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'études

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

Spécialité : système de télécommunication

THEME

ETUDE ET MISE EN OUVRE DES CODAGES DU CANAL : LINEAIRE ET CONVOLUTIF

Proposé et dirigé par :

Mr. *MIMEN Bakhti*

Préparé et présenté par :

M. ***BELAIDI Noureddine***

Promotion : 2013/2014

Dédicace

Avec l'aide de Dieu, tout puissant, j'ai pu achever ce modeste travail que je dédie:

A ceux qui m'ont tout donné sans rien me demander et à qui je dois énormément et qui je ne remercierais jamais assez :

- *A mon très cher père (que Dieu ait son âme).*
- *A ma mère qui m'a éclairée mon chemin. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour, toute au long de mes études.*
- *A mes frère.*
- *A mes sœurs.*

Noureddine BELAIDI

Remerciements

Je tiens à remercier en premier lieu le DIEU qui nous a donné la volonté et le courage afin d'arriver à ce modeste travail.

Je tiens à remercier mon promoteur : Mr. ***B. MIMEN*** *qui m'a orienté et aidé pendant toute la durée de notre travail par leur critique et leur suggestion nombreuse.*

Je tiens à remercier tous les membres de jury qui ont accepté de juger ce travail.

Mes sincères remerciements vont également à tous les enseignants du département de Génie électrique de l'Université de Médéa, et à tous les enseignants qui ont participés à notre formation de la première année de formation jusqu'à la 2ième année Master.

Résumé :

Ce travail consiste à étudier et mettre en œuvre des techniques du codage du canal. Où la perturbation des informations transmises à travers un canal AWGN, nécessite l'application du codage du canal (codage convolutif) et un décodage qui base sur l'algorithme de Viterbi.

Une étude des performances de transmission par codage convolutif, en présence du bruit blancs gaussien, en traçant des courbes de variation du BER en fonction du rapport Eb/N0. L'application du codage a été effectué sur deux approches différentes de configuration, celle à décision Hard et l'autre à décision Soft).

Mots clés :

Codage ; convolutif ; Viterbi ; transmission numérique ;Canal ; BER ; SNR, bruit blanc ; AWGN

Abstract:

This work is to study and implement of technical channel coding. Where the distortion of the transmitted informations through an AWGN channel requires the application of channel coding (convolutional coding) and decoding based on the Viterbi algorithm.

A study of the transmission performance by convolutional coding in presence of Gaussian white noise, by tracing curves of the variation of BER based on Eb / N0. The application of the coding was done on two different approaches of configuration, the hard decision and the Soft one.

Keywords:

coding; convolutif ; Viterbi ; Digital Transmission ; Channel ; BER; SNR; white noise; AWGN

ملخص :

يهدف هذا العمل إلى دراسة و تنفيذ تقنيات تشفير القناة. إن تفادي ضياع المعلومات المنقولة عبر قناة AWGN يتطلب تطبيق تشفير القناة (التشفير الإلتوائي) وفك التشفير استنادا إلى خوارزية فيتاربي.

هناك دراسة لأداء الإرسال باستعمال التشفير الإلتوائي عبر قناة ذات التشويش الأبيض لغوص، و ذلك برسم منحنيات تغير BER بدلالة انسبة EB/N0 . و قد تم تطبيق هذا التشفير في نوعين من الارسال (واحد ذو القرار الصلب و آخر ذو القرارالبرمجي)

كلمات مفتاحية :

تشفير، الالتواء، فيتاربي، قناة الإرسال الرقمي، معدل الخطأ في البثات (BER)، نسبة الإشارة علي الضوضاء (SNR) ،التشويش الأبيض (AWGN).

Sommaire

Table des matières

Dédicace

Remerciement

Résumé

Table de matières

Table des Figures

Table des Tableaux

Introduction générale

Chapitre I : Notions théoriques du codage

I.1. Introduction ……...2

I.2. Définitions ……..3

I.2.1. Le code ……..……...….3

I.2.2. poids d'un code, distance de hamming et norme d'un vecteur ……3

I.2.3. Code linéaire et matrice génératrice……...…….4

I.3. Techniques de codage……....4

I.3.1. Codage Linéaire ……..…….5

I.3.1.1. Codage Linéaire par Bloc …….……..5

I.3.1.2. Code de Hamming …….……..6

I.3.1.3. Codes de Correction des Erreurs Éclat ……...….6

I.3.1.4. Codage cyclique ……..……..6

I.3.2. Codage Convolutionnel ……...……..…….8

I.4 Conclusion ……..12

Sommaire

CHAPITRE II : Étude des codages cyclique et convolutionnel

II.1 Introduction ……..…..13

II.2 Code en Bloc ……...……13

II.2.1 . C C H S …….13

II.2.1.1 Principe de codage…..……..….13

II.2.1.2 Principe de Décodage……...…….13

II.2.1.2.1 Détection d'erreurs (calcul du syndrome) ……...…14

II.2.1.2.2 Décodeur de MEGGIT …….15

II.2.2. C C H A …….…….17

II.2.2.1. Décodage ……..……18

II.2.2.2. Tests et décisions …….18

II.2.3. Algorithmes de protection :(l'entrelacement)…….19

II.3. Codes convolutionnels…….. 20

II.3.1. Définition ……....20

II.3.2. Codage convolutif de rendement 1/n …….……..22

II.3.3. Codage convolutif de rendement k/n …….…..24

II.3.4. Représentation polynômiale ……....25

II.3.5. Codes convolutifs récursifs systématiques …….…….……25

II.4. Représentation en treillis ……...……..27

II.4.1. Décodage : algorithme de Viterbi …….…..28

II.4.2. Présentation de l'algorithme de Viterbi ……..…29

II.4.3. Distance libre ……...……30

II.4.4. Diagramme d'état et Fonction de transfert……30

II.4.5. Décodage à décision douce en entrée…….…..32

II.4.6. Limitation du décodage à une fenêtre…….34

II.4.7. Codes convolutionnels poinçonnés …….…….34

Sommaire

II.5. Comparaison des codes cycliques et des codes convolutionnels…….....35

II.6. Concaténation……..…36

II.7. Entrelacement…….....36

II.8. Conclusion :……..…37

Chapitre III : Implémentation et discussion des résultats

III.1 Introduction : ……..38

III.2. Codage convolutif et décodage de Viterbi ……....…..….38

III.2.1. Cas de décision dure (hard-decision) ……..……....38

III.2.1.1 :calcule de BER , cas de hard-dicision……..……43

III.2.2 :Cas de décision logiciel (soft-décision) ……...…….44

III.2.2.1 :calcule de BER , cas de soft-dicision……..…..45

III.2.3 Évolution de BER en fonction du rapport E_b/N_0 dans les deux cas :….46

III.3 : Codage et décodage Cyclique :…….47

Conclusion Général

Tables des figure

Figure (I.1) : Un système de communication……...…….....2

Figure(I.2) : Codeur de source et codeur de canal……...……...…3

Figure(I.3) : Matrice génératrice sous forme systématique……...…4

Figure (I.4) : Structure d'un Encodeur……...…..5

Figure (I.5) : Diagramme d'état d'un Encodeur Convolutionnel……...……8

Figure (I.6) : Polynôme Générateur (Encodeur Convolutionnel)……..……..…..9

Figure (I.7) : Encodeur Convolutionnel…….....10

Figure (I.8) : Diagramme de Treillis (Encodeur Convolutionnel)…….……..…10

Figure (I.9) : Codage Convolutionnel Systématique Récursive (Schéma de principe)……...……..11

Figure II.1 : Circuit de codage du code C(7,4)…….……...……...…14

Figure II.2 : Calcul du syndrome du code C (7, 4)…….……...16

Figure II.3: circuit de décodage du code C (7, 4)……..…….....17

Figure II.4 : Code convolutif de rendement r=1/n….……23

Figure II.5 : Code convolutif de rendement 2/3 et de longueur de contrainte L=2 ...24

Figure II.6 : Code convolutif NRNSC (7,5)……....26

Figure II.7 : Code convolutif RSC (1,5/7)……..…….27

Figure III.3 : Codage et décodage Cyclique ……..38

Figure(III.2) : Générateur binaire de Bernoulli ainsi que sa boite de dialogue……..39

Figure(III.3) : Bloc du codeur convolutif ainsi que sa boite de dialogue…….…39

Sommaire

Figure(III.4) Bloc de modulation BPSK ainsi que sa boite de dialogue……..40

Figure(III.5) Bloc de démodulation BPSK ainsi que sa boite de dialogue……..……40

Figure(III.6) : Bloc canal de bruit ainsi que la boite de dialogue……...……..41

Figure(III.7) : Bloc de convertisseur ainsi que la boite de dialogue…….41

Figure(III.8) : Bloc décodeur viterbi ainsi que la boite de dialogue……....42

Figure(III.9) : Bloc bit ainsi que la boite de dialogue……...…42

Figure(III.10) : Bloc d'affichage ainsi que la boite de dialogue…….….43

Figure(III.11) : Courbe du BER en fonction de E_b/N_0 ; cas hard-dicision……..44

Figure(III.12) : Chaine de transmission à décision logiciel…….…..44

Figure(III.13) : Courbe du BER en fonction de E_b/N_0 ; cas de soft-dicision……...45

Figure(III.14) : Courbe d'évolution du BER en fonction de E_b/N_0 …….46

Figure (III.15) : schéma bloc de CRC…….……...47

Figure(III.16) : Bloc générateur CRC ainsi que la boite de dialogue…….47

Figure(III.17) : Bloc sélecteur ainsi que la boite de dialogue…….….48

Figure(III.18) : Bloc de PAD ainsi que la boite de dialogue……48

Figure(III.19) : Bloc de buffer ainsi que la boite de dialogue……..…49

Figure(III.20) : Bloc de MUX ainsi que la boite de dialogue……...…49

Liste Des Tableau

Tableau II.1 : Entrelacement……...……20

Figure(III.10) : Bloc d'affichage ainsi que la boite de dialogue……..…44

Tableau [III.2] : résultats de BER en fonction de EB/N0 _ soft-dicision ……....46

Inroduction Générale

Introduction générale :

La théorie des communications s'intéresse aux moyens de transmettre une information depuis la source jusqu'à un utilisateur à travers un canal.

La nature de la source peut être très variée. Il peut s'agir par exemple d'une voix, d'un signal électromagnétique ou d'une séquence de symboles binaires. Le canal peut être une ligne téléphonique, une liaison radio, un support magnétique ou optique où la transmission peut se faire dans l'espace ou dans le temps.

Le codeur représente l'ensemble des opérations effectuées sur la sortie de la source avant la transmission. Ces opérations peuvent être par exemple la modulation, la compression, le brouillage, l'ajout de redondance pour combattre les effets du bruit, ou encore l'adaptation à des contraintes de spectre. Elles ont pour but de rendre la sortie de la source compatible avec le canal. Enfin le décodeur doit être capable, à partir de la sortie du canal, de restituer de façon acceptable l'information fournie par la source.

Dans notre travail, nous nous intéressons à l'étude et la mise en œuvre du codage convolutif ainsi que son décodage à base de l'algorithme de viterbi. Nous allons appliquer ces algorithmes de codage et décodage dans une chaine de transmission numérique d'approche hard décision et soft décision.

Notre mémoire est subdivisé en trois chapitres :

Le premier chapitre sera consacré à la présentation de quelques notions de base de codages.

Le deuxième chapitre présentera l'étude théorique des codages : convolutif et CRC.

Le troisième chapitre se portera sur l'implémentation de l'algorithme de codage convolutif ainsi que son décodage à base de l'algorithme de viterbi en les appliquant sur deux approches de transmission en présence de bruit AWGN (approches : Hard décision et Soft décision), et une discussion des résultats sera effectuée.

Finalement, on terminera par une conclusion générale.

CHAPITRE (I)

Notions théoriques du codage

CHAPITRE I :

Notions th éoriques du codage

I.1. Introduction :

La théorie de l'information a été créée par C.E.Shannon dans les années 40. Il s'agit d'une théorie mathématique qui décrit les aspects les plus fondamentaux des systèmes de communication. Elle consiste en l'élaboration et l'étude de modèles pour la source et le canal qui utilisent différents outils comme les probabilités et les automates finis.

Dans ce chapitre, nous présenterons certains de ces modèles qui, bien que considérablement plus simples que les sources et les canaux physiques, permettent de donner une bonne approximation de leur comportement.

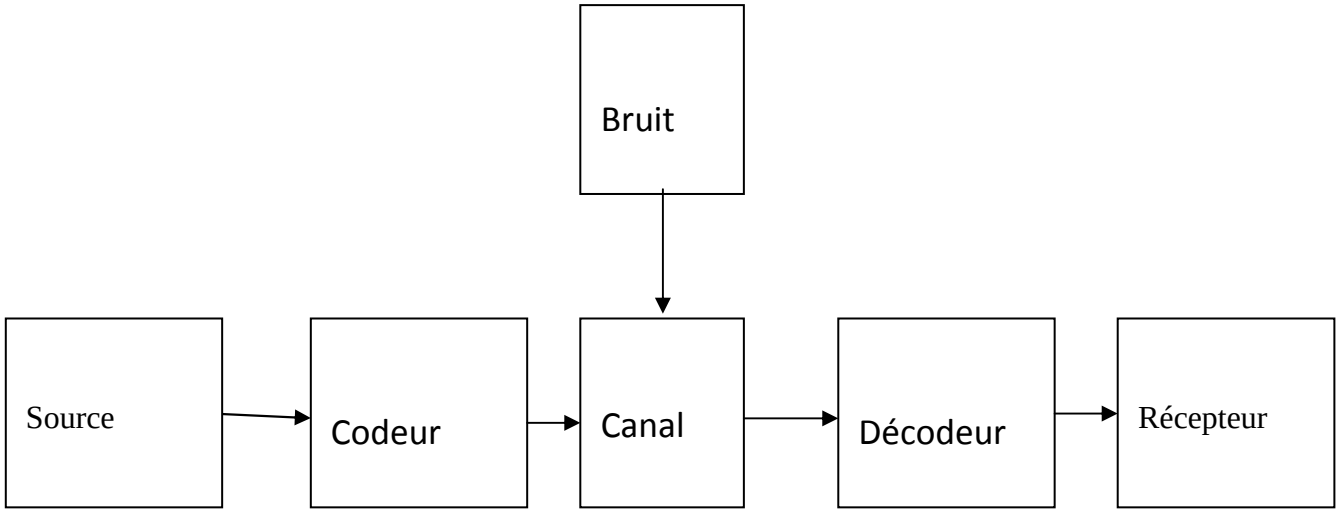
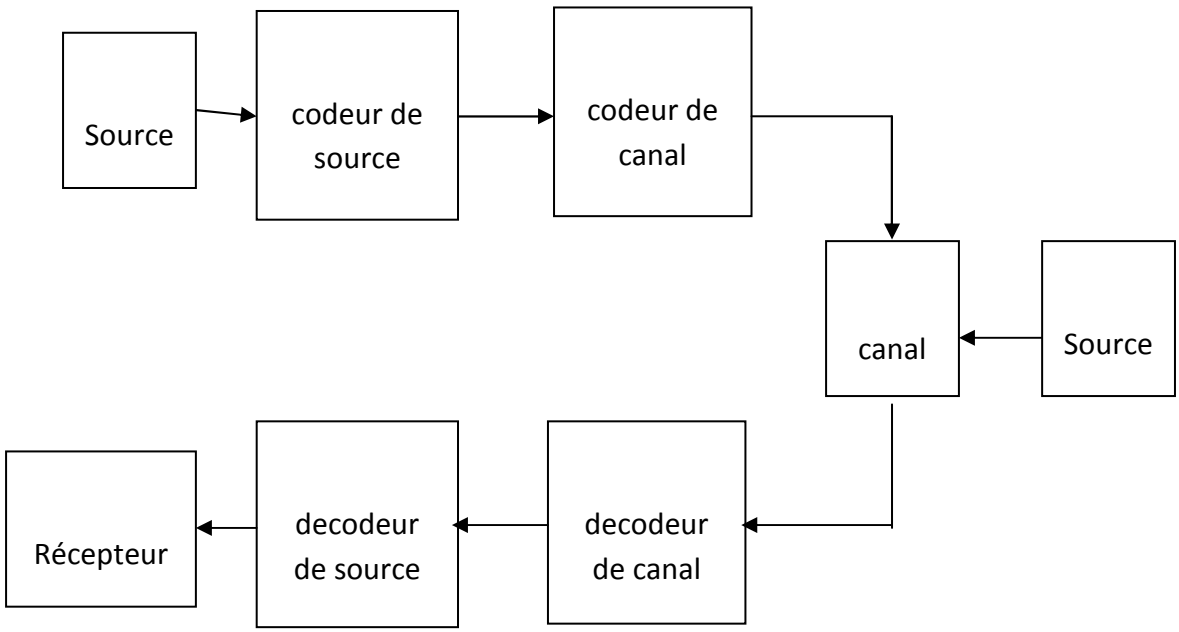


Figure (I.1) : Un système de communication

Pour simplifier, on étudiera séparément les modèles de sources et les modèles de canaux ainsi que leurs codages respectifs.

– Le but du codeur de source est de représenter la sortie de la source en une séquence binaire, et cela de façon la plus économique possible.

– Le but du codeur de canal et de son décodeur est de reproduire le plus fidèlement possible cette séquence binaire malgré le passage à travers le canal bruité.



Figure(I.2) : Codeur de source et codeur de canal

Cette séparation entre codage de source et codage de canal n'implique en pratique aucune limitation sur les performances du système complet

I.2. Définitions :

I.2.1. Le code :

Soit un alphabet A, soit A^* l'ensemble des vecteurs formés à partir des éléments de A. Un code C est un sous-ensemble de vecteurs de A*. C sera dit binaire si A ={0, 1}, un élément de C est appelé mot de code.

I.2.2. poids d'un code, distance de hamming et norme d'un vecteur

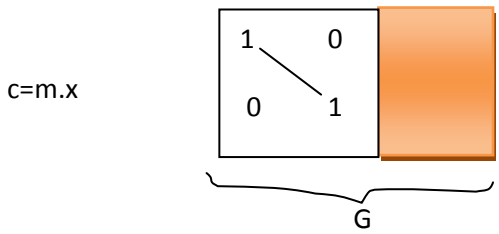
Soient u et v deux vecteurs binaires de taille n ($u = (u_i)$ i=1…n avec $u_i = 0$ ou 1, la même chose pour le vecteur v).

- **a.** le poids de u (noté $w_t(u)$) comme le nombre de 1 dans u.
- **b.** On définit ainsi une norme sur F^n_2. le support de u noté supp(u) comme l'ensemble des indices i tels que $u_i = 1$.
- **c.** la distance de Hamming de u et v (notée d (u, v)) comme le cardinal de l'ensemble des indices i tels que u_i soit différent de v_i. La fonction d (.,.) ainsi définie vérifie les propriétés d'une distance sur F^n_2. On a d (u,v) = $w_t(u + v)$.

Remarque : La distance minimale (notée d) d'un code est le minimum des distances de Hamming entre les mots du code pris distincts deux à deux. C'est aussi le poids du mot (non nul) de plus petit poids du code.

I.2.3. Code linéaire et matrice génératrice

Soit C un code linéaire [n, k, d] sur Fq, une matrice génératrice G est une matrice (k × n) dont les lignes forment une base de C. Si c appartient à C alors c est une combinaison linéaire de lignes de G. Si G = (I_k|A_k×x (n_k)) avec I_k la matrice identité de dimension k et A_k×(n-k) une matrice (k, n -k), on dit que G est sous forme systématique.



Figure(I.3) : Matrice génératrice sous forme systématique

I.3. Techniques de codage

Le codage est une modélisation *(Mapping)* d'une série semi-finie de séquences d'éléments *GF(q),* (Gaulois Field)[1], pour une valeur quelconque du nombre de séquences élémentaires *ko'* dans une fenêtre du codage *m (Encoding Window).*

Parmi les principales propriétés d'un encodeur (Voir Fig. I.4), nomons la contrainte de longueur *(Constraint Length) K = M* + 1, ainsi que le rapport du codage de l'information R. Ces paramètres réunis définissent la longueur de la séquence reçue après codage *(Word Length)* ou le *(Block Length)* n = (M + 1) n_o.

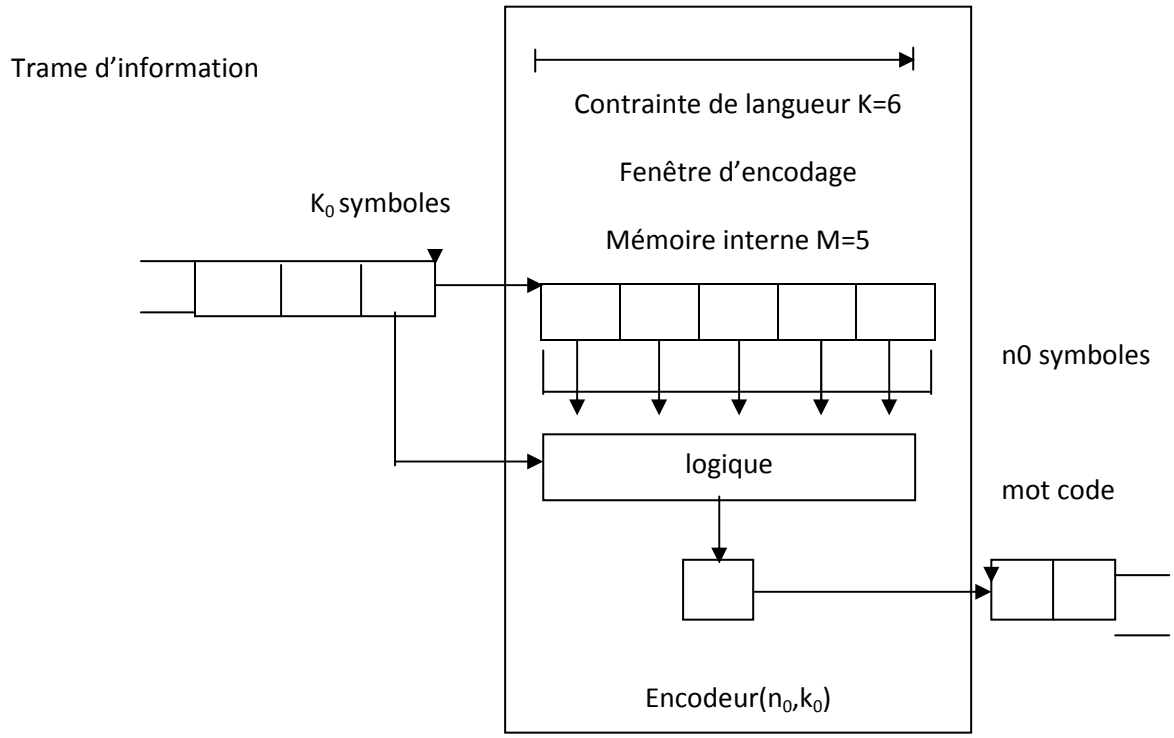


Figure (I.4) : Structure d'un Encodeur

Où *no* définie le nombre d'éléments contenus dans un mot code, après codage de *ko* symboles représentant un seul bit de donnée. La valeur de *ko* déprend du type de modulation choisi pour assurer la transmission des données, cette valeur est réduite à un dans le cas d'une modulation *(Binatry Phase Shift Key).*

I.3.1. Codage Linéaire :

I.3.1.1. Codage Linéaire par Bloc :

Un codage linéaire par *Bloc (n,k)* est complètement défini par $N = 2k$ séquences binaires, de longueur *n* nommées mots codes [2]. Un mot code **C'** est constitué de *N* éléments c_i , pour $1 \leq i \leq 2k$

$$\mathbf{c} = \{c_1, c_2, c_3, \dots, c_N\} \qquad (I.1)$$

Chaque séquence de longueur *n* de C_i est composée de valeurs binaires 0,1.

On désigne par la distance de Hamming entre deux codes C_i et C_j , le nombre des composantes sur lesquels, les deux mots codes diffèrent *(d (C_i ,C_j)).* Par conséquent, la

distance minimale pour un code, au sens de Hamming, est par définition la distance entre n'importe quels deux mots codes *:*

$$d_m = \min_{\substack{i \neq j \\ c_i \neq c_j}} d(c_i, c_j) \qquad \text{(I.2)}$$

Le principal objectif, quant à l'utilisation d'un processus de codage pour un système de communication, est de faire augmenter le plus possible la distance Euclidienne entre les signaux transmis, et par conséquent, réduire la probabilité d'erreur relative à niveau de puissance donné. Ceci revient à dire que le meilleur instrument de mesure, permettant une comparaison des performances entre deux codes, revient tout simplement à une comparaison de la distance de Hamming entre les différents mots codes.

Toutefois, la comparaison de toutes les distances entre n'importe quels deux mots code, reste difficile et parfois impossible. C'est pourquoi la comparaison entre les différents codes est généralement basée sur le calcul de la distance minimale (d_{min}). Pour une valeur donnée de *(n,k),* le code ayant une (d_{min}) élevée est plus performant que celui avec une valeur inférieure de distance minimale *(d_{min}).* Parmi les différentes classes du codage linéaire par Bloc on peut distinguer le codage de Hamming, le codes de correction des erreurs éclats, et le codage cyclique.

I.3.1.2. Code de Hamming :

Un codage linéaire par Bloc défini avec $n = 2^m - 1$, $k = 2^m - m - 1$ et $d_m = 3$, pour un entier $m \geq 2$. Ces codes sont capables de produire une capacité de correction pour des erreurs singulières (sur un seul bit), avant décodage avec un rapport de décodage

$$R_c = \frac{2^m - m - 1}{2^m - 1} \qquad \text{(I.3)}$$

La valeur du rapport de codage tend vers 1 pour des valeurs élevées de *m.*

Les codes de Hamming sont des codes ayant un rapport de codage R élevée, avec une distance minimale d_m relativement basse ($d_m = 3$), c'est pourquoi les codes de Hamming ont une capacité de correction d'erreur limitée.

I.3.1.3. Codes de Correction des Erreurs Éclat :

La plupart des codes linéaires par Bloc ont été conçus afin de corriger les erreurs aléatoires totalement indépendantes des autres erreurs survenues sur le canal de transmission. Certains modèles de canal, y compris un canal *à bruit blanc gaussien additif (Aditve White Gaussian Noise)*, peuvent être catégorisés comme étant un canal à erreurs aléatoires. Cependant, pour d'autres modèles de canaux physiques, cette hypothèse d'une génération d'erreurs indépendamment du canal n'est pas toujours valide, le canal d'évanouissement *(Fading Channel)* en est le parfait exemple [2]. Si le canal est en grande atténuation *(Deep Fade)*, un grand nombre d'erreurs éclat *(Burst Error)* peuvent avoir lieu. On peut utiliser n'importe quel code pour correction d'erreur aléatoire, en vue de corriger les erreurs en éclats, pourvu que le nombre d'erreurs reste inférieur à la moitié de la distance minimale *(Minimum Distance)* du code lui-même.

La méthode effective pour la correction des erreurs en éclats est l'interfoliage des données codées, de manière à ce que les erreurs soient aléatoirement distribuées sur plusieurs mots codes [3][4], plutôt que quelques mots code. Ainsi, les erreurs pouvant apparaître dans un Bloc resteront relativement faibles, ce qui permet l'utilisation d'un code de correction des erreurs aléatoires. Durant la réception, un défoliage *(Deinterleaving)* est utilisé pour neutraliser l'effet de l'interfolieur *(lnterleaver).*

I.3.1.4. Codage cyclique

Les codes cycliques représentent une classe des méthodes de codage linéaire, dont le décalage cyclique d'un mot code aléatoirement choisi. C est aussi un mot code. Il est d'autant plus simple de représenter chacun de ces mots codes par un polynôme appelé polynôme des mots codes *(Code Word Polynomial).* Un polynôme de mot code correspondant à C = (C_1, C_2, C_3,…, C_n) est défini par :

$$C(p) = \sum_{i=1}^{n} c_i P^{n-1} = c_1 P^{n-1} + c_2 P^{n-2} + \cdots + c_{n-1} p + c_n \qquad (I.4)$$

Polynôme de mot code ($C^{(1)}$= (C_2, C_3, C_4,…, C_n)) le décalage cyclique de C est

$$C^{(1)} = C_2 P^{n-1} + C_3 P^{n-2} + \cdots + C_{n-1} P^2 + C_n P + C_1 \qquad (I.5)$$

Avec une relation entre C(I) *(p)* et *C(p)*

$$C^{(1)}(p) = pC(p) \qquad \forall \quad modulo\ (p^n + 1) \qquad \text{(I.6)}$$

D'une façon générale, *i* décalage du polynôme de mot code est:

$$C^{(i)}(p) = p^i C(p) \qquad \forall \quad modulo\ (p^n + 1) \qquad \text{(I.7)}$$

Le fait que n' importe quel polynôme de mot code est un produit d'un générateur polynomial avec le polynôme de la séquence d'information implique que C = *(c_1,c_2, c_3, ..., c_{n-1},c_n),* est une convolution discrète des séquences *U = (u_1,u_2, u_3, …, u_k) et P = (1,P_2,P_3,... ,P_{n-k} ,1).* Ce qui est très important quant au design d'un encodeur cyclique. En comparant un code cycle par rapport aux autres classes des encodeurs d'un codage linéaire par Bloc, un encodeur cyclique possède une structure interne très facile à implémenter avec de simples registres à décalage.

I.3.2. Codage Convolutionnel :

Un processus de codage linéaire et invariant dans le temps, avec une contrainte de longueur finie, est appelé codage convolutionnel (CC) présenté à la Figure(I.7) La représentation du CC varie selon le stade de mise en œuvre [5].

Dans le cas d'une implémentration VLSI, un registre à décalage permet la réalisation d'un code convolutionnel sur silicium. Tandis que pour un processus de décodage séquentiel, la représentation du (CC) apparaît selon l'un des schémas de représentations suivants :

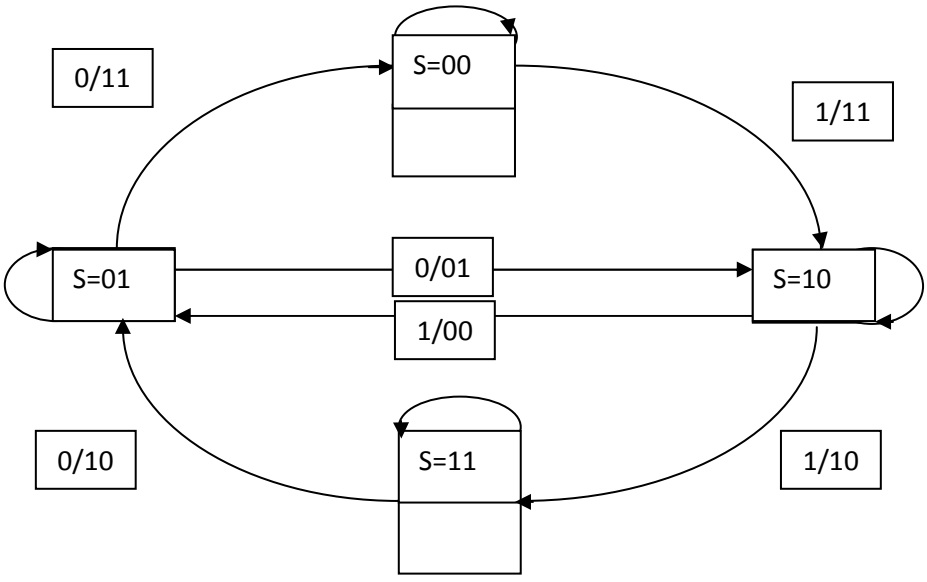


Figure (I.5) : Diagramme d'état d'un Encodeur Convolutionnel

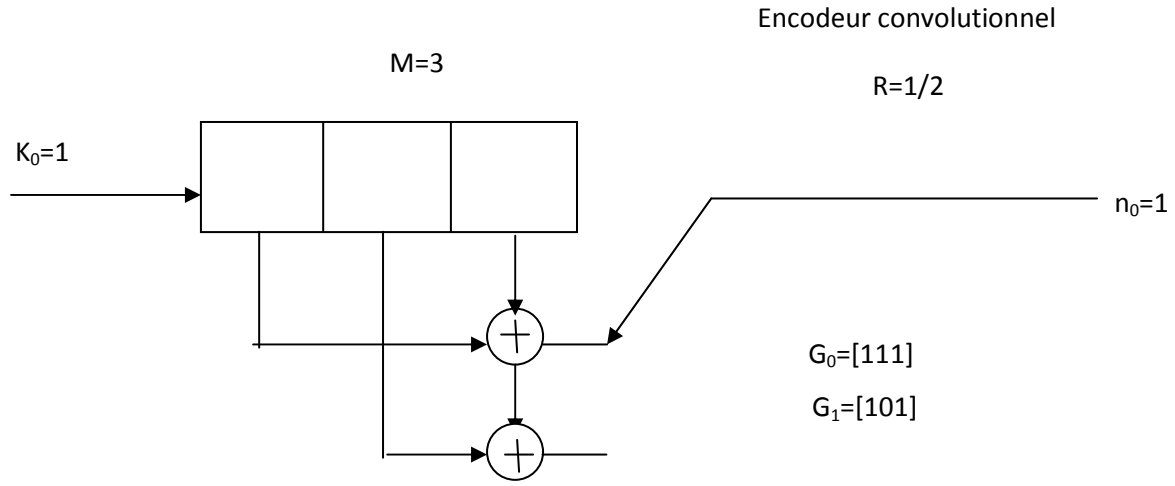


Figure (I.6) : Polynôme Générateur (Encodeur Convolutionnel)

a) Treillis: une représentation temporelle des différentes transitions entre états, utilisée principalement par l'algorithme de décodage de Viterbi (Fig. I.8).

b) Matrice : une représentation utilisée lors d'un décodage par syndrome [1].

c) Polynôme: Pour la vérification de la fiabilité du codage *(Catastrophique/non catastrophique code)* (Fig. I.6).

d) Diagramme d'état: représentation des propriétés de distance entre tous les états possibles (Fig. I.5).

L'encodeur convolutionnel simple est une machine déterministe à nombre fini d'états, par conséquent toutes les transitions probables entre ces états peuvent être prédéfinies, condition essentielle pour la mise en œuvre d'un algorithme de décodage.

Pour un encodeur convolutionnel de rapport R = 1/2 et de contrainte de longueur *K*, l'entrée de l'encodeur en temps *k* est le bit u_k et le mot code correspondant est C_k est le couple *binaire* (X_k,Y_k) avec :

$$X_k = \sum_{i=0}^{L-1} g_{1i} u_{k-i} \qquad modulo\ 2 \qquad g_{1i} = 0.1 \qquad \text{(I.8)}$$

$$Y_k = \sum_{i=0}^{L-1} g_{2i} u_{k-i} \qquad modulo\ 2 \qquad g_{2i} = 0.1 \qquad \text{(I.9)}$$

où G_1:$\{g_{1i}\}$,G_2 :$\{g_{2i}\}$sont les deux polynômes générateurs de l'encodeur. (I.8). (I.9)

D'autre part, l'information en sortie issue de cette encodeur $C_k(X_k,Y_k)$

Figure (I.9) n'est pas en fonction du bit d'entrée U_k seule, mais de toute la séquence $U_{k-1}, U_{k-2}, \ldots, U_{k-M}$ de l'information mémorisée, M représente le nombre de cases mémoires internes.

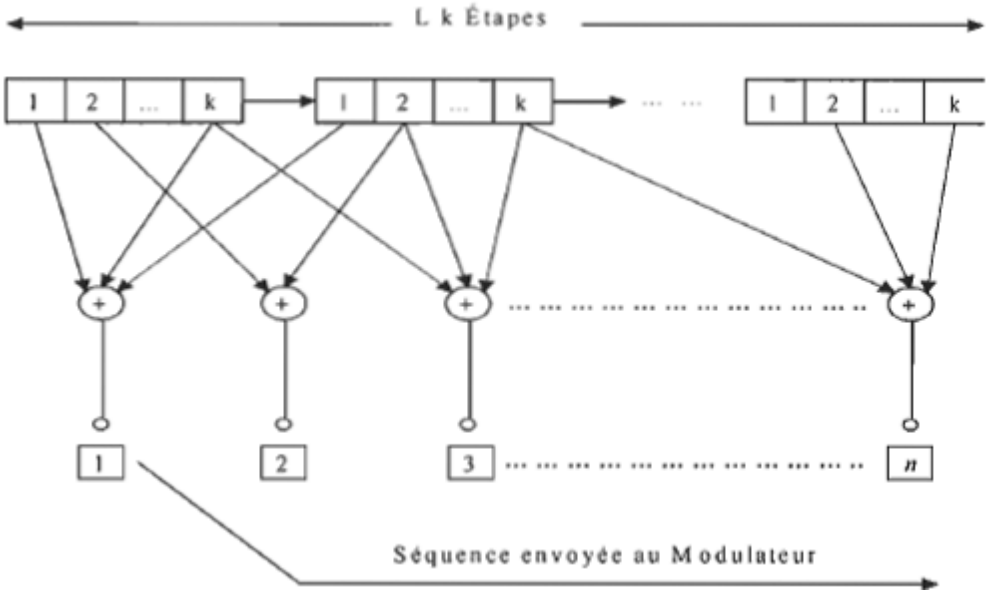


Figure (I.7) : Encodeur Convolutionnel

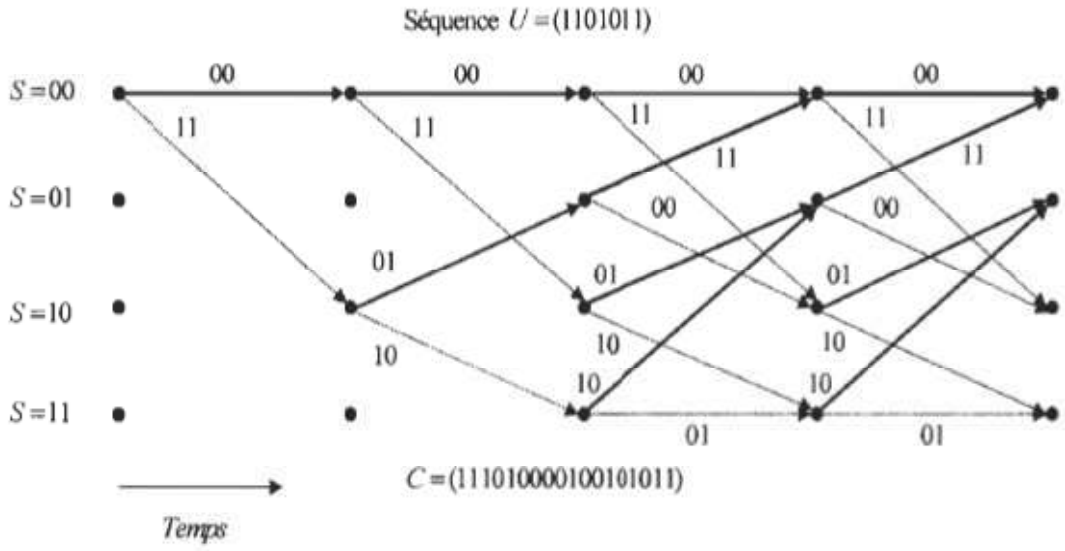


Figure (I.8) : Diagramme de Treillis (Encodeur Convolutionnel)

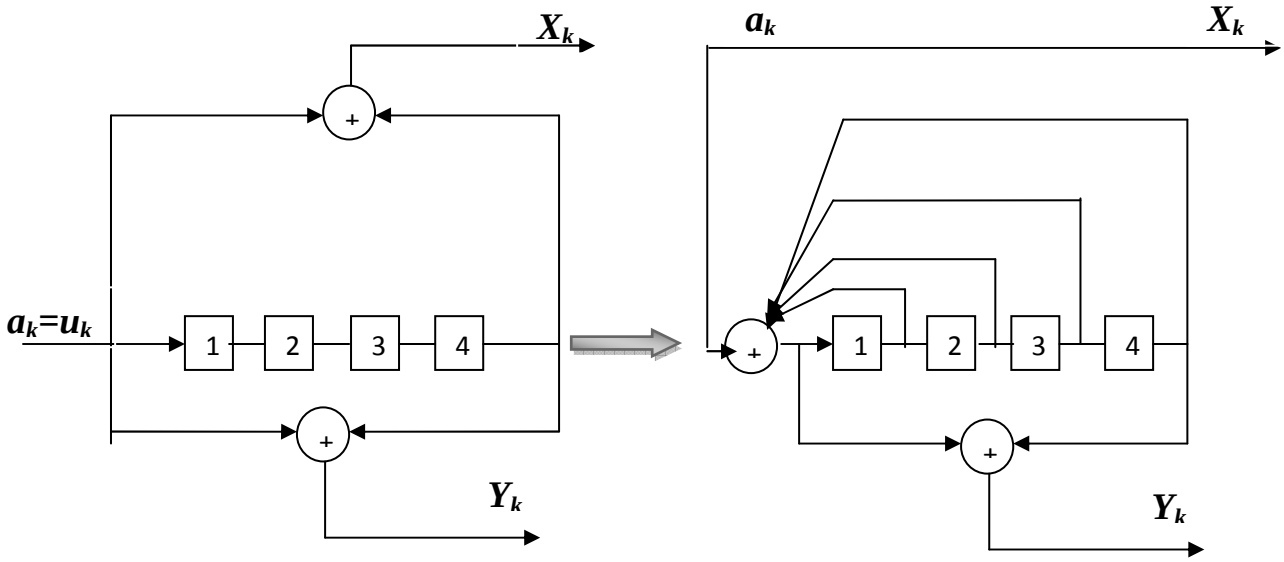


Figure (I.9) : Codage Convolutionnel Systématique Récursive (Schéma de principe)

Bien que cet effet de mémoire donne plus de complexité aux différents algorithmes de décodage, le plus grand avantage reste la possibilité de donner une plus grande immunité au bruit, pour les informations codées avant la transmission.

Il est bien connu que BER d'un encodeur convolutionnel simple NSC *(Non Systematic Code)* est inférieur à celui d'un encodeur systématique, d'une valeur de mémoire interne *M*, pour des valeurs de SNR *(Signal Noise Rate)* élevées. Dans son article [6], Berrou a proposé une nouvelle classe d'encodeur convolutionnel systématique et récursive RSC pour de nouvelles applications. Un RSC binaire de rapport *R =112* est obtenu à partir d'un NSC code, en utilisant une boucle de retour *(Feedback loop)* Fig. I.9, par l'affectation direct d'une des sorties X_k ou Y_k, vers l'entrée u_k. Pour un RSC code, l'entrée du registre à décalage (Mémoire) n'est plus égale à u_k, mais à un nouveau variable a_k . En substituant a_k à la place de U_k ,on a :

$$a_k = u_k + \sum_{i=1}^{L-1} \gamma_i a_{k-i} \qquad \text{(I.10)}$$

La structure du treillis est identique pour le codage en NSC ou le RSC, avec la même *(Free Distance).* Cependant, la différence entre les deux codes est que les séquences de sortie $\{X_k$,$\{Y_k\}$ ne correspondent plus à la séquence d'entrée $\{u_k\}$, que ce soit pour un NSC ou un RSC.

I.4 Conclusion :

Dans ce chapitre nous avons entamé quelques notions de base de codage qui nous intéresse pour la suite de notre travail. Donc, nous avons donné quelques définitions ainsi que les techniques de codages correcteurs du canal, par bloc et convolutif.

CHAPITRE (II)

Étude des codages cyclique et convolutionnel

CHAPITRE II :

Étude des codages cyclique et convolutionnel

II.1 Introduction :

Une grande famille de codes correcteurs d'erreurs est constituée des codes en Blocs et Codes convolutionnel. Ce deuxième chapitre sera consacré pour une analyse plus approfondie des codes, nous traiterons en premier le code en Bloc et ses deux approches, le Code Cyclique de Hamming d'une erreur (C.C.H.S) et le Code Cyclique de Hamming Amélioré (C.C.H.A)

En deuxième, nous présenterons les différents Codes Convolutifs : les Codes Convolutifs de rendement 1/n, le Code Convolutif k/n et les Codes Convolutifs récursifs systématiques. En fin nous présenterons le décodage le plus courant basé sur l'Algorithme de Viterbi.

II.2 Code en Bloc :

Une classe de ses codes est la classe des codes cyclique qui ont la propriété que toute permutation circulaire d'un mot code est un mot code.

Dans cette partie on étudie deux approches de code cyclique de Hamming [7] le code de Hamming cyclique correcteur d'une erreur CCHS et le code cyclique de Hamming améliorés CCHA.

II.2.1 . C C H S :

Un code de Hamming est un code cyclique C(n, k) généré par un polynôme primitif g(x) de degré $m \geq 3$, avec les caractéristiques suivantes :

- Longueur du mot de code $n = 2^m-1$.
- Nombre de bits de contrôle $m = n - k$.
- Nombre de bits d'information $k = 2^m-m-1$.

II.2.1.1 Principe de codage :

Soit $U = (U_0, U_1, \ldots, U_{k-1})$ le k –uplet à coder auquel on associe le polynôme U(x).

Le codage consiste en trois étapes :

a) Pré multiplier le polynôme U(x) associé au k – uplet à coder par x^{n-k}.

b) Obtenir le reste D(x) de la division de $x^{n-k} * U(x)$ par le polynôme générateur g(x).

c) Additionner D(x) et $x^{n-k} * U(x)$ pour obtenir le mot de code $V(x) = D(x) + x^{n-k} * U(x)$.

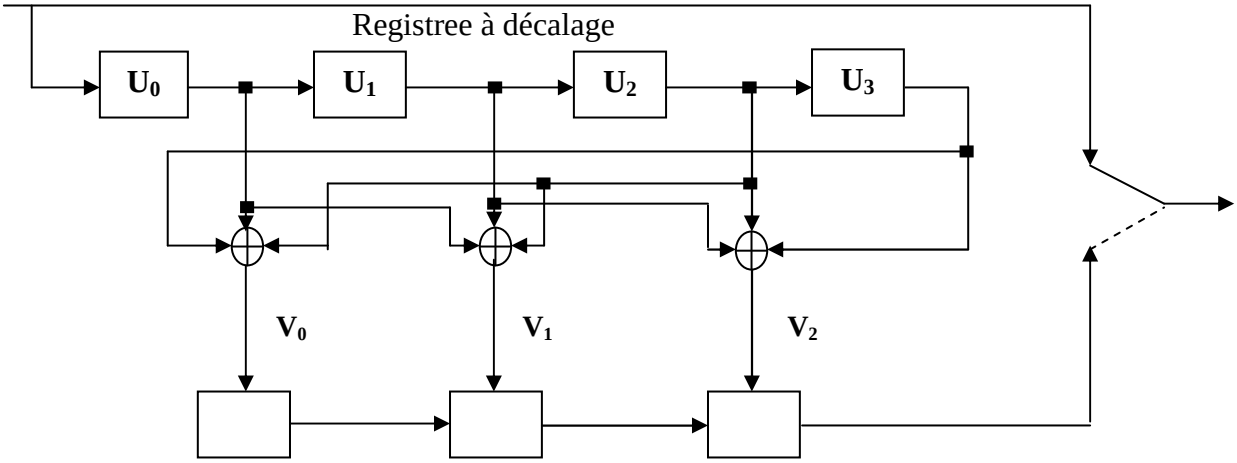


Figure II.1 : *Circuit de codage du code C(7,4).*

On note : V(x) est le polynôme du mot code.

R(x) est le polynôme reçu.

E(x) est le polynôme d'erreur.

V est un mot de code si et seulement si [8] $V(\alpha) = 0$ avec α racine de g(x) (i.e. $g(\alpha) = 0$).

II.2.1.2 Principe de Décodage

Le décodage est l'opération la plus complexe puisqu'elle consiste à vérifier si le mot reçu est correct ou non.

Dans le cas où le mot reçu est erroné, le décodeur doit corriger les erreurs puis extraire l'information utile. Le processus de décodage est divisé en deux étapes :

- Détection d'erreurs dans le mot reçu.
- Correction de ces erreurs dans le cas échéant.

II.2.1.2.1 Détection d'erreurs (calcul du syndrome) :

Le syndrome de R(x) noté S(x) est définit par :

$$S(x) = \textbf{reste}\ (x^{n-k} * R(x) / g(x)) \qquad \text{(II.1)}$$

Avec g(x) : le polynôme générateur et $R_{(i)}(x)$: le polynôme obtenu après le $i^{ième}$ décalage à droite du mot reçu R(x).

Le syndrome correspondant au $i^{ième}$ décalage cyclique de R(x) peut être calculé par:

$$S^{(i)}(x) = \text{reste}\,(x^{n-k} * R_{(i)}(x) / g(x)). \qquad \text{(II.2)}$$

$S^{(i)}(x)$ peut être calculer par [7] :

$$S^{(i)}(x) = \begin{cases} \text{reste}(xS^{(i-1)}(x)/\,g(x)) & \text{si} \quad i > 0 \\ S(x) \quad \text{si} \quad i = 0 \end{cases} \qquad \text{(II.3)}$$

De (II.2) on montre que les $S^{(i)}(x)$ peuvent être calculer à partir de S(x) en utilisant l'équation de récurrence et par conséquent on est pas obligé de réaliser des décalages sur R(x) pour les calculer.

La Formation du syndrome est :

$$S(x) = S^{(0)} + S^{(1)}\,x + S^{(2)}\,x^2 + \cdots + S^{(n-k-1)}\,x^{n-k-1} \qquad \text{(II.4)}$$

La détection se fait suivant la valeur de S(x), on peut dire s'il y a des erreurs ou non, pour cela on a deux cas :

- S(x) = 0, R(x) est un multiple de g(x) donc le mot reçu est considéré comme le mot émis.
- S(x) < > 0, R(x) n'est pas un mot de code c'est à dire que V(x) = R(x) +E(x), on va corriger les erreurs.

S(x) ne dépend que de la configuration d'erreurs E introduite et pas de V.

En effet on a :

$$S(x) = \text{reste}\,(x^{n-k} * R(x) / g(x)) = \text{reste}\,(x^{n-k}\,E(x) / g(x)) \qquad \text{(II.5)}$$

II.2.1.2.2 Décodeur de MEGGIT :

Le décodeur de Meggit est un décodeur très puissant, il est utilisé pour le piégeage de l'erreur dans le mot reçu.

La distance minimum d'un CCHS est $d_{min} = 3$, et il corrige t erreurs (une erreur simple):

$$T = [(d_{min}-1)/2] = 1 \qquad \text{(II.6)}$$

Soit $E(x) = x^{n-1} = x^{2m-2}$, Le principe de correction de E(x).

A- Détection de E(x) (calcul de S(x))

$$S(x) = \text{reste}\,(x^{n-k} * R(x) / g(x)) \qquad \text{(II.7)}$$

$$= \text{reste}\,(x^{n-k} * E(x)/g(x)) \qquad \text{(II.8)}$$

$$= \text{reste } (x^{m} * x^{2m-2} / g(x)) \quad .$$

$$= \text{reste } (x^{m-1} * x^{2m-1} / g(x)) \quad .$$

On a $n= 2^{m-1} \rightarrow x^{n} = x\, 2^{m-1} \rightarrow 1 + x^{n} = 1+ x^{2m-1}$; donc $1+x^{2m-1}$ est divisible

Par g(x):

$$1+x^{2m-1}=a(x).g(x) \qquad \text{(II.9)}$$

Donc le reste $(x^{2m-1} / g(x)) = 1$.

Et puisque le degré de g(x) = m on a alors $S(x) = x^{m-1}$.

Donc si le symbole erroné est rn-1 alors le syndrome correspondant est $S(x) = x^{m-1}$.

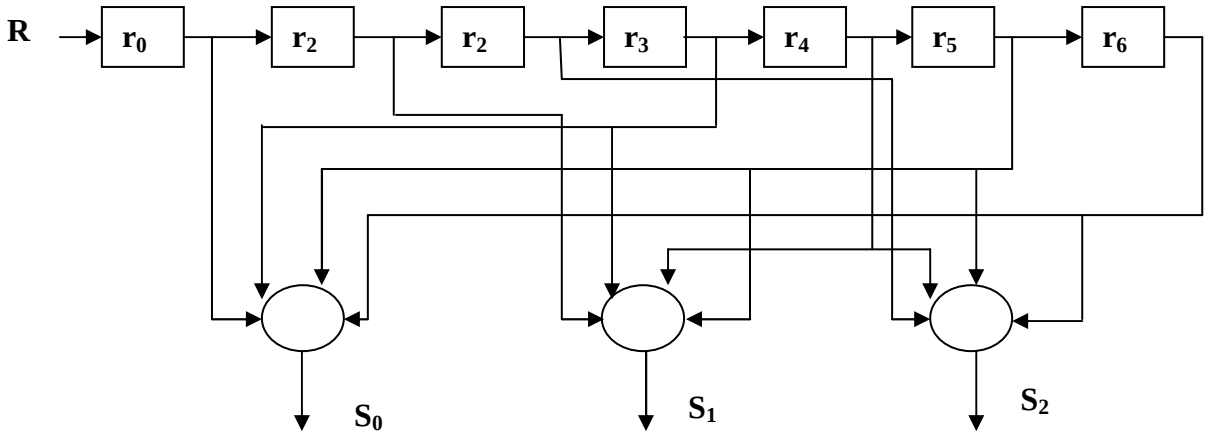


Figure II.2 : *Calcul du syndrome du code C (7, 4)*

B- Correction de E(x)

Si $S(x) = x^{m-1}$ alors la configuration d'erreur $E(x) = x^{2m-2}$ est reconnu le bit r_{n-1} peut être corrigé par :

$$r_{n-1} = r_{n-1} \oplus 1 \qquad \text{(II.10)}$$

Avec $$V(x) = R(x) + x^{n-1} \qquad \text{(II.11)}$$

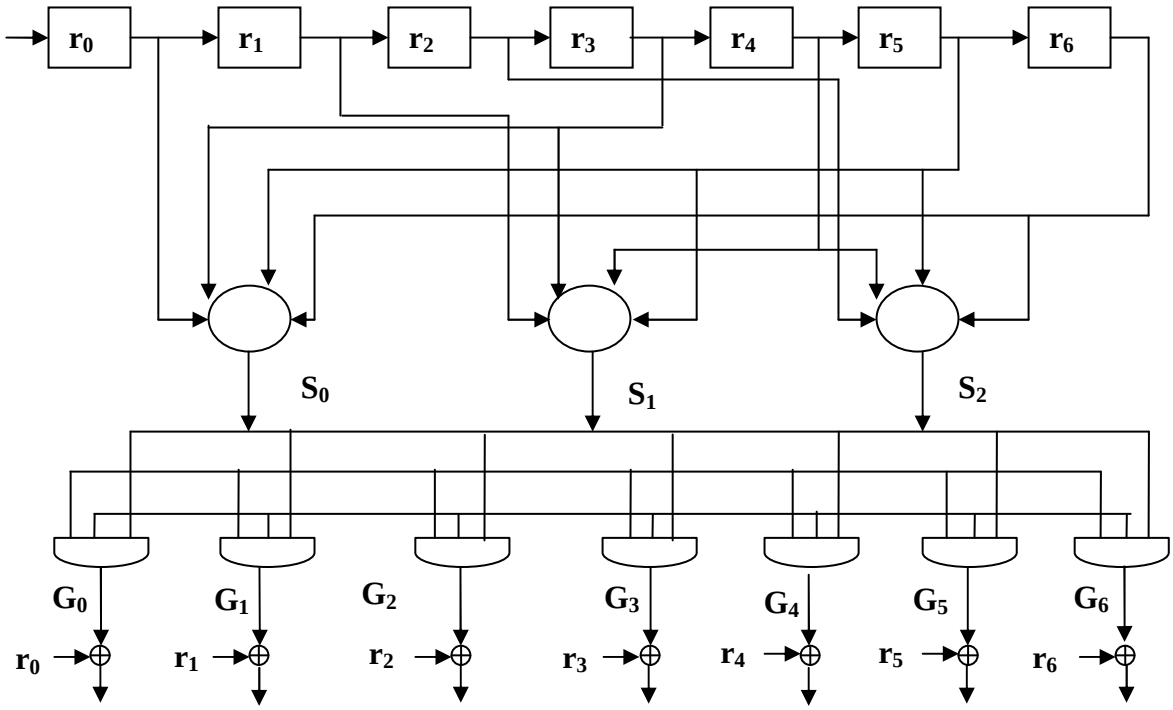


Figure II.3: *circuit de décodage du code C (7, 4)*

C- Correction des autres configurations d'erreur :

Supposons que $E(x) = x^p$ avec p = 0, …, n-1, le bit erroné dans R(x) est r_p, en effectuant des décalages cycliques successifs sur R(x), on peut déplacer r_p jusqu'à la position (n-1), pour cela (n-p-1) décalages cycliques sont nécessaires.

Soit $R^{(n-p-1)}(x)$ le mot obtenu en décalant cycliquement R(x) de (n-p-1) positions, le polynôme d'erreur correspondant à $R^{(n-p-1)}(x)$ est $E(x) = x^{n-1}$.

Si on connaît le syndrome $S^{(n-p-1)}(x)$ du mot $R^{(n-p-1)}(x)$ on peut reconnaître l'erreur et par conséquent corriger r_p, en effet si 'i' représente le nombre de décalages nécessaires pour déplacer l'erreur de la position **p** jusqu'à la position **(n-1)** (i.e. i= n-p-1).

Alors l'erreur est repérée par la position p = n-i-1 et le mot de code émis est :

$$V(x) = R(x)+x^p. \qquad \text{(II.12)}$$

II.2.2. C C H A :

Le code cyclique de Hamming présenté ci-dessus peut être amélioré pour corriger chaque erreur simple (isolée) et en même temps détecter chaque combinaison de deux erreurs, schématiquement la détection de l'erreur double est rendue possible grâce à un bit de contrôle (redondance) supplémentaire.

Le code CCHA est un code cyclique généré par le polynôme

$$g(x) = (1+x)\, P(x) \qquad \text{(II.13)}$$

P(x) est un polynôme primitif de degré m (m ≥3); avec les caractéristiques :

- Longueur du mot de code : $n = 2^m-1$.
- Nombre des symboles de contrôle : $n-k = m+1$.
- Nombre des symboles d'information : $k = 2^m-m-2$.

Le code CCHA se compose de tous les mots du code CCHS qui ont un poids pair [9].

II.2.2.1. Décodage :

Le principe de décodage du décodage du code CCHA peut être inspiré à partir du décodeur précèdent (décodeur du code CCHS).

La distance minimale du code CCHA est $d_{min} = 4$, il peut corriger $t= [(d_{min}-1) / 2]=1$ erreur, et détecter $d_{min} / 2 = 2$ erreurs.

A- Calcul du syndrome est détection d'erreurs

En divisant $x^m * R(x)$ par P(x) et R(x) par (1+x) On obtient :

$$X^m * R(x) = a_1(x).\, P(x) + Sp(x) \qquad \text{(II.14)}$$

$$R(x) = a_2(x).\, (1+x) + U \qquad \text{(II.15)}$$

Avec le degré Sp(x)≤m-1 et u □GF(2) On a

$$u= \text{reste}(R(x) / (1+x)) = WG\,(R(x)) \qquad \text{(II.16)}$$

Où WG désigne le poids du polynôme reçu R.

II.2.2.2. Tests et décisions :

- t1: si (u = 0) et (Sp(x) = 0): le décodeur considère qu'il n'y a pas d'erreurs dans le mot reçu.
- t2: si (u = 0) et (Sp(x) ≠0): le décodeur considère qu'il y a double erreurs dans le mot reçu, aucune action de correction n'est prise.
- t3: si (u = 1) et (Sp(x) = 0): Dans ce cas on a un nombre impaire d'erreurs, le polynôme erreur est divisible par P(x).
- t4: si (u ≠0) et Sp(x) ≠0): le décodeur considère qu'il y a une erreur simple dans le mot reçu, un processus similaire à celui du code CCHS est activé avec

$$g(x) = (1+x)\,P(x) \qquad \text{(II.17)}$$

II.2.3. Algorithmes de protection :(l'entrelacement)

En dehors du masquage d'erreurs utilisant la redondance interne de l'information, il existe des méthodes simples qui permettent de lutter contre les erreurs de transmission, sans considération de codage, parmi ces méthodes il y a l'entrelacement.

Avant la transmission, on mémorise une suite de mots de codes dans une mémoire ayant la forme d'une matrice de λ lignes et de *n* colonnes. On y fait entrer ligne par ligne, λ mots de codes de longueur n, puis on lit la mémoire colonne par colonne afin de transmettre tous les symboles.

La transmission consiste à transmettre les colonnes Colonne 1, Colonne 2, …, Colonne n. A la réception on fait entrer les symboles reçus dans une mémoire identique colonne par colonne et les mots de codes sont restitués en lisant la mémoire ligne par ligne.

Soient V(0), V(1), …, V(λ), λ mots de codes à entrelacer, l'entrelacement consiste à transmettre ces mots de codes dans l'ordre des symboles suivant :

Colonne1	**Colonne 2**	**…**	**Colonne n**
$V_{n-1}(1)$	$V_{n-2}(1)$	. . .	$V_0(1)$
.	.	. . .	.
.	.	. . .	.
.	.	. . .	.
$V_{n-1}(\lambda)$	$V_{n-2}(\lambda)$	. . .	$V_0(\lambda)$

Tableau II.1 : *Entrelacement*

Ce procédé très simple permet de fragmenter les coupures en disséminant les symboles en erreur dans plusieurs mots de code, ce procédé présent deux inconvénients :

- nécessité d'un volume de mémoire non négligeable pour des codes de longueur même modeste.

- possibilité de constitution de coupures à partir d'erreurs isolées, il est possible de proposer un entrelacement à profondeur variable, appelé scrambling. Mais la gestion des mots devient lourde.

L'effet de l'entrelacement est de répartir sur des mots différents les erreurs d'un même paquet d'erreurs, l'entrelacement n'augmente pas la capacité de correction d'un code, il ne fait qu'adapter la puissance de correction aux configurations d'erreurs les plus fréquentes.

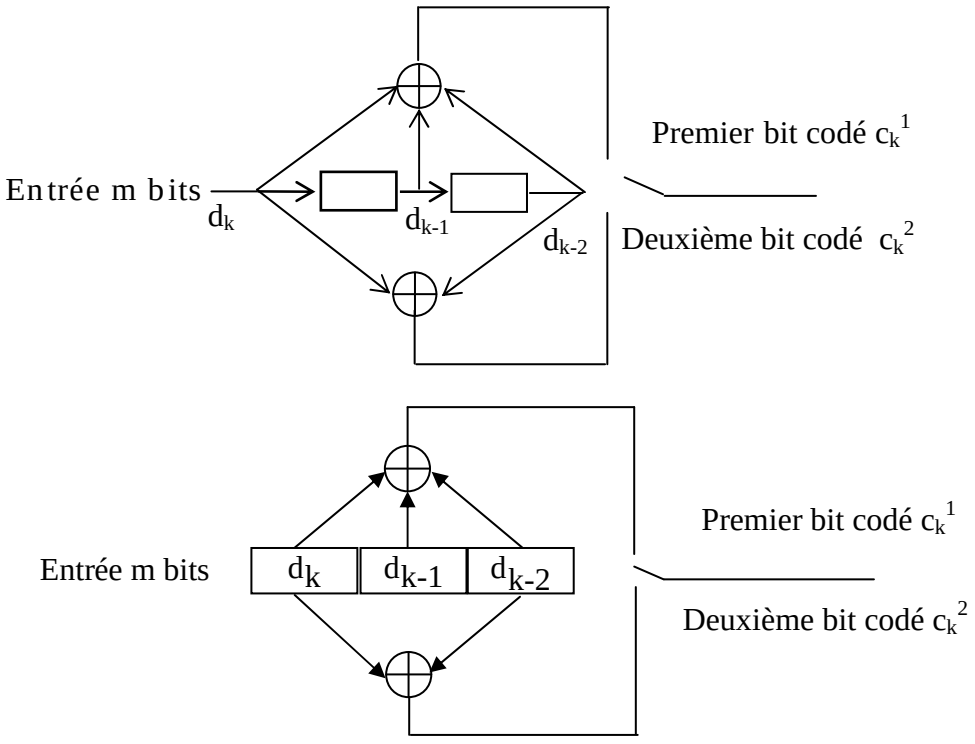
Etant donné un code cyclique C (n, k), il est possible de construire un code cyclique C (λ*n, λ*k) par entrelacement ceci est accompli en arrangeant m mots de codes du code original en m lignes d'un tableau (tableau 1 : Entrelacement) et en les transmettant colonne par colonne, le code obtenu est appelé code entrelacé.

II.3. Codes convolutionnels :

II.3.1. Définition :

Pour ces codes, chaque bloc de n éléments binaires en sortie dépend non seulement des k éléments binaires présents en entrée mais aussi des m blocs de k éléments binaires précédents. m+1 s'appelle la longueur de contrainte, souvent k=1. Le taux de codage est R= n/k. Le codage se fait avec des registres à décalage et des ou exclusif.

Exemple:



bit d'entrée	état du registre d_{k-1} d_{k-2}	contenu du registre	bits de sortie
0	00	000	**00**
1	00	100	**11**
0	01	001	**11**
1	01	101	**00**
0	10	010	**10**
1	10	110	**01**
0	11	011	**01**
1	11	111	**10**

Dans cet exemple à un é.b d'entrée correspond 2 é.b de sortie qui sont déterminés par les équations suivantes :

$$\begin{aligned} C_k^1 &= d_k + d_{k-1} + d_{k-2} \\ C_k^2 &= d_k + d_{k-2} \end{aligned} \qquad \text{(II.18)}$$

Soit en transformée en z :

$$\begin{aligned} c^1(Z) &= (1 + Z^{-1} + Z^{-2})\, \mathrm{d(Z)} \\ c^2(Z) &= (1 + Z^{-2})\mathrm{d(Z)} \end{aligned} \qquad \text{(II.19)}$$

On remplace souvent z^{-1} par D (!).

Et on appelle les polynômes générateurs :

$1 + D + D^{-1}$ $\quad 1 + D^{-2}$

ou en octal [111]=7 et [101]=5

De façon générale le $j^{iéme}$ bit de code a la forme, dans le cas où le bit d'entrée est unique:

$c^j(k) = \sum_{i=0}^{m-1} g_{ji}\, d(k-i)$ où k est l'instant d'observation [10].

Il y a autant de polynômes générateurs que de sorties du code : ces polynômes sont :

$$g_j(D) = \sum_{i=0}^{m-1} g_{ij} D^i \qquad \text{(II.20)}$$

II.3.2. Codage convolutif de rendement 1/n :

Considérons le code en bloc linéaire binaire spécifié par la matrice génératrice suivante:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_0} & \mathbf{g_1} & \mathbf{g_2} & \cdots & \mathbf{g_{L-1}} & & & & \\ & \mathbf{g_0} & \mathbf{g_1} & \mathbf{g_2} & \cdots & \mathbf{g_{L-1}} & & & \\ & & \mathbf{g_0} & \mathbf{g_1} & \mathbf{g_2} & \cdots & \mathbf{g_{L-1}} & & \\ & & & \ddots & & & & \ddots & \\ & & & & \ddots & & & & \ddots \\ & & & & & \mathbf{g_0} & \mathbf{g_1} & \mathbf{g_2} & \cdots & \mathbf{g_{L-1}} \end{bmatrix} \qquad \text{(II.21)}$$

Chaque élément *l* $\mathbf{g}$ $0 \le l \le L-1$, de la matrice est un vecteur ligne binaire à *n* composantes de la forme :

$$\mathbf{g_1} = |\mathbf{g_{l1}}, \mathbf{g_{l2}}, \cdots, \mathbf{g_{ln}}| \qquad \text{(II.22)}$$

Les zones laissées vides en bas à gauche et en haut à droite correspondent à des éléments tous nuls. Chaque ligne s'obtient en décalant la précédente de *n* colonnes vers la droite. Cette matrice, de taille $B \times (nB+L-1)$ décrit donc un code linéaire en bloc dont les mots de code pourraient être engendrés par le dispositif de la Figure 1.

Généralement, les mots de codes sont très longs, c'est à dire $B >> L$ et nous pouvons même considérer cette matrice comme infinie. Ce code linéaire ne possède donc plus à proprement parler de dimension ni de longueur.

Le rendement ne peut alors être défini comme pour les codes en blocs. Cependant en négligeant *la fermeture* du code (le vidage des registres), nous remarquons qu'il existe un rapport $1/n$ entre le nombre de bits d'information et le nombre de bits du mot de code. Le rendement de ce code est donc : $r = 1/n$

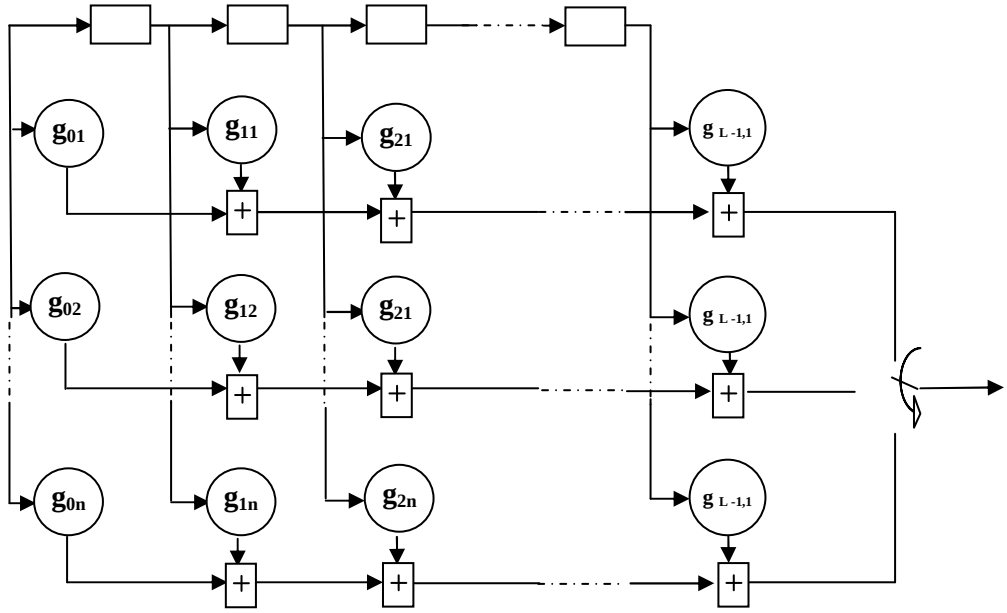


Figure II.4 : *Code convolutif de rendement r=1/n*

La quantité *L* est appelée *longueur de contrainte* du code convolutif. Le terme « convolutif » s'applique à cette classe de codes parce que la suite de bits codés **s** peut s'exprimer comme le résultat de la convolution de la suite de bits d'information **e** par les coefficients *g*. En effet, puisque le code est linéaire, nous avons : **s = e**. G. En observant la forme particulière de G, les *n* bits en sortie du codeur à l'instant *t*, correspondant à une entrée, s'écrivent :

$$\boldsymbol{s_t = \sum_{u=\max(1,t-L+1)}^{t} e_k g_{u-k}} \qquad \text{(II.23)}$$

La complexité du codeur est indépendante de la longueur de la séquence de bits d'information émise et ne dépend que de la longueur de contrainte du code [11].

II.3.3. Codage convolutif de rendement k/n :

Pour obtenir un rendement égal à un nombre rationnel quelconque inférieur à 1, il nous faut généraliser la forme (1) de la matrice génératrice et le schéma à la Figure 1 du codeur. Nous supposons maintenant que *k* bits d'information sont introduits en parallèle à un instant donné dans le codeur ou qu'un décalage temporel du vecteur **e** des bits d'information se fait par bloc de *k* bits. Cela revient à remplacer la ligne [**g**0 **g**1 … **g**L-1] de taille 1×*L.n* par une sous matrice binaire de taille *k*×*L.n*. dont les éléments constituant **g**l sont de taille *k*×*n*. La construction générale par décalage de la matrice reste la même :

$$\mathbf{G} = \begin{bmatrix} \overset{\mathbf{n}}{\leftrightarrow} & & & & & & \\ \mathbf{K} \updownarrow \mathbf{g_0} & \mathbf{g_1} & \cdots & \mathbf{g_{L-1}} & & & \\ & \mathbf{g_0} & \mathbf{g_1} & \cdots & \mathbf{g_{L-1}} & & \\ & & \ddots & & \ddots & & \\ & & & \mathbf{g_0} & \mathbf{g_1} & \cdots & \mathbf{g_{L-1}} \end{bmatrix} \quad \text{(II.24)}$$

Exemple 1 : Soit le code convolutif de rendement 2/3 et de longueur de contrainte *L*=2 défini par les coefficients

Suivants :

$$g_0 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$
$$g_1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{(II.25)}$$

Son codeur possède la structure suivante :

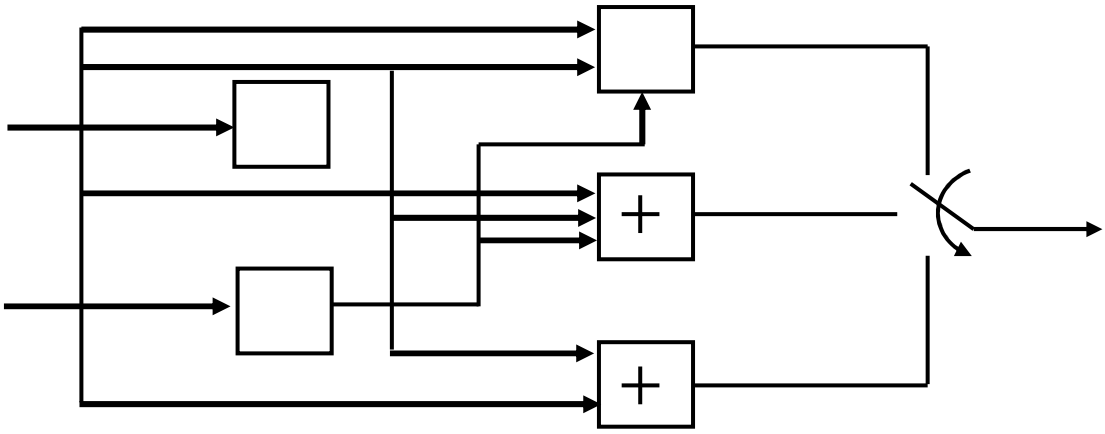


Figure II.5 : *Code convolutif de rendement 2/3 et de longueur de contrainte* L=2

La sortie d'un codeur convolutif à l'instant *t* dépend de ses entrées à l'instant *t* et de l'état des registres du codeur à cet instant. En posant ν=*L*-1, tout code convolutif de rendement *k*/*n* et de longueur de contrainte *L* possède *k*ν registres, qui peuvent donc prendre 2*k*νétats différents. Comme l'entrée est constituée de *k* bits différents, il existe 2 *k* transitions possibles à partir d'un état du codeur à un instant donné.

$$e_i(x) = \sum_{l=0}^{+\infty} e_{il} x^l, \quad i = 1, \ldots, n \quad \text{(II.26)}$$

II.3.4. Représentation polynômiale :

Soit un code convolutif de rendement *k*/*n* et de longueur de contrainte *L*. Son entrée peut être vue comme la donnée de *k* séquences infinies de bits, que nous représentons par des séries en *x* : $e_i(x) = \sum_{l=0}^{+\infty} e_{il}\, x^l$, $i = 1, \dots, k$

Où l'inconue *x* symbolise un retard d'une période. Le coefficient *ei*□ est le bit présent à la *i*-ème entrée du codeur à l'instant ℓ. De la même manière, les *n* sorties du codeur peuvent être indexées par *j* et s'écrivent :

$$\boldsymbol{s_j(x) = \sum_{l=0}^{+\infty} s_{jl} x^l, \quad j = 1, \dots, n} \qquad \text{(II.27)}$$

La relation introduite par le code convolutif entre les entrées et les sorties peut s'écrire de façon élégante avec ses notations, en introduisant les polynômes générateurs. Le polynôme générateur qui lie la $i^{\text{ème}}$ entrée à la $j^{\text{ième}}$ sortie est noté g_{ij} (*x*), et est de degré inférieur ou égal à *L*-1. Le coefficient de degré *r* de ce polynôme vaut 1 si une connexion existe entre la sortie *j* et le bit présent dans le $r^{\text{ième}}$ registre correspondant à l'entrée *i*.

La notation octale est couramment utilisée pour représenter ces polynômes.

Ainsi, $g(x) = 1 + x + x3$ sera représenté par le nombre 15.

Les sorties se calculent en fonction des entrées à l'aide des polynômes générateurs :

$$\boldsymbol{s_j(x) = \sum_{i=1}^{k} g_{ij}(x) e_i(x)} \quad \textbf{j=1,…,n} \qquad \text{(II.28)}$$

Puisque la convolution temporelle correspond à la multiplication polynômiale.

II.3.5. Codes convolutifs récursifs systématiques :

Un code convolutif est dit *récursif* lorsque ses polynômes générateurs sont remplacés par les quotients de deux polynômes. Une partie de la sortie est alors réintroduite dans le registre à décalage selon les connexions définies par les polynômes situés aux dénominateurs.

Un code convolutif est dit *systématique* lorsque une partie de ses sorties est exactement égale à ses entrées. Cela revient à dire que

(g_{11}(x)=1, g_{21}(x)=0, …, g_{k1}(x)=0), (g_{12}(x)=0, g_{22}(x)=1, … , g_{k2}(x)=0), …, et (g_{1k}(x)=0, g_{2k}(x)=0,…, g_{kk}(x)=1).

Un cas particulièrement intéressant est celui de la transformation d'un code convolutif non récursif non systématique (NRNSC) de rendement *k*/ (*k*+1) en un code récursif

systématique (RSC). Prenons l'exemple d'un code NRNSC de rendement ½. Il est défini par la donnée de ses deux polynômes générateurs $g11(x)= g1(x)$ et

$g12(x)= g2(x)$, qui forment la matrice :

$$G = [g_1(x) g_2(x)] \tag{II.29}$$

La transformation en code RSC correspond à considérer le code défini par la matrice génératrice suivante :

$$\boldsymbol{G = \left[1 \quad \frac{g_2(x)}{g_1(x)}\right]} \tag{II.30}$$

Exemple 2 : Soit le code convolutif NRNSC de rendement ½, de longueur de contrainte L=3 et de polynômes générateurs (7,5). Ses polynômes s'écrivent : $g1(x)=1+x+x2$ et $g2(x)=1+x2$. Son codeur possède la structure suivante :

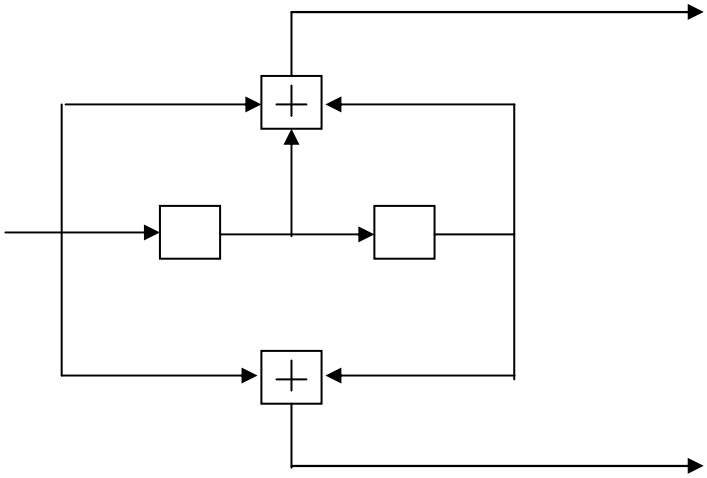


Figure II.6 : *Code convolutif NRNSC (7,5)*

Ses deux sorties $s1(x)$ et $s2(x)$ s'expriment en fonction de l'entrée $e(x)$ de la manière suivante :

$$\boldsymbol{s_1(x) = (1 + x + x^2)e(x) \; et \; s_1(x) = (1 + x^2)e(x)} \tag{II.31}$$

Sa transformation en code RSC correspond aux relations suivantes :

$$s_1(x) = e(x) \; et \; s_2(x) = \frac{1+x^2}{(1+x+x^2)} e(x) \tag{II.32}$$

La deuxième relation s'écrit encore :

$$\boldsymbol{(1 + x + x^2)} s_2(x) = \boldsymbol{(1 + x^2)e(x)} \tag{II.33}$$

soit, en repassant en notation temporelle, à l'instant *k* :

$$s_2^k = e^k + e^{k-2} + s_2^{k-1} + s_2^{k-2} \quad \text{(II.34)}$$

La structure du code RSC est la suivante :

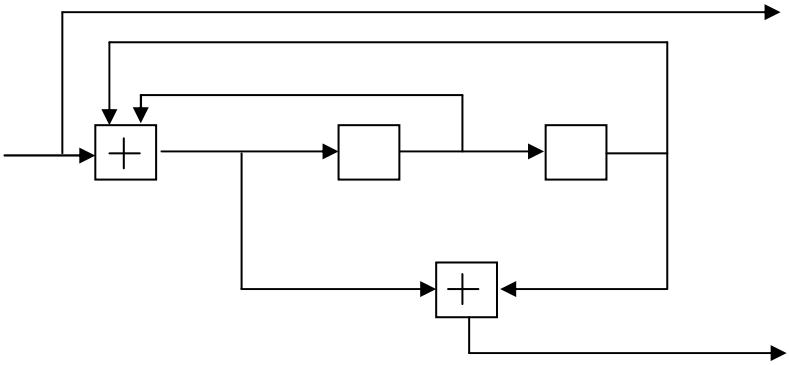


Figure II.7 : *Code convolutif RSC (1,5/7)*

II.4. Représentation en treillis :

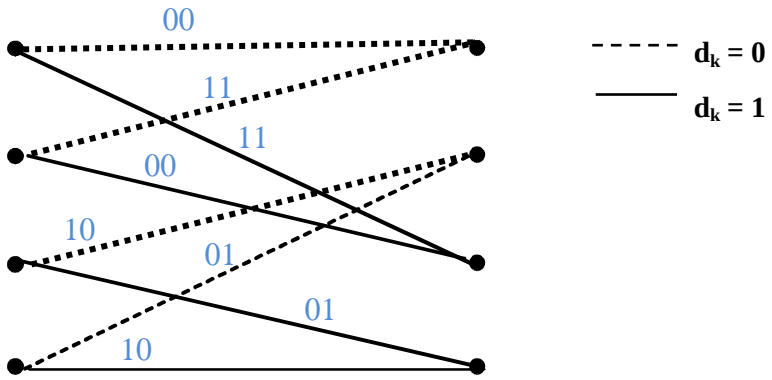
Pour faciliter l'algorithme de décodage, la représentation la plus courante du codage est la représentation en treillis.

L'état du codeur à l'instant k est représenté par l'état $\{d_{k-1}, d_{k-2},\ldots..d_{k-m-1}\}$. A chaque arrivée d'un élément binaire dk, une sortie (un mot de code) est générée, puis juste après le codeur passe dans l'état suivant qui est $\{d_k, d_{k-1},\ldots\ldots d_{k-m}\}$.

Le treillis est formé de noeuds reliés par des branches : les noeuds représentent les différents états du codeur possibles : il y en a 2^{m-1} s'il y a une entrée$2^{(m-1)k}$ s'il y a k entrées, les branches représentent les différentes transitions possibles d'un noeud à un autre (ou d'un état du codeur au suivant) lors de l'arrivée d'un bit d'entrée.

Voici le treillis du code précédent : les états sont 00,01, 10,11.

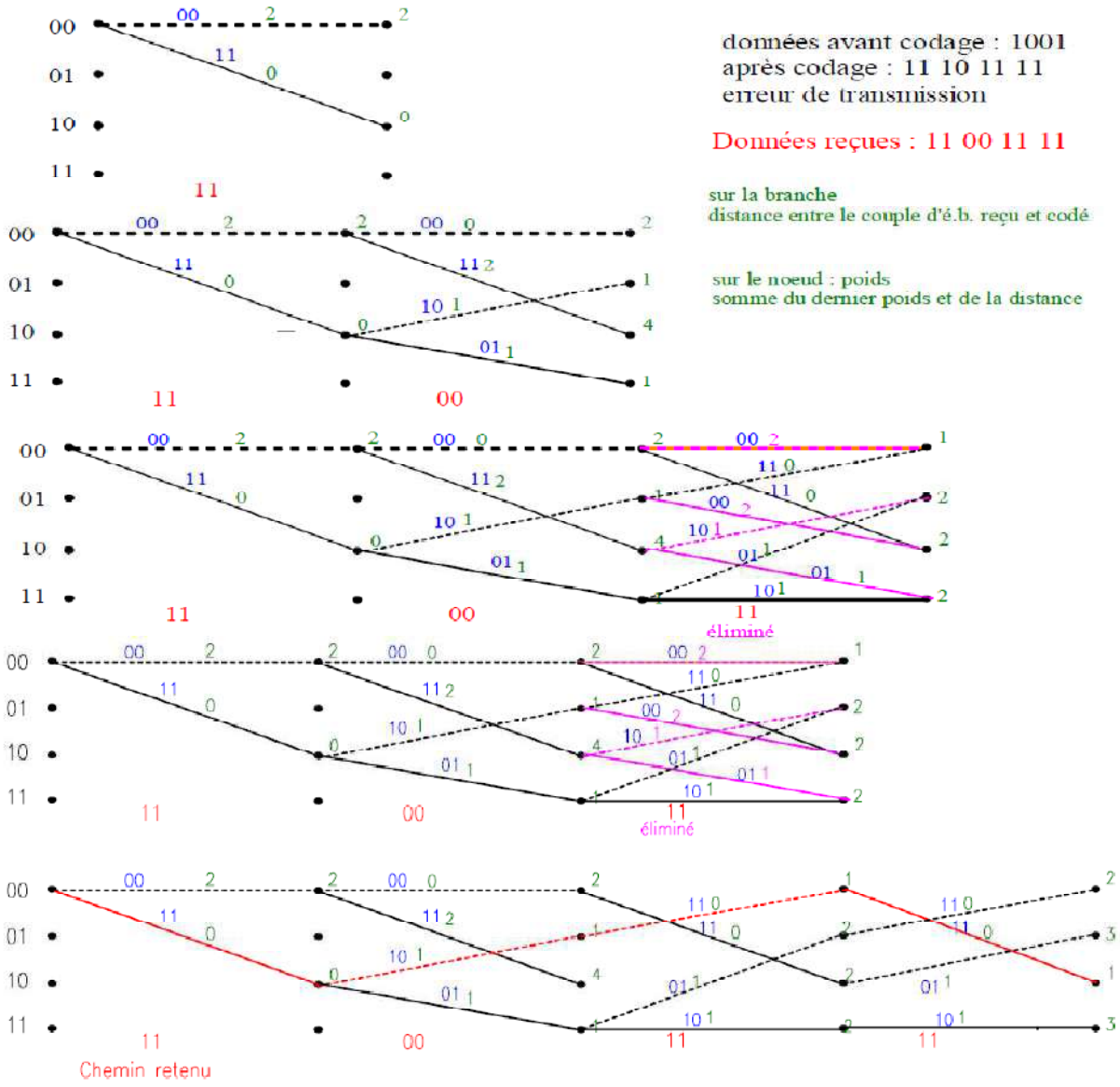
Partant, par exemple de l'état 00, l'arrivée d'un 0 mène le codeur à l'état 00 (transition en pointillé pour l'arrivée d'un 0) et l'arrivée d'un 1 mène le codeur à l'état 10 (transition en trait plein pour l'arrivée d'un 1). A chaque branche on peut associer le mot codé soit les 2 bits de code ici [10].



II.4.1. Décodage : algorithme de Viterbi :

Le décodage le plus courant est basé sur l'algorithme de Viterbi.

Il consiste à rechercher dans l'arbre le chemin qui correspond à la séquence la plus probable, c'est-à-dire celle qui est à la distance minimale de la séquence reçue ou encore la séquence la plus probable. Dans l'exemple qui suit (basé sur le codeur déjà décrit) on suppose que le démodulateur fournit des 0 et des 1 (décision dure). A chaque fois qu'un groupe de 2 é.b. arrive, on examine toutes les branches possibles du treillis, on calcule la distance entre les é.b. affectés aux branches et les é.b. reçus, on ne garde que les branches donnant lieu à une distance minimale (ces branches forment le chemin survivant), et on affecte l'extrémité du chemin survivant d'une métrique égale à la somme de la métrique précédente et de la distance de la branche retenue.



II.4.2. Présentation de l'algorithme de Viterbi :

L'algorithme peut être mis en oeuvre de façon commode [12] à partir d'un diagramme en treillis en appliquant les règles suivantes :

A- Etape 1 : Partir du premier étage du diagramme où deux branches arrivent à chaque état, calculer pour chacun des états la distance d (y~T,y^T) de chacun des deux chemins qui arrivent à cet état, pour chaque état, conserver le chemin dont la distance est la plus faible (le survivant), et éliminer l'autre.

B- Etape 2 : Répéter l'opération pour chaque étage du treillis.

C- Etape 3 : Lorsque l'origine de tous les survivants passe par un chemin unique, ce dernier représente une partie du mot décodé.

II.4.3. *Distance libre :*

C'est la distance minimale entre deux chemins divergents partant du même point et y revenant le « plus vite possible ». Comme le code est linéaire on peut ne considérer que des chemins partant par exemple de l'état 00. Et la distance libre est le poids de la séquence partant de 00 et y revenant le plus vite possible.

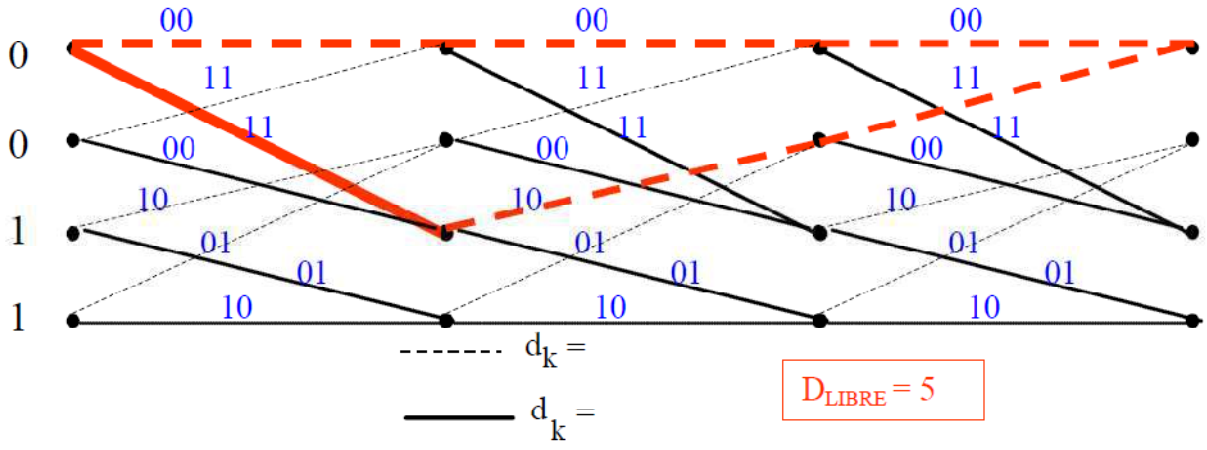
Cette distance représente la distance entre deux chemins les plus proches sans erreur.

Plus elle est grande, plus le code sera performant.

II.4.4. *Diagramme d'état et Fonction de transfert*

On déduit de l'exemple du treillis précédent le diagramme d'état dans lequel le temps n'intervient pas et qui représente les différentes transitions possibles entre états.

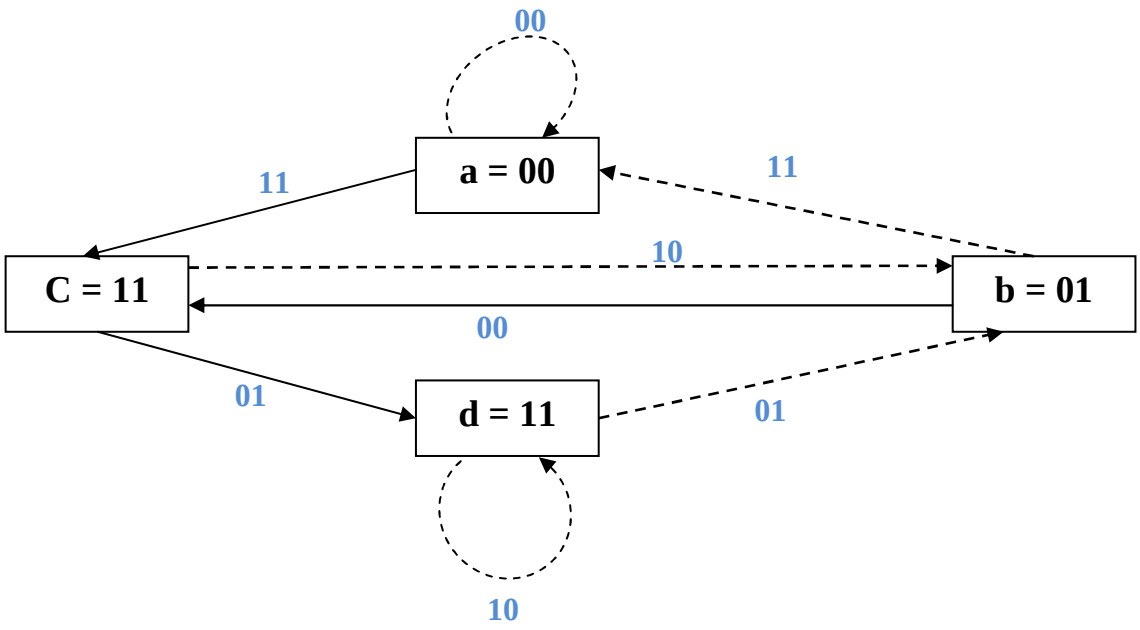
On peut



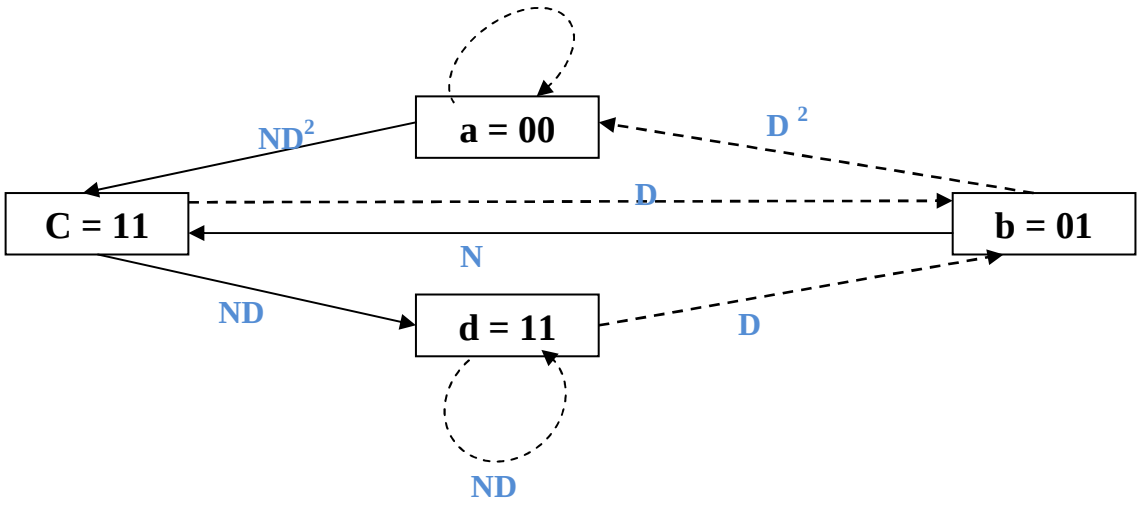
Cette distance représente la distance entre deux chemins les plus proches sans erreur.

Plus elle est grande, plus le code sera performant.

On déduit de l'exemple du treillis précédent le diagramme d'état dans lequel le temps n'intervient pas et qui représente les différentes transitions possibles entre états.

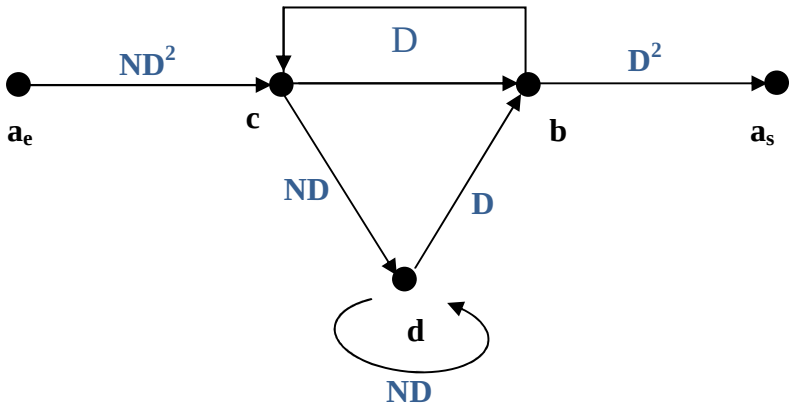


On peut à partir de ce diagramme calculer une fonction de transfert qui dépend des poids des séquences codées, des poids des séquences d'information (entrée). Pour cela, on introduit, à chaque transition, les variables D affecté de l'exposant égal au poids du mot de code et N affecté de l'exposant égal au poids du mot d'information [10].



On va ouvrir ce diagramme pour avoir un nœud d'entrée et un nœud de sortie : ces deux nœuds seront les mêmes de façon à observer les chemins qui partent d'un nœud et qui y retournent.

Par exemple on prend a_e et a_s le nœud a d'entrée et de sortie :



Les équations de passage sont :

$c = ND^2 a_e + Nb$

$b = D_c + D_d$

$d = ND_c + ND_d$

$a_s = D_b$

Après résolution de système :

$$T(D,N) = \frac{a_s}{a_e} = \frac{D^5 N}{1-2DN} \quad \text{(II.35)}$$

Soit en développant le numérateur de la fonction de transfert en série : ∞

$$T(D,N) = \sum_{k=0}^{\infty} 2^k D^{k+5} N^{k+1} \quad \text{(II.36)}$$

On en déduit que :

- il existe 2k chemins partant d'un noeud et aboutissant au même de poids k+5 dont la séquence d'entrée a le poids k+1
- en particulier si k=0, il existe 1 chemin de poids le plus faible 5 :c'est la distance libre.

En général :

$$T(D,N) = \sum_{k=dlib}^{n} a(h) D^h N^{w(h)} \quad \text{(II.37)}$$

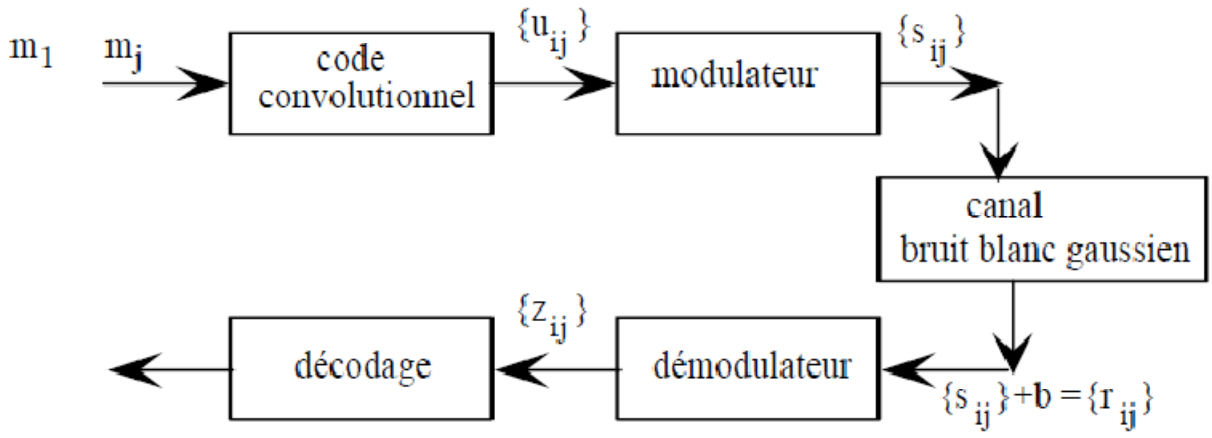
II.4.5. Décodage à décision douce en entrée :

Un des intérêts de l'algorithme de Viterbi est de pouvoir faire le décodage sans être obligé de prendre une décision en sortie du démodulateur : décision douce.

La chaîne de transmission peut être modélisée comme suit :

j : indice de la branche

Branche du treillis i : indice de l'é.b de la branche



Le canal introduit du bruit : donc en sortie du canal on a des signaux non binaires avec des valeurs analogiques (continues) [10].

Le décodage présenté précédemment suppose que le démodulateur a pris une décision 0 ou 1 : décision dure.

On peut effectuer le décodage de Viterbi en supposant que le démodulateur n'a pas pris de décision ou que la valeur du signal est codée sur plusieurs bits : décision souple.

L'algorithme garde le même principe mais les distances utilisées sont les distances euclidiennes :

Si (z^1_k, z^2_k) est le couple d'échantillons en sortie du démodulateur à l'entrée du décodeur à t=k, et (a^1_k, a^2_k) le couple binaire associée à une branche du treillis

$$M_k = \sum_{i=1}^{2} \left(z_k^i - a_k^i\right)^2 \qquad \text{(II.38)}$$

Pour un couple (z^1_k, z^2_k) reçus, on cherche le couple (a^1_k, a^2_k) qui en est à la distance la plus faible.

Par exemple pour une modulation de phase à 2 états (a peut être -1 si 0 est envoyé et +1 si 1 est envoyé), il suffit de comparer les métriques

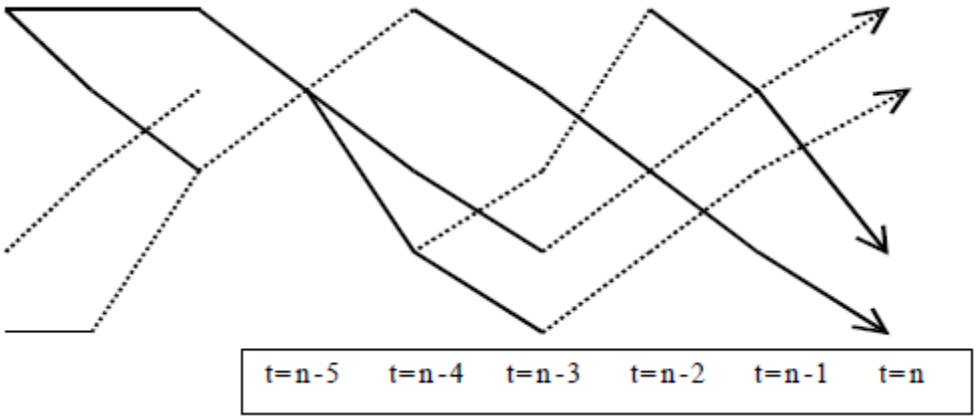
$$\boldsymbol{M_k = -2 \sum_{i=1}^{2} \left(z_k^i a_k^i\right)} \qquad \text{(II.39)}$$

Le décodage corrige alors plus d'erreurs

II.4.6. *Limitation du décodage à une fenêtre :*

En pratique, on n'attend pas que toute la séquence binaire émise soit reçue pour commencer le décodage (trop de mémoire)On s'aperçoit qu'au bout d'un certain temps t = n, les chemins survivants convergent vers un même chemin en amont en t = n-x.

On garde tous les chemins survivants de n-x à n et à t=n, on décide de décoder l'échantillon de t=n-x.



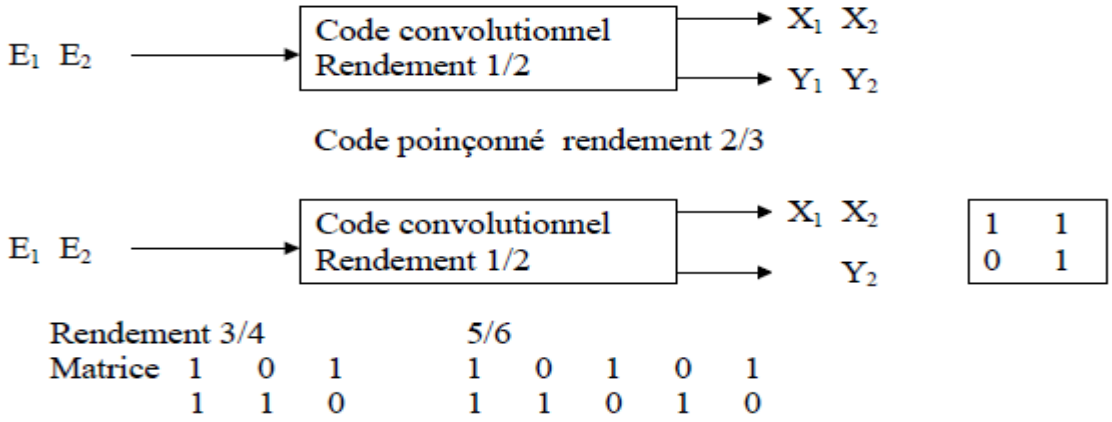
Pour un rendement ½ n~ 5 à 6 fois la longueur de contrainte.

II.4.7. Codes convolutionnels poinçonnés :

Plus le rendement est faible et plus la longueur de contrainte est grande, plus le code est performant.

Mais, plus le rendement est faible, plus la largeur de bande nécessaire à la transmission doit être importante [10].

Les rendements des codes décrits sont de la forme 1/n, donc très faibles, d'ailleurs, on voit rarement des codes convolutionnels de rendement inférieur à ½ Pour fabriquer des codes convolutionnels de rendement supérieur on utilise des codes Poinçonnés: ils sont obtenus à partir des codes normaux auxquels on supprime un bit de sortie périodiquement : ex :



Un code rendement 1/2 dont on supprime un bit de sortie sur 4 devient un code de rendement 2/3.

De même pour construire un code de rendement ¾, on ne transmet que X1, Y1, Y2, X3 etc…

II.5. Comparaison des codes cycliques et des codes convolutionnels :

Rappel : Rendement = n^{bre} bits d'information / n^{bre} bits de code

- Le rendement des codes en blocs est élevé (> 0,9), celui des codes convolutionnels est assez faible malgré la technique de poinçonnage.
- Les codes cycliques particulièrement les codes de Reed-Solomon corrigent des erreurs arrivant par paquets, ou par rafales (burst) Le code de Reed-Solomon (204,188,8) utilisé dans les normes DVB peut corriger un paquet 8 octets soit 64 bits par trame.
- Si les rafales d'erreurs ont une taille supérieure à la possibilité du code, on effectue un entrelacement des bits avant le décodage : le but est de mélanger les bits pour « casser» les paquets d'erreurs.

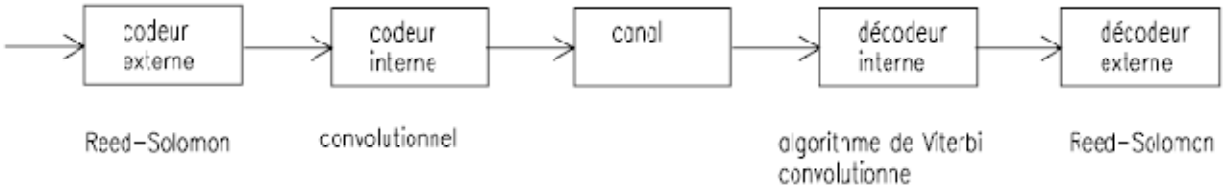
Les rafales d'erreurs peuvent être provoquées par le canal, en particulier les bruits impulsifs (souvent dû aux interrupteurs) dans les câbles, ou par des multi-trajets dont la somme entraîne un signal presque nul.

- Les avantages des codes convolutionnels sont leurs bonnes performances grâce au décodage à décision souple et leur faculté des corriger des erreurs isolées.

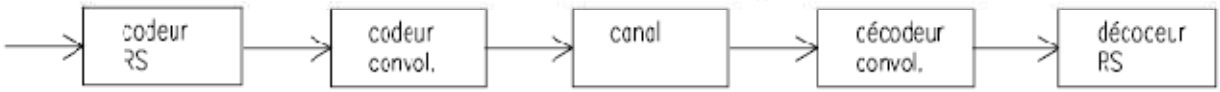
Avec des canaux très bruités (Hertziens) pour profiter des avantages des deux types de codes, on peut les concaténer.

II.6. Concaténation

Le décodage du code convolutionnel corrige les erreurs indépendantes mais a tendance à laisser des paquets d'erreurs (comme le décodage dépend des données actuelles mais aussi des précédentes, une erreur à t a de l'influence sur le déc t+1...) qu'un décodage de Reed-Solomon RS pourra alors corriger. Le schéma du codage et du décodage est alors le suivant :



EX TV numérique satellite et terrestre
décodeur externe : RS décodeur interne convolutif L=7, R=1/2



On ajoutera un entrelaceur entre les deux codes pour diminuer la taille des paquets d'erreurs à l'entrée de décodeur RS

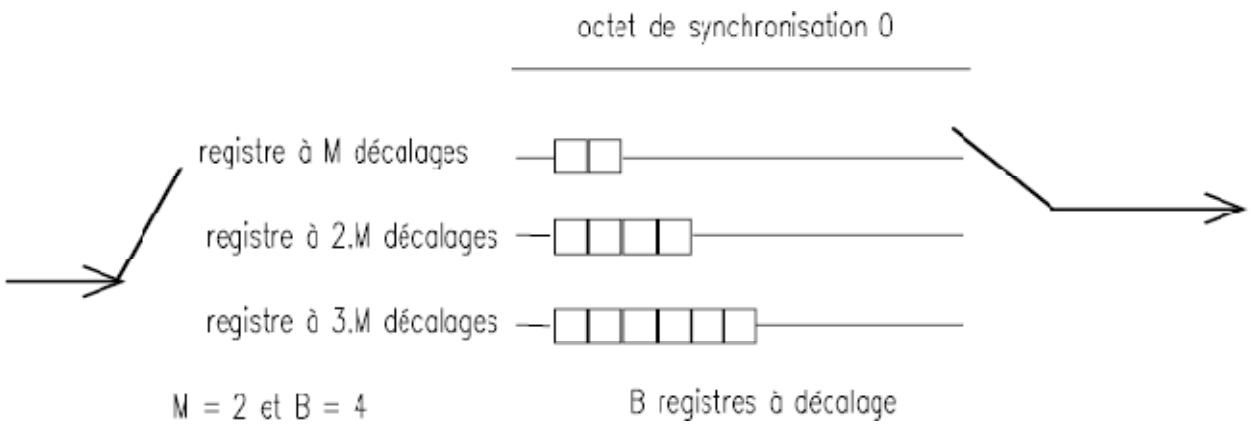
II.7. Entrelacement

Pour diminuer la taille des rafales pour les erreurs arrivant en rafales, on procède à un entrelacement temporel : ainsi la taille des paquets d'erreurs ne dépassera pas les possibilités de correction du code RS.

Le principe est de changer l'ordre des symboles à l'émission les remettre en ordre à laréception. (opérations duales très similaires)Ce module est donc inséré entre les deux décodages (et codages) Principe : opération sur les octets.Un premier type d'entrelacement est l'entrelacement par blocs :

$$\text{ENTREES} \begin{bmatrix} 09 \dots \dots \dots \dots 8*4 \\ 110 \dots \dots \dots \dots 33 \\ 211 \dots \dots \dots \dots 34 \\ \dots \dots \dots \dots \dots \dots \dots \\ 8 \dots \dots \dots \dots \dots 39 \end{bmatrix} \text{SORTIES}$$

Un deuxième type est l'entrelacement convolutionnel. Voici l'exemple de l'entrelacement défini dans les normes DVB télévision satellite et terrestre. B registres de longueur variant entre M et (B-1) M



Normes DVB : M = 17 et B = 12

Un octet est au plus décalé de 11×17 (soit la taille de la trame excepté l'octet de synchronisation) [10].

II.8. Conclusion :

Dans ce chapitre, nous avons présenté les techniques de codages convolutifs et cycliques en détaille ainsi que leurs théories de décodage. Ces techniques on va les implémenter dans ce qui suit en utilisant le simulink sous MATLAB et on discutera les résultats en basant sur l'étude de l'erreur de détection BER en absence et présence de bruit à travers un canal à bruit blanc gaussien.

CHAPITRE (III)

Implémentation et discussion des résultats

Chapitre III :

Implémentation et discussion des résultats

III.1 Introduction :

Dans ce chapitre on va effectuer l'implémentation des techniques du codage de canal.

Dans notre travail, on présente deux approches d'application de l'algorithme du codage convolutif ainsi que son décodage en basant sur l'algorithme de Viterbi et on discute les résultats obtenue telque le BER en présence et en absence du bruit gaussien du canal de transmission. On distingue deux cas,

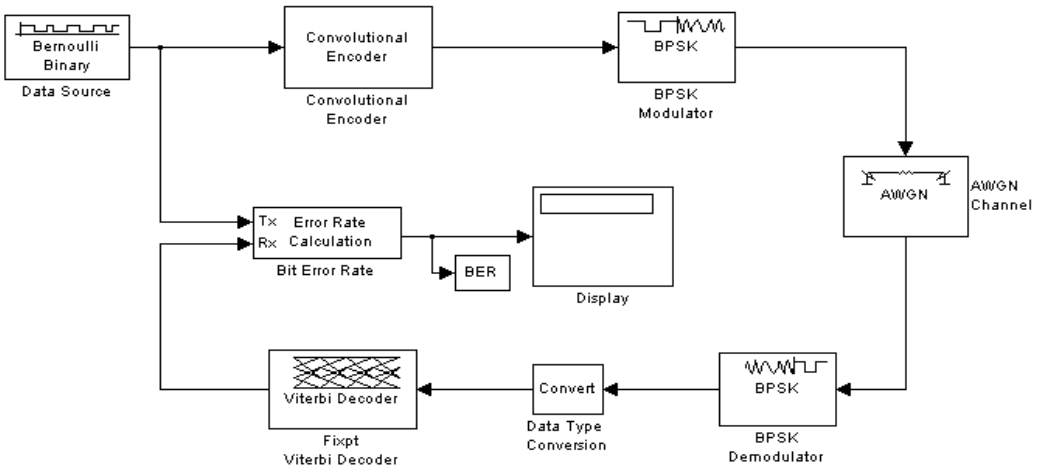
- Le cas du codage convolutif en décision dure (hard decision),
- Le cas de décision logiciel (soft decision).

Dans les deux cas on utilise le simulink sous matlab pour l'implémentation

III.2. Codage convolutif et décodage de Viterbi :

III.2.1. Cas de décision dure (hard-decision) :

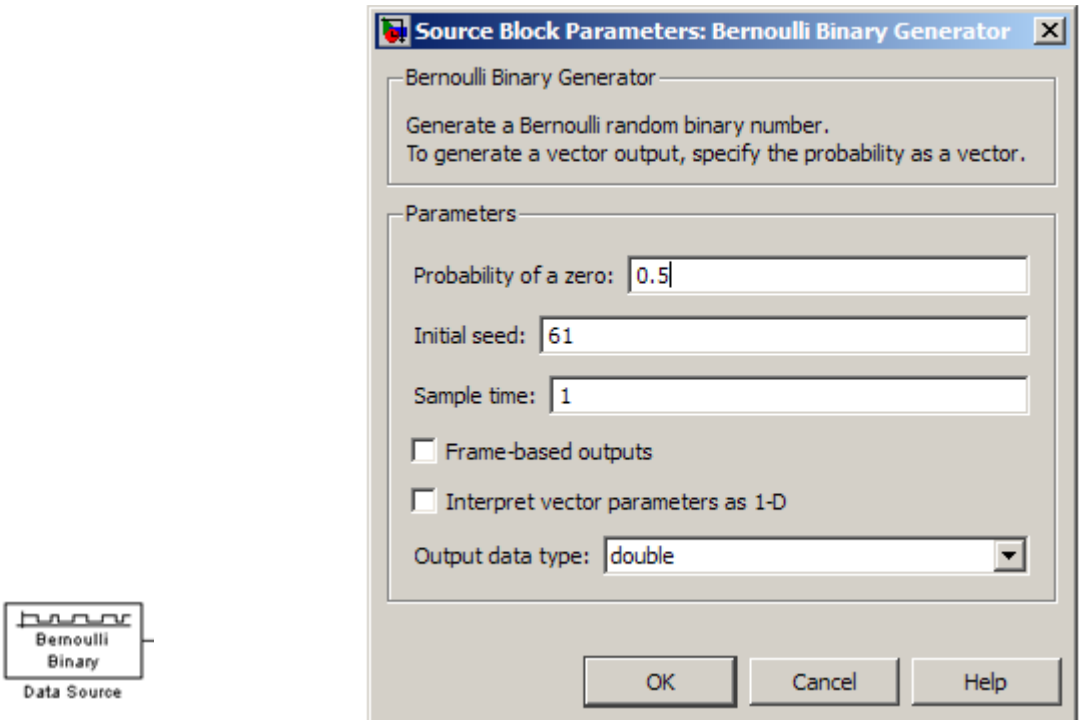
Pour cette partie, on utilise la chaine de transmission suivante, où la transmission à travers un canal AWGN qui conduit à la perturbation de l'information modulée transmise.



Figure(III.1) : chaine de transmission de hard décision

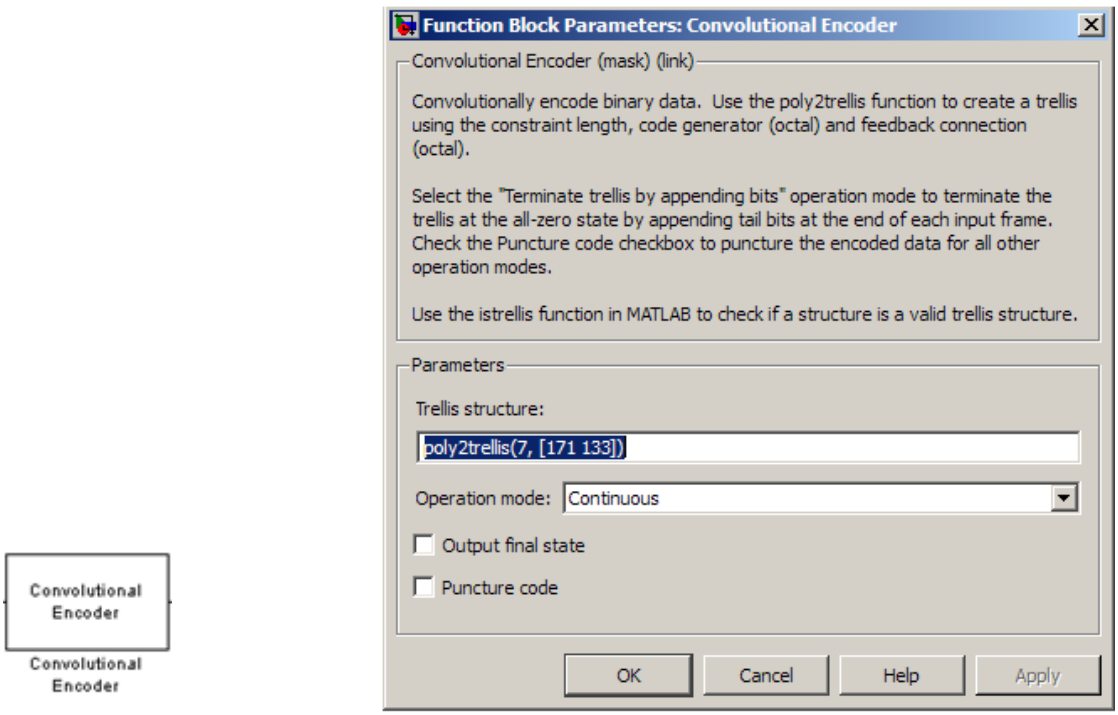
Les éléments constituants cette chaine de transmission sont les suivants :

a- Générateur des séquences de bits binaires aléatoires : c'est une source d'information suivant la configuration de la boite de dialogue.



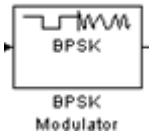
Figure(III.2) : Générateur binaire de Bernoulli ainsi que sa boite de dialogue.

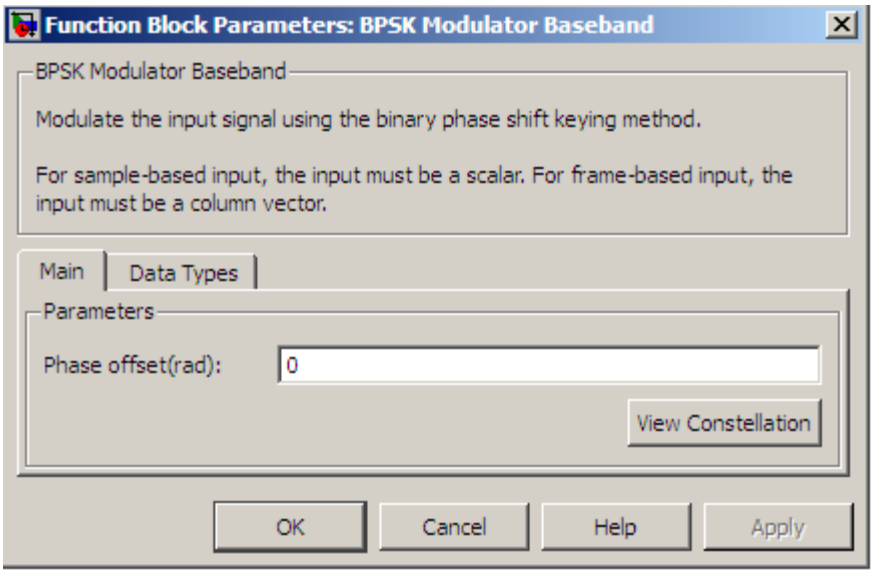
b- Codeur convolutif :



Figure(III.3) : Bloc du codeur convolutif ainsi que sa boite de dialogue.

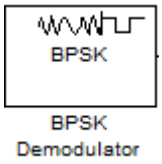
c- Blocs de Modulation Bpsk :

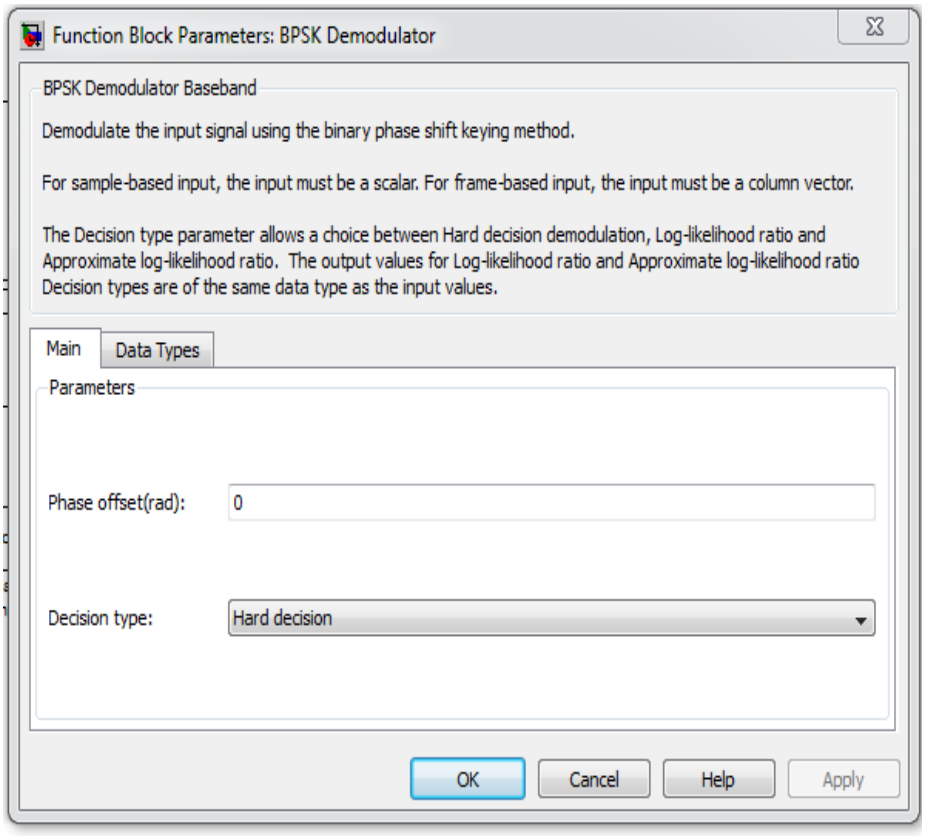




Figure(III.4) Bloc de modulation BPSK ainsi que sa boite de dialogue.

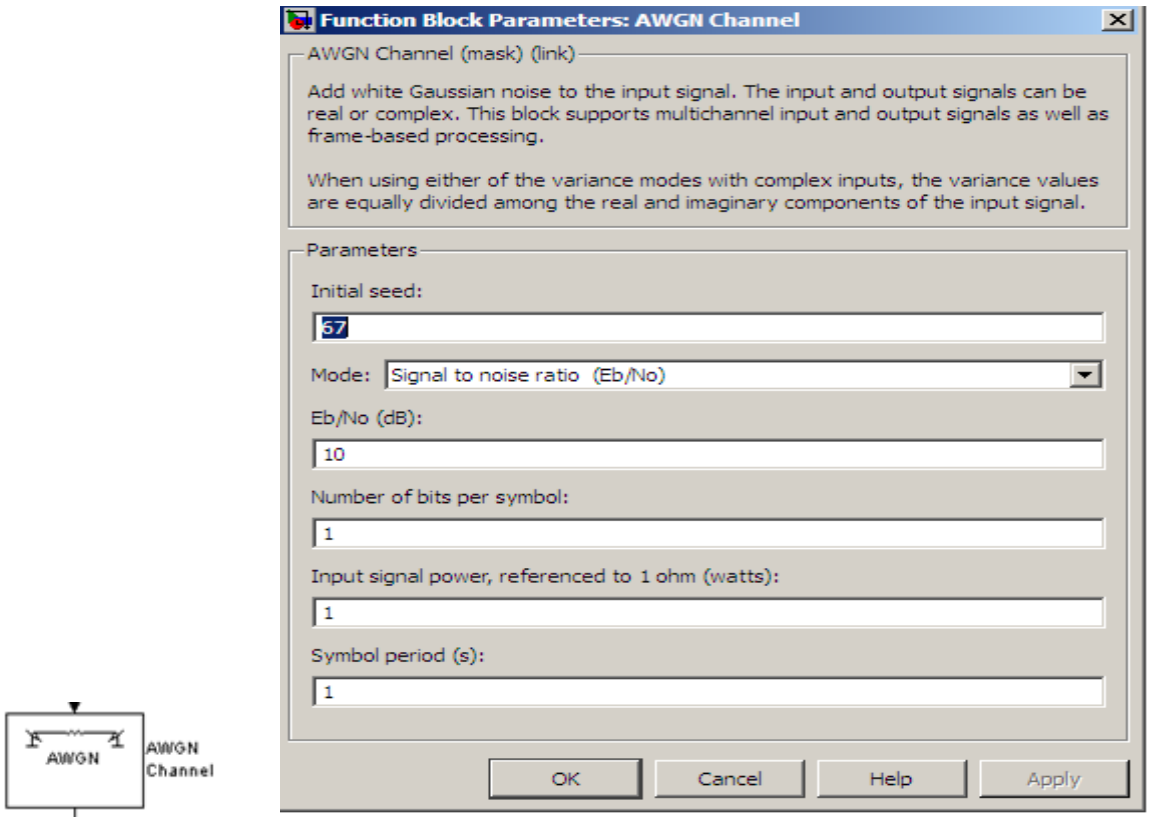
d- Blocs de démodulation Bpsk :





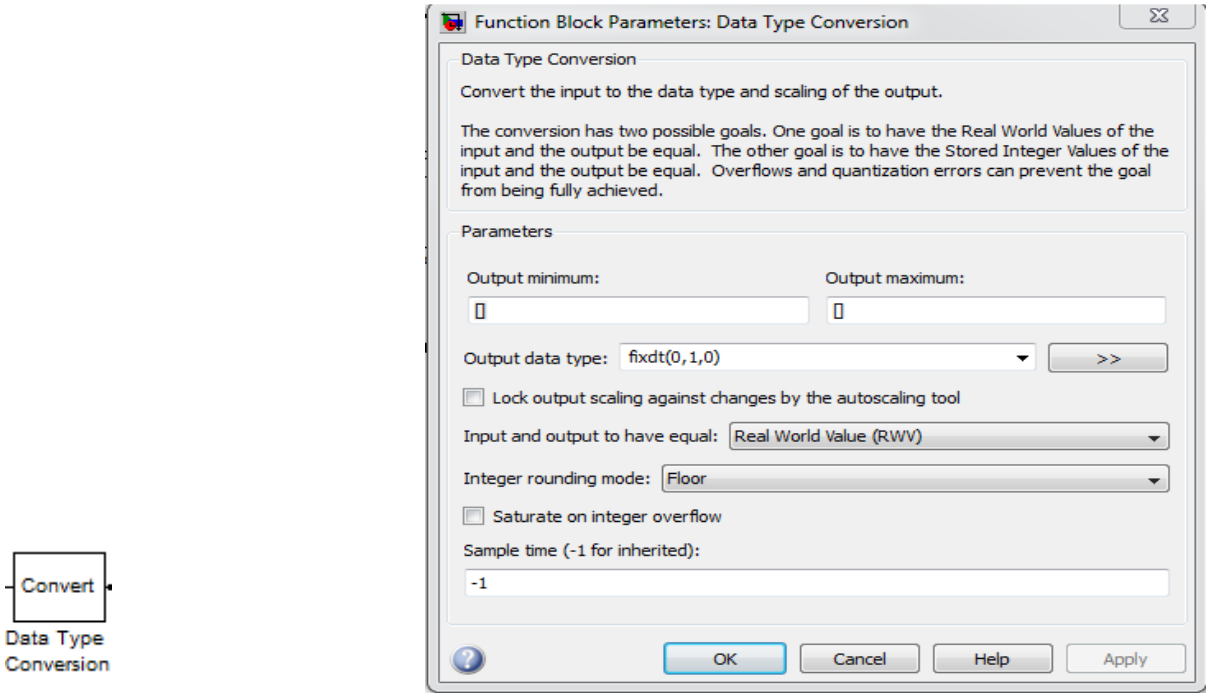
Figure(III.5) Bloc de démodulation BPSK ainsi que sa boite de dialogue.

e- Canal de bruit :



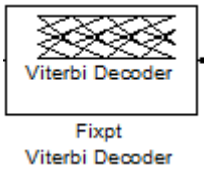
Figure(III.6) : Bloc canal de bruit ainsi que la boite de dialogue.

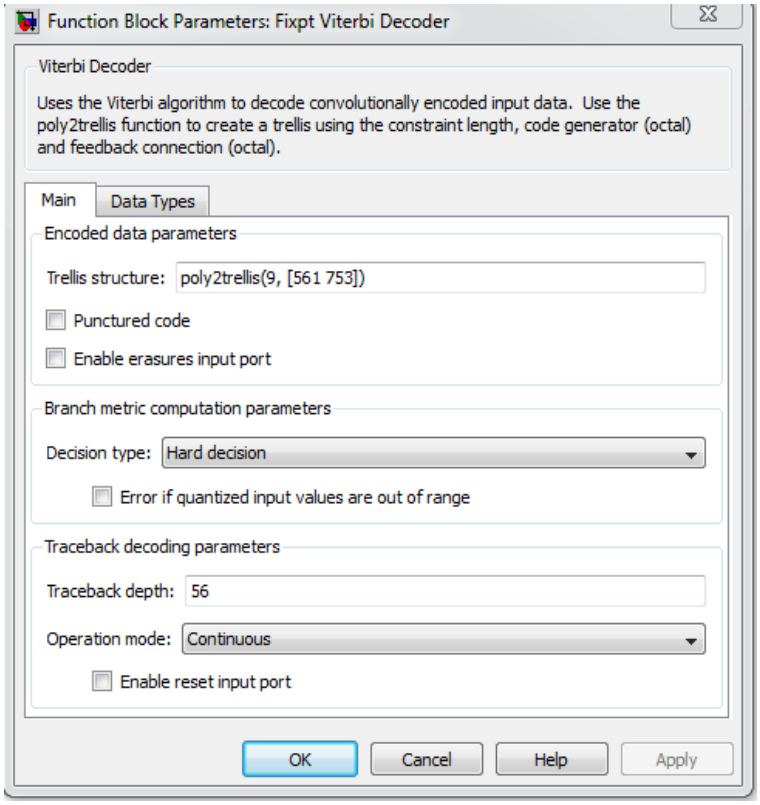
f- Bloc convertisseur :



Figure(III.7) : Bloc de convertisseur ainsi que la boite de dialogue.

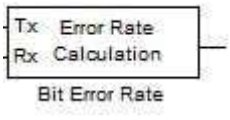
g- Bloc décoder véterbi :

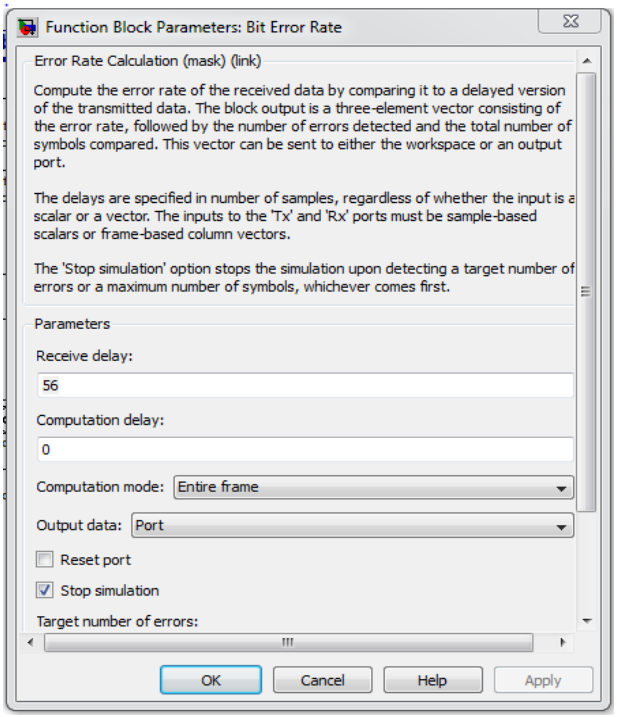




Figure(III.8) : Bloc décodeur viterbi ainsi que la boite de dialogue.

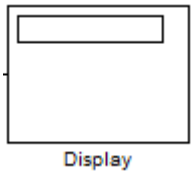
h- Bloc de bite erroné :

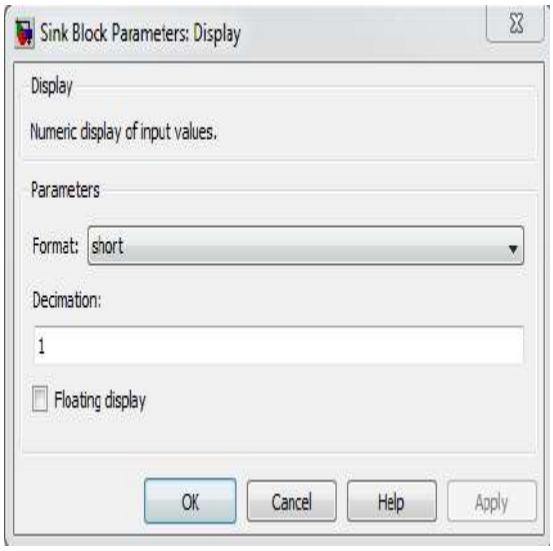




Figure(III.9) : Bloc bit ainsi que la boite de dialogue.

k- Bloc d'affichage :





Figure(III.10) : Bloc d'affichage ainsi que la boite de dialogue.

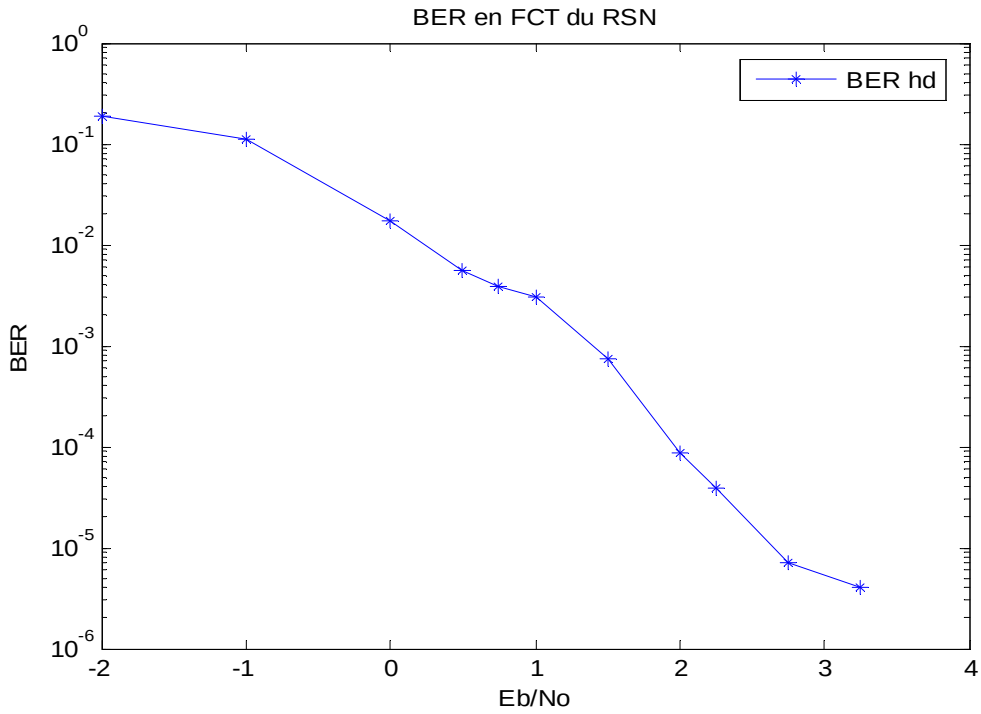
III.2.1.1 : calcul de BER , cas de hard-dicision

$$Eb/N0(dB) = EbN0 - 10log(2)$$

$EB/N0_{dB}$	-2	-1	0	0.5	0.75	1
BER_{HD}	0.19	0.1091	0.01716	0.00556	0.003908	$3.013*10^{-3}$

1.5	2	2.25	2.75	3.25	3.75	4
$0.743*10^{-3}$	$0.088*10^{-3}$	$0.039*10^{-3}$	$0.007*10^{-3}$	$0.004*10^{-3}$	0	0

Tableau [III.1] : résultats de BER en fonction de EB/N0 en hard-dicision

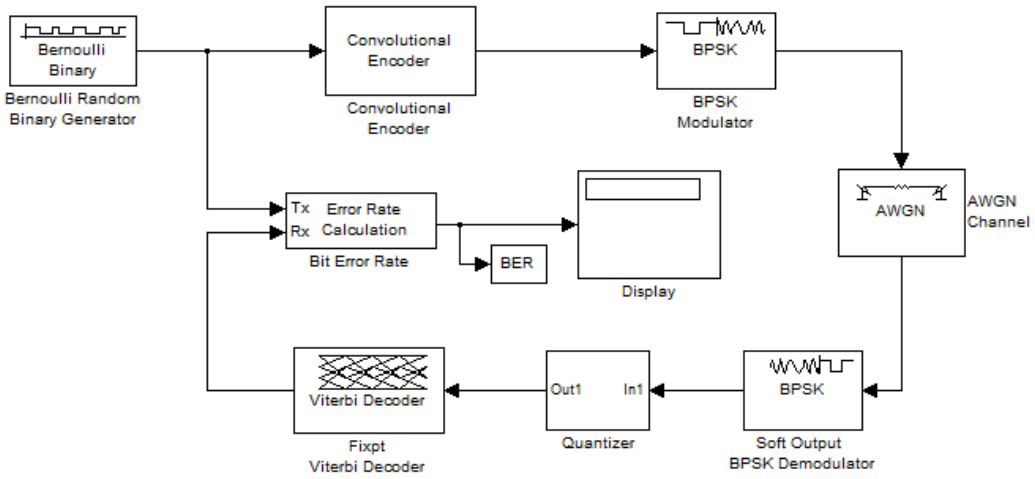


Figure(III.11) : Courbe du BER en fonction de E_b/N_0 ; cas hard-dicision

D'après les résultats obtenus, on remarque que le BER est grand quand le niveau du bruit du canal est relativement grand par rapport à celui de l'énergie des bits (énergie du signal), et devient très faible de l'ordre de 10-5 quand l'énergie du signal devient très grande par rapport à l'énergie du bruit.

III.2.2 :Cas de décision logiciel (soft-décision) :

La chaine de transmission qu'on va l'utiliser est la suivante :



Figure(III.12) : Chaine de transmission à décision logiciel.

Ce schéma de décision soft contient les mêmes éléments que le schéma hard décision saut qu'il y a une variation au nivaux du générateur de bernoulli ainsi que le modulateur et le démodulateur BPSK soft.

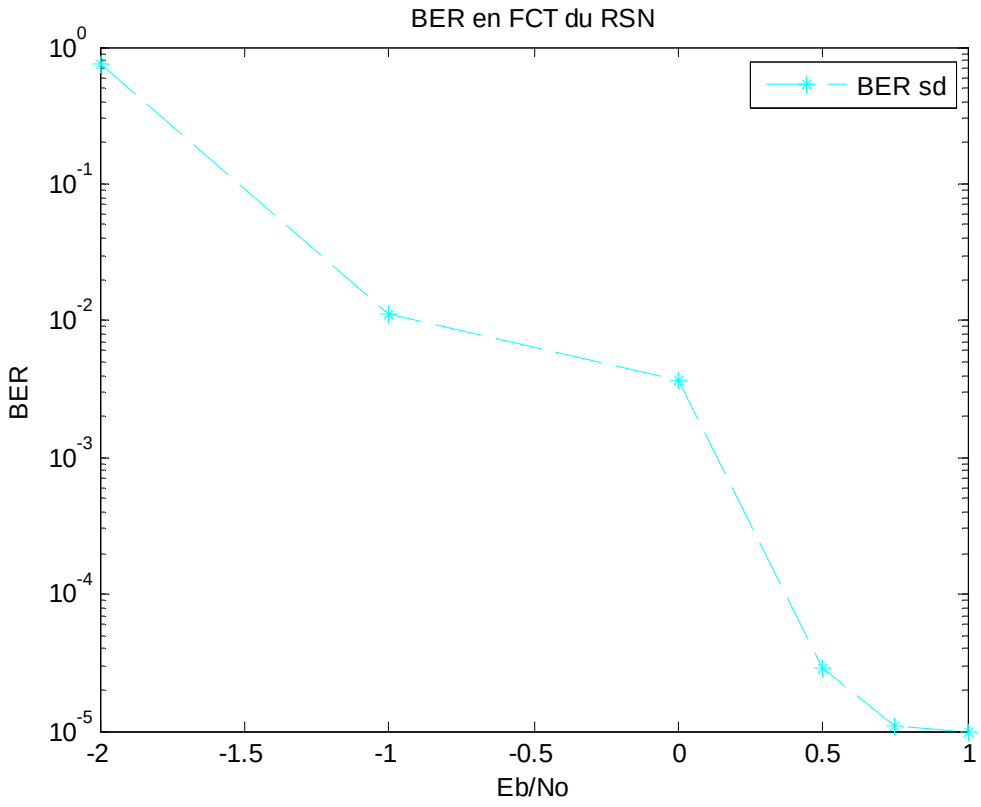
III.2.2.1 Calcul de BER , cas de soft-dicision :

Le tableau suivant montre les résultats de la variation du BER dans le cas de la chaine soft décision :

$EB/N0_{db}$	-2	-1	0	0.5	0.75	1
BER_{SD}	0.75	0.01113	$3.737*10^{-3}$	$2.9*10^{-5}$	$1.1*10^{-5}$	10^{-5}

1.5	2	2.25	2.75	3.25	3.75	4
0	0	0	0	0	0	0

Tableau [III.2] : résultats de BER en fonction de EB/N0 _ soft-dicision

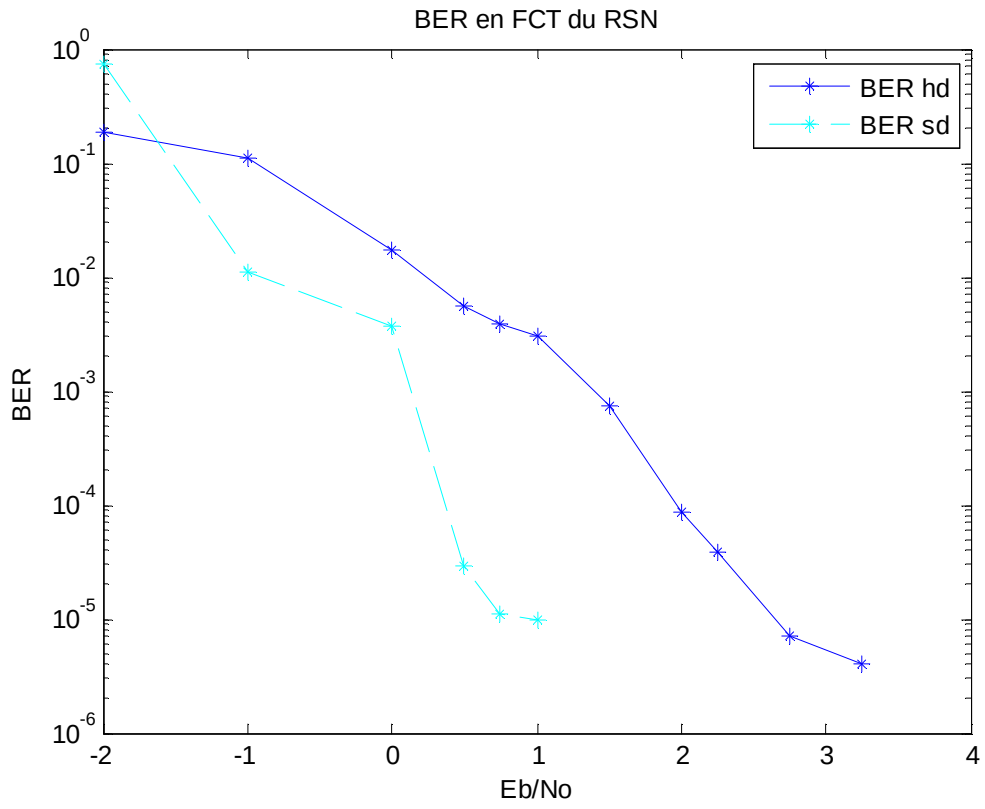


Figure(III.13) : Courbe du BER en fonction de E_b/N_0 ; cas de soft-dicision

D'après les résultats obtenus, on remarque que le BER est grand quand le niveau du bruit du canal est relativement grand par rapport à celui de l'énergie des bits (énergie du signal), et devient très faible de l'ordre de 10-5 quand l'énergie du signal devient très grande par rapport à l'énergie du bruit.

III.2.3 Évolution de BER en fonction du rapport E_b/N_0 dans les deux cas :

Pour comparer les résultats de mise en œuvre des deux systèmes hard décision et soft décision, on trace les deux courbes dans la même figure comme suit :



Figure(III.14) : Courbe d'évolution du BER en fonction de E_b/N_0

D'après les résultats obtenus, on remarque que le BER dans le cas de la soft décision présente de meilleurs résultats due la hard décision pour Eb/N0 >-1dB.

Et que la soft décision devient insensible au bruit pour des rapports (Eb/N0 =1dB). Par contre la hard décision devient insensible au bruit pour des rapports (Eb/N0 =3.25dB).

III.3 **Conclusion** :

Dans ce chapitre nous avons présenté deux approches d'application du codage convolutif, l'approche hard décision et l'approche soft décision.

D'après les résultats obtenus en calculant le BER en fonction du rapport Eb/N0, nous avants trouvé que l'approche soft décision présente de meilleurs résultats de décodage en présence du bruit du canal par rapport à celui de l'approche hard décision.

Conclusion Générale

Conclusion générale :

Dans ce mémoire, nous avons effectué une étude théorique des différentes techniques de codage du canal. Car nous avons basé sur la technique du codage convolutif, pour maître en œuvre ces algorithmes en les appliquant dans une chaîne de transmission numérique.

De plus nous avons utilisé le SIMULINK sous Matlab pour s'assurer du bon fonctionnement du codage convolutif ainsi que son décodage à base d'algorithme de Viterbi, ou la présence du bruit gaussien du canal est considérée.

Une étude des performances de codage convolutif a été effectuée en basant sur l'évolution du BER en fonction du rapport Eb/N0 pour deux approches de chîne de transmission (Hard décision et Soft décision).

Cette étude nous a permet de dire que l'approche Soft décision a été meilleur par rapport à celle de la décision Hard de point de vu robustesse contre la présence du bruit comme il est montré dans le chapitre précédent.

Notre travail a été bénéfique et demande d'être suivi par d'autres techniques de codages, tel que : le codage cyclique… Etc.

Bibliographie

Bibliographie :

[1] M. Bossert, "*Channel Coding for Telecommunications*,**"** John WileyAnd Sons Ltd, 1999.

[2] 1. G. Proakis, "*Digital Communications*," Third Edition, MacGrawHill, 1995.

[3] A. S. Barbelescu and S. S. Pietrobon, "*Interleaver Design for Turbo Codes*,**"** *Electron. Let/.*, vol. 30, no. 25, pp. 2107-2108, Dec. 1994.

[4] S. Dolinar and D. Divsalar, "*Weight Distribution for Turbo Codes using Random and Non-Random Permutations*,**"** *Communications Systems and Research section* TDA report 42-122 August 1995.

[5] 1. G. Proakis, M. Salehi, "*Communication Systems Engineering*,**"** Prentice-Hall, 1994

[6] C. Berrou, A. Galvieux and P. Thitimajshima*" Near Shannon Limit Error-Correcting Coding and Decoding :(Turbo Codes),*"**.** *École Nat. Sup de Télécommunications De Bretagne France.* 1993.

[7] G.C Clark, J.B Cain , "*Error Correcting Coding for Digital Communication***'',** Plenum Press 1981.

[8] D.J Costello, S.Lin, "*Error Control Coding : Fondamentals and Applications***''**, Prentice Hall 1983.

[9] M.A. Sahraoui, A. Zebida , « *Codes Détecteurs et Correcteurs d'erreurs appliquées aux images fixes* **»,** Institut d'Informatique, USTO 1995.

[10] COURS "*codage correcteur d'erreurs convolutionnel***''**

[11] O. Pothier cours "*Codage de canal et turbo-codes, cours 5 « Les codes convolutifs »***''** 13/10/2000.