الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA

FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département du Génie Électrique et de l'Informatique



Projet de Fin d'Etudes

Présenté pour l'obtention du Diplôme
de Master en Génie Électrique

Spécialité : *Systèmes de Télécommunications*

THEME

Etude de la performance de turbo code

Proposé et dirigé par,

Mr. SAHNOUNE Ahmed.
Mr. BAILICHE Med Amine.

Préparé et présenté par,

ESSELIMANI Chouaib.

ALIOUCHE Houda.

Dédicaces

A ma mère et mon père pour leur appui, leur amour, leur compréhension, leur patience et pour tout ce qu'ils ont fait pour moi.

A tous mes frères et sœurs.

A toute ma famille.

A mes amis, Mohamed K, Abderaoufe L, Naser G, walid H, Khaled B.

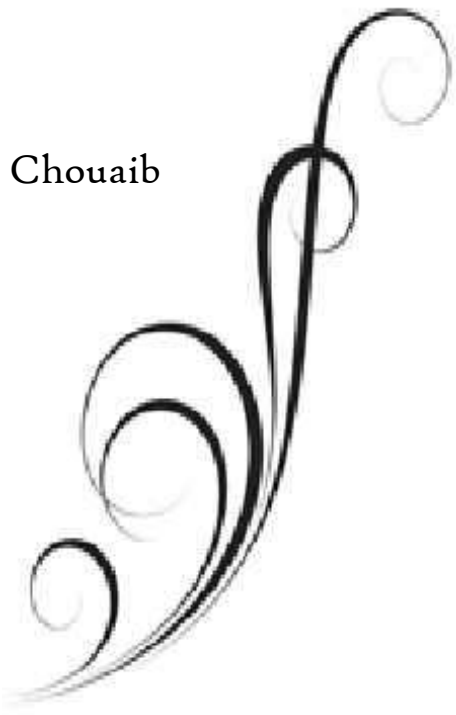
A mes collègues de l'université de Médéa.

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

A surtout mon ami Bachir.

Je dédie ce mémoire.

ESSELIMANI Chouaib



DEDICACE

Je dédie ce modeste travail

A mes très chers parents, pour leurs patience, leurs soutien et Leurs confiance ; mon père ALIOUCHE Mohamed qui s'est sacrifié afin que rien n'entrave le déroulement de mes études, ma mère ALIOUCHE Fadila 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 mes très chères sœurs Fulla et Fatima et frères Yacine et Sidali.

A toute ma famille.

A mon binôme ***Mr. ESSELIMANI Chouaib*** *et toute sa famille.*

A tous mes enseignants sans exception.

A mes collègues de l'université de Médéa.

A tous les étudiants de l'université de Médéa on particulier les étudiants de télécommunication.

A tous ceux qui me sont chères, en particulier :

MOUSAOUI Sara, ABDLAOUI Daoudia, TAHRAOUI Rachida, AOUAD Zakaria.

ALIOUCHE Houda

Remerciements

*Nous tenons tout d'abord à remercier **ALLAH** le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.*

*En second lieu, nous tenons à remercier tout le corps enseignant qui a contribué à notre formation, plus particulièrement notre encadreur **M. SAHNOUN Ahmed** et notre co-directeur **M. BAILICHE Mohamed Amine** pour ses aides et les précieux conseils qu'ils ont bien voulu nous donner.*

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.

ملخص:

في هذا العمل اقترحنا برمجة أداء تربو كود لتصحيح الأخطاء الحاصلة في انتقل المعلومة الرقمية عبر قناة AWGNحيث أن الهدف من ذلك هو برمجة خوارزمية باستعمال برنامج Matlab .

للوصول الى هذا الغرض اتبعنا ثلاث خطوات رئيسية متمثلة في ترميز المعلومة باستعمال رامزين من نفس النوع وتبديل وضعية بتات المعلومة للثاني وأخيرا فك الترميز حيث نستنتج تعقيد فك الترميز من جهة ومدة تشغيل طويلة نوعا ما من جهة أخرى.

الكلمات المفتاحية

تشويش أبيض، ترميز، تبديل الوضعية، فك الترميز، الرسم البياني.

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, codage, entrelacement, décodage, histogramme.

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, coding, interleaving, decoding, histogram.

Table des matières

LISTE DES FIGURES ..... **I**

LISTE DES TABLEAUX ..... **II**

LISTE DES ACRONYMES ..... **III**

I. Introduction générale 1

Chapitre I : Systèmes de communication numériques classiques

1. Introduction **2**

2. le système de communication numérique classique **2**

3. Canal de transmission **4**

3.1. Le canal symétrique 5

3.2. Le canal additif à bruit blanc gaussien 5

4. Codage de source **7**

5. Codage de canal **7**

6. Modulateur BPSK **7**

7. Conclusion **9**

Chapitre II : Concaténation parallèle des codes CRS

1. Introduction **10**

2. Principe de codage convolutionnel **10**

3. Réprésentation des codes convolutionnels **12**

3.1 *Diagramme d'états*……...12

3.2 *Treillis d'encodage*…….. 13

4. Codes récursifs systématiques **14**

5. Entrelacement **20**

5.1 *Entrelaceur déterministe*……..21

5.2 *Entrelaceur aléatoire*……...22

5.2.1 *Entrelaceur pseudo aléatoire*…….22

5.2.2 *Entrelaceur pseudo-aléatoire pair-impair*……..23

6. Concaténation parallèle des codes CRS **23**

7. L'encodeur Turbo……...**24**

8. Conclusion……**25**

Chapitre III : Décodage itératif des codes concaténés

1. Introduction **26**

2. principe de décodage iteratif **26**

3. Probabilité à posteriori 27
4. Maximum à posteriori.....28
5. probabilité conjointe 30
6. l'algorithme MAP.....30
6.1. Définitions…… 31
6.2. Métrique d'état en avant ……....31
6.3. Métrique d'état en arrière…….....32
6.4. Métrique de branche …… 33
7. Conclusion……34
Chapitre IV : Programmation et discussion des résultas
1. Introduction 35
2. Conception d'algorithme 35
3. Résultats de simulation 38
4. Conclution ……43
Conclusion général 44
Bibliographie …… 45

LISTE DES FIGURES

Figure I.1 *:* Le schéma synoptique d'un système de transmission....3
Figure I.2 *:* Canal symétrique binaire de probabilité d'erreur P_e....5
Figure I.3 *:* Canal additif à bruit blanc gaussien....6
Figure I.4 *:* Le modulateur BPSK....8
Figure I.5 *:* Modèle du canal à bruit blanc gaussien additif....8
Figure II.1 :Structure d'un codeur convolutionnel....10
Figure II.2 *:* Diagramme d'état du codeur de la figure II.1 (K=3. R=1/2)....13
Figure II.3 *:* Représentation en treillis du code convolutionnel de la figure II.1....14
Figure II.4 *:* Autre représentation du code R=1/2, *K=3, G_1=5,* G_2=7.....15
Figure II.5 *:* Représentation polynomiale du code R=1/2, K=3, G_1=5, G_2=7.....16
Figure II.6 *:* Structure d'un codeur récursif systématique (K=3, R= 1/2)....18
Figure II.7 *:* Structure du deuxième codeur récursif systématique (K=3, R=1/2)....19
Figure II.8 *:* Principe d'un entrelaceur déterministe....21
Figure II.9 *:* Mode d'opération pour réordonner des symboles....21
Figure II.10 *:* Entrelaceur pseudo-aléatoire pair-impair....23
Figure II.11 *:* Schéma d'une concaténation parallèle des codes CRS....24
Figure II.12 *:* Schéma d'un turbo codeur....25
Figure III.1 *:* Schéma d'un principe de décodage itératif....27
Figure IV.1 *:* Principe de codage turbo....36
Figure IV.2 *:* Principe de décodage turbo....37
Figure IV.3 *:* TEB après 1^{er} itération....38
Figure IV.4 *:* TEB après 2^{eme} itération....39
Figure IV.5 *:* TEB après 3^{eme} itération....39
Figure IV.6 *:* TEB après 4^{eme} itération....40
Figure IV.7 *:* TEB théorique....40
Figure IV.8 *:* Comparaison de TEB avec la limite théoriques....41

LISTE DES TABLEAUX

Tableau IV.1. Valeurs de TEB pour 4 itération et comparaison avec TEB théorique du code convolutionnel correspondant (Certaines valeurs sont supérieures à 1, à cause de bande théorique)…… 42

LISTE DES ACRONYMES

ABBG : Additif à Bruit Blanc Gaussien.

APP: A posteriori Brobability.

AWGN: Additive White Gaussian Noise.

BABG : Bruit Additif Blanc Gaussien.

BCJR: Bahl, Cock, Jelinec, Raviv.

BER: Bit Error Rate.

BPSK: Binary Phase Shift Keying.

CNS : Code Non Systématique.

CRS : Code Récursif Systématique.

DVB-RCS: Digital Video Broacasting-Return over Channel System.

DVB-S2: Digital Video Broadcasting-Second generation.

DVB-T2: Digital Video Broadcasting- Second Generation Terrestrial.

LTE: Long Term Evolution.

MAP: Maximum A posteriori.

NRNSC : Non Récursif Non Systématique Code.

SNR : Rapport Signal sur Bruit.

SOVA : Soft Output Viterbi Algorithm.

TEB : Taux d'Erreur Binaire.

Introduction générale

Le turbo code est le nom générique d'un code correcteur d'erreur imaginé dans les années 1990 [10], qui permet de s'approcher aussi près qu'on le souhaite de la limite de Shannon. Les turbos codes représentent une percée majeure dans le domaine des communications numériques. Ils sont utilisés dans de nombreux standard de téléphonie mobile (UMTS, LTE), de communications par satellites (DVB-S2, DVB-RCS) ou de la télévision numérique terrestre (DVB-T2). Leurs inventeurs sont Claude Berrou et Alain Glavieux, qui brevetèrent cette technologie pour le compte de France Télécom.

La clé de cette trouvaille aurait résidé dans la mise au point de l'information extrinsèque, résultat de la soustraction de l'information a priori (donnée d'entrée) à l'information à posteriori (donnée de sortie). C'est, entre autres, cette soustraction permettant d'éviter les phénomènes de propagation d'erreur dans la boucle "turbo" qui fut difficile à mettre au point.

Le principe du turbo code, comme tous les codes correcteurs 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 identiques séparés par un entrelaceur, l'un des deux utilise une séquence d'information permuté.

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 ».

Notre projet intitulé étude de la performance de turbo code, nous allons représenter cette dernière sous la forme d'une courbe ouvert qui est un rapport signal sur bruit en fonction de taux d'erreur binaire.

Ce mémoire est divisé en quatre chapitres, dans le plan de travail sera comme suit :

Le premier chapitre est consacré à la présentation des systèmes de communication numérique en général, nous avons situé quelques notions.

Le deuxième chapitre introduit la concaténation parallèle des codes récursifs systématiques.

Le troisième chapitre est dédié à l'étude de l'algorithme de décodage où nous avons choisi l'algorithme MAP.

Le chapitre IV contient les résultats de notre programme et les discutions de ces derniers.

Et finalement on termine par une conclusion générale.

Chapitre I

Systèmes de communication numériques classiques

1. Introduction :

Ce chapitre donne un aperçu général du cadre classique d'un système de communications numériques. Il combine à la fois la théorie de l'information et les communications numériques. Les notions de base concernant le traitement numérique de l'information sont abordées dans ce chapitre. Une description plus exhaustive se trouve dans l'ouvrage. [1]

Les systèmes de communications numérique véhiculent de l'information entre une source et un destinataire en utilisant un support physique comme le câble, la fibre optique ou encore, la propagation sur un canal radioélectrique. 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. [2]

2. Le système de communication numérique classique

Un système de communications numériques est principalement constitué d'un bloc d'émission, d'un milieu de transmission et d'un bloc de réception (voir la Figure I.1). Les sources initiales sont généralement représentées sous forme de signaux analogiques à spectre limité. Ces dernières sont ensuite échantillonnées avec une fréquence d'échantillonnage au moins deux fois supérieure au spectre le plus élevé du signal. Après échantillonnage, les sources se transforment en signaux discrets et numériques, qui sont composés d'une suite d'éléments pouvant prendre ses valeurs parmi les Q valeurs possibles, représentées par a_k dans la Figure I.1 dont k est l'indice de temps. L'ensemble de valeurs possibles est appelé « alphabet », noté A. Les éléments, dits Q-aires, peuvent être considérés comme des variables aléatoires définies dans un intervalle limité. Lorsque l'alphabet est égal à A ={0,1}, son élément est appelé l'élément binaire. Dans la théorie des communications numériques, un élément des Q-aires peut être modélisé par P éléments binaires par l'expression :

$$P=[\log_2 Q] \qquad \textbf{(I.1)}$$

Le schéma synoptique d'un système de transmission numérique est donné à la figure I.1 où l'on se limite aux fonctions de base :

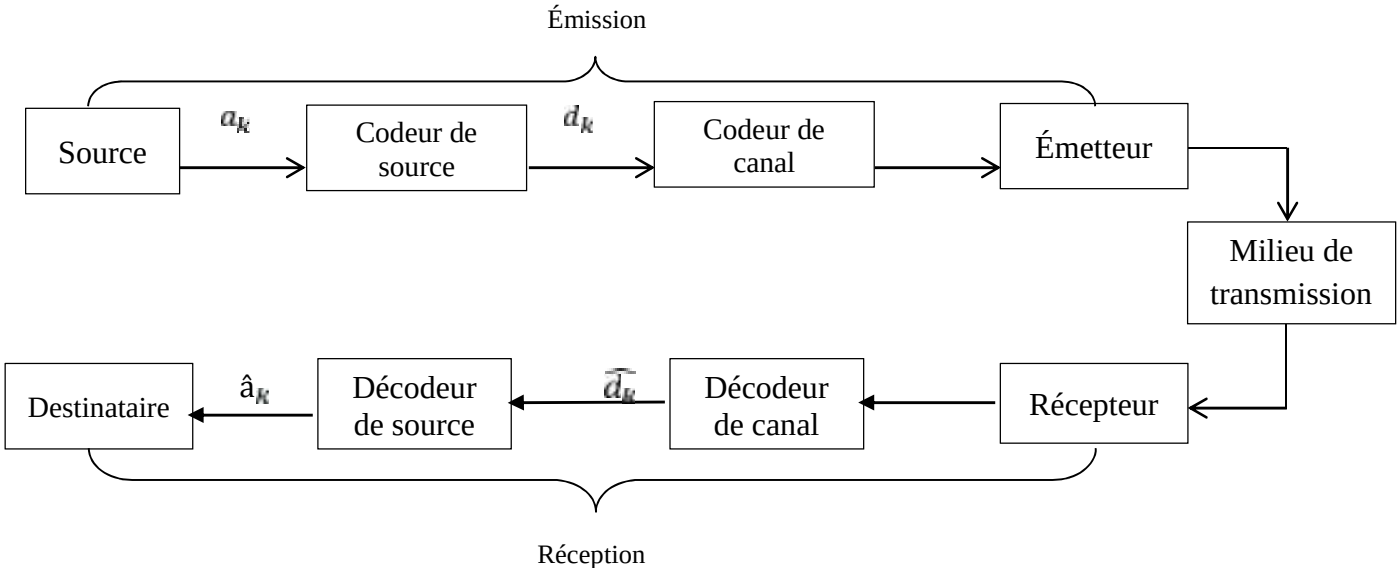


Figure I.1 Le schéma synoptique d'un système de transmission

- La source émet un message numérique sous la forme d'une suite d'éléments binaires.
- Le codeur peut éventuellement supprimer des éléments binaires non significatifs (Compression de données ou codage de source), ou au contraire introduire de la

Redondance dans l'information en vue de la protéger contre le bruit et les perturbations Présentes sur le canal de transmission (codage de canal). Le codage de canal n'est possible que si le débit de source est inférieur à la capacité du canal de transmission (La probabilité d'erreur **Pe** tend dans ce cas vers **0** d'après les travaux de Hartley Shannon).

- La modulation a pour rôle d'adapter le spectre du signal au canal (milieu physique) sur Lequel il sera émis.
- 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. [3]

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 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 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 a_k à partir du résultat d_k . Le résultat de reconstitution a_k est ensuite transmis aux destinataires en transformant le signal numérique en signal analogique. [2]

3. Canal de transmission

Dans la théorie de communications numériques, le canal de transmission correspond aux éléments entre la sortie du traitement en bande de base de l'émetteur et l'entrée du récepteur en bande de base. Notons que l'entrée du canal de transmission traite des éléments binaires,

tandis que la sortie produit une information probabiliste associée aux éléments binaires. Cette information est dite souple et le canal de transmission est dit canal à bruit continu. Si le récepteur prend une décision fermé sur les éléments binaires, alors le canal est dit canal discret et la décision est dite décision dure. 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. [2]

3.1 Le canal symétrique

Pour un canal binaire, la probabilité d'un symbole de sortie Y_k par rapport à celle d'entrée d_k est constante, dite probabilité d'erreur de transition. Le canal est sans mémoire si le symbole reçu Y_k ne dépend que de l'élément d_k. Lorsque la probabilité d'erreur est constante et indépendante du symbole d_k, le canal est appelé canal symétrique binaire, comme montré sur la Figure I.2, où **Pe** représente la probabilité d'erreur par élément binaire. La probabilité d'erreur est alors symétrique et peut être écrite de la manière suivante :

$$\begin{cases} P\ (0 \to 1) = P\ (1 \to 0) = P \\ P\ (0 \to 0) = P\ (1 \to 1) = 1 - P \end{cases} \quad \textbf{(I.2)}$$

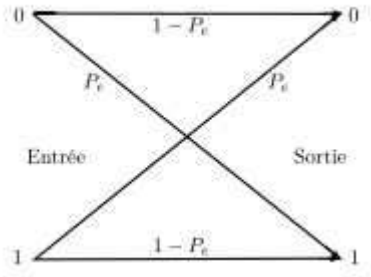


Figure I.2 Canal symétrique binaire de probabilité d'erreur P_e

3.2 Le canal additif à bruit blanc gaussien

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 Gaussian Noise) est le plus utilisé.

Pour ce canal, l'entrée d_k est discrète binaire, à valeur dans l'alphabet $\{0, 1\}$et la sortie est continue, constituée d'échantillons analogiques perturbés par un bruit b_k, additif, blanc, gaussien, stationnaire, centré et indépendant des éléments 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.3.

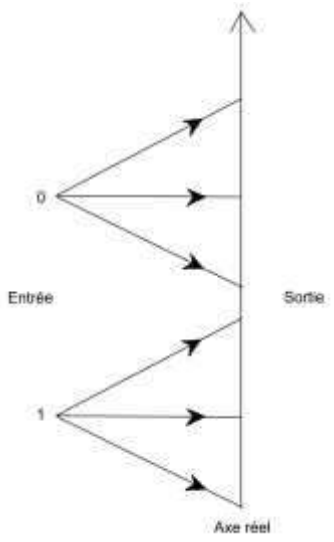


Figure I.3 Canal additif à bruit blanc gaussien

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 σ^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.3).

Considérons que le BABG est une variable aléatoire gaussienne B de moyenne nulle (centré) et de variance σ^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 :

$$P(b) = \frac{1}{\sqrt{2\ \sigma^2}} e \quad \left(\frac{b^2}{2\sigma^2}\right) \qquad \textbf{(I.3)}$$

Où $\sigma^2 = N_0/2$ dont N_0 est la densité spectrale de puissance mono latérale constante. [2]

4. Codage de source

Le codage de source vise à la concision maximale du message, afin de minimiser les ressources nécessaires à la transmission (temps, puissance, bande passante, surface de stockage, etc.). Ce codage peut donc pour diminuer le coût de la transmission, substituer un message aussi court que possible au message émis par la source, dans la mesure où cette substitution est réversible (que le message initial peut être exactement restitué). Le codage de source ne sera pas étudié. Indiquons simplement que ses limites théoriques sont fixées par la théorie de l'information (Premier théorème de Shannon). [4]

5 .Codage de canal

La probabilité d'erreur d'une transmission numérique est fonction du rapport signal à bruit E_b/N_0. Pour améliorer la qualité de transmission, il est donc possible d'améliorer ce rapport en augmentant la puissance d'émission ou diminuant le bruit du récepteur. Cependant, cela n'est pas toujours une solution convenable puisque la puissance émise est généralement limitée par l'équipement, par les standards, ou simplement par le prix matériel des composants. Une autre solution pour lutter contre les erreurs de transmission consiste à répéter la transmission, dite transmission redondante, pour obtenir une meilleure qualité. Bien que cette solution conduise à un accroissement de la complexité des équipements de transmission, elle constitue une approche possible.

Le deuxième théorème de Shannon indique qu'il est possible d'avoir une transmission statistiquement fiable avec un codage approprié. Jusqu'à présent, il existe essentiellement deux familles de codes : les codes en blocs et les codes convolutifs. Les codes en blocs n'étaient pas abordés dans notre étude, nous nous contentons d'une présentation des codes convolutifs. [2]

6. Modulateur 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.4, la sortie générée par celui-ci est :

$$\boldsymbol{Y_K = 2\, d_K - 1} \qquad \textbf{(I.4)}$$

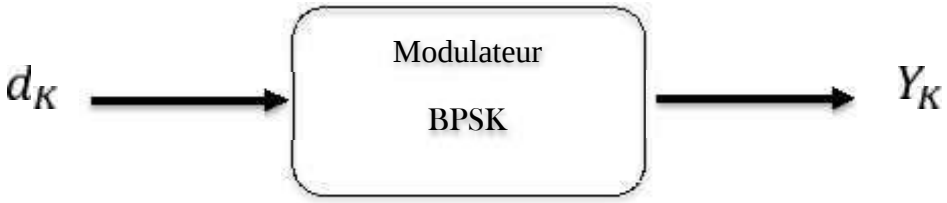


Figure I.4 : Le modulateur BPSK

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.5). [6]

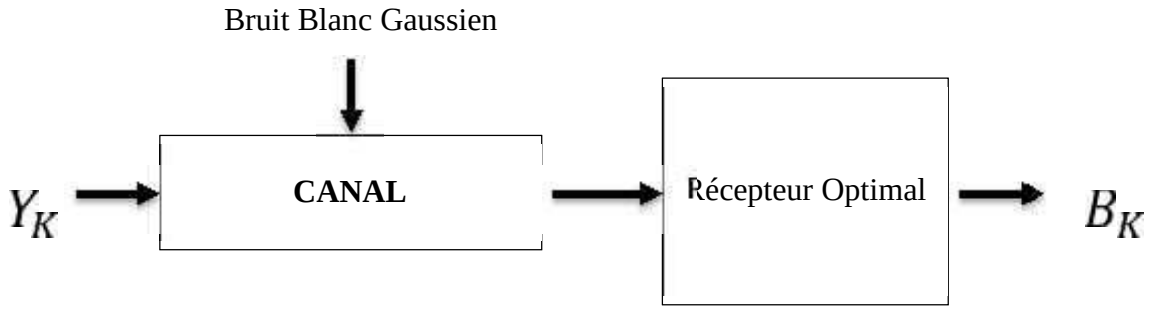


Figure I.5 : Modèle du canal à bruit blanc gaussien additif

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 :

$$\boldsymbol{B_k = Y_k + P_K = 2d_K - 1 + P_K} \qquad \textbf{(I.5)}$$

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 :

$$P(x) = \frac{1}{\sigma\sqrt{2}} e^{\left(-\frac{x^2}{2\sigma^2}\right)} d \quad \text{(I.6)}$$

Où σ^2 est la variance du bruit blanc gaussien, et vaut :

$$\sigma^2 = \frac{1}{2R_t} \frac{E_b}{N_0} \quad \text{(I.7)}$$

R_{total} étant le taux de codage du système entier, comprenant les deux codeurs de taux R_1 et R_2 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. [6]

7. 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

1. Introduction

Les systèmes de communication numériques modernes exigent des performances de plus en plus fiables. Une des façons d'augmenter la fiabilité de la communication est de transmettre les bits de redondance associés au signal d'information numérique émis. De sorte qu'au récepteur certaines erreurs provenant du canal puissent être corrigées. On parle donc de procédure de codage correcteur d'erreurs dont les codes convolutionnels représentent une forme très puissante. Ce chapitre décrit les notions de base du codage convolutionnels ainsi que la concaténation parallèle des codes CRS (codes récursifs systématiques).

2. Principe de codage convolutionnel

Un codeur convolutionnel est constitué d'un registre à décalage formé de K cellules, de V additionneurs modulo-2, d'un ensemble de connexions entre les additionneurs et les cellules du registre à décalage et du commutateur à V positions. Chaque codeur convolutionnel est caractérisé par les paramètres suivants :

- Taux de codage R =b/V

Où b représente le nombre de bits simultanés à l'entrée du codeur et V le nombre de symboles codés à la sortie du codeur

- Longueur de contrainte du codeur K

Où K représente le nombre de cellules du registre à décalage.

Afin de mieux comprendre le fonctionnement du codeur convolutionnel ainsi que L'influence de ces paramètres sur la performance du système, prenons comme exemple un codeur convolutionnel simple représenté par la figure II.1. [6]

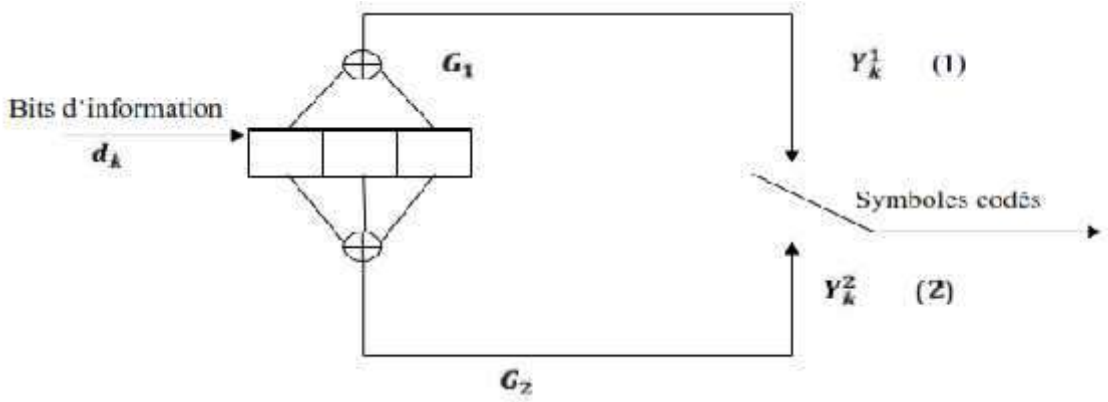


Figure II.1 : Structure d'un codeur convolutionnel

Comme on peut le constater d'après la figure II.1, le nombre d'additionneurs dans cet exemple est V=2**,** le taux de codage devient R= 1/2 car chaque bit à l'entrée du codeur produit deux symboles codés à sa sortie et la longueur de contrainte (nombre de cellules du registre à décalage) est K=3.

L'ensemble des connexions entre le registre à décalage et les additionneurs modulo-2 est représenté par des vecteurs générateurs. Ce sont les vecteurs générateurs du codeur qui déterminent les règles selon lesquelles les bits de redondance doivent être ajoutés aux bits d'information à transmettre. Les vecteurs générateurs G_i peuvent être exprimés par :

$$G_i = \{g_{i1}, g_{i2}, g_{i3}, \dots\dots\dots, g_{ij}\} \quad i = 1, \dots\dots\dots, V \qquad \textbf{(II.1)}$$

La composante g_{ij} vaut 1 si la connexion entre la j-ème cellule du registre et l'i-ème additionneur modulo-2 existe et vaut 0 sinon.

Pour l'exemple de la figure II.1, les générateurs valent : $G_1 = (101)_2$, $G_2 = (111)_2$. Les générateurs sont très souvent exprimés sous forme octale, ce qui nous amène (dans L'exemple considéré) à la notation suivante : G_1=5**,** G_2=7**.** C'est cette dernière notation qu'on utilisera tout au long de ce travail.

Le codeur fonctionne de la manière suivante : avant que le processus d'encodage ne débute, le contenu du registre à décalage est initialisé à zéro. Les bits d'information arrivent à l'entrée du codeur de façon continue. Le bit d'information à l'entrée du codeur est injecté dans le registre à décalage et en utilisant les vecteurs générateurs du codeur, les V symboles codés correspondants sont calculés. La séquence des symboles codés est obtenue en échantillonnant les additionneurs rnodulo-2 à l'aide de commutateur. Une fois tous les bits d'information codés. Une séquence de K -1 zéros est ajoutée (et codée) afin de remettre le registre à l'état initial, c'est-à-dire à l'état zéro. [6]

Evidemment. Le symbole codé ne dépend pas seulement du bit à l'entrée du codeur mais aussi du contenu des K-1 cellules du registre précédant le bit courant. La mémoire M du codeur du taux de codage R=1/ V est définie par :

$$\textbf{M= K-1} \qquad \textbf{(II.2)}$$

Dans ce travail. Où on considère des codes du taux de codage R=1/ V, les bits d'information sont toujours considérés binaires. Alors, selon la valeur du bit d'information. Le contenu des K-1 cellules du registre peut prendre une des 2^{k-1} valeurs possibles et donc :

Nombres d'états d'un codeur = $2^{k-1}=2^{M}$ **(II.3)**

Soit d_k le bit d'information à l'entrée du codeur à l'instant k. Les symboles codés Y_k^i

(i= l, ..., V) peuvent alors s'écrire sous la forme :

$Y_k^i = \sum_{j=0}^{k-1} g_i\, d_{k-j}$ (m) **i=1, 2…, V** **(II.4)**

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

Cette séquence peut s'exprimer sous la forme du produit de convolution de la séquence d'entrée et de la réponse impulsionnelle du codeur (réponse à l'entrée 1000...), d'où le nom des codes convolutionnels. [6]

3. Représentation des codes convolutionnels

Dons cette section. On introduit les représentations en diagramme d'états et en treillis d'encodage d'un code convolutionnel.

3.1 *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.2 représente le diagramme d'état du codeur de la figure II.1.

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.2 représente les symboles codés correspondant au bit d'information.

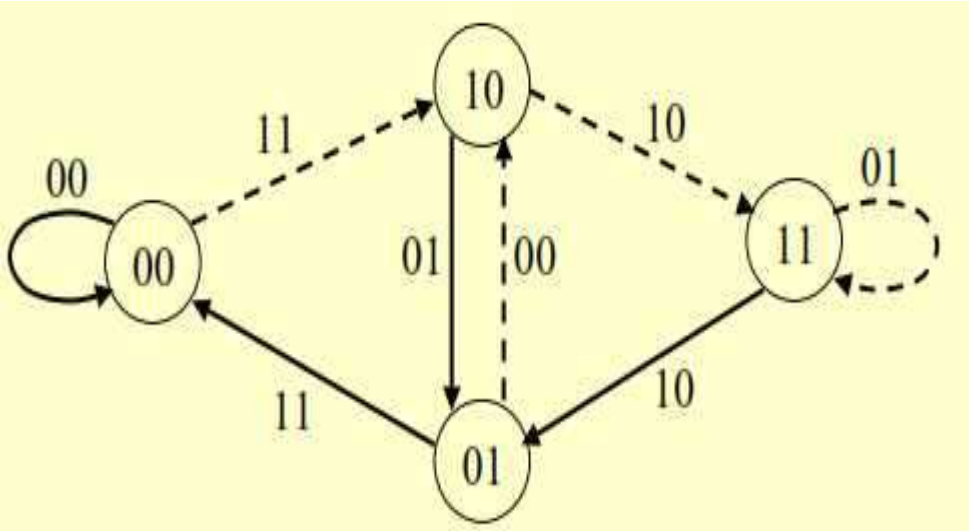


Figure II.2 : Diagramme d'état du codeur de la figure II.1 (K=3. R=1/2)

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 $\underline{0}$ = **(**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. [6]

3.2 *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.

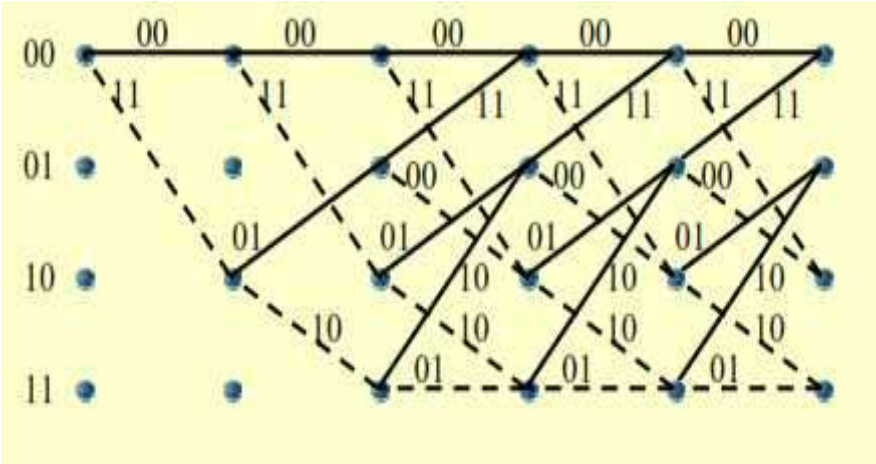


Figure II.3 : Représentation en treillis du code convolutionnel de la figure II.1

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 la prochaine 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^b branches arrivant à chaque nœud et 2^b branches sortant de chaque nœud du treillis. [6]

4. Codes récursifs systématiques

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.4.

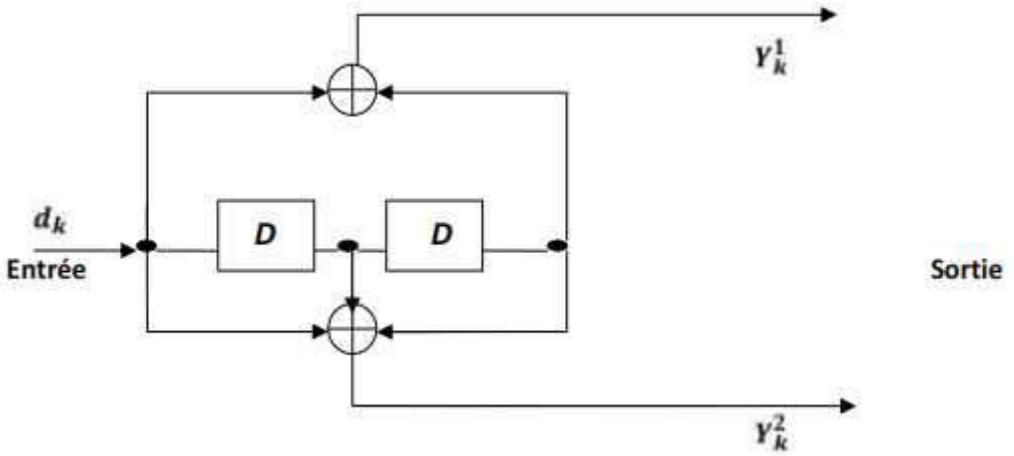


Figure II.4 : Autre représentation du code R=1/2, *K=3, G_1=5,* G_2=7.

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.4 peut être utilisé comme indicateur de temps.

Dans ce dernier cas, l'équation (II.4) devient :

$$Y_k^i(D) = G_i(D)d_k(D) \qquad \textbf{i=1,… V} \qquad \textbf{(II.5)}$$

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

$$G_i(D) = \sum_{j=0}^{k-1} g_{ij} D^j \qquad \textbf{i=1… V} \qquad \textbf{(II.6)}$$

Le codeur de la figure II.4 est repris à la figure II.5 :

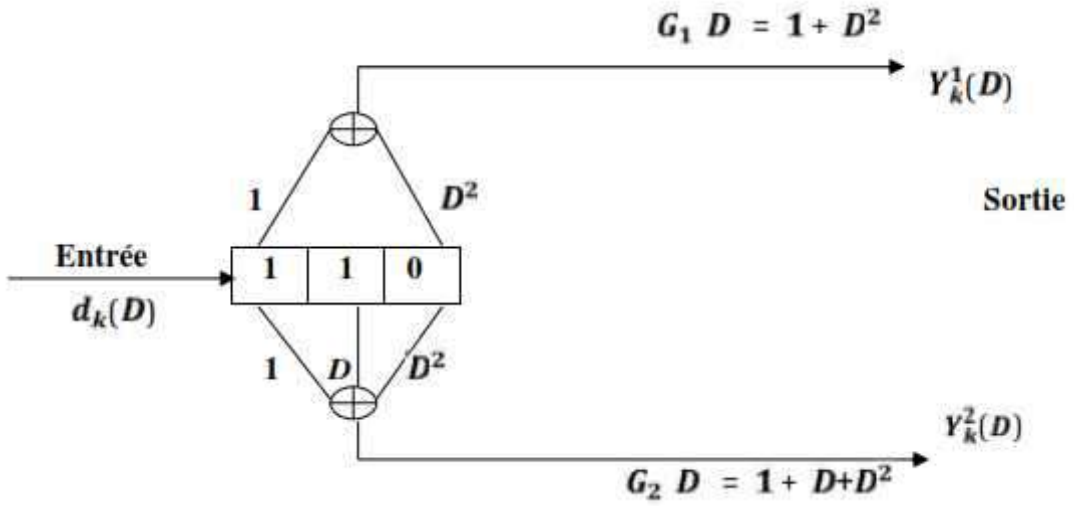


Figure II.5 : Représentation polynomiale du code R=1/2, K=3, G_1=5, G_2=7.

Pour obtenir la version systématique du code de la figure II.5, l'une des deux sorties $Y_k^i(D)$, i=l, 2. Doit être égale à d_k **(***D***).** Pour ce faire**,** on peut diviser les termes de l'équation (II.5) soit par G_1**(***D*) Soit par G_2**(***D*). On choisit G_1**(***D*). Dans cet exemple, On obtient alors :

$$\overline{Y}_K^1(D) = d_k(D) = \overline{G_1}\ (D)\ d_k(D) \qquad \textbf{(II.7a)}$$

$$\overline{Y}_K^2(D) = \frac{G_2(D)}{G_1(D)} d_k(D) = \overline{G_2}(D)\ d_k(D) \qquad \textbf{(II.7b)}$$

Avec

$$\overline{Y}_K^i(D) = \frac{Y_k^i(D)}{G_1(D)} \qquad \textbf{i=1, 2} \qquad \textbf{(II.8)}$$

Les polynômes générateurs du code ainsi obtenu sont alors :

$$\overline{G_1}\ (D)=1 \qquad \textbf{(II.9a)}$$

$$\overline{G_2}(D)= \frac{G_2(D)}{G_1(D)} \qquad \textbf{(II.9b)}$$

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 :

$$A_k(D) = \frac{d_k(D)}{G_1(D)} \qquad \textbf{(II.10)}$$

Les équations (II.7a) et (II.7b) deviennent :

$$\overline{Y}_K^1(D) = d_k(D) \quad \textbf{(II.11a)}$$

$$\overline{Y}_K^2(D) = G_2(D)A_k(D) \quad \textbf{(II.11b)}$$

Le code ainsi construit. Dont les sorties sont définies par les équations (II.11a) et (II.11b) est systématique et récursif**.** Le caractère récursif apparait plus clairement en ramenant l'équation (II.10) dans l'espace temporel. Elle devient alors :

$$d_k = \sum_{j=0}^{k-1} g_1 \; a_{k-j} \quad \textbf{(II.12)}$$

En tenant compte du fait que toutes les opérations sont faites modulo 2, et en faisant l'hypothèse que g_1 = 1**,** le symbole a_k peut s'exprimer récursivement en fonction des symboles a_{k-j} (j=1,2, ….k-1) et du symbole d_k .

$$a_k = d_k + \sum_{j=0}^{k-1} g_1 \; a_{k-j} \quad \textbf{(II.13)}$$

Par conséquent. Dans le cas d'un code CRS**,** ce sont désormais les symboles a_k qui sont contenus dans le registre à décalage du codeur. A partir des relations (II.11a) et (II.11b) (II.12), les sorties Y_k^1 et Y_k^2 du code CRS à un instant k peuvent s'écrire sous la forme :

$$Y_k^1 = d_k = \sum_{j=0}^{k-1} g_1 \; a_{k-j} \quad \textbf{(II.14a)}$$

$$Y_k^2 = \sum_{j=0}^{k-1} g_2 \; a_{k-j} \quad \textbf{(II.14b)}$$

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 d_k **.**

Un codeur CRS équivalent au codeur de la figure II.1 est représenté à la figure II.6. Comme la figure II.6 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.

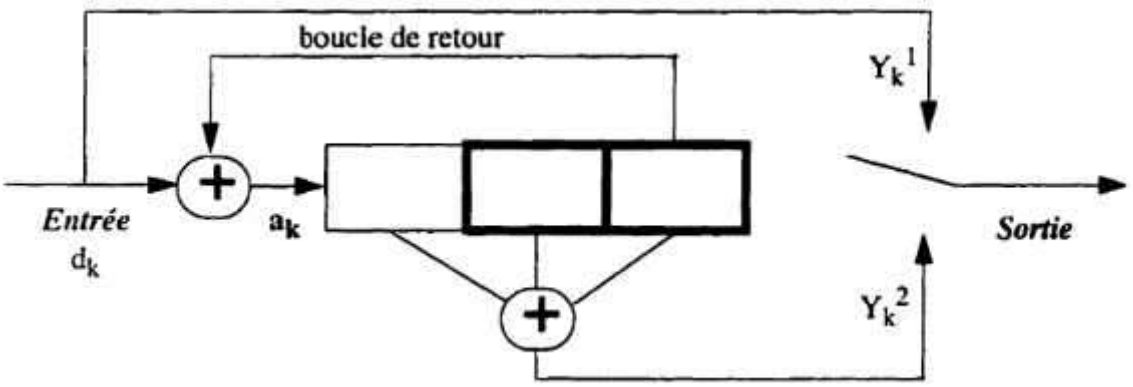


Figure II.6 : structure d'un codeur récursif systématique (K=3, R= 1/2)

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.1 et II.6 respectivement), on remarque que le vecteur générateur du code diffère d'un cas à l'autre. Le vecteur générateur correspondant au code convolutionnel non-systématique représenté à la figure II.3 peut être exprimé comme suit :

$$\underline{G}\ c = (G_1, G_2) = (5, 7) \qquad \textbf{(II.15)}$$

Tandis que le vecteur générateur du codeur récursif systématique illustré à la figure II.6 vaut :

$$\underline{G}\ \textbf{récursif systématique 1 = (1, 7/5)} \qquad \textbf{(II.16)}$$

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.6) 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.6) obtenu à partir du même codeur de la figure II.1 est représenté à la figure II.7. Dans ce cas, le vecteur générateur du code devient :

$$\underline{G}\text{ récursif systématique 2} = (1,5/7) \qquad \textbf{(II.17)}$$

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

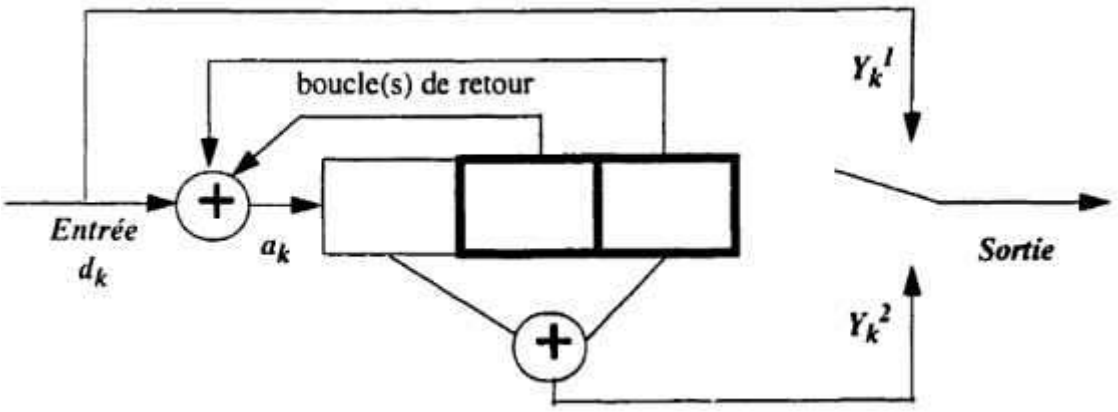


Figure II.7 : Structure du deuxième codeur récursif systématique (K=3, R=1/2)

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. [6]

5. Entrelacement

Un entrelaceur π permet de modifier les indices (positions des bits en paquet d'information) des données par permutations, il représente un élément du groupe de permutation pour N éléments. Des groupes de permutation peuvent être décomposés en utilisant une seule méthode. Par exemple, π_1 = {15,10,1,12,2,0,13,9,5,3,8,11,7,4,14,6} peut-être écrit dans la décomposition de cycle de disjonction avec deux cycles :π_1 = {(0,15,6,13,4,2,1,10,8,5), (3,12,7,9)}

Ce qui signifie simplement que les symboles sont pris par ordre comme suit : 0 → 15 → 6 →13 → 4 → 2 → 1 → 10 → 8 → 5 → 0 →15 dans un cycle de longueur 8, et un deuxième cycle de longueur 4. Alternativement, l'entrelaceur peut être exprimé par sa fonction d'indice. [7]

$$\pi(i) = j, \quad 0 \leq i, j < N. \qquad \textbf{(II.18)}$$

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.

5.1 *Entrelaceur déterministe*

Dans ce type d'entrelaceur, il s'agit d'écrire les symboles d'un bloc à entrelacer ligne par ligne dans une matrice, et de les lire en sortie colonne par colonne comme l'indique la figure II.8 Il y a différentes façons pour lire des symboles dans une matrice, de gauche vers la droite (GD) ou de droite vers la gauche (DG) pour les colonnes et de haut vers le bas (HB) ou de bas vers le haut (BH) pour les lignes. Ainsi, il y a quatre façons qui sont illustrées à la figure II.9. [5]

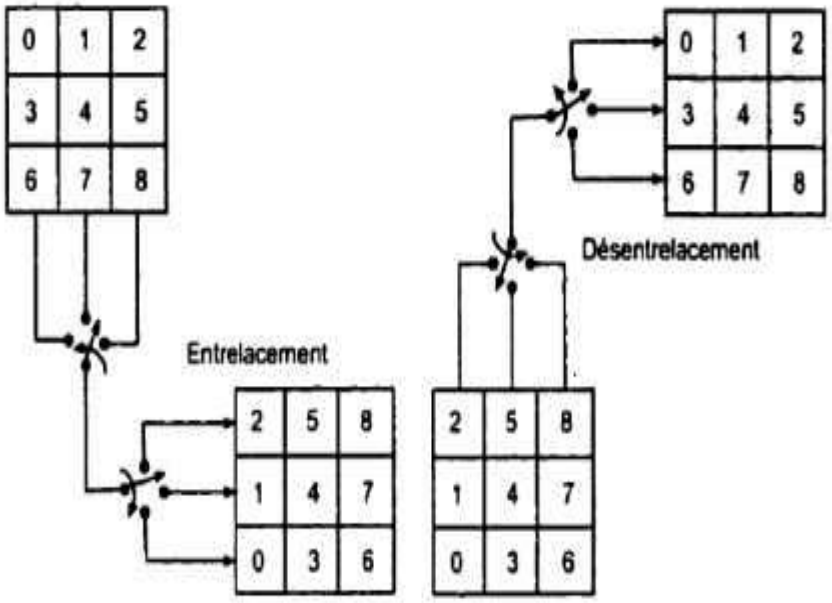


Figure II.8 : Principe d'un entrelaceur déterministe

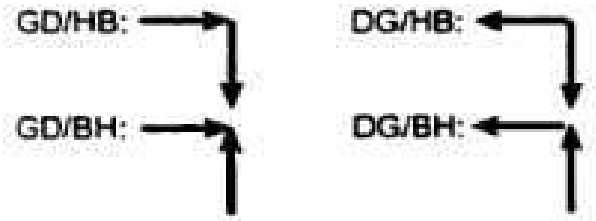


Figure II.9 : Mode d'opération pour réordonner des symboles

L'entrelacement est accompli par la lecture de l'information par la colonne (sortie de l'entrelaceur). Cet entrelaceur a la fonction d'indice :

$$\pi(i) = n + \left[\frac{i}{m}\right] \; Mod\ (N), \qquad 0 \leq i, j < N. \tag{II.19}$$

Où [x] est le plus grand nombre entier < x. Cette fonction ($\left[\frac{i}{m}\right]$) rend l'entrelaceur difficile à analyser, et un entrelaceur «linéarisé" avec la fonction d'indice.

$$\pi(i) = Ki+u \quad mod(N) \qquad 0 \leq i, j < N. \qquad \textbf{(II.20)}$$

Avec k et u sont des nombres entiers fixes et k est relativement primaire à *N*, c'est-à-dire le plus grand diviseur commun de k et N [gcd (k, N)=1]. K est appelé le coefficient angulaire de l'entrelaceur linéaire. Les entrelaceurs en bloc peuvent réaliser de bonnes valeurs de distance libre, mais ils échouent à produire un spectre des distances très étroit. Les coefficients angulaires d'approximativement $\sqrt{N}$ semblent fonctionner mieux, ayant pour résultat la bonne dispersion des points indexés. On dispose aussi pour améliorer la performance du code turbo dans l'intervalle entre les valeurs moyennes et grandes du rapport signal sur bruit des entrelaceurs semi aléatoires. [7]

Ce qui évite explicitement d'étaler des indices proches à l'entrée aux indices proches à la sortie. Ils sont produits de façon analogue aux entrelaceurs aléatoires, choisissant chaque incrément $\pi(i)$ aléatoirement et testant si $|\pi(i) - \pi(l)| \geq T$ pour tout $i\text{-}S < l < i$, c'est-à-dire en contrôlant tous les choix précédents. Si un choix d'indice ne remplit pas cette condition, il est rejeté et un nouvel indice est choisi. Le processus continue jusqu'à ce que tous les N indices aient été choisis. Évidemment, le temps de recherche pour un entrelaceur approprié augmente avec S et T, et l'algorithme peut même ne pas se terminer avec succès. Cependant, des valeurs de $S,T < \sqrt{N/2}$ finissent l'indexage dans un temps raisonnable.

Pour les entrelaceurs déterministes qui ont les propriétés statistiques des entrelaceurs aléatoires, Takeshita et Costello ont conçu ce qu'ils ont appelé les entrelaceurs quadratiques, qui sont définis par la fonction quadratique d'indice suivante :

$$\pi(i) = \frac{k\,(i+1)}{2}\ Mod\ (N). \qquad \textbf{(II.21)}$$

Avec k est une constante impaire. [7]

5.2 *Entrelaceur aléatoire*

5.2.1 *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. [5]

5.2.2 *Entrelaceur pseudo-aléatoire pair-impair*

Les entrelaceus 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.10 montre un exemple d'entrelaceur pseudo-aléatoire pair-impair. [5]

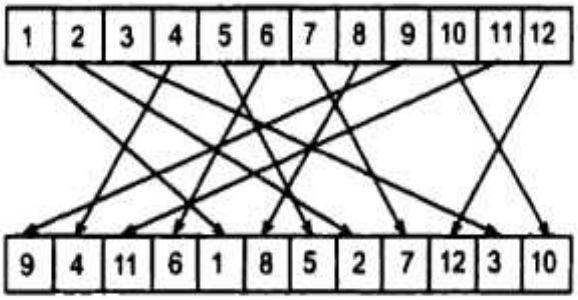


Figure II.10 : Entrelaceur pseudo-aléatoire pair-impair

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 RSC 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 RSC se justifie par le fait que pour des petits rapports signal sur bruit les codeurs RSC 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 RSC.

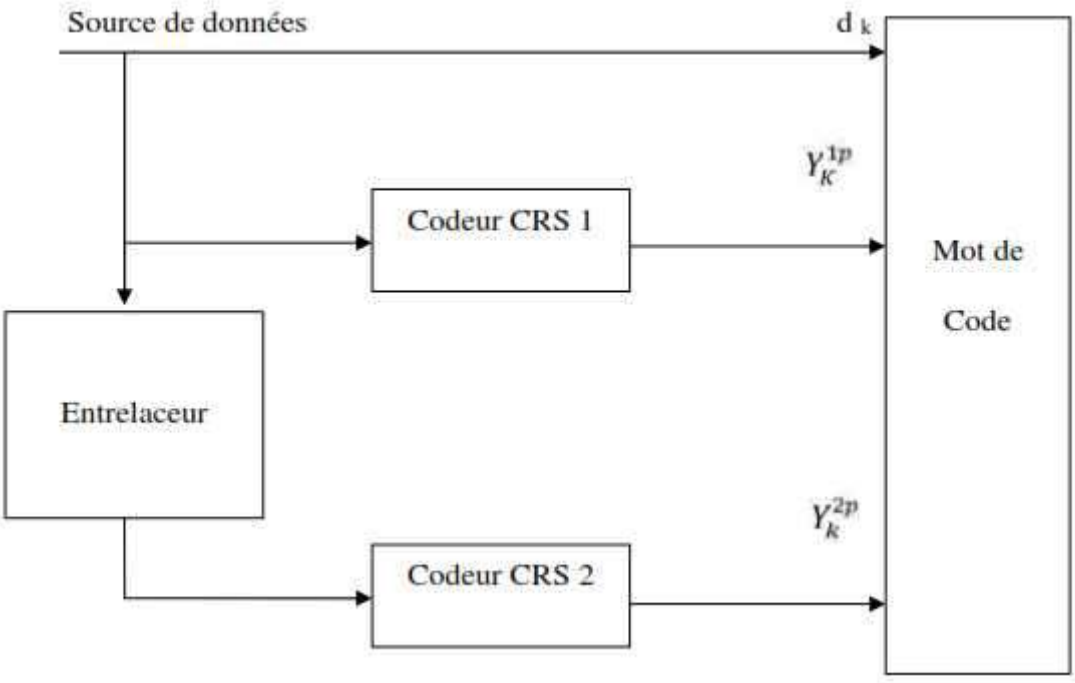


Figure II.11 : Schéma d'une concaténation parallèle des codes CRS

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_g , 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 :

$$\frac{1}{R_g} = \frac{1}{R_1} + \frac{1}{R_2} - 1 \quad \textbf{(II.22)}$$

Soit :

$$R_g = \frac{1}{\frac{1}{R_1}+\frac{1}{R_2}-1} \quad \textbf{(II.23)}$$

$$R_g = \frac{R_1 R_2}{R_1+R_2-R_1 R_2} \quad \textbf{(II.24)}$$

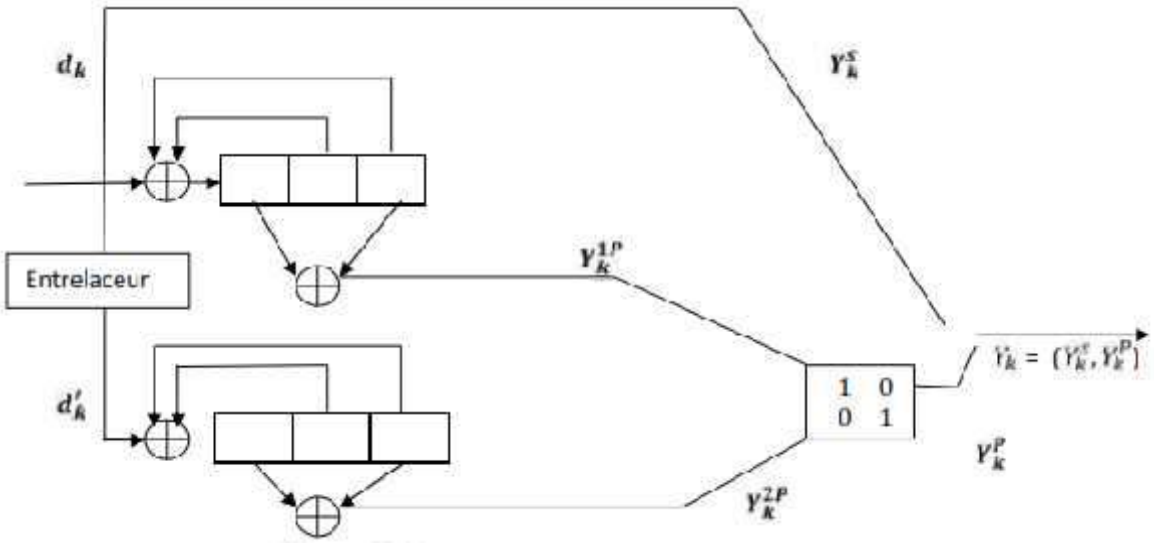


Figure II.12 : Schéma de principe d'un encodeur Turbo

Pour mieux illustrer l'encodeur turbo, nous avons représenté sur la Figure II.12 un exemple d'une concaténation parallèle de deux codes élémentaires dont le taux de codage global est égal à R_g =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. [8]

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

1. Introduction :

En 1974 [10], l'algorithme MAP (ou BCJR) a été proposé par Bahl et al. Dans le but d'estimer l'apostériori, dans le calcul des probabilités des états de transitions observé dans un canal bruité et sans effet de mémoire.
Bahl et al, ont démontré la possibilité de l'application de l'algorithme au décodage des procédés de codage par Blocs, ou par codage convolutionnel.

L'algorithme est optimal en termes de minimisation du BER, de l'information à décoder, contrairement à l'algorithme de Viterbi qui minimise l'erreur de probabilité sur un trajet sélectionné par le décodeur dans le treillis. Néanmoins, pour la plupart des applications, la performance des deux algorithmes reste presque identique.

L'algorithme MAP examine tous les chemins possibles à travers un treillis d'un codeur convolutionnel. Du coup, sa mise en œuvre semblait être compromise pour la plupart des systèmes, vu sa complexité de calcul, jusqu'à la découverte de turbo code, ce qui a permis une large utilisation de cet algorithme. Le MAP ne produit pas seulement une estimation de la séquence de bits transmis, mais aussi une probabilité de véracité pour chaque bit décodé, ceci est primordial pour un décodage itératif, comme celui proposé par Berrou et al. [9]

2. Principe de décodage itératif

Pour une première itération et après avoir décodé les symboles associés au premier codeur, le décodeur 1 fait passer un bloc d'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 détectées par le décodeur 1 et l'inverse. [5]

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 améliorer la performance d'erreur.
La séquence d'information supplémentaire échangée qui se présente à 1'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 conditionnellement sur les bits d'information. 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 délaceur entre les deux décodeurs. Les deux mis en série. [5]

Généralement, les deux décodeurs utilisent un algorithme de décodage à décision pondérée, c'est-à-dire minimisant la probabilité d'erreur par symbole. Un tel décodeur est appelé décodeur APP. Il fournit une probabilité à posteriori (APP : A Posteriori Probability) sur chaque symbole d'information émis conditionnellement à la séquence reçue (les observables). [5]

La figure III.1 montre un diagramme simplifié d'un décodage itératif qui est constitué de deux décodeurs, d'un entrelaceur et d'un désentrelaceur.

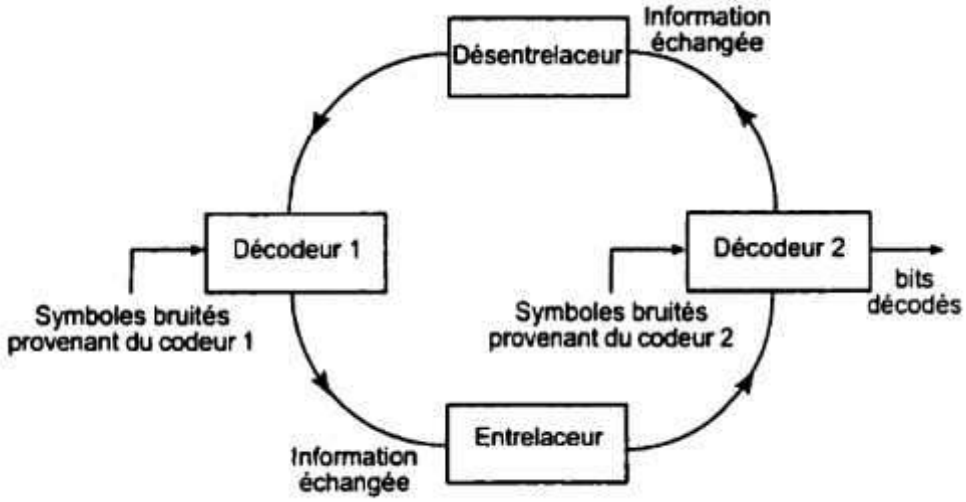


Figure III.1 : Schéma de principe du décodage itératif

3. Probabilité a posteriori :

L'algorithme de Viterbi est un algorithme très utilisé dans les systèmes de communications. Le but de celui-ci est de minimiser la probabilité d'erreur d'un mot ou d'une séquence de symboles. Dans ce cas, la règle de décision est le principe de maximum de vraisemblance. À la différence de l'algorithme de Viterbi, L'algorithme MAP agit sur la probabilité d'erreur par posteriori. [8]

Nous allons maintenant expliquer le terme de maximum a posteriori. Dans notre cas, celui de l'algorithme MAP, nous parlerons de décision pondérée, En effet, les symboles sont décodés à l'aide d'une information supplémentaire. Pour définir la probabilité a posteriori, soient les événements A_i, i = 1,2, ..., M. Ceci constitue l'ensemble des messages transmis dans un intervalle de temps donné. Maintenant, considérons B, le signal reçu que nous désirons décoder. Ce message a été corrompu par du bruit. Il s'agit d'une variable aléatoire dont nous noterons la densité de probabilité p(B). La probabilité a posteriori est définie par la

probabilité que nous reconnaissions le message A_i étant donné que nous avons reçu le message B. Utilisant la règle de Bayes, on peut écrire :

$$\mathbf{P}\ (A_i|B)= \frac{P(B|A_i)P(A_i)}{P(B)} \qquad \textbf{(III.1)}$$

Essayons maintenant de définir cette probabilité plus adéquatement. En notons R la séquence reçue et d_k le symbole reçu, cette équation devient :

$$\mathbf{P}(d_k = i|R)=\frac{P(R|d_k = i)P(d_k=i)}{P(R)} \qquad \textbf{i=0, 1} \qquad \textbf{(III.2)}$$

4. Maximum a posteriori

L'équation (III.2) développée au paragraphe précédent exprime la probabilité a posteriori.

Nous avons déjà mentionné que le principe de l'algorithme MAP consistait 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 :

$$\mathbf{P}(d_k = 1|R)\underset{>H_1}{\overset{<H_0}{}}\mathbf{P}(d_k = 0|R) \qquad \textbf{(III.3)}$$

Nous les voyons, deux hypothèses sont à considérer, à savoir l'hypothèse H_0 qui implique que le bit "0" a été transmis et l'hypothèse H_1 qui sous-entend le contraire. Insérons maintenant l'équation (III.2) dans l'équation (III.3). Il vient que :

$$\frac{P(R|d_k = 1)P(d_k=1)}{P(R)}\underset{>H_1}{\overset{<H_0}{}}\frac{P(R|d_k = 0)P(d_k=0)}{P(R)} \qquad \textbf{(III.4)}$$

Nous pouvons alors simplifier cette double inéquation :

$$\mathbf{P}(R|d_k = 1)P(d_k = 1)\underset{>H_1}{\overset{<H_0}{}}\mathbf{P}(R|d_k = 0)P(d_k = 0) \qquad \textbf{(III.5)}$$

Que nous pouvons mettre aussi sous la forme plus conventionnelle :

$$\frac{P(R|d_k = 1)P(d_k=1)}{P(R|d_k = 0)P(d_k=0)}\underset{>H_1}{\overset{<H_0}{}}1 \qquad \textbf{(III.6)}$$

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

$$\mathbf{P}(d_k = 0) = P(d_k = 1) \quad \textbf{(III.7)}$$

Ce qui nous simplifie de façon intéressante l'équation (III.6) :

$$\frac{P(R|d_k = 1)}{P(R|d_k = 0)} \underset{>H_1}{\overset{<H_0}{1}} \quad \textbf{(III.8)}$$

Or, grâce à la règle de Bayes, on peut écrire :

$$\begin{cases} p(R|d_k = 1) = \frac{P(d_k = 1|R)P(R)}{P(d_k-1)} \\ p(R|d_k = 0) = \frac{P(d_k = 0|R)P(R)}{P(d_k=0)} \end{cases} \quad \textbf{(III.9)}$$

Sachant que les bits "0" et "1"sont équiprobables, en divisant ces deux termes, (III.8) devient :

$$\frac{P(d_k = 1|R)}{P(d_k = 0|R)} \underset{>H_1}{\overset{<H_0}{1}} \quad \textbf{(III.10)}$$

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 avons-nous la règle de décision suivante :

$$\mathbf{LLR}(d_k) = \ln\left[\frac{P(d_k = 1|R)}{P(d_k = 0|R)}\right] \underset{>H_1}{\overset{<H_0}{0}} \quad \textbf{(III.11)}$$

Ceci veut donc dire que :

$$\begin{cases} \text{si } L(d_k) \geq 0, \text{ la décision est "1"} \\ \text{si } L(d_k) < 0, \text{ la décision est "0"} \end{cases} \quad \textbf{(III.12)}$$

Ce rapport de vraisemblance est la base de l'algorithme MAP que nous allons développer. Cette règle de décision va être utilisée par la suite pour définir les étapes de cet algorithme. [8]

5. 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 1/2. Soit :

- d_k = i, i = 0,1, le bit d'information présent à l'entrée du codeur à l'instant *k* ;
- M = K - 1, la mémoire du codeur ;
- $S_k = m, m = 0,1, \dots \dots \dots \dots, 2^M$ l'état du codeur ;
- N, la longueur de la séquence à coder.
- $(d_1, \square, d_k, \square, d_N) = (Y_1^S, \square, Y_k^S, \square, Y_N^S)$, la séquence à coder (aussi la séquence d'information) ;
- $(Y_1^P, \square, Y_k^P, \square, Y_N^P)$, la séquence composée des symboles de parité ;
- $R_k = (r_k^S, r_k^P)$, la version reçue de (Y_k^S, Y_k^P) à l'instant k ;
- $R_1^N = (R_1, \square, R_k, \square, R_N)$ la séquence bruitée reçue de longueur N.

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

$$\mathbf{LLR}\,(d_k) = \ln\left[\frac{P(d_k = 1 | R_1^N)}{P(d_k = 0 | R_1^N)}\right] \qquad \textbf{(III.13)}$$

Nous allons maintenant introduire une nouvelle notion. La probabilité conjointe est définie par :

$$\lambda_k^i(m) = P(d_k = i, S_k = m | R_1^N) \qquad \textbf{(III.14)}$$

Étant donné que nous avons 2^M états :

$$\mathbf{LLR}\,(d_k) = \ln\left[\frac{\sum_{m=0}^{2^M-1} \lambda_k^1(m)}{\sum_{m=0}^{2^M-1} \lambda_k^0(m)}\right] \qquad \textbf{(III.15)}$$

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. [8]

6. L'algorithme MAP

Dans cette partie, nous allons introduire de 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. [8]

6.1. *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*** : $\alpha_k(m) = P\left(R_1^{k-1} \middle| d_k = i, S_k = m, R_k^N\right)$ **(III.16)**

$$\alpha_k(m) = P\left(R_1^{k-1} \middle| S_k = m\right) \quad \textbf{(III.17)}$$

Car les événements après l'instant k n'influencent en rien les événements avant k ;

- ***La métrique d'état en arrière*** : $\beta_k(m) = P\left(R_k^N \middle| S_k = m\right)$ **(III.18)**
- ***La métrique d'état de branche*** : $\gamma_k^i(m) = P\left(d_k = i, S_k = m, R_i^k\right)$ **(III.19)**

Ayant défini ces trois notions, il s'agit maintenant de s'intéresser à la probabilité conjointe et de l'exprimer en fonction des métriques.

6.2. *Métrique d'état en avant*

Nous avons défini la métrique d'état en avant comme étant :

$$\alpha_k(m) = P\left(R_1^{k-1} \middle| S_k = m\right) \quad \textbf{(III.20)}$$

Comme nous pouvons le constater, cette métrique, à l'instant k, dépend de la séquence reçue R_1^k.Cette dernière va jusqu'à l'instant k. Utilisons la règle de Bayes pour simplifier l'expression de cette métrique :

$$\alpha_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} P\left(d_{k-1} = j, S_{k-1} = i, R_1^{k-1} \middle| S_k = m\right) \quad \textbf{(III.21)}$$

$$\alpha_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} P\left(R_1^{k-2} \middle| S_k = m, d_{k-1} = j, S_{k-1} = i, R_{k-1}\right) \times P(d_{k-1} = j, S_{k-1} = i, R_{k-1} | S_k = m) \quad \textbf{(III.22)}$$

Nous savons, de plus, que l'état S_{k-1} a été initialisé par le bit d_{k-1}. Nous pouvons donc écrire S_{k-1} sous la forme S_{k-1} =b (j, m) Cette notation signifie que l'état S_{k-1} , est celui qui précède l'état $S_k = m$, sachant que le bit d'entrée au codeur à l'instant (k-1) est j. La métrique peut alors s'écrire :

$$\alpha_k(m) = \sum_{j=0}^{1} P\left(R_1^{k-2} \middle| S_{k-1} = b(j,m)\right) P(d_{k-1} = j, S_{k-1} = b(j,m), R_{k-1}) \quad \textbf{(III.23)}$$

Ou :

$$\alpha_{k-1}(m) = P\left(R_1^{k-1} \middle| S_{k-1} = m\right) \quad \textbf{(III.24)}$$

Soit :

$$\alpha_{k-1}\left(b(j,m)\right) = P\left(R_1^{k-1} \middle| S_{k-1} = b(j,m)\right) \quad \textbf{(III.25)}$$

De même :

$$\gamma_k^i(m) = P(d_k = i, S_k = m, R_1^k) \quad \textbf{(III.26)}$$

Donc :

$$\gamma_k^i(b(j,m)) = P(d_k = i, S_k = b(j,m), R_1^k) \quad \textbf{(III.27)}$$

Nous reconnaissons ces termes dans l'équation (III.23). Nous en déduisons que :

$$\alpha_k(m) = \sum_{j=0}^{1} \alpha_{k-1}\left(b(j,m)\right)\gamma_{k-1}^j\left(b(j,m)\right) \quad \textbf{(III.28)}$$

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. [8]

6 3. *Métrique d'état en arrière*

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 :

$$\beta_k(m) = P\left(R_k^N \middle| S_k = m\right) \quad \textbf{(III.29)}$$

Développons cette expression à l'aide de la règle de Bayes.

$$\beta_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} P\left(d_k = j, S_{k+1} = i, R_k^N \middle| S_k = m\right) \quad \textbf{(III.30)}$$

$$\beta_k(m) = \sum_{i=0}^{2^M-1} \sum_{j=0}^{1} P\left(R_k^N \middle| S_k = m, d_k = j, S_{k+1} = i, R_k\right) P(d_k = j, S_{k+1} = i, R_k | S_k = m) \quad \textbf{(III.31)}$$

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 :

$$\beta_k(m) = \sum_{j=0}^{1} P\left(R_k^N \middle| S_{k+1} = f(j,m)\right) P(d_k = j, S_k = m, R_k) \quad \textbf{(III.32)}$$

$$\beta_k(m) = \sum_{j=0}^{1} \gamma_k^j(m) \beta_{k+1}(f(j,m)) \quad \textbf{(III.33)}$$

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). [8]

6.4. *Métrique de branche*

Nous avons défini trois métriques et étudié deux d'entre elles. II nous reste donc à nous interroger sur la dernière de ces métriques, à savoir celle de branche. Contrairement aux deux autres métriques, celle-ci ne dépend ni des instants précédents, ni des instants suivants. Seul l'instant présent nous intéresse, Nous avons vu que :

$$\gamma_k^i(m) = P(d_k = i, S_k = m, R_1^k) \quad \textbf{(III.34)}$$

Donc, toujours grâce à la règle de Bayes :

$$\gamma_k^i(m) = P(R_k | d_k = i, S_k = m) P(d_k = i, S_k = m) \quad \textbf{(III.35)}$$

$$\gamma_k^i(m) = P(R_k | d_k = i, S_k = m) P(S_k = m | d_k = i) P(d_k = i) \quad \textbf{(III.36)}$$

Nous savons également que $R_K = (r_k^s, r_k^p)$, donc :

$$\gamma_k^i(m) = P(r_k^s | d_k = i, S_k = m) P\left(r_k^p \middle| d_k = i, S_k = m\right) \times P(S_k = m | d_k = i) P(d_k = i) \quad \textbf{(III.37)}$$

Ceci constitue notre nouvelle expression pour cette métrique. Nous constatons qu'elle dépend des symboles recueillis à la sortie du canal. Ainsi, le type de canal va influer sur notre algorithme. Cette métrique ne changera donc pas en fonction du canal. [8]

7. 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

1. Introduction

Ce chapitre présent 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.

2. 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 **γ** à toutes les étapes pour les deux entrées. L'initialisation des probabilités **α** pour première étape se fait en supposant l'encodeur pour être dans l'état 00 initialement. Donc α est initialisé comme suite :

$\alpha_0(00) = 1$, $\alpha_0(01) = 0$, $\alpha_0(10) = 0$, $\alpha_0(11) = 0$.

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 :

$$\mathbf{P}(X^k = 1) = \frac{e^{\Lambda_1^k}}{1+e^{\Lambda_1^k}} \quad \textbf{(IV.1)}$$

$$\mathbf{P}(X^k = 0) = \frac{1}{1+e^{\Lambda_1^k}} \quad \textbf{(IV.2)}$$

Le deuxième décodeur répète le même processus pour décoder les bits. Dans la simulation ce processus itératif est terminé lorsque la probabilité priori pour l'un de l'entrée devient 1.

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

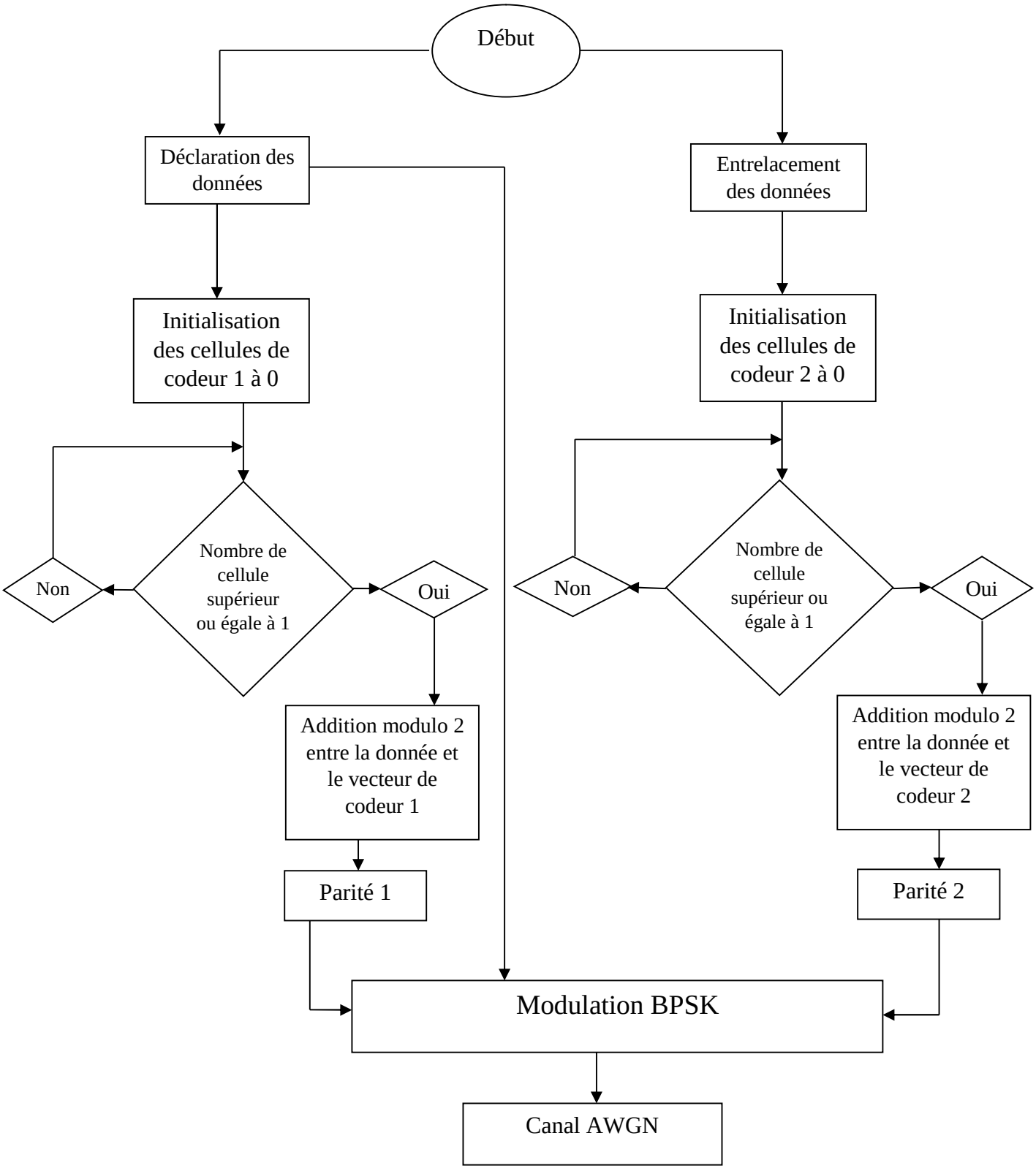


Figure IV.1 : Principe de codage turbo

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

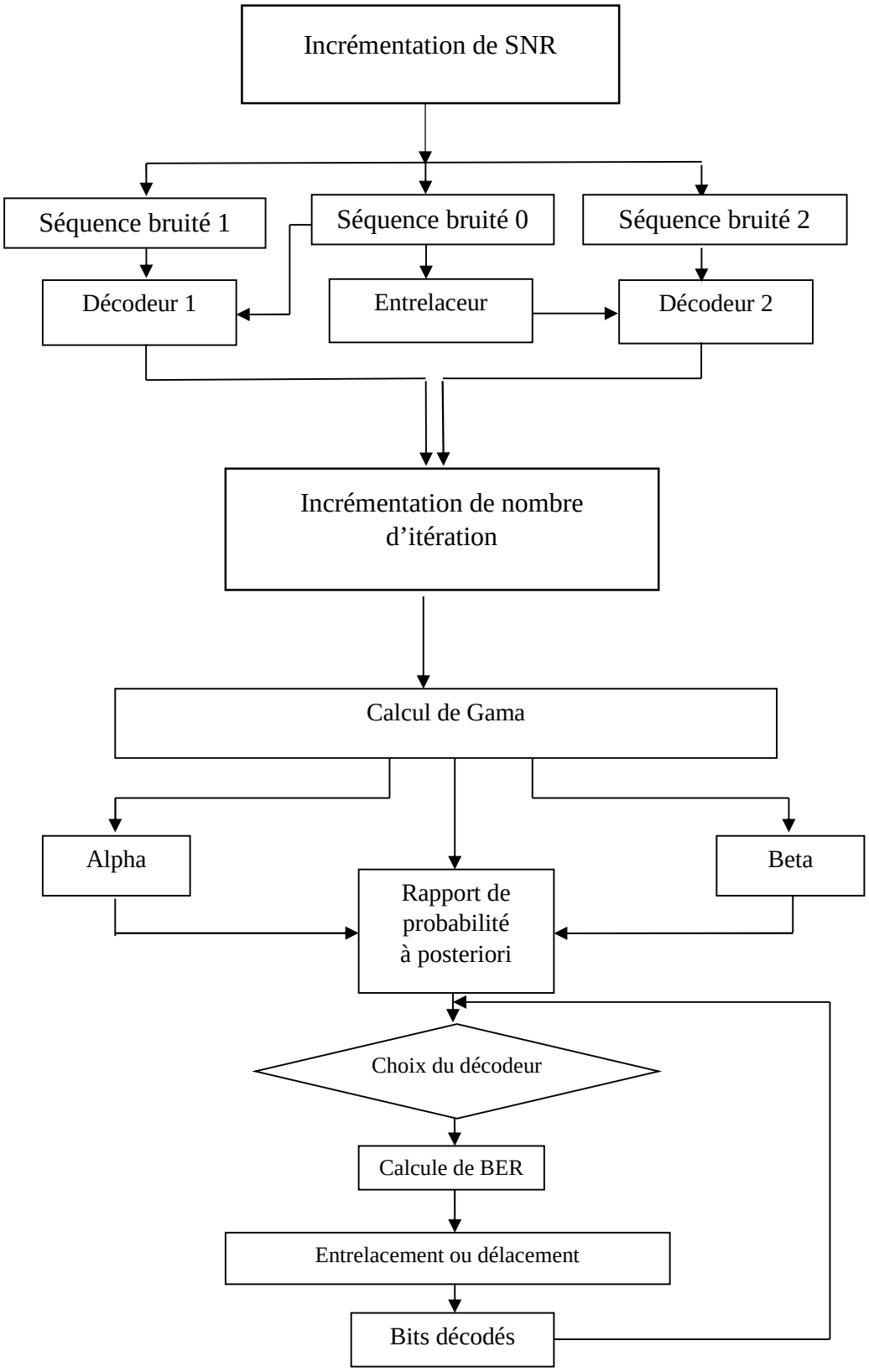


Figure IV.2 : Principe de décodage turbo

3. Résultats de simulation

Pour la simulation, nous avons utilisé N = 10000 bits d'information et SNR varié de 0 dB à 9 dB. Temps total pour 4 itérations et toutes les valeurs de SNR allant avec un incrément de 0,5 dB (soit 19 valeurs de SNR) il a fallu 32 secs. Ainsi le temps pris pour chaque valeur de SNR = 1,68 sec. Pour chaque itération, il est de 0,42 secs, ce qui est très faible compte tenu du grand nombre de bits d'information.

Les TEB simulés sont présentés aux figures suivantes :

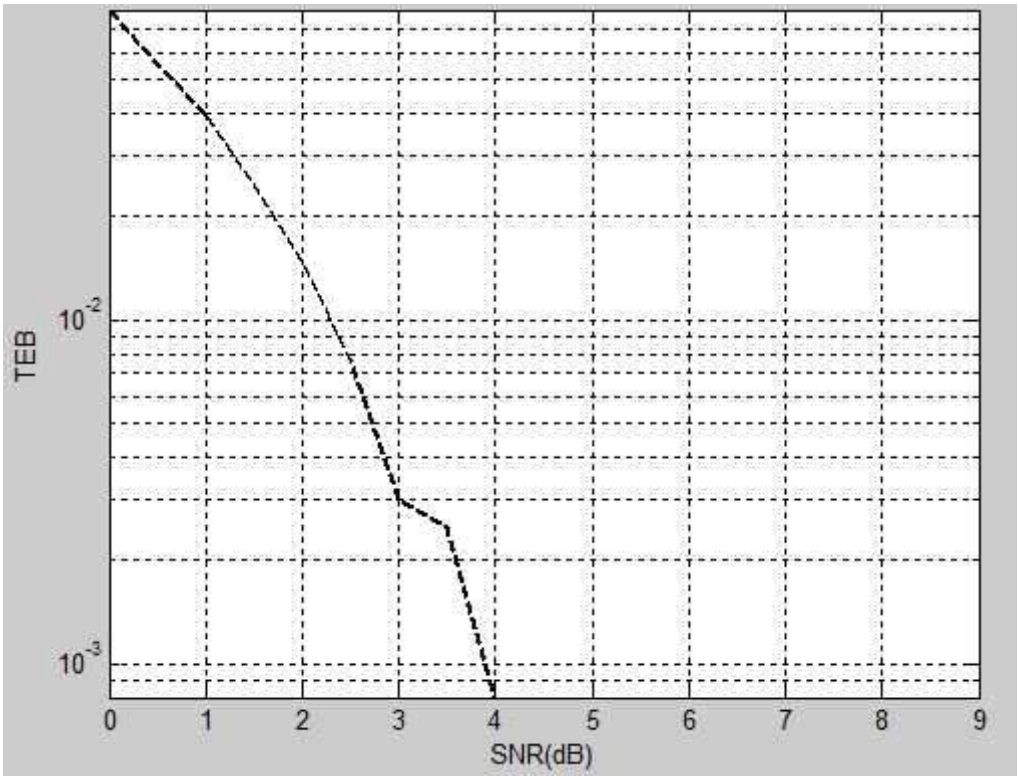


Figure IV.3 : TEB après 1er itération

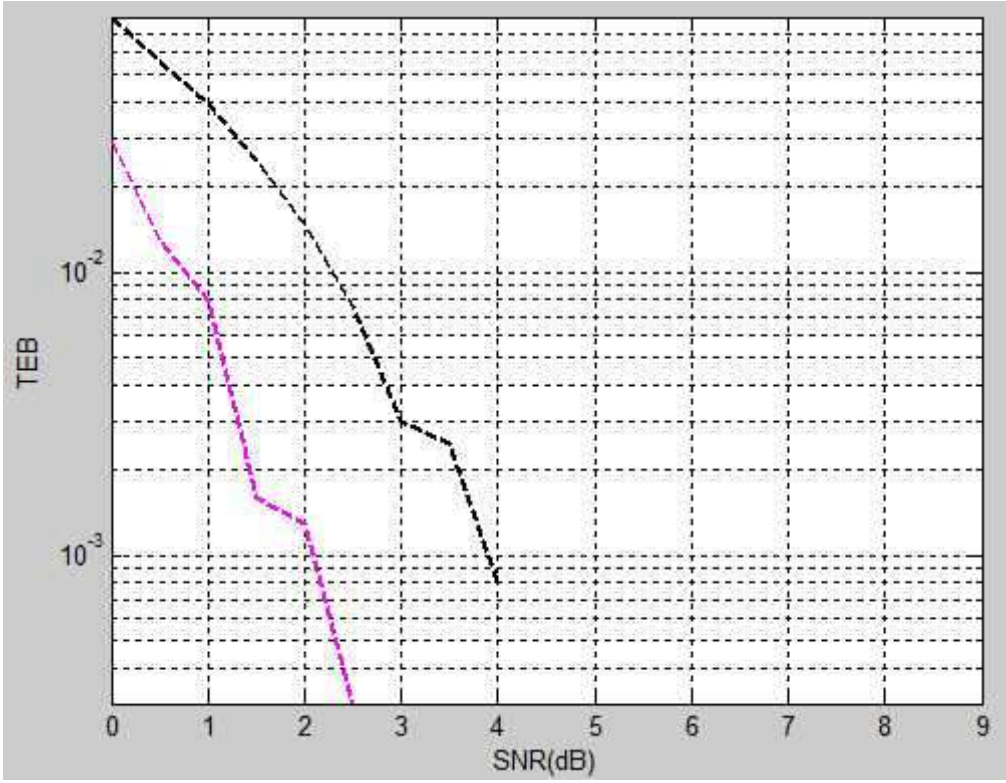


Figure IV.4 : TEB après 2eme itération

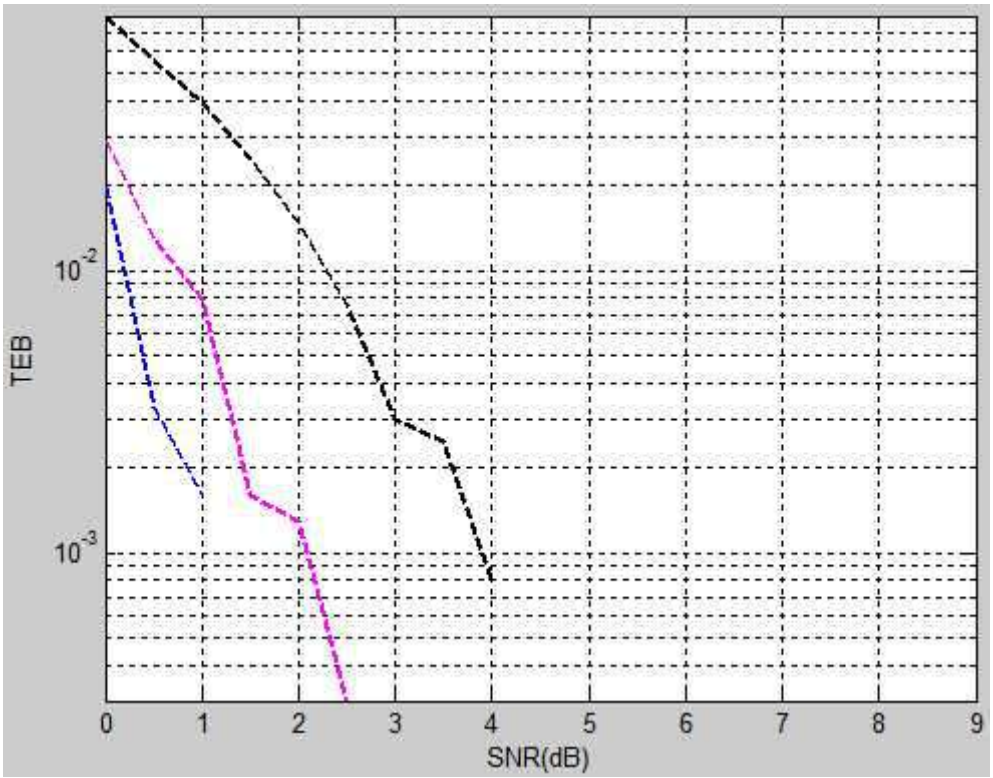


Figure IV.5 : TEB après 3eme itération

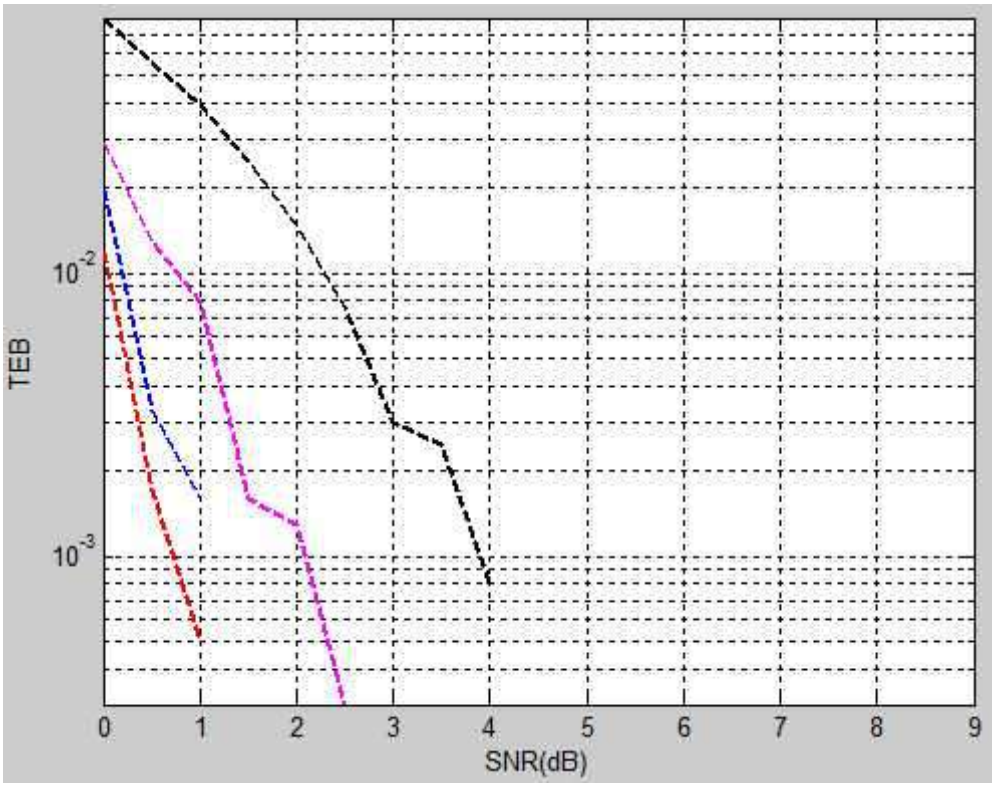


Figure IV.6 : TEB après 4eme itération

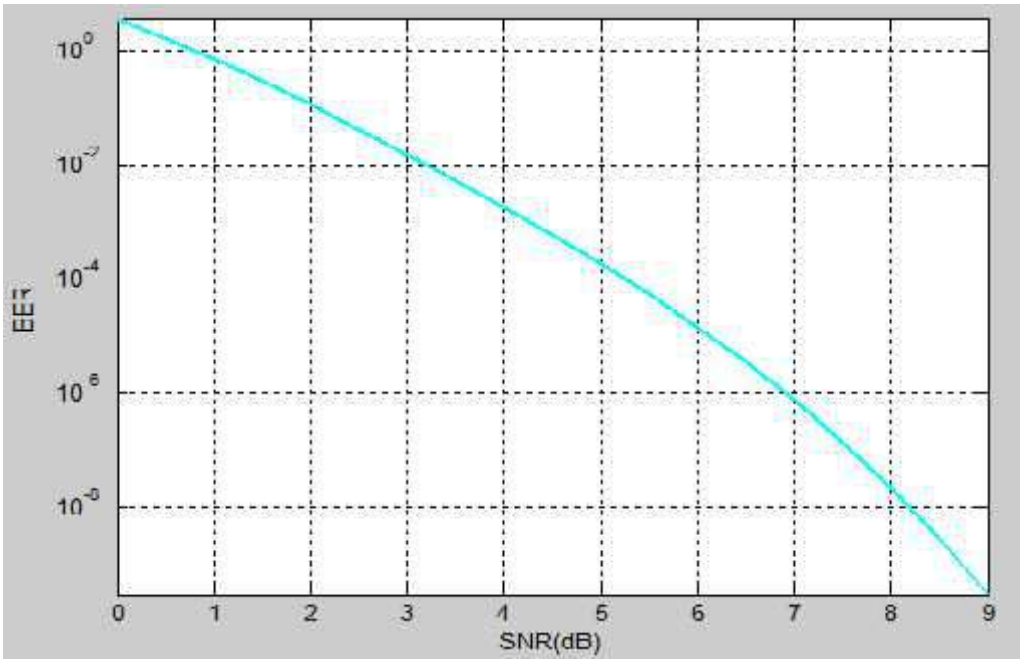


Figure IV.7 : TEB théorique

À partir de ces graphes, nous pouvons observer 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 4 itérations. Aussi le TEB devient presque zéro

après 1 dB en 4 itérations. Donc le code Turbo est beaucoup mieux que code convolutionnel correspondant.

L'expression utilisée pour le TEB théorique est :

$$P_e = \sum_{j-0}^{1} 2^{j+1} (j+5)\, \mathbf{Q}\left(\sqrt{(j+5)S}\right) \qquad \textbf{(IV.3)}$$

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

La comparaison de performances de turbo code est montré dans la Figure IV.8.

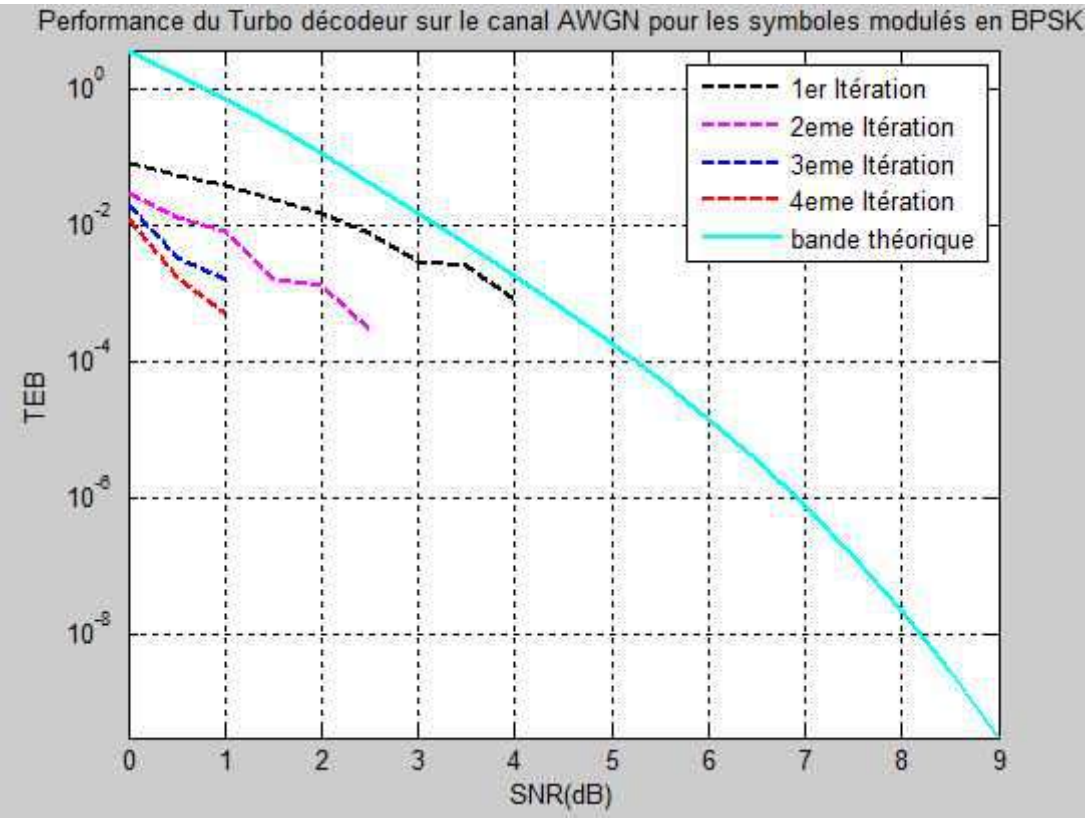


Figure IV.8 : Comparaison de TEB avec la limite théorique

Les valeurs de TEB pour les 4 itérations et la limite théorique sont présentées au tableau IV.1.

Tableau IV.1 : Valeurs de TEB pour 4 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	0.0800	0.0290	0.0173	0.0119	3.4532
0.5	0.0554	0.0129	0.0069	0.0017	1.6124
1.0	0.0392	0.0079	0.0016	0.0005	0.7065
1.5	0.0247	0.0016	0.0000	0.0000	0.2911
2.0	0.0148	0.0006	0.0003	0.0000	0.1134
2.5	0.0076	0.0000	0.0000	0.0000	0.0422
3.0	0.0030	0.0000	0.0000	0.0000	0.0151
3.5	0.0025	0.0002	0.0000	0.0000	0.0053
4.0	0.0008	0.0000	0.0000	0.0000	0.0018
4.5	0.0000	0.0000	0.0000	0.0000	0.0006
5.0	0.0000	0.0000	0.0000	0.0000	0.0002
5.5	0.0000	0.0000	0.0000	0.0000	0.0001
6.0	0.0000	0.0000	0.0000	0.0000	0.0000
6.5	0.0000	0.0000	0.0000	0.0000	0.0000
7.0	0.0000	0.0000	0.0000	0.0000	0.0000
7.5	0.0000	0.0000	0.0000	0.0000	0.0000
8.0	0.0000	0.0000	0.0000	0.0000	0.0000
8.5	0.0000	0.0000	0.0000	0.0000	0.0000
9.0	0.0000	0.0000	0.0000	0.0000	0.0000

4. Conclusion

À partir des résultats de simulation de TEB pour le code Turbo, nous pouvons conclure 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. Bien que la complexité de temps peut être un facteur dans le cas d'un grand nombre de bits, comme une quantité importante de calculs en treillis sont impliqués.

Conclusion générale

Ce projet nous a permis d'approfondir nos connaissances dans le domaine du codage de canal et plus précisément dans le domaine de turbo code.

Nous avons commencé par un rappel sur les systèmes de communication numériques classiques, puis nous avons parlé sur le principe des codeurs convolutionnels et la concaténation de ces derniers ainsi que l'algorithme de décodage plus particulièrement par l'algorithme MAP.

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.

Par la suite ; ce programme peut être utilisé dans plusieurs domaine d'applications tel que la 3G, UMTS, LTE …

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 de calcule Matlab.

En fin nous espérons que le programme que nous avons fait servira les étudiants des promotions futures et leur permettra d'atteindre des résultats plus performants dans ce domaine.

Bibliographie

[1] A. Glavieux & M. Joindot, " Communications numériques introductions ", Masson, 1996.

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

[3] Patrice. Kadonik " Les modulations numériques " , Bordeaux , 2000.

[4] O. Pothier, " Introduction générale au codage canal ", Codage de canal et turbo code, 2000.

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

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

[7] Christian B. Schlegel, Trellis and turbo coding, IEEE Press Series on Digital & Mobile Communication, Ericsson Professor of digital Communication Lund, University Sweden,2004.

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

[9] Mohamed. Zebdi, " Etude comparative des algorithmes de correction d'erreurs sans voie de retour pour la communication sans fil ", Université de Québec à trois Rivières, Avril, 2004.

[10] Claude Berrou, Alain Glavieux et Punya Thitimajshima, " Near Shannon limit error correcting coding and decoding", IEEE International Conference on Communications, vol. 2,23-26 mai 1993.