UNIVERSITÉ DE MONTRÉAL

CODES CONVOLUTIONNELS À TEMPS VARIANT QUASI APÉRIODIQUES

SAMUEL OUZAN
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(GÉNIE ÉLECTRIQUE)
AOÛT 2008

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-46070-2
Our file Notre référence
ISBN: 978-0-494-46070-2

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CODES CONVOLUTIONNELS À TEMPS VARIANT QUASI APÉRIODIQUES

présenté par: OUZAN Samuel
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. CARDINAL Christian, Ph.D, président-rapporteur
M. HACCOUN David, Ph.D, directeur de recherche
M. FRIGON Jean François, Ph.D, membre du jury

À Hannah

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche, Monsieur David Haccoun, pour son appui et sa présence durant toutes les étapes de ce mémoire ainsi que ses conseils et commentaires constructifs sans lesquels ce mémoire n'aurait pu voir le jour. Je lui suis tout particulièrement reconnaissant pour la confiance qu'il ma accordé et pour les moyens financiers et logistiques qu'il a mis à ma disposition.

Je souhaite remercier aussi Olfa, Zouheir, Benjamin, Éric, Stéphane, Florien, Zhao, Laurent, David, Nadia, Lamia, et Oualid. Ainsi que tous les membres de la section des télécommunications de l'École Polytechnique de Montréal.

Finalement, je voudrais tout particulièrement remercier mon épouse Myriam, ma fille Salomé, mon fils Amos, ma mère Ruth, mon père Denis, ma sœur Hannah ainsi que Jérémie. Pour leur aide, leur amour et leur soutien.

RÉSUMÉ

Le travail de recherche présenté dans ce mémoire est consacré au développement d'une nouvelle classe de codes correcteurs d'erreurs intitulés : *codes convolutionnels à temps variant quasi apériodiques*. Ces codes ont pour mission d'élargir la structure aléatoire des codes convolutionnels classiques en faisant varier dans le temps, et de manière aléatoire, les connexions d'un codeur convolutionnel sur une longue période. Cette variation sur une longue période permet d'élargir l'ensemble des codes sur lequel notre code particulier est choisi. Le décodage mis au point pour décoder ce type de codes est un décodeur de Viterbi adaptatif qui s'adapte aux variations du codeur.

Les codes modernes de type Turbo ou LDPC sont des codes très puissants, ils permettent de se rapprocher de manière exceptionnelle de la limite théorique imposée par Shannon en termes de performance d'erreur. Cependant, ces systèmes peuvent s'avérer complexes, quant à leur mise en œuvre, et souffrent de délai associé au décodage important. À l'inverse, les codes convolutionnels sont connus pour être plus simples à implémenter et offrent peu de latence au décodage. C'est la raison pour laquelle malgré que les performances d'erreur des codes convolutionnels soient inférieures à celles des codes de type Turbo ou LDPC, les codes convolutionnels avec décodage de Viterbi demeurent largement utilisés dans les systèmes de communication actuels.

Dans cette optique, le système de codage proposé a pour objectif de garder la composante pratique et optimale du décodage de Viterbi tout en ajoutant de l'aléa par rapport aux codes convolutionnels classiques. Ce travail de recherche constitue une première approche. Il analyse le rapport entre performance d'erreur et complexité de certains codes convolutionnels à temps variant quasi apériodiques. Les codes non systématiques ainsi que les codes récursifs systématiques sont étudiés.

ABSTRACT

The research presented in this thesis is focused on the development of a new class of error correcting codes named: *Quasi-Non Periodic Time Varying Convolutional Codes.* The purpose of these codes is to enlarge the random structure of classical convolutional codes by varying in time and in a random manner, the connections of a convolutional encoder over a long period of time. This variation over a long period of time allows the enlargement of the set of codes from which our specific code is chosen. As a decoder, we develop an adaptive version of the Viterbi algorithm that consists to follow the variations of the coding system.

Modern codes, such as Turbo or LDPC are very powerful; they allow us to come very close to the *Shannon theoretical limit* in terms of error performance. However these systems can prove to be complex when it comes to their practical applications. On the other hand convolutionnal codes are known for being more easily implemented. This is the reason why, in spite of the error performances of convolutional codes being inferior to that of modern codes, the convolutional codes with the Viterbi Decoder are still widespread in today's communication systems.

With that in mind, the purpose of the coding system proposed in this work is to maintain the practical aspect of the Viterbi decoder, while adding a random structure to classical convolutional codes. This work analyzes the connection between error performances versus complexity of certain quasi-non periodic time varying convolutional codes. The non systematic and recursive systematic codes are presented.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET DES SYMBOLES

Sigles

ACS	Add Compare Select
APP	A *posteriori* Probability
AWGN	Additive White Gaussian Noise
BCJR	Bahl Cocke Jelinek Raviv
BPSK	Binary Shift Keying
CTV-P	Convolutionnel à Temps Variant Périodique
CTV-QA	Convolutionnel à Temps Variant Quasi Apériodique
CTV-QA-S	Convolutionnel à Temps Variant Quasi Apériodique Systematique
CTV-QA-RS	Convolutionnel à Temps Variant Quasi Apériodique Récursif Systématique
LDPC	Low Density Parity Check
PAM	Pulse Amplitude Modulation
QAM	Quadrature Amplitude Modulation
SNR	Signal to Noise Ratio

Symboles

$\oplus$	Addition modulo 2
$\lfloor . \rfloor$	Fonction partie entière
$\lceil . \rceil$	Fonction plafond
B	Largeur de bande
C	Capacité du canal
$d_{\min}$	Distance minimale
d_{free}	Distance libre
E_b	Énergie par bit d'information
E_s	Énergie par symbole codé
K	Nombre de cellules dans le registre à décalage d'un codeur convolutionnel
M	Mémoire du codeur
P	Puissance moyenne du signal d'entrée
R	Taux de transmission ou taux de codage suivant le contexte
T	Période d'un codeur convolutionnel à temps variant
ρ	Efficacité spectrale
$I(X,Y)$	Information mutuelle entre X et Y
$D(p \square q)$	Divergence de Kulback- Leibler : $\int_{-\infty}^{+\infty} p(x) \log \frac{p(x)}{q(x)} dx$

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

1.1 Motivations

Le codage de canal a pour objectif d'améliorer la transmission numérique d'information à travers un canal bruité. Cette amélioration peut se traduire soit par un gain en vitesse de transmission, soit par une économie en puissance selon les applications. Le gain procuré par le codage ne peut dépasser une certaine limite. Cette limite appelée capacité du canal fut établie il y a soixante ans par Claude Shannon dans un article fondamental qui s'intitule « *une théorie mathématique des communications* » [1]. Shannon dans cet article démontra analytiquement que pour atteindre cette limite ultime, il fallait utiliser un système de codage pseudo aléatoire de très grande dimension. Il n'indiqua cependant pas de systèmes concrets capables de telles prouesses.

Peter Elias prouva en 1955 d'un point de vue théorique, que n'importe quel code linéaire construit à l'aide d'une matrice génératrice, choisie de façon aléatoire, et de dimension assez grande pouvait atteindre la capacité à l'encodage. Malheureusement, le problème résidait dans le décodage. Il s'agissait de le décoder avec un décodage optimal, en comparant le mot de code reçu avec tous les autres mots de code possibles. Compte tenu de la très grande longueur du code, la complexité devenait vite extrêmement élevée. Il décida alors de chercher des codes avec plus de structures et inventa les codes convolutionnels [2].

En 1967 Andrew Viterbi inventa un décodage à maximum de vraisemblance pouvant décoder les codes convolutionnels de façon optimale [3]. À partir de cet instant les codes convolutionnels ont dépassé la plupart des systèmes de codage existants. Le décodage de Viterbi connut un réel succès et fut l'objet d'un impact économique majeur.

Ce n'est que 25 ans plus tard, pour la première fois qu'un système de codage fut capable d'atteindre à quelques fractions de dB près de la capacité. Ce sont deux chercheurs français et leur étudiant ; Berrou, Glavieux et Thitimajshima qui furent à l'origine de cette invention. Ils proposèrent un système de codage basé sur la concaténation de deux codeurs convolutionnels, séparés par un long entrelaceur pseudo aléatoire [4]. L'entrelaceur avait pour mission d'introduire de l'aléa à l'encodage et pour particularité de pouvoir être décodé par un décodeur itératif. Ce système original fut baptisé Turbo. Les codes Turbo ont donné naissance à d'autres systèmes de codage très performants [5-7]. Ils partagent tous principalement les mêmes particularités. Ils contiennent un long entrelaceur pseudo aléatoire et sont pour la plupart décodés par des techniques itératives.

Malgré les performances exceptionnelles que procurent les codes de type Turbo, les codes convolutionnels décodés à l'aide du décodage de Viterbi restent parmi les systèmes les plus utilisés en pratique. Ils sont actuellement intégrés à travers la planète dans plus d'un milliard de téléphones cellulaires, et l'entreprise Qualcomm a estimé récemment à plus de 10^{15} le nombre de bits décodés à chaque seconde par l'algorithme de Viterbi, dans les téléviseurs numériques [8]. La popularité du décodage de Viterbi est due au fait qu'il bénéficie de très bonnes performances pour une faible complexité. Mais aussi que la latence au décodage est pratiquement inexistante contrairement aux codes de type Turbo où la latence ne peut être négligée.

Ainsi pour pouvoir bénéficier de l'optimalité de l'algorithme de Viterbi, et introduire la composante aléatoire indispensable aux systèmes de codage modernes, nous avons élaboré un système de codage nouveau. Ce système est basé sur la permutation de façon pseudo aléatoire et quasi non répétitive dans le temps des connexions d'un codeur convolutionnel, permettant ainsi de créer des codes moins structurés que les codes convolutionnels classiques. Cette permutation quasi non répétitive ; et donc très longue dans le temps, peut être apparentée au long entrelaceur des codes de type Turbo, puisque

qu'elle a pour objectif de générer l'aléa indispensable au codage aléatoire. L'avantage de ce système est de produire un codage de type aléatoire pouvant être décodé par un décodage de Viterbi adaptatif. Ce décodage est optimal et consiste à trouver le plus court chemin dans un treillis, où les symboles assignés aux branches à chaque instant dépendent des connexions spécifiques de l'encodeur à ce même instant. Ces symboles varient ainsi dans le temps et ne sont plus identiques comme cela était le cas entre deux mêmes états, dans un décodeur de Viterbi classique [9].

Ce mémoire a donc pour objectif d'illustrer cette nouvelle technique de codage, d'en calculer ses performances d'erreur dans un canal Gaussien, d'en étudier la complexité et d'en établir les limitations.

1.2 Contributions

Les contributions apportées par ce travail sont les suivantes :

1. Découverte et élaboration des codes convolutionnels à temps variant quasi apériodiques.

2. Mise en œuvre d'une structure de décodage de Viterbi adaptatif capable de décoder des codes de types pseudo aléatoires.

3. Conception de programmes capables de simuler le système de codage et décodage des codes convolutionnels à temps variant quasi apériodiques, pour des codes récursifs et non récursifs, pour des taux de codage de 1/2 et 1/3 .

4. Analyse et évaluation des performances d'erreur associées à la classe des codes convolutionnels à temps variant quasi apériodiques.

Les simulations ont été effectuées à l'aide d'ordinateurs munis de processeurs Pentium® IV d'Intel® cadencés à 3 et 1.5 GHz et possédant respectivement 1 Gb et 512 Mb de RAM sous l'environnement Windows XP. Les logiciels de programmation Matlab® et Microsoft Visual studio® ont été utilisés pour concevoir les simulateurs.

1.3 Organisation du mémoire

Le mémoire se divise en cinq parties :

- **Première partie – concepts théoriques fondamentaux**

Nous commençons tout d'abord par présenter les systèmes de communication numérique dans un canal Gaussien, pour ensuite discuter de l'impact du codage de canal sur ces systèmes, et aboutir enfin au développement analytique permettant de construire un système de codage optimal.

- **Deuxième partie – Principales techniques de codage**

Nous présentons les systèmes de codage les plus populaires. Nous discutons des contraintes inhérentes aux systèmes pratiques. Et nous faisons le lien avec la partie précédente en montrant que les systèmes de codage modernes les plus performants, respectent au maximum l'aspect théorique du codage de canal.

- **Troisième partie – codage à temps variant et algorithme de Viterbi adaptatif**

Cette partie représente le cœur du mémoire. Elle illustre un nouveau système de codage de type pseudo aléatoire, qui a la particularité de pouvoir être décodé par un algorithme de Viterbi adaptatif, qui est optimal. Ce système s'inspire des deux parties précédentes. Il essaye effectivement de prendre en compte la théorie du codage aléatoire, sans pour autant dénigrer l'aspect réalisation, des systèmes de codage pratiques.

- **Quatrième partie – Introduction de la récursivité**

Dans cette section nous diminuons la complexité au décodage des codes non systématiques en introduisant de la récursivité aux codeurs systématiques.

- **Cinquième partie – synthèse et travaux futurs**

La cinquième partie résume l'ensemble des travaux effectués et propose certaines extensions envisageables, ainsi que des pistes inexplorées relatives à la recherche prosopopée.

CHAPITRE 2

ÉLÉMENTS DE LA THÉORIE DU CODAGE DE CANAL

2.1 Introduction

L'objectif principal de ce chapitre est de présenter les bases de la théorie du codage correcteur d'erreurs pour un canal Gaussien. Cette théorie, développée par Shannon [1], permet d'une part d'établir des limites supérieures sur le taux de transmission et d'autre part, indique comment mathématiquement, il est possible de coder l'information afin d'atteindre ces mêmes limites tout en assurant une transmission avec une probabilité d'erreur aussi petite que désirée. Néanmoins, avant de définir les bases et les hypothèses de travail indispensables à ce développement et à notre recherche en général, nous continuons la section d'introduction en présentant le modèle de transmission numérique.

2.1.1 Modèle de transmission numérique

La figure 2.1 représente le schéma d'un système de communication numérique. Il est organisé de façon symétrique par rapport au canal de transmission.

La *source* est en général analogique, elle est numérisée puis compressée par l'*encodeur de source* pour délivrer une séquence binaire. La compression sert principalement à éliminer la redondance présente dans la séquence originale. Cet aspect ne sera pas traité dans ce mémoire.

La séquence binaire qui représente l'information compressée, est encodée par le *codeur de canal*, qui la transforme en une autre séquence binaire. Cette opération a pour objectif

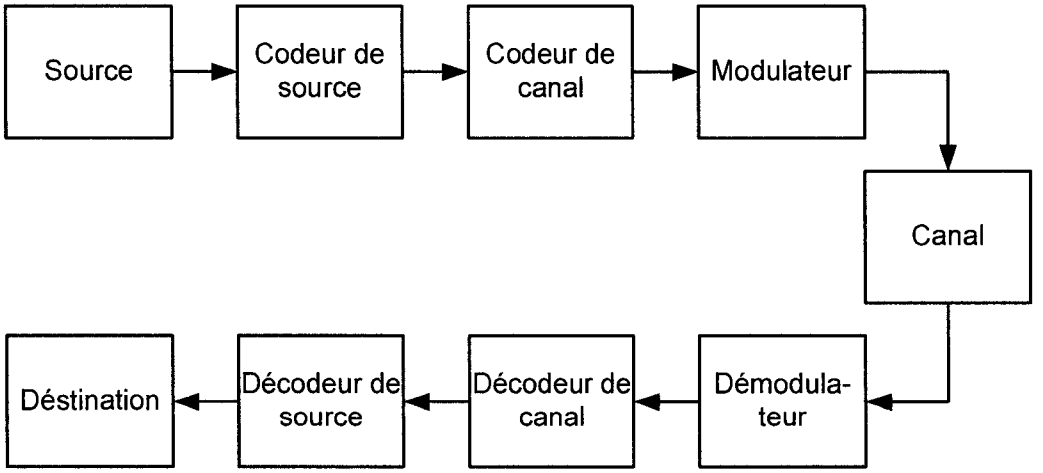

Figure 2.1: *Schéma d'un système de communication numérique*

d'améliorer la transmission à travers le canal bruité. La forme la plus simple de codage est la répétition; en répétant chaque bit plusieurs fois, l'information présente dans le message codé est ainsi mieux protégée. Il est possible de diviser les techniques de codage en trois catégories principales ; les codes en blocs, les codes convolutionnels et les codes en graphes. Le codage algébrique en blocs, consiste à construire entre autres, des codes qui maximisent la distance minimale[1], alors que le codage convolutionnel et le codage en graphe sont plus concernés à trouver des classes de codes, qui optimisent la performance moyenne, en fonction de la complexité du décodage [10]. Les plus populaires des codes en blocs sont les codes de Hamming [11], de Golay [12], de Reed-Muller [13], BCH [14-15], et de Reed-Solomon [16]. Ces codes font en général appel à une construction algébrique très complexe. Les codes modernes, ultra performants, comme les codes LDPC [17] ou les codes Turbo [4], entrent dans la classe des codes en graphes. Ces codes sont directement inspirés de l'approche probabiliste du codage de Shannon [1]. Le codage de canal, convolutionnel à temps variant quasi apériodique,

[1] La distance minimale d'un code linéaire correspond au poids de Hamming (le nombre de composantes non nulles) minimal de tous les mots du code de poids non nul. Cette notion sera traitée plus en détail dans le prochain chapitre

présenté dans ce mémoire, espère faire lien entre le codage convolutionnel classique, et le codage dit *probabiliste* (codes en graphes). Enfin, le codage s'applique principalement à deux régimes distincts : le codage binaire pour les applications limitées en puissance et le codage multiniveaux, pour les applications limitées en fréquence qui disposent d'une énergie relativement abondante [10]. Seul le codage binaire sera considéré dans ce mémoire.

La séquence binaire à la sortie du codeur de canal est ensuite transmise au *modulateur*. Ce dernier assigne un signal analogique, à un ou plusieurs symboles codés selon la modulation utilisée, permettant ainsi la transmission à travers le canal.

Le *canal* est le médium qui véhicule l'information. Dans le cas d'une liaison spatiale, le canal est l'espace intersidéral. Pour une communication téléphonique, il peut être une fibre optique, un lien micro-ondes ou encore l'atmosphère s'il s'agit d'une communication sans fil. Une des difficultés majeures en télécommunication est la modélisation du canal. Les perturbations causées par ce dernier ne sont pas déterministes. C'est la raison pour laquelle sa modélisation fait appel à des outils mathématiques complexes, tels que les processus stochastiques. Dans le cas d'une transmission téléphonique filaire ou d'une communication spatiale, le modèle du canal accepté est le canal Gaussien, alors que pour les applications mobiles, les canaux dits de Raleigh sont plus appropriés [18]. Le modèle du canal, utilisé pour notre recherche est le canal Gaussien. Ce choix provient du fait, que la majorité des découvertes et des travaux de recherche dans le domaine du codage ont été effectués pour un canal Gaussien, mais aussi parce qu'il constitue le point de référence, pour la comparaison entre plusieurs systèmes de codage différents.

Le *démodulateur* a la fonction inverse du modulateur, il change chaque signal analogique reçu en une séquence de symboles numériques.

Le *décodeur* quant à lui, a pour mission de retrouver le mot de code original associé à la séquence reçue, qui a été dégradé par les perturbations du canal. Comme mentionné dans l'introduction, le décodage souffre souvent de latence et de temps de calcul excessifs. Le codeur et le décodeur sont très liés, et ils ne peuvent être imaginés séparément. Les systèmes de codage les plus performants tels que les codes LDPC [17], ou les codes Turbo [4] sont des exemples en termes de compatibilité entre le codeur et le décodeur. Le décodage proposé dans ce mémoire est un décodage de Viterbi adaptatif considéré à la fois pour son optimalité, mais aussi pour sa capacité à s'adapter au codeur. Inversement, le système de codage proposé se veut d'avoir les propriétés du codage aléatoire, tout en pouvant être décodé par l'algorithme de Viterbi [3].

Enfin, le *décodeur de source* reconstruit le message original numérique ou non, correspondant à la séquence d'information numérique fournie par le décodeur.

Après avoir indiqué que le canal utilisé dans notre recherche était Gaussien, et le type de codage binaire, nous allons faire un bref historique de l'évolution du codage pour les missions spatiales [19]. Ces applications sont souvent coûteuses en termes de largeur de bande et extrêmement coûteuses en termes de puissance. Le codage de canal est donc également binaire, et le bruit généré par le canal de transmission est uniquement Gaussien.

2.1.2 Historique des codes pour les missions spatiales

La première technique de codage utilisée était un code convolutionnel de taux 1/2, et de longueur de contrainte 20, pour la mission Pioneer en 1968. Le récepteur utilisant une quantification douce à 3-bit, était un décodeur séquentiel [24] de 1 MHz de cadence d'horloge. Le gain réel de codage pour ce système était de 3.3 dB, pour une probabilité de bit en erreur de $5 \cdot 10^{-3}$ [19].

Ensuite dans les années 70 le standard est devenu une concaténation d'un code convolutionnel interne de taux 1/3 d'une longueur de contrainte de 6, avec un code externe (255, 223, 33) de Reed Solomon [16]. Ces systèmes pouvaient atteindre un gain de codage de 8.3 dB pour une probabilité de bit en erreur de 10^{-6}.

C'est en 1992 pour la Mission Galileo que le plus grand décodeur Viterbi fut développé. Le système utilisé était une concaténation d'un code convolutionnel de taux 1/4 et de longueur de contrainte 14 comme code interne, et un code externe de Reed Solomon à longueur variable. Le gain réel de codage pour ce système a été de 10.6 dB, pour une probabilité de bit en erreur de $2 \cdot 10^{-7}$ [19].

Enfin, le standard actuel est un système de codage Turbo, développé par la NASA qui se rapproche de manière significative (moins de 1dB) de la limite de Shannon [22].

Les applications spatiales sont d'excellentes références en termes d'évolution des systèmes de codage de canal. Effectivement, un gain de codage d'une fraction de dB a un impact économique majeur. Les techniques de codage les plus puissantes ont été développées pour ces applications, et nous pouvons remarquer que le décodage de Viterbi fut utilisé pendant plus de 30 ans. Le travail de recherche effectué pour ce mémoire fut inspiré par l'évolution du codage pour ce genre d'application et c'est la raison pour laquelle il nous a paru essentiel d'en faire la présentation.

2.2 Canal AWGN

Dans cette section nous présentons les principaux paramètres du canal à bruit additif Gaussien (Additif White Gaussian Noise Channel : AWGN). Par le biais du système numérique à modulation d'amplitude (Pulse Amplitude Modulation : PAM), nous établissons le lien entre les paramètres du canal à temps continu et celui à temps discret. Une fois, ce lien établi, le système de la figure 2.1 peut être modélisé de manière entièrement discrète, l'analyse des signaux dans un espace Euclidien est alors possible, et le théorème de la capacité du canal de Shannon [1] devient alors démontrable.

2.2.1 Modulateur PAM

Considérons la figure 2.2, le vecteur d'entrée $\boldsymbol{x} = \{x_j\}, j = 1.....N$ est en général une séquence aléatoire. Dans notre cas, cette séquence représente la sortie de l'encodeur. Pour éviter le phénomène de crénelage (repli de spectre), d'après le théorème de Nyquist [20], il doit entrer jusqu'à $2W$ symboles par seconde dans le modulateur PAM, lorsque la largeur de bande B dans l'intervalle de fréquences positives est égale à W (Hz).

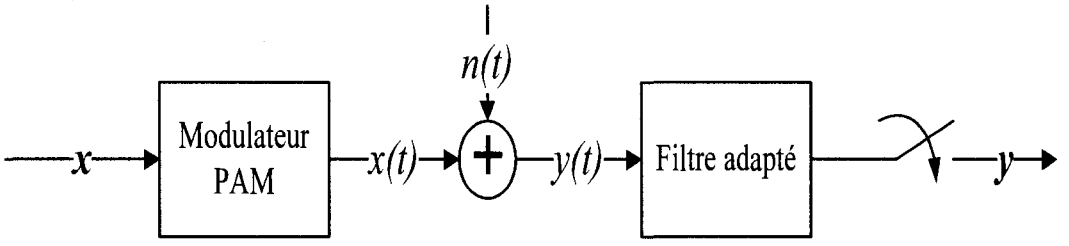

Figure 2.2: *Système PAM*

Le signal à la sortie du modulateur PAM est égal à :

$$x(t) = \sum_{j=1}^{N} x_j \phi_j(t) \tag{2.1}$$

où les $\phi_j(t) = p(t - jT),\ j = 1, 2.....N$ sont des fonctions de base orthonormées. Le processus aléatoire Gaussien *n(t)* de densité de puissance spectrale unilatérale N_0 qui représente le bruit généré par le canal, qui s'ajoute au signal *x(t)* :

$$y(t) = x(t) + n(t) \tag{2.2}$$

L'échantillonnage à la sortie du filtre adapté s'effectue tous les $t = jT$ secondes, $j = 1, 2.....N$, la séquence de sortie discrète $\mathbf{y} = \{y_j\},\ j = 1.....N$, où les éléments y_j sont donnés par :

$$y_j = \int_{-\infty}^{+\infty} y(\tau) p(\tau - jT) d\tau \tag{2.3}$$

La puissance moyenne du signal d'entrée est dénotée *P*. Le rapport signal à bruit pour ce canal est donné par :

$$\text{SNR} = \frac{P}{N_0 W} \tag{2.4}$$

où SNR représente le rapport entre la puissance du signal et la puissance du bruit, dans toute la largeur de bande de transmission.

Comme nous l'avons indiqué au début de la section, nous voulons faire le lien entre les paramètres à temps continu et ceux à temps discret. Le développement de Karhunen et Loève [57] permet, dans le cas des récepteurs optimaux (filtres adaptés), de décomposer

le bruit blanc Gaussien stationnaire en série de fonctions membres de processus stochastiques :

$$n(t) = \sum_{j=1}^{N} n_j \phi_j(t) \tag{2.5}$$

où les fonctions de base orthonormées $\phi_j(t)$ sont les mêmes que celles utilisées pour le signal d'entrée $x(t)$. Par conséquent, le bruit $\boldsymbol{n} = \{n_j\}, j = 1,2....N$ est une séquence Gaussienne dont les éléments sont indépendamment et identiquement distribués avec une moyenne égale à zéro et une variance de $N_0/2$. $\boldsymbol{n}$ le bruit d'entrée a un spectre de densité de puissance égale à $N_0/2$.

En fonction du choix des fonctions de base orthonormées, x_j peut prendre plusieurs valeurs. Pour le codage binaire $x_j, j = 1,2.....N$ est égale à $\pm\alpha$. Le modèle à temps discret de la figure 2.3 ci-dessous équivaut au modèle à temps continu de la figure 2.2 .

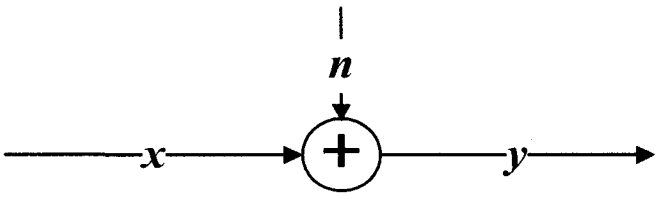

Figure 2.3: *Modèle du canal à temps discret*

La figure 2.3 est définie comme le modèle du canal vectoriel. Le problème de la transmission d'une fonction du temps à toutes les T secondes revient donc à celui de la transmission d'un vecteur N dimensionnel reçu sous une forme corrompue par un

vecteur aléatoire Gaussien à N dimensions. Nous verrons à la fin de ce chapitre que la modélisation du canal à temps discret a permis d'établir les fondements du codage aléatoire de Shannon.

2.2.2 Capacité du canal

La formule de la capacité du canal, pour un canal Gaussien, est sans doute la formule, la plus importante en télécommunication. C'est une borne supérieure sur le débit d'information envoyé à travers le canal qui garantit une probabilité d'erreur qui tend vers 0. Afin de définir cette borne, il est essentiel de présenter en premier lieu, la notion d'efficacité spectrale et de taux de transmission. La figure 2.4 représente le codeur qui transforme K bits d'information en N symboles codés, où $N \geq K$.

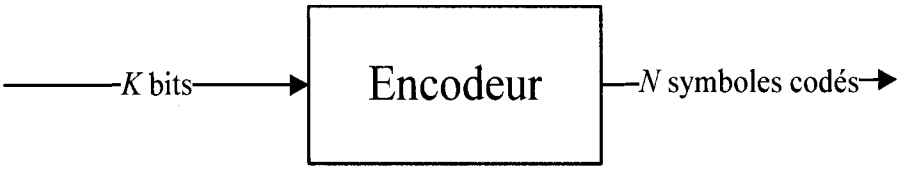

Figure 2.4: *Modèle de l'encodeur*

Le taux de transmission définit comme R est le nombre de bits par seconde délivrés par la source. L'efficacité spectrale est le rapport entre le taux de transmission et la largeur de bande :

$$\rho = \frac{R}{W} \ (\text{bits/s./Hz}) \tag{2.6}$$

Sous l'hypothèse de l'échantillonnage au taux de Nyquist, nous pouvons envoyer jusqu'à $2W$ symboles par seconde. Le rapport entre l'efficacité spectrale du canal à

temps continu et celui à temps discret devient alors immédiat. Effectivement, le nombre de bits par seconde par Hertz envoyé sur le canal devient le nombre de bits par symbole complexe.

$$\frac{K}{N} = \frac{R}{2W} = \frac{\rho}{2} \Rightarrow \rho = \frac{2K}{N} \tag{2.7}$$

Encore une fois, l'analogie entre le canal à temps continu et celui à temps discret est établie. L'efficacité spectrale qui est une mesure qui prend en compte la largeur de bande est maintenant facile à obtenir. Par exemple, dans le cas d'une transmission 2-PAM sans codage, l'efficacité spectrale serait 2 puisque nous pouvons envoyer un bit par dimension, ou 8 s'il s'agissait de 256-QAM.

La seule contrainte évoquée jusqu'à maintenant était l'échantillonnage. Pour les applications spatiales par exemple, malgré que le canal dispose de beaucoup de fréquence, nous ne pouvons pas augmenter $2W$ symboles par seconde autant que nous le souhaitons. La probabilité d'erreur du système doit être contrôlée. La formule de la capacité du canal de Shannon, qui est une borne sur le taux transmission ou sur l'efficacité spectrale dans un canal à bruit blanc et Gaussien s'exprime à l'aide de la relation suivante [1]:

$$\begin{cases} R \le C = W \log_2(1+SNR) \text{ (bits/s.)} \\ \rho \le \rho^c = \log_2(1+SNR) \text{ (bits/s./Hz)} \end{cases} \tag{2.8}$$

Ces relations indiquent qu'il est possible théoriquement, de transmettre l'information à un taux de transmission égal à C, avec une probabilité d'erreur aussi faible que désirée, en utilisant un code correcteur d'erreur approprié. Mais inversement, si le débit de transmission est supérieur à cette capacité C alors il sera impossible d'assurer que la

probabilité d'erreur tende vers 0. L'efficacité spectrale et le taux de codage sont équivalents à un facteur de 2 prés. Le taux de codage est le nombre de bits d'information par symbole codé *K/N* envoyés dans le canal alors que l'efficacité spectrale est égale à *2K/N*. Pour un régime limité en puissance, le taux de codage sera inférieur à 1 et l'efficacité spectrale inférieure à 2. À partir de la relation (2.9), nous pouvons déterminer le rapport signal à bruit minimal $(E_b / N_0)_{\min}$ à partir duquel il est possible de transmettre l'information de manière fiable. Le tableau ci-dessous fournit ce rapport pour plusieurs taux de codage différents.

Tableau 2.1 : Limite des E_b / N_0 pour plusieurs taux de codage.

Taux de codage R	$R \to 0$	1/4	1/3	1/2	2/3	3/4	1
ρ (bits/s./Hertz)	$\rho \to 0$	2/4	2/3	1	4/3	6/4	2
$(E_b / N_0)_{\min}$ (dB)	-1,59	-0,82	-0,55	0	0,569	0,86	1,76

Malgré le fait que théoriquement, il soit possible de transmettre l'information aux valeurs E_b / N_0 minimaux illustrés, ce n'est que très récemment que l'on a découvert des systèmes de codage capables de pouvoir transmettre l'information à des rapports signal à bruit aussi faibles. Shannon, en établissant ces relations a suscité un véritable défi pour des générations de chercheurs qui l'ont suivi. Pendant plus de 50 ans, mathématiciens et ingénieurs ont cherché sans grand succès des codes et des techniques de décodage pratiques qui pouvaient se rapprocher de ces limites.

Après avoir illustré les paramètres essentiels à temps discret et les limites pour une transmission fiable, nous finirons le chapitre 2 en illustrant la notion de codage aléatoire de Shannon.

2.3 Codage aléatoire

Les travaux de Shannon, ont permis de montrer que pour transmettre de manière efficace l'information à un taux se rapprochant de la capacité, il fallait coder l'information de manière pseudo aléatoire. Par aléatoire, nous entendons simplement que le code utilisé provient d'un ensemble de codes, dont chaque symbole est choisi de manière indépendante et aléatoire.

2.3.1 Ensembles aléatoires et décodage optimal

Dans le but d'illustrer l'efficacité du codage aléatoire. Nous allons, montrer comment il est possible d'un point de vue théorique, de transmettre l'information par le biais du codage, à un taux aussi proche que possible de la limite ultime de Shannon. Pour cela, nous devons utiliser la théorie des ensembles aléatoires, le décodage à maximum de vraisemblance, la borne de Chernoff, la loi faible des grands nombres [Annexe I], ainsi que l'un des résultats de la théorie des grandes déviations [56]. Mais avant tout, il est indispensable de recourir à l'un des résultats de la section précédente qui permet de modéliser un système de communication numérique de manière entièrement discrète.

Pour reprendre le modèle précédent, et sans perte de généralité, $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$ représente le vecteur à la sortie du canal. Le taux de transmission est de $2W$ symboles par seconde et l'énergie moyenne allouable par dimension est $S_x = P/2W$. L'ensemble de codes aléatoires de Shannon est défini comme suit. Chaque symbole $c_i, i = 1,2,....N$ de chaque mot de code $\boldsymbol{c} \in C$ est choisi de manière indépendante et aléatoire parmi un ensemble de codes suivant une distribution Gaussienne de moyenne 0 et de variance S_x, où l'énergie moyenne par symbole (dimension) sur l'ensemble des codes est aussi S_x. En appliquant la loi des grands nombres, l'énergie moyenne par symbole sur n'importe quels codes en

particulier est également S_x. Nous pouvons à présent considérer la probabilité d'erreur selon plusieurs scénarios. Un code C est choisi aléatoirement parmi l'ensemble décrit ci-dessus et ensuite un mot de code particulier $\boldsymbol{c}_0$ est désigné à la transmission. Le canal ajoute un bruit Gaussien avec une moyenne 0 et une variance $S_n = N_0/2$ par symbole. Le récepteur connaît le code C utilisé et reçoit $\boldsymbol{y} = \boldsymbol{c_0} + \boldsymbol{n}$. Le décodeur à maximum de vraisemblance implémente une règle de décision simple. S'il y a un seul mot de code $\boldsymbol{c} \in C$ espacé d'une distance au carré $\|\boldsymbol{y}-\boldsymbol{c}\|^2$ égale à $N \cdot (S_n \pm \varepsilon)$ où N est le nombre de symboles codés, il décide $\boldsymbol{c}$ sinon il arrête le décodage. Une erreur de décision peut se produire seulement dans l'un des deux cas de figure :

1) La distance au carré $\|\boldsymbol{y}-\boldsymbol{c}_0\|^2$ entre $\boldsymbol{y}$ et le mot de code transmis $\boldsymbol{c}_0$ n'est pas dans l'intervalle $N \cdot (S_n \pm \varepsilon)$.
2) La distance au carré $\|\boldsymbol{y}-\boldsymbol{c}_j\|^2$ entre $\boldsymbol{y}$ et un autre mot de code $\boldsymbol{c}_j \neq \boldsymbol{c}_0$ est dans l'intervalle $N \cdot (S_n \pm \varepsilon)$.

En appliquant la borne de Chernoff et la loi faible des grands nombres [Annexe I], la probabilité que le premier type d'erreur se produise tend vers zéro de manière exponentielle, lorsque N tend vers l'infini.

Définissons à présent comme *typique* le mot de code $\boldsymbol{c}_j \neq \boldsymbol{c}_0$ dont la distance au carré $\|\boldsymbol{y}-\boldsymbol{c}_j\|^2$ est égale à $N \cdot (S_n \pm \varepsilon)$. Basé sur l'un des résultats de la théorie des grandes déviations [56], la probabilité qu'un mot de code $\boldsymbol{c}_j \neq \boldsymbol{c}_0$ soit *typique* tend vers 0 lorsque e^{-NE} tend vers 0.

L'exposant E est donné à un terme prés d'ordre ε par l'entropie relative (Kullback-Leibler divergence) [60].

$$D(p_{XY} \square p_X p_Y) = \int_{-\infty}^{+\infty} dxdy p_{XY}(x,y) \log \frac{p_{XY}(x,y)}{p_X(x) p_Y(y)} \tag{2.9}$$

où les distributions $p_X(x)$, $p_Y(y)$ sont les densités de probabilité respectives de la séquence codée et de la séquence reçue à la sortie du canal, et $p_{XY}(x,y)$ représente la distribution conjointe entre la séquence codée et la séquence reçue. Dans le cas où la base du logarithme est 2, l'entropie relative est égale à l'information mutuelle $I(X;Y)$: [60].

$$\begin{aligned} I(X;Y) &= E_{XY}\left\{-\frac{1}{2}\log_2 2\pi S_n - \frac{(y-x)^2 \log_2 e}{2S_n} + \frac{1}{2}\log_2 2\pi S_y + \frac{y^2 \log_2 e}{2S_y}\right\} \\ &= \frac{1}{2}\log_2 \frac{S_y}{S_n} \text{ bits} \end{aligned} \tag{2.10}$$

où $S_y = S_x + S_n$. D'après le développement précédent la probabilité qu'un mot de code $\boldsymbol{c}_j \neq \boldsymbol{c}_0$ soit *typique* tend vers 0 lorsque $2^{-N(I(X;Y)-\delta(\varepsilon))}$ tend vers 0 où $\delta(\varepsilon)$ tend vers 0 à mesure que ε tende vers 0. Il est possible à présent d'en déduire, d'après la relation (2.10) et la borne union que la probabilité de n'importe lequel des $M-1 < 2^{\rho N/2}$ mots de code $\boldsymbol{c}_j \neq \boldsymbol{c}_0$ soit *typique* est donnée par :

$$P\left\{2^{\text{em}} \text{ type d'erreur}\right\} < 2^{\rho N/2} 2^{-N(\frac{1}{2}\log_2 \frac{S_y}{S_n} - \delta(\varepsilon))} \tag{2.11}$$

Par conséquent, afin que la probabilité des deux types d'erreurs tende vers zéro à mesure que N augmente il faut s'assurer que :

$$\rho < \frac{2}{2}\log\frac{S_y}{S_n} = \log(1+\mathrm{SNR}) \tag{2.12}$$

Cette analyse démontre qu'en dessous de la limite $\log(1+\mathrm{SNR})$, qui est la limite de Shannon sur l'efficacité spectrale (2.8), une probabilité d'erreur aussi faible que possible peut être atteinte, à condition d'utiliser les ensembles aléatoires, un code de grande dimension et le décodage à maximum de vraisemblance.[2]

L'analyse effectuée montre que sur l'ensemble des codes aléatoires de Shannon, il est possible d'atteindre la capacité. Cependant, afin de montrer qu'un code en particulier est capable d'atteindre la capacité, il faut observer que si en moyenne, sur l'ensemble des codes il est possible d'atteindre une probabilité d'erreur aussi petite soit elle, il y a au moins un code de l'ensemble qui peut atteindre une performance similaire. Mais il est intéressant aussi de considérer les autres codes de l'ensemble. Notons P_e la probabilité d'erreur moyenne, pas plus que $1/K$ des codes de l'ensemble ont une probabilité d'erreur supérieure à KP_e. Ce qui signifie qu'au moins 99% des codes de l'ensemble ont une probabilité d'erreur inférieure à $100P_e$. Étant donné que P_e est sensé être arbitrairement faible, $100P_e$ reste une très bonne performance.

[2] Ce développement s'inspire d'un des développements du cours en ligne « Principles of Digital Communication II », qui participe au programme « MIT Open Course Ware» donné par le Professeur Forney.

2.4 Conclusion

Après avoir introduit les bases de la théorie du codage, nous avons présenté la notion de capacité du canal. Nous avons vu qu'il est possible, en théorie, de transmettre l'information de manière optimale, en utilisant les ensembles aléatoires et le décodage à maximum de vraisemblance. Malheureusement, en pratique, il est difficile de concevoir des systèmes réels capables d'atteindre de telles performances. En effet, la complexité de calculs et la latence au décodage sont des exemples de phénomènes que l'on ne peut négliger. Le prochain chapitre présentera les systèmes de codage les plus populaires et essayera de faire le lien entre la théorie et la pratique, en montrant comment, les systèmes de codage les plus performants s'inspirent de l'approche probabiliste de Shannon.

CHAPITRE 3

PRINCIPALES TECHNIQUES DE CODAGE

3.1 Introduction

Dans ce chapitre, on présente les principales techniques de codage. Comme mentionné précédemment, il est possible de les diviser en trois catégories : les codes en blocs (codage algébrique), les codes convolutionnels et les codes en graphes. Historiquement, les premiers codes développés ont été les codes en blocs. Ces codes n'ont jamais réussi à se rapprocher de manière pratique et significative de la limite de Shannon [10]. Ils ne sont utilisés en pratique que pour des applications très spécifiques [21]. Néanmoins, ils représentent les bases du codage de canal et sont utiles à la compréhension de ce mémoire. C'est la raison pour laquelle nous en donnerons un aperçu général. Les codes convolutionnels avec décodage de Viterbi et les codes en graphes sont, quant à eux, utilisés dans diverses applications telles que les transmissions spatiales [22] ou la téléphonie mobile [8]. Ces codes sont étroitement liés à la nouvelle classe de codes proposée dans notre travail de recherche. Par conséquent, ils constituent la partie essentielle de ce chapitre.

3.2 Codes en blocs linéaires

3.2.1 Définition

Un code binaire, linéaire en blocs C, transforme un bloc d'information de K bits en un bloc de N symboles ; $N > K$. Le code est donc constitué de 2^K mots de code ayant chacun N symboles. Pour pouvoir assurer la linéarité de ce dernier, les éléments qui le constituent doivent former un espace vectoriel sous $\mathbb{F}_2$. Il est possible de prouver que les seules conditions nécessaires à la linéarité d'un code binaire sont que n'importe quel

ensemble sur les 2^K mots de code aie la propriété de fermeture, sous l'addition modulo-2 élément par élément de chaque symbole, et que le mot de code $(0,0,0,.....0)_N$ fasse partie du code.

$$\begin{cases} c_i \oplus c_j = c_e \text{ où } \{c_i, c_j, c_e\} \in C \\ (000.....0)_N \in C \end{cases} \tag{3.1}$$

Afin de compléter cette définition, les notions de distance et de poids de Hamming doivent être définies.

- Le poids de Hamming $W_H(\boldsymbol{x})$ d'un mot de code $\boldsymbol{x}$ de longueur N, est le nombre de composantes non nulles dans $\boldsymbol{x}$.
- La distance de Hamming $d_H(\boldsymbol{x}, \boldsymbol{y})$ entre deux mots de code $\boldsymbol{x}$ et $\boldsymbol{y}$ est le nombre de positions où les deux mots de code diffèrent.

Prenons pour exemple, les mots de code $\boldsymbol{x} = (010101011)$ et $\boldsymbol{y} = (111001111)$, les poids de Hamming sont $W_H(\boldsymbol{x}) = 5$ et $W_H(\boldsymbol{x}) = 7$ et la distance de Hamming $d_H(\boldsymbol{x},\boldsymbol{y})$ est égale à 4.

Nous pouvons ainsi représenter un code par le triplet (N, K, d) où N est le nombre de symboles codés, K le nombre de bits d'information et d la distance minimale. La distance minimale correspond à la plus petite distance de Hamming entre n'importe quels mots de code. Il est simple de prouver que puisque le code est linéaire, d équivaut au mot de code $\boldsymbol{z}$ n'incluant pas $(0,0,0,.....0)_N$, ayant un poids de Hamming minimal.

$$W_H(\boldsymbol{z}) = d = \min_{\boldsymbol{x} \neq \boldsymbol{y} \in C} d_H(\boldsymbol{x}, \boldsymbol{y}) \tag{3.2}$$

3.2.2 Principales caractéristiques des codes en blocs

Il est possible de définir un code linéaire, en utilisant uniquement leurs fonctions génératrices.

Illustrons comme exemple le code $(3,2,2)$ constitué de quatre mots de code $\{000,\ 011,\ 110,\ 101\}$. Les deux fonctions génératrices sont $g_1 = 101$ et $g_2 = 110$. En effectuant toutes les combinaisons linéaires $\{\alpha_1 g_1 + \alpha_2 g_2 \ | \alpha_1, \alpha_2 \in \mathbb{F}_2\}$, on obtient le code dans son intégralité.

Les fonctions génératrices sont représentées par les mots de code linéairement indépendants entre eux. Une manière simple de les construire est de choisir n'importe quel mot de code $\boldsymbol{c}_1$ différent de $(0,0,0,.....0)_N$ comme première fonction génératrice, puis un autre mot de code $\boldsymbol{c}_2$ linéairement indépendant de $\boldsymbol{c}_1$ et ainsi de suite. L'opération générale d'encodage peut être représentée par un ensemble de N équations :

$$\boldsymbol{c}_{m,j} = \sum_{l=0}^{K-1} \alpha_{m,l} g_{l,j} \text{ avec } g_{i,j} \in \mathbb{F}_2 \text{ et } j=0,1.....,N-1 \tag{3.3}$$

ou de manière plus simple, sous forme matricielle :

$$C = \left\{ \boldsymbol{uG} \text{ avec } \boldsymbol{u} \in \{\mathbb{F}_2\}^K \right\} \tag{3.4}$$

où $\boldsymbol{u}$ sont les bits d'informations et $\boldsymbol{G}$ la matrice génératrice de dimension $(K \times N)$:

$$\boldsymbol{G} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,N-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ g_{K-1,0} & g_{K-1,1} & \cdots & g_{K-1,N-1} \end{bmatrix} \tag{3.5}$$

Une deuxième représentation possible du code, sous forme de contrôle de parité est donnée par le système d'équations suivant :

$$C = \left\{ \boldsymbol{y} \text{ tel que } \boldsymbol{y}\boldsymbol{H}^T = 0 \right\} \tag{3.6}$$

où $\boldsymbol{H}$ est la matrice de parité. Cette matrice est de dimension $(N-K \times N)$:

$$\boldsymbol{H} = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,N-1} \\ h_{1,0} & h_{1,1} & \cdots & h_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ h_{N-K,0} & h_{N-K,1} & \cdots & h_{N-K,N-1} \end{bmatrix} \tag{3.7}$$

La majeure partie des codes linéaires peuvent être définis ainsi. Les meilleurs codes en blocs ont deux caractéristiques principales ; ils font en sorte de maximiser la distance minimale et contiennent assez de structure pour pouvoir être décodés avec une complexité raisonnable. En guise d'exemple, nous présentons les codes de Reed-Muller [13]. Nous essayons ensuite, de manière intuitive de comprendre pourquoi, ce genre de codage ne fut pas l'instrument idéal pour se rapprocher de la capacité [10].

3.2.3 Codes de Reed Muller

Les codes de Reed Muller, dénotés codes RM sont une famille de codes (N,K,d) qui présente un bon compromis entre performance et complexité. Les codes RM(r,m) sont caractérisés par les paramètres r et m , $0 \le r \le m$, tels que $N = 2^m$ et $d = 2^{m-r}$. Par

exemple le code $\mathrm{RM}(0,m)$ est le code binaire à répétition constitué uniquement de deux mots de code $(0,0,....,0)_N$ et $(1,1,.....,1)_N$ et $\mathrm{RM}(m,m)$ est le code où K=m=N, qui représente toutes les 2^m combinaisons possibles. La construction des codes Reed Muller s'effectue par récursivité. $\mathrm{RM}(r,m)$ provient de $\mathrm{RM}(r-1,m-1)$ et $\mathrm{RM}(r,m-1)$ suivant la règle :

$$\mathrm{RM}(r,m)=\left\{ (\boldsymbol{u},\boldsymbol{u}+\boldsymbol{v}) \middle|\, \boldsymbol{u}\in \mathrm{RM}(r,m-1),\ \boldsymbol{v}\in \mathrm{RM}(r-1,m-1)\right\} \tag{3.8}$$

Le décodeur associé à cette classe de code a été introduit par Reed [23]. Il repose sur un simple décodeur à quantification dure qui était attrayant pour la technologie des années cinquante.

Nous avons montré dans la section 2.3.1 que le théorème de la capacité du codage de Shannon a été prouvé grâce aux ensembles aléatoires et au décodage optimal, lorsque la longueur du code tendait vers l'infini. La relation (3.8) témoigne de la structure inhérente à la construction des codes de Reed Muller. Ce qui présente un inconvénient, car cette construction ne s'inspire pas réellement de l'approche probabiliste, où tous les symboles de chaque mot de code sont choisis de manière aléatoire. Le deuxième point faible est lié à la dimension finie ainsi qu'a la complexité du décodage qui augmente de manière exponentielle avec la longueur du code N. Nous montrons dans la prochaine section, comment, il est possible d'encoder l'information avec des codes de longueur infinie, où la complexité ne dépend pas de la longueur du code, mais tout en ayant la possibilité de les décoder avec un décodage à maximum de vraisemblance [3].

3.3 Codes convolutionnels

Dans cette section, le fonctionnement et la structure des codes convolutionnels sont illustrés. Nous nous intéressons aux codes systématiques ainsi qu'aux codes non systématiques et enfin aux codes récursifs. En termes de décodage, nous présentons exclusivement l'algorithme de Viterbi, étant donné qu'il est optimal et qu'il représente la base du décodage développé dans ce projet.

3.3.1 Bref historique

Les codes convolutionnels ont été inventés en 1955 au MIT (Massachussetts Institute of Technology) par Elias [2]. Ces codes ont été mis au point dans le but, entre autres, de créer des codes de longueur dite *infinie* contenant assez de structure, pour pouvoir être décodés en durée de temps polynomiale. En 1961 Wozenscraft et Reiffen ont introduit le premier décodeur pratique capable de décoder les codes convolutionnels appelé décodage séquentiel [24]. En 1963, Massey a proposé le décodage à seuil pour les codes en blocs ainsi que les codes convolutionnels [25]. Cet algorithme utilise les équations de parité et est optimal pour plusieurs sortes de codes en blocs, mais reste sous optimal quand on l'applique aux codes convolutionnels. Quelques années plus tard, Fano et Jelinek [26-27] ont introduit un algorithme de décodage séquentiel modifié qui améliore les performances de l'algorithme de Wozenscraft et Reiffen. Cependant, cet algorithme demeure encore sous optimal. Viterbi a introduit en 1967 une troisième sorte de décodage pour les codes convolutionnels, et démontra que son algorithme était asymptotiquement optimal [3]. Ce n'est que cinq ans plus tard que Forney prouva que l'algorithme de Viterbi était en fait un décodage à maximum de vraisemblance donc optimal, pour les codes convolutionnels [9].

3.3.2 Principe d'encodage

Un codeur convolutionnel de taux $1/n$ bit par symbole peut être représenté comme une machine linéaire à nombre d'états finis consistant en un registre à décalage de K cellules et n additionneurs modulo-2, connectés à certaines cellules du registre. La figure 3.1 illustre le schéma de ce type de codeur convolutionnel. L'encodage est le mécanisme qui consiste à passer la séquence d'information $\boldsymbol{u}$ dans le registre à décalage un bit à la fois et en générer chacun des n symboles codés à la sortie de chaque additionneur modulo-2.

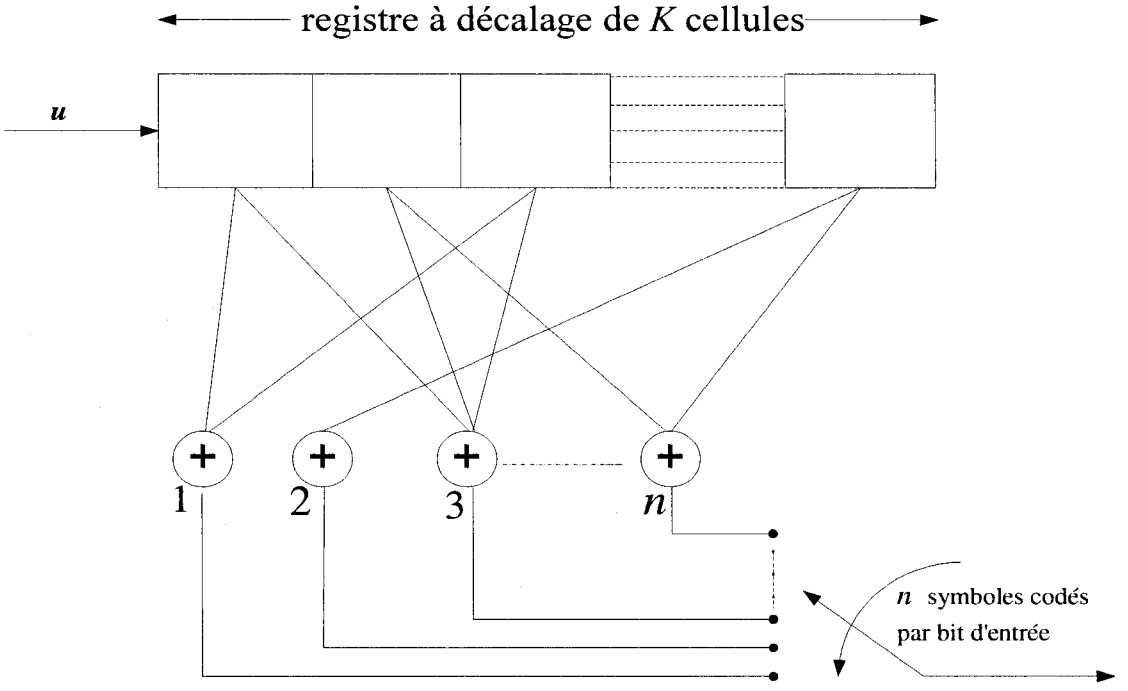

Figure 3.1 : *Schéma d'un codeur convolutionnel de taux 1/n à l'aide d'un registre à décalage*

Il est possible de décrire le code à l'aide du vecteur, appelé vecteur des connexions $\boldsymbol{G}_j = (g_{j1}, g_{j2}, g_{j3}, \ldots\ldots\ldots g_{jK}),\ j = 1, 2, \ldots . n$ qui exprime la présence ou l'absence de

connexions entre l'additionneur j et les K cellules du registre. Une cellule est connectée à un additionneur dans le cas d'une valeur 1 et ne l'est pas si la valeur est 0. On nomme K la longueur de contrainte. La mémoire du code pour un code de taux de codage $1/n$ bit par symbole est égale à $M = K - 1$.

La figure 3.2 présente une autre manière d'illustrer le même codeur que celui de la figure 3.1. Dans ce mémoire nous utiliserons, en fonction du contexte l'une ou l'autre de ces deux représentations.

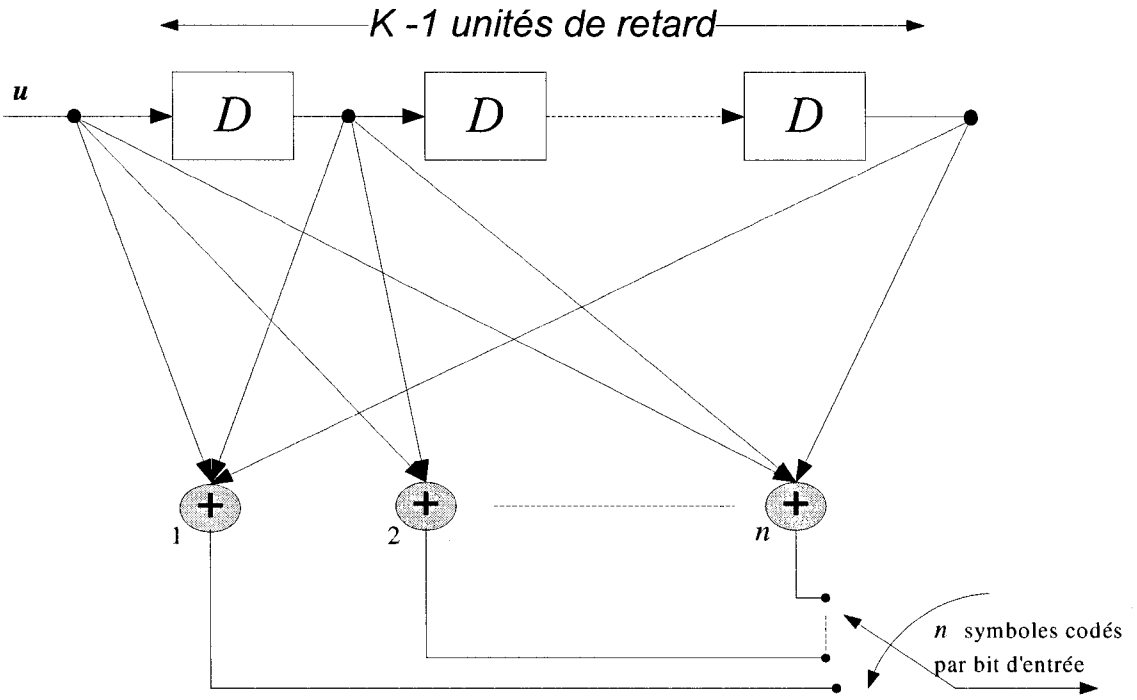

Figure 3.2: *Schéma d'un codeur convolutionnel de taux 1/n à l'aide d'unités de retard*

Dans l'objectif de mieux conceptualiser les codes convolutionnels, il est intéressant de définir la matrice génératrice du code. Pour cela, il est préférable de réécrire le vecteur

des connexions sous une forme un peu différente $\boldsymbol{G}_i^* = (g_{1i}, g_{2i}, g_{3i}, ...g_{ni})$, $i = 1,2,....K$,[3] (schéma 3.1) qui exprime la présence ou l'absence de connexions entre la cellule du registre i et les n additionneur .où chaque composante du vecteur indique la connexion de tous les additionneurs avec une cellule particulière et non la connexion de toutes les cellules avec un additionneur particulier comme ce qui est le cas dans la notation classique :

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_1^* & \boldsymbol{G}_2^* & \cdots & \boldsymbol{G}_n^* & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{G}_1^* & \boldsymbol{G}_2^* & \cdots & \boldsymbol{G}_n^* & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{G}_1^* & \boldsymbol{G}_2^* & \cdots & \boldsymbol{G}_n^* & 0 & 0 \\ \vdots & 0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & 0 & 0 & \ddots & \ddots & \ddots & \ddots \\ \vdots & \vdots & \vdots & 0 & 0 & \ddots & \ddots & \ddots \\ \vdots & \vdots & \vdots & \vdots & 0 & 0 & \ddots & \ddots \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & \ddots \end{bmatrix} \tag{3.9}$$

Le vecteur $\boldsymbol{c}^u = (c_1^u, c_2^u, ...c_n^u)$ des sorties codées correspondantes à l'entrée $\boldsymbol{u}$, est obtenu par le produit matriciel modulo-2 de $\boldsymbol{u} = (u_1, u_2, ...u_k)$ qui désigne le vecteur des entrées au registre à décalage avec la matrice génératrice $\boldsymbol{G}$.

$$\boldsymbol{c}^u = \boldsymbol{u}\boldsymbol{G} \tag{3.10}$$

Selon la définition de Forney [28] les polynômes générateurs $\boldsymbol{G}_j = (g_{j1}, g_{j2}, g_{j3}, ...g_{jK})$, $j = 1,2,....n$ sont donnés par la relation suivante :

$$G_j(D) = \sum_{i=0}^{K-1} g_{ji} D^i \tag{3.11}$$

[3] Le symbole * est une notation, il ne représente ni la transposé ni la complexe conjugué du vecteur.

En associant les séquences en D qui est une variable temporelle appelée unité de retard, le code est défini suivant la relation :

$$C = \{u(D)g(D) \mid u(D) \in \mathbb{F}_2(D)\} \tag{3.12}$$

où $g(D) = \{G_j(D),\ j = 1, 2.....n\}$ et $u(D)$ équivaut à l'information $\boldsymbol{u}$ sous forme polynomiale. $g(D)$ dénoté également $G(D)$ en fonction des éléments de retard D fait souvent référence à la matrice génératrice sous forme polynomiale du code.

Prenons pour exemple le codeur de la figure 3.3 :

$$\boldsymbol{G}_1 = [1111]\ ,\ \boldsymbol{G}_2 = [1101] \tag{3.13}$$

Le code qu'il génère peut être entièrement défini sous forme polynomiale $g(D) = [1 + D + D^2 + D^3, 1 + D + D^3]$.

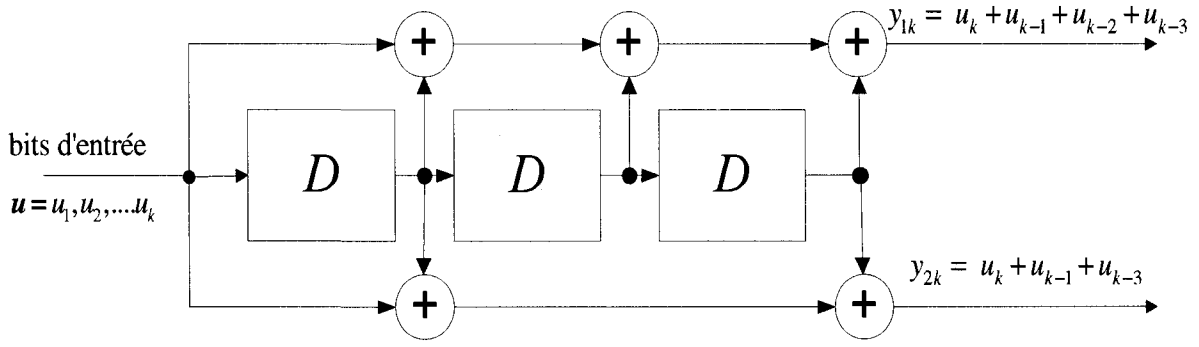

Figure 3.3: *Codeur convolutionnel de taux 1/2, n=2, M=3 ayant* $g(D) = [1 + D + D^2 + D^3, 1 + D + D^3]$

Le codeur convolutionnel de taux *1/n* peut être généralisé en un codeur convolutionnel de taux $b/n, (b \geq 1)$ où le registre à K cellules est alors remplacé par un ensemble de b registres à $b \times K$ cellules en parallèle. Chacun des registres peut être perçu comme un codeur convolutionnel binaire défini par ses propres vecteurs de connexion, donc peuvent être différents. Dans ce cas il est plus commode de parler de la mémoire de codeur et non de sa longueur de contrainte.

3.3.3 Représentations graphiques

Un code convolutionnel peut être représenté de plusieurs façons : diagramme d'état, arbre ou encore par le biais d'un treillis d'encodage.

3.3.3.1 Diagramme d'état

Le codeur convolutionnel est une machine à états finis, il peut donc être caractérisé par un diagramme d'état. Comme mentionné précédemment, la sortie du codeur à un instant donné dépend du bit d'information entré dans le codeur et de l'état du codeur, soit le contenu des *K-1* dernières cellules du registre. La figure 3.4 décrit le diagramme d'état associé au codeur de la figure 3.3. Les états sont représentés par des cercles, les transitions du codeur d'un état à l'autre sont définies comme les branches reliant les différents états. Le premier chiffre qui se trouve sur les branches indique le bit entrant dans le codeur et les deux autres chiffres qui suivent la barre oblique désignent les symboles codés. Le diagramme d'état est un outil pratique pour la caractérisation des codes, il ne permet cependant pas de suivre leurs évolutions dynamiques dans le temps. Pour pallier à cet inconvénient, deux représentations plus complètes sont alors utilisées : la représentation en arbre et la représentation en treillis. Ces dernières fournissent une information temporelle essentielle au suivi de l'évolution du code dans le temps. Nous pourrons constater dans le chapitre suivant, lorsque nous traiterons de la nouvelle classe de codes introduite dans ce travail de recherche, que le diagramme d'état est inadapté et

que seules les représentations sous forme d'arbre ou de treillis peuvent caractériser les codes convolutionnels à temps variant.

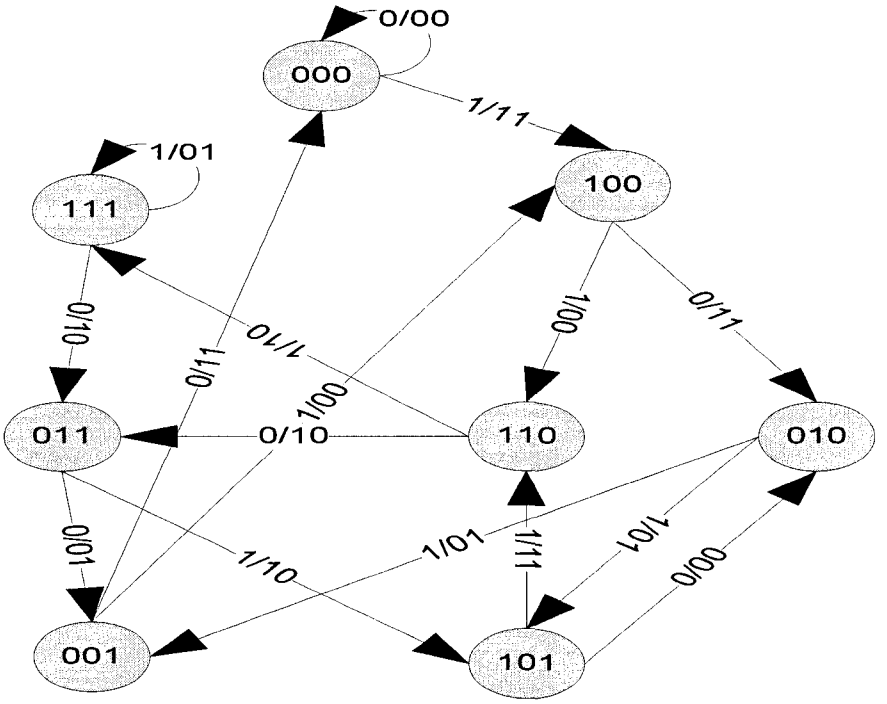

Figure 3.4: *Diagramme d'état associé au codeur convolutionnel de la figure 3.3*

3.3.3.2 Représentation en arbre

Cette représentation de l'opération du codage est plus complète, car elle prend en compte l'évolution du code dans le temps. Les transitions entre les états du code sont illustrées comme un cheminement à partir d'un état à un autre, où l'état de départ est souvent choisi comme étant zéro. Un arbre se compose de nœuds et de branches. Dans le cas binaire, deux branches sortent de chaque nœud. Chaque nœud illustre un état du codeur, la branche supérieure correspond à un bit '0' d'information qui entre dans le

codeur et la branche inférieure à un bit d'information '1', les symboles codés à la sortie du codeur sont indiqués sur chacune des branches. La figure 3.5 illustre l'arbre correspondant au codeur de la figure 3.3 où les $S_j, j = 1,2.....8$ représentent les huit états possibles du codeur.

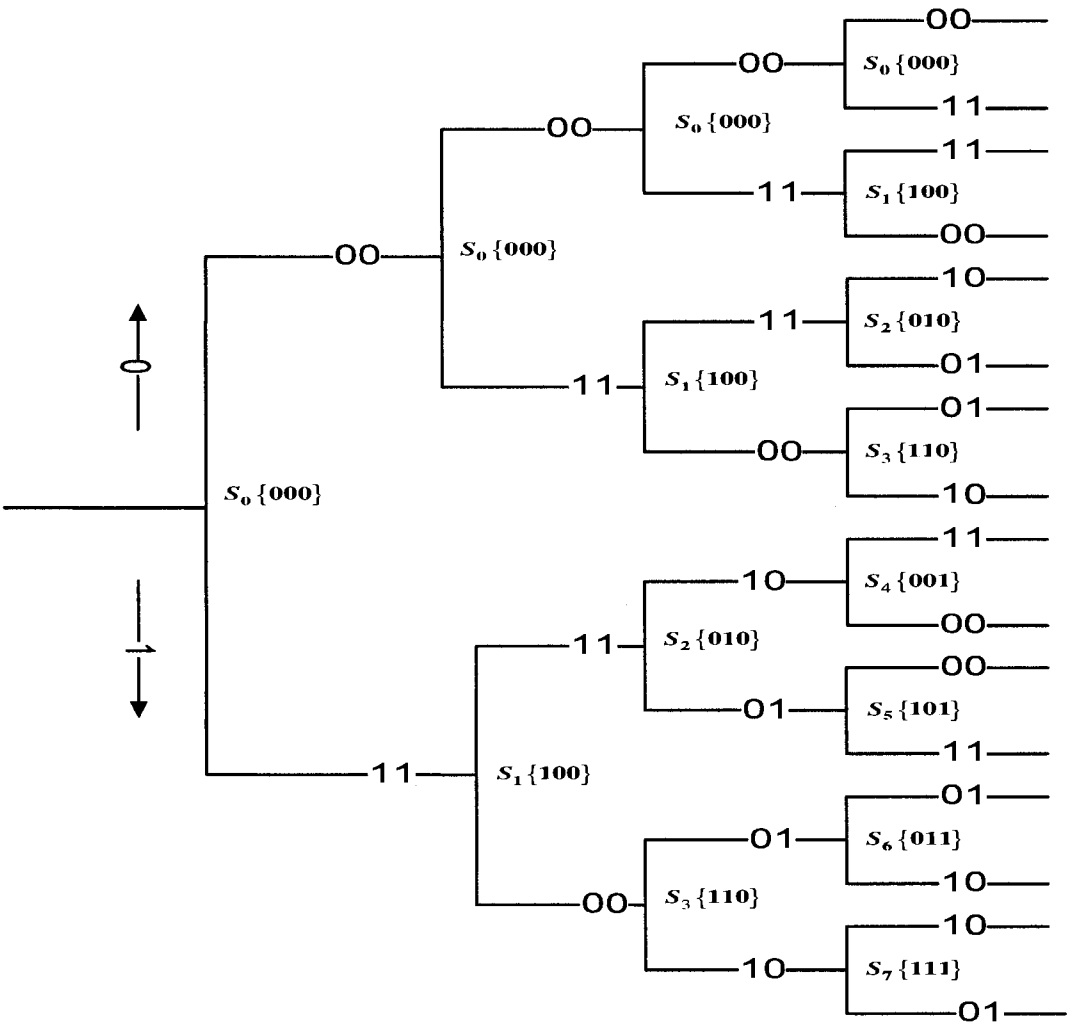

Figure 3.5: *Arbre associé au codeur convolutionnel de la figure 3.3*

3.3.3.3 Représentation en treillis

Le treillis est un outil important, il permet de décrire l'évolution temporelle du code de manière plus compacte que dans la représentation sous forme d'arbre. Effectivement, le treillis offre l'avantage d'unir les chemins passant par le même état. Dans les prochaines sections, nous verrons que le treillis est nécessaire à la compréhension du décodage de Viterbi

La figure 3.6 illustre le treillis correspondant au code convolutionnel de la figure 3.3. De manière similaire à la représentation sous forme de diagramme d'état, les états sont représentés par des cercles, les transitions du codeur d'un état à l'autre sont représentées par des branches reliant les différents états. Le premier chiffre indique le bit entrant dans le codeur et les deux autres chiffres qui suivent la barre oblique désignent les symboles codés. Chaque chemin partant de l'état initial (zéro) et se prolongeant à travers le treillis représente un mot de code particulier. De cette manière, tous les mots de code sont représentés par l'ensemble des chemins possibles traversant le treillis. Les codes convolutionnels sont par définition linéaires, puisque la première partie de la relation (3.1) s'applique entre chaque colonne d'états et s'étend naturellement sur tout le treillis. Si la séquence d'information $\boldsymbol{u}$ qui entre dans le codeur est une séquence finie, nous parlons de codes convolutionnels de longueur finie. Nous verrons que ces notions sont importantes dans le décodage de Viterbi.

Si deux séquences d'information sont identiques sauf sur n bits consécutifs, alors les séquences codées correspondantes seront distinctes sur $(n+K-1)$ branches consécutives. Dans le cas où deux chemins différents ont $(K-1)$ bits d'information consécutifs identiques, alors ces deux arbres reconvergent et les sous-arbres émergeant du point de reconvergence sont identiques.

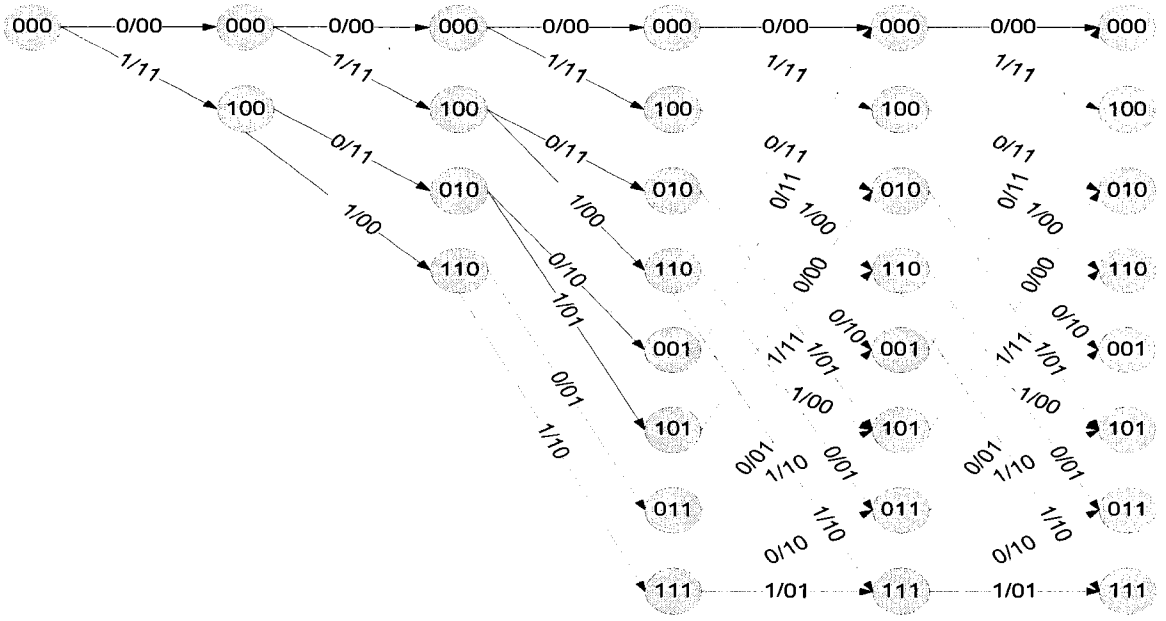

Figure 3.6: *Treillis associé au code convolutionnel de la figure 3.3*

. Plusieurs notions de distances peuvent être définies pour les codes convolutionnels.

- La *distance minimale* $d_{\min}$ est la distance minimale entre les chemins de deux mots de code de longueur K branches et qui ont leur première branche différente.

- la *distance des colonnes* d'ordre q $d_c(q)$ est la distance de Hamming minimale entre deux mots de code de longueur q branches qui ont leur première branche différente.

- $d_{\min}$ correspond aussi à la distance de colonne d'ordre q=K :

$$d_{\min} = d_c(K) \tag{3.14}$$

- La distance libre d_{free} quant à elle, est la limite de la distance de colonne lorsque q tend vers l'infini :

$$d_{free} = \lim_{q \to \infty} d_c(q) \tag{3.15}$$

Par définition :

$$d_{free} \geq d_{\min} \tag{3.16}$$

La distance libre d_{free} est la distance minimale entre deux mots de code de longueur infinie qui ont leur première branche différente. Sa détermination n'est pas systématique peut être quelque peu élaboré, généralement d_{free} est égale à 3 ou 4 fois la longueur de contrainte *K*.

3.3.4 Codes convolutionnels systématiques

Dans le cas d'un codeur convolutionnel systématique de taux *1/n* bit par dimension, les symboles codés à la sortie du codeur sont constitués d'un bit d'information et de *n-1* symboles de parité. La figure 3.7 illustre un tel code. Il est plus simple de décoder les codes systématiques. Plusieurs systèmes tels que le décodage à seuil de Massey [25] ou bien encore le décodage itératif développé par Cardinal, Haccoun et Gagnon [29], au laboratoire de communication avancée de l'École Polytechnique de Montréal sont adaptés aux codes systématiques. Les codeurs systématiques ont la propriété de ne pas être *catastrophiques*. Un codeur est *catastrophique* si et seulement si un nombre fini d'erreurs sur les symboles codés peut causer un nombre infini de bits d'information décodés en erreur. Les codeurs catastrophiques seront traités plus en détail dans le prochain chapitre. Les performances des codes systématiques sont en moyenne

inférieures à celles des codes convolutionnels non systématiques [28]. Une des façons de pallier à cet inconvénient est l'introduction des codes récursifs.

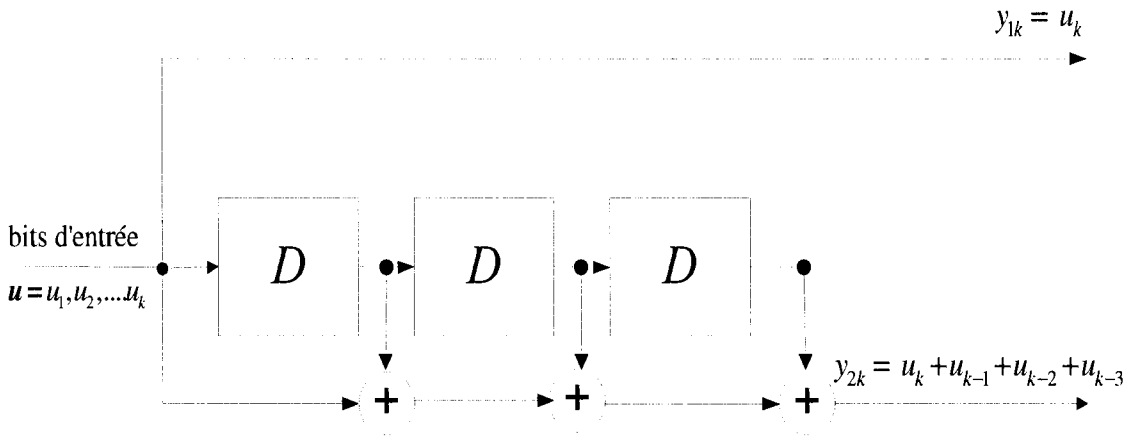

Figure 3.7: *codeur convolutionnel systématique K=4, M=3, R=1/2*

$$g(D) = [1, 1 + D + D^2 + D^3].$$

3.3.5 Codes convolutionnels systématiques récursifs

Les codes convolutionnels systématiques récursifs sont générés par un codeur convolutionnel systématique dont certaines connexions du registre sont effectuées par une boucle de retour. Les codes convolutionnels systématiques récursifs permettent de bénéficier des avantages des codes convolutionnels systématiques sans souffrir de leurs inconvénients. Afin d'éclaircir ce point, nous devons établir la notion de codes convolutionnels équivalents. Deux générateurs $g(D)$ et $g'(D)$ sont équivalents s'ils génèrent le même code, et deux générateurs sont équivalents si et seulement si [28] :

$$g(D) = u(D)g'(D) \mid u(D) \in \mathbb{F}_2(D),\ u(D) \neq 0 \qquad (3.17)$$

Cette relation permet de décrire un codeur systématique récursif équivalent à celui de la figure 3.3, en remplaçant le générateur $g'(D)=[1+D+D^2+D^3,1+D+D^3]$ par $g(D)=[1,\frac{1+D+D^3}{1+D+D^2+D^3}]$:

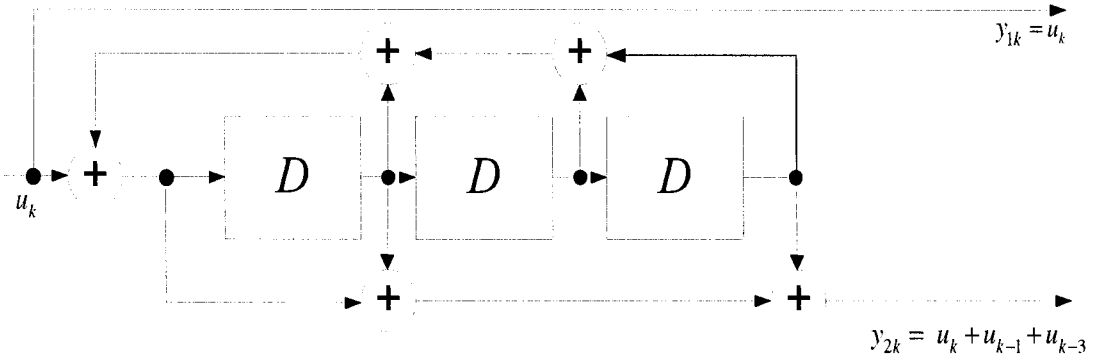

Figure 3.8 : codeur convolutionnel récursif $g(D)=[1,\frac{1+D+D^3}{1+D+D^2+D^3}]$

Un codeur convolutionnel récursif a la particularité de pouvoir générer un code de longueur infinie avec une séquence d'information finie. Prenons pour exemple le codeur convolutionnel de la figure 3.3. Si la séquence d'information $\{1101\}$ entre dans le codeur, le mot de code $\{..0000011100011000101 0000..\}$ sera généré. Cependant, si la même séquence d'information entre dans le codeur récursif de la figure 3.8, le mot de code généré sera $\{...0000111101110101000100010001000100010001.....\}$ avec une période $\{0001\}$.

3.3.6 Décodage de Viterbi

Considérons une séquence codée générée à l'aide d'un codeur convolutionnel. Cette séquence $y(D)=u(D)g(D)$ est transmise dans un canal Gaussien AWGN en assignant à chaque symbole codé $y_{jk}\in\{0,1\}$ la valeur $s(y_{jk})\in\{\pm\alpha\}$, $j=0,1$ et $k=0,1.....N$ par le biais de la modulation PAM. La séquence reçue au décodeur est :

$$r(D)=s(y(D))+n(D) \tag{3.18}$$

où $n(D)$ est le bruit Gaussien identiquement distribué avec moyenne nulle et une variance de $N_0/2$ par dimension. Le décodage à maximum de vraisemblance équivaut à trouver la séquence codée $y(D)$ qui minimise la métrique $\|r(D)-s(y(D))\|^2$ [9] pour toute les valeurs de $s(y(D))$. Puisque chaque chemin qui traverse le treillis correspond à une séquence codée possible, il est donc possible d'assigner la métrique $\|\boldsymbol{r}_k-s(\boldsymbol{y}_k)\|^2$ à chaque branche du treillis. Pour cela il faut simplement remplacer les valeurs $y_{jk}\in\{0,1\}$ par $s(y_{jk})\in\{\pm\alpha\}$ de chaque branche, en assigner à chaque instant la valeur reçue $\boldsymbol{r}_k$, d'en faire la soustraction puis l'élévation au carré. Une fois que le treillis incorpore ces données, l'algorithme de Viterbi consiste à trouver le chemin de poids minimal sur l'ensemble de tous les chemins du treillis. Une question majeure se pose alors : Comment est-il possible que la complexité du décodage n'augmente pas avec la longueur de la séquence codée ?

La réduction de calcul (complexité) est possible d'une part grâce à la structure spécifique du treillis et d'autre part grâce à la propriété de reconvergence que nous avons présenté dans la section 3.3.3.3. À chaque instant, l'algorithme garde les 2^M chemins de poids minimal qui arrivent à chaque nœud, ou M est la mémoire du codeur.

Ce mécanisme est défini par l'opération dite ACS (Add Compare Select). En effet, si à une profondeur donnée dans le treillis, la métrique aux nœuds j a une valeur Γ_j, l'algorithme calcule pour chaque branche partant de ce nœud, la somme $\Gamma_j + \gamma_i$ où γ_i est la métrique de branche (Add). Les différentes valeurs obtenues en un même nœud sont alors comparées (Compare) et seule la branche de la métrique minimale est ensuite conservée (Select). Ces métriques minimales conservées aux nœuds j permettent à chaque instant et pour chacun des 2^M états, de conserver un seul chemin appelé *survivant*. Les *survivants* permettent de conserver à chaque instant, l'information nécessaire à l'algorithme en seulement 2^M chemins différents du treillis au lieu de $2^{2.j}$. Ceci réduit considérablement la complexité et empêche son augmentation exponentielle avec la longueur de la séquence codée.

Pour conclure, l'algorithme de Viterbi est un décodage optimal avec une complexité de l'ordre de 2^M [9], et est utilisé dans plusieurs systèmes de communication actuels. Les codes convolutionnels avec décodage de Viterbi de longueur de contrainte inférieure à dix atteignent des performances supérieures à la majorité des codes en blocs pour une complexité équivalente. Cependant, en pratique, ils n'arrivent malheureusement pas à atteindre la limite de Shannon. Il est possible d'émettre une hypothèse sur les raisons de leurs bonnes, mais insuffisantes performances. Pour reprendre le développement sur le codage aléatoire de Shannon décrit dans le chapitre précédent. Les consignes théoriques à respecter pour l'obtention d'un système de codage optimal sont :

1) Codes de longueurs très grandes.
2) Décodage optimal à maximum de vraisemblance.
3) Ensembles aléatoires.

Les codes convolutionnels avec décodage de Viterbi satisfont aux deux premières conditions, mais malheureusement pas à la troisième, puisque le codeur génère un code suivant une règle bien précise, qui est l'addition modulo-2 répétitive des connexions fixes d'une machine à états finis. Les symboles des mots de code sont donc dépendants les uns des autres et manquent par conséquent d'aléa.

Il est cependant important de noter que théoriquement l'algorithme de Viterbi permet d'atteindre les performances dictées par Shannon énoncées dans le chapitre précédent. Néanmoins, que ce soit lors de l'introduction de l'algorithme de Viterbi [3] ou des codes convolutionnels [2], la condition théorique nécessaire à l'obtention de telles performances est de choisir un code parmi un ensemble de codes convolutionnels variant dans le temps sans périodicité.

La prochaine section traite des codes en graphes, ces codes à l'opposé des codes illustrés jusqu'à présent, possèdent une structure aléatoire.

3.4 Codes en graphes

Le codage probabiliste a vu le jour lors de l'introduction des codes Turbo à la conférence ICC de Genève en 1993 [4]. Ces codes sont capables de se rapprocher de manière significative de la limite de Shannon. Cette découverte eut un impact considérable, elle mit fin à la conjecture qui s'était installée au fil des années, à savoir, qu'il était impossible sur le plan pratique, de concevoir un système de codage optimal au sens de Shannon. La recherche de nouveaux systèmes de codages incitée par la *révolution Turbo* permit, peut de temps après à Spielman [30] et McKay [31] de redécouvrir les codes à faible densité de parité (LDPC) de Gallager [17]. Ils montrèrent que les codes LDPC avec une longueur de séquence entre 10^3 et 10^4 pouvaient eux aussi parvenir à des performances similaires à celles des codes Turbo. Basé sur le travail de Tanner [32], Wieberg, dans sa thèse de doctorat [33] montra, que les codes dits probabilistes pouvaient être perçus, comme une classe de codes modélisés par un graphe clairsemé et décodés par un algorithme *somme-produit.* Cette découverte permit d'unifier les codes modernes et donna naissance à un nouvel axe de recherche intitulé *codes en graphes.*

Dans cette section, la représentation graphique des codes linéaires est traitée. Afin d'étudier l'aspect probabiliste des codes en graphe, deux sortes de codes sont illustrés : les codes LDPC et les codes Turbo.

3.4.1 Représentation graphique des codes linéaires

Pour pouvoir représenter les codes linéaires sous forme de graphe, il est indispensable d'exprimer à l'aide d'un système d'équations, la relation (3.4) ou (3.6), où chaque équation correspond à une condition que le code doit respecter, incluant des variables dites internes et externes. Deux types de graphes sont illustrés dans cette section : les graphes de Tanner et les graphes de Forney. Les graphes de Forney sont une extension de ceux de Tanner. Ils simplifient la compréhension du décodage dans certaines classes de codes. Afin d'illustrer ces deux représentations, prenons comme exemple le code (7,4) défini par la matrice génératrice suivante :

$$G=\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.19}$$

Cette matrice implique une réalisation de sept équations linéaires homogènes sous $\mathbb{F}_2$ incluant quatre variables d'états :

$$\begin{aligned} y_0 &= u_1 + u_3 \\ y_1 &= u_1 \\ y_2 &= u_2 \\ y_3 &= u_4 \\ y_4 &= u_1 + u_2 + u_4 \\ y_5 &= u_1 + u_2 + u_3 + u_4 \\ y_6 &= u_3 + u_4 \end{aligned} \tag{3.20}$$

Il est possible d'exprimer ce code grâce à un graphe de Tanner ou de Forney à partir de la relation (3.20)

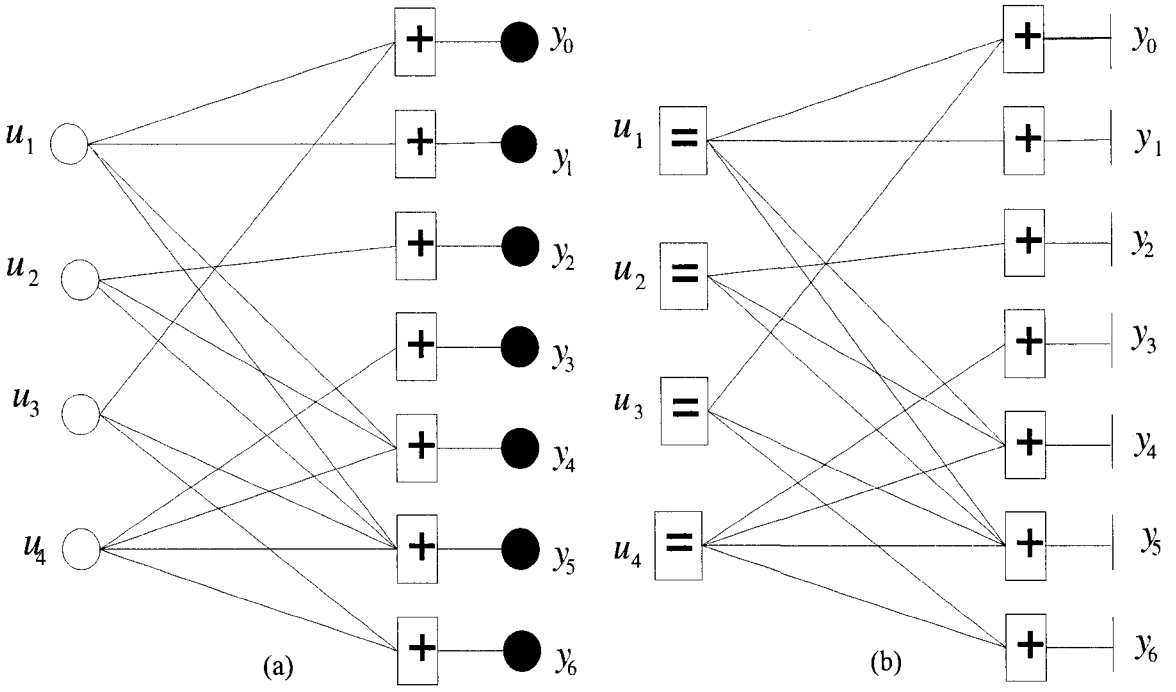

Figure 3.9*: Représentation graphique d'un code défini à l'aide de sa matrice génératrice (a) graphe de Tanner (b) graphe de Forney*

Les graphes de Forney et de Tanner sont des graphes bipartis [34]. Les graphes de Forney et Tanner représentés par la figure 3.9 sont en réalité deux variantes d'un même type de graphe appelé *factor graph* [41]. Même si le graphe de Tanner demeure la référence [32], le graphe de Forney est de plus en plus utilisé, car il facilite d'une part, la compréhension du décodage de certains codes en graphes, mais aussi, car il permet d'établir une dualité entre la représentation graphique d'un code défini à l'aide de sa matrice génératrice et celle où le code est défini à l'aide de sa matrice de parité [39]. Dans le graphe de Tanner de la figure 3.9 les variables externes (symboles codés) sont représentées par des cercles pleins, les variables internes (bits d'information) par des cercles vides et les contraintes par des rectangles de signe *plus*. Si une variable interne est reliée avec une contrainte, cela indique que cette même variable fait partie d'une des

équations du système. Le graphe de Forney est différent, il représente les variables par des arrêtes, avec une forme spéciale en $\perp$ pour les variables externes, où les contraintes sont illustrées par des rectangles de signes différents.

3.4.2 Codes LDPC

Les codes LDPC sont des longs codes définis par l'ensemble des $N-K$ équations de parité :

$$\boldsymbol{y}\boldsymbol{H}^T = 0 \tag{3.21}$$

où $\boldsymbol{H}$ est la matrice de parité binaire de dimension $(N-K)\times N$:

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & \cdots & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}. \tag{3.22}$$

La distribution des 1's est effectuée de manière aléatoire, elle demeure relativement faible, de l'ordre de N au lieu N^2. Le graphe de Forney associé aux codes LDPC est illustré par la figure 3.10, ce graphe à l'inverse de l'exemple illustré par la figure 3.9 représente un code défini l'aide de sa matrice de parité et non à l'aide de sa matrice génératrice, le graphe de Forney traduit les équations de parité de la relation (3.21) sous

forme de graphe. Le symbole π représente une longue permutation reliant les nœuds ensemble. Nous verrons que cette permutation permet d'établir un lien avec d'autres classes de codes qui sont munis d'entrelaceurs. Les premiers codes LDPC développés par Gallager en 1961 étaient des codes réguliers [17]. Les codes réguliers ont la particularité d'inclure le même nombre d_λ de variables externes différentes et d'avoir le même nombre d_ρ de variables externes dans chaque équation du système. d_λ et d_ρ peuvent être aussi interprétés respectivement soit par le nombre de 1's de chaque colonne et de chaque rangée de la matrice $\boldsymbol{H}$, soit par le degré des nœuds du coté gauche (rectangles de signe *égal*) et du coté droit (rectangles de signe *plus*) d'un graphe de Forney. Le nombre d'arêtes est donc égal à :

$$Nd_\lambda = (N-K)d_\rho \tag{3.23}$$

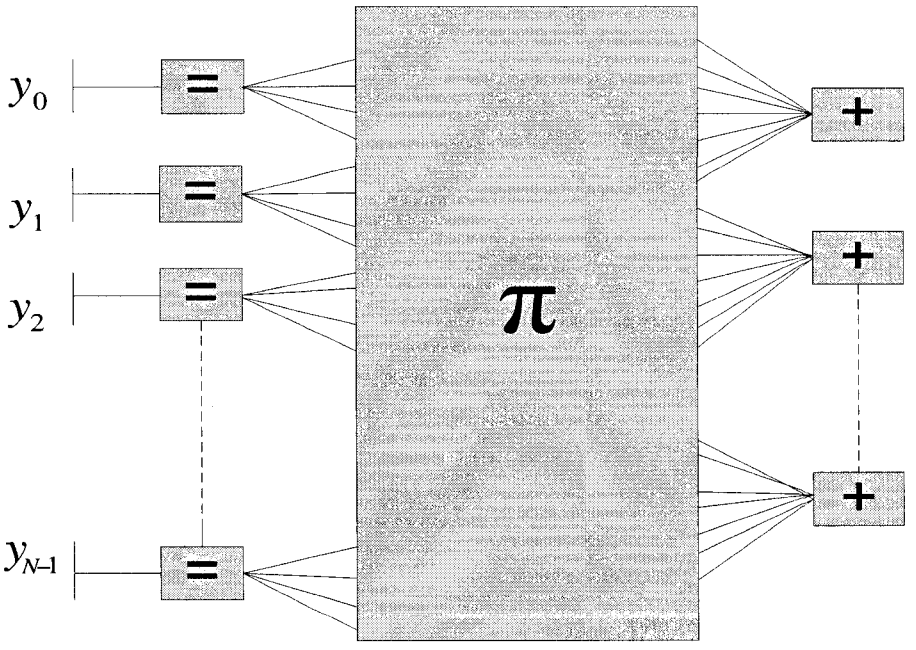

Figure 3.10: Graphe de Forney associé aux codes LDPC

À partir de la relation (3.23), il est possible de définir le taux de codage R simplement à l'aide des degrés d_λ et d_ρ des nœuds du graphe :

$$1-R=\frac{N-K}{N}=\frac{d_\lambda}{d_\rho} \tag{3.24}$$

Une fois que la structure du code prend en compte ces deux contraintes, les 1's de la matrice $\boldsymbol{H}$ sont choisis de façon pseudo-aléatoire.

Le décodage des codes LDPC s'effectue à l'aide d'une version itérative de l'algorithme *somme-produit* [17]. À chaque itération, l'algorithme affine les probabilités a posteriori $\{p(\boldsymbol{y}/\boldsymbol{r}), \boldsymbol{y}\in C\}$ (APP), où $\boldsymbol{r}$ représente la séquence reçue observable. C'est la raison pour laquelle cet algorithme s'appelle aussi *belief propagation*. L'algorithme *somme-produit* est exact lorsque le graphe ne contient aucun cycle [40], où un cycle d'un graphe représente un sous-ensemble ordonné d'arrêtes, tel que tout couple d'arrêtes consécutives puisse s'écrire $\{(u,v),(v,w)\}$, (u et v sont des sommets) et que la première arrête et la dernière arrête s'écrivent (a, b) et (c, a). Contrairement, lorsque le graphe contient des cycles, le décodage devient itératif et n'est plus optimal à maximum de vraisemblance. Cependant, les cycles sont indispensables pour réduire la complexité [35]. Les techniques de décodage itératives ont, depuis la redécouverte des codes LDPC, connu un succès important, car elles permettent de décoder certains codes de manière quasi optimale en en réduisant considérablement la complexité [35]. La faible densité des 1's de la matrice $\boldsymbol{H}$ assure que le graphe est défini localement comme un arbre et permet le bon fonctionnement de l'algorithme de décodage. Le nombre d'itération reste cependant élevé, entre 100 et 200.

L'amélioration majeure des codes LDPC depuis leur découverte dans les années soixante est l'introduction des codes irréguliers [36]. Les codes LDPC irréguliers

n'imposent aucune contrainte sur les degrés des nœuds du graphe. Récemment, un code irrégulier développé par S. Y. Chung et al [37] fut capable d'atteindre littéralement la capacité en se rapprochant de 0,0045 dB de la limite de Shannon. La réelle amélioration des codes LDPC en relaxant la contrainte de régularité, réaffirme l'importance de la notion d'aléatoire. Effectivement, les codes LDPC réguliers contiennent déjà une structure aléatoire conséquente et le fait de relaxer simplement les conditions sur les rangées et les colonnes de la matrice $\boldsymbol{H}$ améliore significativement les performances.

Après avoir défini les caractéristiques principales des codes LDPC, et exposé dans le chapitre 2 les éléments essentiels à la réalisation d'un codage optimal, nous pouvons aisément tenter d'expliquer les raisons de l'excellente performance des codes LDPC. Effectivement, les codes LDPC sont en parfait accord avec la théorie, puisqu'ils renferment une structure aléatoire, sont longs et sont décodés par un algorithme quasi optimal.

3.4.3 Codes Turbo

Les codes Turbo sont générés à l'aide de deux codeurs convolutionnels séparés par un entrelaceur. Comme illustré dans la figure 3.11, les symboles de sortie du codeur sont constitués de bits d'informations, de symboles codés générés par le premier codeur convolutionnel et de symboles codés provenant des bits d'information permutés puis encodés par un deuxième codeur convolutionnel, qui est en général identique au premier, mais qui peut être différent. La permutation s'effectue grâce à un entrelaceur. Ce dernier joue un rôle majeur, car il permet d'une part d'ajouter de l'aléa, et d'autre part de «brasser » l'information et d'augmenter la distance libre du code. Les performances d'erreur du code augmentent avec la longueur de l'entrelaceur. La figure 3.11 illustre le premier codeur Turbo qui fut présenté à la conférence ICC en 1993. Il demeure l'un des meilleurs codes Turbo actuels. Il contient deux codeurs convolutionnels récursifs de mémoire $M = 4$. L'entrelaceur utilisé est de taille L=65536

bits et le taux de codage est égal à 1/3. Cependant, il est possible de le perforer [38] pour obtenir des taux de codage 1/2 ou supérieurs à 1/2. Le graphe de Forney associé au code Turbo est présenté dans la figure 3.12. On retrouve la longue permutation qui s'apparente à celle du graphe de Forney des codes LDPC.

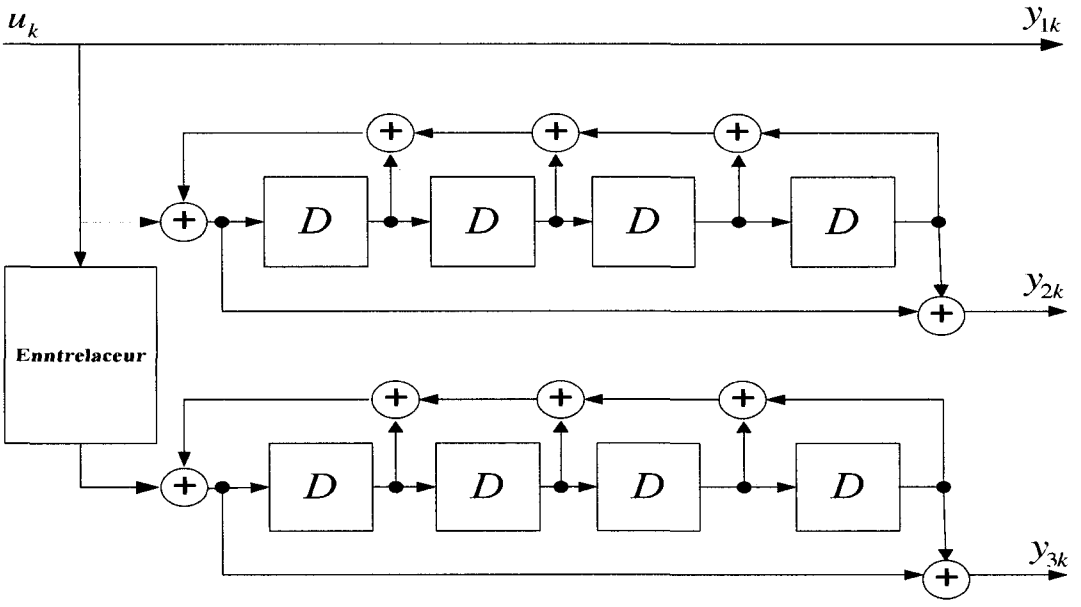

Figure 3.11: codeur Turbo, R=1/3 ayant

$$g_1(D) = g_2(D) = [\frac{1+D^4}{1+D^2+D^3+D^4}]$$

Le décodage des codes Turbo s'effectue également à l'aide d'une version itérative de l'algorithme *somme-produit* [39], le décodage s'opère sur le graphe en alternant à l'aide de la permutation, entre le treillis de gauche et celui de droite. Sur chaque treillis, l'algorithme *somme-produit* est réduit à un algorithme de décodage de bits BCJR [40] où l'algorithme BCJR est un algorithme qui maximise les APP sur un treillis.

La longue permutation, également retrouvée dans le graphe de Forney des codes LDPC, signale l'aspect aléatoire des codes Turbo. Les codes Turbo peuvent atteindre

d'excellentes performances. Par exemple, le code généré par le codeur de la figure 3.11 peut atteindre à la dix-huitième itération une probabilité de bit en erreur de 10^{-5} à 0,7dB de la capacité du canal [4]. Par contre en dessous de cette probabilité d'erreur, un phénomène de *plancher d'erreur* (error floor) intervient. Ce dernier est causé par la faible distance minimale inhérente aux codes Turbo [61], même si la distance minimale augmente avec la mémoire des codeurs convolutionnels qui intègre le codeur Turbo, la complexité devient ingérable pour des codes Turbo de mémoire supérieures à 6. Cette distance minimale relativement faible est due en partie au fait que les codeurs convolutionnels intervenant dans le système Turbo sont petits (longueurs de contrainte K faibles). Il est montré que les codes Turbo sont asymptotiquement moins performants que les codes LDPC et souffrent d'une distance minimale cinq à dix fois inférieure à ces derniers [10].

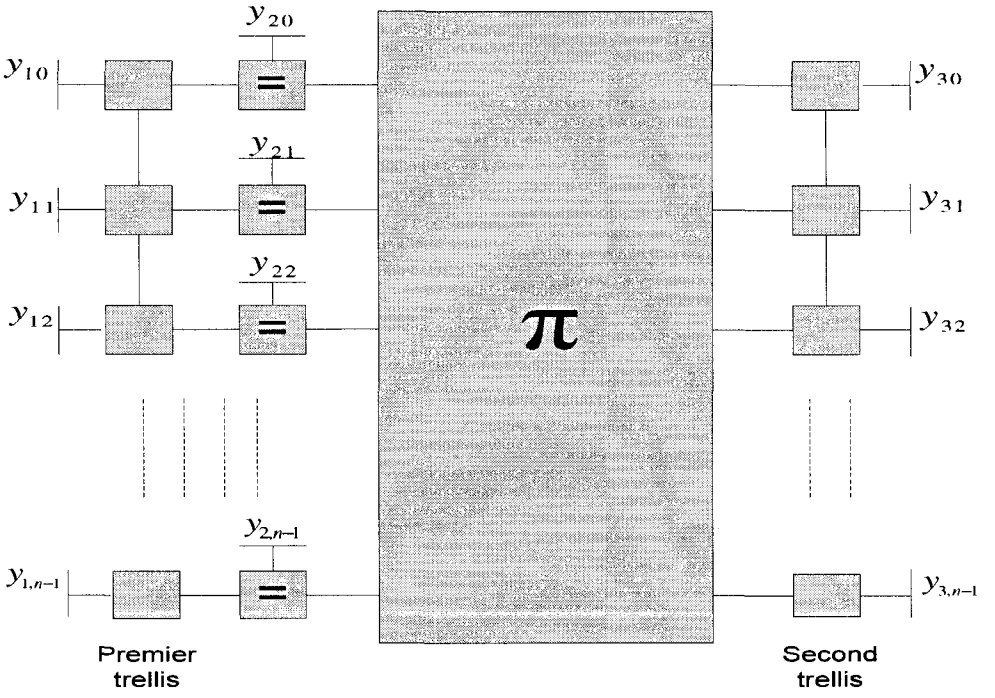

Figure 3.12: Graphe de Forney associé aux codes Turbo

Les codes Turbo restent cependant plus attrayants en termes d'application [41]. Effectivement, les codes Turbo sont peu complexes et relativement simples à implémenter. Même si l'entrelaceur constitue un handicap pour certaines communications à temps réel, à cause du délai au décodage important qu'il engendre, les codes LDPC sont en comparaison avec les codes Turbo beaucoup plus complexes. Cela provient essentiellement du fait que les codes LDPC sont très longs, ce qui demande de très nombreux calculs tant à l'encodage qu'au décodage, mais aussi que le décodage requiert un très grand nombre d'itération.

3.5 Conclusion

Dans ce chapitre ont été exposées trois classes de codes. Le lien entre la théorie de Shannon et les systèmes de codage les plus performants fut établi. Le codage aléatoire s'est révélé être la pierre angulaire du codage moderne. Les codes LDPC respectent les directives de Shannon pour atteindre la capacité et sont asymptotiquement optimaux. Cependant, les codes Turbo et les codes convolutionnels avec décodage de Viterbi sont les codes les plus utilisés en pratique. Les codes Turbo sont supérieurs aux codes convolutionnels en termes de performance d'erreur, mais souffrent de complexité de mise en œuvre et le délai associé au décodage est élevé. L'objectif de la prochaine section et de celui du mémoire dans son ensemble, est d'ajouter de l'aléa aux codes convolutionnels sans souffrir de délai de décodage excessif causé par un entrelaceur, et profiter en même temps de l'optimalité de l'algorithme de Viterbi.

CHAPITRE 4

CODES CONVOLUTIONNELS À TEMPS VARIANT QUASI APÉRIODIQUES

4.1 Introduction

Comme indiqué dans les chapitres 2 et 3, la notion d'aléatoire est essentielle à la réalisation d'un système de codage performant, que ce soit sur le plan théorique avec les ensembles aléatoires de Shannon, que sur le plan pratique avec les codes en graphes. Toutefois, compte tenu de l'excellent rapport entre performance et complexité des codes convolutionnels avec décodage de Viterbi, ces derniers font partie des systèmes, parmi les plus utilisés en pratique [54], [55]. Afin de combiner l'aspect aléatoire des codes modernes et l'efficacité pratique des codes convolutionnels avec décodage de Viterbi, nous avons pensé faire varier les connexions du codeur convolutionnel de manière pseudo-aléatoire sur une longue période, et de modifier l'algorithme de Viterbi pour qu'il puisse s'adapter aux variations des connexions du codeur. Compte tenu du fait que ces variations s'effectuent sur une longue période, nous avons nommé les codes générés par ce système : *codes convolutionnels à temps variant quasi apériodiques*. On note que les codes convolutionnels à temps variant ont déjà été introduits dans le passé dans divers contextes, que ce soit dans le but d'obtenir une distance minimale supérieure aux codes fixes [42], [43], ou encore plus récemment, afin de construire une variante des codes LDPC [45]. L'originalité de notre travail de recherche se situe dans l'aspect aléatoire des connexions choisies sur une longue période et l'utilisation d'un algorithme de Viterbi adaptatif comme technique de décodage pratique.

Dans la première partie de ce chapitre, nous introduisons le mécanisme du codeur pour cette nouvelle classe de codes, ainsi que le mécanisme et les propriétés de l'algorithme de Viterbi adaptatif. Ensuite dans la deuxième partie, nous analysons les codes

convolutionnels à temps variant quasi apériodiques systématiques et non-systématiques. La simulation de ces deux systèmes sera présentée ainsi que leurs performances d'erreur.

4.2 Codes convolutionnels à temps variant périodiques

4.2.1 Principes du codeur

Un codeur convolutionnel à temps variant périodique (CTV-P) de taux $1/n$ bit par symbole peut être représenté comme une machine linéaire à nombre d'états finis consistant en un registre à décalage de K cellules et n modulo-2 additionneurs, connectés à chaque instant donné dans une période T à certaines cellules différentes du registre, impliquant que les connexions sont changées à chaque instant. La figure 4.1 illustre le schéma de ce type de codeur. Les modules *connexions actives* au-dessus des additionneurs modulo-2 ainsi que les connexions en pointillés de la figure 4.1 indiquent que les connexions de chaque additionneur modulo-2 sont différentes à chaque instant donné dans une période T, puis se répètent sur une autre période T et ainsi de suite.

Il est possible de décrire les codes CTV-P à l'aide des matrices des connexions $\boldsymbol{G}_j^{TV}$, $j=1,2,..n$ qui sont définies comme les vecteurs temporels des vecteurs des connexions définis dans la section 3.3.1.

$$\boldsymbol{G}_j^{TV} = (\boldsymbol{G}_j^1, \boldsymbol{G}_j^2 ..... \boldsymbol{G}_j^T) = \begin{bmatrix} g_{j,1}^1 & g_{j,1}^2 & \cdots & g_{j,1}^T \\ g_{j,2}^1 & g_{j,2}^2 & \cdots & g_{j,2}^T \\ \vdots & \vdots & \cdots & \vdots \\ g_{j,K}^1 & g_{j,K}^2 & \cdots & g_{j,K}^T \end{bmatrix}, \quad j=1,2,..n \tag{4.1}$$

où chaque $\boldsymbol{G}_j^i, i=1.....T$ équivaut au vecteur des connexions d'un codeur convolutionnel fixe tel qu'illustré dans le chapitre précédent. L'indice temporel supérieur i indique que

les vecteurs des connexions changent avec le temps. La périodicité des codes implique que :

$$\boldsymbol{G}_j^{i+kT} = \boldsymbol{G}_j^i, \quad i = 1, 2.....T \quad , k = 1, 2, ...\lfloor (h-T)/T \rfloor \tag{4.2}$$

où h représente la longueur de la séquence d'information. La mémoire du codeur CTV-P pour un code de taux $1/n$ bit par symbole est égale à $M = K - 1$.

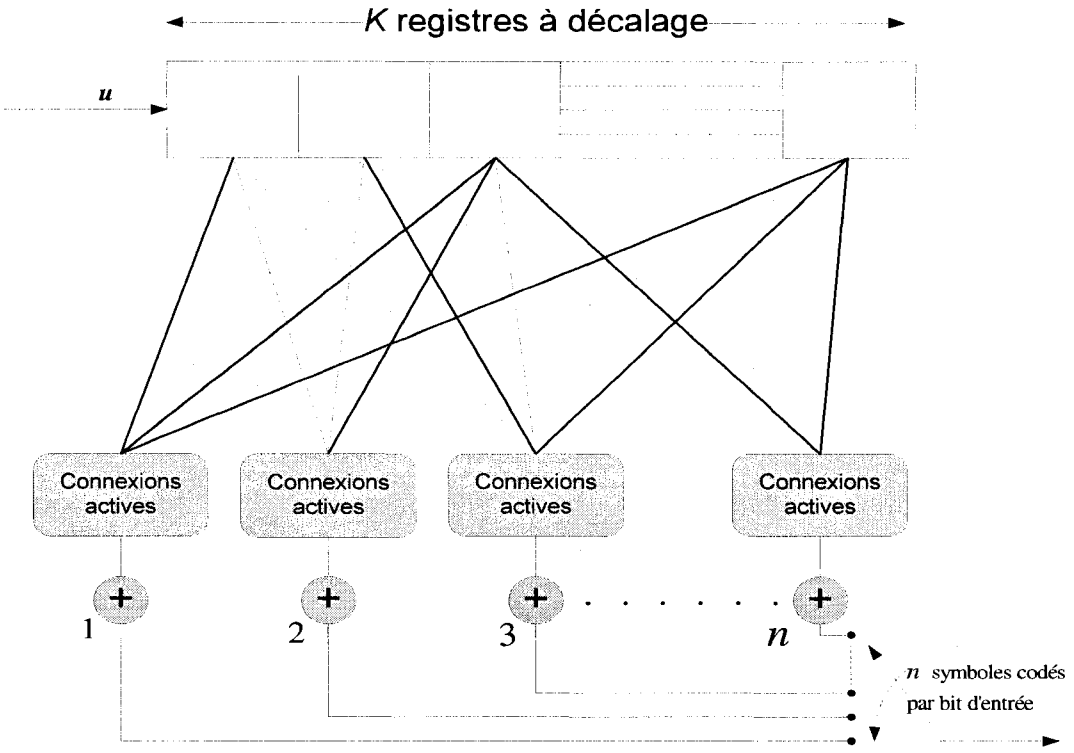

Figure 4.1*: Schéma d'un codeur CTV-P de taux de codage 1/n*

La matrice des connexions exprime la présence ou l'absence de connexions entre l'additionneur j et les K cellules du registre à chaque instant donné de la période T avec $j = 1, 2, ..n$. Une cellule du registre est connectée à un additionneur si la valeur est égale à '1' et ne l'est pas si la valeur est égale à '0'. Afin d'illustrer la matrice génératrice des

codes CTV-P, nous devons redéfinir la matrice des connexions à l'aide de $\boldsymbol{G}_i^*$ $i = 1, 2, ...K$, tel qu'illustré dans le chapitre précédent :

$$\boldsymbol{G}_i^{*TV} = (\boldsymbol{G}_i^{1*}, \boldsymbol{G}_i^{2*} .....\boldsymbol{G}_i^{T*}) \qquad i = 1, 2, ...K \tag{4.3}$$

Cette nouvelle notation permet d'alléger considérablement la notation de la matrice génératrice des codes CTV-P, qui est définie par :

$$\boldsymbol{G} = \begin{bmatrix}
\boldsymbol{G}_1^{*1} & \boldsymbol{G}_2^{*1} & \cdots & \cdots & \boldsymbol{G}_n^{*1} & 0 & 0 & 0 & \vdots & \vdots & \vdots \\
0 & \boldsymbol{G}_1^{*2} & \boldsymbol{G}_2^{*2} & \cdots & \cdots & \boldsymbol{G}_n^{*2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 \\
\vdots & 0 & 0 & 0 & \boldsymbol{G}_1^{*T} & \boldsymbol{G}_2^{*T} & \cdots & \cdots & \boldsymbol{G}_n^{*T} & 0 & 0 \\
\vdots & \vdots & 0 & 0 & 0 & \boldsymbol{G}_1^{*1} & \boldsymbol{G}_2^{*1} & \cdots & \cdots & \boldsymbol{G}_n^{*1} & 0 \\
\vdots & \vdots & \vdots & 0 & 0 & 0 & \boldsymbol{G}_1^{*2} & \boldsymbol{G}_2^{*2} & \cdots & \cdots & \boldsymbol{G}_n^{*2} \\
\vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots \\
\vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & \ddots & \ddots & \ddots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & \boldsymbol{G}_1^{*T} & \boldsymbol{G}_2^{*T} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & \ddots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0
\end{bmatrix} \tag{4.4}$$

Cette matrice est différente de la matrice génératrice des codes convolutionnels fixes puisque les rangées ne sont pas strictement répétitives avec un décalage à droite.

Nous pouvons comparer par exemple, le code fixe de la figure 3.2 et le code CTV-P de taux de codage R=1/2, de période $T = 3$ et de mémoire $M = 3$ où les connexions variant dans le temps sont dans l'ordre [17 15], [17 13] et [15 13], où chaque nombre de ces connexions correspond au vecteur des connexions $\boldsymbol{G}_j^i$ $i = 1, 2, ...T$, $j = 1, 2...n$ sous

forme octale à un instant i donné de la période T. Étant donné que la période T du code CTV-P est égale à 3 et que le taux de codage est égal à 1/2, dans notre exemple, l'indice $i = 1,2,3$ et l'indice $j = 1,2$. Les vecteurs des connexions du code fixe et les matrices des connexions du code CTV-P sont donnés respectivement par :

$$\boldsymbol{G}_1 = (1111)\ ,\ \boldsymbol{G}_2 = (1101) \tag{4.5}$$

$$\boldsymbol{G}_1^{TV} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{G}_2^{TV} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \tag{4.6}$$

À partir des relations (4.3), (4.5) et (4.6), il est possible de reformuler les vecteurs ainsi que les matrices des connexions sous la forme $\boldsymbol{G}_i^*$ et $\boldsymbol{G}_i^{*TV}$, et d'en déduire la matrice génératrice du code convolutionnel fixe (4.7), ainsi que celle du code CTV-P (4.8) de l'exemple.

$$\boldsymbol{G}_{\text{Fixe}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & \cdots & \cdots \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \cdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \cdots & \cdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \cdots & \cdots & \cdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \tag{4.7}$$

$$\boldsymbol{G}_{\text{CTV-P}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & \cdots & \cdots \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ \cdots & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ \cdots & \cdots & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ \cdots & \cdots & \cdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \tag{4.8}$$

Nous constatons que la première rangée de la matrice génératrice des codes convolutionnels fixes de la relation (4.7) se répète, avec un décalage d'une position, d'une rangée à l'autre, alors que dans la matrice génératrice des codes CTV-P de la relation (4.8), la répétition s'effectue par blocs de $T = 3$ rangées, ce qui équivaut à la longueur de la période. Lors de l'introduction du codeur CTV-QA dans la prochaine section, la structure de la matrice génératrice sera comparée avec celle d'autres classes de codes.

4.2.2 Génération des codes CTV-P

Les symboles à la sortie du codeur de la figure 4.1 représentent les symboles codés faisant partie de l'un des mots de code. Le code CTV-P dans son intégralité est obtenu par l'ensemble des séquences codées générées par l'ensemble des entrées possibles. Ce qui mathématiquement équivaut au produit matriciel modulo-2 de la matrice d'information $\boldsymbol{U}$ de dimension $(2^h \times h)$ avec la matrice génératrice $\boldsymbol{G}$, où $\boldsymbol{U}$ représente l'ensemble des 2^h séquences d'information $(u_1, u_2, ...u_h)$ et h représente la longueur de la séquence d'information :

$$C = \boldsymbol{UG} \tag{4.9}$$

4.3 Introduction aux codes convolutionnels à temps variant quasi apériodiques

Les codes convolutionnels à temps variant quasi apériodiques (CTV-QA) représentent une extension des codes CTV-P, lorsque la période T est longue, entre 10^3 et 10^4 bits, que la longueur de la séquence d'entrée varie entre $1000\times T$ et $2000\times T$ et lorsque les variations des connexions du codeur sur cette période s'effectuent de manière pseudo aléatoire.

4.3.1 Lien avec les codes Turbo et LDPC

La longue période des codes convolutionnels à temps variant quasi apériodiques (CTV-QA) donne lieu à une matrice génératrice moins structurée que celle des codes CTV-P. Ce qui, comme mentionné à plusieurs reprises dans les chapitres précédents est un avantage tant sur le plan pratique que théorique. Le lien que nous tentons d'établir entre les codes CTV-QA et les codes LDPC et Turbo n'a pas pour but de placer les trois classes de codes au même niveau. Nous voulons simplement appuyer les motivations sous-jacentes à notre travail de recherche et montrer que le lien entre les codes Turbo et LDPC avec les codes CTV-QA est plus prononcé que celui avec les codes convolutionnels fixes ou avec les codes CTV-P.

Il a été montré dans le chapitre précédent, que les *uns* de la matrice de parité $\boldsymbol{H}$ qui caractérise les codes LDPC, sont générés de manière pseudo aléatoire, avec certaines conditions pour les codes dits *réguliers* et aucune pour les codes dits *irréguliers*. Nous avons mentionné également, que le fait de relaxer les conditions de régularité améliore les performances du code. De manière similaire, les codes CTV-QA reproduisent un peu le schéma des codes LDPC, dans le sens où les *uns* de la matrice génératrice sont aussi distribués de façon pseudo aléatoire sur T rangées de longueur $n\times K$. La génération des

uns de manière pseudo aléatoire de la matrice génératrice des codes CTV-QA est liée aux variations pseudo aléatoires des connexions du codeur sur une période *T*.

La figure 4.2 illustre la matrice de parité d'un code LDPC irrégulier et la matrice génératrice d'un code CTV-QA.

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & \cdots & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

(a)

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots & \cdots & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots & \cdots & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

(b)

Figure 4.2: *comparaison entre la matrice génératrice des codes LDPC irrégulier (a) et la matrice génératrice des codes CTVQA (b)*

La comparaison entre les codes Turbo et les codes CTV-QA est moins facile à effectuer. Les codes Turbo sont beaucoup moins propices à l'analyse que la majorité des codes correcteurs d'erreurs [46]. Le lien entre les codes Turbo et les codes CTV-QA réside principalement dans leur structure réciproque. Effectivement, les deux classes de codes utilisent les codes convolutionnels classiques comme base. Le codeur Turbo illustré dans la figure 3.2 du chapitre précédent est une concaténation de deux codeurs convolutionnels séparés par un entrelaceur. Comme indiqué dans le chapitre 3, l'entrelaceur a pour objectif partiel d'introduire de l'aléa [46]. Nous avons pensé qu'il

est possible d'assimiler la longue permutation des codes Turbo à la longue période T sur laquelle, de façon pseudo-aléatoire, les connexions du codeur CTV-QA varient. Mais d'une certaine manière ces variations peuvent être considérées aussi comme une modification des bits d'entrée pour un certain code convolutionnel fixe.

4.3.2 Génération des codes CTV-QA

La génération des codes CTV-QA ressemble à celle des codes CTV-P. La seule différence réside dans la façon de choisir les variations des connexions du codeur. Il est néanmoins indispensable de noter que pour des codes variant sur une longue période de manière pseudo-aléatoire, un préprocesseur avant le codeur de la figure 4.1 est sans doute indispensable d'un point de vue matériel, afin de garder en mémoire toutes les futures variations du codeur. D'un point de vue analytique, étant donné qu'un code CTV-QA est un code CTV-P avec une longue période, la représentation mathématique (4.9) demeure inchangée.

4.3.3 Représentation graphique des codes CTV-QA

Les codes CTV-QA peuvent être représentés soit sous forme d'arbre adaptatif, soit sous forme de treillis adaptatif. Le diagramme d'état est inapproprié puisqu'il ne prend pas en compte l'évolution temporelle.

4.3.3.1 Arbre adaptatif

La représentation sous forme d'arbre des codes CTV-QA ressemble à celle des codes convolutionnels fixes présentée au chapitre 3. La différence existe uniquement au niveau des symboles codés sur les branches. Ces symboles s'adaptent à présent aux variations des connexions du codeur. La figure 4.3 représente un arbre associé à un code CTV-QA de mémoire M=3 et de taux de codage R=1/2 où les connexions du codeur sont choisies de façon pseudo aléatoire sur une période T=1000. Les quatre premiers ensembles de

connexions sur lesquels le codeur varie sont, donnés dans l'ordre, sous forme octale par [17 13], [17 15], [15 13] et [17 11].

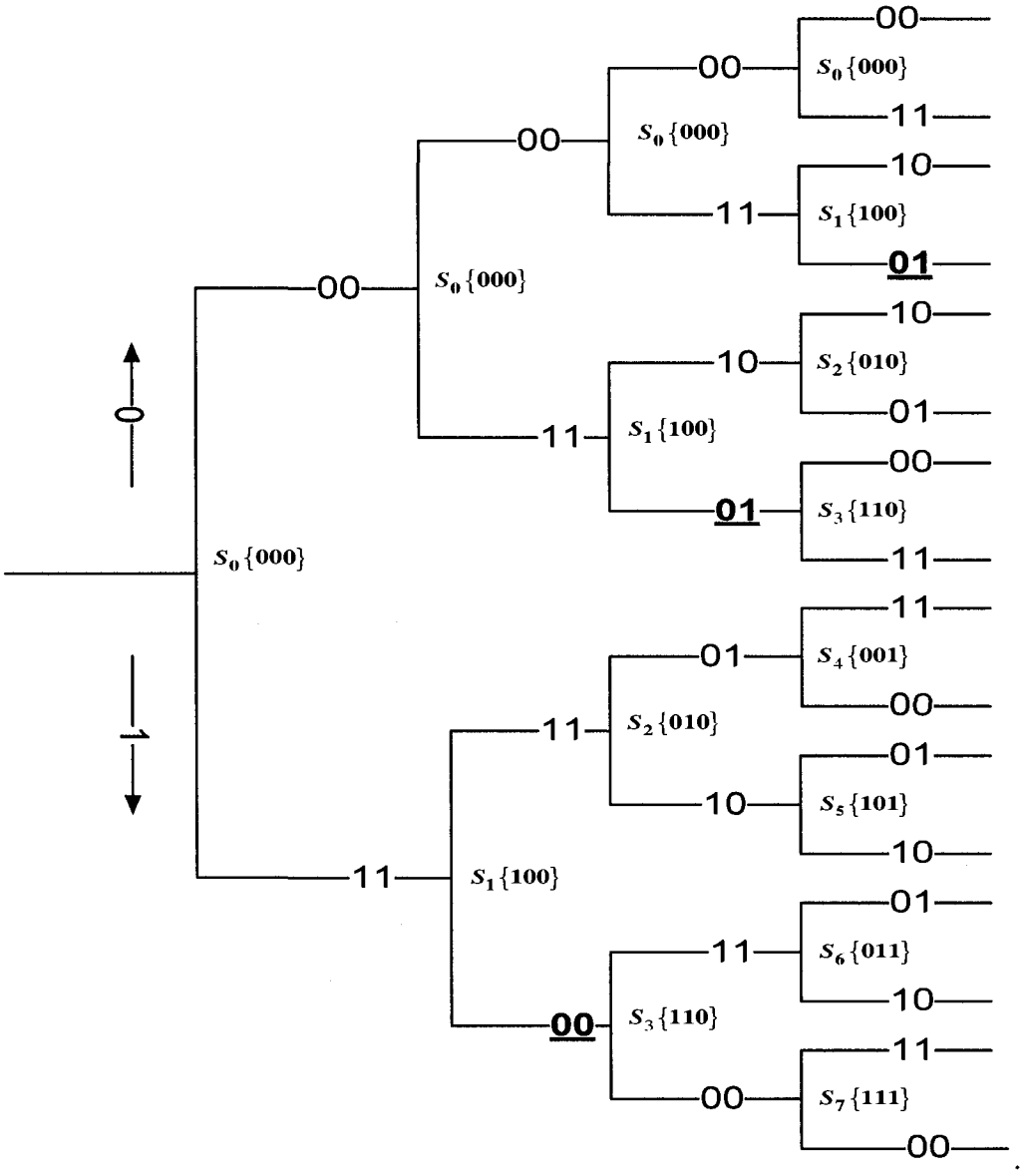

Figure 4.3: *arbre associé au codeur CTV-QA*

L'arbre de la figure 4.3 illustre seulement le début du code, où les symboles codés sur les branches, aux endroits où ils devaient être identiques dans le cas d'un code convolutionnel classique sont soulignés.

4.3.3.2 Treillis adaptatif

Le treillis des codes CTV-QA est considéré comme un treillis à temps variant. Comme dans la représentation en arbre, les symboles codés sur les branches évoluent en fonction des variations des connexions du codeur. La figure 4.4 illustre un treillis associé au même code utilisé pour la représentation en arbre. Mais ici, la figure couvre cinq unités de temps au lieu de quatre, où les cinq premiers ensembles de connexions variant dans le temps sont, dans l'ordre, sous forme octale : [17 13], [17 15], [15 13], [17 11] et [15 17].

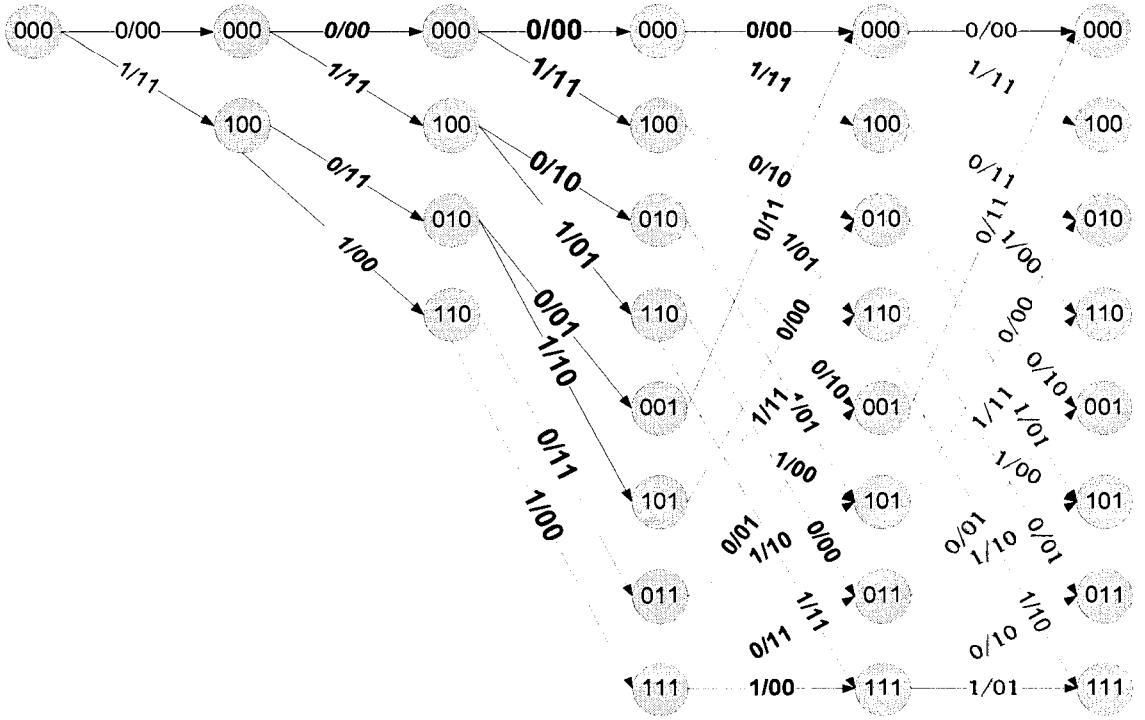

Figure 4.4: *treillis associé au codeur CTV-QA*

Les symboles codés sur chaque branche à chaque unité de temps de la figure 4.4 sont inscrits avec un caractère différent, afin d'indiquer que les symboles sur les branches qui transitent entre deux mêmes états ne sont pas obligatoirement identiques à mesure que l'on traverse le treillis. Le reste du treillis dépend des variations des connexions du codeur. Les symboles sur les branches sont encore différents sur *T-5* unités de temps, ensuite, le même treillis se répétera sur *T* unités de temps et ainsi de suite. Maintenant que le treillis adapté aux codes CTV-QA a été présenté, nous pouvons introduire l'algorithme de Viterbi adaptatif.

4.4 Algorithme de Viterbi adaptatif

Avant de présenter l'algorithme de Viterbi adaptatif, il est préférable de clarifier la notion d'aléatoire présente dans les codes CTV-QA. Le principe est le même que celui des codes LDPC ou Turbo. Le décodeur est informé des connexions actives du codeur à chaque instant de la période *T*. Le fait que le décodeur connaisse les connexions actives à chaque instant ne dénature cependant pas la notion d'aléatoire présente dans les codes CTV-QA. Ce point est assez délicat et porte souvent à confusion. Effectivement, comme mentionné dans le chapitre précédent, le codage aléatoire consiste entre autres à assurer que le code choisi à la transmission fasse partie d'un vaste ensemble. Il n'a pas pour objectif d'introduire de l'incertitude dans le système de transmission.

Le décodage de Viterbi adaptatif est une variante de l'algorithme de Viterbi classique évoluant sur le treillis à temps variant illustré dans la sous-section 4.3.2. Étant donné que le treillis incorpore les variations des connexions du codeur, chaque chemin possible de ce dernier équivaut à un mot de code éventuel.

Considérons une séquence codée générée à l'aide d'un codeur CTV-QA. Cette séquence est transmise à travers un canal Gaussien AWGN en assignant à chaque $y_{jk} \in \{0,1\}$ le

symbole $s(y_{jk}) \in \{\pm\alpha\}$, par le biais de la modulation 2-PAM. La séquence reçue au décodeur est :

$$\boldsymbol{r} = s(\boldsymbol{y}) + \boldsymbol{n} \tag{4.10}$$

où $\boldsymbol{n}$ est une séquence Gaussienne indépendamment distribuée avec une moyenne 0 et une variance $N_0/2$ par dimension. Le décodage à maximum de vraisemblance équivaut comme dans le cas des codes convolutionnels classiques, à trouver la séquence codée $\boldsymbol{y}$ qui minimise la métrique $\|\boldsymbol{r} - s(\boldsymbol{y})\|^2$ pour en déduire ensuite la séquence d'information qui a été transmise. Il est possible d'assigner la métrique $\|\boldsymbol{r}_k - s(\boldsymbol{y}_k)\|^2$ à chaque branche du treillis. Une fois que le treillis incorpore ces données, le décodage de Viterbi adaptatif consiste à trouver la métrique minimale grâce à l'algorithme ACS illustré dans le chapitre précédent, sur l'ensemble du treillis adapté au code CTV-QA transmis.

L'algorithme de Viterbi adaptatif présenté dans cette section garde son optimalité et n'est pas plus complexe (temps de calcul) que l'algorithme de Viterbi classique [47]. Étant donné que le nombre d'opérations à chaque nœud est le même, puisque nous utilisons aussi l'algorithme ACS. La complexité de ce dernier est donc également de l'ordre de 2^M, où M représente la mémoire du codeur CTV-QA.

La prochaine section introduit les codes CTV-QA systématiques La simulation d'un système de codage CTV-QA systématique pour un canal Gaussien sera aussi présentée.

4.5 Analyse des codes CTV-QA systématique

Nous commençons par introduire les codes CTV-QA systématiques, car se sont, les codes CTV-QA les plus simples à mettre en œuvre d'un point de vue réalisation[4]. La figure 4.5 illustre le codeur que nous utilisons pour la simulation des codes CTV-QA systématiques de mémoire *M=4*.

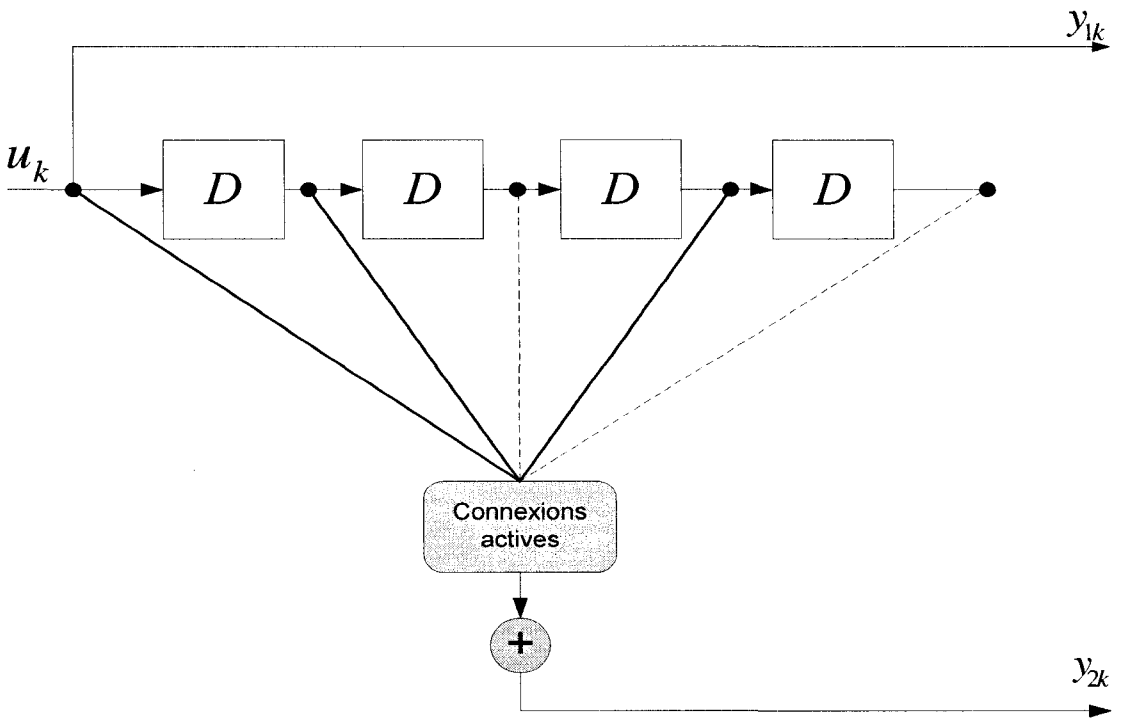

Figure 4.5: *Codeur CTV-QA systématiques, R=1/2, M=4*

Le module *Connexions actives* active certaines connexions à chaque instant sur une période *T*, réactive [5]ces mêmes connexions sur une autre période *T* et ainsi de suite.

[4] Ce point s'éclaircira lors de l'introduction des codes CTV-QA non systématiques.
[5] Ce mécanisme est possible grâce au prétraitement que nous présenterons dans la sous section 4.5.2.1.

Le programme élaboré pour simuler les codes CTV-QA systématiques incorpore bien évidemment un décodeur. Afin d'illustrer son fonctionnement, nous devons présenter l'algorithme de Viterbi adaptatif pratique utilisé dans la simulation des codes CTV-QA.

4.5.1 Algorithme de Viterbi adapté aux codes de longueur dite *infinie*

L'algorithme de Viterbi diffère un peu quand il s'agit des codes convolutionnels de longueur dite *infinie*. Afin de présenter la méthode utilisée pour simuler toutes les variantes des codes CTV-QA illustrés dans ce mémoire, nous présentons dans cette sous-section, l'algorithme de décodage de la manière dont nous l'avons programmé.

Lorsque les codes sont longs, ou de longueur dite *infinie* il n'est pas raisonnable d'attendre que l'algorithme ait traversé tout le treillis avant de pouvoir décider du mot de code (chemin du treillis) qui correspond à la séquence d'information qui a été vraisemblablement transmise. Cela impliquerait d'une part une latence importante au décodage, puisque le décodeur devrait attendre la transmission de l'intégralité de la séquence codée avant de pouvoir décider du premier bit transmis et d'autre part, cela obligerait le programme à garder en mémoire des matrices de taille $2^M \times N$, où N est la longueur de la séquence codée et M la mémoire du codeur. La façon de contourner ce problème est de prendre une décision à un instant donné sur les symboles codés transmis il y'a plus d'un certain temps. Pour cela, il faut reculer d'une distance fixe (traceback length) dans le treillis pour prendre une décision.

Dans le cas d'un code de taux de codage égal à $1/n$ bit par symbole, n symboles sont sélectionnés sur une branche entre les instants k et k-1 d'après le chemin de poids

cumulé minimal sur le treillis entre les nœuds aux instants k et $k+t_b$ [6], signifiant que le délai au décodage est de t_b bits.

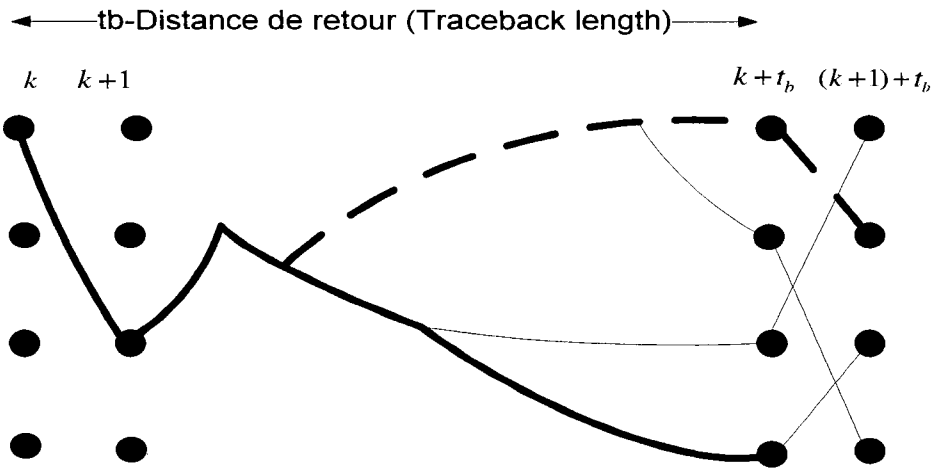

Figure 4.6: *Mécanisme de retour pour l'algorithme de Viterbi adapté aux codes de distance dite infinie.*

La figure 4.6 illustre un exemple où deux chemins de poids minimaux successifs sont choisis (ligne pleine en caractère gras et ligne pointillée en caractère gras). Dans cet exemple nous pouvons remarquer qu'à l'instant $(k+1)+t_b$ tous les chemins minimaux arrivant aux 2^M nœuds, M=2 proviennent d'un seul et unique chemin avec une forte probabilité. Appuyant ainsi la thèse que si l'on prend une longueur de retour assez grande, ce procédé permet de ne pas attendre la transmission de la séquence dans son intégralité pour que l'algorithme prenne une décision. Ce phénomène est une tendance et non une certitude et plus t_b est choisi assez grand, plus il y a des chances que cette tendance se réalise. La distance de retour t_b choisie pour les codes convolutionnels fixes

[6] Par chemin de poids cumulé, nous voulons dire que l'opération ACS s'opère de manière classique sur tout le treillis, et qu'à la profondeur k, les poids sur les nœuds correspondent aux poids des 2^M survivants.

est généralement de $5M$. Dans le cas des codes CTV-QA, les simulations ont montré que la longueur de retour égale à $10M$ était suffisante.

4.5.2 Simulation des codes CTV-QA-S de taux de codage 1/2

Le programme de simulation des codes CTV-QA-S est divisé en deux parties : un prétraitement et un simulateur.

4.5.2.1 Prétraitement

Le prétraitement se divise en plusieurs étapes :

1. Le programme génère des ensembles de connexions de façon pseudo aléatoire pour un codeur de mémoire donnée, où le nombre de connexions dépend de la mémoire du codeur.

2. Le programme teste chaque ensemble de connexions sur un codeur convolutionnel fixe avec décodage de Viterbi et retient ceux qui génèrent des performances d'erreur supérieures à un seuil prédéterminé ψ.

3. Une fois que plusieurs ensembles de connexions aient été sélectionnés, le programme génère aléatoirement une distribution de ces ensembles de connexions sur une période T de longueur comprise entre 10^3 et 10^4.

4.5.2.2 Simulation et analyse des performances

Une fois le prétraitement terminé, le programme de simulation opère aussi en plusieurs étapes :

1. Le programme encode l'information avec le codeur CTV-QA-S où les connexions de ce dernier varient d'après la distribution des ensembles de connexions conservée lors du prétraitement.

2. Ensuite le programme ajoute du bruit Gaussien aux symboles codés.

3. Puis le programme décode le signal reçu à l'aide de l'algorithme de Viterbi adaptatif que nous avons présenté précédemment.

4. Finalement, il calcule les performances d'erreur sur une large gamme de valeurs de E_b / N_0.

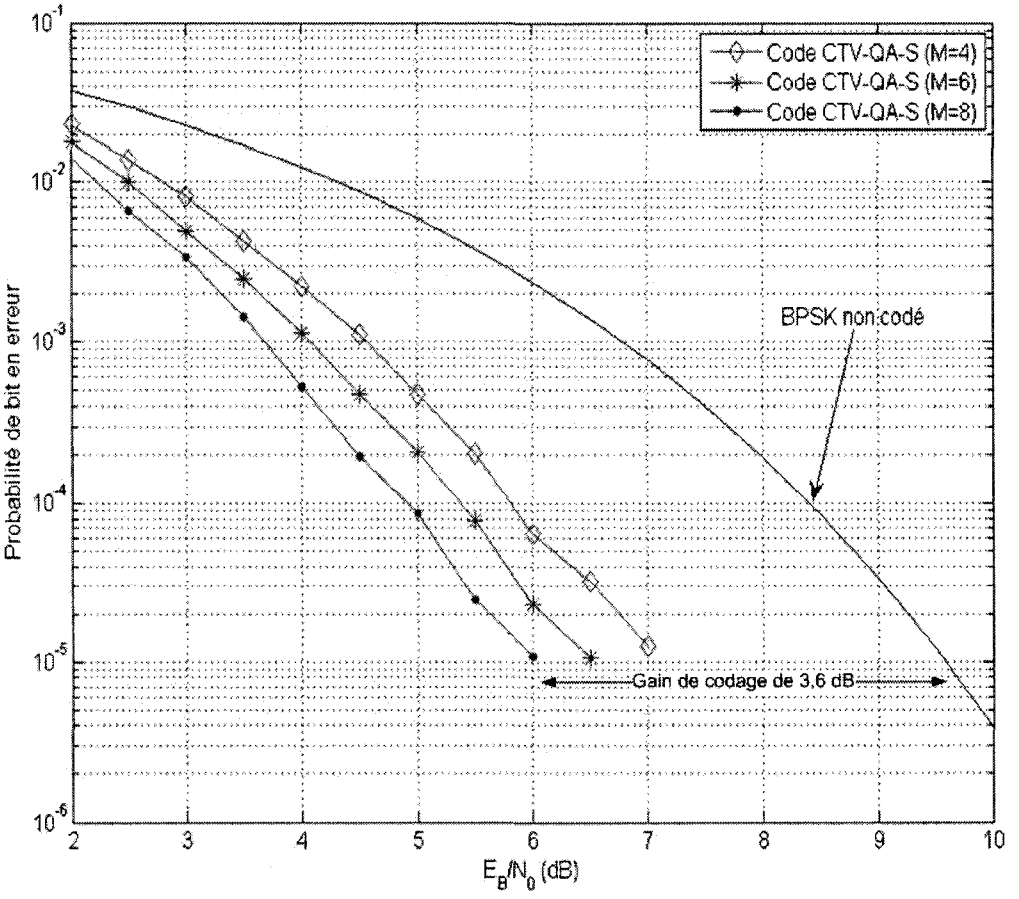

Figure 4.7: *Performances d'erreur de codes CTV-QA-S, R=1/2, T=5000, quantification douce à 3 bits*

La quantification utilisée pour le décodeur est une quantification douce à 8 niveaux[7].

La figure 4.7 illustre les performances d'erreur de codes CTV-QA-S pour des codeurs de mémoires relativement faibles. Le tableau 4.1 relate les ensembles des connexions sous forme octale sur lesquels chaque codeur varie de manière pseudo-aléatoire sur une période $T = 5000$, où chacun des ensembles des connexions sous forme octale du tableau 4.1 correspond aux j vecteurs des connexions $\boldsymbol{G}_j^i$ sous forme octale, où $j = 1, 2...n$ et i peut prendre aléatoirement n'importe quelle valeur entre 1 et T, d'après la distribution générée lors du prétraitement. Le nombre de possibilités d'ensembles de connexions pour un code de mémoire M est donné par 2^{M-1}.

Tableau 4.1: Ensembles des connexions sur lesquels varie le codeur CTV-QA systématique de taux de codage R=1/2

M	Ensembles des connexions
4	{[1 33], [1 31], [1 23], [1 25], [1 27], [1 35]}
6	{[1 135], [1 137], [1 107], [1 145], [1 153], [1 107], [1 133], [1 175], [1 147], [1 115], [1 123], [1 151], [1 173], [1 113], [1 161], [1 131], [1 167]}
8	{[1 727], [1 647], [1 757], [1 557], [1 677], [1 755], [1 515], [1 551], [1 661] [1 455], [1 433], [1 613], [1 625], [1 465], [1 645], [1 725], [1 467], [1 741], [1 547], [1 435], [1 763], [1 573], [1 627], [1 523], [1 745], [1 675], [1 735], [1 545], [1 751], [1 463], [1 417], [1 703], [1 527], [1 573], [1 555], [1 655], [1 705], [1 763], [1 563], [1 643], [1 745], [1 607], [1 775], [1 543], [1 531], [1 613], [1 541], [1 725], [1 567], [1 527], [1 517], [1 447], [1 711], [1 667], [1 637], [1 717], [1 637], [1 565],}

[7] La quantification douce utilisée sera détaillée dans l'annexe II

Les performances d'erreur de la figure 4.7 indique que les gains de codage obtenus par des codes de longueur de mémoire *M=4, M=6* et *M=8* sont respectivement de 2,6 dB, 2,9 dB et 3,5 dB pour une probabilité de bit en erreur de 10^{-5}.

Les performances d'erreur illustrées par la figure 4.7 sont les résultats obtenus après l'optimisation de plusieurs paramètres, tels que le nombre d'ensembles de connexions sur lesquels varient le codeur, la longueur de la période, et le seuil ψ. Cette optimisation s'est effectué par le biais de plusieurs simulations répétitives de codes CTV-QA pour plusieurs de ces paramètres. Cependant malgré qu'un gain de codage de 3,6 dB ait été obtenu par un code systématique relativement petit (faible longueur de contrainte), les codes présentés sont loin de rivaliser avec la plupart des codes convolutionnels fixes non systématiques avec décodage de Viterbi [58]. C'est la raison pour laquelle nous n'étendrons pas plus l'analyse de ces codes.

Une façon naturelle d'améliorer les performances des codes CTV-QA systématiques est d'introduire les codes CTV-QA non systématiques. Nous verrons cependant, que ces codes soulèvent d'autres problèmes liés à leur mise en œuvre.

4.6 Codeurs catastrophiques

Les meilleurs codes convolutionnels sont des codes convolutionnels non systématiques non récursifs [48], [49]. L'extension aux codes CTV-QA non systématiques non récursifs peut paraître simple. Cependant à partir du moment où le codeur n'est plus systématique, certains problèmes peuvent survenir, notamment le fait que le codeur peut être catastrophique.

4.6.1 Codeurs convolutionnels fixes catastrophiques

Dans le cas des codes non systématiques le décodage de Viterbi reproduit avec le plus de fidélité possible la séquence codée initiale. Un autre module se charge normalement de transformer la séquence codée estimée en bits d'information. La figure 4.8 illustre un système de communication utilisant un codeur convolutionnel non systématique de taux de codage b/v et un décodeur de Viterbi classique où le décodeur est divisé en deux parties [28]. C'est le dernier module de la figure 4.8 qui a pour mission de retrouver les bits d'information qui ont été transmis à partir de l'estimation du mot de code. Ce module est caractérisé par la matrice sous forme polynomiale $G^{-1}(D)$ de dimension $v \times b$ qui résout l'équation [59] :

$$G(D)G^{-1}(D) = I_b D^l, \quad \text{pour } l \geq 0 \tag{4.11}$$

où I_b est la matrice unité de dimension $b \times b$.

Jusqu'ici, il n'avait pas parût essentiel de diviser le décodeur en deux parties, car les codes que nous avions simulés étaient systématiques. Sachant, que pour les codes systématiques, on peut retrouver directement les bits d'information grâce à l'opération ACS de l'algorithme de Viterbi.

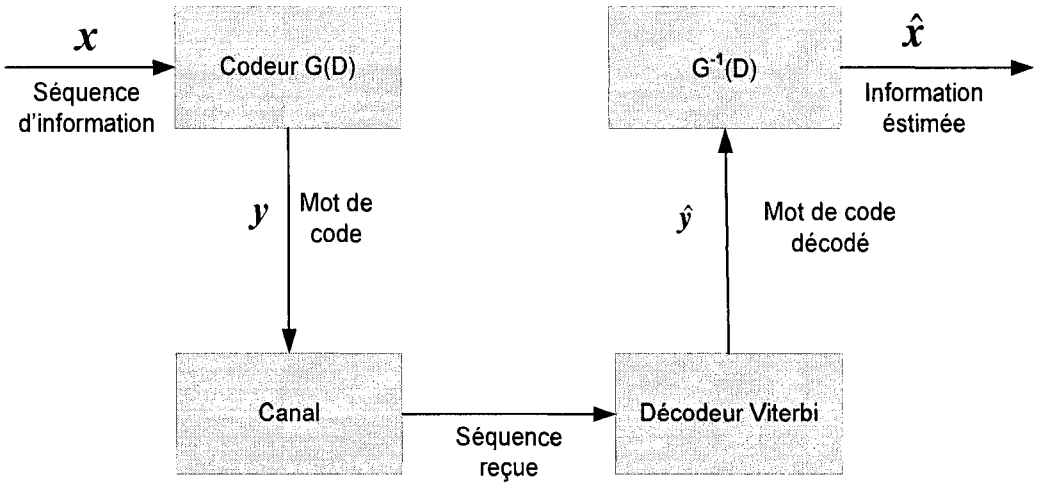

Figure 4.8: *Système de transmission incluant un codeur convolutionnel où le décodeur est divisé en deux parties*

Un problème grave peut alors subvenir pour les codes non systématiques, lorsque le *mot de code décodé,* contient un simple symbole en erreur qui cause une infinité d'erreurs dans la *séquence d'information décodée.* Le codeur capable de produire un tel scénario est appelé *codeur catastrophique.* Puisque les modules $G(D)$ et $G^{-1}(D)$ sont complémentaires, un codeur est catastrophique lorsqu'il peut générer une séquence codée contenant un nombre *fini* d'éléments différents de 0 avec une séquence d'information contenant un nombre *infini* d'éléments différents de 0 [28].

Par exemple, le codeur convolutionnel non récursif non systématique de taux 1/1[8] caractérisé par la matrice génératrice sous forme polynomiale $G(D)=1+D$ est catastrophique.

[8] un codeur convolutionnel de taux 1/1 comprend une entrée et une sortie.

Dans le chapitre précédent, les codeurs systématiques récursifs ont été présentés entre autres, comme des codeurs capables, avec une séquence d'information *finie*, de générer des séquences codées de longueur dite *infinie*. Dans notre cas, la situation est semblable, seulement au décodage et non à l'encodage. Effectivement, le polynôme inverse de l'exemple $\frac{1}{G(D)} = 1 + D + D^2 + D^3 .....$, qui doit retrouver la séquence d'information transmise a les mêmes propriétés qu'un codeur systématique récursif. Si le mot de code constitué entièrement de 0's[9] est transmis à travers le canal, et qu'au décodage une erreur se produit sur un seul symbole, l'*information décodée* sera constituée d'un nombre *infini* d'éléments différents de 0. D'où la définition de catastrophique, une simple erreur perpétrée au décodage peut induire un nombre infini de bits en erreur à la sortie du système .Ce qui correspond aussi à une boucle de poids zéro autre que la branche qui boucle sur l'état zéro dans le diagramme d'état.

Plusieurs méthodes existent pour déterminer si un codeur convolutionnel est catastrophique [50-52]. L'inconvénient principal est que la complexité de ces méthodes augmente avec la mémoire et le nombre d'entrées et de sorties du codeur.

4.6.2 Codeurs convolutionnels à temps variant catastrophiques

Que les codes soient des codes convolutionnels fixes ou à temps variant, la définition d'un codeur catastrophique reste la même. Par contre, la détermination des codes convolutionnels à temps variant catastrophiques [53], implique de mettre en relation les codes convolutionnels fixes et ceux à temps variant. D'après [43], tout code convolutionnel à temps variant avec une mémoire M et une période T de taux de codage b/v est équivalent à un code convolutionnel fixe de taux de codage bT/vT avec une matrice génératrice sous forme polynomiale :

---

[9] Le mot de code constitué entièrement de zéros fait, par linéarité, partie des mots de code des codes convolutionnels

$$\boldsymbol{G}(D) = \sum_{j=0}^{M^*} \boldsymbol{K}_j D^j \tag{4.12}$$

où $M^* = \lceil M/T \rceil$ et :

$$\boldsymbol{K}_j = \begin{bmatrix} \boldsymbol{G}_{jT+1}^{1*} & \boldsymbol{G}_{jT+2}^{2*} & \cdots & \boldsymbol{G}_{jT+T}^{T*} \\ \boldsymbol{G}_{jT}^{1*} & \boldsymbol{G}_{jT+1}^{2*} & \cdots & \boldsymbol{G}_{jT+T-1}^{T*} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{G}_{(j-1)T+2}^{1*} & \boldsymbol{G}_{(j-1)T+3}^{2*} & \cdots & \boldsymbol{G}_{jT+1}^{T*} \end{bmatrix} \tag{4.13}$$

où les $\boldsymbol{G}_j^{u*}, j = 1,2,...K,\ u = 1,2....T$ ont été définis dans la section 4.2 et par convention $\boldsymbol{G}_j^{u*} = 0$ pour $j > K$ et $j \leq 0$.

Prenons comme exemple le code convolutionnel à temps variant de taux de codage R=1/2 et de période $T = 2$ défini par les matrices des connexions $G_i^{TV}, i = 1,2...n$ définies dans la section 4.2 :

$$G_1^{TV} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, G_2^{TV} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.14}$$

Ce qui correspond aux matrices génératrices sous la forme polynomiale pour chaque instant de la période $T = 2$:

$$\boldsymbol{G}_1(D) = [1 + D^3, 1 + D + D^2] \qquad \boldsymbol{G}_2(D) = [1 + D, 1 + D^3] \tag{4.15}$$

Nous devons à présent redéfinir les matrices des connexions $G_i^{TV}, i=1,2..n$ sous la forme $\boldsymbol{G}_i^{*TV}=(\boldsymbol{G}_i^{1*},\boldsymbol{G}_i^{2*}.....\boldsymbol{G}_i^{T*}),\ i=1,2,..K$:

$$\begin{aligned}
\boldsymbol{G}_1^{*TV} &= \left[\boldsymbol{G}_1^{*1},\boldsymbol{G}_1^{*2}\right]=[1111] \\
\boldsymbol{G}_2^{*TV} &= \left[\boldsymbol{G}_2^{*1},\boldsymbol{G}_2^{*2}\right]=[0110] \\
\boldsymbol{G}_3^{*TV} &= \left[\boldsymbol{G}_3^{*1},\boldsymbol{G}_3^{*2}\right]=[0100] \\
\boldsymbol{G}_4^{*TV} &= \left[\boldsymbol{G}_4^{*1},\boldsymbol{G}_4^{*2}\right]=[1001]
\end{aligned} \tag{4.16}$$

La matrice génératrice sous forme polynomiale du code convolutionnel fixe équivalent de taux de codage R=2/4 est ainsi donnée par :

$$\boldsymbol{G}(D)=\sum_{j=0}^{M^*=2}\boldsymbol{K}_j D^j=\begin{bmatrix}\boldsymbol{G}_1^{1*} & \boldsymbol{G}_2^{2*}\\ \boldsymbol{G}_0^{1*} & \boldsymbol{G}_1^{2*}\end{bmatrix}\cdot 1+\begin{bmatrix}\boldsymbol{G}_3^{1*} & \boldsymbol{G}_4^{2*}\\ \boldsymbol{G}_2^{1*} & \boldsymbol{G}_3^{2*}\end{bmatrix}\cdot D+\begin{bmatrix}\boldsymbol{G}_5^{1*} & \boldsymbol{G}_6^{2*}\\ \boldsymbol{G}_4^{1*} & \boldsymbol{G}_5^{2*}\end{bmatrix}\cdot D^2$$

$$\boldsymbol{G}(D)=\begin{bmatrix}1 & 1 & 1 & 0\\ 0 & 0 & 1 & 1\end{bmatrix}+\begin{bmatrix}0 & D & 0 & D\\ 0 & D & 0 & 0\end{bmatrix}+\begin{bmatrix}0 & 0 & 0 & 0\\ D^2 & 0 & 0 & 0\end{bmatrix} \tag{4.17}$$

$$\boldsymbol{G}(D)=\begin{bmatrix}1 & 1+D & 1 & D\\ D^2 & D & 1 & 1\end{bmatrix}$$

où $T=2$, $M=3$, $M^*=\lceil 3/2\rceil$=2 et par convention $\boldsymbol{G}_j^{u^*}=0$ pour $j>K$ et $j\le 0$

L'équivalence entre les codes convolutionnels à temps variant et les codes convolutionnels fixes étant à présent établie, déterminer si un codeur convolutionnel à temps variant de taux de codage b/v est catastrophique équivaut à déterminer si un codeur convolutionnel fixe de taux de codage bT/vT est catastrophique.

Soit $\boldsymbol{G}(D)$ la matrice génératrice sous forme polynomiale d'un code convolutionnel C fixe de taux de codage bT / vT équivalente à celle d'un code convolutionnel à temps variant C^* de taux de codage b / v et de période T. Si [50] :

$$\text{PGCD}(\Delta_1(D), \Delta_2(D), \ldots\ldots \Delta_F(D)) = D^d \tag{4.18}$$

où $\Delta_i(D)$ représente le déterminant de la i^{em} des $F = \begin{pmatrix} vT \\ bT \end{pmatrix}$ sous matrice de $\boldsymbol{G}(D)$ de taille $[bT \times bT]$, $d \geq 0$ et PGCD représente le plus grand diviseur commun, alors le code convolutionnel à temps variant C^* n'est pas catastrophique.

La complexité de l'algorithme augmente malheureusement de manière exponentielle avec la période T car il implique le calcul de plusieurs déterminants de matrices de taille $[bT \times bT]$. O'Dhonoghu et Burkley [53] utilisent aussi le lien entre les codes convolutionnels fixes et ceux à temps variant pour trouver un algorithme qui détermine si un code convolutionnel à temps variant est catastrophique de manière plus rapide. Le temps de calcul pour définir si un code CTV-QA est catastrophique demeure cependant trop élevé, car les codes CTV-QA proposés ont par définition une très longue période. Nous allons donc supposer dans la prochaine section que les codes utilisés ne sont pas catastrophiques, en faisant varier les connexions du codeur CTV-QA uniquement sur des connexions de codeurs fixes non catastrophiques.

4.7 Analyse des codes CTV-QA non systématiques

4.7.1 Réalisation du système

La présentation du décodeur divisé en deux parties distinctes ainsi que l'équivalence entre les codes convolutionnels fixes et ceux à temps variant permettent de définir le système de transmission utilisant un codeur CTV-QA non systématique. Dans le ca de notre recherche nous n'avons pas cherché à retrouver la séquence d'information directement à partir de l'algorithme Viterbi adaptatif, ce qui pourrait être envisageable.

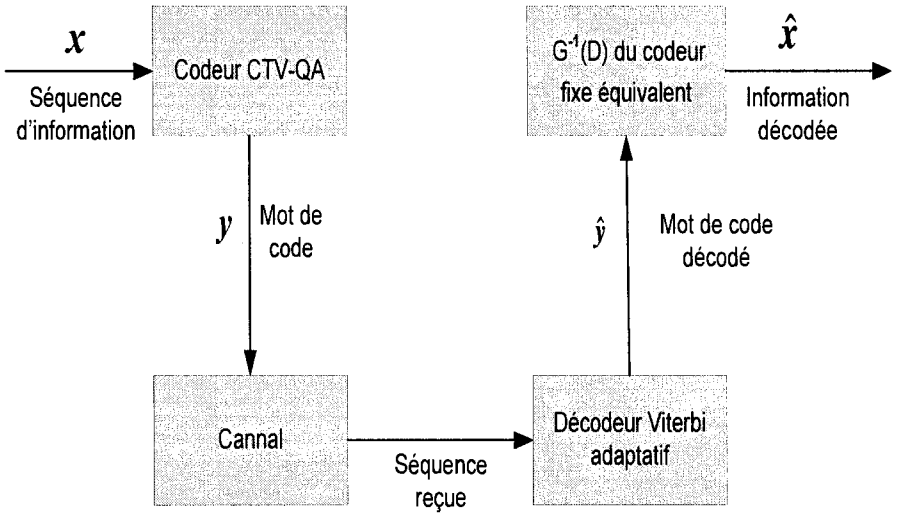

Figure 4.9: *Système de transmission utilisant un codeur CTV-QA non systématique*

Hormis le fait que le codeur CTV-QA non systématique puisse s'avérer catastrophique, le système de la figure 4.9 ajoute de la complexité au décodage par rapport aux codes CTV-QA non systématiques. Car le dernier module de la figure 4.9 inexistant pour les codes CTV-QA systématiques, est associé à une matrice sous forme polynomiale $\boldsymbol{G}^{-1}(D)$ de taille $vT \times bT$, puisqu'un code CTV-QA de taux b/v équivaut à un code

convolutionnel fixe de taux bT / vT [53], impliquant donc une latence au décodage de l'ordre de T et donc une complexité additionnelle au décodage de l'ordre $O(T^3)$ [10].

4.7.2 Simulation des codes CTV-QA de taux de codage 1/2

La figure 4.10 illustre le codeur que nous utilisons pour la simulation des codes CTV-QA non systématiques de taux de codage R=1/2 et de mémoire M=4.

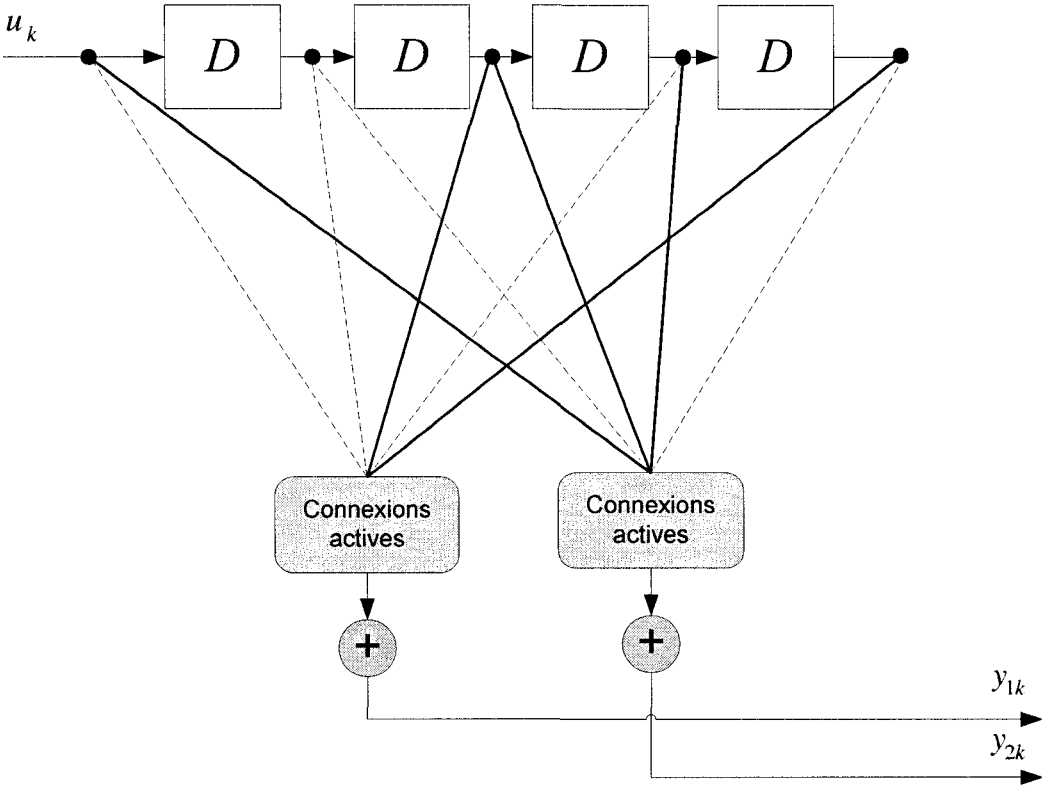

Figure 4.10: Codeur CTV-QA non systématique, R=1/2, M=4

Le programme qui simule les codes CTV-QA est essentiellement le même que celui développé pour les codes CTV-QA systématiques à l'exception du fait, que les

[10] $O(n^3)$ correspond à la complexité de la multiplication matricielle de deux matrices de taille $n \times n$

connexions choisies à chaque instant correspondent à celles d'un codeur convolutionnel non catastrophique.

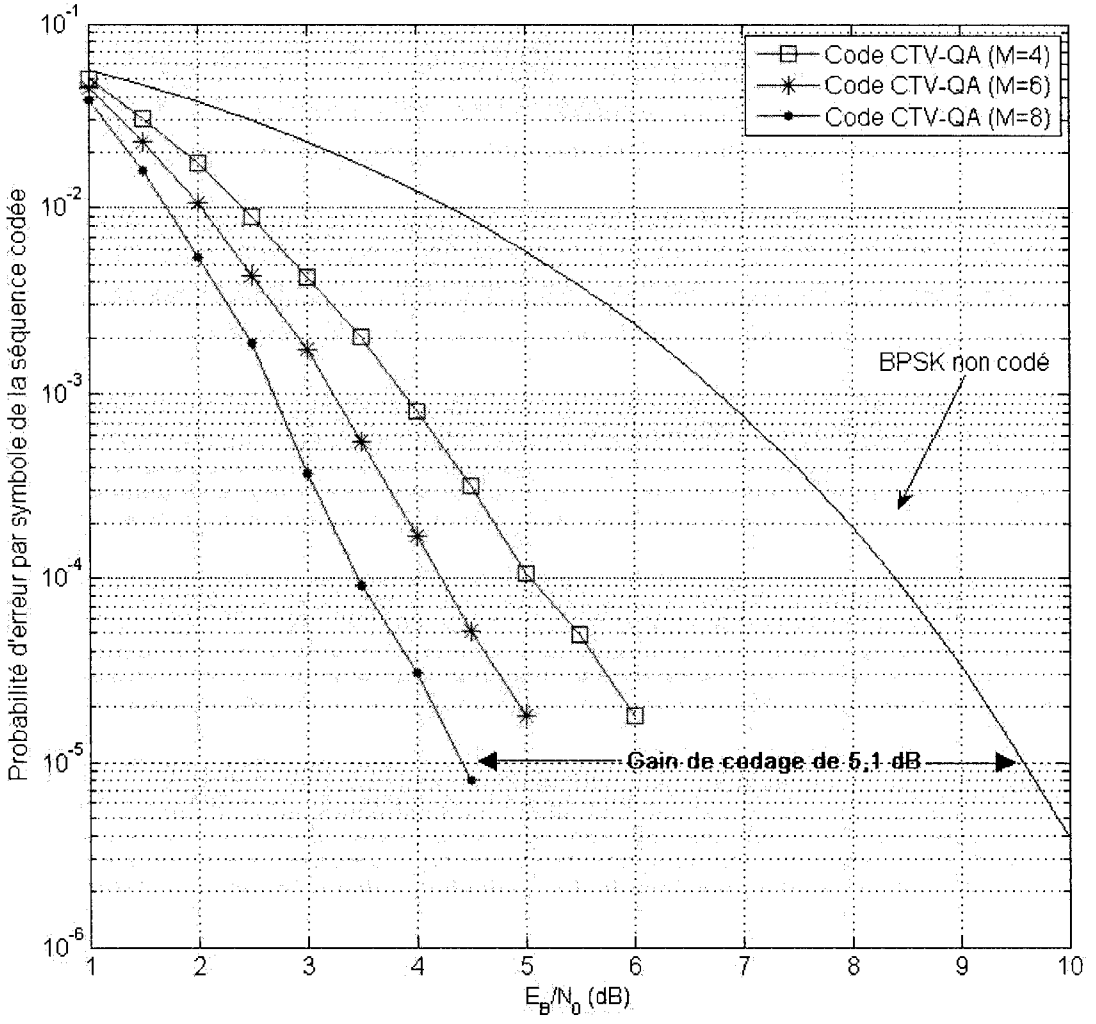

Figure 4.11: *Performances d'erreur de codes CTV-QA, T=5000, R=1/2, quantification douce à 3 bits*

Les performances d'erreur sont obtenues en comparant la séquence codée transmise initiale et celle estimée après le décodeur de Viterbi adaptatif. Le programme n'inclut pas le dernier module de la figure 4.9 qui transforme le *mot de code estimé* en une

séquence d'information estimée. Cette opération est censée, du moment où le codeur n'est pas catastrophique, retranscrire avec exactitude la *séquence d'information estimée* [28]. Nous pouvons donc ignorer cette étape à condition que nous prenions bien en compte le fait que la complexité du système a réellement augmenté en comparaison avec les codes systématiques. La figure 4.11 illustre les performances d'erreur de codes CTV-QA générés par des codeurs de mémoire 4, 6 et 8. Le tableau 4.2 relate les ensembles des connexions sous forme octale sur lesquels chaque codeur varie de manière pseudo-aléatoire sur une période $T = 5000$.

Tableau 4.2 : Ensembles des connexions sur lesquels varie le codeur CTV-QA de taux de codage R=1/2

M	Ensembles des connexions
4	{[25 23], [27 24], [21 36], [23 35], [37 33], [33 27], [21 33], [31 35], [33 35], [27 25], [33 37], [21 33], [25 31], [21 26], [33 31]}
6	{[151 117], [137 167], [165 136], [125 104], [177 131], [161 123], [131 134], [177 115], [133 171], [143 164], [171 126], [117 172], [157 122], [175 115], [145 113], [117 172], [157 135]}
8	{[521 427], [661 533], [677 575], [457 421], [673 465], [561 753], [607 571], [737 453], [433 715], [513 401], [611 477], [565 613], [547 525], [403 627], [735 473], [703 421], [661 453], [717 471], [561 755], [633 411], [611 523]}

Nous constatons que les performances d'erreur sont supérieures à celles des codes systématiques. Le gain de codage obtenu est de 5.1 dB pour une probabilité de bit en erreur de 10^{-5}, pour un code de longueur de contrainte K=9. Les performances de ces derniers restent malheureusement légèrement en dessous des meilleures performances des codes convolutionnels fixes pour une même longueur de contrainte [58]. La raison essentielle à cela est que la complexité de l'algorithme de Viterbi adaptatif augmente de manière exponentielle avec la mémoire du codeur et oblige donc le codeur a disposé

d'une mémoire relativement faible. Ce qui restreint considérablement l'ensemble sur lequel le code est choisit de manière aléatoire et affaiblit considérablement la composante aléatoire du système de codage proposé dans ce mémoire. Les résultats obtenus sont le fruit de l'optimisation de plusieurs paramètres tels que la période T, le nombre de connexions différentes sur lesquelles le codeur varie ou encore le seuil ψ.

Les résultats présentés viennent confirmer le potentiel qu'offre la nouvelle technique de codage proposée dans ce mémoire. Malgré que les longueurs de contrainte soient relativement faibles, un gain de codage substantiel a été obtenu. Ce gain ne dépasse pour le moment pas celui obtenu par les codes convolutionnels fixes avec décodage de Viterbi qui génèrent les meilleures performances pour une même longueur de contrainte [58] et n'est cependant pas suffisant, pour rivaliser avec les techniques de codages modernes tels que les codes Turbo ou LDPC. Même si les codes CTV-QA renferment une structure probabiliste, nous pensons que la distance minimale des codes CTV-QA simulés reste faible. Puisque la distance minimale des codes convolutionnels à temps variant est apparentée à celle des codes convolutionnels fixes de même longueur de contrainte [43] et que pour des codes convolutionnels fixes de longueur de contrainte comprise entre 5 à 9, la distance minimale des meilleurs d'entre eux est comprise respectivement entre 5 et 12 [57]. Une façon simple d'augmenter la distance minimale est de simuler des codes de mémoires plus grandes. Seulement, comme mentionné précédemment, la complexité de l'algorithme de Viterbi augmente de manière exponentielle avec la mémoire du codeur. Le simulateur développé pour cette recherche est le premier en son genre, et reste un prototype. Le temps de simulation devient ingérable pour des codeurs de mémoire supérieure à 8.

Afin d'augmenter la distance minimale des codes CTV-QA sans pour autant augmenter la longueur de contrainte de ces derniers, la simulation des codes de taux de codage 1/3 est présentée. La réduction du taux de codage a cependant quelques désavantages.

Hormis le fait que la complexité au décodage ait augmenté, ce procédé requiert une augmentation significative de la largeur de bande.

4.7.3 Simulation des codes CTV-QA de taux de codage 1/3

La figure 4.12 illustre le codeur que nous utilisons pour la simulation des codes CTV-QA de taux de codage R=1/3 et de mémoire M=6.

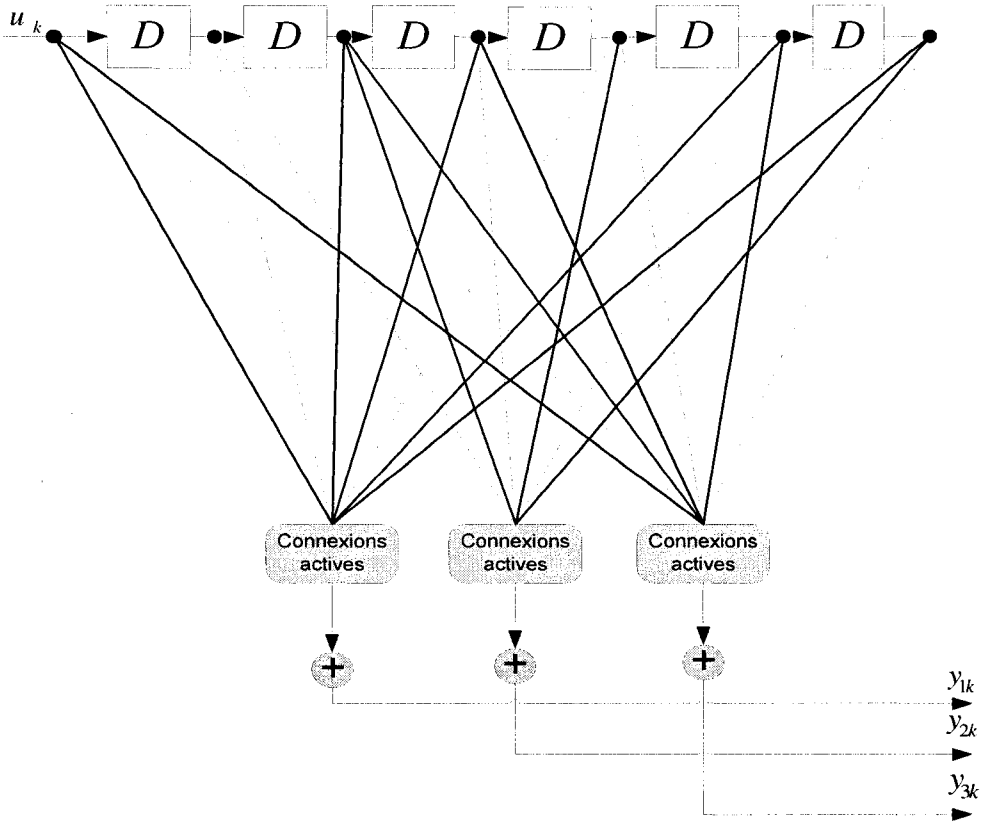

Figure 4.12: *Codeur CTV-QA non systématique, R =1/3, M=6*

Les ensembles des connexions sur lesquels le codeur CTV-QA varie sont normalement choisis à l'aide d'un prétraitement décrit dans la section 4.5.2.1. Néanmoins, après plusieurs simulations, nous avons remarqué que pour les codes de mémoire 8 il n'y a pas

d'intérêt réel à choisir certains ensembles de connexions particuliers plutôt que d'autres. Au contraire, le fait que le codeur varie sur une très large gamme de connexions différentes améliore légèrement les performances. La figure 4.13 illustre les performances d'erreur des codes CTV-QA de taux de codage 1/3.

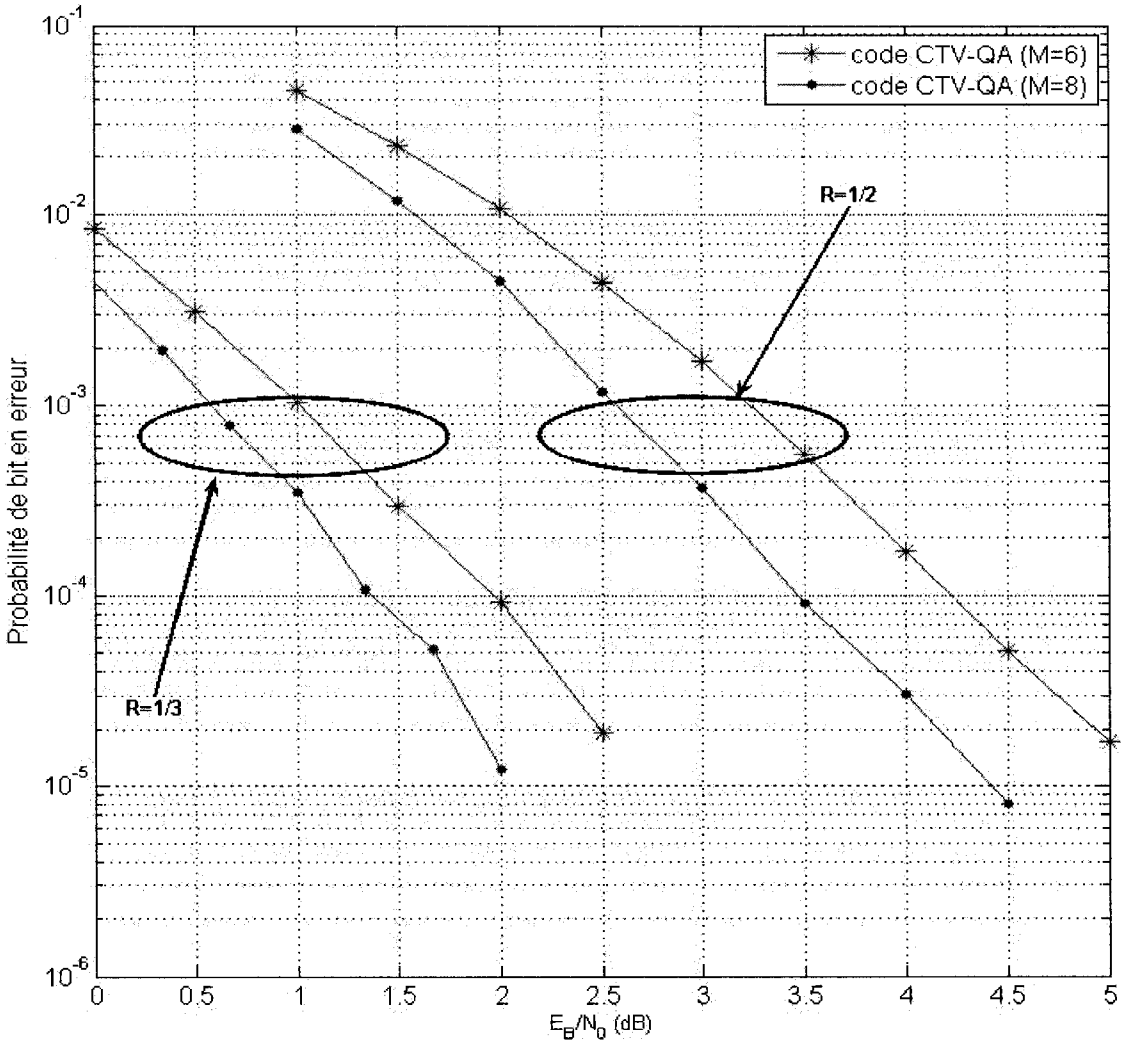

Figure 4.13: Comparaison entre les codes CTV-Q-RS, R=1/3, M=6 et M=8, T=10000 et les codes convolutionnels fixes , R=1/2, M=6 et M=8, quantification douce à 3 bits

La figure 4.13 indique que les performances d'erreur des codes CTV-QA de taux de codage 1/3 sont nettement supérieures à celles des codes CTV-QA de taux de codage 1/2. Le fait que la distance minimale des codes ait augmenté est sans doute la cause de cette amélioration. L'amélioration des performances d'erreur implique cependant une augmentation de la largeur de bande de 33,33% [57]. Les performances d'erreur des codes CTV-QA de taux de codage 1/3 sont malheureusement tout de même inferieures à celles des meilleurs codes convolutionnels fixes de taux de codage 1/3 [48]. Le tableau 4.3 relate les ensembles des connexions sous forme octale sur lesquels le codeur de mémoire 6 varie de manière pseudo aléatoire sur une période $T = 10000$.

Tableau 4.3 : Ensembles des connexions sur lesquels varie le codeur CTV-QA de taux de codage R=1/3

M	Ensembles des connexions
6	{[145 157 121], [105 135 147], [157 135 166], [175 121 163], [147 113 175], [131 103 177], [163 155 131], [117 125 155], [123 157 105], [127 135 143], [1113 127 142], [171 101 113], [161 123 163], [157 177 105],}
8	Plus de 7000 ensembles de connexions non catastrophiques.

Nous avons remarqué que le fait de supprimer le prétraitement sur le choix des ensembles des connexions avait un léger impact positif sur les performances du code de mémoire 8. Le nombre d'ensembles de connexions sur lesquels notre codeur varie ayant augmenté, implique que l'ensemble des codes sur lesquels notre code est choisi a lui aussi augmenté. Il semblerait donc, qu'il y'ait un conflit de priorité entre l'importance de la distance minimale du code et l'impact de la structure aléatoire de notre codage. Il est possible qu'en dessous d'une certaine distance minimale, cela n'a pas énormément d'intérêt de chercher à créer un système de codage avec une structure aléatoire.

4.8 Conclusion

Dans ce chapitre nous avons présenté un système de codage basé sur la permutation des connexions d'un codeur convolutionnel, sur une longue période de façon pseudo aléatoire, utilisant un algorithme de Viterbi adaptatif comme décodeur. L'objectif était de proposer un système de codage pratique, bénéficiant des avantages de l'algorithme de Viterbi tout en ajoutant de l'aléa par rapport aux codes convolutionnels classiques.

Les performances d'erreur des codes convolutionnels à temps variant quasi apériodiques systématiques et non systématiques ont été présentées. Nous n'avons pour le moment pas trouvé de codes CTV-QA dont les performances d'erreurs sont supérieures à celles des codes convolutionnels qui génèrent les meilleures performances pour une même longueur de contrainte [58]. La raison essentielle à cela, est la limitation du système lié à la complexité au décodage des codes CTV-QA, qui oblige le codeur à disposer d'une mémoire relativement faible et restreint considérablement l'ensemble de codes sur lequel notre code particulier est choisi de manière aléatoire. Les codes non systématiques se sont révélés être, en termes de performance, supérieurs aux codes systématiques. Cependant, nous avons remarqué que la longue période des codes proposés dans notre recherche augmente de manière significative la complexité (temps de calcul) au décodage lorsqu'il s'agit d'un système de codage non systématique.

L'objectif premier de la recherche était de bénéficier des avantages de l'algorithme de Viterbi tout en introduisant de l'aléa aux codes convolutionnels classiques. Le fait que le la complexité (temps de calcul) au décodage des codes CTV-QA non systématique soit beaucoup plus élevée que celle des codes convolutionnels fixes équivalents dénature de fait, un peu notre quête initiale. Dans l'objectif entre autres, de réduire la complexité au décodage tout en proposant un système de codage compétitif en termes de performances,

nous introduirons dans le prochain chapitre les codes convolutionnels à temps variant quasi apériodiques systématiques récursifs.

CHAPITRE 5

INTRODUCTION AUX CODES CONVOLUTIONNELS À TEMPS VARIANT QUASI APÉRIODIQUES RÉCURSIFS SYSTÉMATIQUES

5.1 Introduction

Nous présentons dans ce chapitre, les codes convolutionnels à temps variant quasi apériodiques récursifs systématiques (CTV-QA-RS). Dans le chapitre précédent, il a été montré que les codes CTV-QA non systématiques sont beaucoup plus complexes que les codes convolutionnels fixes non systématiques. Dans le cas des codes convolutionnels systématiques (CTV-QA ou fixe), le décodage de Viterbi (adaptatif ou pas) permet de retrouver directement la séquence d'information. De fait, la complexité au décodage des codes CTV-QA systématiques est nettement inférieure à celle des codes CTV-QA non systématiques, car comme nous l'avons vu dans le chapitre précédent, dans le cas des codes CTV-QA non systématiques, c'est le module qui se charge de retrouver à partir de la séquence décodée, la séquence d'information, qui augmente la complexité au décodage de manière considérable. Les codes CTV-QA-RS sont par définition systématiques, ils ont donc une complexité (temps de calcul) assez proche de celle des codes convolutionnels fixes. Puisque la complexité au décodage des codes CTV-QA-RS est réduite à la complexité de l'algorithme de Viterbi adaptatif, qui, en termes de calcul, est très proche de l'algorithme de Viterbi classique.

Dans le chapitre 3, nous avons vu que les codeurs qui génèrent des codes systématiques sont non catastrophiques. Par conséquent, les codeurs CTV-QA-RS ne sont eux aussi, pas catastrophiques.

Dans un premier temps, les propriétés des codes CTV-QA-RS qui diffèrent de celles des codes CTV-QA non récursifs sont définies, ensuite à l'aide de simulations, nous

analysons les performances d'erreur de certains codes CTV-QA-RS de taux de codage R=1/2 et nous les comparons avec les performances d'erreur des codes convolutionnels fixes utilisant l'algorithme de Viterbi, étant donné que la complexité au décodage des codes convolutionnels fixes et des codes CTV-QA-RS est du même ordre.

5.2 Propriétés des codes CTV-QA-RS

Un codeur CTV-QA-RS est un codeur CTV-QA systématique auquel il a été ajouté une boucle de retour fixe. La figure 5.1 illustre le schéma d'un codeur CTV-QA-RS de mémoire M=4.

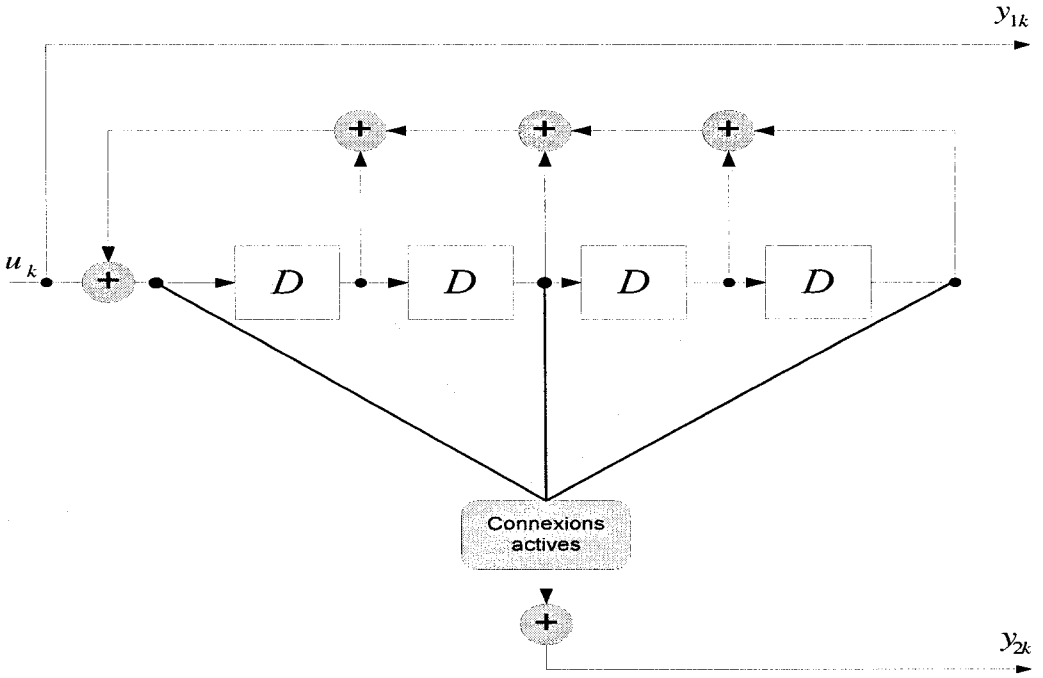

Figure 5.1: *Codeur CTV-QA-RS, R=1/2, M=4*

Comme dans le cas des codes CTV-QA non récursifs, les modules *connexions actives* au-dessus des additionneurs modulo-2 ainsi que les connexions en pointillés de la figure 5.1 indiquent que les connexions de chaque modulo-2 additionneur sont différentes à

chaque instant d'une période T. Les matrices des connexions $\boldsymbol{G}_j^{TV}$, $j=1,2,..n$ définies dans le chapitre précédent ne sont pas bien adaptées à la définition des codes CTV-QA-RS. La boucle de retour implique que la matrice génératrice sous forme polynomiale du code fixe dont les connexions correspondent aux connexions du codeur CTV-QA-RS à un instant donné peut être semi-infinie. De fait, il est plus simple de caractériser les codes CTV-QA-RS à l'aide d'une suite de termes écrits commodément sous forme octale :

$$\{[A,a_{1,1},a_{2,1},....a_{n-1,1}]_1,[A,a_{1,2},a_{2,2},....a_{n-1,2}]_2.....[A,a_{1,T},a_{2,T},....a_{n-1,T}]_T\} \qquad (5.1)$$

Chaque élément de (5.1) correspond aux connexions du codeur à un instant donné de la période T. La lettre A donne une information sur les connexions constituant la boucle de retour, cette dernière, représente le nombre octal équivalent au nombre binaire $(x_1,x_2,x_3.....x_K)$, $x_j \in \{0,1\}$, où $x_j=1$ si la sortie à la j^{eme} unité de retard est connectée à la boucle de retour et $x_j=0$ dans le cas contraire. Les éléments $a_{i,j}$ sont définis comme les vecteurs des connexions $\boldsymbol{G}_i^{\,j}$, $j=1,2,..T$, $i=1,2,...n-1$ correspondant aux n-1 symboles de parité définis dans les chapitres 3 et 4. Par exemple, dans le cas de la figure 5.1, A=37, ce qui correspond au vecteur binaire (11111), puisque la boucle de retour est connectée à la sortie de toutes les unités de retard.

Dans le chapitre 3, nous avons vu que l'une des particularités des codes convolutionnels récursifs est de pouvoir générer un code de longueur infinie avec une séquence d'information finie. Cependant, nous avons constaté que la séquence infinie générée par le codeur récursif contenait une période. Dans le cas d'un codeur CTV-QA-RS, la récursivité implique que la séquence codée est infinie. Seulement, la période est très longue, de l'ordre de T. Un seul bit non nul d'information peut générer donc une séquence codée quasi apériodique.

5.3 Simulation des codes CTV-QA-RS de taux de codage 1/2

Le codeur utilisé pour les simulations des codes CTV-QA-RS de mémoire *M=4* est celui de la figure 5.1. À chaque instant donné de la période *T*, des connexions différentes sont choisies de manière pseudo aléatoire à l'aide d'un procédé identique à celui du chapitre précédent décrit dans la section 4.5.2. Le décodage utilisé est l'algorithme de Viterbi adaptatif illustré dans le chapitre précédent. La figure 5.2 illustre les performances

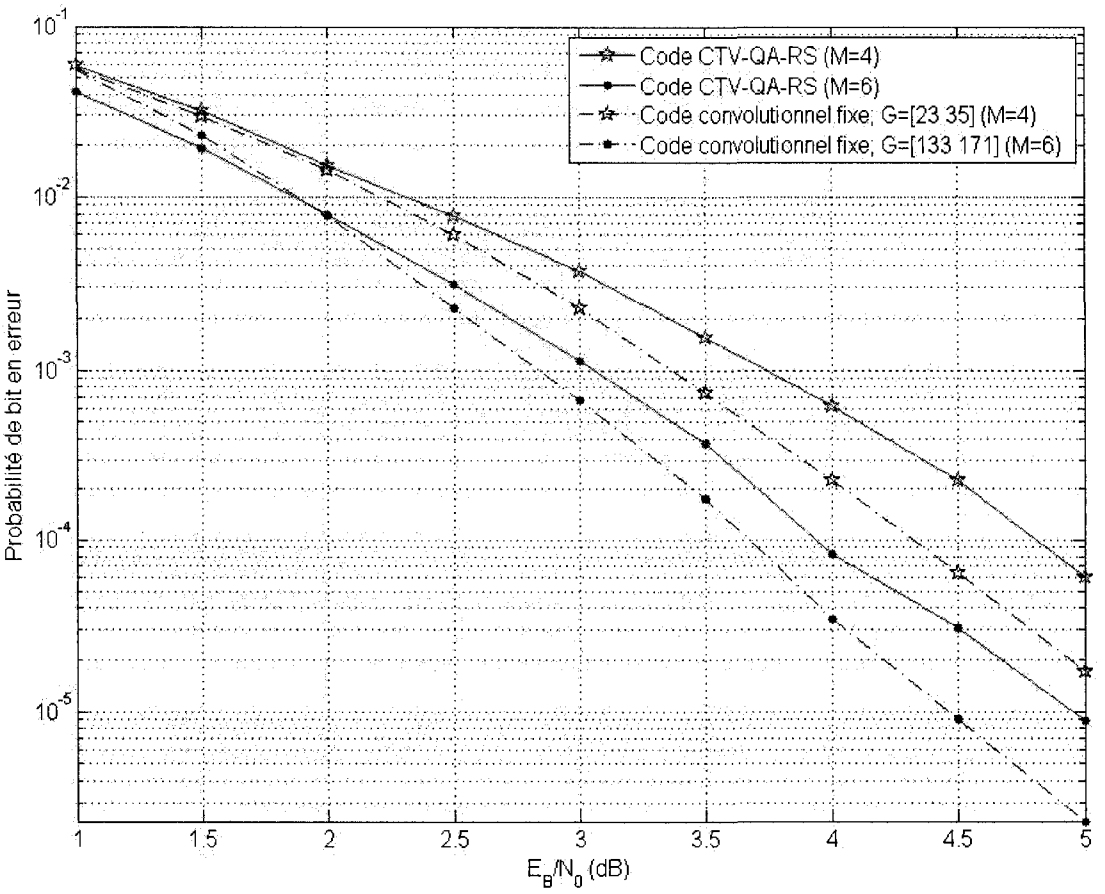

Figure 5.2: *Comparaison entre les codes CTV-Q-RS, R=1/2, M=4 et M=6 T=10000 et les codes convolutionnels fixes , R=1/2, M=4 et M=6, quantification douce à 3 bits*

d'erreur des codes CTV-QA-RS et les compare avec les codes convolutionnels qui génèrent les meilleures performances pour une même longueur de contrainte [58].

Nous constatons que les performances des codes CTV-QA-RS sont à peu prés équivalentes à celles des codes CTV-QA non récursifs (figure 4.11) mais demeurent légèrement en dessous de celles des codes convolutionnels qui génèrent les meilleures performances pour une même longueur de contrainte (figure 5.2). Les ensembles des connexions sous forme octale sur lesquels le codeur varie de manière pseudo aléatoire sur une période $T = 10000$ sont données dans le tableau 5.1.

Tableau 5.1 : Ensembles des connexions sur lesquels varie le codeur CTV-QA-RS de taux de codage R=1/2

M	Ensembles des connexions
4	{[37 31], [37 27], [37 21], [37 22], [37 33], [37 22], [37 26]}
6	{[177 131], [177 155], [177 135], [177 127], [177 153], [177 115], [177 131], [177 133], [177 132], [177 161]}

Les codes que nous avons simulés sont définis de la manière où chaque élément de la suite de T éléments de (5.1) représente un des ensembles des connexions du tableau 5.1 pour une mémoire donnée.

Les codes convolutionnels fixes que nous avons simulés sont des codes non systématiques. Les codes systématiques récursifs ou non sont connus pour avoir des performances inférieures aux codes non systématiques [59]. Donc bien que les performances obtenues par les codes CTV-QA-RS soient légèrement inférieures à celles des codes convolutionnels, il reste cependant important de prendre en compte le fait que la composante systématique du code a tendance à dégrader les performances d'erreur.

5.4 Conclusion

Dans ce chapitre, nous avons introduit une nouvelle classe de codes intitulée : *codes convolutionnels à temps variant quasi apériodiques récursifs systématiques.* Ces codes sont basés sur la permutation des connexions d'un codeur convolutionnel sur une longue période dans le temps, où les registres du codeur sont connectés à une boucle de retour. De bonnes performances d'erreur ont été obtenues. Cependant, nous n'avons pas réussi à trouver un code CTV-QA-RS qui surpasse, pour une même longueur de contrainte et pour un même taux de codage, le meilleur des codes convolutionnels fixes existants. La raison à cela est la même que celle qui a été évoquée dans le chapitre précédent, à savoir que les limitations pratiques provenant de la complexité du système obligent le codeur a disposé d'une mémoire relativement faible. La complexité (temps de calcul) au décodage des codes CTV-QA-RS est équivalente à celle des codes convolutionnels fixes. Nous avons vu dans le chapitre précédent que le décodage des codes CTV-QA non récursifs peut s'avérer très complexe, car il implique des manipulations de matrices de très grandes tailles, relatives à la longueur de la période T. Nous avons également remarqué que les performances d'erreur des codes CTV-QA-RS sont assez proches de celle des codes CTV-QA non récursifs. Ainsi, il est possible de conclure que les codes CTV-QA-RS sont nettement plus compétitifs que les codes CTV-QA non récursifs non systématiques en termes de rapport entre performance et complexité.

Le fait que les recherches n'ont pas abouti à la découverte d'un code CTV-QA-RS qui surpasse le meilleur des codes convolutionnels fixes pour une même longueur de contrainte ne signifie pas qu'il sera impossible dans le futur de trouver un code CTV-QA-RS capable d'une telle prouesse. Effectivement, il est très possible que la simulation de codes CTV-QA-RS de mémoire plus grande permettra d'atteindre cet objectif. Même si l'augmentation de la mémoire reste limitée car la complexité de l'algorithme de Viterbi augmente de manière exponentielle avec la mémoire du codeur.

CHAPITRE 6

CONCLUSION

6.1 Bilan de recherche réalisé

Dans ce mémoire nous avons proposé une nouvelle technique de codage basée sur une construction aléatoire, dans l'objectif de profiter des avantages de l'algorithme de Viterbi tout en ajoutant de l'aléa par rapport aux codes convolutionnels classiques. Cela à été possible grâce d'une part, à la réalisation d'un codeur convolutionnel dont les connexions varient dans le temps de manière aléatoire et d'autre part, grâce à l'élaboration d'un algorithme de Viterbi adaptatif approprié au codeur. Nous avons présenté les codes récursifs et non récursifs et nous avons analysé le rapport entre performance et complexité de ces codes.

Les performances d'erreur obtenues ne sont pour le moment pas suffisantes pour justifier l'utilisation de ce système de codage dans des systèmes de communication actuels, car les performances des codes proposés n'ont pas réussi à surpasser les meilleurs codes convolutionnels fixes et de ce fait, il n'y a pas pour le moment de justification à leurs utilisations sur un plan pratique. La simulation des codes de taux 1/3 a révélé l'impact de la construction aléatoire du codage pour des codes de mémoire 8. Effectivement, dans ce cas précis, il n'y a pas d'intérêt réel à choisir certains ensembles de connexions particuliers plutôt que d'autres. Au contraire, le fait que le codeur varie sur une très large gamme de connexions différentes améliore légèrement les performances.

En termes de complexité au décodage, seuls les codes systématiques se sont révélés être compétitifs aux codes convolutionnels fixes. Nous pensons que ceux sont les codes

systématiques récursifs qui pourraient être dans des travaux futurs, compétitifs en termes de rapport performance contre complexité aux codes convolutionnels fixes. Étant donné que les performances d'erreur des codes récursifs systématiques sont nettement supérieures à celles des codes systématiques non récursifs et sont légèrement inférieures à celles des meilleurs codes convolutionnels fixes.

Nous pensons que la composante aléatoire des codes proposés dans ce mémoire est largement atténuée par les contraintes pratiques qui restreignent le codeur à disposer d'une mémoire très petite (inférieure à 8). Dans ce sens, nous envisageons dans la section suivante plusieurs améliorations futures possibles liés à ce problème.

6.2 Améliorations envisageables

Dans un premier temps, le moyen de rendre les nouveaux codes proposés dans ce mémoire plus performants est sans doute la mise en œuvre d'un codeur de mémoire plus grande. Malgré, que le temps de calcul au décodage augmente de manière exponentielle avec l'augmentation de la mémoire du codeur, celui mis en œuvre pour ce mémoire reste un prototype et il est possible de l'optimiser en le réalisant en matériel.

Deuxièmement, une analyse théorique approfondie ainsi que la recherche de bornes supérieures sur la probabilité d'erreur du système en fonction de la distribution des variations des connexions serait nécessaire. Cette analyse facilitera la recherche de bons codes.

Ensuite, il serait intéressant de tester les codes proposés dans des canaux à évanouissement. Effectivement, il se peut que la variation dans le temps ainsi que la structure aléatoire des codes proposés dans ce mémoire ajoute de la diversité et soit ainsi bénéfique pour ce genre d'application.

Nous pouvons envisager aussi une extension des codes convolutionnels à temps variant quasi apériodiques récursifs en faisant varier dans le temps les connexions qui sont reliés à la boucle de retour.

Finalement, nous pouvons imaginer un système de codage identique qui ferait varier les connexions d'un codeur convolutionnel sur une longue période, seulement cette fois le codeur serait de mémoire beaucoup plus grande et pourrait être ainsi décodé par un algorithme séquentiel. Malgré que le décodage ne soit plus optimal, nous pourrions bénéficier pleinement de la structure aléatoire du système de codage proposé dans ce mémoire.

RÉFÉRENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.,* vol. 27, pp. 379-423, Juillet, 1948.

[2] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.,* pt. 4, pp. 37-46, Mars, 1955.

[3] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transaction on Information Theory,* vol. IT-13, pp. 260-269, Avril, 1967.

[4] C. Berrou, A. Glavieux, P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo codes," *Proc. 1993 International Conference of Communication* (Genève), pp. 1064-1070, Mai, 1993.

[5] D. Divsalar, H. Jin, R.J. McEliece, "Coding theorem for 'turbo-like' codes," *Proceedings of the Allerton conference 1998,* (Allerton, IL), pp. 201-210, Septembre, 1998.

[6] L. Ping, K.Y. Wu, "Concatenated tree codes: a low-complexity, high-performance approach," *IEEE trans. Information Theory,* vol. 47, pp. 791-799, Février, 2001.

[7] A. Abbasfar, D. Divsalar, Y. Kung, "Accumulate-repeat-accumulate codes," *GLOBECOM 2004*, pp.509-513, Dallas, TX, USA, Décembre, 2004.

[8] R. Padovani, "Ten years of progress in CDMA," Viterbi Conference, Univ. So. Calif., Los Angeles, Mars, 2005.

[9] G. D. Forney, Jr., "The Viterbi algorithm," *Proceedings IEEE*, vol.61, pp. 268-278, Mars, 1973.

[10] D. J. Costello, Jr. G. D. Forney, Jr., "Channel Coding: The Road to Channel Capacity," *Proceedings IEEE,* vol. 95, pp. 1150-1177, Juin, 2007.

[11] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.,* vol. 29, pp. 147-160, 1950.

[12] M. J. E. Golay, "Notes on digital coding," *Proceedings IRE.,* vol. 37, pp. 657, Juin, 1949.

[13] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *IRE Trans. Electron. Comput.,* vol. EC-3, pp. 6-12, Septembre, 1954.

[14] R. C. Bose, D. K. Ray-Chaudhuri, "On a class of error-correcting group codes," *Information Control,* vol. 3, pp. 68-79, Mars, 1960.

[15] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres.,* vol. 2, pp. 147-146, 1959.

[16] I. S. Reed, G. Solomon, "Polynomial codes over certain finite fields," *J. SIAM.,* vol. 2, pp. 300-304, Juin, 1960.

[17] R. G. Gallager, *Low density parity-check codes*. Cambridge, MA: MIT press, 1962.

[18] D. Tse, P. Viswanath, *Fundamental of wireless communication.* Cambridge: University Press, Mai 2005.

[19] D. J. Costello, Jr., J. Hagenauer, H. Imai, S. B. Wicker, "Applications of error-control coding," *IEEE Transaction on Information Theory,* vol. 44, pp. 2531-2560, Octobre, 1998.

[20] H. Nyquist, "Cetain factor affecting telegraph speed," *Bell Syst. Tech. J.,* vol. 3, pp. 324-346, 1923.

[21] Y. Louët, A. Le Glaunec, P. Leray, "A soft decision decoding of product BCH and Reed-Muller codes for error control and peak-factor reduction in OFDM," *Proceedings of IEEE CSCC 2000,* Athens, Juillet, 2000.

[22] C. Berrou, R. Pyndiah, P. Adde, C. Douillard, R. Le Bidan, "An Overview of Turbo Codes and Their Applications," *The European Conference on Wireless Technology 2005,* pp.1-9, Paris, France, Octobre, 2005.

[23] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE Transaction on Information Theory,* vol. IT-4, pp. 38-49, Septembre, 1954.

[24] J. M. Wozencraft, B. Reiffen, *sequential decoding,* Cambridge, MA: MIT Press, 1961.

[25] J. L. Massey, *Threshold Decoding,* Cambridge, MA, MIT Press, 1963.

[26] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Transaction on Information Theory,* IT-9, pp. 64-74, Avril, 1963.

[27] F. Jelinek "A Fast Sequential decoding using a stack," *IBM Journal of Research and Development,* Vol. 13, pp 675-685, Novembre, 1969.

[28] G. D. Forney, Jr, "Convolutional codes I: Algebraic structure," *IEEE Transaction on Information Theory,* IT-6 , pp. 720-738, Novembre, 1970.

[29] D. Haccoun, C. Cardinal, F. Gagnon, "Search and determination of convolutional self-Doubly orthogonal codes for iterative threshold decoding," *IEEE Transactions on Communications,* vol. 53 p. 802-809, Fevrier 2005.

[30] D. J. C. MacKay, R.M. Neal, "Near Shannon limit performance on Low Density Parity Check Codes," *Electronic Letter,* vol. 32, pp. 1645-1648, Aout, 1996.

[31] D.A Spielman, "Linear-time encodable and decodable error correcting codes," *IEEE Transaction on Information Theory,* vol 42, pp. 1710-1722, Novembre, 1996.

[32] R. M. Tanner, "A Recursive Approach of Low Complexity Codes," *IEEE Transaction on Information Theory,* IT-27, pp. 533-547, Septembre, 1981.

[33] N. Wiberg, "Codes and decoding on general graphs," Ph.D Thesis, U. Linkoping, Suède, 1994.

[34] B. Bollobás, *Modern Graph Theory,* New York, NY: Springer-Verlag, 1998.

[35] A. Reznik, *Iterative decoding of codes defined on graphs*, M.Sc. thesis MIT, Cambridge, MA, Mai, 1998.

[36] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Transaction on Information Theory*, vol.47, pp. 585–598, Fevrier, 2001.

[37] S.Y. Chung, G. D. Forney, T. J. Richardson, R. Urbanke, "On the design of low density parity check codes within 0.0045 dB of the Shannon limit," *IEEE Communication Letter,* vol. 5, pp. 58-60, Fevrier, 2001.

[38] I. Chatzigeorgiou, M. R. D. Rodrigues, I. J. Wassell, R. Carrasco, "Punctured Binary Turbo-Codes with Optimized Performance," *VTC'05*, Dallas, Texas, USA, Septembre, 2005.

[39] G. D. Forney Jr., "Codes on graphs: normal realizations," *IEEE Transaction on Information Theory,* vol. 47, pp. 520-548, Février, 2001.

[40] L. Bahl, J. Cocke, F. Jelinek, J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate ," *IEEE Transactions on Information Theory,* vol. 20, pp. 284-287, Mars, 1974.

[41] G. D. Forney Jr, "Codes on graphs: recent progress," *Physica A*, vol. 302, Issue 1-4, p. 1-13, Décembre, 2001.

[42] P. J. Lee, "There are many good periodically time-varying convolutionnal codes," *IEEE Transaction on Information Theory,* vol. 35, pp. 460-463, Mars, 1989.

[43] M. Mooser, "Some periodic convolutional codes are better than any fixe codes," *IEEE Transaction on Information Theory,* vol. IT-29, pp. 750-751, Mars, 1983.

[44] J. L. Massey, "Error bound for tree codes, trellis codes and convolutional codes with encoding and decoding procedures," *Coding and complexity,* G. Longo, Ed., C.I.S.M. Courses and Lectures No 216: New York: pp 1-57, Springer-Verlag, 1974.

[45] A. J. Felstrom, K. S. Zigangirov, "Time-varying periodic convolutional codes with low density parity check matrix," *IEEE Transaction on Information Theory,* vol. 45, pp. 2181-2191, Septembre 1999.

[46] G. Battail, "A conceptual framework for understanding turbo codes," *IEEE Journal on. Selected Areas in Communications*, vol. 16, No. 2, Fevrier, 1998.

[47] G. Shulman, M. Feder, "Imptoved error exponent for time-invariant and periodically time-variant convolutional codes," *IEEE Transaction on Information Theory*, vol. 46 pp 97-103, Janvier, 2000.

[48] J. Conan, "The weight spectra of some short low-rate convolutional codes," *IEEE Transaction on Communication,* vol.32 pp 1050-1053, Septembre, 1984.

[49] J. P. Odenwalder, "Optimum decoding of convolutional codes," Ph.D. Dissertation, University of California at Los Angeles, 1970.

[50] J. L. Massey, M. K. Sain, "Inverses of linear sequential circuits, " *IEEE Trans. Computers*, vol. C-17, pp. 330-337, Avril 1968

[51] K J. Hole, "An Algorithm for determining if a rate (n- l)/n punctured convolutional encoder is catastrophic," *IEEE Trans. Communication*, vol. 39, pp. 386-389, Juillet 1991.

[52] D. Lavoie, D. Haccoun, "On the determination of catastrophic convolutional encoders" *IEEE International Symposium on Information Theory* Trondheim, Norvège, Juin, 2000.

[53] C. O'Donoghue, C. Burkley, "Catastrophicity test for time-varying convolutional encoders" *Cryptography and Coding*: 7th IMA International Conference, Cirencester, UK, Décembre 1999.

[54] A. J. Viterbi, "A personal history of the Viterbi algorithm," *IEEE Signal processing Magazine,* vol.23 pp 120-142, Juillet, 2006.

[55] G. D. Forney Jr, "Viterbi algorithm: A personal history" [arXiv.cs /0504020v2].

[56] R. Azencott "Grandes déviations et applications" *École d'Été de Proba. de Saint Flour VIII 1978*, Lect. Notes in Math. 774, pp. 1-76, Springer-Verlag, Berlin, 1980.

[57] J. M. Wozencraft, I. M. Jacobs, *Principles of communication engineering*, Wiley, New York, 2006.

[58] J. Chang, D. Hwang, M. Lin, "Some extended results on the search for good convolutional codes," *IEEE Transaction on Information Theory,* vol. 43 pp 1682-1697, Septembre, 1997.

[59] J. C. Moreira, P. G. Farrell, *Essentials of error-control coding*, Wiley, New York, 2006.

[60] S. Kullback, *Information theory and statistics*, Wiley, New York 1959.

[61] J. Ould-Cheikh-Mouhamedou, S. Crozier, P. Kabal D. , "Efficient distance measurement method for Turbo codes that use structured interleavers," *IEEE Communication Letters,* vol. 10 pp 477-479, Juin, 2006.

Annexe I

Loi faible des grands nombres et borne de Chernoff

I.1 Loi faible des grands nombres

Si l'on considère N variables aléatoires indépendantes définies sur un même espace probabilisé, ayant même variance finie et même espérance notée *E(X)*, la loi faible des grands nombres stipule que, pour tout réel ε strictement positif, la probabilité que la moyenne empirique $Y_N = \frac{Y_1 + Y_2 + ...Y_N}{N}$ s'éloigne de l'espérance de plus de ε, tend vers 0 lorsque $N \to \infty$.

I.2 Borne de Chernoff

Soit S_N, la somme de N variables aléatoires $X_1, X_2, ...X_N$ uniformément et identiquement distribuées avec une moyenne $\overline{X} = E_X[X]$. Pour tout $\tau > \overline{X}$, la probabilité que $S_N \geq N\tau$ est borné supérieurement par :

$$P_r\{S_N \geq N\tau\} \leq e^{-NE_c(\tau)} \tag{I.1}$$

où *l'exposant de Chernoff* $E_c(\tau)$ est donné par :

$$E_c(\tau) = \max_{s \geq 0} s\tau - \mu(s) \tag{I.2}$$

lorsque $\mu(s) = \log E_X[e^{sX}]$.

Annexe II

Quantification du canal

Dans le cas d'un décodeur de Viterbi idéal, les symboles codés corrompus par le bruit à la sortie du canal peuvent prendre n'importe quelle valeur dans □ . Seulement en pratique, ces derniers sont quantifiés en un nombre fini de valeurs discrètes afin de réduire la complexité de l'algorithme de Viterbi.

II.1 Quantification dure

Dans le cas d'une quantification dure, les symboles codés corrompus par le bruit à la sortie du canal peuvent prendre deux valeurs :

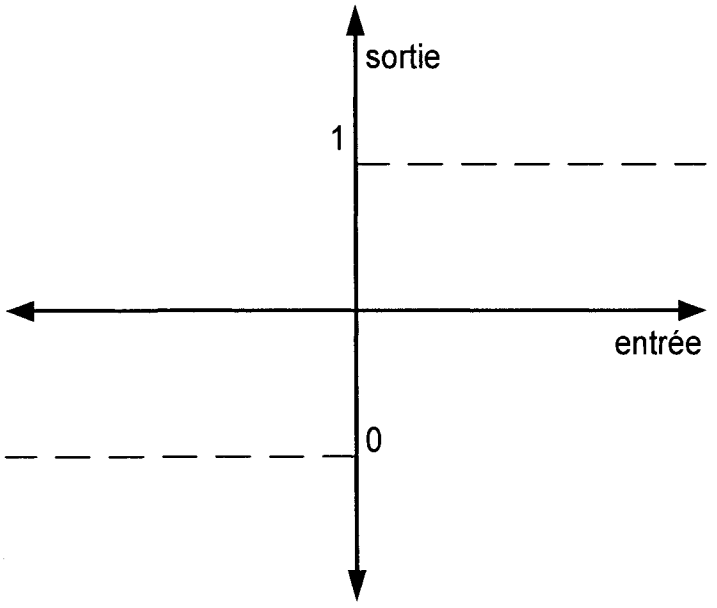

Figure II.1 : *Schéma d'un quantificateur dure*

II.2 Quantification douce

Dans le cas de la quantification douce à λ bits, les symboles codés corrompus par le bruit à la sortie du canal vont être transformés par un quantificateur en 2^{λ} valeurs discrètes (ou binaires) lorsque $\lambda \geq 2$.

Afin d'illustrer un exemple de quantification douce à 3 bits, nous allons supposer que les composantes du vecteur transmis à l'entrée du canal vectoriel sont $\{\pm\alpha\} = \{\pm 1\}$ et que le bruit ajouter par le canal ait une moyenne nulle et une variance égale à $N_0 / 2$. La figure II.2 illustre un quantificateur doux à 3 bits.

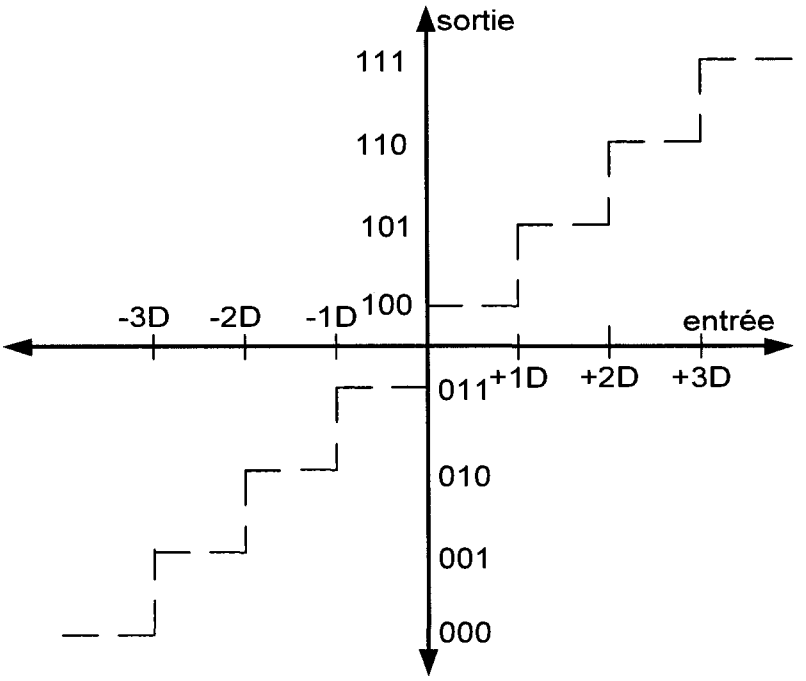

Figure II.2 : *Schéma d'un quantificateur doux à 3 bits*

où D est égal à :

$$D = 0,5\sqrt{\frac{1}{2(E_S / N_0)}} \qquad \text{(II.2)}$$

lorsque E_S représente l'énergie par symbole.