RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Amar Telidji - Laghouat

Faculté de Technologie

THÈSE DE DOCTORAT EN SCIENCES

Spécialité : Télécommunications

CHELLALI Sefouane

THÈME

Contribution à une amélioration en performances des codes correcteurs d'erreurs dans des systèmes de communication par satellite

JURY :

M. BENTRIA Bachir	Professeur UATL	Président du jury
M^{me} CHOUIREB Fatima	Maitre de conférences A UALT	Directrice de thèse
M. Adda Ali Pacha	Professeur USTO	Examinateur
M. HADJ HABDERRAHMANE Lahcène	Directeur de recherche CDS-ASAL	Examinateur
M. OUCHAR Ali	Professeur UATL	Examinateur

Résumé

La communication numérique par satellite est en forte expansion. Ainsi, le codage du canal est devenu un élément indispensable puisqu'il permet d'assurer la protection de l'information transmise en contrôlant les erreurs de transmission. Dans cette thèse, le concept des turbocodes classique est abordé dans le but d'améliorer l'efficacité de puissance dans le cas des systèmes de communications par satellites. La structure d'un turbocodeur classique est réalisée en faisant une concaténation parallèle des deux codeurs convolutifs systématiques récursifs, séparés par un entrelaceur des bits d'informations. La structure d'un turbodécodeur itératif est réalisée par deux décodeurs composants liés par des entrelaceurs dans une structure semblable à celle de turbocodeur. Ces décodeurs composants utilisent des algorithmes de décodage d'entrée soft et de sortie soft (notion d'information intrinsèque et d'information extrinsèque) comme c'est le cas de l'algorithme MAP ou SOVA.

Les recherches actuelles sur les turbocodes s'intéressent à l'optimisation des entrelaceurs (interleavers) et à l'optimisation des algorithmes de décodage du canal, parties intégrantes des turbocodes. Ce travail optimise les entrelaceurs par un nouveau modèle mathématique proposé. Ce modèle est basé sur la dimension d'entrelaceur, un nouveau concept développé.

Abstract

Digital satellite communication is rapidly expanding. Thus, the channel coding has become an essential element since it enables the protection of the information transmitted by controlling the transmission errors. In this thesis, the concept of traditional turbo codes is addressed in order to improve power efficiency in the case of satellite communication systems. The structure of a conventional turbo encoder is achieved by parallel concatenation of two recursive systematic convolutional encoders separated by an interleaver of information bits. The structure of an iterative turbo decoder is carried out by two sub-decoders bound by interleavers in a structure similar to that of the turbo encoder. These sub-decoders use as soft input decoding algorithms and soft output (notion of intrinsic information and extrinsic information) as is the case of the MAP or SOVA algorithm.

The current research on turbo codes interested in optimizing interleavers and optimization of channel decoding algorithms, integral parts of the turbo code. This work optimizes interleavers by a new proposed mathematical model. This model is based on the interleaver dimension, a new developed concept.

ملخص

تشهد الاتصالات الرقمية عبر الأقمار الصناعية توسعا متزايدا. لذا، أصبح ترميز القناة عنصرا أساسيا في حماية المعلومات المرسلة عبرها. في هذه الاطروحة، تم تناول مفهوم الرموز النفاثة التقليدية لتحسين فعالية استهلاك الطاقة لدى أنظمة الاتصالات عبر الأقمار الصناعية. بنية المرمز النفاث التقليدي تتكون من مرمزين إلتفافيين آليين مربوطين على التوازي ومفروقين بمبعثر للمعلومات. بنية فاك الرموز النفاثة شبيهه ببنية المرمز النفاث مع استبدال المرمزات التحتية بفاكات رموز تحتية. فاكات الرموز التحتية تستعمل خوارزميات مرنة المدخلات والمخرجات (مفهوم المعلومة الداخلية والخارجية) كما هو الحال عند الخوارزميات SOVA وMAP.

الأبحاث الجارية حاليا حول الرموز النفاثة تهتم بتحسين المبعثرات وخوارزميات الفك. هذا العمل يهدف الى تحسين المبعثرات وذلك باقتراح نموذج رياضي جديد لبنيتها. هذا النموذج مؤسس على ما نسميه ببعد المبعثر، وهو مفهوم جديد مطور.

Table des matières

Liste des figures ..... **v**
Liste des tableaux ..... **vi**
Liste des abréviations ..... **vii**
Dédicaces ..... **viii**
Remerciements ..... **ix**
Introduction générale **1**
Contexte du travail et objectifs 1
Organisation de la thèse 1
Contributions 2
Chapitre 1 Contexte des codes correcteurs **3**
1.1 Système de communication numérique 3
1.2 Lutte contre le bruit du canal 4
1.3 Modulation 7
1.4 Canal de transmission 8
Chapitre 2 Les turbocodes classiques **10**
2.1 Introduction 10
2.2 Concaténation parallèle des codes convolutionels (PCCC) 11
2.2.1 encoder convolutionel systématique récursif (RSC) 13
2.3 Entrelacement 17
2.4 Poinçonnage 19
2.5 Terminaisons 19
2.6 Turbodécodeur 20
2.6.1 Rapports de Logarithme de vraisemblance 21
2.7 Algorithme MAP 24
2.7.1 Introduction et préliminaire mathématiques 24
2.7.2 Calcul récursif en avant de $\alpha_k(s)$ 28
2.7.3 Calcul récursif en arrière de $\beta_k(s)$ 29
2.7.4 Calcul de $\gamma_k(s',s)$ 30
2.7.5 Résumé des étapes pour la mise en œuvre de l'algorithme MAP 32
2.8 Principes de turbodécodage itératif 32
2.8.1 Turbodécodage et préliminaires mathématiques 32
2.8.2 Turbodécodage itératif 34
2.9 Modifications sur l'algorithme MAP 35

2.9.1 Introduction 35
2.9.2 L'algorithme Max-Log-MAP 35
2.9.3 Algorithme Log-MAP (correction de l'approximation) 37
2.10 Algorithme SOVA (Soft Output Viterbi Algorithm) 37
2.10.1 Description de SOVA 37
2.10.2 Résumé des étapes de la mise en œuvre de l'algorithme SOVA 40
2.11 Conclusion 40
Chapitre 3 Optimisation des entrelaceurs 41
3.1 Introduction 41
3.2 La permutation régulière 42
3.3 Rôle de désordre 45
3.4 Permutation irrégulière 47
3.4.1 L'entrelaceur DRP 47
3.4.2 L'entrelaceur ARP 48
3.4.3 Les entrelaceurs par section d'or 49
3.4.4 L'entrelaceur dimensionnel DI 53
3.5 Résultats et comparaison entre performances 54
3.6 Conclusion 57
Conclusion générale 58
Conclusions 58
Suggestions pour les travaux futures 58
Articles 60
Références 61
Annexe A. " Fonction Matlab des entrelaceurs " 64

Liste des figures

Figure1.1 Schéma d'un système de communication numérique. 3
Figure 1.2 Forme d'un signal BPSK. 7
Figure 1.3 Constellation d'un signal BPSK. 7
Figure 1.4 Modèle d'un canal de Rayleigh. 8
Figure 2.1 Schéma d'un encodeur turbo. 12
Figure 2.2 Diagrammes de treillis des RSCs pour l'exemple de turbocode (5, 2, 16). 12
Figure 2.3 Schéma d'un encodeur convolutionnel systématique récursif. 14
Figure 2.12 Principe de poinçonnage. 20
Figure 2.13 Schéma d'un turbodécodeur. 21
Figure 2.14 Transitions possibles pour un RSC de K=3. 26
Figure 2.15 Treillis d'un décodeur pour un RSC de K=3. 27
Figure 2.16 Schéma d'un turbodécodeur. 34
Figure 3.1. Un turbocode binaire $(15, 13)_8$. 41
Figure 3.2. Permutation régulière sous forme rectangulaire (a) et circulaire (b). 43
Figure 3.3. Exemple de motifs d'erreurs possibles avec une permutation régulière. 46
Figure 3.4. principe d'entrelaceur DRP. 47
Figure 3.5. représentation rectangulaire d'entrelaceur ARP. 48
Figure 3.6. Illustration du principe de section d'or. 49
Figure 3.7. Distance minimale entre les points en fonction du nombre de points avec un incrément d'or. 51
Figure 3.8. BER pour N=1024 bits. 55
Figure 3.9. BER pour N=512 bits. 56
Figure 3.10. BER pour N=120 bits. 57

Liste des tableaux

Tableau 2.1 Code convolutionnel avec R=1/2. ..... *15*
Tableau 2.2 Code convolutionnel avec R=1/3. ..... *15*
Tableau 2.3 Code convolutionnel avec R=2/3. ..... *16*
Tableau 2.4 Code convolutionnel avec R=1/4. ..... *16*
Tableau 2.5 Code convolutionnel avec R=1/5. ..... *16*
Tableau 2.6 Code convolutionnel avec R=1/6. ..... *17*
Tableau 2.7 Code convolutionel avec R=1/7. ..... *17*
Tableau 2.8 Code convolutionnel avec R=1/8. ..... *17*

Liste des abréviations

PCM	Pulse Coded Modulation (modulation à impulsion codée)
BER	Bit Error Rate (taux d'erreur par bit)
BPSK	Binary Phase Shift Keying (modulation par déplacement de phase -2)
BFSK	Binary Frequency Shift Keying (modulation par déplacement de fréquence -2)
MAP	Maximum A Posteriori
SOVA	Soft Output Viterbi Algorithm (algorithme de Viterbi à sortie soft)
SISO	Soft-In Soft-Out (entrée soft - sortie soft)
FSK	Frequency Shift Keying (modulation par déplacement de fréquence)
AWGN	Additif White Gaussien Noise (bruit additif blanc Gaussien)
RSC	Recursive Systematic Coder (codeur convolutif systématique récursif)
PCCC	Parallel Concatenated Convolutive Code (Concaténation parallèle des codes convolutionels)
PDF	Probability Density Function (fonction de densité de probabilité)
LLR	Log Likelihood Ratio (rapport de Logarithme de vraisemblance)
SNR	Signal to Noise Ratio (rapport signal sur bruit)
ML	Maximum Likelihood (maximum de vraisemblance)
VLSI	Very Large Scale Integration (intégration à très grande échelle)

Dédicaces

A ma grande et petite famille,

A mes amis.

Remerciements

Tout d'abord, je tiens à remercier mon directrice de thèse M^{me} CHOUIREB Fatima, maitre de conférences à l'université Amar Telidji-Laghouat (UATL) pour ses conseils et la confiance qu'elle m'a accordée pendant le déroulement de cette thèse.

Tous mes remerciements sont adressés à M. BOUZOUAD Mouloud, directeur de LTSS pour l'accueil chaleureux qu'il m'a réservé dans son labo.

Je voudrais remercier M. BENTRIA Bachir d'avoir accepté de présider ce jury.

Je voudrais exprimer aussi ma gratitude à M. OCHAR Ali, M. HADJ ABDERRAHMANE Lahcène et M. ALI PACHA Adda, membres de ce jury, d'avoir accepté d'examiner cette thèse.

Je voudrais remercier aussi, tous qui ont porté un plus à cette thèse.

Introduction générale

Contexte du travail et objectifs

Les transmissions à haut débit par satellite, telles qu'on les rencontre en observation de la terre ou en télécommunication par exemple, sont de plus en plus souvent confrontées aux problèmes de cohabitation avec d'autres émissions ou services partageant la même bande de fréquence, en raison de leur occupation spectrale importante liée au rythme des données transmisses.

D'autre part, la recherche permanente d'un bon compromis entre l'efficacité spectrale et de puissance du signal, lorsque ce dernier subit des fluctuations durant son parcours (cas d'un canal satellite), permet à la fois d'augmenter la capacité et d'améliorer la disponibilité du système.

L'objectif de ce travail est d'améliorer l'efficacité de puissance dans les systèmes de communication par satellite par le moyen des codes correcteurs d'erreurs (codage de canal). La modulation utilisée est la modulation BPSK. Le modèle du canal satellite utilisé est celui de AWGN. Les performances mesurées en terme de BER ont été analysées pour une chaîne concaténée de turbocode.

L'enveloppe constante de la porteuse, les différents paramètres ayant un impact sur les performances du signal à transmettre, ainsi que les bonnes performances en puissance lors d'un processus de décodage itératif (codage turbo) sont les principales causes de ce choix.

Organisation de la thèse

Cette thèse est organisée comme suit :

Le chapitre 1 donne les notions générales sur le codage du canal. Il trace le concept de base d'un système de communication numérique ou on évoque aussi le problème de la lutte contre le bruit du canal à l'aide des codes correcteur d'erreurs, le type de modulation utilisée et le modèle du canal utilisé, à savoir le canal AWGN.

Dans le chapitre 2, nous avons décrit en détail la structure d'un turbocode classique et ses composants, la concaténation parallèle des deux codeurs convolutifs systématiques récursifs

séparés par un entrelaceur des bits d'informations, le concept de perforation des bits de parité et finalement les terminaisons des RSCs par des queues (tails) de bits. Ensuite, nous avons détaillé le principe de décodage itératif turbo et l'introduction de la notion d'information intrinsèque et d'information extrinsèque, avec les différents algorithmes de décodage d'entrée soft et de sortie soft (SISO) à savoir le MAP et le SOVA.

Au chapitre 3, nous avons analysé en détail à l'aide du principe des séquences RTZ (Return To Zero), le rôle de l'entrelaceur dans l'opération de lutte contre les paquets d'erreurs (burst errors). Ensuite, nous avons représenté quelques entrelaceurs populaires aujourd'hui. Ce chapitre est finalisé par la représentation de notre entrelaceur proposé et bien sûr comparé en performance avec ces derniers entrelaceurs.

Finalement, nous récapitulons par une conclusion générale sur l'ensemble des résultats obtenus et nous proposons des futurs travaux pour la recherche.

Contributions

Cette recherche a permis d'apporter les contributions suivantes :

1. Le développement d'un nouveau concept appelé "dimension d'entrelaceur" pour étudier le comportement des entrelaceurs au lieu d'indice d'incrémentation d'entrelaceur utilisé habituellement.

2. Une nouvelle définition (formulation) de la dispersion normalisée pour une permutation aléatoire.

3. Proposition d'un nouveau type d'entrelaceurs nommés : entrelaceur dimensionnel.

Chapitre 1

Contexte des codes correcteurs

1.1 Système de communication numérique

Un système de communication est un moyen de transport de l'information d'un usager à un autre en général éloigné. Le système est appelé numérique ce qui signifie que l'information est représentée par une séquence de symboles choisis parmi un alphabet (binaire par exemple). Nous avons présenté les éléments de base d'un système de communication numérique à la figure1.1. Un système de communication consiste à faire subir à l'information émise plusieurs opérations comme le codage de source, le cryptage et le codage du canal avant la transmission.

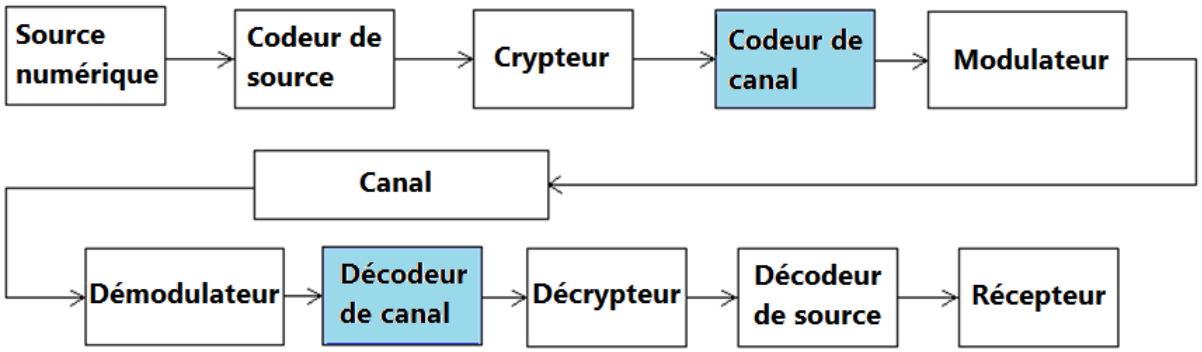

Figure1.1 Schéma d'un système de communication numérique.

- Le codage de source est utilisé afin de réduire la redondance dans l'information issue de la source. La source de données peut être analogique ou numérique. Dans le premier cas le signal représentant la source doit d'abord être échantillonné, ensuite quantifié et enfin transporté en une suite de '0' et '1' par les techniques de conversion analogique-numérique. Dans le cas où la source est numérique, le codage de source est directement appliqué à la source. Nous pouvons citer le codage de Huffman à titre d'exemple.

- Le cryptage empêche un usager non autorisé de comprendre le message envoyé ou d'injecter des fausses informations.

- Le codage de canal appelé codage pour la correction d'erreur est utilisé afin d'ajouter de redondance à l'information transmise dans le but de la protéger contre le bruit et les perturbations introduites dans le canal. On note ici que l'ordre de ces trois premiers blocs (codeur de source, encrypteur, codeur de canal) ne peut être modifié. Par contre, il faut mentionner que plusieurs chercheurs se sont intéressés à trouver une alternative qui puisse regrouper ces trois blocs en un seul ou au moins faire les opérations de codage de source et de codage de canal en une seule opération.

- Le modulateur adapte la séquence codée au canal physique qui peut être modélisé sous plusieurs formes qui dépendent du type d'application (liaison satellite, terrestre par câble ou sans fil).

- À la réception, les étapes inverses de l'émission sont appliquées à l'information reçue : le signal est démodulé, ensuite il est décodé par le décodeur du canal. Le décodage est effectué dans le but de reproduire exactement la séquence émise et de minimiser la probabilité d'erreur. Le décodage peut être décrit comme un algorithme qui exécute des opérations sur les symboles bruités reçus du canal afin de les corriger. Enfin, le signal est décrypté avant de subir le décodage de la source afin de récupérer l'information originale [1].

1.2 Lutte contre le bruit du canal

Les premières utilisations des codes correcteurs d'erreur en communication furent dans le domaine spatial où l'objectif était de surmonter la perte de propagation dans le vide entre le récepteur (ou transmetteur) de l'engin spatial et le transmetteur (ou récepteur) de la station terrestre. Les vaisseaux ou engins spatiaux envoyés au-delà de l'orbite terrestre devraient être de petite taille ce qui implique une restriction au niveau de la taille des antennes (aussi la batterie), limitant ainsi la puissance de l'émetteur (de l'engin). Le gain du codage offert par les codes correcteurs d'erreur a permis des liaisons spatiales de plus en plus éloignées de la terre. La largeur de bande était un facteur moins déterminant dans les liaisons spatiales que dans le domaine des communications terrestre. L'efficacité d'un code se reflète dans le gain du codage. Ce dernier est défini comme étant la différence entre le rapport signal sur bruit E_b/N_0 requis pour atteindre un taux d'erreur binaire avec un système codé et celui requis pour atteindre le même taux d'erreur binaire (BER) avec un système non codé.

Nous allons poursuivre avec l'exemple des engins spatiaux où la modulation binaire de phase BPSK semble être le choix logique pour ce type de communication du fait de son efficacité au niveau de la puissance par rapport aux autres types de modulation. Par exemple, la modulation BPSK offre un gain de 3 dB par rapport à la modulation binaire BFSK [2]. Un système BPSK non codé nécessite un E_b/N_0 de 9.6 dB pour atteindre un BER égale à 10^{-5}. Le standard du codage utilisé par la NASA pour les missions spatiales consistait en une concaténation en série d'un code de Reed-Solomon avec un code convolutionnel. En supposant une modulation BPSK, le standard permet d'atteindre un BER $=10^{-5}$ pour E_b/N_0 égal à 2.2 dB. Ce standard permet un gain du codage de 9.6-2.2=7.4 dB.

En diminuant la valeur de E_b/N_0 requis à l'entrée du modulateur, le code correcteur d'erreur qui est appelé aussi code du canal permet d'assouplir les exigences vis à vis des autres paramètres de la liaison de communication. Par exemple, les communications fiables seront possibles pour des distances plus grandes ou la puissance de l'émetteur de l'engin spatial peut être réduite (accroissement de la durée de vie et/ou réduction de la taille de la batterie). Il est aussi possible de réduire la taille des antennes et par conséquent réduire le poids de l'engin. Tous ces facteurs ont contribué à ce que le code correcteur d'erreur joue un rôle incontournable dans la conception des systèmes de communication des engins spatiaux.

De point de vue purement économique, l'impact est impressionnant. A la fin des années soixante, chaque dB du gain de codage est estimé à 1 million de dollars de coût qui inclut le développement et le lancement. Actuellement, la valeur d'un dB pour les communications spatiales est de 80 millions de dollars [2].

La question est de savoir quel est le gain du codage maximal qui pourrait être obtenu dans un système de communication. Shannon a répondu à cette question en considérant le cas d'un canal à bruit additif et gaussien (AWGN). Il a ainsi défini la capacité du canal comme :

$$C = W \cdot \log_2\left(1 + \frac{S}{N}\right) \tag{1.1}$$

Avec $N = W \cdot N_0$ et $S = \frac{n \cdot E_b}{T} = R_t \cdot E_b$

Où W est la largeur de bande du canal en Hertz, S est la puissance moyenne du signal et N_0 est la densité spectrale de puissance du bruit unilatérale dans le canal, n est la longueur du mot code, T est la période du signal et R_t est le débit (taux) de transmission.

En manipulant l'équation de la capacité, une limite de gain du codage pour la modulation BPSK peut être trouvée. Cette limite est exprimée en fonction de l'efficacité spectrale η d'une modulation donnée et du rapport E_b/N_0. L'efficacité spectrale est le nombre moyen de bits d'informations transmis dans un intervalle de signalisation de durée T. Si T est normalisée et reliée à l'inverse de la longueur de bande, η s'exprime en terme de bits par seconde par Hz (b/s/Hz). La valeur E_b/N_0 en fonction de η s'exprime sous la forme :

$$\frac{E_b}{N_0} > \frac{2^\eta - 1}{\eta} \tag{1.2}$$

Or l'efficacité spectrale pour une modulation BPSK est de 1b/s/Hz donc la limite inférieure du E_b/N_0 est égale à 1 ou 0 dB. Par conséquent le gain du codage maximum avec une BER de 10^{-5} est (9.6-0)=9.6 dB. Dans les années 80 et au début des années 90 plusieurs efforts sont fournis afin d'obtenir un gain du codage supérieur à 7.4 dB. La majorité des efforts se sont concentrés sur les systèmes utilisant des concaténations en série, ce qui a conduit à des décodeurs de Viterbi

très complexes. La conséquence est l'obtention de plusieurs dixièmes de dB de gain mais au prix d'une grande complexité et de délai. La réalité est que 50 ans après la publication du théorème de Shannon, 2 dB de gain important séparaient les systèmes de correction d'erreur les plus avancés des résultats théoriques. Cet écart a presque disparu avec l'invention des codes turbo. Avant d'introduire les codes turbo, nous allons évoquer le fonctionnement des codes concaténés en série qui font partie du processus progressif de la découverte des codes turbo.

Forney dans son ouvrage sur les codes concaténés (Concatenated Codes ,1966), a examiné la concaténation en série d'un code Reed-Solomon avec un code convolutionnel. Le décodeur Viterbi utilisé pour décoder les codes convolutionnels faisait une décision hard sur les symboles afin de les fournir au décodeur Reed-Solomon. Un entrelaceur est inséré entre les codeurs afin de disperser les paquets d'erreurs. Cette forme de concaténation permet d'avoir de très bonnes valeurs de BER et une faible complexité au niveau des décodeurs, ce type de codage a été utilisé par la NASA.

L'utilisation d'une décision hard à la sortie du premier décodeur limite les performances d'erreur. Si le décodeur fournissait une information de fiabilité au deuxième décodeur au lieu de faire une décision sur le symbole codé, la capacité du système de codage concaténé serait accrue. Mais malheureusement le décodeur de Reed-Solomon n'est pas adéquat pour fournir une décision soft. Par contre, il existe des algorithmes plus pratiques qui peuvent recevoir de l'information soft à l'entrée (soft in) et qui produisent une décision soft à la sortie (soft out). L'algorithme MAP (maximum a posteriori) et SOVA (soft output viterbi algorithm) sont parmi les meilleurs choix. Le développement de ces deux algorithmes a conduit tout droit à la concaténation de deux codes convolutionnels.

L'événement marquant fut l'aboutissement des propositions afin de s'approcher de la borne de Shannon. En 1993, Berrou, Glavieux et Thitimajshima ont montré avec l'introduction des codes turbo qu'on n'est plus qu'à 0.7 dB de la borne de Shannon soit un nouveau gain de codage de (2.2-0.7)=1.5 dB. Les codes turbo sont des codes convolutionnels concaténés en parallèle et décodés itérativement. Le premier codeur code l'information à transmettre alors qu'un entrelaceur sépare l'information du deuxième codeur.

Chaque code systématique est codé en utilisant un décodeur Soft-In Soft-Out (SISO). La sortie du premier décodeur alimente le deuxième décodeur, ce qui correspond à une itération complète du décodeur turbo. Contrairement à l'algorithme de Viterbi qui fournit la séquence la plus vraisemblablement émise, l'algorithme MAP détermine le bit d'information le plus probable qui a été transmis dans une séquence codée. Pour des faibles valeurs de BER, la différence au niveau de la performance entre le MAP et le Viterbi est quasi-négligeable. Etant donné que l'algorithme MAP est nettement plus complexe que l'algorithme de Viterbi, il a été longtemps ignoré.

Cependant, pour des faibles valeurs de E_b/N_0 et de BER élevés, le MAP améliore le gain par rapport à Viterbi de 0.5 dB. Ceci est très important car dans le cas des codes turbo les performances en BER sont élevés aux premières itérations. Par conséquent, la moindre amélioration obtenue pour des BER élevées implique une croissance au niveau des performances.

L'algorithme SOVA qui est aussi utilisé pour le décodeur turbo demeure le rival du MAP. Toutefois, les propriétés statistiques des sorties générées sont moins bonnes que celles qui sont fournies par l'algorithme MAP, ce qui donne une dégradation des performances de 0.8 dB [1].

1.3 Modulation

Dans notre travail, nous avons considéré le cas où la modulation est BPSK. La modulation BPSK (Binary Phase Shift Keying) est parmi les modulations numériques les plus utilisées [3]. En effet, elle est plus performante au niveau de la probabilité d'erreur et donc la plus adéquate pour les communications spatiales (un gain de 3 dB par rapport au FSK). En BPSK, les phases opposées de la porteuse (0 et π) sont transmises toutes les T secondes (si on considère que la durée d'un bit est T) et sont basées sur les symboles d'information +1 ou -1. BPSK peut être considérée comme un type de modulation d'amplitude où la porteuse est multipliée par -1 ou +1 suivant le bit de parité comme l'indique la figure 1.2 où les données +1 et -1 sont représentées respectivement par $\cos(w_c t)$ et $\cos(w_c t+\pi)$. Nous pouvons représenter ceci sous forme de constellation dans un plan x/y. BPSK se limite à deux signaux sur l'axe des x tel qu'il est montré dans la figure 1.3 où E_b est l'énergie par bit [1].

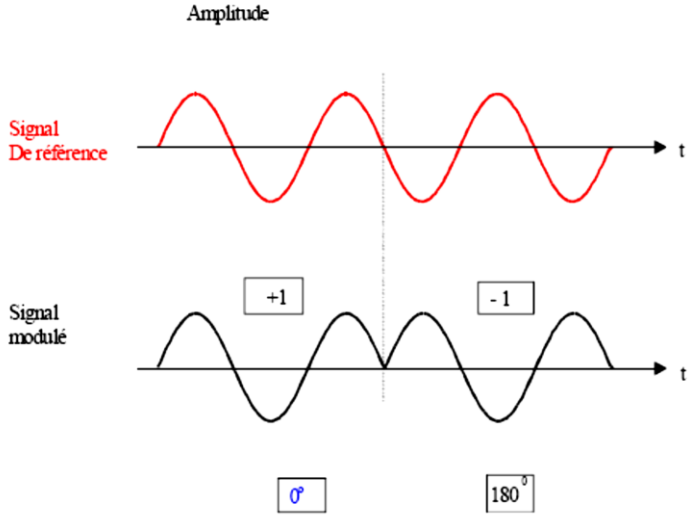

Figure 1.2 Forme d'un signal BPSK.

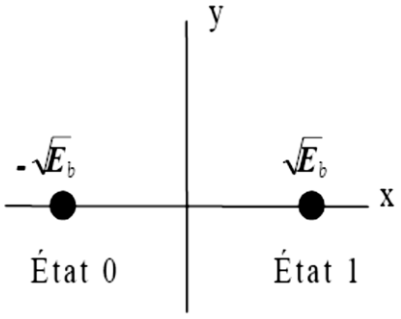

Figure 1.3 Constellation d'un signal BPSK.

1.4 Canal de transmission

Le bruit introduit par le canal peut prendre différentes formes. Le modèle de bruit le plus commun est celui du bruit additif blanc Gaussien (AWGN). Ce dernier modèle est plus simple et il permet de fournir un modèle presque parfait dans certains systèmes de communication. Par exemple, pour les canaux satellites où les communications sont en visibilité directe, le modèle AWGN est exact. Mentionnons que l'adjectif 'additif ' dans AWGN signifie que l'impact du bruit sur le signal transmis peut être modélisé comme variable aléatoire n qui s'additionne au signal modulé. La variable n est supposée Gaussienne de moyenne nulle et de variance σ^2 et dont la densité de probabilité est définie par :

$$P(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-n^2/2\sigma^2} \tag{1.3}$$

Le terme 'blanc' indique que la densité spectrale de puissance du bruit est considérée unilatérale (c'est-à-dire de la fréquence 0 à ∞) et constante de valeur N_0. Si x_k est la variable qui représente le signal modulé correspondant à un bit d'information à l'instant k, alors à la sortie du canal nous recueillons le signal réceptionné y_k bruité :

$$y_k = x_k + n \tag{1.4}$$

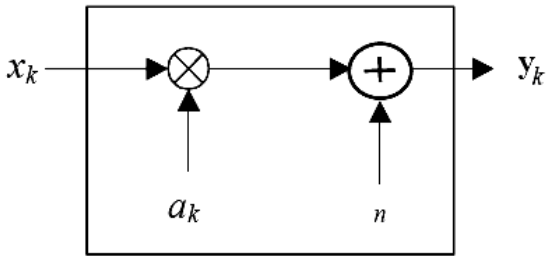

Figure 1.4 Modèle d'un canal de Rayleigh.

Dans les communications radiomobiles, l'atténuation du signal varie avec la vitesse du véhicule (qui est le récepteur) et la nature des obstacles. Ce type de canal est appelé canal à évanouissement, l'amplitude a_k du signal est souvent modélisée suivant la loi de Rayleigh, le signal est sous la forme :

$$y_k = a_k \cdot x_k + n \tag{1.5}$$

Où a_k est appelé aussi l'enveloppe du signal. Sa fonction de densité de probabilité est donnée par :

$$P(a_k) = \begin{cases} \frac{a_k}{\sigma^2} \cdot e^{-\frac{a_k^2}{2\sigma^2}} & a_k \geq 0 \\ 0 & \text{ailleurs} \end{cases} \quad (1.6)$$

Le modèle de canal de Rayleigh est schématisé par la figure 1.4 où y_k est la version bruité de x_k à l'entrée du canal.

Chapitre 2

Les turbocodes classiques

2.1 Introduction

Le second théorème de Shannon, publié en 1948 dans la référence [4], montre comment en principe le codage aléatoire permettrait d'atteindre la capacité d'un canal bruité, avec un taux d'erreurs arbitrairement faible. Cependant, bien que presque tous les codes aléatoires générés de cette façon sont en principe bons, il est nécessaire d'utiliser des longueurs de blocs assez importantes, ce qui conduit à des tables de code de taille gigantesque (la taille croît exponentiellement avec la longueur des blocs, et cela d'autant plus que le débit souhaité est grand). En conséquence, le décodage devient pratiquement irréalisable.

C'est la raison pour laquelle, depuis cette publication, de nombreux travaux ont porté sur la mise au point de codes de canal structurés (approche algébrique) de manière à en rendre le décodage faisable. Mais, le codage de canal n'a pas atteint ce niveau de maturité à l'heure actuelle. Par exemple il n'existe pas à l'heure actuelle de méthode universelle de codage de canal qui pourrait approcher asymptotiquement les limites de Shannon [5].

En effet, l'invention des turbocodes par C. Berrou et A. Glavieux et P. Thitimajishima en 1993, a enfin permis de disposer de codes de complexité raisonnable ayant des performances très proches de la limite de Shannon.

Une étude de la NASA montre que ce système de codage /décodage permet de réduire le taux d'erreur à 10^{-5} lorsque le rapport signal sur bruit est de l'ordre de 0 dB ou même lorsque celui-ci est négatif (Puissance de bruit plus forte que la puissance du signal).

Le terme turbo vient de la notion de bouclage semblable à celle utilisée dans les moteurs turbo. L'idée est d'utiliser plusieurs séquences redondantes d'un même message. L'une étant calculée à partir du message original et l'autre à partir d'une version permutée de ce message. La structure des turbocodes résulte de la composition de deux ou plusieurs codeurs[1]. Ces codeurs peuvent être de type convolutif systématique récursif (RSC), convolutif classique (CC)

[1] Un codeur est une machine d'état réalisée à partir des bascules (avec un délai D pour chaque bascule) qui jouent le rôle d'une mémoire.

ou codes en bloc. Cette composition est appelée concaténation. Les deux plus connues sont celles qui sont réalisées en parallèle et en séries. Il est possible de combiner ces deux modes, on parle alors de concaténations hybrides. Mais cette dernière est décrite dans la littérature comme peu intéressante face au gain de performance déjà assuré par les modèles simples.

Dans tous les cas les codeurs sont reliés par des entrelaceurs identiques entre eux. L'entrelacement est nécessaire lors du transfert de bits d'un codeur (ou décodeur) vers le suivant. Tout le principe des turbocodes est basé sur cette notion. L'entrelacement consiste à permuter les données émises. Plusieurs méthodes de permutation sont possibles, cependant le choix de la structure de l'entrelaceur est un facteur clé qui détermine les performances d'un turbocode [6].

2.2 Concaténation parallèle des codes convolutionels (PCCC)

Un encodeur pour un turbocode classique consiste en une concaténation parallèle de deux codeurs RSC (codeur convolutionel récursif systématique) ou plus, habituellement identiques, et un entrelaceur pseudo aléatoire comme il est illustré en figure 2.1. Cette structure de codeur est appelée concaténation parallèle parce que les deux encodeurs opèrent sur le même bloc des bits d'entrée, par rapport à la concaténation série où le deuxième codeur (RSC2) opère sur les bits de sortie du premier codeur (RSC1). Dans tout ce qui suit, nous considérons seulement les turbocodes avec deux RSC, bien que cette méthode de codage puisse facilement être prolongée au cas où trois encodeurs ou plus sont employés [7]. Cependant, les conclusions majeures et les résultats performants sont réalisables avec seulement deux codeurs RSC.

L'entrelaceur est employé pour permuter les bits d'entrée tels que les deux codeurs fonctionnent sur le même bloc de bits d'entrée, mais dans un ordre différent. Le premier encodeur reçoit directement le bit l'entrée u_k et produit la paire $(u_k, v_k^{(1)})$ tandis que le deuxième encodeur reçoit le bit l'entrée u_k' et produit la paire $(u_k, v_k^{(2)})$ où u_k' est rejeté. Les séquences du bit d'entrée sont groupées dans des blocs de longueur finie égale à N (N qui est la taille de l'entrelaceur). Puisque les deux encodeurs sont systématiques et opèrent sur le même ensemble des bits d'entrée, il est seulement nécessaire de transmettre les bits d'entrée une fois, et le code global a le taux R=1/3. Afin d'augmenter le taux du code à R=1/2, les deux séquences de parité, $v^{(1)}(D)$ et $v^{(2)}(D)$ peuvent être perforés, typiquement en supprimant alternativement $v_k^{(1)}$ et $v_k^{(2)}$. Nous nous référerons à un turbocode dont les RSCs ont les polynômes générateurs des bits de parité $h_0(D)$ et $h_1(D)$, et dont l'entrelaceur est de longueur N, par (h_0, h_1, N) turbocode, où h_0 et h_1 sont les représentations octales des polynômes.

Afin d'illustrer les principes de base de la concaténation parallèle, on a considéré que l'encodeur turbo est constitué de deux RSCs $(2, 1, 2)^2$ identiques comme composants d'encodeur turbo, avec des polynômes générateurs des bits de parité $h_0(D)=D^2+1=5$ et $h_2(D)=D=2$. Pour cette illustration on suppose un petit entrelaceur de taille N=16 qui produit le turbocode (5, 2, 16). L'entrelaceur est réalisé comme une matrice de 4×4 remplie séquentiellement ligne par ligne, avec u_k bits d'entrée. Une fois que l'entrelaceur a été rempli, les bits d'entrée au deuxième

[2] (n : nombre de bits de parité, k : nombre de bits d'information, K : longueur de contrainte).

codeur u_k' sont obtenus en lisant l'entrelaceur d'une façon pseudo aléatoire jusqu'à ce que chaque bit ait été lu une fois et seulement une fois. L'entrelaceur pseudo aléatoire dans notre exemple est représenté par la permutation π_{16}={15, 10, 1, 12, 2, 0, 13, 9, 5, 3, 8, 11, 7, 4,14 ,6}, ce qui implique $u_0'= u_{15}$, $u_1'= u_{10}$, et ainsi de suite.

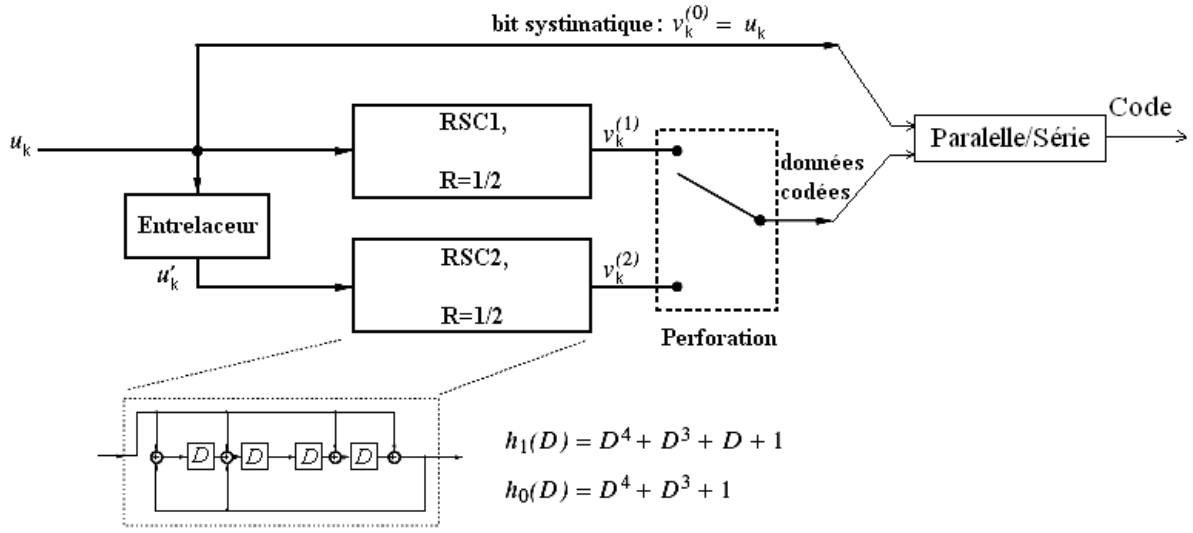

Figure 2.1 Schéma d'un encodeur turbo.

Si {u_0, ..., u_{15}}={1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0} est la séquence d'entrée, et l'entrelaceur est représenté par la permutation π_{16}, alors la séquence u'=π_{16}({u_0, ..., u_{15}})={0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0} est l'entrée du second codeur. Les diagrammes de treillis pour les deux codeurs constitutifs avec ces entrées sont montrés dans la figure 2.2. La séquence de parité correspond au premier codeur est {0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0} et pour le deuxième codeur est {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1}. Sans perforation le mot code résultant à un poids de Hamming d=w(u)+w($v^{(1)}$)+w($v^{(2)}$)=4+3+3=10. Si le code est perforé et que la perforation est débutée avec $v_0^{(1)}$, alors le mot code résultant à un poids d=4+3+2=9. Si, c'est l'inverse, c'est-à-dire que la perforation est débutée avec $v_0^{(2)}$, alors le mot code résultant à un poids d=4+0+1=5.

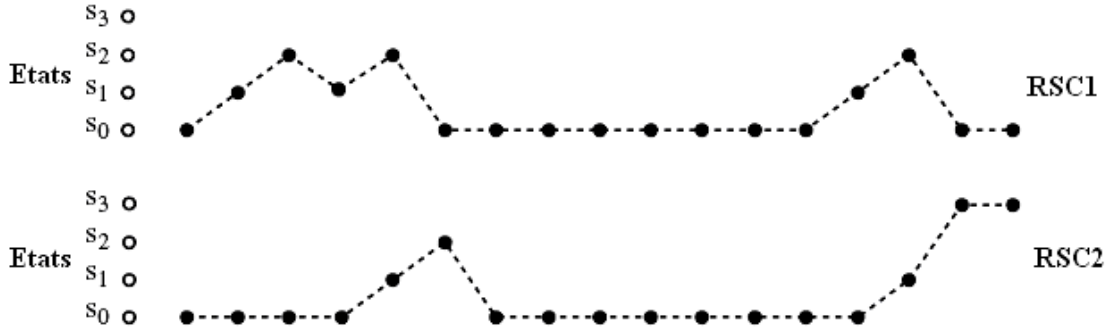

Figure 2.2 Diagrammes de treillis des RSCs pour l'exemple de turbocode (5, 2, 16).

Ce simple exemple illustre plusieurs points saillants au sujet de la structure des mots code dans un turbocode. D'abord, parce que l'entrelaceur pseudo aléatoire permute les bits d'entrée, les deux séquences d'entrée u et u' sont presque toujours différentes, cependant de même poids, et les deux codeurs produisent (avec une probabilité élevée) des séquences de parité de différents poids. En second lieu, on le voit facilement qu'un mot code peut se composer d'un certain nombre de détours distincts dans chaque encodeur, comme illustré sur la figure 2.2. Puisque les codeurs constitutifs (RSC1 et RSC2) sont réalisés sous la forme systématique récursive, un bit d'entrée non nul est exigé pour renvoyer le codeur à l'état zéro et tous les détours sont associés ainsi aux séquences d'information de poids 2 ou plus grand, au moins un pour le départ et un pour la fusion. Finalement, avec un entrelaceur pseudo aléatoire il est fortement peu probable que les deux encodeurs seront retournés à l'état zéro à la fin du mot code même lorsque les derniers (K-1) bits[3] de la séquence d'entrée u sont utilisés pour forcer le premier codeur de nouveau à l'état zéro.

Si ni l'un ni l'autre encodeur n'est forcé à l'état zéro, ça veut dire qu'aucun bits de queue (tail) n'est employé pour terminer le premier encodeur, donc la séquence de N-1 zéros suivis d'un '1' est une séquence d'entrée u valide au premier encodeur. Pour quelques entrelaceurs, cette séquence sera permutée de telle sorte que u'=u. Dans ce cas, le poids maximum du mot code avec la perforation, et ainsi la distance libre du code, seront seulement deux. Pour cette raison, il est commun (fort probable) de terminer le premier encodeur à l'état zéro. Une étude de l'ambiguïté de l'état final du deuxième encodeur a été montrée par simulation dans les références [8], [9]. Des structures spéciales d'entrelaceur qui ont comme résultat les deux encodeurs retournant à l'état zéro sont discutés dans les références [10], [11], et [12].

Chercher la distance libre et le spectre de distance d'un turbocode est compliqué par le fait que les codes convolutionels concaténés en parallèles sont en général des codes variables dans le temps dus à l'entrelaceur. C'est-à-dire, pour la séquence décalée D·u(D) la première séquence de parité est D·$v^{(1)}$(D), mais la séquence d'entrée au deuxième encodeur (RSC2) n'est pas D·u'(D) (avec la probabilité élevée) dû à l'entrelaceur. Ainsi, la deuxième séquence de parité n'est pas D·$v^{(2)}$(D).

Soit l'exemple précèdent avec une nouvelle séquence d'entrée, nous avons π_{16}(D·u(D))={1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0} et la seconde séquence de parité est donnée par {0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}. Ainsi, le décalage temporaire des bits d'entrée résulte des mots code qui différent en position de bit et en poids total de Hamming.

2.2.1 encoder convolutionel systématique récursif (RSC)

La figure 2.3 montre un exemple simple d'un RSC. Cet encodeur est dit systématique parce que l'information est émise, aussi cet encodeur est dit récursif car il dispose de boucles de retour qui réinjecte le contenu de la mémoire à l'entrée de l'encodeur.

[3] *K* : longueur de contrainte ou la taille du registre.

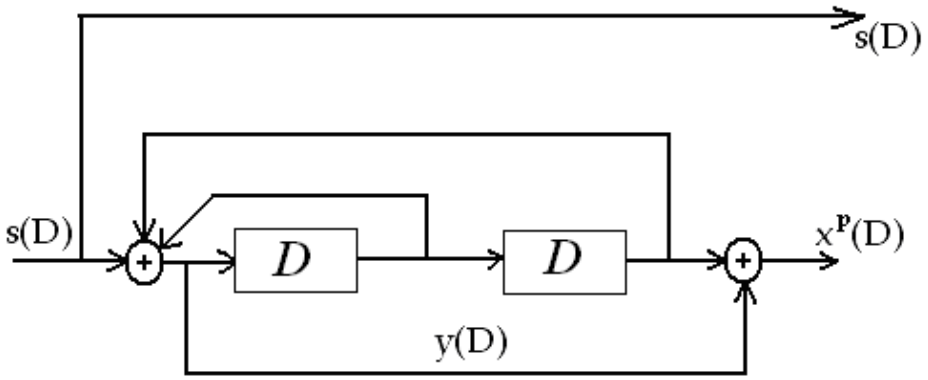

Figure 2.3 Schéma d'un encodeur convolutionnel systématique récursif.

Ce type de dispositif a la particularité de disposer d'une réponse impulsionnelle g(D) de durée infinie. En effet, supposons que la suite à l'entrée s(D) soit une impulsion en t=0, $s(D)=1+0\cdot D+0\cdot D^2+0\cdot D^3+...+0\cdot D^k+...$; on aura en sortie la suite g(D) qu'on peut déterminer de la manière suivante :

- on a $y(D)=s(D)+D\cdot y(D)+D^2\cdot y(D)$ et donc $s(D)=(1+D+D^2)\cdot y(D)$.
- Par ailleurs, on a $x^p(D) = y(D) + D^2 y(D) = (1+D^2)y(D)$.
- En éliminant y(D) on trouve

$$g(D) \triangleq \frac{x^p(D)}{s(D)} = \frac{1+D^2}{1+D+D^2} = \frac{(1+D)^3}{1+D^3} \qquad (2.1)$$

- on a :

$$g(D) = \frac{(1+D)^3}{1+D^3} = (1+D+D^2+D^3)\cdot\left(1+D^3+D^6+D^9+\cdots+D^{3k}\right) \qquad (2.2)$$

Nous avons mis en évidence le caractère périodique de g(D) (le second facteur est le développement en série de $(1+D^3)^{-1}$). Notons la similitude de ce type de machine avec un générateur de nombres aléatoires.

On peut déduire de la forme de g(D) que seules les entrées qui sont des multiples de $(1+D+D^2)$ donneront lieu à une réponse de durée finie. Cela est gênant, dans la mesure où cela signifie qu'un nombre non négligeable de messages source auront un vecteur de parité à poids faible et risquent de ce fait d'être confondus avec le message nul. (Nous savons que le poids minimal des mots de code non-nuls d'un code est égal à la distance minimale de ce code). La seule manière d'éviter ce problème serait d'augmenter la taille de la mémoire de l'encodeur et nous savons que le prix à payer en termes de complexité de décodage est prohibitif. Cela explique la raison pour laquelle les encodeurs récursifs n'ont pas intéressé beaucoup les chercheurs dans le domaine, jusqu'à la découverte par les auteurs des turbocodes d'un moyen subtil de restaurer une distance minimale importante et d'un algorithme de décodage efficace correspondant [5].

2.2.1.1 Choix d'un bon encodeur (ou code)

Des calculs à l'aide d'ordinateur sont habituellement utilisées pour trouver les bons codes, c'est-à-dire trouver une distance libre d_{free} maximale avec une longueur de contrainte K minimale. De cette façon, les codes dans les tableaux 2.1-2.8, ont été trouvés [13].

Tableau 2.1 Code convolutionnel avec R=1/2.

K-1	$g^{(1)}$	$g^{(0)}$	d_{free}
2	5	7	5
3	15	17*	6
4	23	35	7
5	65	57	8
6	133	171	10
7	345	237	10
8	561	753	12
9	1161	1545	12
10	2335	3661	14
11	4335	5723	15
12	10533	17661	16
13	21675	27123	16
14	56721	61713	18
15	111653	145665	19
16	347241	246277	20

*: les connections sont en octal: par exemple g=17=1111

Tableau 2.2 Code convolutionnel avec R=1/3.

K-1	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	5	7	7	8
3	13	15	17	10
4	25	33	37	12
5	47	53	75	13
6	133	145	175	15
7	225	331	367	16
8	557	663	711	18
9	1117	1365	1633	20
10	2353	2671	3175	22
11	4767	5723	6265	24
12	10533	10675	17661	24
13	21645	35661	37133	26

Tableau 2.3 Code convolutionnel avec R=2/3.

K-1	$g_2^{(2)}, g_1^{(2)}$	$g_2^{(1)}, g_1^{(1)}$	$g_2^{(0)}, g_1^{(0)}$	d_{free}
2	3, 1	1, 2	3, 2	3
3	2, 1	1, 4	3, 7	4
4	7, 2	1, 5	4, 7	5
5	14, 3	6, 10	16, 17	6
6	15, 6	6, 15	15, 17	7
7	14, 3	7, 11	13, 17	8
8	32, 13	5, 33	25, 22	8
9	25, 5	3, 70	36, 53	9
10	63, 32	15, 65	46, 61	10

Tableau 2.4 Code convolutionnel avec R=1/4.

K-1	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	5	7	7	7	10
3	13	15	15	17	13
4	25	27	33	37	16
5	53	67	71	75	18
6	135	135	147	163	20
7	235	275	313	357	22
8	463	535	733	745	24
9	1117	1365	1633	1653	27
10	2387	2353	2671	3175	29
11	4767	5723	6265	7455	32
12	11145	12477	15537	16727	33
13	21113	23175	35527	35537	36

Tableau 2.5 Code convolutionnel avec R=1/5.

K-1	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	7	5	5	13
3	17	17	13	15	15	16
4	37	27	33	25	35	20
5	75	71	73	65	57	22
6	175	131	135	135	147	25
7	257	233	323	271	357	28

Tableau 2.6 Code convolutionnel avec R=1/6.

K-1	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	7	7	5	5	16
3	17	17	13	13	15	15	20
4	37	35	27	33	25	35	24
5	73	75	55	65	47	57	27
6	173	151	135	135	163	137	30
7	253	375	331	235	313	357	34

Tableau 2.7 Code convolutionel avec R=1/7.

K-1	$g^{(6)}$	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	7	7	5	5	5	18
3	17	17	13	13	13	15	15	23
4	35	27	25	27	33	35	37	28
5	53	75	65	75	47	67	57	32
6	165	145	173	135	135	147	137	36
7	275	253	375	331	235	313	357	40

Tableau 2.8 Code convolutionnel avec R=1/8.

K-1	$g^{(7)}$	$g^{(6)}$	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	$g^{(0)}$	d_{free}
2	7	7	5	5	5	7	7	7	21
3	17	17	13	13	13	15	15	17	26
4	37	33	25	25	35	33	27	37	32
5	57	73	51	65	75	47	67	57	36
6	153	111	165	173	135	135	147	137	40
7	275	275	253	371	331	235	313	357	45

2.3 Entrelacement

Un entrelaceur π permet de modifier les indices (positions des bits en paquet d'information) des données par permutations, il représente un élément du groupe de permutation pour N éléments. Des groupes de permutation peuvent être décomposés en utilisant une seule méthode. Par exemple, π_{16}={15, 10, 1, 12, 2, 0, 13, 9, 5, 3, 8, 11, 7, 4, 14, 6} peut être écrit dans la décomposition de cycle de disjonction avec deux cycles :

$$\pi_{16} = \{(0, 15, 6, 13, 4, 2, 1, 10, 8, 5), (3, 12, 7, 9)\} \quad (2.3)$$

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

$$\pi(i) = j, \quad 0 \leq i, j < N \quad (2.4)$$

Soit l'entrelaceur le plus commun, l'entrelaceur de bloc de taille N=m×n. Ce type d'entrelaceur est basé sur l'écriture en ligne (entrée d'entrelaceur) et la lecture en colonne (sortie d'entrelaceur) d'une matrice d'information.

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

$$\pi(i) = j = ni + \left[\frac{i}{m}\right] \mod(N), \quad 0 \leq i, j < N \quad (2.5)$$

Où [x] est le plus grand nombre entier < x. Cette fonction ([i/m]) rend l'entrelaceur difficile à analyser, et un entrelaceur "linéarisé" est présenté dans la référence [14] avec la fonction d'indice

$$\pi(i) = j = ki + u \mod(N), \quad 0 \leq i, j < N \quad (2.6)$$

Avec k et u sont des nombres entiers fixes et k est relativement primaire à N, c'est-à-dire le plus grand diviseur commun de k et N [gcd (k, N)=1]. k est appelé le coefficient angulaire de l'entrelaceur linéaire.

Les entrelaceurs en bloc peuvent réaliser de bonnes valeurs de distance libre, mais ils échouent à produire un spectre des distances très étroit. Les coefficients angulaires k≈√N semblent fonctionner mieux, ayant pour résultat la bonne dispersion des points indexés [14].

On dispose aussi pour améliorer la performance du turbocode dans l'intervalle entre les valeurs moyennes et grandes du rapport signal sur bruit des entrelaceurs semi aléatoires [15], ce qui évite explicitement d'étaler des indices proches à l'entrée aux indices proches à la sortie. Ils sont produits de façon analogue aux entrelaceurs aléatoires, en choisissant chaque incrément π(i) aléatoirement et testant si |π(i)- π(l)| ≥ T pour tout i-S < l < i, c'est-à-dire en contrôlant tous les choix précédents. Si un choix d'indice ne remplit pas cette condition, il est rejeté et un nouvel

indice est choisi. Le processus continue jusqu'à ce que tous les N indices aient été choisis. Évidemment, le temps de recherche pour un entrelaceur approprié augmente avec S et T, et l'algorithme peut même ne pas se terminer avec succès. Cependant, selon la référence [15], des valeurs de S,T < √(N/2) finissent l'indexage dans un temps raisonnable.

Pour les entrelaceurs déterministes qui ont les propriétés statistiques des entrelaceurs aléatoires, Takeshita et Costello [14] ont conçu ce qu'ils ont appelé les entrelaceurs quadratiques, qui sont définis par la fonction quadratique d'indice suivante :

$$\pi(i) = \frac{ki \cdot (i+1)}{2} \mod(N) \qquad (2.7)$$

Avec k est une constante impaire [13].

2.4 Poinçonnage

Différents débits de code sont réalisés en poinçonnant les séquences x_{k1}^{p} et x_{k2}^{p} de bit de parité. Le poinçonnage de x_{k}^{s}, séquence de bits d'informations, mène à une dégradation des performances du turbocode. La figure 2.12 illustre le processus de poinçonnage. Un nombre '1' dans les tables représente un bit de code qui est inclus dans la séquence de bits transmis, et un nombre '0' représente un bit de code qui est exclu ou poinçonné. Du côté droit de chaque table est la liste des bits de code qui sont inclus dans la séquence transmise du code.

- En a), le code est non poinçonné et de taux de 1/3.
- En b), les bits de parité alternatifs de chaque encodeur sont poinçonnés chaque fois à l'instant k. Le résultat est un taux de code égal à 1/2.
- En c), le code est poinçonné pour former un code de taux de 3/4 [17].

2.5 Terminaisons

Les turbocodes considérés sont des codes en blocs[4], de longueur N égale à la taille de l'entrelaceur. On a obligé de forcer les deux RSCs de retourner à l'état initial 0 par une séquence de bits appelés queue, ajoutée à la séquence de bits d'information.

Il est clair que le fait d'ajouter une séquence de queue force l'état final du premier encodeur (RSC1) à 0. Cependant, à cause d'entrelacement, l'état final du deuxième encodeur (RSC2) ne sera pas connu. Des études sur cet aspect sont présentées dans [11] et [12]. Cependant, on a trouvé expérimentalement que la non prise en compte de l'état final du deuxième encodeur mène aux différences non significatives en terme de performance du décodeur. Il suffit d'initialiser

[4] Les mots codes sont indépendants les uns des autres.

uniformément les probabilités des états finaux en treillis pour le deuxième décodeur (SISO2) dans l'exécution de l'algorithme du décodage [18].

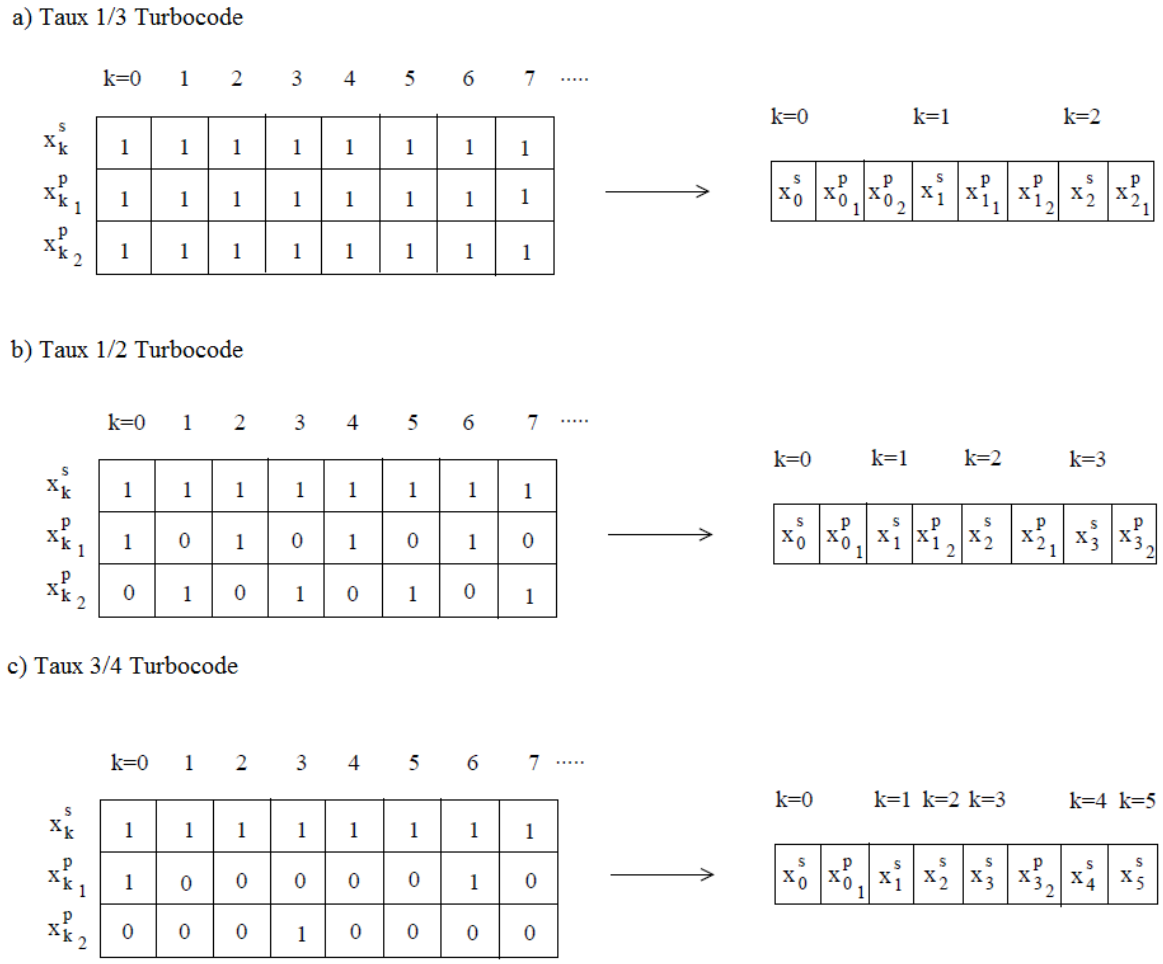

Figure 2.12 Principe de poinçonnage.

2.6 Turbodécodeur

La structure générale d'un turbodécodeur itératif est montrée dans la figure 2.13. Deux décodeurs (deux SISO) sont liés par des entrelaceurs dans une structure semblable à celle de l'encodeur turbo. Chaque décodeur reçoit trois entrées, les bits systématiquement codés, les bits de parité transmis de l'encodeur associé et l'information de l'autre décodeur au sujet des valeurs probables des bits concernés. Cette information de l'autre décodeur est désignée sous le nom de l'information à priori. Les décodeurs doivent exploiter les deux ; les entrées soft par le canal et l'information à priori. Ils doivent également fournir des sorties soft[5] pour les bits décodés. Les décodeurs doivent également donner les probabilités associées à chaque bit qui a été correctement décodé. Les sorties soft sont typiquement représentées en termes appelés rapports de logarithme de vraisemblance (LLRs). Ces LLRs sont décrits dans la section 2.6.1. Deux

[5] Information brute sans quantification au niveau du décodeur.

décodeurs appropriés sont l'algorithme de Viterbi de Sortie Soft (SOVA) [19] et l'algorithme de MAP [20], qui sont décrits dans les sections 2.10 et 2.7 respectivement.

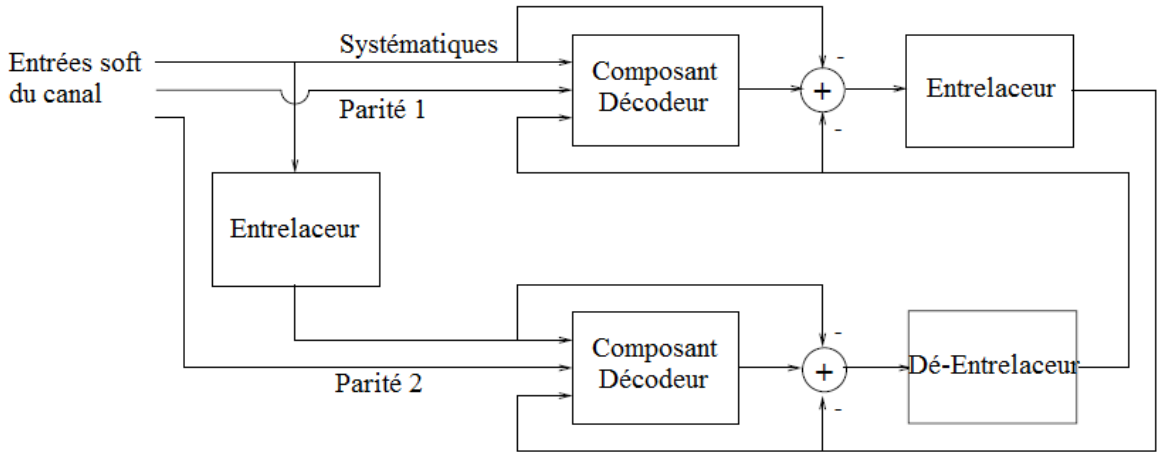

Figure 2.13 Schéma d'un turbodécodeur.

Le décodeur de la figure 2.13 fonctionne itérativement. Dans la première itération, le premier décodeur prend seulement les valeurs d'entrées soft par le canal (bits systématiques, bits de parité et les LLRs d'information à priori mis à zéro), et fournit une sortie soft exprimant une estimation des bits d'information. La sortie soft du premier décodeur est alors employée comme étant une information additionnelle pour le deuxième décodeur, ce dernier utilise cette information avec les bits reçus pour estimer les bits d'information.

Dans la deuxième itération, le premier décodeur décode les bits reçus avec des informations additionnelles sur les valeurs des bits d'entrée fournis par la sortie du deuxième décodeur dans la première itération. Cette information additionnelle permet au premier décodeur d'obtenir une sortie soft plus précise, qui est alors employée par le deuxième décodeur comme étant information à priori. Ce cycle est répété, et on remarque qu'à chaque itération le taux d'erreur par bit (BER) s'améliore. Après un certain nombre d'itération le BER reste constant.

En raison de la nature itérative du décodage, le concept d'information extrinsèque et intrinsèque a été employé dans le travail original de Berrou décrivant le décodage itératif des turbocodes [21].

Dans ce qui suit, les concepts et les algorithmes utilisés dans le décodage itératif des turbocodes seront abordés [22].

2.6.1 Rapports de Logarithme de vraisemblance

Le concept des rapports de logarithme de vraisemblance (LLRs) a été montré par Robertson [9] pour simplifier le passage d'information du premier décodeur au deuxième décodeur dans l'opération du décodage itératif des turbocodes. Le LLR d'un bit u_k de données est dénoté par

$L(u_k)$ et est défini pour être simplement le logarithme du rapport des probabilités du bit prenant ses deux valeurs possibles, on a alors :

$$L(u_k) \triangleq \ln\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right) \tag{2.8}$$

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

Etant donné le LLR $L(u_k)$, il est possible de calculer la probabilité de u_k=+1 ou u_k=-1 comme suit. Rappelons que $P(u_k=-1)=1-P(u_k=+1)$ et en prenant l'exposant des deux côtés dans l'équation (2.8), nous pouvons écrire :

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)} \tag{2.9}$$

Ainsi :

$$P(u_k = +1) = \frac{e^{L(u_k)}}{1 + e^{L(u_k)}} = \frac{1}{1 + e^{-L(u_k)}} \tag{2.10}$$

De la même façon

$$P(u_k = -1) = \frac{1}{1 + e^{+L(u_k)}} = \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}} \tag{2.11}$$

Et par conséquent, nous pouvons écrire :

$$P(u_k = \pm 1) = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}\right) \cdot e^{\pm L(u_k)/2} \tag{2.12}$$

Comme le LLR $L(u_k)$ est basé sur les probabilités sans conditions $P(u_k=\pm 1)$, nous sommes également intéressés par le calcul des LLRs basé sur des probabilités conditionnelles. Par

exemple, dans la théorie du codage de canal nous sommes intéressés par la probabilité de u_k=±1 conditionnée par la séquence reçu $\underline{y}$. Le LLR conditionnel $L(u_k/\underline{y})$ est défini par:

$$L\left(u_k/\underline{y}\right) \triangleq \ln\left(\frac{P\left(u_k = +1/\underline{y}\right)}{P\left(u_k = -1/\underline{y}\right)}\right) \tag{2.13}$$

Les probabilités conditionnelles $P(u_k = \pm 1/\underline{y})$ sont connues comme étant des probabilités à posteriori du bit décodé u_k. Ces probabilités sont fournies par les décodeurs d'entrée soft et de sortie soft (SISO) décrits dans les sections précédentes.

Indépendamment du LLR conditionnel $L(u_k/\underline{y})$ basé sur les probabilités à posteriori $P(u_k = \pm 1/\underline{y})$, nous employons également le LLR conditionnel basé sur la probabilité pour que la sortie du canal serait y_k étant donné que le x_k transmis correspondant au bit +1 ou -1. Ce LLR conditionnel est noté par $L(y_k/x_k)$ et est défini par :

$$L(y_k/x_k) \triangleq \ln\left(\frac{P(y_k/x_k = +1)}{P(y_k/x_k = -1)}\right) \tag{2.14}$$

Noter la différence conceptuelle entre les définitions de $L(u_k/\underline{y})$ dans l'équation (2.13) et $L(y_k/x_k)$ dans l'équation (2.14).

Si nous supposons que le bit transmis x_k=±1 a été envoyé en employant un canal gaussien ou un canal d'évanouissement en utilisant la modulation BPSK, alors nous pouvons écrire la probabilité de y_k à la sortie du canal comme :

$$P(y_k/x_k = +1) = \frac{e^{-\left(\frac{E_b}{2\sigma^2}\cdot(y_k - a)^2\right)}}{\sigma\sqrt{2\pi}} \tag{2.15}$$

Où E_b est l'énergie transmise par bit, σ^2 est la variance de bruit et a est l'amplitude d'évanouissement (nous avons a = 1 pour le canal AWGN). De même, nous avons :

$$P(y_k/x_k = -1) = \frac{e^{-\left(\frac{E_b}{2\sigma^2}\cdot(y_k + a)^2\right)}}{\sigma\sqrt{2\pi}} \tag{2.16}$$

Par conséquent, si nous utilisons la modulation BPSK dans un canal gaussien (probablement à évanouissement), nous pourrons réécrire l'équation (2.14) comme suit:

$$L(y_k/x_k) = \frac{E_b}{2\sigma^2} \cdot 4a \cdot y_k = L_c \cdot y_k \tag{2.17}$$

Avec :

$$L_c = 4a \cdot \frac{E_b}{2\sigma^2} \tag{2.18}$$

L_c est défini comme étant la valeur de fiabilité du canal, et dépend seulement du SNR et de l'amplitude d'évanouissement du canal. Par conséquent, pour la BPSK utilisée dans un canal gaussien (probablement à évanouissement), le LLR conditionnel $L(y_k/x_k)$, qui est désigné sous le nom de sortie soft du canal, est simplement y_k à la sortie du canal multiplié par la valeur de fiabilité du canal L_c.

Après avoir donné l'expression de logarithme de vraisemblance, nous procédons maintenant à décrire l'algorithme à posteriori maximum, qui est l'un des algorithmes d'entrée soft et sortie soft (SISO : soft-in soft-out) utilisés dans le turbodécodage itératif [22].

2.7 Algorithme MAP

2.7.1 Introduction et préliminaire mathématiques

En 1974, on a proposé un algorithme, connu sous le nom d'algorithme à posteriori maximum (MAP), par Bahl, Cocke, Jelinek et Raviv pour estimer les probabilités à posteriori des états et les transitions d'une source de Markov observée et soumise au bruit sans mémoire. Cet algorithme est également devenu l'algorithme de BCJR, baptisé des noms de ses inventeurs. Pour les codes convolutionels, l'algorithme est optimal en terme de réduction au minimum du taux d'erreur par bit décodé, à la différence de l'algorithme de Viterbi [23] qui réduit au minimum la probabilité d'un chemin incorrect par le treillis choisi par le décodeur. Néanmoins, comme indiqué par Bahl [20], dans la plupart des applications les performances des deux algorithmes seraient presque identiques.

Le MAP fait la correction bit par bit. Il examine chaque chemin possible du treillis pour décider lequel est le chemin correct (mot de code correct), ce qui traduit par une augmentation en complexité. Pour cette raison et la raison des performances presque identique avec l'algorithme de Viterbi, le MAP était ignoré pour longtemps et il n'était pas utilisé jusqu'à l'arrivé des turbocodes, car il s'adapte avec la philosophie des turbocodes et le décodage itératif. C'est-à-dire, il forme un SISO qui accepte une entrée soft et il fournit une sortie soft.

Nous employons la règle de Bayes à plusieurs reprises dans toute cette section. Cette règle donne la probabilité conjointe de a et de b, P(a∧b), en termes de probabilité conditionnelle de a si b donnée, comme :

$$P(a \wedge b) = P(a/b) \cdot P(b) \tag{2.19}$$

Une conséquence utile de règle de Bayes est que :

$$P[(a \wedge b)/c] = P[a/(b \wedge c)] \cdot P(b/c) \tag{2.20}$$

Ce qui peut être dérivé de l'équation (2.19) en considérant x≡ a∧b et y≡ b∧c comme suit. De l'équation (2.19) nous pouvons écrire :

$$\begin{aligned} P[(a \wedge b)/c] &= P(x/c) = P(x \wedge c)/P(c) = P(a \wedge y)/P(c) \\ &= P[a/(b \wedge c)] \cdot P(b/c) \end{aligned} \tag{2.21}$$

L'algorithme MAP donne, pour chaque bit décodé u_k la probabilité que ce bit est +1 ou -1 si la séquence reçue $\underline{y}$ est donnée. Comme expliqué dans la section 2.7.1, ceci est équivalent à trouver le LLR à posteriori $L(u_k/\underline{y})$, où :

$$L\left(u_k/\underline{y}\right) = \ln\left(\frac{P(u_k = +1/\underline{y})}{P(u_k = -1/\underline{y})}\right) \tag{2.22}$$

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

$$L\left(u_k/\underline{y}\right) = \ln\left(\frac{P(u_k = +1 \wedge \underline{y})}{P(u_k = -1 \wedge \underline{y})}\right) \tag{2.23}$$

Maintenant considérons la figure 2.14 qui montre les transitions possibles de l'RSC (K=3, h_0=7, h_1=5), que nous avons employée comme composant du turbocodeur dans la majeure partie de ce travail.

Pour K=3, on a quatre états possibles. Etant donné un code binaire, on a pour chaque état deux transitions possibles, une si le bit d'entrée est -1 (montrée par une ligne continue) et l'autre si le bit de sortie est +1 (montrée par une ligne discontinue).

On peut voir de la figure 2.14 que si l'état précédent S_{k-1} et l'état présent S_k sont connus, alors la valeur de bit d'entrée u_k causée par la transition de S_{k-1} à S_k est aussi connue.

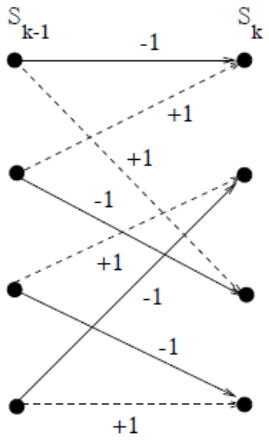

Figure 2.14 Transitions possibles pour un RSC de K=3.

Les quatre transitions possibles de S_{k-1} à S_k quand u_k=+1 sont montrées avec des lignes discontinues, c'est-à-dire que ces transitions sont mutuellement exclusives (une seule peut être réalisée par le RSC). Alors la probabilité pour que u_k=+1 est égale à la somme des probabilités des quatre transitions possibles causées par le bit d'entrée +1. Le même raisonnement s'applique dans le cas de u_k=-1. Ainsi, on peut réécrire l'équation (2.23) comme suit :

$$L\left(u_k/\underline{y}\right) = \ln\left(\frac{\sum_{(s',s)u_k=+1} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})}{\sum_{(s',s)u_k=-1} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})}\right) \qquad (2.24)$$

Où $(s',s)_{u_k=+1}$ est l'ensemble des transitions possibles de l'état précédent S_{k-1} à l'état présent S_k faites par l'entrée de bit +1 au RSC. Le même raisonnement s'applique pour $(s',s)_{u_k=+1}$. Pour réduire l'écriture on va remplacer l'écriture $P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})$ par l'écriture $P(s' \wedge s \wedge \underline{y})$.

On peut diviser la séquence $\underline{y}$ en trois sections : $\underline{y}_k$ le mot code reçu associé avec la transition présente, $\underline{y}_{j<k}$ la séquence reçue avant cette transition et $\underline{y}_{j>k}$ la séquence reçue après la transition. Cette division est montrée dans la figure 2.15. Ainsi, on peut écrire la probabilité $P(s' \wedge s \wedge \underline{y})$ comme suit :

$$P\left(s' \wedge s \wedge \underline{y}\right) = P\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k \wedge \underline{y}_{j>k}\right) \tag{2.25}$$

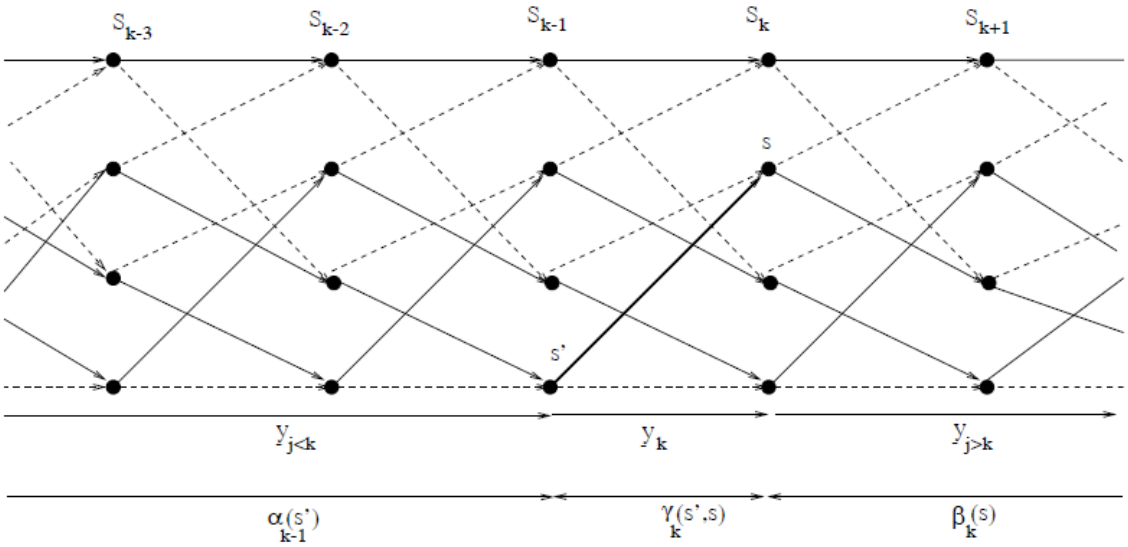

Figure 2.15 Treillis d'un décodeur pour un RSC de K=3.

Si on suppose que le canal est sans mémoire, alors $\underline{y}_{j>k}$ ne dépend que de l'état présent s et non de s'. A l'aide de règle de Bayes P(a∧b)= P(a/b)·P(b), on peut écrire :

$$\begin{aligned} P\left(s' \wedge s \wedge \underline{y}\right) &= P\left[\underline{y}_{j>k}/\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right)\right] \cdot P\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) \\ &= P\left(\underline{y}_{j>k}/s\right) \cdot P\left(s' \wedge s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) \end{aligned} \tag{2.26}$$

Encore avec la règle de Bayes l'équation (2.26) peut se réduire à :

$$\begin{aligned} P\left(s' \wedge s \wedge \underline{y}\right) &= P\left(\underline{y}_{j>k}/s\right) \cdot P\left[\left(\underline{y}_k \wedge s\right)/s'\right] \cdot P\left(s' \wedge \underline{y}_{j<k}\right) \\ &= \beta_k(s) \cdot \gamma_k(s', s) \cdot \alpha_{k-1}(s') \end{aligned} \tag{2.27}$$

Où

$$\alpha_{k-1}(s') = P\left(S_{k-1} = s' \wedge \underline{y}_{j<k}\right) \tag{2.28}$$

C'est la probabilité que le treillis est en état s' à l'instant k-1 et la séquence reçue par le canal à cet instant est $\underline{y}_{j<k}$, comme il est montrée dans la figure 2.15, et que :

$$\beta_k(s) = P\left(\underline{y}_{j>k}/S_k = s\right) \tag{2.29}$$

C'est la probabilité que le treillis est en état s à l'instant k et la séquence future à recevoir par le canal est $\underline{y}_{j>k}$. Finalement on a :

$$\gamma_k(s', s) = P\left[(\underline{y}_k \wedge S_k = s)/S_{k-1} = s'\right] \tag{2.30}$$

C'est la probabilité que le treillis était en état s' l'instant k-1, déplaçant vers s à l'instant k et la séquence reçue par le canal pour cette transition est $\underline{y}_k$.

L'équation (2.27) montre qu'on peut diviser le calcul de probabilité $P(s' \wedge s \wedge \underline{y})$ en trois calculs : α, β et γ. L'algorithme MAP permet le calcul de $\alpha_k(s)$ et $\beta_k(s)$ pour tous les états s dans tout le treillis, c'est-à-dire pour k=0, 1, 2, …, N-1, et $\gamma_k(s',s)$ pour toutes transitions possible de l'état $S_{k-1}=s'$ à l'état $S_k=s$, pour k=0, 1, 2, …, N-1. Ces valeurs sont utilisées pour trouver la probabilité $P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})$ de l'équation (2.27), qui est utilisée dans l'équation (2.24) pour donner $L(u_k/\underline{y})$ pour chaque bit u_k. Dans ce qui suit, nous allons décrire comment le calcul des coefficients $\alpha_k(s)$, $\beta_k(s)$ et $\gamma_k(s',s)$ se fait [22].

2.7.2 Calcul récursif en avant de $\alpha_k(s)$

De l'équation (2.28) nous pouvons écrire :

$$\begin{aligned}\alpha_k(s) &= P\left(S_k = s \wedge \underline{y}_{j<k+1}\right) = P\left(s \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) \\ &= \sum_{\text{tous } s'} P\left(s \wedge s' \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right)\end{aligned} \tag{2.31}$$

Utilisant la règle de Bayes et avec la supposition que le canal est sans mémoire on peut encore développer $\alpha_k(s)$ comme suit :

$$\alpha_k(s) = \sum_{\text{tous } s'} P\left(s \wedge s' \wedge \underline{y}_{j<k} \wedge \underline{y}_k\right) = \sum_{\text{tous } s'} P\left[\left(s \wedge \underline{y}_k\right)/\left(s' \wedge \underline{y}_{j<k}\right)\right] \cdot P\left(s' \wedge \underline{y}_{j<k}\right)$$

$$= \sum_{\text{tous } s'} P\left[\left(s \wedge \underline{y}_k\right)/s'\right] \cdot P\left(s' \wedge \underline{y}_{j<k}\right)$$

$$= \sum_{\text{tous } s'} \gamma_k(s', s) \cdot \alpha_{k-1}(s') \qquad (2.32)$$

Ainsi, une fois que les valeurs de $\gamma_k(s',s)$ sont connu, les valeurs de $\alpha_k(s)$ seront calculées récursivement utilisant l'équation (2.32). Supposant que le treillis à l'état initial S_0, les conditions initiales pour cette récursivité sont :

$$\alpha_0(S_0 = s) = 0$$

$$\alpha_0(S_0 = s) = 0 \qquad \text{pour tous } s \neq 0. \qquad (2.33)$$

2.7.3 Calcul récursif en arrière de $\beta_k(s)$

De l'équation (2.29) nous pouvons écrire :

$$\beta_k(s') = P\left(\underline{y}_{j>k-1}/S_{k-1} = s'\right) \qquad (2.34)$$

Utilisant la règle de Bayes sous la forme de l'équation (2.20) et avec la supposition que le canal est sans mémoire on peut encore développer $\beta_{k-1}(s')$ comme suit :

$$\beta_k(s') = P\left(\underline{y}_{j>k-1}/S_{k-1} = s'\right)$$

$$= \sum_{\text{tous } s} P\left[\left(\underline{y}_{j>k-1} \wedge s\right)/s'\right]$$

$$= \sum_{\text{tous } s} \beta_k(s) \cdot \gamma_k(s', s) \qquad (2.35)$$

Ainsi, une fois que les valeurs de $\gamma_k(s',s)$ sont connus, on peut calculer $\beta_{k-1}(s')$ récursivement en arrière par les valeurs de $\beta_k(s)$ utilisant l'équation (2.35).

Si on suppose que le treillis se termine à l'état S_0, alors les valeurs initiales sont $\beta_N(0)=1$ et $\beta_N(s)=1$ pour tous les états $s \neq 0$, comme l'avez utilisé Berrou dans [21]. Par contre si on suppose

que le treillis est sans terminaisons, c'est-à-dire que la probabilité que le treillis se termine à un état s, $\forall s$, est équiprobable. Alors les valeurs initiales[6] sont $\beta_N(s)=1/2^{K-1}$ [22].

2.7.4 Calcul de $\gamma_k(s',s)$

L'utilisation de la définition de $\gamma_k(s',s)$ de l'équation (2.30) et la règle de Bayes sous la forme de l'équation (2.20) nous donne :

$$\gamma_k(s',s) = P\left[\left(\underline{y}_k \wedge s\right)/s'\right] = P\left[\underline{y}_k/(s' \wedge s)\right] \cdot P(u_k) \tag{2.36}$$

Où u_k est le bit d'entrée nécessaire pour faire la transition de $S_{k-1}=s'$ à $S_k=s$, et $P(u_k)$ est la probabilité à priori de ce bit. De l'équation (2.12), ceci peut être écrit comme suit :

$$P(u_k) = \left(\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}}\right) \cdot e^{(u_k L(u_k)/2)} = C^{(1)}_{L(u_k)} \cdot e^{(u_k L(u_k)/2)} \tag{2.37}$$

Où

$$C^{(1)}_{L(u_k)} = \left(\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}}\right) \tag{2.38}$$

Cette équation ne dépend que de $L(u_k)$ et non de la valeur de u_k (-1 ou +1).

$P\left[\underline{y}_k/(s' \wedge s)\right]$ de l'équation (2.36) est équivalente $P(\underline{y}_k/\underline{x}_k)$ à, où $\underline{x}_k$ est le mot code transmis avec la transition de $S_{k-1}=s'$ à $S_k=s$. On suppose aussi que le canal est sans mémoire, alors :

$$P\left[\underline{y}_k/(s' \wedge s)\right] = P\left(\underline{y}_k/\underline{x}_k\right) = \prod_{l=1}^{n} P\left(\underline{y}_{kl}/\underline{x}_{kl}\right) \tag{2.39}$$

Où x_{kl} et y_{kl} sont les bits qui composent $\underline{x}_{kl}$ et $\underline{y}_{kl}$, et n est le nombre de bits (longueur de mot code). Supposant que le bit transmis $x_{kl}=\pm 1$ a été transmis en utilisant un canal gaussien ou un canal à évanouissement. En utilisant la modulation BPSK, nous pouvons écrire la probabilité $P(\underline{y}_k/\underline{x}_k)$ comme suit :

[6] Pour d'autres méthodes d'initialisation de β voir [9] et [24].

$$P(y_{kl}/x_{kl}) = \frac{e^{-\left(\frac{E_b}{2\sigma^2}\cdot(y_{kl}-a\cdot x_{kl})^2\right)}}{\sigma\sqrt{2\pi}} \tag{2.40}$$

En remplaçant l'équation (2.40) dans l'équation (2.39) on aura :

$$P\left[\underline{y}_k/(s' \wedge s)\right] = \frac{e^{-\left(\frac{E_b}{2\sigma^2}\cdot\sum_{l=1}^{n}(y_{kl}-a\cdot x_{kl})^2\right)}}{\left(\sigma\sqrt{2\pi}\right)^n} = C_{\underline{y}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)} \cdot e^{\left(\frac{E_b}{2\sigma^2}\cdot 2a\sum_{l=1}^{n} y_{kl}\cdot x_{kl}\right)} \tag{2.41}$$

Où

$$C_{\underline{y}_k}^{(2)} = \frac{e^{-\left(\frac{E_b}{2\sigma^2}\sum_{l=1}^{n} {y_{kl}}^2\right)}}{\left(\sigma\sqrt{2\pi}\right)^n} \tag{2.42}$$

Le terme de gauche dans l'équation (2.42) ne dépend que du SNR et de l'amplitude de $\underline{y}_{kl}$, et $C_{\underline{x}_k}^{(3)}$ est défini comme étant :

$$C_{\underline{x}_k}^{(3)} = e^{-\left(\frac{E_b}{2\sigma^2}\cdot a^2\cdot\sum_{l=1}^{n} {x_{kl}}^2\right)} = e^{-\left(\frac{E_b}{2\sigma^2}\cdot a^2\cdot n\right)} \tag{2.43}$$

Le terme $C_{\underline{x}_k}^{(3)}$ ne dépend que du SNR et de l'amplitude a, donc on peut écrire $\gamma_k(s',s)$ comme suit :

$$\begin{aligned}\gamma_k(s',s) &= P(u_k)\cdot P\left[\underline{y}_k/(s' \wedge s)\right] = C\cdot e^{(u_k L(u_k)/2)}\cdot e^{\left(\frac{E_b}{2\sigma^2}\cdot 2a\sum_{l=1}^{n} y_{kl}\cdot x_{kl}\right)} \\ &= C\cdot e^{(u_k L(u_k)/2)}\cdot e^{\left(\frac{L_c}{2}\sum_{l=1}^{n} y_{kl}\cdot x_{kl}\right)}\end{aligned} \tag{2.44}$$

Ainsi :

$$C = C_{L(u_k)}^{(1)} \cdot C_{\underline{y}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)} \tag{2.45}$$

Le terme C ne dépend pas de signe de u_k, ni de mot code transmis $\underline{x}_k$. C c'est une constante dans la sommation en numérateur et dénominateur de l'équation (2.24).

De l'équation (2.24) et (2.27) nous pouvons écrire :

$$L\left(u_k/\underline{y}\right) = \ln\left(\frac{\sum_{(s',s)u_k=+1} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})}{\sum_{(s',s)u_k=-1} P(S_{k-1} = s' \wedge S_k = s \wedge \underline{y})}\right)$$
$$= \ln\left(\frac{\sum_{(s',s)u_k=+1} \beta_k(s) \cdot \gamma_k(s',s) \cdot \alpha_{k-1}(s')}{\sum_{(s',s)u_k=-1} \beta_k(s) \cdot \gamma_k(s',s) \cdot \alpha_{k-1}(s')}\right) \quad (2.46)$$

Finalement, $L(u_k/\underline{y})$ est le LLR conditionnel délivré par le décodeur MAP [22].

2.7.5 Résumé des étapes pour la mise en œuvre de l'algorithme MAP

1. Initialisation des valeurs de $\alpha_0(s')$ comme il est montré dans l'équation (2.33), et de $\beta_N(s)$ par $1/2^{K-1}$ $\forall$ s,
2. Calcul des valeurs de $\gamma_k(s',s)$ selon l'équation (2.40),
3. Calcul des valeurs de $\alpha_{k-1}(s')$ selon l'équation (2.32),
4. Calcul des valeurs de $\beta_k(s)$ selon l'équation (2.35),
5. Finalement, le calcul des LLRs $L(u_k/\underline{y})$ à posteriori délivrés par le décodeur MAP utilisant l'équation (2.46).

2.8 Principes de turbodécodage itératif

2.8.1 Turbodécodage et préliminaires mathématiques

Dans cette section nous expliquons le concept de l'information extrinsèque et intrinsèque comme il a été utilisé par Berrou [21], et le décodage itératif pour les turbocodes avec les algorithmes SISO (exp. MAP).

En considérant l'expression de $\gamma_k(s',s)$ dans l'équation (2.44), on a :

$$\gamma_k(s',s) = C \cdot e^{(u_k L(u_k)/2)} \cdot e^{\left(\frac{L_c}{2}\sum_{l=1}^{n} y_{kl} \cdot x_{kl}\right)} \quad (2.47)$$

Dans les codes dits systématiques, un des n bits transmis est le bit systématique u_k. Si nous supposons qu'il est le premier bit transmit, alors $x_{k1}=u_k$, et on peut écrire l'équation (2.47) sous la forme :

$$\gamma_k(s',s) = C \cdot e^{(u_k L(u_k)/2)} \cdot e^{\left(\frac{L_c}{2} \cdot y_{ks} \cdot u_k\right)} \cdot e^{\left(\frac{L_c}{2} \cdot \sum_{l=2}^{n} y_{kl} \cdot x_{kl}\right)}$$

$$= C \cdot e^{(u_k L(u_k)/2)} \cdot e^{\left(\frac{L_c}{2} \cdot y_{ks} \cdot u_k\right)} \cdot X_k(s',s) \qquad (2.48)$$

Où y_{ks} est la valeur réceptionnée de $x_{kl}=u_k$ et

$$X_k(s',s) = e^{\left(\frac{L_c}{2} \cdot \sum_{l=2}^{n} y_{kl} \cdot x_{kl}\right)} \qquad (2.49)$$

En remplaçant l'équation (2.48) dans l'équation (2.46). On a :

$$L\left(u_k/\underline{y}\right) = \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s) \cdot \gamma_k(s',s) \cdot \alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s) \cdot \gamma_k(s',s) \cdot \alpha_{k-1}(s')}\right)$$

$$= L(u_k) + L_c \cdot y_{ks} + \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s) \cdot X_k(s',s) \cdot \alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s) \cdot X_k(s',s) \cdot \alpha_{k-1}(s')}\right)$$

$$= L(u_k) + L_c \cdot y_{ks} + L_e(u_k) \qquad (2.50)$$

Donc :

$$L_e(u_k) = \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s) \cdot X_k(s',s) \cdot \alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s) \cdot X_k(s',s) \cdot \alpha_{k-1}(s')}\right) \qquad (2.51)$$

Ainsi nous pouvons voir que le LLR à posteriori $L(u_k/\underline{y})$ calculé en utilisant l'algorithme MAP peut être considéré comme étant une somme de trois termes $L(u_k)$, $L_c \cdot y_{ks}$ et $L_e(u_k)$. Le LLR à priori $L(u_k)$ est dérivé de $P(u_k)$ dans l'expression de $\gamma_k(s',s)$ dans l'équation (2.36). Cette probabilité est dérivée d'une source indépendante (émetteur) et est appelée la probabilité à priori de $k^{ième}$ bit étant +1 ou -1. Dans plusieurs cas on ne sait pas la valeur de cette probabilité, donc on prend le LLR $L(u_k)$ égale à zéro c'est-à-dire $P(u_k)=0.5$. Cependant, en turbodécodage itératif, chaque décodeur (SISO) peut fournir à l'autre décodeur le LLR $L(u_k)$ estimé. Le second terme $L_c \cdot y_{ks}$ dans l'équation (2.50) est la sortie soft du canal du bit systématique u_k, qui a été transmis directement à travers le canal et reçu comme étant y_{ks}. Le dernier terme dans l'équation (2.50) $L_e(u_k)$ est extrait de la soustraction de $L_e(u_k)$ et $L_c \cdot y_{ks}$. C'est donc la décision propre de décodeur (algorithme) SISO. Alors $L_e(u_k)$ s'appelle le LLR extrinsèque.

L'équation (2.50) montre que l'information extrinsèque $L_e(u_k)$ du décodeur MAP peut être obtenue par la soustraction de l'information à priori $L(u_k)$ et l'entrée systématique de canal $L_c \cdot y_{ks}$, de la sortie soft $L(u_k/\underline{y})$ du décodeur SISO [22].

2.8.2 Turbodécodage itératif

Nous allons décrire maintenant comment le décodage itératif des turbocodes fonctionne. La figure 2.13 de section 2.6 montre la structure de turbodécodeur itératif, elle se répète ici pour une raison conventionnelle comme étant la figure 2.16.

Considérant premièrement le premier décodeur dans la première itération. Ce décodeur reçoit la séquence du canal $L_c \cdot \underline{y}^{(1)}$ qui contienne $L_c \cdot y_{ks}$, la forme reçue des bits systématiques. $L_c \cdot y_{kl}$ sont les bits de parité du premier encodeur (RSC1). Pour un taux de codage R=1/2 on insert des zéros à la place des bits de parité perforés. Le premier composant décodeur (SISO1) peut traiter alors les entrées soft par canal et estimer les LLRs conditionnels $L_{11}(u_k/\underline{y})$ des bits de donnée u_k, k=0, 1, 2, …N. L'indice 11 de $L_{11}(u_k/\underline{y})$ indique qu'il est le LLR à posteriori de la première itération du premier décodeur. Notons qu'en première itération le premier décodeur n'aurait pas l'information à priori des bits, donc $L(u_k)$ dans l'équation (2.44) qui donne $\gamma_k(s',s)$ est nul. Ceci correspondant à la probabilité à priori égale à 0.5.

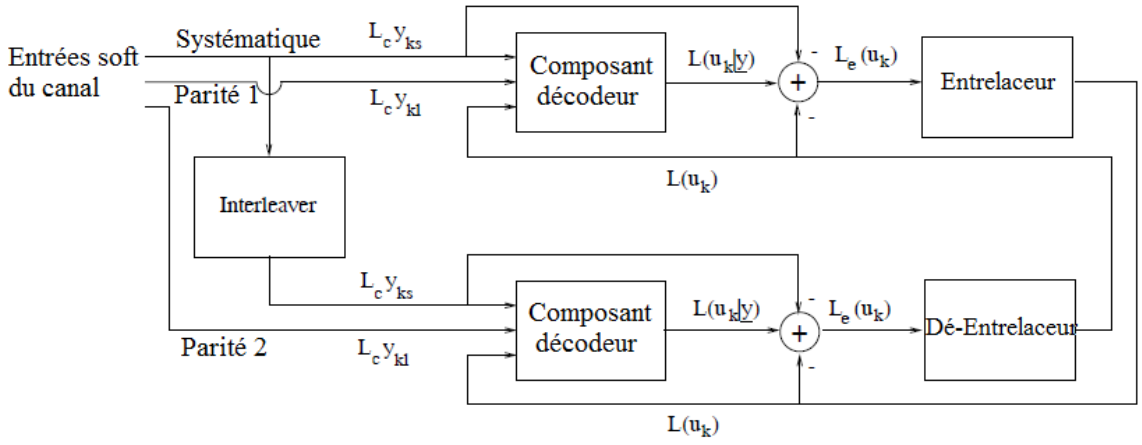

Figure 2.16 Schéma d'un turbodécodeur.

Deuxièmement le second décodeur (SISO2) entre dans l'opération. Il reçoit la séquence du canal $L_c \cdot \underline{y}^{(2)}$ qui contienne la forme reçue des bits systématiques entrelacés et les bits de parité du second encodeur (RSC2). Si le code est perforé, on insert des zéros à la place des bits perforés. En plus de la séquence reçu du canal $L_c \cdot \underline{y}^{(2)}$, le décodeur peut utiliser le LLR conditionnel $L_{11}(u_k/\underline{y})$ fournit par le premier composant décodeur comme étant LLR à priori $L(u_k)$ dans le second décodeur. Après soustraction des $L_c \cdot y_{ks}$, $L(u_k)$ de SISO1 et

l'entrelacement, on obtient l'information extrinsèque $L_e(u_k)$. Ainsi le second décodeur utilise $L_c \cdot \underline{y}^{(2)}$ et les LLRs à priori $L(u_k)$ (dérivé d'entrelacement de l'information extrinsèque $L_e(u_k)$ du premier décodeur) pour produire les LLRs à posteriori $L_{12}(u_k/\underline{y})$. Ceci est la fin de la première itération.

Pour la deuxième itération, le premier décodeur traite encore la séquence du canal reçue $L_c \cdot \underline{y}^{(1)}$, avec la présence des LLRs à priori $L(u_k)$ fournis par l'information extrinsèque $L_e(u_k)$ des LLRs à posteriori $L_{12}(u_k/\underline{y})$ fournis par le second décodeur, d'où on peut produire un LLR à posteriori $L_{21}(u_k/\underline{y})$ amélioré. Le second décodeur utilisant le LLR à posteriori $L_{21}(u_k/\underline{y})$ amélioré du premier décodeur et délivre un LLR à priori $L(u_k)$ amélioré qui est utilisé avec la séquence reçue du canal $L_c \cdot \underline{y}^{(2)}$ pour calculer $L_{22}(u_k/\underline{y})$. Ce processus itératif va continuer et à chaque nouvelle itération le BER moyen s'améliore [22].

L'algorithme MAP est extrêmement complexe à cause des multiplications nécessaires en équations (2.32) et (2.35). Dans ce qui suit, une description d'autres algorithmes SISO moins complexes que le MAP, pouvant le remplacé dans le turbodécodage itératif.

2.9 Modifications sur l'algorithme MAP

2.9.1 Introduction

L'algorithme MAP comme il est décrit en section 2.7 est plus complexe que l'algorithme de Viterbi avec la décision hard (décodage après quantification). Les performances dans la plupart des cas sont identiques. Le MAP est ignoré pour plus de 20 ans. Cependant, son utilisation en turbocodes a tiré l'attention à nouveau pour minimiser sa complexité. Initialement l'algorithme Max-Log-MAP est proposé par Koch et Baier [25] et Erfanian [26]. Cette technique transfère les récursivités en domaine logarithmique et invoque à une approximation qui réduit dramatiquement la complexité. A cause de cette approximation, les performances sont moins bonnes par rapport aux performances de MAP. Cependant, Rebertson [27] en 1995 proposent l'algorithme Log-MAP qui donne les performances identiques à celle de l'algorithme MAP, mais avec une certaine complexité [22].

2.9.2 L'algorithme Max-Log-MAP

Le Max-Log-MAP simplifie le MAP par la transformation de ces équations en domaine logarithmique et l'utilisation de l'approximation :

$$\ln\left(\sum_i e^{x_i}\right) \approx \max_i(x_i) \qquad (2.52)$$

Où $\max_i(x_i)$ prend le maximum des x_i. $A_k(s)$, $B_k(s)$ et $\Gamma_k(s',s)$ sont définis comme suit :

$$A_k(s) \triangleq \ln\left(\alpha_k(s)\right) \tag{2.53}$$

$$B_k(s) \triangleq \ln\left(\beta_k(s)\right) \tag{2.54}$$

$$\Gamma_k(s',s) \triangleq \ln\left(\gamma_k(s',s)\right) \tag{2.55}$$

On peut réécrire l'équation (2.32) comme suit :

$$A_k(s) \triangleq \ln\left(\alpha_k(s)\right) = \ln\left(\sum_{\text{tous } s'} \gamma_k(s',s)\cdot\alpha_{k-1}(s')\right) = \ln\left(\sum_{\text{tous } s'} e^{\left(A_{k-1}(s')+\Gamma_k(s',s)\right)}\right)$$
$$\approx \max_{s'}\left(A_{k-1}(s')+\Gamma_k(s',s)\right) \tag{2.56}$$

Similairement, On peut réécrire l'équation (2.35) comme suit :

$$B_{k-1}(s') \triangleq \ln\left(\beta_{k-1}(s')\right) = \ln\left(\sum_{\text{tous } s} \beta_{k-1}(s)\cdot\gamma_k(s',s)\right) = \ln\left(\sum_{\text{tous } s} e^{\left(B_k(s)+\Gamma_k(s',s)\right)}\right)$$
$$\approx \max_{s}\left(B_k(s)+\Gamma_k(s',s)\right) \tag{2.57}$$

On remarque en équations (2.56) et (2.57) que le calcul de la probabilité d'état fait par la sélection de métrique maximale des chemins, et que la métrique du chemin est la somme de la métrique du chemin à l'instant k-1 plus la métrique de branche à l'instant k qui est équivalente à la récursivité de l'algorithme de Viterbi.

En utilisant l'équation (2.44), la métrique de branche $\Gamma_k(s',s)$ peut être écrite comme suit :

$$\Gamma_k(s',s) \triangleq \ln\left(\gamma_k(s',s)\right) = \ln\left(C\cdot e^{(u_k L(u_k)/2)}\cdot e^{\left(\frac{L_c}{2}\cdot\sum_{l=1}^{n} y_{kl}\cdot x_{kl}\right)}\right)$$
$$= \hat{C} + \frac{u_k L(u_k)}{2} + \frac{L_c}{2}\cdot\sum_{l=1}^{n} y_{kl}\cdot x_{kl} \tag{2.58}$$

Où $\hat{C}=\ln(C)$ est une constante qui ne dépend pas de u_k ou bien de mot code transmit $\underline{x}_k$. La métrique de branche est équivalente à celle utilisé dans l'algorithme de Viterbi, avec l'addition du LLR à priori $u_k \cdot L(u_k)$.

Finalement, de l'équation (2.46) nous pouvons écrire les LLRs à posteriori $L(u_k/\underline{y})$ qui sont calculés par le Max-Log-MAP comme suit :

$$L\left(u_k/\underline{y}\right) = \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} \beta_k(s)\cdot\gamma_k(s',s)\cdot\alpha_{k-1}(s')}{\sum_{(s',s)_{u_k=-1}} \beta_k(s)\cdot\gamma_k(s',s)\cdot\alpha_{k-1}(s')}\right)$$

$$= \ln\left(\frac{\sum_{(s',s)_{u_k=+1}} e^{\left(A_{k-1}(s')+\Gamma_k(s',s)+B_k(s)\right)}}{\sum_{(s',s)_{u_k=-1}} e^{\left(A_{k-1}(s')+\Gamma_k(s',s)+B_k(s)\right)}}\right)$$

$$\approx \max_{(s',s)\Rightarrow u_k=+1}\left(e^{\left(A_{k-1}(s')+\Gamma_k(s',s)+B_k(s)\right)}\right) - \max_{(s',s)\Rightarrow u_k=-1}\left(e^{\left(A_{k-1}(s')+\Gamma_k(s',s)+B_k(s)\right)}\right) \qquad (2.59)$$

2.9.3 Algorithme Log-MAP (correction de l'approximation)

L'algorithme Max-Log-MAP donne une dégradation en performance comparée à l'algorithme MAP à cause de l'approximation de l'équation (2.52). Quand elle est utilisée en turbodécodage itératif, Robertson [27] trouvait que cette dégradation est de 0.35 dB. Cependant, l'approximation de l'équation (2.52) peut être rendu exacte par l'utilisation de logarithme Jacobien :

$$\ln(e^{x_1}+e^{x_2}) = \max(x_1,x_2) + \ln\left(1+e^{-|x_1-x_2|}\right) = \max(x_1,x_2) + f_c(|x_1-x_2|)$$

$$= g(x_1,x_2) \qquad (2.60)$$

Où $f_c(x)$ peut être considéré comme le terme de correction. C'est alors la base d'algorithme Log-MAP proposé par Robertson, Villerbrun et höher [27]. Similairement à l'algorithme Max-Log-MAP, les valeurs $A_k(s) \triangleq \ln\left(\alpha_k(s)\right)$ et $B_k(s) \triangleq \ln\left(\beta_k(s)\right)$ sont calculées par l'utilisation d'une récursivité en avant et en arrière.

2.10 Algorithme SOVA (Soft Output Viterbi Algorithm)

2.10.1 Description de SOVA

Dans cette section nous décrivons une variante de l'algorithme de Viterbi, appelé l'algorithme de Viterbi à sortie soft (SOVA) [19, 28]. Cet algorithme a deux modifications qui permettent d'être exploitable par le décodeur (SISO) du turbodécodeur. Premièrement la métrique du

chemin est modifiée pour tenir compte de l'information à priori quand on sélectionne le chemin de maximum de vraisemblance (ML) à travers le treillis. Deuxièmement l'algorithme est modifié pour fournir des sorties soft sous la forme de LLR $L(u_k/\underline{y})$ pour chaque bit décodé.

Pour la première modification, on considère la séquence d'états $\underline{s}_k^s$ jusqu'à l'instant k dans le treillis. La probabilité que ce chemin (séquence) est correct à travers le treillis est donnée par :

$$P\left(\underline{s}_k^s/\underline{y}_{j\le k}\right) = \frac{P\left(\underline{s}_k^s \wedge \underline{y}_{j\le k}\right)}{P\left(\underline{y}_{j\le k}\right)} \tag{2.61}$$

Parce que la probabilité de la séquence reçue $P(\underline{y}_{j\le k})$ est constante pour tous les chemins à travers le treillis jusqu'à l'instant k, la probabilité que ce chemin est correct à travers le treillis est proportionnelle à $P(\underline{s}_k^s \wedge \underline{y}_{j\le k})$. Donc la métrique doit être définie par le fait que quand elle est maximisée, ceci implique la maximisation de $P(\underline{s}_k^s \wedge \underline{y}_{j\le k})$. On suppose que le canal soit sans mémoire, on a :

$$P\left(\underline{s}_k^s \wedge \underline{y}_{j\le k}\right) = P\left(\underline{s}_{k-1}^{s'} \wedge \underline{y}_{j\le k-1}\right) \cdot P\left(S_k = s \wedge \underline{y}_k/S_{k-1} = s'\right) \tag{2.62}$$

Soit $M(\underline{s}_k^s)$ la métrique de chemin pour $\underline{s}_k^s$, où :

$$\begin{aligned} M\left(\underline{s}_k^s\right) &= \ln\left(P\left(\underline{s}_k^s \wedge \underline{y}_{j\le k}\right)\right) \\ &= M\left(\underline{s}_{k-1}^{s'}\right) + \ln\left(P\left(S_k = s \wedge \underline{y}_k/S_{k-1} = s\right)\right) \end{aligned} \tag{2.63}$$

En utilisant l'équation (2.30) nous avons alors :

$$M\left(\underline{s}_k^s\right) = M\left(\underline{s}_{k-1}^{s'}\right) + \ln\left(\gamma_k(s', s)\right) \tag{2.64}$$

Où $\gamma_k(s',s)$ est la probabilité de transition pour le chemin de $S_{k-1}= s'$ à $S_k=s$. De l'équation (2.58) nous pouvons écrire :

$$\ln\left(\gamma_k(s', s)\right) = \Gamma_k(s', s) = \hat{C} + \frac{u_k L(u_k)}{2} + \frac{L_c}{2} \cdot \sum_{l=1}^{n} y_{kl} \cdot x_{kl} \tag{2.65}$$

Parce que le terme $\hat{C}$ est constant, on peut le négligé et nous pouvons réécrire l'équation (2.64) comme suit :

$$M\left(\underline{s}_k^s\right) = M\left(\underline{s}_{k-1}^{s'}\right) + \frac{u_k L(u_k)}{2} + \frac{L_c}{2} \cdot \sum_{l=1}^{n} y_{kl} \cdot x_{kl} \tag{2.66}$$

La métrique dans l'algorithme SOVA est mise en forme comme dans le cas de l'algorithme de Viterbi, avec le terme additionnel $u_k \cdot L(u_k)$ qui est l'information à priori prise en compte. Notons que c'est équivalant à la récursivité en avant dans l'équation (2.56) utilisée pour calculer $A_k(s)$ dans l'algorithme Max-Log-MAP.

On étudie maintenant la deuxième modification, c'est-à-dire la donnée issue de la sortie soft de l'algorithme de Viterbi. Dans un treillis binaire, on a deux chemins pour atteindre l'état à l'instant k. On calcule la métrique de ces deux chemins selon l'équation (2.66). Si $\underline{s}_k^s$ et $\hat{\underline{s}}_k^s$ sont deux chemins qui peuvent atteindre l'état S_k=s à l'instant k, et qui ont les deux métrique de chemins respectivement $M(\underline{s}_k^s)$ et $M(\hat{\underline{s}}_k^s)$, pour déterminer le chemin le plus correct on définit la notion de la différence de métrique Δ_k^s comme suit :

$$\Delta_k^s = M\left(\underline{s}_k^s\right) - M\left(\hat{\underline{s}}_k^s\right) \tag{2.67}$$

Alors, si $\Delta_k^s > 0$ donc le chemin $\underline{s}_k^s$ est le plus probable d'être correct, et si $\Delta_k^s < 0$, alors le chemin $\hat{\underline{s}}_k^s$ est le plus probable d'être correct. Δ_k^s est le LLR de bit u_k en instant k. On peut généraliser ce résultat pour avoir les LLRs $L(u_k/\underline{y})$ pour tous les états comme suit :

$$L\left(u_k/\underline{y}\right) = |\Delta_k^s|_{\max(M(\underline{s}_k^s))} \quad \text{pour } s = 0, 1, 2, \dots, 2^{K-1} \tag{2.68}$$

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

L'algorithme considéré dans cette section est le moins complexe des algorithmes de décodage SISO discutés dans ce chapitre. Dans [27], l'algorithme SOVA est moins complexe que l'algorithme Max-Log-MAP. Cependant, l'algorithme SOVA est aussi mois précis que les autres algorithmes décrits dans ce chapitre, et lorsqu'il est utilisé en turbodécodage itératif, les performances sont moindre de 0.6 dB [27] que le décodage qui utilise l'algorithme MAP.

2.10.2 Résumé des étapes de la mise en œuvre de l'algorithme SOVA

1. Initialisation des valeurs de $M(\underline{s}_0^s)$ $\forall$ s,
2. Calcul de $M(\underline{s}_k^s)$ et $M(\hat{\underline{s}}_k^s)$ $\forall$ s en utilisant l'équation (2.66),
3. Calcul de Δ_k^s en utilisant l'équation (2.67);
4. Calcul des LLRs $L(u_k/\underline{y})$ à posteriori délivrés par le décodeur SOVA en utilisant l'équation (2.68).

2.11 Conclusion

Dans ce chapitre, nous avons décrit les turbocodes classiques et les techniques utilisées en turbodécodage itératif. On s'est intéressé surtout à l'algorithme de décodage, ou on a décrit quatre algorithmes : le MAP, Log-MAP, Max-Log-MAP et le SOVA. L'algorithme MAP semble meilleur, mais il est extrêmement complexe à cause du nombre de multiplications présentes dans les équations (2.32), (2.35) et (2.44). L'algorithme Log-MAP réduit la complexité de l'algorithme MAP par sa transformation en domaine logarithmique (réduit l'effet des multiplications) en gardant les mêmes performances. L'algorithme Max-Log-MAP simplifie la complexité de MAP par l'approximation de l'équation (2.52), mais ceci dégrade les performances de 0.35 dB [27] par rapport à l'algorithme MAP. Le dernier algorithme SOVA simplifie lui aussi la complexité de l'algorithme MAP, mais il présente une dégradation en performances de 0.6 dB [27] par rapport à l'algorithme MAP.

Chapitre 3

Optimisation des entrelaceurs

3.1 Introduction

Qu'on l'appelle entrelacement ou permutation, la technique consistant à disperser des données dans le temps a toujours rendu de grands services en communications numériques. On l'utilise avec profit par exemple pour réduire les effets des atténuations plus ou moins longues dans les transmissions affectées d'évanouissements et plus généralement dans des situations où des perturbations peuvent altérer des symboles consécutifs. Dans le cas des turbocodes aussi, la permutation permet de lutter efficacement contre l'apparition de paquets d'erreurs, sur l'une au moins des dimensions du code composite. Mais son rôle ne s'arrête pas là : elle détermine aussi, en relation étroite avec les propriétés des codes constituants, la distance minimale du code concaténé [30].

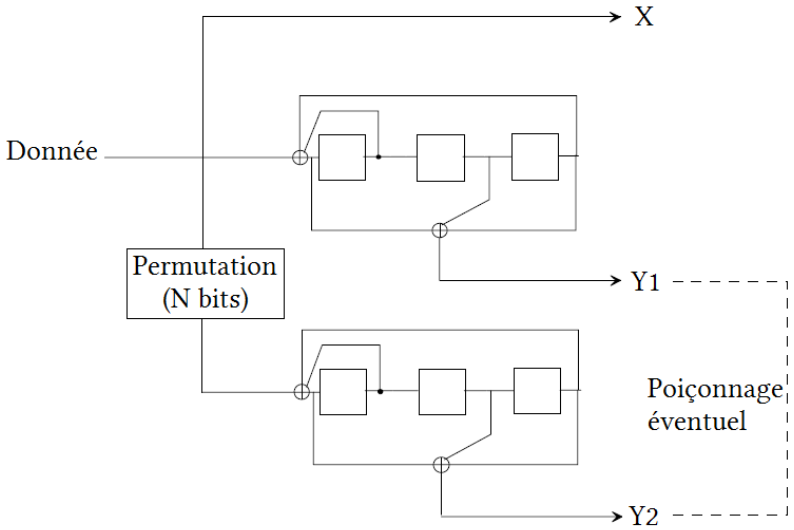

Figure 3.1. Un turbocode binaire (15, 13)₈.

Considérons le turbocode représenté en figure 3.1. La plus mauvaise des permutations qu'on puisse utiliser est bien sûr la permutation identité, qui minimise la diversité du codage (on a alors Y1 = Y2). À l'opposé, la meilleure des permutations que l'on pourrait imaginer mais qui n'existe probablement pas, permettrait au code concaténé d'être équivalent à une machine séquentielle dont le nombre d'états irréductibles serait 2N+6. Il y a en effet N+6 éléments binaires de mémorisation dans la structure : N pour la mémoire de permutation et 6 pour les deux codes convolutifs. Si on pouvait assimiler cette machine séquentielle à un codeur convolutif et pour les valeurs usuelles de N, le nombre d'états correspondant serait très grand, en tout cas assez grand pour garantir une large distance minimale. Par exemple, un codeur convolutif avec une mémoire de code de 60 (10^{18} états !) affiche une distance libre de l'ordre de la centaine (pour R = 1/2), ce qui est bien suffisant.

Ainsi, de la plus mauvaise à la meilleure des permutations, le choix est large et l'on n'a pas encore découvert de permutation parfaite. Cela dit, de bonnes permutations ont quand même pu être définies pour élaborer des schémas de turbocodage normalisés.

Il existe deux manières de spécifier une permutation, la première par des équations liant les adresses avant et après permutation, la seconde par un tableau (look-up table) fournissant la correspondance entre les adresses. La première est préférable du point de vue de la simplicité dans la spécification du turbocode (les comités de normalisation sont sensibles à cet aspect) mais la seconde peut conduire à de meilleurs résultats car le degré de liberté est généralement plus grand dans le travail de conception de la permutation [30].

3.2 La permutation régulière

Le point de départ dans la conception d'un entrelacement est la permutation régulière, qui est décrite en figure 3.2 sous deux formes différentes. La première suppose que le bloc contenant N bits peut être organisé comme un tableau de L lignes et C colonnes. L'entrelacement consiste alors à écrire les données dans une mémoire ad hoc, ligne par ligne, et à les lire colonne par colonne (figure 3.2(a)). La seconde s'applique sans hypothèse sur la valeur de N. Après écriture des données dans une mémoire linéaire (adresse j, $0 \leq j \leq N-1$), le bloc est assimilé à un cercle, les deux extrémités (j = 0 et j = N−1) étant alors contiguës (figure 3.2(b)). Les données binaires sont alors extraites de telle sorte que la i-ième donnée lue ait été préalablement écrite à la place j, de valeur :

$$j = \pi(i) = Pi + j_0 \mod(N) \tag{3.1}$$

Où P est un entier premier avec N et j_0 est l'indice de départ.

Pour une permutation circulaire, définissons la distance spatiale cumulée $S(i_1, i_2)$ comme la somme des deux distances spatiales séparant deux bits, avant et après permutation, dont les indices de lecture sont i_1 et i_2 :

$$S(i_1, i_2) = f(i_1, i_2) + f(\pi(i_1), \pi(i_2)) \tag{3.2}$$

où :

$$f(u, v) = \min\{|u - v|, N - |u - v|\} \tag{3.3}$$

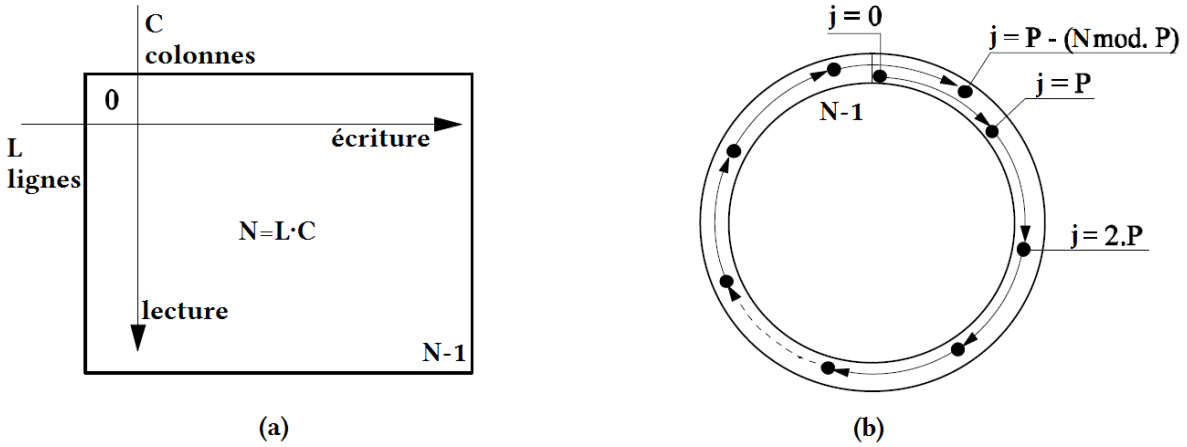

Figure 3.2. Permutation régulière sous forme rectangulaire (a) et circulaire (b).

La fonction f est introduite pour tenir compte du caractère circulaire des adresses. Finalement, nous appelons S_{min} la plus petite des valeurs de $S(i_1, i_2)$, pour toutes les paires i_1 et i_2 possibles:

$$S_{min} = \min_{i_1, i_2}\{S(i_1, i_2)\} \tag{3.4}$$

Il est démontré dans [31] que la borne supérieure de S_{min} est :

$$\sup S_{min} = \sqrt{2N} \tag{3.4}$$

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

$$P = P_0 = \sqrt{2N} \tag{3.6}$$

Et :

$$N = \frac{P_0}{2} \mod(P_0) \tag{3.7}$$

Considérons maintenant une séquence de poids quelconque qui s'écrit, après permutation :

$$\tilde{d}(D) = \sum_{i=0}^{N-1} a_i D^i \tag{3.8}$$

Où a_i peut prendre la valeur binaire 0 (pas d'erreur) ou 1 (une erreur) et, avant permutation :

$$d(D) = \sum_{j=0}^{N-1} a_j D^j = \sum_{i=0}^{N-1} a_{\pi(i)} D^{\pi(i)} \tag{3.9}$$

Notons i_{min} et i_{max} les indices i correspondant aux première et dernière valeurs a_i non nulles dans $\tilde{d}(D)$. Nous définissons de la même manière j_{min} et j_{max} pour la séquence d(D). Alors, la permutation régulière satisfaisant (3.6) et (3.7) garantit la propriété :

$$(i_{max} - i_{min}) + (j_{max} - j_{min}) > \sqrt{2N} \tag{3.10}$$

Cela vient de ce que d(D) et $\tilde{d}(D)$, toutes deux considérées entre les indices min et max, contiennent au moins 2 bits dont la distance spatiale cumulée, telle que définie par (3.2), est maximale et égale à √2N. Il nous faut maintenant considérer deux cas :

- les séquences d(D) et $\tilde{d}(D)$ sont toutes deux du type RTZ (Return To Zero) simple, c'est-à-dire qu'elles débutent dans l'état 0 du codeur et y reviennent une seule fois, à la fin. Les bits de parité produits par ces séquences sont des 1, une fois sur deux statistiquement. Compte tenu de (3.10), pour des valeurs usuelles de N (N > 100), les poids de la redondance sont élevés et ces séquences RTZ ne contribuent pas à la DMH (Distance Minimale de Hamming) du turbocode.
- l'une au moins des séquences d(D) et $\tilde{d}(D)$ est du type RTZ multiple, c'est-à-dire qu'elle correspond à plusieurs passages par l'état 0 du codeur. Si ces passages par l'état 0 sont

longs, la parité associée à la séquence peut être de poids réduit et la distance associée faible. Généralement, dans ce type de situation, les séquences avant et après permutation sont toutes deux RTZ multiples.

La performance d'un turbocode, à faible taux d'erreurs, est intimement liée à la présence de motifs RTZ multiples et la permutation régulière n'est pas une bonne réponse pour éliminer ces motifs [30].

3.3 Rôle de désordre

En admettant toujours que les motifs d'erreurs qui ne sont pas RTZ ont des poids suffisamment élevés pour ne pas avoir d'incidence sur la performance, une permutation idéale pour un turbocode pourrait être définie par la règle suivante :

Si une séquence est du type RTZ avant permutation, alors elle ne l'est plus après permutation et vice-versa.

La règle précédente est impossible à satisfaire en pratique et un objectif plus réaliste est :

Si une séquence est du type RTZ avant permutation, alors elle ne l'est plus après permutation ou elle est devenue une séquence RTZ simple longue et vice-versa.

Le dilemme dans la conception de bonnes permutations pour turbocodes réside dans la nécessité de satisfaire cet objectif pour deux classes distinctes de séquences d'entrée qui demandent des traitements opposés : les séquences RTZ simples et les séquences RTZ multiples, telles que définies plus haut. Pour bien mettre ce problème en évidence, considérons un turbocode de rendement 1/3, avec une permutation rectangulaire régulière (écriture suivant L lignes, lecture suivant C colonnes) portant sur des blocs de N = LC bits (figure 3.3). Les codeurs élémentaires sont des codeurs à 8 états dont la période est 7 (générateur de récursivité 15).

Le premier motif (a) de la figure 3.3 concerne une séquence d'erreurs possible de poids d'entrée w = 2 : 10000001 pour le code C_1, qu'on appellera aussi code horizontal. Il s'agit de la séquence RTZ minimale de poids 2 pour le codeur considéré. La redondance produite par ce codeur est de poids 6 (exactement : 11001111). La redondance produite par le codeur vertical C_2, pour lequel la séquence considérée est aussi RTZ (sa longueur est multiple de 7), est autrement plus informative parce que RTZ simple et délivrée sur sept colonnes. En admettant que Y_2 est égal à 1 une fois sur deux en moyenne, le poids de cette redondance est environ $w(Y_2) \approx 7L/2$. Lorsqu'on fait tendre N vers l'infini à travers les valeurs de L et C ($L \approx C \approx \sqrt{N}$), la redondance produite par l'un des deux codes, pour ce type de motif, tend aussi vers l'infini. On dit alors que le code est bon.

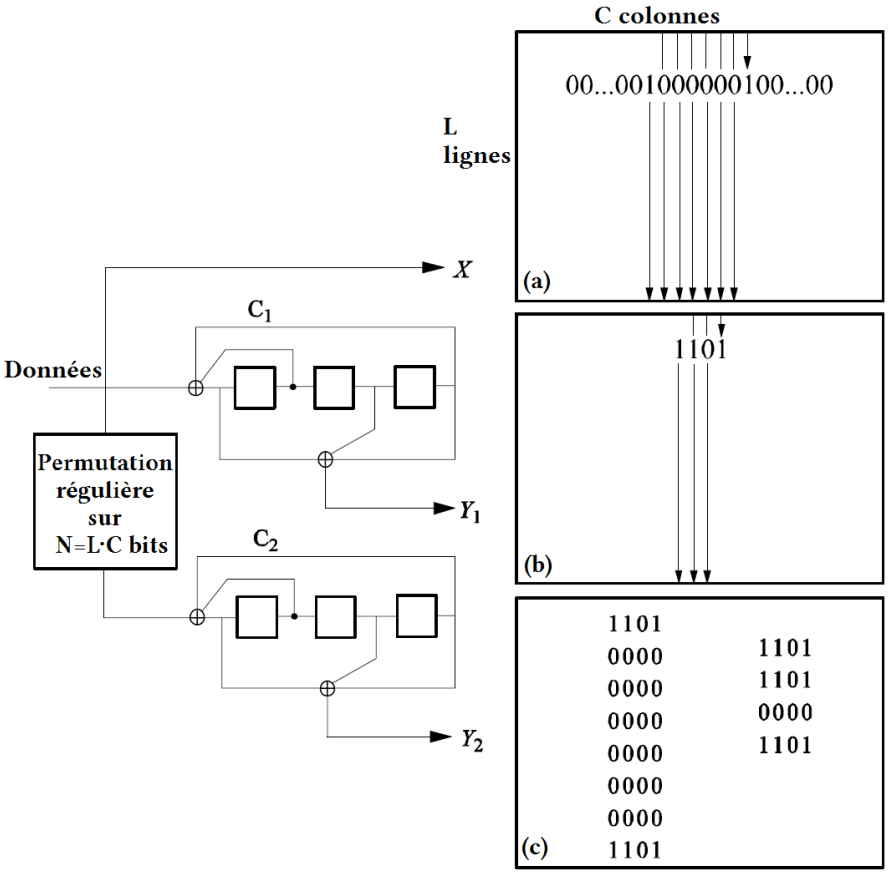

Figure 3.3. Exemple de motifs d'erreurs possibles avec une permutation régulière.

Le second motif (b) est celui de la séquence RTZ minimale de poids d'entrée 3. Là aussi, la redondance est pauvre sur la première dimension et bien plus informative sur la seconde. Les conclusions sont les mêmes que précédemment.

Les deux autres dessins (c) représentent des exemples de séquences RTZ multiples, constituées de courtes séquences RTZ sur chacune des deux dimensions. Les poids d'entrée sont 6 et 9. Les distances associées à ces motifs (respectivement 30 et 27 pour ce code de rendement 1/3) ne sont généralement pas suffisantes pour assurer une bonne performance à faible taux d'erreurs. De plus, ces distances sont indépendantes de la taille du bloc et donc, vis-à-vis des motifs considérés, le code n'est pas bon.

La permutation régulière est donc une bonne permutation pour la classe des motifs d'erreurs RTZ simple. Pour les motifs RTZ multiples en revanche, la permutation régulière n'est pas appropriée. Une bonne permutation se doit de « casser » la régularité des motifs composites rectangulaires comme ceux de la figure 3.3(c), en introduisant un certain désordre. Mais cela ne doit pas se faire au détriment des motifs pour lesquels la permutation régulière est bonne. Le désordre doit donc être bien géré ! C'est là tout le problème dans la recherche d'une permutation qui doit conduire à une distance minimale suffisamment élevée. Une bonne permutation ne peut

être trouvée indépendamment des propriétés des codes élémentaires, de leurs motifs RTZ, de leurs périodicités, etc. [30].

3.4 Permutation irrégulière

Nous ne ferons pas, dans cette section, une description de toutes les permutations irrégulières qui ont pu être imaginées jusqu'à ce jour et qui ont fait l'objet de nombreuses publications ou de plusieurs chapitres d'ouvrages (voir [32] par exemple). Nous avons plutôt retenu de présenter ce qui semble être, pour le moment, le type de permutation à la fois le plus simple et le plus performant. Il s'agit de permutations circulaires presque régulières, appelées ARP (almost regular permutation, [33]), DRP (dithered relative prime, [34]), DGI (dithered golden interleaver, [16]) et finalement notre entrelaceur dimensionnel DI (dimensional interleaver, [35]) suivant les auteurs. Dans tous les cas, l'idée est de ne pas trop s'éloigner de la permutation régulière, bien adaptée aux motifs d'erreurs RTZ simples et d'instiller un petit désordre contrôlé pour contrer les motifs d'erreurs RTZ multiples [30].

3.4.1 L'entrelaceur DRP

La figure 3.4 donne un exemple, tiré de [34], de ce que peut être ce petit désordre. Avant que soit effectuée la permutation circulaire régulière, les bits subissent une permutation locale. Cette permutation s'effectue par groupes de C_W bits. C_W, qui est le cycle du désordre à l'écriture (writing cycle), est un diviseur de la longueur k du message. De même, une permutation locale de cycle C_R (reading cycle) est appliquée avant la lecture définitive.

En pratique C_W et C_R peuvent être de valeurs identiques $C_W = C_R = C$, typiquement, 4 ou 8. Cette manière d'introduire du désordre, par de petites fluctuations locales, ne diminue pas significativement la distance spatiale cumulée, dont la valeur maximale est $\sqrt{2}N$. Elle permet en retour de supprimer les motifs d'erreurs comparables à ceux des figures 3.3(b) et 3.3(c) à la condition que les hauteurs et largeurs de ces motifs ne soient pas tous deux multiples de C [30].

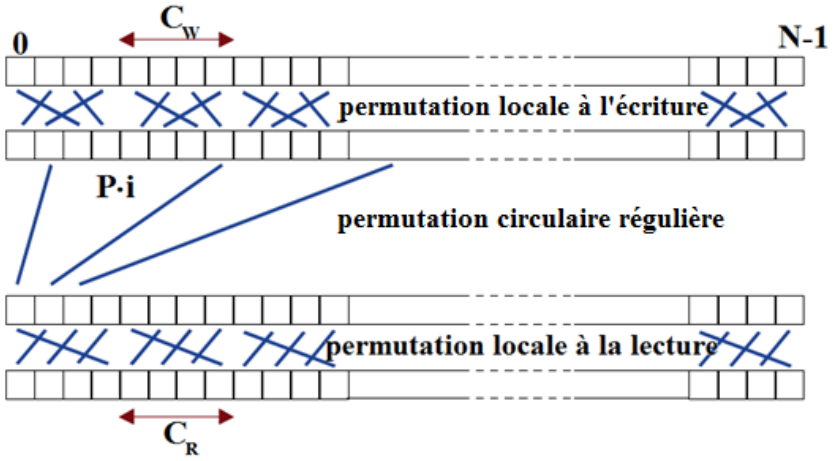

Figure 3.4. principe d'entrelaceur DRP.

3.4.2 L'entrelaceur ARP

Une autre manière de perturber la permutation régulière de façon contrôlée est décrite par la figure 3.5. La permutation est représentée ici sous sa forme rectangulaire, visuellement plus accessible, mais elle s'applique très bien aussi à la permutation circulaire. Une information (bit ou symbole) est positionnée à chaque croisement de ligne et de colonne. Avec la permutation régulière, ces données sont donc mémorisées ligne par ligne et lues colonne par colonne. Dans la figure 3.5, le désordre est introduit par l'intermédiaire de quatre vecteurs de déplacement V_1, · · ·, V_4 qui sont alternativement appliqués lors de la lecture. Ces vecteurs sont de petite amplitude par rapport aux dimensions de la matrice de permutation.

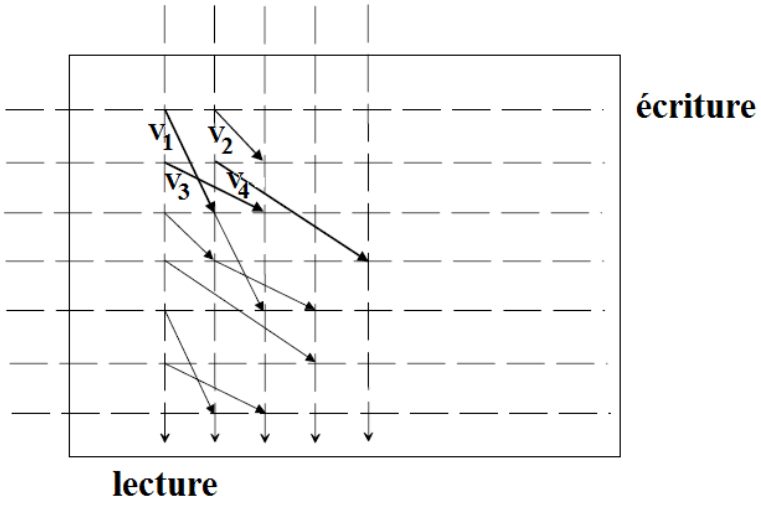

Figure 3.5. représentation rectangulaire d'entrelaceur ARP.

Le modèle mathématique associé à cette permutation presque régulière, sous sa forme circulaire, est une extension de (3.1) :

$$j = \pi(i) = Pi + Q(i) + j_0 \mod(N) \tag{3.11}$$

Si l'on choisit :

$$Q(i) = A(i)P + B(i) \tag{3.12}$$

Où les entiers positifs A(i) et B(i) sont périodiques, de cycle C (diviseur de N), alors ces grandeurs correspondent à des décalages positifs appliqués respectivement avant et après la permutation régulière. C'est la différence avec la permutation décrite par la figure 3.4, dans

laquelle les perturbations en écriture et en lecture se réalisent à l'intérieur de petits groupes de données et non par décalage.

Pour que la permutation soit bien une bijection, les paramètres A(i) et B(i) ne sont pas quelconques. Une condition suffisante pour assurer l'existence de la permutation est que les paramètres soient tous multiples de C. Cette condition n'est pas très contraignante vis-à-vis de l'efficacité de la permutation. (3.12) peut alors être récrit sous la forme :

$$Q(i) = C\big(\alpha(i)P + \beta(i)\big) \tag{3.13}$$

Où α(i) et β(i) sont le plus souvent de petits entiers, de valeurs 0 à 8. Par ailleurs, puisque les propriétés d'une permutation circulaire ne sont pas modifiées par une simple rotation, l'une des valeurs Q(i) peut être systématiquement 0 [30].

3.4.3 Les entrelaceurs par section d'or

a) Section d'or

La section d'or surgit dans beaucoup de problèmes mathématiques intéressants. La figure 3.6 illustre le principe de section d'or en relation avec le problème d'entrelacement. Soit un segment de ligne de longueur 1, le problème est de diviser ce segment en longueur égal à g, et un segment plus court de longueur égal à 1-g, tel que le rapport du segment le plus long au segment entier est identique au rapport du segment le plus court au segment plus long.

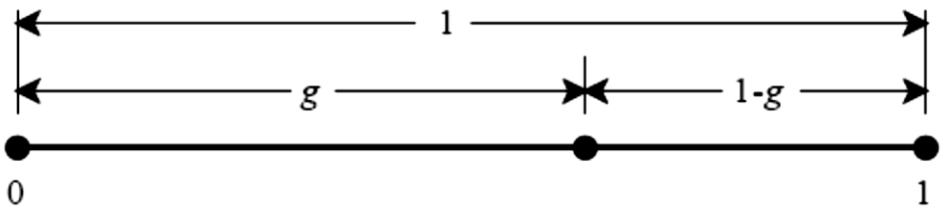

Figure 3.6. Illustration du principe de section d'or.

$$\frac{g}{1} = \frac{1-g}{g} \tag{3.14}$$

La solution de cette équation quadratique simple pour g donne la valeur de section d'or

$$g = \frac{\sqrt{5} - 1}{2} \approx 0.618 \tag{3.15}$$

Considérant maintenant les points produits en commençant à 0 et en ajoutant des incréments de g, en utilisant l'arithmétique modulo 1. Après le premier incrément, on dispose de deux points à 0 et à g, en utilisant l'arithmétique modulo1. Des distances de modulo 1 sont employées pour tenir compte de l'option d'avoir le premier point à n'importe quelle position. De l'équation (3.14), la distance de 1-g est la même que g^2. Après le deuxième incrément, le premier et le troisième point déterminent la distance minimale qui égale à g^3. Après le troisième incrément, le premier et le quatrième point déterminent la distance minimale qui égale à g^4. La distance minimale après le cinquième point est la même. La distance minimale après le sixième point est g^5. Les mêmes distances peuvent également être produites avec (1-g) = $g^2 \approx 0.382$. Des puissances plus élevées peuvent également être utilisées pour la valeur d'incrément, mais les distances minimales initiales sont réduites à la valeur la plus petite d'incrément.

La figure 3.7 montre un tracé des distances minimales en fonction du nombre de points considérés, car des points sont ajoutés en utilisant un incrément de g avec l'arithmétique modulo 1. La figure 3.7 montre également une limite (borne) supérieure pour chaque nombre spécifique de points. C'est-à-dire, les n points donnés, qui pourraient être uniformément espacés avec une distance minimum de 1/n. Naturellement les résultats d'incrément de section d'or sont valides pour tous les nombres de points en même temps. La limite supérieure ne l'est pas. Néanmoins, les résultats d'incrément de section d'or sont montrés pour se rapprocher de la limite supérieure. Noter cela même lorsque la distance minimale chute, la plupart des points seront toujours à la même distance minimale précédente des points voisins, avec une distance moyenne entre les points égale à la limite supérieure.

Les propriétés de distance de l'incrémentation de section d'or, illustrées sur la figure 3.7, sont souhaitables pour les entrelaceurs en général, mais sont en particulier souhaitables pour les entrelaceurs de turbocode. On montre maintenant comment ces propriétés peuvent être utilisées en concevant un certain nombre d'entrelaceurs pratiques [16].

b) L'entrelaceur GRPI (Golden relative prime interleaver)

Pour les entrelaceurs GRPI, la fonction d'entrelaceur est calculée comme suit :

$$\pi(i) = Pi + s \mod(N) \quad i = 0, \dots, N-1 \tag{3.16}$$

Où s est un nombre entier utilisé comme valeur initiale, P est un indice d'incrément entier, et N est la longueur d'entrelaceur. N et P doivent être des nombres relativement premiers pour s'assurer que chaque élément est lu une fois et seulement une fois. L'indice d'initialisation s est

habituellement placé à 0, mais d'autres valeurs de nombre entier s peuvent être choisies. L'indice premier relatif P est choisi "proche" d'une des valeurs du nombre non entier suivant :

$$N(g^m + w)/h \tag{3.17}$$

Où g est la valeur de section d'or, m est un entier positif plus grand que zéro, h est l'indice d'espacement (distance) entre les éléments voisins à étaler au maximum, et w est un nombre entier modulo h. Comme il a été montré précédemment, les valeurs préférées pour m sont en général 1 ou 2. Dans une mise en place typique où des éléments adjacents doivent être au maximum étalés, w est placé à 0 et h à 1 [16].

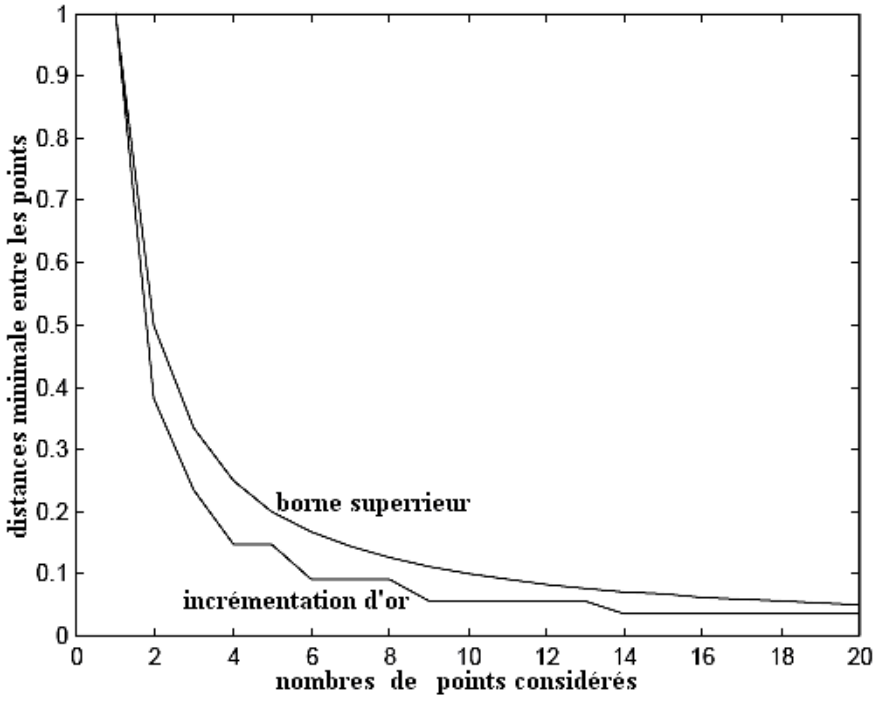

Figure 3.7. Distance minimale entre les points en fonction du nombre de points avec un incrément d'or.

c) L'entrelaceur GI (Golden interleaver)

Les entrelaceurs GI (entrelaceur d'or) n'utilisent pas le nombre entier premier et l'arithmétique de modulo du nombre entier, mais plutôt ils utilisent le tri des nombres à valeurs réelles dérivées de la section d'or. La première étape est de calculer la valeur de section d'or g. La deuxième étape est de calculer la valeur réelle P d'incrément définie par (3.17). La troisième étape est de produire le vecteur d'or v à valeurs réelles. Les éléments de v sont calculés comme suit :

$$v(i) = Pi + s \mod(N), \quad i = 0, \dots, N-1 \tag{3.18}$$

Où s est n'importe quelle valeur initiale réelle. L'étape suivante consiste à trier (ordre décroissant des valeurs) le vecteur d'or v et de trouver le vecteur d'indice z qui définit ce tri. C'est-à-dire, trouver le vecteur z de tri tels que a(i) = v (z(i)), i = 0… N-1, où a = tri(v). Les indices de l'entrelaceur d'or sont alors données par $\pi(z(i)) = i$, i = 0… N-1. En fait, le vecteur z est l'entrelaceur inverse pour π.

La valeur initiale de s est habituellement placée à 0, mais d'autres valeurs réelles de s peuvent être choisies. Les valeurs préférées pour m sont en général 1 ou 2. Pour étaler au maximum les éléments adjacents, w est placé à 0 et h à 1 [16].

d) L'entrelaceur DGI

On a trouvé que pour les turbocodes, les entrelaceurs possédant un aspect aléatoire tendent à des performances meilleures que des entrelaceurs complètement structurés, particulièrement pour les blocs de grande taille de l'ordre de 1000 bits ou plus. Cependant, les propriétés de la distance spatiale cumulée d'entrelaceurs d'or sont toujours très souhaitables afin de maintenir une bonne distance minimum et assurer la convergence rapide en étalant efficacement l'erreur rafale dans tout le bloc. Ces deux dispositifs sont entourés dans l'entrelaceur d'or perturbé (dithered golden interleaver). La seule différence entre l'entrelaceur GI et l'entrelaceur DGI est l'inclusion d'un désordre dans le vecteur d'or v. C'est-à-dire,

$$v(i) = Pi + b(i) + s \mod(N), \quad j = 0, \dots, N-1 \tag{3.19}$$

Où b(i) est la i-ième composante du vecteur de désordre. Le désordre est une distribution aléatoire uniforme entre 0 et ND, où D est la largeur normalisée du désordre. Le vecteur d'or perturbé v est trié (ordre décroissant des valeurs) et les indices d'entrelaceur sont produits d'une façon semblable à l'entrelaceur d'or (GI) décrit précédemment.

Expérimentalement, pour les turbocodes et pour n'importe quelle longueur de bloc, D est prise égale à 0.01.

On remarque que l'entrelaceur d'or (GI) est maintenant un cas particulier de l'entrelaceur d'or perturbé (DGI) avec D = 0 [16].

3.4.4 L'entrelaceur dimensionnel DI

Dans notre courante approche[7], on va créer un nouveau concept pour représenter l'indice d'incrémentation d'un entrelaceur basé sur l'expression de la borne sphérique. La borne sphérique est développée par E. Boutillon et D. Gnaedig dans la référence [31] à l'aide de la théorie d'empilements de sphères dans un volume fini (sphere packings). L'utilisation d'expression de la borne sphérique à la place de P (équation (3.20)) nous permet de transformer l'étude de l'indice d'incrémentation en étude de sa dimension. Ce nouveau concept que nous avons nommé dimension d'entrelaceur est donné pour P en équation (3.19) (composante régulière) par :

$$P = \left(N^{(d-1)} \cdot \Gamma(d+1)\right)^{\frac{1}{d}} \tag{3.20}$$

Où d est la dimension de l'entrelaceur régulier. La valeur maximum[8] n de d est donnée par :

$$\Gamma(n+1) = N \tag{3.21}$$

L'addition d'une composante aléatoire (un désordre aléatoire) à la composante régulière améliore sa dispersion et donc la performance contre les motifs d'erreurs RTZ multiples. La composante aléatoire b(i) dans la structure DI est une distribution aléatoire uniforme entre 0 et 2μN. Où μ est la moyenne normalisée de la distribution. μ est déterminée par le modèle expérimental suivant :

$$\mu = \left(1 - \frac{1}{\Gamma(d+1)}\right)\left(\frac{1}{\Gamma(n^{\gamma_r}+1)} - \frac{1}{\Gamma(n+1)}\right) \tag{3.22}$$

Où γ_r est la dispersion normalisée de la composante aléatoire (entrelaceur aléatoire). Dans ce qui suit nous allons décrire comment déterminer la dimension d'incrémentation de la composante aléatoire [35].

[7] C'est la version finale du réf [35] qui est en cours de validations sous titre " Interleaver design for turbo code based on the sphere bound expression " chez Annals of telecommunications, Springer-Verlag France.

[8] La fonction Gamma est défini par : $\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt$

3.4.4.1 une nouvelle définition de la dispersion normalisée aléatoire

Nous proposons une nouvelle formulation pour mesurée la dispersion normalisée aléatoire réalisée par des permutations aléatoires locales :

$$\gamma_r = 1 - \left(1 - \frac{2}{\Gamma(n+1)}\right)\log_n(r) \tag{3.23}$$

Où r est la dimension d'entrelaceur aléatoire.

La dispersion normalisée moyenne d'une permutation aléatoire est autour de 0.81 [32]. Alors, on peut calculer la valeur r à partir de l'équation (3.23). Si on remplace cette valeur dans l'expression de la borne sphérique, elle va donner des valeurs proches de 4 dans le cas des paquets courts et des valeurs proches de 8 dans le cas des paquets longs. Donc, cette nouvelle définition explique et résoude le choix de la longueur du désordre (permutation locale) M dans l'entrelaceur DRP et l'entrelaceur ARP. Alors, un autre moyen de sélectionner r est de résoudre l'équation suivante :

$$M = \left(N^{(r-1)} \cdot \Gamma(r+1)\right)^{\frac{1}{r}} \tag{3.24}$$

Enfin on peut réécrire la moyenne normalisée μ dans l'équation (3.22) comme suit :

$$\mu = \left(1 - \frac{1}{\Gamma(d+1)}\right)\left(\frac{1}{\Gamma\left(n^{(1-(2/\Gamma(n+1))\cdot\log_n(r))}+1\right)} - \frac{1}{\Gamma(n+1)}\right) \tag{3.25}$$

Alors, les entrelaceurs DGI et DRP deviennent maintenant des cas particuliers de l'entrelaceur DI avec $P = N(g^m+w)/h$ et $P = \text{floor}(\sqrt{2}N)$, respectivement [35].

3.5 Résultats et comparaison entre performances

Il est important de décrire les conditions dans lesquelles nous avons fait nos simulations. Un simulateur d'un système de communication numérique programmé en Matlab a été utilisé comme base de nos simulations sur les turbocodes. Une partie des programmes de ce simulateur est inspirée des programmes de Yufei [29]. Ce simulateur a été programmé en Matlab dans le but de tester rapidement est efficacement les changements en cours. La machine utilisée est un PC Intel (i3 CPU 3.0 GHZ).

On a utilisé des paquets de longueurs variables N (appelée aussi taille d'entrelaceur). On a pris N=120 bits (paquet court), N=512 bits (paquet moyen) et N=1024 bits (paquet long). Le turbocode utilise deux codeurs convolutifs systématiques en parallèle avec les polynômes générateurs $(7, 5)_8$ et un taux de code R fixé à 1/2. Le nombre maximum d'itérations pour le

décodage est 5. Le Log-MAP est utilisé comme algorithme de décodage. La méthode de terminaison est utilisée seulement pour le premier RSC. On a pris aussi la modulation BPSK parce qu'elle représente un bon choix pour les systèmes de communication satellitaires. Dans notre simulation, on a utilisé le canal AWGN. Ce dernier est un bon modèle pour représenter les différents types de systèmes de communication, surtout où la liaison radioélectrique est directe (canaux satellitaires par exemple). Notons aussi que la méthode de Monte Carlo avec un intervalle de confidence égal à 95% est comptée pour arrêter la simulation [36].

La figure 3.8 montre une comparaison en performance entre le DGI avec D=0.01 pour N=1024 bits, le DRP (P=45, M=8), et le DI avec les deux différentes valeurs de (d, r, s) comme suit :

1. (d=2, r=1.404, s=0) qui correspond à une structure similaire à celle du DRP.
2. (d=4.88, r=1.404, s=0) qui correspond à une structure similaire à celle du DGI.

D'après les courbes de la figure 3.8, le DRP et le DI (d=2, r=1.404, s=0) ont une performance identique. Outre, on peut observer une différence en performance entre les entrelaceurs DI (d=4.88, r=1.404, s=0), DRP et DGI. À un BER=1.25×10^{-5}, on a un E_b/N_0=2.5 dB pour l'entrelaceur DGI, E_b/N_0=2.3 dB pour l'entrelaceur DRP et E_b/N_0=2.15 dB pour l'entrelaceur DI (d=4.88, r=1.404, s=0). Une différence en puissance de 0.35 dB entre le DI (d=4.88, r=1.404, s=0) et le DGI est très significative comme gain surtout dans les communications satellitaires, et une différence en puissance de 0.2 dB entre le DI (d=4.88, r=1.404, s=0) et le DRP est aussi significative dans les communications satellitaires.

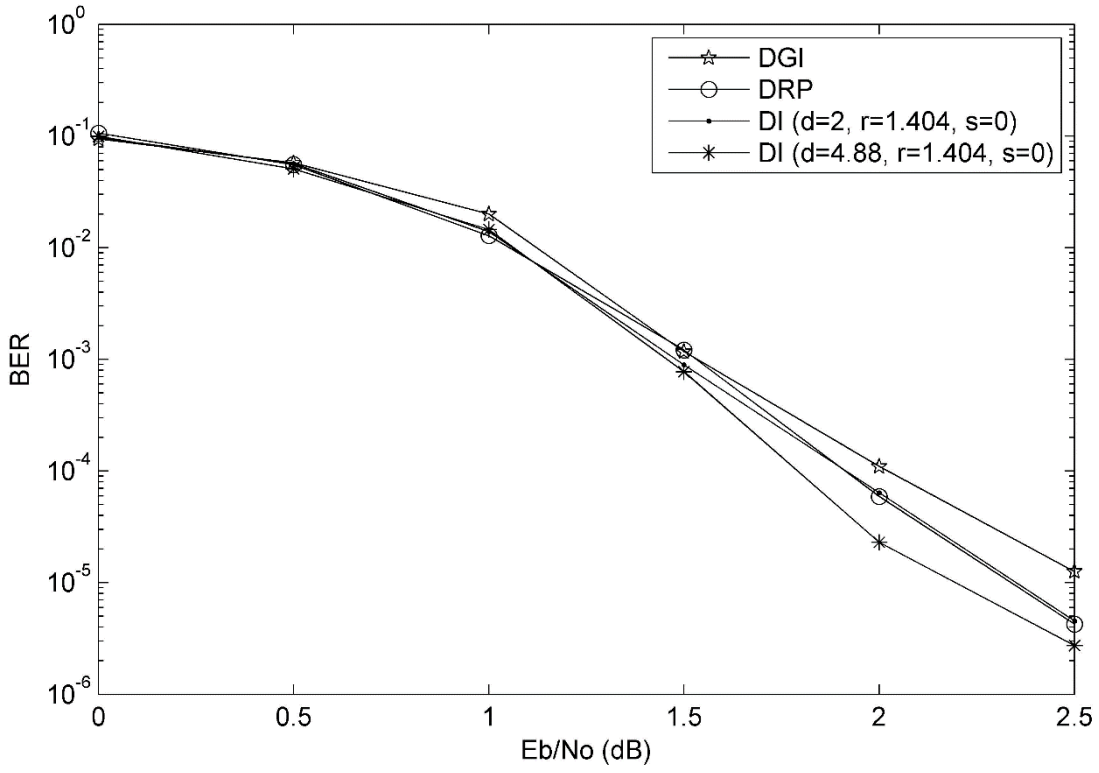

Figure 3.8. BER pour N=1024 bits.

La figure 3.9 montre une comparaison en performance entre le DGI avec D=0.01 pour N=512 bits, le DRP (P=31, M=8), et le DI avec les deux différents valeurs de (d, r, s) comme suit :

1. (d=2, r=1.441, s=0) qui correspond à une structure similaire à celle du DRP.
2. (d=4.555, r=1.441, s=0) qui correspond à une structure similaire à celle du DGI.

D'après les courbes de la figure 3.9, le DRP et le DI (d=2, r=1.441, s=0) ont une performance identique. Outre, on peut observer une différence en performance entre les entrelaceurs DI (d=4.555, r=1.441, s=0), DRP et DGI. À un BER=3.32×10^{-6}, on a un E_b/N_0=3 dB pour l'entrelaceur DGI, E_b/N_0=2.9 dB pour l'entrelaceur DI (d=4.555, r=1.441, s=0) et E_b/N_0=2.75 dB pour l'entrelaceur DRP. Une différence en puissance de 0.1 dB entre le DI (d=4.555, r=1.441, s=0) et le DGI est significative dans les communications satellitaires, et une différence en puissance de 0.25 dB entre le DI (d=2, r=1.441, s=0) et le DGI est aussi significative dans les communications satellitaires.

La figure 3.10 montre une comparaison en performance entre le DGI avec D=0.01 pour N=120 bits, le DRP (P=13, M=4), et le DI avec les deux différents valeurs de (d, r, s) comme suit :

1. (d=1.888, r=1.353, s=0) qui correspond à une structure similaire à celle du DRP.
2. (d=3.839, r=1.353, s=0) qui correspond à une structure similaire à celle du DGI.

D'après les courbes de la figure 3.10, tous les entrelaceurs ont presque les mêmes performances.

À partir des résultats des figures 3.8, 3.9 et 3.10 on peut conclure que l'influence de désordre sur la performance d'entrelaceur augmente avec l'augmentation de la taille de N.

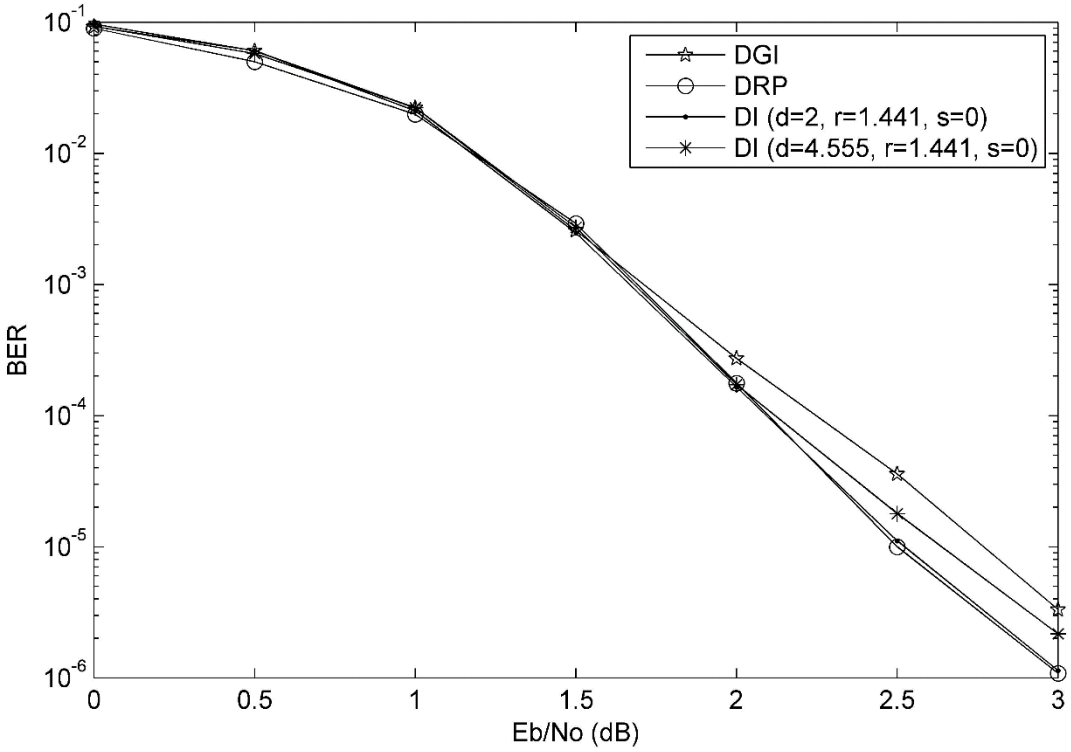

Figure 3.9. BER pour N=512 bits.

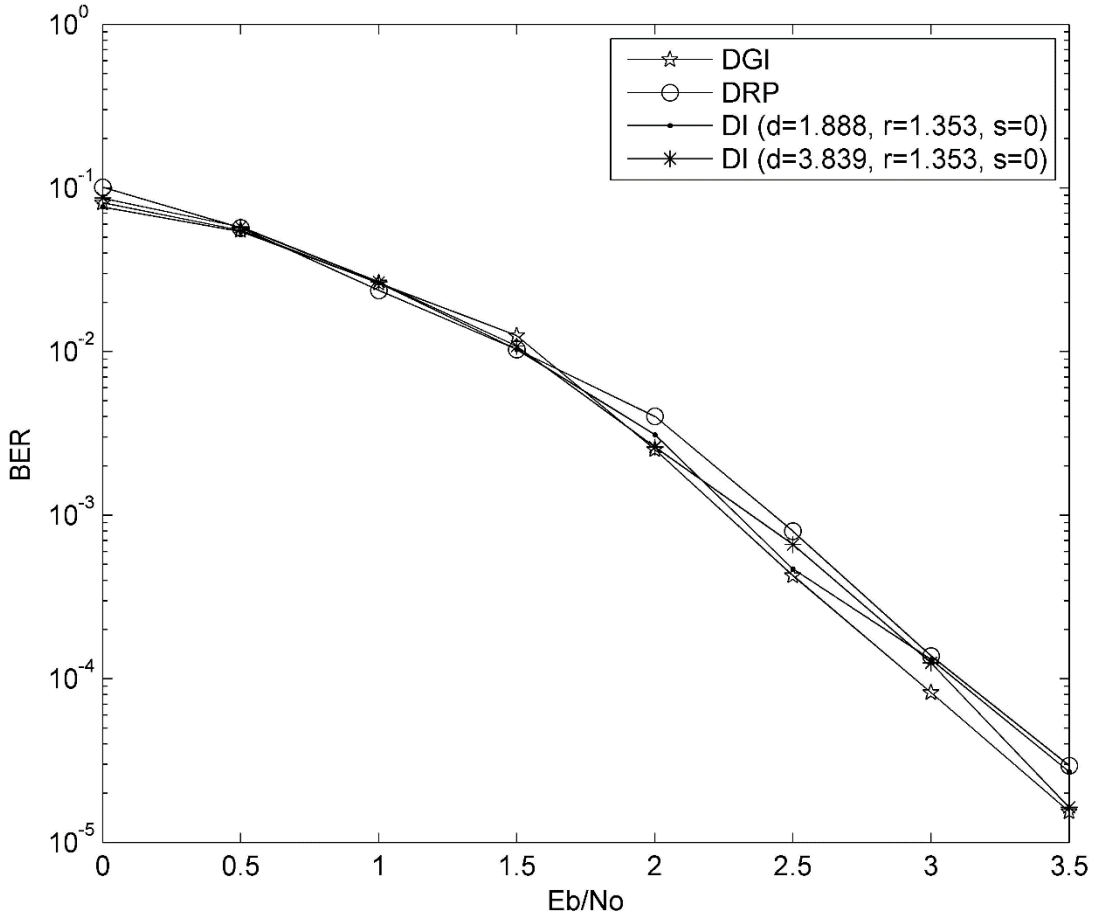

Figure 3.10. BER pour N=120 bits.

3.6 Conclusion

Au chapitre 3, nous avons analysé en détail à l'aide du principe des séquences RTZ (Return To Zero), le rôle de l'entrelaceur dans l'opération de lutte contre les paquets d'erreurs (burst errors). Ensuite, nous avons représenté quelques entrelaceurs populaires aujourd'hui tels que le DGI, le DRP et l'ARP. Ce chapitre est finalisé par la représentation de notre entrelaceur DI proposé et bien sûr comparé en performance avec ces derniers entrelaceurs. Les résultats discutés montrent une bonne performance d'entrelaceur DI.

Conclusion générale

Conclusions

Ce travail concerne une contribution à une amélioration de l'efficacité de puissance tout en cherchant de trouver un bon compromis.

L'utilisation du turbocode permet d'améliorer l'efficacité de puissance d'un système de communication par rapport aux autres codes classiques. En fait, les performances en terme d'efficacité de puissance sont meilleurs que celles de toute technique antérieure, et par ailleurs, très proches des limites théoriques optimales. En fonction des contraintes du système, ce gain pourra être utilisé pour augmenter le débit transmis et réduire la puissance d'émission. Au début, on note ici qu'il y a deux axes de recherche d'actualité sur les turbocodes : ceux concernant les entrelaceurs dans la partie codage et ceux traitant les algorithmes de décodage.

Pour les entrelaceurs, partie intégrante des codes turbo, nous avons proposé une nouvelle structure d'entrelaceurs puissants corrigeant efficacement les erreurs produites en rafale [35]. L'idée de ces entrelaceurs est basée sur l'expression de la borne sphérique d'E. Boutillon et D. Gnaedig [31]. Avec l'expression de la borne sphérique nous avons remplacé l'utilisation d'indice d'incrémentation d'entrelaceur par l'utilisation de la dimension d'indice d'incrémentation d'entrelaceur. Ce nouveau concept que nous le nommé dimension d'entrelaceur nous permet aussi de proposé une nouvelle définition de la dispersion normalisée d'une permutation aléatoire, cette dernière a permet de donner une solution au point faible d'entrelaceur DRP. On parle ici sur le choix de la taille de la fenêtre du désordre. Outre, à l'aide du paramètre de dimension d'entrelaceur nous avons proposé une nouvelle formule expérimentale qui remplace la valeur du paramètre D (larguer normalisé) non déterminé par S. Crozier dans la composante aléatoire pour l'entrelaceur DGI [16].

Suggestions pour les travaux futures

Comme suggestions pour les travaux futures on propose d'aborder les sujets suivants :

- Une étude statistique sur les erreurs produites en rafale qui varient d'une façon aléatoire en longueur devrait faire l'objet d'un thème de recherche. Cette étude permettrait de

combiner entre la propriété d'indexage aléatoire des entrelaceurs aléatoires et la propriété d'étalement maximal des entrelaceurs algébriques.

- Dans le domaine du VLSI qui est le domaine de la conception matérielle des puces électroniques, une nouvelle technologie apparaît tout les six mois. Par conséquent, il serait peut être inutile de simplifier l'algorithme de décodage MAP, surtout dans le cas où les futures technologies permettraient l'augmentation de la vitesse de transmission et l'implémentation des algorithmes de grande complexité à un prix commercialement abordable.

- Le décodage réalisé en utilisant l'algorithme MAP ou SOVA est opéré sur des blocs d'informations séparés, c'est-à-dire chaque bloc est décodé (corrigé) indépendamment des autres blocs constituant le message d'information. A travers cette remarque nous suggérons de faire un algorithme de décodage à base de dictionnaire pour décoder le message d'information, ce dictionnaire définit la relation entre les blocs d'information. Pour illustrer cette idée prenons l'exemple suivant :
  Soit le message 'matrice' à transmettre dans un canal bruité, si nous recevons ce message par exemple comme suit : 'm#trice'. Grâce au dictionnaire nous pouvons corriger le bloc erroné '#'par le bloc origine 'a'. Cet algorithme à base de dictionnaire peut être utilisé seul, comme il peut être combiné avec les algorithmes de décodage dédiés à la correction par bloc comme le MAP, SOVA et l'algorithme de Viterbi ou d'autres.

- Faire une étude plus détaillée sur les modulations numériques à plusieurs niveaux, dites aussi les modulations à constellation, dans le but d'améliorer la capacité du système de transmission en terme d'efficacité spectrale.

Articles

- Chellali S, Chouireb F. "A d-dimensional irregular compact interleaver design for turbo code". Annals of telecommunications, Vol. 70, n°5-6, May-June 2015, Springer-Verlag France.

Références

[1] Affif Hani Osseiran, "On the decoding of turbo codes" thèse de Master, Université de Montréal, Ecole polytechnique, 1999.

[2] S. B. Wicker, "Error control systems for digital communications and storage" Englewood Cliffs: Prentice Hall, 1995.

[3] Mehrotra, A, "Cellular radio performance engineering", Artech house, Inc., 1994.

[4] C. E. Shannon, "A mathematical theory of communications" Bell Syst. Tech. J., vol. 27, pp. 379–423, July 1948.

[5] Louis Wehenkel, polycopie de cours "théorie de l'information et du codage", Université de Liége, faculté des sciences appliquées, Octobre 2001. Web : http://www.montefiore.ulg.ac.be/~lwh/Info/

[6] Cédric Duchene, "Réalisation de turbo codes. Analyse de décodage itératif par EXIT CHARTS", projet de DEA, laboratoire ETIS de l'ENSEA, 2003. Web: http://www.lis.inpg.fr/pages_perso/duchene/index.html.fr

[7] D. Divsalar and F. Pollara, "Turbo codes for deep-space communications", JPL TDA Progress Report 42-120, Feb. 1995.

[8] D. Arnold and G. Meyerhans, "The Realization of the Turbo-Coding System", Swiss Federal Institute of Technology, Zurich, Switzerland, July 1995.

[9] P. Robertson, "Illuminating the Structure of Parallel Concatenated Recursive Systematic (TURBO) Codes" Proceedings, GLOBECOM '94, vol. 3, pp. 1298–1303, San Francisco, Nov. 1994.

[10] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo codes" Electron. Lett, vol. 30, no. 25, p. 2107, Dec. 8, 1994.

[11] A. Barbnlescu and S. Pietrobon, "Terminating the Trellis of Turbo-Codes in the Same State", Electron. Lett, vol. 31, no. 1, pp. 22-23.1995.

[12] O. Joerssen and H. Meyr, "Terminating the trellis of turbo-codes" Electron. Lett., vol. 30, no. 16, pp. 1285–1286, Aug. 4, 1994.

[13] Christian B. Schlegel and Lance C. Pérez, "trellis and turbo coding", Wiley-IEEE Press, 2005.

[14] O. Y. Takeshita and D. J. Costello, Jr. "New deterministic interleaver designs for turbo codes" IEEE Trans. Inform. Theory, vol. 46, no. 6, pp. 1988–2006, Sept. 2000.

[15] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations" JPL TDA Progress Report 42-122, pp. 56–65, Aug. 1995.

[16] S. Crosier, J. Lodge, P. Guinand, and A. Hunt, "Performance of turbo-codes with relative prime and golden interleaving strategies." Communication Research Centre, Station H, Ottawa, Canada, 1999. email: stewart.crozier@crc.ca, web: www.crc.ca/fec

[17] George White, "Optimised Turbo Codes for Wireless Channels" thesis of Ph. D., Communications Research Group, Department of Electronics, University of York, UK, October 2001.

[18] Todd K. Moon, "Error correction coding: Mathematical methods and algorithms" John Wiley & Sons, Inc., Publication, 2005.

[19] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications" in IEEE Globecom, pp. 1680–1686, 1989.

[20] L.R. Bahl and J. Cocke and F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimising Symbol Error Rate", IEEE transactions on information theory, vol. 20, pp. 284–287, march 1974.

[21] C. Berrou and A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes" in Proceedings of the International Conference on Communications, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[22] L. Hanzo, T.H. Liew, B.L. Yeap, "Turbo Coding, Turbo Equalisation and Space-Time Coding", Wiley-IEEE Press, July 2002.

[23] G. Forney, "The Viterbi algorithm" Proceedings of the IEEE, vol. 61, pp. 268–278, March 1973.

[24] M. Breiling, "Turbo coding simulation results" tech. Rep. University of Karlsruhe, Germany and Southampton University, UK, 1997.

[25] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference" IEEE Globecom, pp. 1679–1684, December 1990.

[26] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol dectectors with parallel structures for ISI channels" IEEE Transactions on Communications, vol. 42, pp. 1661–1671, 1994.

[27] P. Robertson and E. Villebrun and P. Höher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain" in Proceedings of the International Conference on Communications, (Seattle, United States), pp. 1009–1013, June 1995.

[28] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A low complexity soft-output viterbi decoder architecture" in Proceedings of the International Conference on Communications, pp. 737–740, May 1993.

[29] Wu, Yufei, Turbo Code Simulator, Nov 1998, MPRG lab, Virginia Tech. http://www.ee.vt.edu/~yufei

[30] C. Berrou, "Codes et turbocodes", Springer-Verlag, 2007.

[31] E. Boutillon and D. Gnaedig, "Maximum Spread of D-dimensional Multiple Turbocodes", IEEE Trans. Commun., Vol. 53, N° 8, pp 1237 - 1242, Aug. 2005.

[32] C. Heegard and S. B. Wicker, "Turbo coding", Chap. 3, Kluwer Academic Publishers, 1999.

[33] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan and M. Jézéquel, "Designing good permutations for Turbocodes: towards a single model", ICC'04, Paris, France, June 2004.

[34] S. Crozier and P. Guinand, "Distance upper bounds and true minimum distance results for turbo-codes with DRP interleavers", Proc. of 3rd Int'l Symposium on Turbocodes & Related Topics, pp. 169-172, Brest, France, Sept. 2003.

[35] Chellali S, Chouireb F. "A d-dimensional irregular compact interleaver design for turbo code". Annals of telecommunications, Vol. 70, n°5-6, May-June 2015, Springer-Verlag France.

[36] M. C. Jeruchim, P. Balaban, and K. Sam Shanmugan. "Simulation of communication systems". chapter 5, pp. 496–503, Plenum, 1992.

Annexe A. " Fonction Matlab des entrelaceurs "

```
function [temp, alpha]= entrelaceur(N,NumEntrelaceur)

% temp : vecteur du paquet permuté.
% alpha : vecteur d'indices de la permutation j=π(i).
% N : Longuer du paquet à permuter.
% NumEntrelaceur : numéro d'entrelaceur.

switch NumEntrelaceur

    case 0 % Entrelaceur aléatoire uniforme.

        [temp, alpha]=sort(rand(1,N));

    case 1 % DGI ( Dithered Golden Interleaver).

        g=((sqrt(5)-1)/2);
        m=1;j=0;r=1;s=0;
        P=N*((j+g^m)/r);
        D=0.01;
        d=unifrnd(0,D*N,1,N);
        vect=mod((P.*(0:N-1)+s+d),N);
%         for i=0:N-1
%            vect(i+1)=mod((c*i+s+d(i+1)),N);
%         end
        [temp, alpha]=sort(vect);


    case 2 % DIC ( D-dimenssion Irregular Compact Interleaver).

        n=6.186658375;
        d=2;
        r=1.385779642;
        s=0;
        P=(N^(d-1)*gamma(d+1))^(1/d);
        D=(1-sqrt(1-(1-2/N)*log(r)/log(n)))^2;
        vect=mod((P.*(0:N-1)+unifrnd(0,d*D*N,1,N)+s),N);
        [temp, alpha]=sort(vect);
```

```
    case 3 % DRP ( Dithered relative Prime Interleaver).

        % R, W: fenêtres du désordre en lecture et écriture respectivement.

        P=45;R=8;W=8;
        s=0;
        r=randperm(R)-1;
        w=randperm(W)-1;
        temp=0:N-1;
        Ia=R*floor(temp/R)+r(mod(temp,R)+1);
        Ib=mod(temp*P+s,N);
        Ic=W*floor(temp/W)+w(mod(temp,W)+1);
        for j=0:N-1;
        alpha(j+1)=Ia(Ib(Ic(j+1)+1)+1)+1;
        end

    case 4 % QPP ( Quadratic polynomial Permutation).

        % q0, q1, q2: les coefs du QPP.

        q0=0;
        q1=31;
        q2=64;
        temp=0:N-1;
        alpha=mod(q0+(q1*(0:N-1))+(q2*(0:N-1).^2),N)+1;

    case 5 % DI ( Dimentional Interleaver).

        n=6.186658375;
        d=2;
        r=1.385779642;
        s=0;
        P=(N^(d-1)*gamma(d+1))^(1/d);
        mu=(1-1/gamma(d+1))*(1/gamma(n^(1-(1-2/N)*log(r)/log(n))+1)-1/N);
        vect=mod((P.*(0:N-1)+unifrnd(0,2*mu*N,1,N)+s),N);
        [temp, alpha]=sort(vect);

Otherwise

        error('erreur: On n''a pas ce type d''entrelaceur.');
end
```