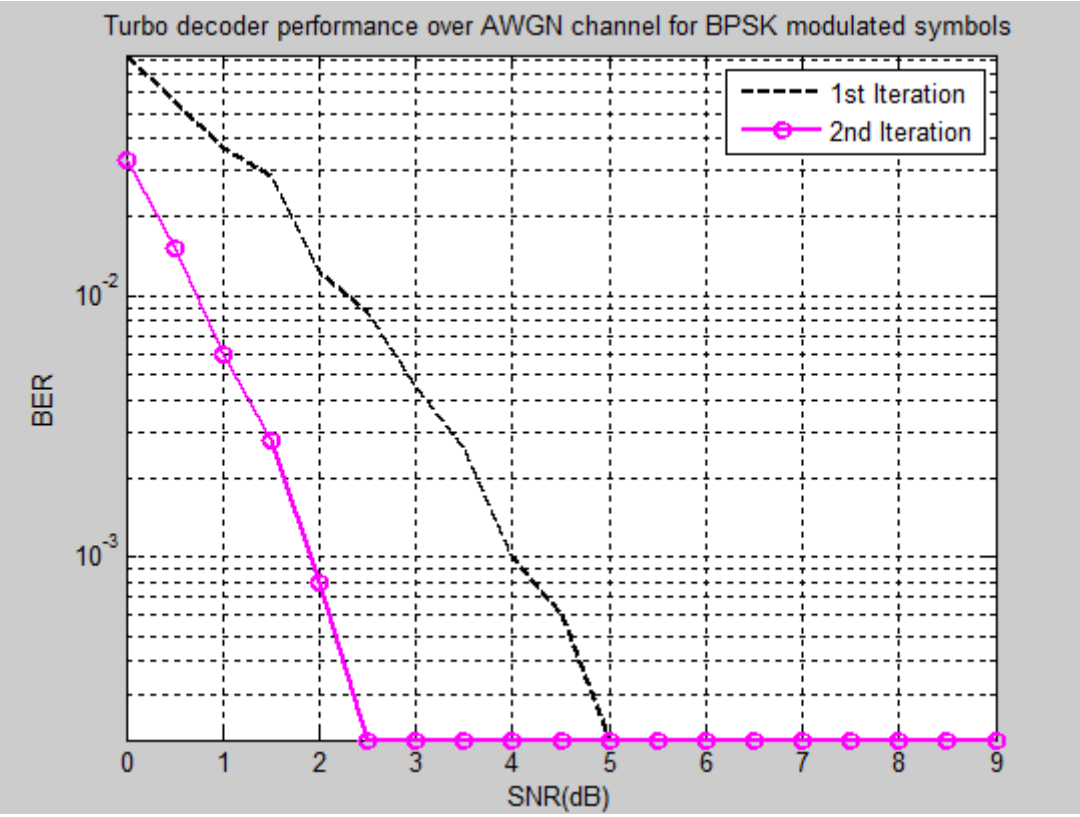


Figure VI.8 : TEB après 2eme itération.

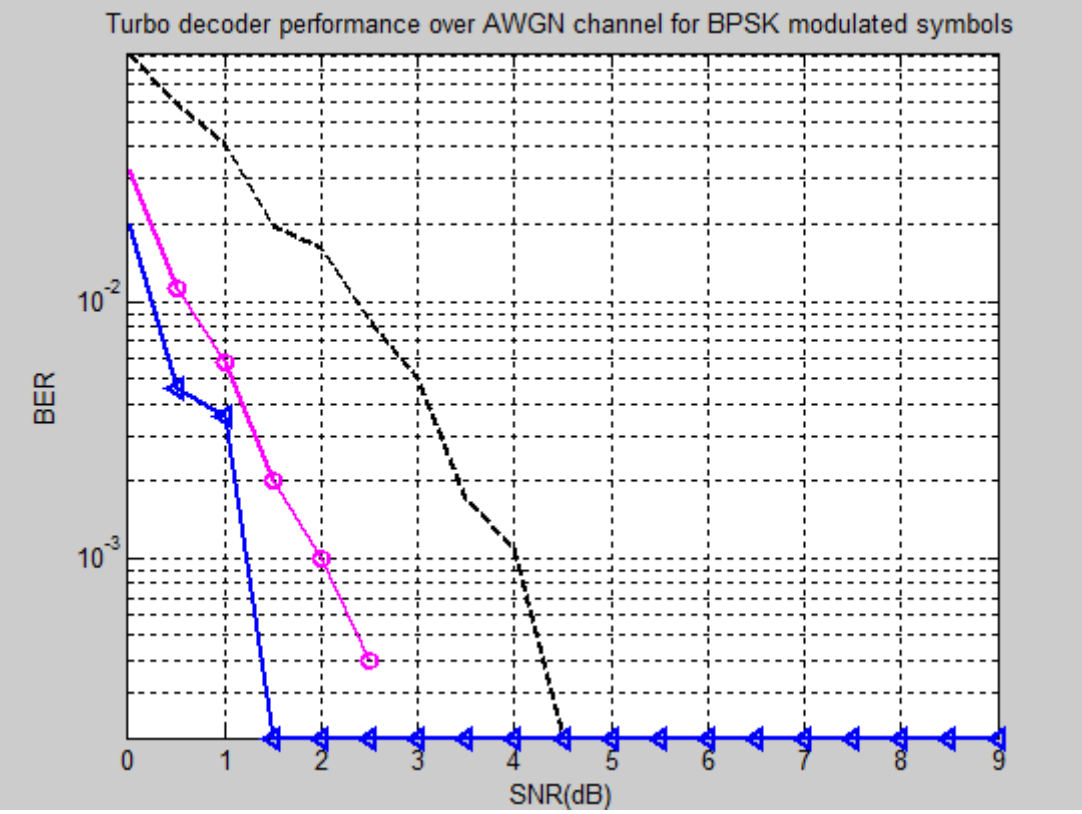


Figure IV.9 : TEB après 3eme itération.

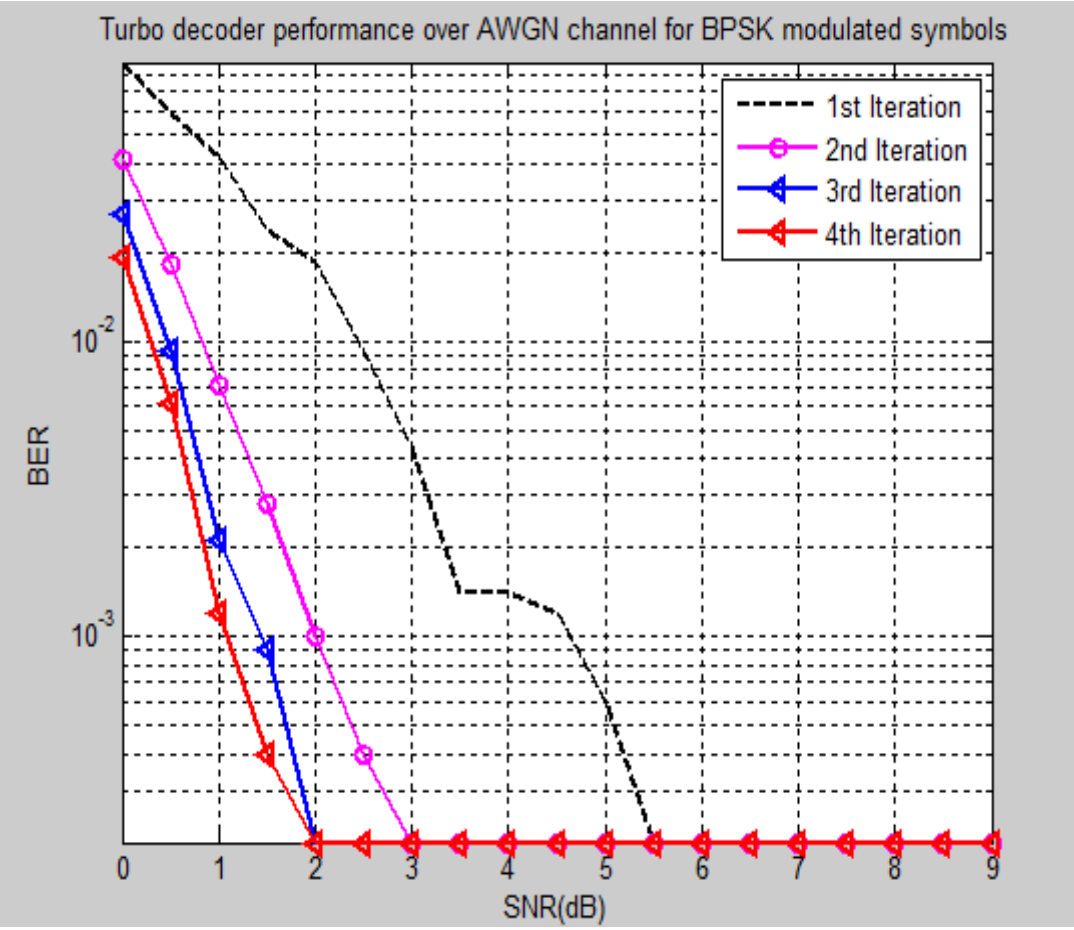


Figure IV.10 : TEB après 4eme itération.

Nous pouvons observer à partir les graphes précédant que TEB après chaque itération diminue pour tous les SNR. Après la 4eme itération, le TEB reste presque la même, de sorte que l'algorithme de BCJR converge après presque 4eme itérations. Aussi le TEB devient presque zéro après 1 dB en 4eme itérations. Donc le code Turbo est beaucoup mieux que code convolutionnel correspondant.

L'expression utilisée pour le TEB théorique est :

$$\boldsymbol{P_e = \sum_{j=0}^{10} 2^{j+1}(j+5)Q(\sqrt{(j+5)SNR})} \qquad (\mathbf{4.11})$$

Q : c'est la fonction de Morcom de probabilité d'erreur.

La comparaison de performances de turbo code est montré dans la Figure IV.11.

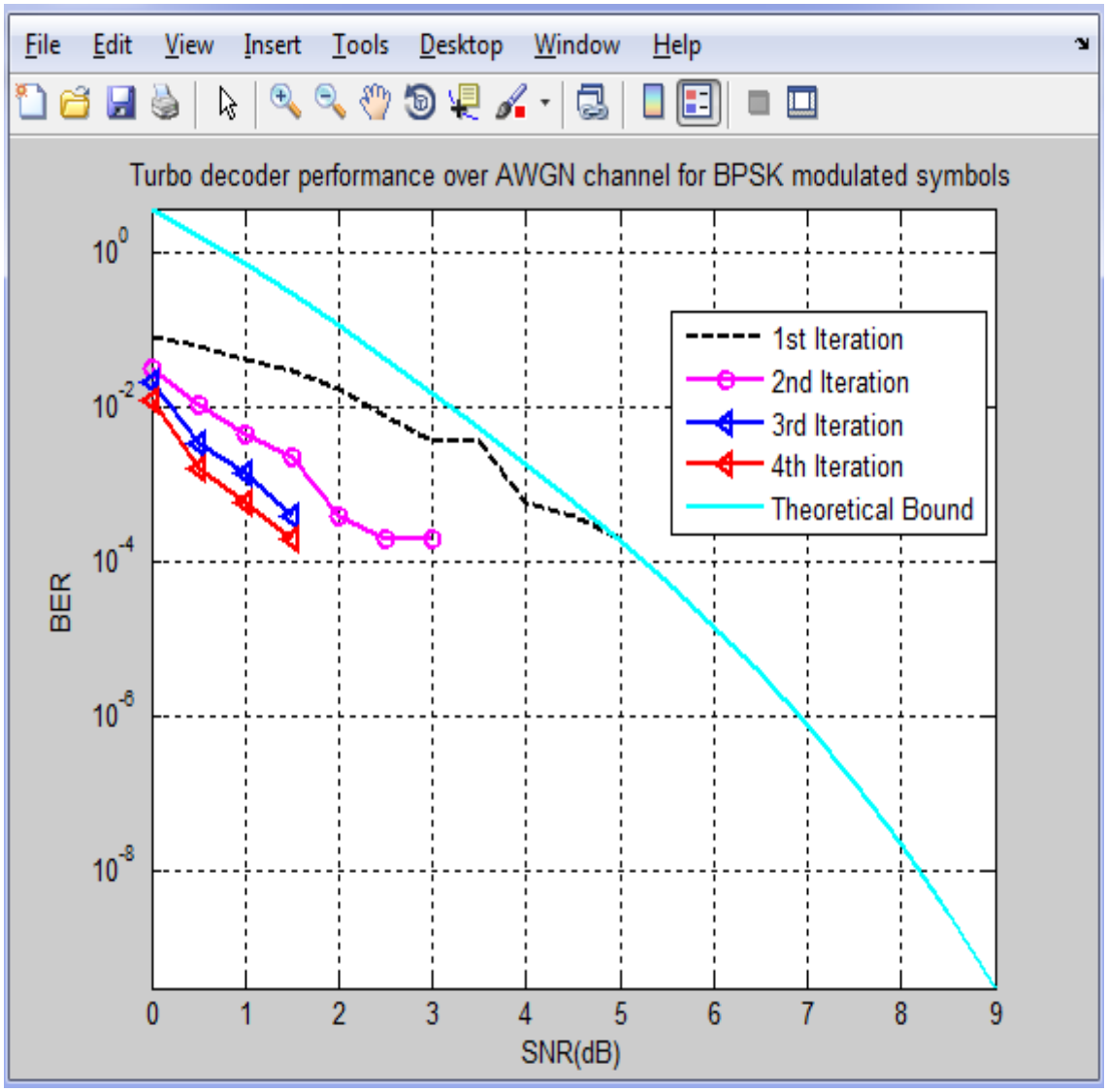


Figure VI.11 : Comparaison de TEB avec la limite théorique.

Les valeurs de TEB pour les 4 itérations et la limite théorique sont présentées au tableau suivant.

Tableau IV.1 : Valeurs de TEB pour 4eme itération et comparaison avec TEB théorique du code convolutionnel correspondant (Certaines valeurs sont supérieures à 1, à cause de bande théorique).

SNR (dB)	**1ere Itération**	**2eme Itération**	**3eme Itération**	**4eme Itération**	**TEB théorique de code convolutionel**
0	0.0000	0.0000	0.0000	0.0000	3.4532
0.5	0.6170	0.0108	0.0033	0.0016	1.6124
1	0.0424	0.0045	0.0014	0.0006	0.7065
1.5	0.0296	0.0022	0.0004	0.0002	0.2911
2	0.0168	0.0002	0.0000	0.0000	0.1134
2.5	0.0075	0.0002	0.0000	0.0000	0.0422
3	0.0037	0.0000	0.0000	0.0000	0.0151
3.5	0.0035	0.0000	0.0000	0.0000	0.0053
4	0.0006	0.0000	0.0000	0.0000	0.0018
4.5	0.0004	0.0000	0.0000	0.0000	0.0006
5	0.0001	0.0000	0.0000	0.0000	0.0002
5.5	0.0000	0.0000	0.0000	0.0000	0.0001
6	0.0000	0.0000	0.0000	0.0000	0.0000
6.5	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.0000	0.0000	0.0000	0.0000	0.0000
7.5	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.0000	0.0000	0.0000	0.0000	0.0000
8.5	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0000	0.0000	0.0000	0.0000	0.0000

IV.8. Conclusion :

À partir de cette simulation le TEB de turbo code, on peut observer que la performance de décodeur BCJR itératif est beaucoup mieux que le décodage de l'algorithme Viterbi pour les codes convolutionnel. L'algorithme BCJR est capable de décoder les bits transmis avec 0 probabilité d'erreur même à très bas SNR.

Conclusion générale :

Dans ce mémoire, nous nous sommes intéressés à l'étude des codes turbo, Les codes turbo font partie des codes correcteurs d'erreur les plus performants connus à ce jour.

Nous nous sommes intéressés tout d'abord aux notions fondamentales de la théorie du l'information, ensuite, nous avons concentré notre étude sur des généralités du codage convolutionnel afin de constituer la base pour tout ce qui suit, donc dans un premier temps, un système de communication a été présenté, pour ensuite passer au codage convolutionnel. Les codeurs convolutionnels récursifs et non récursifs ont été étudiés et par la suite, la concaténation parallèle des codes convolutionnels systématiques et récursifs a été introduite.

Une fois les bases établies, il s'agissait de décrire les différentes parties qui composent les codes turbo. Nous avons décidé de commencer par un élément très important, à savoir l'algorithme de décodage. Il en existe plusieurs, mais nous avons concentré notre étude à l'algorithme **MAP**. L'étude qui a été menée fut très théorique et plusieurs notions furent introduites, comme les métriques de branche.

Nous avons décrit le décodage turbo. Tout d'abord nous avons introduit la notion de décodage itératif. Ensuite, nous avons décrit en détail le processus de décodage turbo et nous avons aussi présenté le schéma général de ce décodeur. Et Nous avons programmé un algorithme qui base sur le codage turbo, l'entrelacement et le décodage, vu la complexité de ce dernier, sa performance de correction d'erreur.

En ce qui concerne la partie programmation, nous avons fait plusieurs itérations pour voire la performance de turbo code (4 itérations et la limite théorique), ensuit on a mis tous les graphes obtenus dans une seul figure, elle nous a permis d'apprendre et de comprendre de plus en plus les possibilités données en utilisant le logiciel Matlab.

Bibliographie :

[1]. Adriana Dinu, Joëlle Barral, ‘’ Codes correcteurs d’erreurs: implémentation des turbo codes ‘’.

[2]. Berrou, Glavieux et Thitirnajshima, ‘’ Near Shannon limit error correcting coding and decoding: turbo codes ‘’, IEEE transmission and communication, proceedings of patents, Geneva, Switzerland, pp.1064-1070, May 1993.

[3]. Khaled Lajnef, ‘’ Etude des performances des codes turbo ‘’, mémoire de maitrise, école polytechnique de Montréal, Juin 2001.

[4]. Afif Hani Osseiran, ‘’ On the decoding of turbo codes ‘’, mémoire de maitrise, université de Montréal, Octobre 1999.

[5]. Proakis, J.G., Digital Communication. Third edition, Mc Graw-Hill, 1995.

[6]. Mohsen El-Bendary, Atef Abou El-Azm, ‘’ an efficient chaotic interleaver for image transmission over IEEE 802.15.4 Zigbee Network ‘’, journal of telecommunication and information technology, pp.67-73, February 2011.

[7]. Haithem Ben Chikha, ‘’ Etude et amélioration de turbo-codage distribué pour les réseaux coopératifs ‘’, thèse de doctorat, université de Valenciennes et du Hainaut-Cambrésis et l’école nationale d’ingénieurs de Tunis, Tunisie, Avril 2012.

[8]. Aroua Briki, ‘’ Une nouvelle approche de placement de données en mémoire: application à la conception d’architectures d’entrelaceurs parallèles ‘’, thèse de doctorat, université de Bretagne Sud, France, Janvier 2014.

[9]. A. Moulay Lakhdar, R. Méliani, S. Azzedine, M. Kandouci, ‘’ Etude de l’effet de l’entrelacement sur les turbo-codes ‘’, colloque national sur le traitement du signal et ses applications (CNTSA), université de Guelma, Algérie, 19- 20 Septembre 2005.

[10]. Charan Langton, ‘’ Turbo coding and MAP decoding - part 1 ‘’, intuitive guide to principles of communications, website: Complex to Real, 2007.

[11]. Charan Langton, ‘’ Turbo coding and MAP decoding - part 2 ‘’, intuitive guide to principles of communications, website: Complex to Real, 2007.

[12] Haicheng. LIU, " contribution à la maitrise de la consommation dans des turbo décodeurs", Université de Bretagne-sud ,2009.

[13] Adamovic. Melita," Codes convolutionnels perforés de faible taux et applications aux systèmes à concaténation sérielle ", Université de Montréal, Canada, May 2000.

[14] Lajnaf. Khaled, " Etude des performances de turbo code ", Université de Montréal, Canada ,2001.

[15] Grégory. Royer," Évaluation des entrelaceurs au sein des Codes Turbo par simulations ", Université de Montréal, Canada ,2000.

[16] *Mr IFTENE* Essedik « Etude des structures d'entrelaceurs pour le codage turbo du canal pour l'optimisation des systèmes de communication par satellite » Oran Mohamed Boudiaf, 2015/2016.

[17]. Chemsa Ali, Saigaa Djamel, Taleb-Ahmed Abdelmalik, '' Nouvelle approche pour optimiser un décodeur itératif '', CORESA, Lille, France, 24 Mai 2012.

[18]. Said Najah, Mostafa Mrabti, '' Implémentation d'un turbo codeur-décodeur appliqué pour le CDMA IS-2000 sur un circuit VLSI du type FPGA '', third international conference SETIT, Tunisia, March 27-31, 2005.